Vol. 46 No. SIG 17(TOM 13) IPSJ Transactions on Mathematical Modeling and Its Applications Dec. 2005

Regular Paper

Polynomial Time Learnability of a Sub-class of Linear Languages

Yasuhiro Tajima,† Yoshiyuki Kotani† and Matsuaki Terada†

We propose some PAC like settings for a learning problem of a sub-class of linear languages, and show its polynomial time learnability in each of our settings. Here, the sub-class of linear languages is newly defined, and it includes the class of regular languages and the class of even linear languages. We show a polynomial time learning algorithm in either of the following settings with a fixed but unknown probability distribution for examples. (1) The first case is when the learner can use randomly drawn examples, membership queries, and a set of representative samples. (2) The second case is when the learner can use randomly drawn examples, membership queries, and both of the size of a grammar which can generate the target language and d. Where d is the probability such that the rarest rule in the target grammar occurs in the derivation of a randomly drawn example. In each case, for the target language L_t, the hypothesis L_h satisfies that $Pr[P(L_h \Delta L_t) \leq \varepsilon] \geq 1 - \delta$ for the error parameter $0 < \varepsilon \leq 1$ and the confidential parameter $0 < \delta \leq 1$.

1. Introduction

In this paper, we propose some PAC [8] like settings for a learning of a sub-class of linear languages, and show its polynomial time learnability on each of our settings. The sub-class of linear languages is newly defined by us and called *Mode Selective Linear Languages* (MSLLs). It includes the class of regular languages and the class of even linear languages [7]. The class of linear grammars which generates MSLLs is called *Mode Selective Linear Grammars* (MSLGs). In grammatical inference, it has been shown that queries and additional information for the learner are powerful for polynomial time exact learning [1),2),5),7)], while there are not many results about PAC learnability or PAC like learning settings. Thus, our result contributes to understanding a PAC learning of grammars.

At first, we show an exact learning algorithm for MSLLs via membership queries and a set of representative samples. Here, a set of representative samples is a subset of the target language given to the learner apriori [1]. The time complexity of the exact learning algorithm is bounded by a polynomial of the following parameters; the time complexity to find a symmetric difference of MSLGs, n (the size of a grammar which generates the target language), and l (the maximum length of a word in a given representative samples). The main propositions of this paper is to show that, using this exact learning algorithm, the class of MSLLs is polynomial time learnable in either of the following settings.

(1) The first case is when the learner can use polynomial number of random examples, membership queries, and a set of representative samples. Here, random examples are drawn according to a distribution on Σ^* which is independent of both of the learner and the teacher.

(2) The second case is when the learner can use polynomial number of random examples, membership queries and both of n and d. Where n is the size of a grammar which generates the target language, and d is the probability such that the rarest rule in the target grammar occurs in the derivation of a randomly drawn example.

In each case, for the target language L_t, the hypothesis L_h satisfies the PAC criterion that is $Pr[P(L_h \Delta L_t) \leq \varepsilon] \geq 1 - \delta$ for the error parameter $0 < \varepsilon \leq 1$ and the confidential parameter $0 < \delta \leq 1$.

We note that both of our setting differs from standard PAC learning, and the parameter d depends on the probability distribution of examples.

In the first case, the time complexity to construct a hypothesis is bounded by a polynomial of n, l_e (the maximum length of a word in random examples or a set of representative samples), and PAC learning parameters ε and δ. In the second setting, the time complexity is bounded by a polynomial of d, n, l_e ε and δ.

In addition, for a corollary of the second case, we show that if the learner does not have

† Institute of Symbiotic Science and Technology, Tokyo University of Agriculture and Technology

to terminate with a small probability, a hypothesis can be constructed in polynomial time only from random examples and membership queries.

2. Preliminaries

A *context-free grammar* (CFG) is a 4-tuple $G = (N, \Sigma, P, S)$ where N is a finite set of *nonterminals*, Σ is a finite set of *terminals*, P is a finite set of *rewriting rules* (rules) and $S \in N$ is the *start symbol*. Let σ be the word whose length is 0. Assume that all CFGs are σ-free. In this paper, $|\beta|$ denotes the length of β if β is a string and $|W|$ denotes the cardinality of W if W is a set.

Let β, γ, $\gamma' \in N^*$ and $A \to \beta$ be in P. Then $\gamma A \gamma' \underset{G}{\Rightarrow} \gamma\beta\gamma'$ denotes the *derivation* from $\gamma A \gamma'$ to $\gamma\beta\gamma'$ in G. We define $\overset{*}{\underset{G}{\Rightarrow}}$ to be the reflexive and transitive closure of $\underset{G}{\Rightarrow}$. A *word* generated from $\gamma \in (N \cup \Sigma)^*$ by G is $w \in \Sigma^*$ such that $\gamma \overset{*}{\underset{G}{\Rightarrow}} w$ and the *language* generated from γ by G is denoted by $L_G(\gamma) = \{w \in \Sigma^* \mid \gamma \overset{*}{\underset{G}{\Rightarrow}} w\}$. A word generated from S by G for the start symbol S is called a word generated by G and the language generated by G is denoted by $L(G) = L_G(S)$. A nonterminal $A \in N$ is said to be *reachable* if $S \overset{*}{\underset{G}{\Rightarrow}} wA\beta$ for some $w \in \Sigma^*$, $\beta \in N^*$, and a nonterminal $B \in N$ is said to be *live* if $L_G(B) \neq \emptyset$. For two CFGs G_1 and G_2, $L(G_1)\Delta L(G_2)$ denotes the set $\{w \in \Sigma^* \mid w \in (L(G_1) - L(G_2)) \cup (L(G_2) - L(G_1))\}$ and a word $w \in L(G_1)\Delta L(G_2)$ is called a symmetric difference.

A CFG G is a *simple deterministic grammar* iff

- every rule in G is of the form $A \to aBC$ or $A \to aB$ or $A \to a$ for $a \in \Sigma$ and $A, B, C \in N$, and
- there is at most one rule which is of the form $A \to a\beta$ in P where $\beta \in N^*$ for every pair of $a \in \Sigma$ and $A \in N$.

The language generated by a simple deterministic grammar is called a *simple deterministic language*.

A CFG G is a *linear grammar* iff every rule in G is of the form $A \to aBb$ or $A \to aB$ or $A \to Bb$ or $A \to a$ for $a, b \in \Sigma$ and $A, B \in N$. The language generated by a linear grammar is called a *linear language*. A linear grammar $G_e = (N_e, \Sigma, P_e, S_e)$ is an *even linear grammar* iff every rule in G_e is of the form $A \to aBb$ or $A \to a$ or $A \to ab$ for $a, b \in \Sigma$ and $A, B \in N_e$. The language generated by an even linear grammar is called an *even linear language*. For any other definitions about formal language theories, the reader refers to the text book [3].

Let D be a probability distribution over Σ^* and let $Pr(w)$ be the probability for $w \in \Sigma^*$. Let L_h be a hypothesis and L_t be the target language. The learning from randomly drawn examples is called a PAC learning if a hypothesis L_h satisfies

$$Pr[P(L_h \Delta L_t) \leq \varepsilon] \geq 1 - \delta \qquad (1)$$

where $P(L_h \Delta L_t)$ is the probability of difference between L_h and L_t, i.e. the total of the probability for every $w \in L_h \Delta L_t$ on the distribution D. Even though the learner can use either some queries or additional information, we call L_h a PAC hypothesis if L_h satisfies (1). For any other definitions about PAC learning, the reader refers to the text book [4].

A membership query $MEMBER(w)$ for $w \in \Sigma^*$ on a linear language L_t is defined as follows.

$$MEMBER(w) = \begin{cases} 1 & \ldots \text{if } w \in L_t, \\ 0 & \ldots \text{if } w \notin L_t. \end{cases}$$

For a CFG G and $x \in \Sigma^*$, we can solve a membership query in $O(|x|^3)$ time.

In this paper, we assume that the learner can take a response of $MEMBER(w)$ for any $w \in \Sigma^*$ in unit time, and can take an example which is a pair of an example word $w \in \Sigma^*$ and whether $w \in L_t$ or not drawn according to D in unit time, too.

3. Mode Selective Linear Languages

We define a new sub-class of linear languages to show the learnability via membership queries and examples.

3.1 Definitions and Properties

A linear grammar $G = (N, \Sigma, P, S)$ which satisfies the following is called a *mode selective linear grammar* (MSLG):

- If a rule $A \to aBc$ is in P for $A, B \in N$ and $a, c \in \Sigma$, then
  - (1) there is no rule in P such as $A \to aCc$ for any $C(\neq B) \in N$, and
  - (2) neither $A \to aD$ nor $A \to Dc$ is in P for any $D \in N$.
- If a rule $A \to aB$ is in P for $A, B \in N$ and $a \in \Sigma$, then
  - (1) there is no rule in P such as $A \to aC$ for any $C(\neq B) \in N$, and
  - (2) neither $A \to aDb$ nor $A \to Db$ is in P for any $b \in \Sigma$ and any $D \in N$.

- If a rule $A \to Ba$ is in P for $A, B \in N$ and $a \in \Sigma$, then
  (1) there is no rule in P such as $A \to Ca$ for any $C(\neq B) \in N$, and
  (2) neither $A \to bDa$ nor $A \to bD$ is in P for any $b \in \Sigma$ and any $D \in N$.

In other words, when the derivation for $w \in L(G)$ is proceeded to $S \overset{*}{\underset{G}{\Rightarrow}} uAv$ where $u, v, z \in \Sigma^*$, $a, b \in \Sigma$ and $uazbv = w$, then there exists exactly one rule which can be applied to the first derivation of $A \overset{*}{\underset{G}{\Rightarrow}} azb$. It implies that every MSLG is unambiguous.

We define a *mode selective linear language* (MSLL) as the language generated by an MSLG. Throughout this paper, we assume that the target language is an MSLL denoted by L_t and G_t denotes some MSLG such that $L(G_t) = L_t$.

Theorem 1 The class of MSLLs contains the class of regular languages.

Proof: For a regular grammar $G_r = (N_r, \Sigma, P_r, S_r)$, all rules can be written of the form $A \to aB$ or $A \to a$ where $A, B \in N_r$ and $a \in \Sigma$. If $A \to aB$ is in P_r then $A \to aC$ is not in P_r for any $C \in N_r$ such that $C \neq B$. Thus G_r is an MSLG. On the other hand, $L = \{a^i b^i \mid i \geq 1\}$ is not a regular language but an MSLG $G = (N, \{a, b\}, P, A)$ where

$$N = \{A, B\},$$
$$P = \{A \to aB, \quad B \to aBb, \quad B \to b\}$$

is $L(G) = L$. □

Theorem 2 The class of MSLLs is incomparable to the class of simple deterministic languages.

Proof: The class of simple deterministic languages is incomparable to the class of linear languages. Thus, there exists L which is a simple deterministic language but is not an MSLL. Now, $L_1 = \{a^i b^i \mid i \geq 1\} \cup \{a^i c^i \mid i \geq 1\}$ is not a simple deterministic language but an MSLG $G_1 = (N_1, \{a, b\}, P_1, S)$ where

$$N_1 = \{S, A, B\},$$
$$P_1 = \{S \to Ab, \quad S \to Bc, \quad A \to aAb, \quad A \to a, \quad B \to aBc, \quad B \to a\}$$

satisfies that $L(G_1) = L_1$.

Thus this theorem holds. □

Theorem 3 The class of MSLLs contains the class of even linear languages.

Proof: The language $L_m = \{(ac)^i b^i \mid i \geq 1\}$ is an MSLL represented by $G_m = (N_m, \{a, b, c\}, P_m, S)$ such that

$$N_m = \{S, A, B, C\},$$
$$P_m = \{S \to Bb, \quad B \to aCb, \quad B \to Ac, \quad A \to a, \quad C \to cB\}$$

but L_m is not an even linear language. On the other hand, every even linear grammar $G_e = (N_e, \Sigma, P_e, S_e)$ can be written in an MSLG by the following algorithm.

(1) For every subset $X \subseteq N_e$, make a new nonterminal denoted by X, i.e. $N_M = N_e \cup \{X \mid X \subseteq N_e\}$.
(2) For every 3-tuple (A, a_1, a_2) where $A \in N_e$ and $a_1, a_2 \in \Sigma$, let $P_{(A,a_1,a_2)} = A \to a_1 X a_2$ where X is a new nonterminal such that $X = \{C \in N_e \mid A \to aCb \textit{ in } P_e\}$. Then, let P_0 be the union of $P_{(A,a,b)}$ for every 3-tuple (A, a, b), i.e. $P_0 = \{P_{(A,a,b)} \mid A \in N_e, a, b \in \Sigma\}$.
(3) Let $P_0' = \{A \to ab \mid A \in N_e, a, b \in \Sigma, A \to ab \textit{ in } P_e\}$ and $P_0'' = \{A \to a \mid A \in N_e, a \in \Sigma, A \to a \textit{ in } P_e\}$.
(4) For every subset $X \subseteq N_e$ and every pair of $b_1, b_2 \in \Sigma$, let $P_{(X,b_1,b_2)} = X \to b_1 Y b_2$ where $Y = \{C \in N_e \mid B \to b_1 C b_2 \textit{ in } P_e \textit{ for some } B \in X\}$. Then, let $P_1 = \{P_{(X,c,d)} \mid X \subseteq N_e, c, d \in \Sigma\}$.
(5) Let $P_1' = \{X \to ab \mid X \subseteq N_e, a, b \in \Sigma, B \to ab \textit{ in } P_e \textit{ for some } B \in X\}$ and $P_1'' = \{X \to a \mid X \subseteq N_e, a \in \Sigma, B \to a \textit{ in } P_e \textit{ for some } B \in X\}$.
(6) Let $P_M = P_0 \cup P_0' \cup P_0'' \cup P_1 \cup P_1' \cup P_1''$, then P_M has at most one rule which is of the form $A \to aBb$ for every 3-tuple (A, a, b) where $A, B \in N_M$, $a, b \in \Sigma$.

Now, we show that $L(G_e) = L(G_M = (N_M, \Sigma, P_M, S_e))$. Suppose that

$$S_e \underset{G_M}{\Rightarrow} a_1 X_1 b_1 \underset{G_M}{\Rightarrow} a_1 a_2 X_2 b_2 b_1 \underset{G_M}{\Rightarrow} \cdots \underset{G_M}{\Rightarrow} a_1 a_2 \cdots a_n w b_n b_{n-1} \cdots b_1$$

for $a_i, b_i \in \Sigma$, $X_i \in N_M$ $(i = 1, 2, \cdots n)$ and $w \in \Sigma^+$ such that $|w| \leq 2$. From the steps 4 and 5 of the above algorithm, there exist $A_i \in X_i$ $(i = 1, 2, \cdots n)$ such that $A_i \to a_{i+1} A_{i+1} b_{i+1}$ is in P_e for $i = 1, 2, \cdots n-1$ and $A_n \to w$ is in P_e. It implies that if $S_e \underset{G_M}{\Rightarrow} u$ then $S_e \underset{G_e}{\Rightarrow} u$ for any $u \in \Sigma^*$.

On the other hand, if

$$S_e \underset{G_e}{\Rightarrow} a_1 A_1 b_1 \underset{G_e}{\Rightarrow} a_1 a_2 A_2 b_2 b_1 \underset{G_e}{\Rightarrow} \cdots \underset{G_e}{\Rightarrow} a_1 a_2 \cdots a_n w b_n b_{n-1} \cdots b_1$$

for $a_i, b_i \in \Sigma$, $A_i \in N_e$ $(i = 1, 2, \cdots n)$ and $w \in \Sigma^+$ such that $|w| \leq 2$. There also exist $X_i \in N_M$ $(i = 1, 2, \cdots n)$ such that $A_i \in X_i$ $(i = 1, 2, \cdots n)$, $X_i \to a_{i+1}X_{i+1}b_{i+1}$ is in P_M for $i = 1, 2, \cdots n-1$ and $X_n \to w$ is in P_M. It implies that if $S_e \underset{G_e}{\Rightarrow} u$ then $S_e \underset{G_M}{\Rightarrow} u$ for any $u \in \Sigma^*$.

We can conclude that $L(G_e) = L(G_M = (N_M, \Sigma, P_M, S_e))$.

(7) For every pair of $A \in N_M$ and $a \in \Sigma$, let $P_2(A, a) = \{A \to a\alpha \ in \ P_M \mid \alpha \in (N_M \cup \Sigma)^+\}$. For every $P_2(A, a) \neq \emptyset$,
- add a new nonterminal $B'_{(A,a)}$ to N_M,
- let $P_{B'_{(A,a)}} = \{B'_{(A,a)} \to \alpha \mid A \to a\alpha \ in \ P_2(A, a)\}$,
- delete all rules in $P_2(A, a)$ from P_M, and
- add $A \to aB'_{(A,a)}$ and all rules in $P_{B'_{(A,a)}}$ to P_M.

Now, $G_M = (N_M, \Sigma, P_M, S)$ is an MSLG. Thus, this lemma holds. □

Lemma 4 Let $G = (N, \Sigma, P, S)$ be an MSLG and $w \in L(G)$. Consider the derivation

$$S \underset{G}{\overset{*}{\Rightarrow}} w_1 A w_2 \underset{G}{\overset{*}{\Rightarrow}} w$$

where $w_1 aub w_2 = w$ for $A \in N$, $w_1, w_2 \in \Sigma^*$, $a, b \in \Sigma$ and $u \in \Sigma^+$. Then, there is exactly one rule in P for a derivation of $A \underset{G}{\Rightarrow} \beta$ where $\beta \in (N \cup \Sigma)^+$ such that $aub \in L_G(\beta)$ and $\beta \neq A$.

Proof: From the definition of an MSLG, P includes exactly one of $A \to aB$, $A \to Bb$ or $A \to aBb$ for some $B \in N$. Thus, there exists exactly one rule in P for the derivation $A \underset{G}{\Rightarrow} \beta$. □

From this lemma, any MSLG is not ambiguous and it holds that $aub \in L(A)$ iff $w_1 aub w_2 \in L_t$. That is, we can observe behavior of a nonterminal by membership queries for L_t.

3.2 Representative Samples

We define a set of representative samples of an MSLL L to develop learning algorithms for MSLLs.

Definition 5 Let $G = (N, \Sigma, P, S)$ be an MSLG such that every $A \in N$ is reachable and live. Let Q be a finite subset of $L(G)$. Then Q is a set of *representative samples* (RS) of G iff the following holds.

- For any $A \to aBc$ in P where $A, B \in N$ and $a, c \in \Sigma$, there exists a word $w \in Q$ such that $S \underset{G}{\overset{*}{\Rightarrow}} xAy \underset{G}{\Rightarrow} xaBcy \underset{G}{\overset{*}{\Rightarrow}} w$ for some $x, y \in \Sigma^*$.

□

From this definition, for any MSLG $G = (N, \Sigma, P, S)$, there exists a set of RS Q such that $|Q| \leq |P|$.

Definition 6 For an MSLL L, a finite set $Q \subseteq L$ is a set of RS iff there exists an MSLG $G = (N, \Sigma, P, S)$ such that $L(G) = L$ and Q is a set of RS of G. □

We note that the definition of a set of RS for an MSLL is independent of representation. Assume that both G_1 and G_2 are MSLGs and $L(G_1) = L(G_2) = L_t$ holds, then a set Q is a set of RS for L_t even though Q is a set of RS for G_1 and not for G_2.

Let $G = (N, \Sigma, P, S)$ be an MSLG. For every $A \in N$, define $u_A, y_A, w_A \in \Sigma^*$ as satisfying the following.

- There exists a derivation such that $S \underset{G}{\overset{*}{\Rightarrow}} u_A A y_A \underset{G}{\overset{*}{\Rightarrow}} u_A w_A y_A$.
- It holds that $|u_A| + |y_A| \leq |u| + |y|$ for any $u, y \in \Sigma^*$ such that $S \underset{G}{\overset{*}{\Rightarrow}} uAy$.
- It holds that $|w_A| \leq |w|$ for any $w \in \Sigma^+$ such that $w \in L_G(A)$.

Then,

$$\begin{aligned} Q = \{&u_A a y_A \mid A \in N, A \to a \ is \ in \ P\} \\ &\cup \{u_A v_B w_B x_B y_A \mid v_B, x_B \in \Sigma^*, \\ &\quad A \in N, A \to v_B B x_B \ is \ in \ P\} \end{aligned}$$

is a set of RS for G.

4. The Exact Learning Algorithm

In this section, we describe the exact learning algorithm for MSLLs via membership queries and a set of RS. This algorithm is constructed by applying the learning algorithm for simple deterministic languages proposed in our previous work [6].

4.1 The Algorithm

Let Q be the given set of RS. The following R is the set of candidates for nonterminals.

$$\begin{aligned} R = \{&(x, y, z) \mid x, z \in \Sigma^*, y \in \Sigma^+, \\ &x \cdot y \cdot z \in Q\} \end{aligned}$$

and we define the *observation* $T : R \times \Sigma^* \to \{0, 1\}$ as

$$T((u, v, w), x) = MEMBER(u \cdot x \cdot w).$$

Assume that $W \subseteq \Sigma^*$ is a set of words which will be used for partitioning R. At the beginning of the learning algorithm, $W = \emptyset$ and it grows up step by step. We define an equivalence relation $\overset{W}{=}$ over R such that

$$r \overset{W}{=} r' \iff \begin{array}{c} T(r,w) = T(r',w) \\ \textit{for any } w \in W \end{array}$$

where $r, r' \in R$. In addition, we define the equivalence class $B_{\overset{W}{=}}(r) = \{r' \in R \mid r' \overset{W}{=} r\}$ for $r \in R$.

Now, a CFG $G_h = (N_h, \Sigma, P_h, S_h)$ is defined as follows where σ is the word whose length is 0.

$$N_h = \{B_{\overset{W}{=}}(r) \mid r \in R\},$$
$$S_h = B_{\overset{W}{=}}((\sigma, w, \sigma)),$$

where $w \in Q$, and

$$P_h = P_s \cup P_l \cup P_r \cup P_b.$$

Here,

$$\begin{aligned} P_s = \ & \{B_{\overset{W}{=}}((u_1, a, u_3)) \to a \mid a \in \Sigma, \\ & \qquad (u_1, a, u_3) \in R\}, \\ P_l = \ & \{B_{\overset{W}{=}}((u_1, au_2, u_3)) \to \\ & \qquad a \cdot B_{\overset{W}{=}}((u_1 a, u_2, u_3)) \mid a \in \Sigma, \\ & \quad (u_1, au_2, u_3), (u_1 a, u_2, u_3) \in R\}, \\ P_r = \ & \{B_{\overset{W}{=}}((u_1, u_2 a, u_3)) \to \\ & \qquad B_{\overset{W}{=}}((u_1, u_2, au_3)) \cdot a \mid a \in \Sigma, \\ & \quad (u_1, u_2 a, u_3), (u_1, u_2, au_3) \in R\}, \\ P_b = \ & \{B_{\overset{W}{=}}((u_1, au_2 b, u_3)) \to \\ & \qquad a \cdot B_{\overset{W}{=}}((u_1 a, u_2, bu_3)) \cdot b \mid a, b \in \\ & \quad \Sigma,\ (u_1, au_2 b, u_3), (u_1 a, u_2, bu_3) \\ & \qquad\qquad \in R\}. \end{aligned}$$

A subset of P_h is a candidate for a hypothesis rule set. From this CFG G_h, the learner deletes some rules according to following conditions.

Condition 7 For every pair of $u_1, u_2 \in \Sigma^*$ and every pair of $r_A, r_B \in R$ such that $B_{\overset{W}{=}}(r_A) \to u_1 B_{\overset{W}{=}}(r_B) u_2$ is in P_h, if there exists $w \in W$ such that $T(r_A, u_1 w u_2) \neq T(r_B, w)$ then delete $B_{\overset{W}{=}}(r_A) \to u_1 B_{\overset{W}{=}}(r_B) u_2$ from P_h. □

In other words, when $B_{\overset{W}{=}}(r_B)$ should generate w in a hypothesis but $B_{\overset{W}{=}}(r_A)$ cannot generate $u_1 w u_2$ in G_h or vice versa, the rule $B_{\overset{W}{=}}(r_A) \to u_1 B_{\overset{W}{=}}(r_B) u_2$ is deleted from candidates.

Condition 8 For every pair of $u_1, u_2 \in \Sigma^*$ and every pair of $r_A, r_B \in R$ such that $B_{\overset{W}{=}}(r_A) \to u_1 B_{\overset{W}{=}}(r_B) u_2$ is in P_h, if there exists $w \in W$ such that $T(r_B, w) = 1$ and $w \notin L_{G_h}(B_{\overset{W}{=}}(r_B))$ then delete $B_{\overset{W}{=}}(r_A) \to u_1 B_{\overset{W}{=}}(r_B) u_2$ from P_h. □

This condition means that if $B_{\overset{W}{=}}(r_B)$ should generate w in a hypothesis but there are not enough rules in G_h to generate w then all rules whose right-hand side contains $B_{\overset{W}{=}}(r_B)$ are deleted.

When the above deletions are repeated $|P_h|$ times, there is no rule which satisfies either of the above conditions. Such a set of rules P_h is called *reduced*.

The reduced P_h satisfies the following lemma. Here, let P_Σ be the set of all rules in P_h whose right-hand side is a terminal such as $A \to a$.

Lemma 9 For any set of rules P_1 such that $P_\Sigma \subseteq P_1 \subseteq P_h$ and $G_1 = (N_h, \Sigma, P_1, S_h)$ is an MSLG, it holds that

$$T(r,w) = 1 \iff B_{\overset{W}{=}}(r) \overset{*}{\underset{G_1}{\Rightarrow}} w$$

for any $r \in R$ such that $B_{\overset{W}{=}}(r) \in N_h$ is reachable and live in G_1 and any $w \in W$.

Proof: We prove this lemma by induction on the length of w.

Base step: Assume that $|w| = 1$. Then, from the definition of T and the assumption of P_1, it holds that $T(r,w) = 1 \iff B_{\overset{W}{=}}(r) \to w$ is in P_1. Thus, $T(r,w) = 1 \iff B_{\overset{W}{=}}(r) \overset{*}{\underset{G_1}{\Rightarrow}} w$ holds.

Induction step: Now suppose that this lemma holds for any $w \in \Sigma^*$ such that $|w| \leq n$. Let $u \in W$ such that $|u| = n + 1$.

Suppose that $T(r,u) = 1$. Then, from Condition 8, if $u \notin L_{G_h}(B_{\overset{W}{=}}(r))$ then all rules in P_1 whose right-hand side contains $B_{\overset{W}{=}}(r)$ are deleted. On the other hand, there exists a rule in P_1 whose right-hand side contains $B_{\overset{W}{=}}(r)$ from the assumption such that $B_{\overset{W}{=}}(r)$ is reachable and live in G_1. Thus, it holds that $u \in L_{G_h}(B_{\overset{W}{=}}(r))$. It implies that there exists a rule $B_{\overset{W}{=}}(r) \to v_1 B_{\overset{W}{=}}(r') v_2$ in P_1 such that $v_1 w' v_2 = u$ for some $B_{\overset{W}{=}}(r') \in N_h$, $v_1, v_2 \in \Sigma^*$ and $w' \in \Sigma^+$.

If $T(r', w') \neq 1$ then the rule $B_{\overset{W}{=}}(r) \to v_1 B_{\overset{W}{=}}(r') v_2$ is deleted from P_1 because of Condition 7. Thus, $T(r', w') = 1$ holds. From the assumption and $|w'| \leq n$, $B_{\overset{W}{=}}(r') \overset{*}{\underset{G_1}{\Rightarrow}} w'$ holds. Then, it also holds that $B_{\overset{W}{=}}(r) \overset{*}{\underset{G_1}{\Rightarrow}} u$.

Conversely, if $B_{\overset{W}{=}}(r) \overset{*}{\underset{G_1}{\Rightarrow}} u$, then there exists a rule $B_{\overset{W}{=}}(r) \to v_1 B_{\overset{W}{=}}(r') v_2$ in P_1 such that $v_1 w' v_2 = u$ and $B_{\overset{W}{=}}(r') \overset{*}{\underset{G_1}{\Rightarrow}} w'$ for $r' \in R$ and $v_1, v_2 \in \Sigma^*$. Thus, $T(r,u) = 1$ from $|w'| \leq n$ and Condition 7. □

From this lemma, we can find a correct grammar by selecting rules from P_h but such selections exist exponential order, unfortunately. However, we can select appropriate rules by

comparing a polynomial size set of MSLGs called *base grammars*.

We define an MSLG $G(A \to \beta) = (N_h, \Sigma, P_{A\to\beta}, S_h)$ for every rule $A \to \beta$ in P_h as follows, where $\leq_R$ is an arbitrary total order on N_h.

(1) Let $P_{A\to\beta} = \{A \to \beta\}$.

(2) For $C \in N_h$ and $a, b \in \Sigma$, if
- there exists a rule which is of the form $C \to aDb$ in P_h for some $D \in N_h$, and
- there does not exist either $C \to a\gamma$ or $C \to \gamma b$ in $P_{A\to\beta}$ for any $\gamma \in (N_h \cup \Sigma)^+$,

then let $E_0(C, a, b) = \{D \in N_h \mid C \to aDb \text{ in } P_h\}$, and add $C \to aD_0b$ to $P_{A\to\beta}$ where $D_0 \leq_R D_1$ for any $D_1 \in E_0(C, a, b)$.

(3) For $C \in N_h$ and $a \in \Sigma$, if
- there exists a rule which is of the form $C \to aD$ in P_h for some $D \in N_h$, and
- there does not exist either $C \to a\gamma$ or $C \to Eb$ in $P_{A\to\beta}$ for any $\gamma \in (N_h \cup \Sigma)^+$, any $E \in N_h$ and any $b \in \Sigma$,

then let $E_1(C, a) = \{D \in N_h \mid C \to aD \text{ in } P_h\}$, and add $C \to aD_0$ to $P_{A\to\beta}$ where $D_0 \leq_R D_1$ for any $D_1 \in E_1(C, a)$.

(4) For $C \in N_h$ and $b \in \Sigma$, if
- there exists a rule which is of the form $C \to Db$ in P_h for some $D \in N_h$, and
- there does not exist either $C \to \gamma b$ or $C \to aE$ in $P_{A\to\beta}$ for any $\gamma \in (N_h \cup \Sigma)^+$, any $E \in N_h$ and any $a \in \Sigma$,

then let $E_2(C, b) = \{D \in N_h \mid C \to Db \text{ in } P_h\}$, and add $C \to D_0b$ to $P_{A\to\beta}$ where $D_0 \leq_R D_1$ for any $D_1 \in E_2(C, b)$.

(5) For $C \in N_h$ and $a \in \Sigma$, if there exists a rule which is of the form $C \to a$ in P_h, then add $C \to a$ to $P_{A\to\beta}$.

(6) Now, $G(A \to \beta) = (N_h, \Sigma, P_{A\to\beta}, S_h)$ is an MSLG.

The set of *base grammars* $\mathbf{G}$ is defined as

$$\mathbf{G} = \{G(A \to \beta) \mid \textit{for every rule } A \to \beta \textit{ in } P_h\}.$$

Then, for every $A \in N_h$ and every pair of $G_1 \in \mathbf{G}$ and $G_2 \in \mathbf{G}$, the learner checks whether

$$L_{G_1}(A) = L_{G_2}(A)$$

or not, and finds a symmetric difference

$$w_{1,2} \in (L_{G_1}(A) \Delta L_{G_2}(A))$$

if it is not equivalent.

If all grammars in $\mathbf{G}$ generate the same language then the learner outputs any $G \in \mathbf{G}$ and terminates. Otherwise, for every i, j such that $w_{i,j}$ exists, the learner adds all sub-words of $w_{i,j}$ into W.

Example 10 For example, we construct the set of base grammars for the CFG $G_h = (N_h, \Sigma, P_h, S)$ where $N_h = \{S, A, B, C, D, E, F\}$, $\Sigma = \{a, b\}$, $P_h = \{$
$S \to aA, S \to aSb, S \to Bb,$
$A \to aC, A \to aAb, A \to Sb, A \to b,$
$B \to aS, B \to aBb, B \to Db, B \to a,$
$C \to aE, D \to Fb, E \to b, F \to a$
$\}$. Here, $L(G_h) = \{a^i b^i \mid i \geq 1\}$.

For $A \to aC$, $G(A \to aC)$ is constructed as follows. Here, we assume that $S \leq_R A \leq_R B \leq_R C \leq_R D \leq_R E \leq_R F$.

(1) Initializing $P_{A\to aC} = \{A \to aC\}$.

(2) For $S \in N_h$ and $a, b \in \Sigma$, there exists $S \to aSb$ in P_h. Then, $S \to aSb$ is added to $P_{A\to aC}$. For $A \in N_h$ and $a \in \Sigma$, there already exists $A \to aC$ in $P_{A\to aC}$. We also add $B \to aBb$ to $P_{A\to aC}$. Now, $P_{A\to aC} = \{A \to aC, S \to aSb, B \to aBb\}$.

(3) For $C \in N_h$ and $a \in \Sigma$, there exists $C \to aE$ in P_h. Then, $C \to aE$ is added to $P_{A\to aC}$.

(4) For $C \in N_h$ and $b \in \Sigma$, there exists $D \to Fb$ in P_h. Then, $D \to Fb$ is added to $P_{A\to aC}$.

(5) All of $A \to b$, $B \to a$, $E \to b$ and $F \to a$ are added to $P_{A\to aC}$.

(6) Now, $P_{A\to aC} = \{A \to aC, S \to aSb, B \to aBb, C \to aE, D \to Fb, A \to b, B \to a, E \to b, F \to a\}$. This is the rule set for the MSLG $G(A \to aC)$.

In the same manner, we construct

$$\begin{aligned}
P_{S\to aA} &= \{S \to aA, A \to aAb, B \to aBb,\\
&\quad C \to aE, D \to Fb, A \to b,\\
&\quad B \to a, E \to b, F \to a\},\\
P_{S\to aSb} &= \{S \to aSb, A \to aAb, B \to aBb,\\
&\quad C \to aE, D \to Fb, A \to b,\\
&\quad B \to a, E \to b, F \to a\},\\
P_{S\to Bb} &= \{S \to Bb, A \to aAb, B \to aBb,\\
&\quad C \to aE, D \to Fb, A \to b,\\
&\quad B \to a, E \to b, F \to a\},\\
P_{A\to Sb} &= \{S \to aSb, A \to Sb, B \to aBb,\\
&\quad C \to aE, D \to Fb, A \to b,\\
&\quad B \to a, E \to b, F \to a\},\\
P_{B\to aS} &= \{S \to aSb, A \to Sb, B \to aS,\\
&\quad C \to aE, D \to Fb, A \to b,
\end{aligned}$$

```
Algorithm 1
INPUT : Q : a set of RS;
OUTPUT: a hypothesis G_h;
begin
  R := {(x, y, z) | x, z ∈ Σ*, y ∈ Σ+,
       x · y · z ∈ Q};
  W := {y ∈ Σ+ | x, z ∈ Σ*, x · y · z ∈ Q};
  do
    find T(r, w) for ∀r ∈ R and ∀w ∈ W;
    construct G_h and make P_h reduced;
    find G;
    W' := ∅;
    for every pair of G_1, G_2 ∈ G and
    every A ∈ N_h do
      find w ∈ L_{G_1}(A)ΔL_{G_2}(A);
      W' := W' ∪ {w};
    done
    for all w ∈ W' do
      W := W ∪ {y ∈ Σ+ | x, z ∈ Σ*,
           x · y · z = w};
    done
  while (W' ≠ ∅);
  output any G ∈ G;
end.
```

Fig. 1 The exact learning algorithm.

$$P_{B\to Db} = \begin{aligned}\{&B \to a, E \to b, F \to a\},\\ \{&S \to aSb, A \to Sb, B \to Db,\\ &C \to aE, D \to Fb, A \to b,\\ &B \to a, E \to b, F \to a\}.\end{aligned}$$

We note that $P_{S\to aSb} = P_{A\to aAb} = P_{A\to b} = P_{B\to aBb} = P_{B\to b} = P_{C\to aE} = P_{D\to Fb} = P_{E\to b} = P_{F\to a}$.
Thus, $|\mathbf{G}| = 7$ and every MSLG in $\mathbf{G}$ has one of the above rule set. □

The entire description of the learning algorithm is shown in **Fig. 1**.

4.2 Termination and Time Complexity

We assume that $G_t = (N_t, \Sigma, P_t, S)$ is an MSLG such that $L(G_t) = L_t$.

Definition 11 We define that $(w_1, w_2, w_3) \in R$ corresponds to $A \in N_t$ if there exists a derivation such that $S_t \overset{*}{\underset{G_t}{\Rightarrow}} w_1 A w_3 \overset{*}{\underset{G_t}{\Rightarrow}} w_1 w_2 w_3$. In addition, for $B_{\underline{\underline{W}}}(r_0) \to u_1 B_{\underline{\underline{W}}}(r_1) u_2$ in P_h where $u_1, u_2 \in \Sigma^*$ and $r_i \in R$ $(i = 0, 1)$, we define that $B_{\underline{\underline{W}}}(r_0) \to u_1 B_{\underline{\underline{W}}}(r_1) u_2$ corresponds to $C_0 \to u_1 C_1 u_2$ in P_t if r_i corresponds to C_i for every $i = 0, 1$, respectively. □

The following lemma holds for the learning algorithm.

Lemma 12 If $L_{G_1}(A) = L_{G_2}(A)$ holds for every $A \in N_h$ and every pair of $G_1 \in \mathbf{G}$ and $G_2 \in \mathbf{G}$ then any $G \in \mathbf{G}$ satisfies that $L(G) = L_t$.

Proof: It is sufficient to show that $w \in L_G(B_{\underline{\underline{W}}}(r_A)) \iff w \in L_{G_t}(A)$ for every $A \in N_t$, $G \in \mathbf{G}$ and $w \in \Sigma^*$ where $r_A \in R$ corresponds to A. We prove the above by induction on the length of w.

Base step: Suppose that $|w| = 1$. If $A \to w$ is in P_t then there exists $B_{\underline{\underline{W}}}(r_A) \to w$ is in P_h which corresponds to $A \to w$ and $T(r_A, w) = 1$ holds from Lemma 9. Conversely, if $A \to w$ is not in P_t then it holds that $T(r_A, w) = 0$ from Lemma 9. Thus $w \in L_G(B_{\underline{\underline{W}}}(r_A)) \iff w \in L_{G_t}(A)$.

Induction step: Suppose that this lemma holds for any $w \in \Sigma^*$ such that $|w| \le n$. For every rule $A \to u_1 B u_2$ in P_t where $u_1, u_2 \in \Sigma^*$, there exists $B_{\underline{\underline{W}}}(r_A) \to u_1 B_{\underline{\underline{W}}}(r_B) u_2$ in P_h which corresponds to $A \to u_1 B u_2$. Thus, if $w \in L_{G_t}(A)$ then $w \in L_G(B_{\underline{\underline{W}}}(r_A))$ holds from the assumption. Conversely, assume that $w \notin L_{G_t}(A)$. Then, it is sufficient to consider the following cases.

(1) If it holds that $A \overset{}{\underset{G_t}{\Rightarrow}} u_1 B u_2$ for some $u_1, u_2 \in \Sigma^*$ such that $u_1 w' u_2 = w$ and $w' \notin L_{G_t}(B)$, then $w' \notin L_G(B_{\underline{\underline{W}}}(r_B))$ holds and there exists a derivation $B_{\underline{\underline{W}}}(r_A) \overset{}{\underset{G}{\Rightarrow}} u_1 B_{\underline{\underline{W}}}(r_B) u_2$. It implies that $w \notin L_G(B_{\underline{\underline{W}}}(r_A))$ because every G is not ambiguous.

(2) If there is no derivation such that $A \overset{}{\underset{G_t}{\Rightarrow}} u_1 B u_2$ for some $u_1, u_2 \in \Sigma^*$ where $u_1 w' u_2 = w$ and $w' \in L_{G_t}(B)$, then $w' \in L_G(B_{\underline{\underline{W}}}(r_B))$ holds. It implies that $T(r_B, w') = 1$ and $T(r_A, w) = 0$ from Lemma 9. Thus, from Condition 7, there is no derivation such that $B_{\underline{\underline{W}}}(r_A) \overset{}{\underset{G}{\Rightarrow}} u_1 B_{\underline{\underline{W}}}(r_B) u_2$.

In both cases, it holds that $w \notin L_G(B_{\underline{\underline{W}}}(r_A))$. □

The following lemma guarantees the termination of the exact learning algorithm.

Lemma 13 For $G_1, G_2 \in \mathbf{G}$,$w \in \Sigma^+$ and $A \in N_h$, suppose that $w \in L_{G_1}(A) \Delta L_{G_2}(A)$ and let $W_w = W \cup \{w' \in \Sigma^+ \mid u, v \in \Sigma^*, uw'v = w\}$. Now, assume that $\overset{W_w}{=}$ is the equivalence relation over R by W_w and $G'_h = (N'_h, \Sigma, P'_h, S'_h)$ is the CFG constructed from $\overset{W_w}{=}$ by the algorithm in Fig. 1. Then, at least one of the following holds.

(1) There exist $u, v \in \Sigma^*$ and $r_0, r_1 \in R$ such that

$$B_{\underline{\underline{W}}}(r_0) \to uB_{\underline{\underline{W}}}(r_1)v$$

is in P_h but

$$B_{\underline{\underline{W_w}}}(r_0) \to uB_{\underline{\underline{W_w}}}(r_1)v$$

is not in P'_h.

(2) It holds that the partition $R/\stackrel{W_w}{=}$ is finer than the partition $R/\stackrel{W}{=}$.

Proof: Assume that this lemma does not hold. Then, $(R/\stackrel{W_w}{=}) = (R/\stackrel{W}{=})$ and any rule in P_h is not deleted by Conditions 7 or 8. It implies that G_1 and G_2 are in the new set of base grammars and $B_{\underline{\underline{W}}}(r_A) = B_{\underline{\underline{W_w}}}(r_A)$ for any $r_A \in R$

Without loss of generality, we assume that $w \in L_{G_1}(B_{\underline{\underline{W}}}(r_A))$ but $w \notin L_{G_2}(B_{\underline{\underline{W}}}(r_A))$. Now, from Lemma 9, $w \in L_{G_1}(B_{\underline{\underline{W}}}(r_A))$ implies that $T(r_A, w) = 1$ in the updated T. On the other hand, $w \notin L_{G_2}(B_{\underline{\underline{W}}}(r_A))$ also implies that $T(r_A, w) = 0$ in the updated T. This is a contradiction. □

From the definition of P_h, P_h contains $3|R|^2$ rules whose right-hand side has a nonterminal. Thus, the loop of the algorithm in Fig. 1 can be repeated at most $3|R|^2$ times from Lemma 13.

Now, the following theorem holds.

Theorem 14 The class of MSLLs is exact learnable via membership queries and a set of RS. Here, the time complexity of the learning is bounded by a polynomial of

- the time complexity to check an equivalence and find a symmetric difference of MSLGs,
- the size of rules $|P_t|$,
- $\max\{|w| \mid w \in Q\}$ and $|Q|$. □

5. Constructing a PAC Hypothesis

Modifying the algorithm in Fig. 1, we can obtain polynomial time learning algorithms which outputs a PAC hypothesis. In this section, we describe the modifications.

5.1 Replacing Equivalence Checking by Random Examples

It is unknown whether checking equivalence and finding a symmetric difference of MSLGs can be solvable in polynomial time or not. Thus, it is unknown whether the time complexity of the algorithm in Fig. 1 is bounded by a polynomial of $|P_t|$, $|Q|$ and $\max\{|w| \mid w \in Q\}$. However, it has been proved that an exact learning algorithm which uses equivalence queries can be transformed to a PAC learning algorithm by replacing an equivalence query

```
Algorithm 1'
INPUT : Q:a set of RS,
    ε, δ:error and confidential parameters;
OUTPUT: a hypothesis G_h or failure;
begin
  R := {(x, y, z) | x, z ∈ Σ*, y ∈ Σ+,
        x · y · z ∈ Q};
  W := {y ∈ Σ+ | x, z ∈ Σ*, x · y · z ∈ Q};
  i := 1;
  repeat
    find T(r, w) for ∀r ∈ R and ∀w ∈ W;
    construct G_h and make P_h reduced;
    find G;
    W' := ∅;
    take n_i examples;
    if (∃G ∈ G is consistent with n_i examples)
      output G ∈ G and terminate;
    else
      let W' be the set of words in n_i examples;
    fi
    for all w ∈ W' do
      W := W ∪ {y ∈ Σ+ | x, z ∈ Σ*,
           x · y · z = w};
    done
    i := i + 1;
  until (i > 3|R|^2);
  output failure (the learning fails);
end.
```

Fig. 2 The learning algorithm 1'.

with consistency checking of polynomial number of examples [2]. That is, when i-th equivalence query is asked with a hypothesis G_h in the exact learning algorithm, then the learner takes n_i examples such that

$$n_i \geq \frac{1}{\varepsilon}\left(\log(\frac{1}{\delta}) + (\log 2)(i+1)\right)$$

and checks consistency for n_i examples. If there exists a word such that $w \in L_t \Delta L(G_h)$ in n_i examples then the w is given to the exact learning algorithm as a counterexample. If there exists no such a word in n_i examples then the hypothesis is PAC.

From our exact learning algorithm, we can obtain a polynomial time learning algorithm which outputs a PAC hypothesis by replacing equivalence checking with consistency checking for n_i examples. In **Fig. 2**, we show the learning algorithm for MSLLs via a set of RS, membership queries and random examples.

Theorem 15 The class of MSLLs is polynomial time learnable with a PAC hypothesis via polynomial number of randomly drawn examples, membership queries and a set of RS. Here, the time complexity of the learning is bounded by a polynomial of

- PAC parameters ε and δ,
- $\max\{|w| \mid w \in Q\}$ and $|Q|$, and

- the size of rules $|P_t|$.

Proof: It is sufficient that the algorithm in Fig. 2 is a polynomial time learning algorithm with a PAC hypothesis via randomly drawn examples, membership queries and a set of RS.

If there exists $G \in \mathbf{G}$ which is consistent with n_i examples in the algorithm of Fig. 2, then it holds that $Pr[P(L(G)\Delta L_t) \leq \varepsilon] \geq 1-\delta$ for any given ε and δ [2].

If any $G \in \mathbf{G}$ is not consistent with n_i examples then, for every $G \in \mathbf{G}$, there exists $w_G \in \Sigma^*$ which is a word in n_i examples such that $w_G \in L(G)\Delta L_t$. Let $W_{n_i} = \{w_G \mid G \in \mathbf{G}\}$.

Now, we consider that the algorithm in Fig. 2 continues with the set W' of words in n_i examples. Here, $W' \supseteq W_{n_i}$ holds. Assume that the new $G''_h = (N''_h, \Sigma, P''_h, S_h)$ is constructed from R, W'' and T where $W'' = W \cup W'$. Then, for $w \in W_{n_i}$ such that $w \in L_t$, there exists $w_2 \in \Sigma^+$ such that

- $w = w_1 w_2 w_3$ for some $w_1, w_3 \in \Sigma^*$,
- $T(r, w_2) = 1$ but $w \notin L_G(B_{\overset{=}{W}}(r))$ for some $G \in \mathbf{G}$.

On the other hand, for $w \in W_{n_i}$ such that $w \notin L_t$, there also exists $w_2 \in \Sigma^+$ such that

- $w = w_1 w_2 w_3$ for some $w_1, w_3 \in \Sigma^*$,
- $T(r, w_2) = 0$ but $w \in L_G(B_{\overset{=}{W}}(r))$ for some $G \in \mathbf{G}$.

Thus, if the algorithm in Fig. 2 repeats the loop with W'', then at least one of the following holds.

(1) There exist $u, v \in \Sigma^*$ and $r_0, r_1 \in R$ such that

$$B_{\overset{=}{W}}(r_0) \to u B_{\overset{=}{W}}(r_1) v$$

is in P_h but

$$B_{\overset{=}{W''}}(r_0) \to u B_{\overset{=}{W''}}(r_1) v$$

is not in P''_h.

(2) It holds that the partition $R/\overset{=}{W''}$ is finer than the partition $R/\overset{=}{W}$.

Thus, if Q is a complete set of RS then this algorithm outputs a PAC hypothesis in a finite time while repeating the "repeat - until" loop. That is, this algorithm never fails. It implies that this theorem holds. □

We note that if a complete set of RS is given to the algorithm in Fig. 2 then the algorithm outputs a PAC hypothesis before it reaches the last line. But, it may fails when an incomplete set of RS is given to the algorithm.

5.2 Selecting a Set of Representative Samples from Random Examples

We consider that how many random examples are needed to construct a set of RS. For every rule $A \to \beta$ where $\beta \in (N_t \cup \Sigma)^+$ in P_t, let

$$Z(A \to \beta) = \{w \in \Sigma^* \mid S_t \overset{*}{\underset{G_t}{\Rightarrow}} \alpha_1 A \alpha_2 \underset{G_t}{\Rightarrow} \alpha_1 \beta \alpha_2 \overset{*}{\underset{G_t}{\Rightarrow}} w, \ for\ some\ \alpha_1, \alpha_2 \in (N_t \cup \Sigma)^*\}.$$

Then, a probability $Pr(A \to \beta)$ is defined as follows;

$$Pr(A \to \beta) = \sum_{u \in Z(A \to \beta)} Pr(u).$$

It is to say that $Pr(A \to \beta)$ is an occurrence probability of $A \to \beta$ when a sample word is drawn randomly. Now, let $d = \min\{Pr(A \to \beta) \mid A \to \beta \ in \ P_t\}$, then the probability that the rule $A \to \beta$ does not appear in derivations of m examples is bounded by $(1-d)^m$. There are $|P_t|$ rules, thus a set of m examples which satisfies

$$|P_t|(1-d)^m < \delta$$

is a set of RS with a probability at least $1-\delta$. Let

$$m > \frac{1}{d}\log(\frac{|P_t|}{\delta}),$$

then it holds that

$$|P_t|(1-d)^m \leq |P_t| e^{-dm} < \delta.$$

Now, we suppose that the learner constructs a candidate for a set of RS from

$$\hat{m} > \frac{1}{d}\log\left(\frac{|P_t|}{1-\sqrt{1-\delta}}\right)$$

examples. Then, the probability that the set of $\hat{m}$ examples contains a correct set of RS is at least $\sqrt{1-\delta}$. In addition, if the equivalence checking and finding a symmetric difference in Algorithm 1' are done using

$$n_i \geq \frac{1}{\varepsilon}(\log(\frac{1}{1-\sqrt{1-\delta}}) + (\log 2)(i+1))$$

examples, then we can claim that Algorithm1' outputs a hypothesis G_h such that

$$Pr[P(L(G_h)\Delta L_t) \leq \varepsilon] \geq \sqrt{1-\delta}.$$

It implies that if the learner constructs a candidate for a set of RS from $\hat{m}$ examples and, with this set, makes a hypothesis by Algorithm1', then the hypothesis G_h satisfies that

```
Algorithm 2
INPUT : ε, δ, |P_t| and d;
OUTPUT: a hypothesis G_h;
begin
  take m̂ examples where m̂ > (1/d) log( |P_t| / (1−√(1−δ)) );
  (let M be the set of example words)
  Q := Q ∪ {w ∈ M | w ∈ L_t};
  execute Algorithm 1' with Q as a set of RS,
    ε and 1 − √(1−δ);
  if (Algorithm 1' does not fail)
    let G_h be the hypothesis of Algorithm 1';
    output G_h;
  else
    output G_h = (∅, Σ, ∅S);
  fi
end.
```

Fig. 3 The learning algorithm 2.

```
Algorithm 3
INPUT : ε and δ;
OUTPUT: a hypothesis G_h;
begin
  j := 1, Q := ∅;
  repeat
    take j examples;
    (let M be the set of example words)
    Q := Q ∪ {w ∈ M | w ∈ L_t};
    execute Algorithm 1' with Q as a set of RS,
    ε and 1 − √(1−δ);
    if (Algorithm 1' does not fail)
      output G_h;
      (the hypothesis of Algorithm 1')
    fi
    j := j + 1;
  until (forever)
end.
```

Fig. 4 The algorithm 3.

$$Pr[P(L(G_h)\Delta L_t) \leq \varepsilon] \geq 1 - \delta.$$

Thus, the algorithm in **Fig. 3** is a learning algorithm with a PAC hypothesis via examples and membership queries. With this algorithm, we can obtain the following theorem.

Theorem 16 The class of MSLLs is polynomial time learnable via random examples and membership queries with a PAC hypothesis, if the learner knows ε, δ, $|P_t|$ and d. The time complexity of the learning is bounded by a polynomial of ε, δ, $|P_t|$, d and the maximum length of examples. □

We can claim the following corollary from the algorithm in **Fig. 4**.

Corollary 17 For the class of MSLL, there exists an algorithm such that

- it constructs a hypothesis G_h such that $P(L(G_h)\Delta L_t) \leq \varepsilon$ via membership queries and random examples with a probability at least $1 - \delta$,
- the algorithm runs forever with a probability at most δ,
- when the algorithm terminates, the time complexity of the algorithm is bounded by a polynomial consists of ε, δ, $|P_t|$, d and the maximum length of examples.

Proof: It is sufficient that the algorithm in Fig. 4 terminates in this conditions. Obviously, if the algorithm terminates then the guessed hypothesis satisfies $Pr[P(L(G_h)\Delta L_t) \leq \varepsilon] \geq 1 - \delta$. When the loop of the algorithm is repeated and

$$i > \frac{1}{d} \log\left(\frac{|P_t|}{1 - \sqrt{1-\delta}}\right)$$

holds, Q is a set of RS with the probability at least $1 - \delta$. Thus, this theorem holds. □

We note that the algorithm in Fig. 4 runs forever with the probability δ. Thus, this is not a learning algorithm.

6. Conclusions

We have shown an exact learning algorithm of the class of MSLLs via membership queries and a set of RS. Modifying the algorithm, we can obtain a polynomial time learning algorithm which outputs a PAC hypothesis via either of the following settings.

- The learner can use membership queries, random examples, a set of RS, ε and δ.
- The learner can use membership queries, random examples, ε, δ, $|P_t|$ and d.

In the second setting, the learner constructs a set of RS from polynomial number of random examples where the polynomial consists of the given parameters.

PAC learnability is important not only theoretical aspects but also to make a practical algorithm, though it is hard to show standard PAC learnability of practical or useful grammars. Our results mean that additional information helps the learner not only in an exact learning but also in a PAC learning of grammars.

Acknowledgments This research report is partially supported by Grant-in-Aid for Young Scientists, No.16700007 to the first author from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

1) Angluin, D.: A note on the number of queries needed to identify regular languages, *Inf. &*

Cont., Vol.51, No.1, pp.76–87 (1981).
2) Angluin, D.: Learning regular sets from queries and counterexamples, *Inf. & Comp.*, Vol.75, No.2, pp.87–106 (1987).
3) Hopcroft, J.E. and Ullman, J.D.: *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA (1979).
4) Natarajan, B.K.: *Machine Learning: A Theoretical Approach*, Morgan Kaufmann Publishers, San Mateo, CA (1991).
5) Sakakibara, Y.: Learning context-free grammars from structural data in polynomial time, *Theor. Comp. Sci.*, Vol.76, No.2 & 3, pp.223–242 (1990).
6) Tajima, Y., Tomita, E., Wakatsuki, M. and Terada, M.: Polynomial time learning of simple deterministic languages via queries and a representative sample, *Theor. Comp. Sci.*, Vol.329, No.1–3, pp.203–221 (2004).
7) Takada, Y.: A hierarchy of language families learnable by regular language learning, *Inf. & Comp.*, Vol.123, No.2, pp.138–145 (1995).
8) Valiant, L.G.: A theory of the learnable, *Comm. ACM*, Vol.27, pp.1134–1142 (1984).

(Received August 24, 2004)
(Revised January 9, 2005)
(Accepted February 25, 2005)

Yasuhiro Tajima was born in 1971. He received his B.E., M.E. and Ph.D. from The University of Electro-Communications in 1994, 1996 and 2001, respectively. He had joined Ishikawajima-Harima Heavy Industries Co., Ltd. from 1996 to 1998. Currently, he is a research associate at Tokyo University of Agriculture and Technology. His research interests are in Machine Learning, Computational Learning Theory and Formal Language Theory. He is a member of JSAI and IEICE.

Yoshiyuki Kotani was born in 1949. He received his Ph.D. from The University of Tokyo and he was appointed Assistant Professor at Tokyo University of Agriculture and Technology in 1977. Currently, he is a Professor of Tokyo University of Agriculture and Technology. His research interests include Artificial Intelligence, Natural Language Processing, Knowledge Acquisition, Game and Puzzle, and Educational Systems. He is a member of JSAI and IEICE. He is the former president of Computer Shougi Association (CSA).

Matsuaki Terada received his B.E. from Okayama University in 1970 and D.E. from Osaka University in 1992. He joined the Systems Development Laboratory of Hitachi, Ltd in 1970, and was engaged in the research and development of distributed control processing systems, high performance protocol processing, LAN, VoIP and the next generation Internet. He has been a Professor, Department of Computer, Information & Communication Sciences, Tokyo University of Agriculture and Technology since 1999. He has also presided over the Information Media Center of the same university as a director since 2003. He is a member of IEEE, ACM and IEICE.