Vol.2012-AL-139 No.8
2012/3/14

# 単純決定性言語のある部分言語族に対する teachability

但　馬　康　宏[†1]

単純決定性言語のある部分言語族に対する多項式サンプルでの教示可能性 (teachability) を示す．文法推論における教示可能性は，サンプルの語の長さに大きな影響を受け，多項式サンプルでの困難さがいくつか示されている．本研究では，単純決定性言語のある部分言語族に対して，質問による学習アルゴリズムを示すことにより，教示可能性を導く．このアルゴリズムは，アルファベットの種類を定数とみなせば多項式時間の学習アルゴリズムであるが，教示可能性を示す teaching set は，学習文法のサイズ，および反例の長さに関する多項式である．

# Teachability on a subclass of simple deterministic languages

Yasuhiro Tajima[†1]

We show a subclass of simple deterministic languages which is teachable in polynomial examples. Teachability on grammatical inference is studied with the setting called "identification in the limit from polynomial time and data", and there are negative results for famous languages. In this paper, we show a learning algorithm from membership queries and counterexamples for a subclass of simple deterministic languages. The time complexity of this algorithm is bounded by a polynomial if we regard the cardinality of alphabet is constant. On the other hand, we can construct a teaching set for the subclass in polynomial time without the condition of constant.

## 1. Introduction

Teaching and learning via queries are deeply related. Teaching in machine learning problems is a mathematical model which concerns not only the learner's algorithm but also the teacher's ability. Goldman has shown that representations which are example based query learnable are also teachable[7]. The polynomial time query learning algorithm of regular languages[4] is an important result, and it leads the polynomial teachability of regular languages.

In grammatical inference, it is known that teachability is different from other representations, DNF formula for example. That is because the length of examples is an important parameter for grammatical inference, thus it is hard to show the teachability or query learnability in polynomial only depend on the size of the representation of the target language. A modification of teachability for grammatical inference is studied and called "identification in the limit from polynomial time and data" by de la Higuera[8]. Neverthless, many famous subclass of context-free languages are not identifiable in the limit from polynomial time and data. Simple deteministic languages are also not polynomial identifiable because there exists a grammar such that the length of a positive example can not be bounded by a polynomial of the size of the grammar. So, we define a new teachability which contains the length of generated words in polynomial parameters, and call it "teachable in polynomial examples."

In this setting, we show that the class of stack uniform simple deterministic grammar is teachable in polynomial examples. On the other hand, this subclass of simple deterministic languages is learnable via membership queries and counterexamples, but the time complexity is not bounded by a polynomial of the size of the grammar and the length of counterexamples.

## 2. Preliminaries

A context-free grammar (CFG for short) is denoted by $G = (N, \Sigma, P, S)$ where $N$ is a finite set of nonterminals, $\Sigma$ is a finite set of teminals, $P$ is a finite set of production rules and $S \in N$ is the start symbol. We call $w \in \Sigma^*$ a word and the 0-length word is denoted by $\varepsilon$. The length of a word $w \in \Sigma^*$ is denoted by $|w|$, and the cardinality of a set $B$ is also denoted by $|B|$.

A CFG $G$ is simple deterministic (SDG for short) if $G$ is in Greibach normal form, $\varepsilon$-free and holds the following conditions.

- If $A \to a\beta \in P$ for $A \in N$, $a \in \Sigma$ and $\beta \in N^*$, then $A \to a\gamma \notin P$ for any $\gamma \in N^*$

[†1] 岡山県立大学 情報システム工学科
Department of Systems Engineering, Okayama Prefectural University

where $\gamma \neq \beta$.

We denote a derivation by $\gamma A \gamma' \underset{G}{\Rightarrow} \gamma a \beta \gamma'$ where $\gamma, \gamma' \in (N \cup \Sigma)^*$ and $A \to a\beta \in P$. The reflexive and transitive closure of derivations is denoted by $\overset{*}{\underset{G}{\Rightarrow}}$ or $\overset{*}{\Rightarrow}$ when $G$ is obvious. The simple deterministic language (SDL for short) generated by an SDG $G$ is $L(G) = \{w \in \Sigma^* \mid S \overset{*}{\underset{G}{\Rightarrow}} w\}$. In this paper, we assume that every nonterminal $A \in N$ is reachable and live, i.e. for $\alpha, \beta \in (N \cup \Sigma)^*$, there exists a derivation $S \overset{*}{\underset{G}{\Rightarrow}} \alpha A \beta$ for every $A \in N$ and there also exists a derivation $A \overset{*}{\underset{G}{\Rightarrow}} w$ for some $w \in \Sigma^*$.

Let $G = (N, \Sigma, P, S)$ be an SDG and $A \in N \cup \Sigma$. We define a set of derivationi trees

$$D_G(A) = \begin{cases} \{a\} & \ldots (A = a \in \Sigma) \\ \{A(t_1, \cdots, t_k) \mid A \to B_1 \cdots B_k \in P, & \\ t_i \in D_G(B_i), \forall i = 1, 2, \cdots, k\} \ldots (A \in N) & \end{cases}$$

We call $D_G(S)$ the set of derivation trees of $G$. A skeleton $sk(t)$ for $t \in D_G(S)$ is defined as follows.

$$sk(t) = \begin{cases} a & \ldots (t = a, a \in \Sigma) \\ \sigma(sk(t_1), \cdots, sk(t_n)) & \\ \qquad \ldots (t = A(t_1, \cdots, t_n), A \in N) & \end{cases}$$

here $\sigma$ is special symbol such that $\sigma \notin N \cup \Sigma$. In other words, $sk(t)$ is a tree whose all internal node labels are replaced by $\sigma$.

Let $L_t$ be the learning target language and a representation $G_t$ be a grammar such that $L(G_t) = L_t$. A set of word $w \in \Sigma^*$ and its membership ($w \in L_t$ or not) is called an example.

## 3. Stack uniform simple deterministic languages

**Definition1** An SDG $G = (N, \Sigma, P, S)$ is stack uniform SDG (uniSDG for short) if the followings are hold.

- If $A \to a\beta \in P$ for $A \in N$, $a \in \Sigma$, $\beta \in (N \cup \Sigma)^*$, then $|\gamma| = |\beta|$ holds for any rule $B \to a\gamma \in P$.

A stack uniform SDL (uniSDL for short) is the language generated by $G$ of a uniSDG. A pushdown automata which accepts a uniSDL moves its stack hight in "uniform" according to the input symbol. The equivalence problem on stack uniform DPDAs is solved[1)] before the equivalence problem between DPDAs are solved.

**Definition2** Let $G = (N, \Sigma, P, S)$ be a uniSDG, $a \in \Sigma$ and $A \to a\beta \in P$. We define $n_{(a,G)} = |\beta|$. If there are no rules $B \to c\gamma \in P$ for any $B \in N$ and $\gamma \in N^*$, then let $n_{(c,G)} = 1$ for $c \in \Sigma$. This is because, such $c \in \Sigma$, we can assume $B \to cZ$ is in $P$ where $Z$ is not live.

Without loss of generality, we call $\vec{n_G} = (n_{(a_1,G)}, n_{(a_2,G)}, \cdots, n_{(a_j,G)})$ the parameter vector of $G$, for $\Sigma = \{a_1, a_2, \cdots, a_j\}$. ${}^t\vec{n} = {}^t(n_1, n_2, \cdots, n_j)$ denotes the transposed vector of $vecn = (n_1, n_2, \cdots, n_j)$.

We define

$$m_{(x,a)} = |\{u \cdot a | x = u \cdot a \cdot v, \ u, v \in \Sigma^*\}|$$

for $x \in \Sigma^*$ and $a \in \Sigma$. In other words, $m_{(x,a)}$ is the number of $a$ in $x$.

**Definition3** For a finite set of words $X = \{x_1, x_2, \cdots, x_k\} \subset \Sigma^+$, we define

$$M_X = \begin{pmatrix} m_{(x_1,a_1)} & m_{(x_1,a_2)} & \cdots & m_{(x_1,a_j)} \\ m_{(x_2,a_1)} & m_{(x_2,a_2)} & \cdots & m_{(x_2,a_j)} \\ & & \vdots & \\ m_{(x_k,a_1)} & m_{(x_k,a_2)} & \cdots & m_{(x_k,a_j)} \end{pmatrix}$$

where $j = |\Sigma|$ and $k = |X|$.

For a uniSDG $G$ and $x \in L(G)$, it holds that

$$\sum_{a \in \Sigma, m_{(x,a)} \neq 0} m_{(x,a)}(n_{(a,G)} - 1) = -1 \qquad (1)$$

for $x \in \Sigma^+$ from definitions. Thus, if $x' \in \Sigma^*$ does not satisfy the equation (1) then it holds that $x' \notin L(G)$. In addition, if the inverse matrix $M_X^{-1}$ of $M_X$ for $X \subset \Sigma^+$ does not exist then there is no uniSDG $G$ such that $X \subseteq L(G)$.

It is clear that the class of uniSDGs is included in the class of SDGs because of its definition.

**Theorem4** The class of uniSDLs is proper subclass of the class of SDL.

**Proof:** Let $G = (\{S, A, B\}, \{a, b\}, P, S)$ with

$P = \{S \to aA, A \to aAB, A \to b, B \to b\}$

be an SDG, then $L(G) = \{a^i b^i \mid i \geq 1\}$. Suppose $X = \{ab, aabb\} \subset L(G)$, then $M_X$ does not have the inverse matrix. Thus $L(G)$ can not express in a uniSDG.

**Definition5** We define a regular language $L$ with an end marker as follows.

- $L$ is regular.
- $\forall w \in L$, it holds that $w = u\#$ for $u \in (\Sigma - \{\#\})^*$ and $\# \in \Sigma$.

In other words, every word in $L$ is ended by # and the end marker # must not apper in middle of any words.

**Theorem6** The class of regular languages with end marker is proper contained in the class of uniSDLs.

**Proof:** Without loss of generality, any regular language with an end marker can be represented by a uniSDG $G = (N, \Sigma, P, S)$ which has the production rules of the form $A \to aB$ or $A \to \#$ where $A, B \in N$, $a \in \Sigma$ and $\# \in \Sigma$ is the end marker.

On the other hand, let $G = (\{S, T\}, \{a, b\}, P, S)$ be a uniSDG such that

$P = \{S \to aST,\ S \to b,\ T \to b\}$

then $L(G) = \{a^i b^{i+1} \mid i \geq 0\}$ and this is not regular language. Thus, this theorem holds.

Following class of linear languages is polynomial time learnable via membership queries and counterexamples[2].

**Definition7** Let $G = (N, \Sigma, P, S)$ be a CFG and $\pi$ is a rule in $P$ such that $\pi = (A \to \beta) \in P$. We denote the derivation with $\pi$ by

$$A \underset{G}{\overset{\pi}{\Rightarrow}} \beta.$$

Let $\Pi_G = \{\pi \in P\}$ and $C \subseteq \Pi_G^*$. Then, we define

$$L(G, C) = \{w \in \Sigma^* \mid S \underset{G}{\overset{c}{\Rightarrow}} w, c \in C\}$$

**Definition8 (Takada[2])** A CFG $G = (\{S\}, \Sigma, P, S)$ with

$$\begin{aligned} P = &\{S \to aSb \mid a, b \in \Sigma\} \\ &\cup \{S \to a \mid a \in \Sigma\} \\ &\cup \{S \to \varepsilon\} \end{aligned}$$

is called a universal ELG. Let $U$ be the set of all universal ELGs for all alphabet $\Sigma$, and $\mathcal{R}$ be the class of regular languages. We define $\mathcal{L}_\rangle$ as follows.

$$\mathcal{L}_0 = \mathcal{R}$$
$$\mathcal{L}_i = \{L(G, C) \mid G \in U, C \in \mathcal{L}_{i-1}\}$$

**Theorem9** For every $i \geq 1$, $\mathcal{L}_i$ is incomparable to the class of uniSDLs.

**Proof:** Let $G_1 = (N_1, \Sigma, P_1, S) \in \mathcal{L}_1$ be

$$\begin{aligned} N_1 &= \{S, B, C\} \\ \Sigma &= \{a, b, c\} \\ P_1 &= \{S \to aBb, \quad S \to aCc \\ &\qquad B \to aBb, \quad B \to \varepsilon \\ &\qquad C \to aCc, \quad C \to \varepsilon\} \end{aligned}$$

then $L(G_1) = \{a^i b^i \cup a^i c^i \mid i \geq 1\}$. This is not an SDL[3].

On the other hand, let $G_2 = (N_2, \Sigma, P_2, S)$ be a uniSDG such that

$$\begin{aligned} N_2 &= \{S, B, C\} \\ \Sigma &= \{a, b, c\} \\ P_2 &= \{S \to aSBC, S \to b, B \to b, C \to c\} \end{aligned}$$

then $L(G_2) = \{a^i b(bc)^i \mid i \geq 0\}$. Assume that $L(G_2) \in \mathcal{L}_j\}$ for some $j > 0$, then there exist $G = (\{S\}, \{a, b, c\}, P, S) \in U$ and $C \in \mathcal{L}_{j-1}$ such that $L(G_2) = L(G, C)$. However, $G$ is a linear grammar, we can not consturct such $C$.

Thus, this theorem holds.

The class of uniSDGs has the following property.

**Lemma10** Let $G = (N, \Sigma, P, S)$ be a uniSDG. For any $w \in L(G)$, we can construct the skeleton $sk(t_w)$ from the parameter vector $\vec{n_G}$ and $\Sigma$, where $t_w$ is the derivation tree of $w$ on $G$. The time complexity to construct $sk(t_w)$ is $O(|w|)$.

**Proof:** Let $A \in N$ and $w = a_1 a_2 \cdots a_n \in \Sigma^*$ for $a_i \in \Sigma (i = 1, 2, \cdots, n)$. Reading $w$ from left to right, $sk(t_w)$ can be recursively constructed as follows.

(1) Make root node and let it the current node.

(2) Read one terminal symbol from $w$ and make children according to $\vec{n_G}$.

(3) Change the current node besed on the depth first search and back to the previous step.

Formally, we can show the algorithm in Fig. 1.

When this procedure is called recursively, the input word will be decreased. Thus,

```
Procedure make_skeleton
INPUT: $w = a_1a_2\cdots a_n \in \Sigma^*$
OUTPUT: skeleton $sk$ and $u \in \Sigma$ of $w$'s suffix
begin
    if ($|w| = 1$ then output $sk = \sigma(a_1)$ and $u = \varepsilon$, and terminate;
    for $i = 1, 2, \cdots, n_{(a_1,G)}$ do
        if ($i = 1$ then
            call make_skeleton with INPUT: $a_2a_3\cdots a_n$;
            (let OUTPUT: $sk_1, u'_1$)
        else
            call make_skeleton with INPUT: $u'_{i-1}$;
            (let OUTPUT: $sk_i, u'_i$)
        fi
    end
    Let $sk = \sigma(a_1, sk_1, sk_2, \cdots, sk_{n_{(a_1,G)}})$;
    Let $u = u'_{n_{(a_1,G)}}$;
    output $sk$ and $u$;
end.
```

**図 1** The skeleton construction algorithm

the number of resursive call is at most $|w|$, it implies that the time complexity of this procedure is $O(|w|)$.

**Example11** We show an example run of lemma 10. Assume a uniSDG $G = (\{S, T\}, \{a, b\}, \{S \to aST, S \to b, T \to b\}, S)$ and try to make the skeleton for $aabbb \in L(G)$. The parameter vector of $G$ is

$$\vec{n_G} = (n_{(a,G)}, n_{(b,G)}) = (2, 0)$$

and we start the procedure "make_skeleton(INPUT: aabbb)." Then $\sigma(a, sk_1, sk_2)$ and *abbb* are returned where $sk_1$ and $sk_2$ are constructed by recursive call of make_skeleton(). To make $sk_1$, make_skeleton(INPUT: abbb) is called then $\sigma(a, sk_3, sk_4)$ and *bbb* are returned. To make $sk_3$, make_skeleton(INPUT: bbb) is called then $\sigma(b)$ and *bb* are returned. To make $sk_4$, make_skeleton(INPUT: bb) is called then $\sigma(b)$ and *b* are returnd. Now, $sk_1 = \sigma(a, \sigma(b), \sigma(b))$. To make $sk_2$, make_skeleton(INPUT: b) is called then $\sigma(b)$ and $\varepsilon$ are returned. The final output is $\sigma(a, \sigma(a, \sigma(b), \sigma(b)), \sigma(b))$ and this is the skeleton of the derivation tree. The time complexity is also $O(|w|)$.

**Lemma12** Let $X \subset \Sigma^+$ hold that

$$\{a \in \Sigma \mid u, w \in \Sigma^*, uaw \in X\} = \Sigma.$$

In other words, for every $a \in \Sigma$, there exists $x \in X$ whose derivation uses $a$. Then it holds that

$$|\{\vec{n_G} \mid X \subseteq L(G), G \in STK_\Sigma\}| = O(l^{|\Sigma|})$$

where $l = \max\{|x| \mid x \in X\}$.

**Proof:** Every uniSDG $G$ such that $X \subseteq L(G)$ holds the equation (1) for any $x \in X$. Suppose that $j = |\Sigma|$,

$$N = \begin{pmatrix} n_{a_1} \\ n_{a_2} \\ \vdots \\ n_{a_j} \end{pmatrix}, \quad \vec{-1} = \left( \left. \begin{matrix} -1 \\ -1 \\ \vdots \\ -1 \end{matrix} \right\} j \right)$$

and $k = |X|$, then $|\{\vec{n_G} \mid X \subseteq L(G), G\text{is uniSDG}\}|$ is bounded by the number of solutions of

$$M_X(N + \vec{-1}) = \vec{-1} \qquad (2)$$

with

$$n_{a_i} \geq 0$$

for $i = 1, 2, \cdots, j$.

Now, if there exists $i$ such that $n_{a_i} \geq l$ then some $h$ $(1 \leq h \leq k)$ hold that $m_{(x_h,a_i)} > 0$. In addition, it holds that

$$\sum_{i'=1}^{j} m_{(x_h,a_{i'})}(n_{a_{i'}}-1)$$
$$= m_{(x_h,a_i)}(n_{a_i}-1)$$
$$+ \sum_{i'=1\ (i'\neq i)}^{j} m_{(x_h,a_{i'})}(n_{a_{i'}}-1)$$
$$\geq (l-1)-(l-1) > -1$$

thus $n_{a_i} < l$ holds for any $i$ $(1 \leq i \leq j)$ if the equation (2) holds. It implies that the number of solutions of (2) is bounded by $l^{|\Sigma|}$.

## 4. Learning via membership queries and counterexamples

We show a plynomial time learning algorithm for uniSDGs via membership queries and counterexamples. In this algorithm, the following queries are used. Let $L_t$ be the target uniSDL.

**[Membership query ]**

INPUT: $w \in \Sigma^*$

OUTPUT: yes if $w \in L_t$,

no if $w \notin L_t$

**[Equivalence query ]**

INPUT: a hypothesis uniSDG $G_h$

OUTPUT: yes if $L(G_h) = L_t$,

no and $w \in \Sigma$ if $L(G_h) \neq L_t$

where $w \in \Sigma$ is a counterexample such that $w \in (L_t - L(G_h)) \cup (L(G_h) - L_t)$.

In the following of this paper, let $G_t = (N_t, \Sigma, P_t, S_t)$ be a uniSDG such that $L(G_t) = L_t$. We define the following query.

**[Equivalence query with a structural counterexample ]**

INPUT: a hypothesis uniSDG $G_h$

OUTPUT: yes if $L(G_h) = L_t$,

no and $sk(t_w)$ if $L(G_h) \neq L_t$

where $w \in (L_t - L(G_h)) \cup (L(G_h) - L_t)$, and if $w \in L_t$ then $sk(t_w)$ is the skeleton of the derivation tree of $w$ on $G_t$, else if $w \in L(G_h)$ then $sk(t_w)$ is that on $G_h$.

We have shown the following.

**Theorem13** SDLs are polynomial time learnable via membership queries and equivalence queries with a structural counterexample[5].

The learning algorithm in[5] makes membership queries and equivalence queries with a structural counterexample, then outputs a hypothesis SDG $G_h$ in polynomial of size of $G_t$ and counterexamples. We call this algorithm $LA'$. The learning algorithm $LA$ which is proposed in this paper uses $LA'$ as follows.

(1) $LA$ makes an equivalence query for the empty hypothesis $G_h = (\{S\}, \Sigma, \{\}, S)$, and obtains a positive counterexample.

(2) $LA$ finds all parameter vectors which are consistent with all positive examples obtained by this step. We denote the parameter vectors $\vec{n_{G_1}}, \vec{n_{G_2}}, \cdots, \vec{n_{G_k}}$. From lemma 12, $k$ is $O(l^{|\Sigma|})$. Here, $l$ is the length of the longest counterexample.

(3) For every $n_{G_j}$ $(j = 1, 2, \cdots, k)$, $LA$ runs $LA'$ in parallel. We denote these algorihtms $LA'_1, LA'_2, \cdots, LA'_k$. For some $j$, if $LA'_j$ makes an equivalence query with a structural counterexample whose input is the hypothesis $G_h$, then $LA$ makes an equivalence query with $G_h$ and obtains a counterexample $w \in \Sigma^*$. Then, $LA$ makes $sk(t_w)$ from $\vec{n_{G_j}}$ and return the skeleton to $LA'_j$. Such a skeleton can be constructed in polynomial time from lemma 10.

Let $f_{LA'}$ be the time complexity of the algorithm $LA'$. The time complexity of our learning algorithm $LA$ is $O(l^{|\Sigma|} \cdot f_{LA'})$ because one of $\{n_{G_j} \mid j = 1, 2, \cdots, k\}$ is the correct parameter for $G_t$.

Let $j$ be the indicator of the correct parameter vector. We show that the hypothesis of $LA'_j$ can be rewrite into uniSDG. When $LA'_j$ terminates and outputs a hypothesis SDG $G_j = (N_j, \Sigma, P_j, S_j)$ then $P_j$ can be separate into 2 parts[5]:

$$P_j = P_0 \cup P_1$$

here $P_0$ is a rule set of uniSDG whose parameter vector is $\vec{n_{G_j}}$ and all rules in $P_1$ are of the form $A \rightarrow w$ for $A \in N_j$ and $w \in \Sigma^*$. Now, every $A \rightarrow w \in P_1$ holds

$$M_{\{w\}}({}^t\vec{n_G} + \vec{-1}) = \vec{-1}$$

because $B \overset{*}{\underset{G_t}{\Rightarrow}} w$ for some $B \in N_t$, then we can construct a uniSDG $G_w$ such that $\vec{n_{G_w}} = \vec{n_{G_j}}$ and $\{w\} = L(G_w)$. Thus, the rule set $P_j$ is in uniSDG and its parameter is

$n_{\vec{G_j}}$.

The time complexity of $LA$ is evaluated by the following theorem.

**Theorem14** A uniSDL $L(G_t)$ $(G_t = (N_t, \Sigma, P_t, S_t))$ can be learnable with uniSDGs via membership queries and counterexamples. If we consider $|\Sigma|$ is a constant, the time complexity is bounded by a polynomial of $|N_t|$ and $l$, here $l$ is the length of the longest counterexample.

**Proof:** Let $X_+$ be the positive examples and $X_-$ be the negative examples such that $LA$ obtained. From lemma 12, the number of solutions of $M_{X_+}(N + \vec{-1}) = \vec{-1}$ is $O(l^{|\Sigma|})$. From theorem 13, $LA'$ terminates in polynomial time of $l$ and the size of $G_t$. We denote it $f_{LA'}$. Thus, both $|X_+|$ and $|X_-|$ is $O(l^{|\Sigma|} \cdot f_{LA'})$.
The time complexity to transform the hypothesis into uniSDG is also $O(l \cdot f_{LA'})$ and construct the skeleton for $LA'$ is $O(l)$.

**Example15** This is an example run of $LA$. Let the target language be $L_t = \{a^i b^i c \mid i > 0\}$ and the grammar be $G_t = (\{S, A, B, C\}, \Sigma = \{a, b, c\}, P_t, S)$ where $P_t = \{S \to aAC, A \to aAB, A \to b, B \to b, C \to c\}$. The first hypothesis of $LA$ is $G_h = (\{S_h\}, \Sigma, \{\}, S_h)$, and let the first counterexample is *abc*, i.e. $X_+ = \{abc\}$. Solutions of equations $M_{X_+}(N + \vec{-1}) = \vec{-1}$ are $N_1 = (n_a, n_b, n_c) = (2, 0, 0)$ and $N_2 = (n_a, n_b, n_c) = (1, 1, 0)$. Now, two algorithms, $LA'_{N_1}$ whose parameter vector is $N_1$ and $LA'_{N_2}$ whose parameter vector is $N_2$, are run in parallel.
$LA$ executes $LA'_{N_1}$ and $LA'_{N_2}$ alternately. Either $LA'_{N_1}$ or $LA'_{N_2}$ makes an equivalence query with structural counterexample, then $LA$ makes an equivalence query. If positive counterexample has been returned, $LA$ adding the skelton according to the parameter vector of $LA'_{N_1}$ or $LA'_{N_2}$.
When either $LA'_{N_1}$ or $LA'_{N_2}$ terminates, $LA$ terminates with the hypothesis. In this example, $LA'_{N_1}$ terminates in polynomial time. The time complexity of $LA$ is $2f_{LA'_{N_1}}$ where $f_{LA'_{N_1}}$ is that of $LA'_{N_1}$.

## 5. Teachability

Teachability is one of a variation of machine learning problems. There are some different settings[6)7)] but we concern Goldman and Mathias's T/L-teachablity[7)]. In this setting, the teacher makes a teaching set $T$ which is a set of examples for the learner at first. Then, the adversary adds a set of examples $A$ to $T$ arbitrarily. The learner takes $A \cup T$ and try to identify the target language. If the teacher can make $T$ in polynomial time of the size of a representation $G_t$ for the target language $L_t$ and the learner can identify in polynomial time of the size of $G_t$ and the size of $A \cup T$, then such a representation class is polynomially T/L-teachable. If the teacher's complexity is not bounded then we call such a representation class is semi-poly T/L-teachable.

It is known that learnability via queries leads teachability.

**Theorem16 (Goldman and Mathias)** A representation class which is polynomial time learnable via example based queries is semi-poly T/L-teachable.
Here, both of membership query and equivalence query are example based query.

Neverthless, on grammatical inference, there is difficulty in polynomial teachability. Let $T$ be a finite set of examples of the target language. A consistency-easy class is a representation class which can express any $T$ and we can find such a representation in polynomial time of the size and the total length of $T$. It is trivial that uniSDG, SDG, CFG are consistency-easy class.

**Definition17 (de la Higuera[8)])** A representation class $R$ is identifiable in the limit from polynomial time and data iff there exist two polynomials $p()$ and $q()$ and an algorithm $A$ such that :

(1) Given any examples $S$ of size $m$, $A$ returns a representation $r \in R$ compatible with $S$ in $O(p(m))$ time.

(2) For each representation $r$ of size $n$, there exists a characteristic example $CS$ of size less than $q(n)$ for which , if $CS \subseteq S$, $A$ returns a representation $r'$ equivalent with $r$.

In this setting, some positive identifiability has been shown[9)]. Then, following theorems hold.

**Theorem18 ([8)])** A consistency-easy class is identifiable in the limit from polynomial time and data iff it is semi-poly T/L-teachable.

**Theorem19 ([8)])** The class of SDGs is not identifiable in the limit from polynomial time and data.

**Proof:** Suppose $G_t = (\{A_i \mid i = 1, 2, \cdots, n\}, \{a, b\}, P, A_1)$ where

$$\begin{aligned} P = \{ \quad & A_i \to aA_{i+1}A_{i+1} \ (i = 1, 2, \cdots, n-1), \\ & A_n \to b \ \} \end{aligned}$$

then $L(G_t)$ contains just one word and the length is $2^n - 1$. If the teacher makes a teaching set $T$, $T$ must contain the positive example, but the size of $T$ is not bounded by a polynomial of $|N_t| = n$.
This $G_t$ is also uniSDG. Thus, the class of uniSDG is also not identifiable in the limit from polynomial time and data.

On the other hand, the length of a counterexample is an important parameter for polynomial time query learning. We think, for teachiability, that the length of words which generated by $G_t$ is corresponds to that of counterexamples.

**Definition20** Let $G_t = (N_t, \Sigma, P_t, S_t)$ be a uniSDG. The thickness of $A \in N_t$ is the length of the shortest word which is generated from $A$, and is denoted by $tck(A)$. The thickness of $A \to \beta \in P_t$ is the length of the shortest word which is generated from $\beta$, and is denoted by $tck(A \to \beta)$. The thickness of $G_t$ is $\max(\{tck(A) \mid A \in N_t\} \cup \{tck(A \to \beta) \mid A \to \beta \in P_t\})$, and denoted by $tck(G_t)$.
If a class of CFGs is T/L-teachable in polynomial of the size of $G_t$ and $tck(G_t)$, then we call the class of CFGs *teachable in polynomial examples*.

We can claim immediately that the class of uniSDG is teachable in polynomial examples if $|\Sigma|$ is considered a constant.

In addition, the following theorem removes $l^{|\Sigma|}$ factor from the polynomial.

**Theorem21** The class of uniSDGs is teachable in polynomial examples.

**Proof:** The number of solutions of the equation (2) has been decreased if positive examples are increased. At most $|\Sigma|$ positive examples minimize the number of solutions. In addition, all such solutions can become the parameter vector of $G_t$. Thus, teacher can make a teaching set $T$ whose size is polynomial of the size of $G_t$ and $tck(G_t)$.
The learner can also find one of the parameter vector of $G_t$ in polynomial time of the size of given examples. Thus, the time complexity to identify $G_t$ is $O(l \cdot |\Sigma| \cdot f_{LA'})$.

## 6. Conclusions

We have shown that:

- The class of uniSDLs is learnable via membership queries and counterexamples. But the time complexity is $O(l^{|\Sigma|} \cdot f_{LA'})$, here $l$ is the maximum length of counterexamples and $f_{LA'}$ is the time complexity of the learning algorithm for SDLs via membership queries and structural counterexamples.
- The class of uniSDLs is teachable in polynomial examples. The size of the teaching set $T$ can be bounded by $O(l \cdot |\Sigma| \cdot f_{LA'})$, here $l$ is the thickness of $G_t$.

## 参 考 文 献

1) M. Linna, "Two decidability results for deterministic pushdown automata", Journal of Computer and System Sciences, vol.18, no.1, pp.92–107, Feb. 1979.
2) Y. Takada, "A hierarchy of languages families learnable by regular language learning", Information and Computation, vol.123, no.2, pp.138–145, 1995.
3) M. A. Harrison, "Introduction to Formal Language Theory", Addison-Wesley, Reading MA, 1978.
4) D. Angluin, "Learning regular sets from queries and counterexamples", Information and Computation, vol.75, no.2, pp.87-106, 1987.
5) Y. Tajima, E. Tomita and M. Wakatsuki, "Polynomial time MAT learning of simple deterministic languages with structural counterexamples", The IEICE Transactions on Information and Systems (in Japanese), vol.J82-D-I, no.4, pp.521–532, 1999.
6) A. Shinohara and S. Miyano, "Teachability in computational learning", New Generation Computing, vol.8, pp.337–347, 1991.
7) S. A. Goldman and H. D. Mathias, "Teaching a smart learner", Journal of Computer and System Sciences, vol.52, pp.255–267, 1996.
8) C. de la Higuera, "Characteristic sets for polynomial time grammatical inference", Machine Learning, vol.27, no.2, pp.125–138, 1997.
9) C. de la Higuera and J. Oncina, "On sufficient conditions to identify in the limit classes of grammars from polynomial time and data", LNAI 2484 (Proc. of ICGI 2002), pp.134–148, 2002.