Polynomial Time Probabilistic Learning of a Subclass of Linear Languages with Queries

Yasuhiro Tajima and Yoshiyuki Kotani

Department of Computer and Information Sciences,
Tokyo University of Agriculture and Technology,
Naka-chou 2-24-16, Koganei, Tokyo, 184-8588, Japan
{ytajima,kotani}@cc.tuat.ac.jp

Abstract. We show a probabilistic learnability of a subclass of linear languages with queries. Learning via queries is an important problem in grammatical inference but the power of queries to probabilistic learnability is not clear yet. In probabilistic learning model, PAC (Probably Approximately Correct) criterion is an important one and many results have been shown in this model. Angluin has shown the ability of replacement from equivalence queries to random examples in PAC criterion but there are also many hardness results. We have shown that the class of simple deterministic languages is polynomial time learnable from membership queries and a representative sample. Also, we have shown that a representative sample can be constructed from polynomial number of random examples with the confidence probability. In this paper, we newly define a subclass of linear languages called strict deterministic linear languages and show the probabilistic learnability with membership queries in polynomial time. This learnability is derived from an exact learning algorithm for this subclass with membership queries, equivalence queries and a representative sample.

Keywords: learning via queries, linear language, PAC learning, representative sample.

1 Introduction

Learning via queries is an important problem in grammatical inference started from Angluin's work[2]. Many results about learning via queries have been shown for various language classes. A model which uses membership queries and equivalence queries is called MAT (Minimally Adequate Teacher) model and regular languages are polynomial time learnable from MAT[2]. This result is extend to some subclasses of linear languages[9].

Learning via queries and some additional information is studied about regular languages. In [1], a representative sample or a live-complete set is useful for polynomial time learning with membership queries. With this setting, a learnability of simple deterministic languages has been shown[8].

On the other hand, PAC (Probably Approximately Correct) learning model[10] is one of the most important probabilistic learning model. Learning of probabilistic deterministic finite automata[3] is studied but there are not many results about

A. Clark, F. Coste, and L. Miclet (Eds.): ICGI 2008, LNAI 5278, pp. 187–199, 2008.

PAC model in grammatical inference. It has been shown that equivalence query can be replaced by polynomial number of random examples in PAC criterion[2], but the power of queries to probabilistic learnability is not clear yet.

In this paper, we show a probabilistic learnability of a subclass of linear languages with membership queries. This language class called strict deterministic linear languages is newly defined and it is unambiguous and incomparable to the class of simple deterministic languages. The probabilistic learnability is derived from an exact learning algorithm for this subclass with membership queries, equivalence queries and a representative sample. Where a representative sample can be constructed from polynomial number of random example with the confidence probability. For the class of linear languages, an equivalence problem is unsolvable[5], and an equivalence problem in our newly defined languages is unknown, thus our exact learning algorithm uses powerful queries. Nevertheless, from the result of conversion from equivalence query to random examples, we can obtain a probabilistic learning algorithm of this language class.

2 Preliminaries

A *context-free grammar* (CFG for short) 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 for short) and $S \in N$ is the *start symbol*. Let ε be the word whose length is 0. If there exists no rule of the form $A \to \varepsilon$ for any $A(\neq S) \in N$, then G is called ε-free. In this paper, we assume that every CFG is ε-free.

The length of β is denoted by $|\beta|$ if β is a string and for a set W, $|W|$ denotes the cardinality of W. For $W \subset \Sigma^*$ and $u \in \Sigma^*$, we define $u\backslash W = \{v \in \Sigma^* | uv \in W\}$ and $W/u = \{v \in \Sigma^* | vu \in W\}$.

Let $A \to \beta$ be in P where $A \in N$ and $\beta \in (\Sigma \cup N)^*$. Then $\gamma A \gamma' \underset{G}{\Rightarrow} \gamma\beta\gamma'$ denotes the *derivation* from $\gamma A\gamma'$ to $\gamma\beta\gamma'$ in G where $\gamma, \gamma' \in (N \cup \Sigma)^*$. We define $\overset{*}{\underset{G}{\Rightarrow}}$ to be the reflexive and transitive closure of $\underset{G}{\Rightarrow}$. When it is not necessary to specify the grammar G, $\alpha \Rightarrow \alpha'$ and $\alpha \overset{*}{\Rightarrow} \beta$ stand for $\alpha \underset{G}{\Rightarrow} \alpha'$ and $\alpha \overset{*}{\underset{G}{\Rightarrow}} \beta$, respectively. 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 $S \overset{*}{\underset{G}{\Rightarrow}} uAw$ for some $u, w \in \Sigma^*$. and a nonterminal $D \in N$ is said to be *live* if $L_G(D) \neq \emptyset$.

A CFG $G = (N, \Sigma, P, S)$ is a *linear grammar* if every rule in P is of the form $A \to uVw$ or $A \to a$ where $A \in N$, $a \in \Sigma$, $u, w \in \Sigma^*$ and $uv \neq \varepsilon$. For every linear grammar G, there exists a grammar $G' = (N', \Sigma, P', S')$ such that $L(G) = L(G')$ and every rule in G is of the form $A \to aBc$, $A \to aB$, $A \to Ba$ or $A \to a$ where $A, B \in N$ and $a, c \in \Sigma$.

3 Our Target Language

We newly define a normal form of a linear grammar. A linear grammar $G = (N, \Sigma, P, S)$ is an **RL-linear** grammar iff the followings hold.

1. Every rule in P is of the form $A \to aB$, $A \to Ba$ or $A \to a$ for $A, B \in N$ and $a \in \Sigma$.
2. If $A \to aB$ and $A \to aC$ are in P then $B = C$ for $A, B, C \in N$ and $a \in \Sigma$.
3. If $A \to Ba$ and $A \to Ca$ are in P then $B = C$ for $A, B, C \in N$ and $a \in \Sigma$.

Theorem 1. *For any linear grammar $G = (N, \Sigma, P, S)$, there exists an* **RL-linear** *grammar $G' = (N', \Sigma, P', S')$ such that $L(G) = L(G')$.*

Proof. We suppose that every rule in P is of the form $A \to aBc$, $A \to aB$, $A \to Ba$ or $A \to a$ for $A, B \in N$ and $a, c \in \Sigma$. We can make G' from G by the following two step conversion. Let $N' = N, P' = P, S' = S$.

1. For every rule of the form $A \to aBc$ in P', replace it by $A \to aB'$ and $B' \to Bc$ where B' is a new nonterminal in N'. After this conversion, every rule in P' is of the form $A \to aB$, $A \to Ba$ or $A \to a$.
2. Delete nondeterminism in P'. Let $Body_R(A, a) = \{B \in N' | A \to aB \in P'\}$ and $Body_L(A, a) = \{B \in N' | A \to Ba \in P'\}$ for $A \in N'$ and $a \in \Sigma$.
   For every pair of $A \in N'$ and $a \in \Sigma$ such that $|Body_R(A, a)| \geq 2$ (or $|Body_L(A, a)| \geq 2$), delete all rules of the form $A \to aB$ (or $A \to Ba$) in P', then add $A \to a \cdot Z(Body_R(A, a))$ (or $A \to Z(Body_L(A, a)) \cdot a$) where $Z(Body_R(A, a))$ (or $Z(Body_L(A, a))$) is a new nonterminal.
   In addition, for every $b \in \Sigma$ and every new nonterminal $Z(Q)(Q \subseteq N')$, add

$$Z(Q) \to bZ(U), \quad \text{if } U \neq \emptyset,$$

$$Z(Q) \to Z(V)b, \quad \text{if } V \neq \emptyset,$$

   and $Z(Q) \to c$ to P' where

$$U = \{C \in N | D \in Q, D \to bC \in P\},$$

$$V = \{C \in N | D \in Q, D \to Cb \in P\},$$

$$c \in \{d \in \Sigma | D \in Q, D \to d \in P\}.$$

   If a new nonterminal $Z(Q)$ for some $Q \subseteq N'$ newly appears then repeat this step.

Then, delete all non-reachable or non-live nonterminals in N'. Now, every rule $A \to \beta \in P'$, we can derive $A \overset{*}{\underset{G'}{\Rightarrow}} \beta$. It implies that G' is equivalent to G. Moreover, G' is an **RL-linear** grammar. □

We assume that every linear grammar in this paper is in **RL-linear**. It is important for grammatical inference that the grammar is unambiguous or not. If the target language is ambiguous then membership query is not powerful because it would not check the membership about nonterminals. We define the following subclass of linear grammars which is unambiguous. Learning the subclass of linear languages generated by the grammars is our goal.

Definition 1. *An* **RL-linear** *grammar* $G = (N, \Sigma, P, S)$ *is a* strict deterministic *linear grammar if the followings holds.*

- *If* $A \to aB$ *is in* P *for* $A, B \in N$ *and* $a \in \Sigma$ *then every rule whose left-hand side is* A *is of the form* $A \to bC$ *for some* $C \in N$ *and* $b \in \Sigma$.
- *If* $A \to Ba$ *is in* P *for* $A, B \in N$ *and* $a \in \Sigma$ *then every rule whose left-hand side is* A *is of the form* $A \to Cb$ *for some* $C \in N$ *and* $b \in \Sigma$.

Thus, for every $A \in N$*, there are only right linear rules or only left linear rules whose left-hand side is* $A \in N$.

For example, the following grammar is a strict deterministic linear grammar.

$$
\begin{aligned}
&G = (N, \Sigma, P, S) \qquad (1)\\
&N = \{S, A, B, C, D, E\}\\
&\Sigma = \{a, b, c\}\\
&P = \{S \to aA,\\
&\qquad A \to Bb, \quad A \to Cc, \quad A \to b, \quad A \to c,\\
&\qquad B \to aD, \quad D \to Bb, \quad D \to b,\\
&\qquad C \to aE, \quad E \to Cc, \quad E \to c \quad \}
\end{aligned}
$$

This grammar generates $a^i b^i \cup a^i c^i$ $(i \geq 1)$ which can not be generated by regular grammars or LL(1).

We denote the language class which generated by strict deterministic linear grammars by **strict-det**. A strict deterministic linear grammar G satisfies that

$$S \underset{G}{\overset{*}{\Rightarrow}} uAv \iff u \backslash L(G) / v = L_G(A)$$

for any nonterminal A and $u, v \in \Sigma^*$. In [7], some subclasses of deterministic linear languages and learnability are studied.

Definition 2. *A deterministic linear language (***DL***) is represented by a linear grammar* $G = (N, \Sigma, P, S)$ *such that*

- *all rules are of the form* $A \to aBu$ *or* $A \to \varepsilon$*, and*
- *if both of* $A \to aBu$ *and* $A \to aCv$ *are in* P *then* $B = C$ *and* $u = v$*,*

here $A, B, C \in N$*,* $a \in \Sigma$ *and* $u, v \in \Sigma^*$.

It is shown that **DL** is identifiable in the limit from polynomial time and data[7].

Theorem 2. **DL** $\subset$ **strict-det**.

Proof. The language $a^i b^i \cup a^i c^i$ $(i \geq 1)$ which is generated by the grammar in Example (1) is **strict-det**, but it is not in **DL**[7].

Suppose that $G = (N, \Sigma, P, S)$ is a linear grammar such that $L(G)$ is in **DL**. We can make a strict deterministic linear grammar $G' = (N', \Sigma, P', S')$ from G as follows.

1. Let $N' = N, P' = P, S' = S$.
2. Suppose that $A \to \varepsilon$ is in P. Then, delete $A \to \varepsilon$ from P' and for all rules $B \to aAu$ in P where $u \in \Sigma^*$, $a \in \Sigma$ and $B \in N$, add $B \to aZ_u$ to P' where Z_u is a new nonterminal in N'. Then add
$$Z_{b_1 b_2 \cdots b_n} \to Z_{b_1 b_2 \cdots b_{n-1}} b_n \quad (n = 2, 3, \cdots |u|)$$
and $Z_{b_1} \to b_1$ to P' where $u = b_1 b_2 \cdots b_n$, $b_i \in \Sigma$ $(i = 1, \cdots, n)$.
3. Let $A, B \in N'$ and $a, b_i \in \Sigma$. Replace every rule $A \to aBb_1 b_2 \cdots b_n$ in P' with
$$A \to aZ_{Bb_1 b_2 \cdots b_n}$$
where $Z_{Bb_1 b_2 \cdots b_n}$ is a new nonterminal, then add
$$\begin{aligned} Z_{Bb_1 b_2 \cdots b_n} &\to Z_{Bb_1 b_2 \cdots b_{n-1}} b_n, \\ Z_{Bb_1 b_2 \cdots b_{n-1}} &\to Z_{Bb_1 b_2 \cdots b_{n-2}} b_{n-1}, \\ Z_{Bb_1 b_2 \cdots b_{n-2}} &\to Z_{Bb_1 b_2 \cdots b_{n-3}} b_{n-2}, \\ &\vdots \\ Z_{Bb_1} &\to Bb_1 \end{aligned}$$
to P' with these new nonterminals.
4. Delete nondeterminism in P' with the same method in Theorem1.
Now, $L(G')$ is in **strict-det** and $L(G') = L(G)$.

□

4 Queries and Probabilistic Learning

Two types of queries are important for grammatical inference since Angluin's work[2]. Let $L_t \in$**strict-det** be the target language, and $G_t(N_t, \Sigma, P_t, S_t)$ be a strict deterministic linear grammar such that $L(G_t) = L_t$. Throughout this paper, we call the class of grammars which can generate the target language target grammars. The class of grammars by whom the learner outputs a hypothesis is called hypothesis grammars.

Membership query. For $w \in \Sigma^*$ as input, "yes" is responded if $w \in L_t$ and "no" is responded otherwise. $MEMBER(w) = 1$ denotes that the membership query for $w \in \Sigma^*$ responds "yes", and $MEMBER(w) = 0$ denotes that the query responds "no."

Equivalence query. For a hypothesis grammar G_h as input, "yes" is responded if $L(G_h) = L_t$ and "no" is responded otherwise. In addition, the learner can obtain a counterexample $v \in (L_t - L(G_h)) \cup (L(G_h) - L_t)$ if the response is "no."

A learning algorithm which outputs a hypothesis G_h such that $L_t = L(G_h)$ is called an exact learning algorithm. A learning algorithm which uses membership queries and equivalence queries and which outputs a hypothesis with "yes" response of an equivalence query is called an exact learning algorithm with queries.

Membership queries are useful for checking whether a word can be generated by a nonterminal or not. The result of a membership query for uxv means whether $x \in L_{G_t}(A)$ or not for $A \in N_t$ if target grammars hold that

$$u \backslash L_t / v = L_{G_t}(A) \Leftrightarrow S_t \stackrel{*}{\Rightarrow} uAv \stackrel{*}{\Rightarrow} uwv$$

for $u, v, w \in \Sigma^*$.

We define probabilistic learning with queries as follows.

Definition 3. *Let D be a distribution on Σ^*. The probability for $w \in \Sigma^*$ is denoted by $Pr_D(w)$ and $Pr_D(T) = \sum_{w \in T} Pr_D(w)$ for a set $T \subset \Sigma^*$. A random example is a pair of a word $w \in \Sigma^*$ drawn according to D and the sign whether $w \in L_t$ or not.*

For any distribution D on Σ^*, if a learning algorithm outputs a hypothesis G_h such that

$$Pr(Pr_D((L_t - L(G_h)) \cup (L(G_h) - L_t)) \leq \varepsilon) \geq 1 - \delta$$

for given $0 < \varepsilon \leq 1$ and $0 < \delta \leq 1$, then the learning algorithm is a probabilistic learning algorithm.

In addition, a probabilistic learning algorithm which uses membership queries in it is called a probabilistic learning algorithm with queries.

We define that a language class is probabilistic learnable with queries if there exists a probabilistic learning algorithm with queries for the language class. Then, our goal is to show that **strict-det** is polynomial time probabilistic learnable with queries.

Equivalence queries in an exact learning algorithm can be replaced by polynomial number of random examples.

Theorem 3 (Angluin(1987)). *A learning algorithm which uses equivalence queries can be converted to a probabilistic learning algorithm without equivalence queries. In the converted algorithm, n_i random examples are needed instead of i-th equivalence query in the original algorithm where*

$$n_i = \frac{1}{\varepsilon}\left(\ln\frac{1}{\delta} + (\ln 2)(i+1)\right).$$

In other words, we can construct a wrapper algorithm which includes an exact learning algorithm with equivalence queries. When an equivalence query is asked by the included algorithm, the wrapper simulates the equivalence oracle with n_i random examples. Thus, the wrapper algorithm works as a probabilistic learning algorithm from outside viewpoints.

We note that this theorem shows the one-way conversion from equivalence queries to random examples. We can not declare the exact learnability from equivalence queries and membership queries if there exists a probabilistic learning algorithm with queries.

5 A Representative Sample

We define a representative sample which is a set of positive examples and it covers every rule usage. A representative sample is introduced in [1] to show the polynomial time learnability of regular sets from membership queries and it. For a strict deterministic linear grammar, we define a representative sample as follows.

Definition 4. *A word set $R \subset \Sigma^*$ is a representative sample for a strict deterministic linear grammar G_t if there exists $w \in R$ for every rule $A \to \beta$ in P_t such that*

$$S_t \overset{*}{\Rightarrow} uAv \Rightarrow u\beta v \overset{*}{\Rightarrow} w$$

where $u, v \in \Sigma^*$.

Nevertheless, we can easily construct a representative sample for a strict deterministic linear grammar, it is difficult to determine whether a set $W \subset \Sigma^*$ is a representative sample or not for a language in **strict-det**. Thus, in our learning model, the teacher would need a strict deterministic linear grammar which generates the target language to construct a representative sample of it.

A representative sample also can be replaced by polynomial number of examples.

Definition 5. *Let D be a distribution on Σ^*, and G_t be a strict deterministic linear grammar. For a rule $A \to \beta$ in P_t, we define*

$$p(A \to \beta) = \sum_{S_t \underset{G_t}{\overset{*}{\Rightarrow}} uAv \Rightarrow u\beta v \underset{G_t}{\overset{*}{\Rightarrow}} w} Pr_D(w).$$

Theorem 4 (Tajima et al.(2004)). *Let $d = \min\{p(A \to \beta) | A \to \beta \in P_t\}$. Then, m random examples contains a representative sample for G_t where*

$$m > \frac{1}{d} \ln \left(\frac{|P_t|}{\delta} \right).$$

From this theorem, we can make a probabilistic learning algorithm with queries from an exact learning algorithm which uses membership queries, and a representative sample.

Combining Theorem 3 and 4, we can construct a wrapper algorithm A_w such that

- A_w uses random examples and membership queries,
- A_w includes an exact learning algorithm which uses membership queries, equivalence queries and a representative sample, and
- A_w works as a probabilistic learning algorithm with queries.

Fig.1 is the overview of the wrapper algorithm.

Teachability[4] is one of the most important study on a special examples and learnability. When we think about the teachability on grammatical inference, identification in the limit from polynomial time and data[7] is a suitable model. The

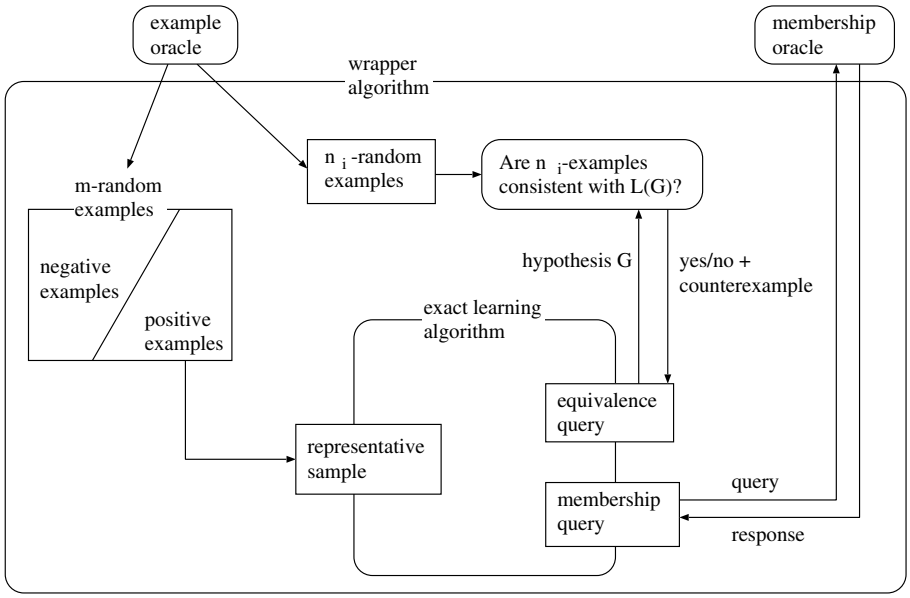

Fig. 1. The wrapper algorithm for probabilistic learning

learner can use a superset of special examples which helps the learning in these models.

In contrast, our learning model needs queries after a representative sample is given. Thus, we can not directly conclude that a language class which is probabilistic learnable with queries is also polynomial time teachable. The relation between our model and teachability is still unknown.

6 The Learning Algorithm

We show an exact learning algorithm for a strict deterministic linear language from membership queries, equivalence queries and a representative sample. It is unknown whether an equivalence problem of strict deterministic grammars is solvable or not. Nevertheless, polynomial number of random examples can substitute for both of equivalence queries and a representative sample. Thus, we can obtain a probabilistic learning algorithm with queries. This learning algorithm is a modification of our previous algorithm[8].

[Nonterminals construction.] If a representative sample R is given, there exists a 3-tuple $(u_A, v_A, w_A) \in \Sigma^* \times \Sigma^+ \times \Sigma^*$ for every $A \in N_t$ such that $u_A v_A w_A \in R$ and $S_t \overset{*}{\underset{G_t}{\Rightarrow}} u_A A w_A \overset{*}{\underset{G_t}{\Rightarrow}} u_A v_A w_A$. Thus, the following set M_h of 3-tuples is a set of candidates for nonterminals in hypothesis.

$$M_h = \{(u, v, w) \in \Sigma^* \times \Sigma^+ \times \Sigma^* | uvw \in R\}$$

We define an equivalence class on M_h with a test word set $T \subset \Sigma^*$. This is a similar process of making an observation table in [2]. At the start of the learning algorithm, $T = \Sigma$. For $(u_1, v_1, w_1) \in R$ and $(u_2, v_2, w_2) \in R$, we define the equivalence relation $\overset{T}{=}$ by

$$(u_1, v_1, w_1) \overset{T}{=} (u_2, v_2, w_2) \Leftrightarrow MEMBER(u_1 x w_1) = MEMBER(u_2 x w_2)$$

for any $x \in T$.

The equivalence class for $(u, v, w) \in M_h$ is denoted by $A((u, v, w), \overset{T}{=})$ and the classification derived by $\overset{T}{=}$ on M_h is denoted by $N_h = M_h / \overset{T}{=}$.

Obviously, for any $w_1, w_2 \in T$, it holds that

$$(\varepsilon, w_1, \varepsilon) \overset{T}{=} (\varepsilon, w_2, \varepsilon).$$

Thus, these $(\varepsilon, w_1, \varepsilon)$ and $(\varepsilon, w_2, \varepsilon)$ are in the same equivalence class.

[Rules construction] Next, candidates Q_h of production rules is made. For every $(u, avb, w) \in M_h$ and $(u, a, v) \in M_h$ here $a, b \in \Sigma, u, v, w \in \Sigma^*$, we add

$$\begin{aligned} &A((u, avb, w), \overset{T}{=}) \to aA((ua, vb, w), \overset{T}{=}) \quad \text{if } (ua, vb, w) \in M_h, \\ &A((u, avb, w), \overset{T}{=}) \to A((u, av, bw), \overset{T}{=})b \quad \text{if } (u, av, bw) \in M_h, \textit{and} \\ &\quad A((u, a, w), \overset{T}{=}) \to a \end{aligned}$$

to Q_h.

Next, we delete inappropriate rules from Q_h. Let $A \to \beta$ be in Q_h. If there exists $t \in T$ which is an evidence such that A can derive t but β can not derive it or vice versa, then $A \to \beta$ must be deleted. In other words, a rule $A \to \beta$ is inappropriate if $A \to \beta$ conflicts with results of the observation on T. The deletion procedure is as follows.

1. Delete $A((u, av, w), \overset{T}{=}) \to aA((ua, v, w), \overset{T}{=})$ from Q_h if there exist

   $$(x_1, y_1, z_1) \in A((u, av, w), \overset{T}{=}), (x_2, y_2, z_2) \in A((ua, v, w), \overset{T}{=}) \text{and } ay \in T$$

   such that

   $$MEMBER(x_1 \cdot ay \cdot z_1) \neq MEMBER(x_2 a \cdot y \cdot z_2).$$

2. Delete $A((u, va, w), \overset{T}{=}) \to A((u, v, aw), \overset{T}{=})a$ from Q_h if there exist

   $$(x_1, y_1, z_1) \in A((u, va, w), \overset{T}{=}), (x_2, y_2, z_2) \in A((u, v, aw), \overset{T}{=}) \text{and } ya \in T$$

   such that

   $$MEMBER(x_1 \cdot ya \cdot z_1) \neq MEMBER(x_2 \cdot y \cdot az_2).$$

After this deletion, if there is an inappropriate nonterminal $B \in N_h$ such that B can not derive $t \in T$ but $MEMBER(utw) = 1$ where $(u, v, w) \in B$, then we delete such nonterminals at the next step. In addition, since a hypothesis should be a strict deterministic linear grammar, we must delete all rules which are of the form $A \to aB$ for $A, B \in N_h$ and $a \in \Sigma$ if there exists $b \in \Sigma$ such that $A \to bC$ is not in Q_h but $MEMBER(ubxw) = 1$. These deletions can be written as follows.

3. Let $B \in N_h$ and $a \in \Sigma$. Suppose that $B \to aC$ is not in Q_h for any $C \in N_h$. If there exist
$$at \in T, t \in \Sigma^* \text{and } (u, v, w) \in B$$
such that
$$MEMBER(u \cdot at \cdot w) = 1$$
then delete all rules of $B \to bD$ for every $b \in \Sigma$ and every $D \in N_h$.
4. Let $B \in N_h$ and $a \in \Sigma$. Suppose that $B \to Ca$ is not in Q_h for any $C \in N_h$. If there exist
$$ta \in T, t \in \Sigma^* \text{and } (u, v, w) \in B$$
such that
$$MEMBER(u \cdot ta \cdot w) = 1$$
then delete all rules of $B \to Db$ for every $b \in \Sigma$ and every $D \in N_h$.
5. Repeat back to 3. $|R|$ times.
6. Delete all nonterminals which are not live or not reachable.

We define $S_h = A((\varepsilon, w, \varepsilon), \stackrel{T}{=})$.

With this process, every nonterminal $A((u, v, w), \stackrel{T}{=}) \in N_h$ can derive $x \in T$ such that $MEMBER(uxw) = 1$ by a linear grammar $G = (N_h, \Sigma, Q_h, S_h)$.

[Make a hypothesis] Now, $G = (N_h, \Sigma, Q_h, S_h)$ is a **RL-linear** grammar but is not strict deterministic linear grammar. The hypothesis G_h is constructed as follows.

1. Let $P_h = \{A \to a | A \in N_h, a \in \Sigma, A \to a \in Q_h\}$.
2. Assign the type of rule to every nonterminal $A \in N_h$. If there are both form of rules such that $A \to aB$ and $A \to Bb$ where $a, b \in \Sigma$ in Q_h then assign "Left" or "Right" to A randomly. Otherwise, (if A has only left or right linear rules,) assign the type to A.
3. For every $A \in N_h$ and every $a \in \Sigma$, chose a rule randomly which is of the form $A \to aB$ if A is assigned "Right" or $A \to Ba$ if A is assigned "Left", then add the rule to P_h.

Now, $G = (N_h, \Sigma, P_h, S_h)$ is a strict deterministic linear grammar and make an equivalence query for G_h. If a counterexample w is responded then update T by

$$T := T \cup \{y \in \Sigma^+ | x, z \in \Sigma^*, xyz = w\}.$$

Fig.2 is the whole learning algorithm.

INPUT: a representative sample R;
OUTPUT: correct hypothesis G_h;
begin
  $M_h := \{(u, v, w) \in \Sigma^* \times \Sigma^+ \times \Sigma^* | uvw \in R\}$;
  $T := \Sigma$;
  finish := 0;
  while (finish == 0)
  begin
    make N_h of nonterminals with the equivalence relation $\overset{T}{=}$;
    make Q_h of rules and delete inappropriate rules;
    make a hypothesis $G_h = (N_h, \Sigma, P_h, S_h)$;
    if (equivalence query for G_h responds "yes")
    then
      output G_h, and finish := 1;
    else
      let $w \in \Sigma^*$ be the counterexample;
      $T := T \cup \{y \in \Sigma^+ | x, z \in \Sigma^*, xyz = w\}$;
    endif
  end
end.

Fig. 2. The exact learning algorithm with queries and a representative sample

Now, we show the correctness of the learning algorithm in Fig.2 and its time complexity. If the algorithm terminates then the correctness of the hypothesis is clear because of the definition of equivalence query. We are concerned with the time complexity of it.

Lemma 1. *The learning algorithm in Fig.2 terminates in polynomial time of* $|N_t|$, $|\Sigma|$, $|P_t|$, $|R|$ *and* $\max\{|w||w \in R\}$ *where* R *is the given representative sample.*

Proof. Let $l = \max\{|w||w \in R\}$. Obviously, $|M_h| \leq \frac{l(l+1)}{2}|R|$ holds, thus $|N_h| \leq \frac{l(l+1)}{2}|R|$ also holds. For $(u, avb, w) \in M_h$, $u, v, w \in \Sigma^*$ and $a, b \in \Sigma$, there are at most 2 rules added to Q_h such that

$$A((u, avb, w), \overset{T}{=}) \rightarrow aA((ua, vb, w), \overset{T}{=})$$

and

$$A((u, avb, w), \overset{T}{=}) \rightarrow A((u, av, bw), \overset{T}{=})b$$

and $A((u, a, w), \overset{T}{=}) \rightarrow a$ is added to Q_h for $(u, a, w) \in M_h$. Thus, $|Q_h| < 2|M_h|$ and $|P_h| \leq |Q_h|$ hold.

Throughout the learning algorithm, M_h is not increased. It implies that $|N_h|, |Q_h|, |P_h|$ are bounded by a polynomial during the learning.

Assume that a counterexample w is responded by an equivalence query. Since T is monotone increasing, $(u_1, v_1, w_1) \overset{T}{\neq} (u_2, v_2, w_2)$ holds once, they are never

contained in the same equivalence class. It implies that $|Q_h|$ is also monotone decreasing if N_h is not changed by w.

We can claim that if a counterexample w is given, either of the following holds.

- $|Q_h|$ decreases.
- $|N_h|$ increases.

On the other hand, assume that $(u, avb, w) \in M_h$ and $(ua, vb, w) \in M_h$ correspond to a nonterminal $A \in N_t$ and $B \in N_t$ in the target grammar G_t, respectively, i.e. it holds that

$$S \overset{*}{\underset{G_t}{\Rightarrow}} uAw \underset{G_t}{\Rightarrow} uaBw \overset{*}{\underset{G_t}{\Rightarrow}} uavbw$$

for $(u, avb, w) \in M_h$ and $(ua, vb, w) \in M_h$. Then, the rule

$$A((u, avb, w), \overset{T}{=}) \rightarrow aA((ua, vb, w), \overset{T}{=})$$

is never deleted from Q_h. Thus, Q_h contains at least $|P_t|$ rules. We can conclude the polynomial time termination of the learning algorithm. □

We have the main theorem.

Theorem 5. strict-det *is polynomial time exact learnable from membership queries, equivalence queries and a representative sample.*

Proof. It is clear from Lemma1 and the learning algorithm in Fig.2. □

Theorem 6. strict-det *is polynomial time probabilistic learnable with queries.*

Proof. From Theorems3 and 4, we can replace both of equivalence queries and a representative sample by polynomial number of random examples. Then, the learning algorithm becomes a probabilistic learning algorithm with queries. Obviously, its time complexity is bounded by a polynomial of $|N_t|$, $|\Sigma|$, $|P_t|$, $|R|$ and $\max\{|w||w \in R\}$. □

7 Conclusions

We have shown that **strict-det** is probabilistic learnable in polynomial time with membership queries. This result is derived from the polynomial time exact learning algorithm of **strict-det** from equivalence queries, membership queries and a representative sample. We can show the power of a representative sample in learning via queries but there are some problems for the future. In the study of teachability, a learning algorithm with example based queries can derive the teachability by making a teaching set contains enough many words to simulate the learning algorithm[4]. On the other hand, a representative sample can not be constructed from example based queries, thus it is hard to derive some teachability from our result. Thus, study of teachability and a representative sample is one of future works.

In [6], it has been shown that characteristic samples are important in the learning from positive and negative examples. Relation between a representative sample and characteristic sample is another future work.

References

1. Angluin, D.: A note on the number of queries needed to identify regular languages. Info. & Contl. 51, 76–87 (1981)
2. Angluin, D.: Learning regular sets from queries and counterexamples. Info. & Comp. 75, 87–106 (1987)
3. Clark, A., Thollard, F.: PAC-learnability of probabilistic deterministic finite state automata. J. of Machine Learning Research 5, 473–497 (2004)
4. Goldman, S.A., Mathias, H.D.: Teaching a smarter learner. J. of Comp. & Sys. Sci. 52, 255–267 (1996)
5. Harrison, M.A.: Introduction to formal language theory. Addison-Wesley, Reading (1978)
6. de la Higuera, C.: Characteristic sets for polynomial grammatical inference. Machine Learning 27, 125–138 (1997)
7. de la Higuera, C., Oncina, J.: Inferring deterministic linear languages. In: Kivinen, J., Sloan, R.H. (eds.) COLT 2002. LNCS (LNAI), vol. 2375, pp. 185–200. Springer, Heidelberg (2002)
8. Tajima, Y., Tomita, E., Wakatsuki, M., Terada, M.: Polynomial time learning of simple deterministic languages via queries and a representative sample. Theor. Comp. Sci. 329, 203–221 (2004)
9. Takada, Y.: A hierarchy of language families learnable by regular language learning. Info. & Comp. 123, 138–145 (1995)
10. Valiant, L.G.: A theory of the learnable. Comm. of the ACM 27, 1134–1142 (1984)