PAPER

Teachability of a Subclass of Simple Deterministic Languages

Yasuhiro TAJIMA[†a)], ***Member***

SUMMARY We show teachability of a subclass of simple deterministic languages. The subclass we define is called stack uniform simple deterministic languages. Teachability is derived by showing the query learning algorithm for this language class. Our learning algorithm uses membership, equivalence and superset queries. Then, it terminates in polynomial time. It is already known that simple deterministic languages are polynomial time query learnable by context-free grammars. In contrast, our algorithm guesses a hypothesis by a stack uniform simple deterministic grammar, thus our result is strict teachability of the subclass of simple deterministic languages. In addition, we discuss parameters of the polynomial for teachability. The "thickness" is an important parameter for parsing and it should be one of parameters to evaluate the time complexity.

key words: *teachability, learning via query, polynomial time learning, simple deterministic language*

1. Introduction

Teachability is one of machine learning problems. In this problem, the learner finds out a correct hypothesis via a special set of examples. There are some different models of teachability [2], [4], [9]. In Goldman and Mathias's setting [2], the teacher makes a set of examples called teaching set which is helpful for the learner. Then, adversary adds some examples to the teaching set with the aim of confusing the learner. If the learner can guess a correct hypothesis from the given examples, the language class is called teachable. Especially, if the teacher can make a teaching set in polynomial time and the learner can also guess a correct hypothesis in polynomial time, then such language is called polynomially T/L-teachable. If only the learner can guess a correct hypothesis in polynomial time, then such language is called semi-poly T/L-teachable. In this setting, it is known that a language class is semi-poly T/L-teachable if there exists a polynomial time learning algorithm which uses example based queries.

"Identification in the limit from polynomial time and data" is defined by de la Higuera [4]. This setting adds one more consistency condition to Goldman and Mathias's teachability, i.e. if the learner can construct a consistent hypothesis for any examples then de la Higuera's setting and Goldman and Mathias's setting are equivalent, that is semi-poly T/L-teachable. The polynomial in de la Higuera's setting only consists of the size of the target representation and the size of teaching set. We think it is too strong because any grammar class which contains $G_t = (\{A_i \mid i = 1, 2, \cdots, n\}, \{a, b\}, P, A_1)$ where

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

can not be identifiable in the limit from polynomial time and data. Thus, we use the length of the shortest examples for every nonterminal as a parameter of the polynomial.

In this paper, we show a polynomial time query learning algorithm for a subclass of simple deterministic languages. This algorithm uses membership, equivalence and superset queries. Then, it guesses a correct hypothesis in time of polynomial of the size of the grammar which generates the target language and the longest length of counterexamples. It implies that this language class is semi-poly T/L-teachable on our parameters of the polynomial.

It is shown by de la Higuera [4] that if a grammar class whose equivalence problem is undecidable then such class is not identifiable in the limit from polynomial time and data. It implies that such class of languages is not query learnable in polynomial time. For both context-free languages and linear languages, the equivalence problems are undecidable [3]. Thus, these languages are not identifiable in the limit from polynomial time and data. It implies that these languages can not be query learnable in polynomial time. For simple deterministic languages, the equivalence problem is decidable [12]. Ishizaka [7] has shown the polynomial time learning algorithm via membership and equivalence queries. Thus, simple deterministic languages are semi-poly T/L-teachable. But, the representation in Ishizaka's algorithm is a context-free grammar. It is unsolvable that the equivalence problem between a context-free grammar and a simple deterministic grammar [11]. Then, the result of Ishizaka's algorithm can not be compared to a simple deterministic grammar. We should say that simple deterministic languages are semi-poly T/L-teachable *with a context-free grammar*.

In contrast, the representation of our learning algorithm is the restricted simple deterministic grammar which can express all languages in the target class. This brings some advantages. At first, an equivalence check between the learning result and any other simple deterministic grammar can be solvable in polynomial time. The second advantage is that we can parse any word in linear time, since the hypothesis of Ishizaka's algorithm needs CYK like algorithm whose complexity is at least $O(l^2)$ for the l length word. Dupont

Manuscript received March 6, 2013.
Manuscript revised July 19, 2013.

[†]The author is with the Department of Systems Engineering, Okayama Prefectural University, Soja-shi, 719–1197 Japan.
a) E-mail: tajima@cse.oka-pu.ac.jp

DOI: 10.1587/transinf.E96.D.2733

et al. [6] have applied a grammatical inference algorithm to investigate software behavior. When we apply some learning algorithm to such a practical problem, the representation of the learner is important. The equivalence check is useful to check compatibility between two results of learning or teaching. Fast parsing is also useful to apply the result to some practical systems. At last, we show that the time complexity of our learning algorithm is not slower than Ishizaka's algorithm.

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, Σ is a finite set of terminals, 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 ε. 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 a simple deterministic grammar (SDG for short) if G is in Greibach normal form and ε-free and G satisfies the following conditions.

- If $A \rightarrow a\beta \in P$ where $A \in N$, $a \in \Sigma$ and $\beta \in N^*$, then $A \rightarrow a\gamma \notin P$ for any $\gamma \in N^*$ such that $\gamma \neq \beta$.

We denote a derivation by $\gamma A \gamma' \underset{G}{\Rightarrow} \gamma a\beta\gamma'$ where $\gamma, \gamma' \in (N \cup \Sigma)^*$ and $A \rightarrow a\beta \in P$. 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 $L(G) = \{w \in \Sigma^* \mid S \overset{*}{\underset{G}{\Rightarrow}} w\}$. For $A \in N$, we define $L_G(A) = \{w \in \Sigma^* | A \overset{*}{\underset{G}{\Rightarrow}} w\}$. If there exists a derivation $S \overset{*}{\underset{G}{\Rightarrow}} \alpha A\beta$ for $A \in N$ where $\alpha, \beta \in (N \cup \Sigma)^*$, then A is reachable. In addition, if there exists a derivation $A \overset{*}{\underset{G}{\Rightarrow}} w$ for some $w \in \Sigma^*$ then A is live.

Let $G = (N, \Sigma, P, S)$ be an SDG. A rooted ordered tree t_w is a derivation tree for $w \in L(G)$ if the followings hold.

1. The root of t_w is labeled by S.
2. Let $A \in N$ and $B_1, B_2, \cdots, B_n \in N \cup \Sigma$. If an internal node labeled by A has child nodes labeled by $B_1, B_2, \cdots, B_n$, respectively, then $A \rightarrow B_1 B_2 \cdots B_n$ is in P.
3. The yield of t_w is w.

A skeleton $sk(t)$ for a derivation tree t is a tree whose all internal nodes are labeled by a special symbol σ such that $\sigma \notin N \cup \Sigma$.

Let v be a node of a derivation tree or a skeleton t, and the yield of t be w. The yield of the subtree rooted at v is denoted by $word(v)$. We define

$$pre(v) = \begin{cases} \varepsilon \ldots \text{if } v \text{ is the root} \\ pre(u) \cdot word(a_1) \cdots word(a_i) \ldots \text{otherwise} \end{cases}$$

here we assume that u is the parent of v and u has child nodes $a_1, \cdots, a_i, v, b_1, \cdots b_j$, respectively. In other words, $pre(v)$ is the prefix word of w which is the yield before the depth-first search of t reaches at v. In addition, $post(v)$ is the suffix word of w such that $pre(v) \cdot word(v) \cdot post(v) = w$.

We define corresponding nodes for two trees t_1, t_2 as follows.

- The root of t_1 and the root of t_2 are corresponding nodes.
- If a node v_1 of t_1 and a node v_2 of t_2 are corresponding nodes, and both of v_1 and v_2 have the same number of children, then x_i and y_i are corresponding nodes where x_i is the i-th child node of v_1 and y_i is the i-th child node of v_2.

Let V_1 be the set of nodes of t_1 and V_2 be the set of nodes of t_2. If $v_1 \in V_1$ and $v_2 \in V_2$ are corresponding nodes then $f(v_1) = v_2$. Two trees t_1 and t_2 are isomorphic iff the mapping $f : V_1 \rightarrow V_2$ is a one-to-one mapping.

We denote a tree t by the followings.

- If t has only one node whose label is $a \in \Sigma$, then $t = a$.
- Let A be the label of root and v_i be the i-th child node of the root ($i = 1, \cdots, m$). Then, $t = A(t_1, t_2, \cdots, t_m)$ where t_i is the subtree rooted at v_i for $i = 1, \cdots, m$.

Let L_t be the learning target language. Let $G_t = (N_t, \Sigma, P_t, S_t)$ be a grammar such that $L(G_t) = L_t$ and every $A \in N_t$ be reachable and live. 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

Definition 1: An SDG $G = (N, \Sigma, P, S)$ is a stack uniform SDG (suSDG for short) if the following holds.

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

A stack uniform SDL (suSDL for short) is the language generated by an suSDG G.

A pushdown automaton which accepts an suSDL moves its stack height in "uniform" according to the input symbol. The equivalence problem on stack uniform deterministic pushdown automata (DPDAs) is solved [8] before the equivalence problem between DPDAs is solved. Let $M = (\Sigma, \Gamma, Q, F, \Delta, c_s)$ be a DPDA, where Σ, Γ, and Q are finite sets of input symbols, stack symbols, and states, respectively, $F \subseteq Q \times (\{\Omega\} \cup \Gamma)$ is the set of accepting modes (here Ω is a special empty stack symbol), Δ is a finite set of transition rules and $c_s = (s_0, Z_0) \in Q \times \Gamma$ is the initial configuration. A DPDA M is stack uniform iff

- M has no ε-rules, and
- for any $s_1, s_2, s_1', s_2' \in Q$, $A, A' \in \Gamma$, $a \in \Sigma$ and $w, w' \in \Gamma^*$, if $(s_1, A) \rightarrow^a (s_2, w)$ and $(s_1', A') \rightarrow^a (s_2', w')$ are in Δ, then it holds that $|w| = |w'|$.

An SDL can be represented by a DPDA such that $|Q| = 1$. Then, it is trivial that an suSDG is equivalent to a stack uniform DPDA with $|Q| = 1$.

Definition 2: Let $G = (N, \Sigma, P, S)$ be an suSDG, $a \in \Sigma$ and $A \to a\beta \in P$. We define $n_{(a,G)} = |\beta|$. If the grammar G is obvious, n_a also denotes $n_{(a,G)}$. If there are no rules in P of the form $B \to c\gamma$ for any $B \in N$ and $\gamma \in N^*$, then let $n_{(c,G)} = 1$ for $c \in \Sigma$. This is because we can assume $B \to cZ$ is in P for such $c \in \Sigma$ where Z is not live.

Without loss of generality, we assume an order on Σ and let $\Sigma = \{a_1, a_2, \cdots, a_j\}$. We call $\vec{n_G} = \begin{pmatrix} n_{(a_1,G)} \\ n_{(a_2,G)} \\ \vdots \\ n_{(a_j,G)} \end{pmatrix}$ the parameter vector of G.

We define

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

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

Definition 3: 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 $k = |X|$.

Lemma 4: For an suSDG $G = (N, \Sigma, P, S)$ 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 $x \in \Sigma^+$.

Proof: Suppose $A \to a\beta$ is in P and there is a derivation $\gamma_1 A \gamma_2 \underset{G}{\Rightarrow} \gamma_1 a \gamma_3$. Then, $|A| + |\gamma_2| + n_{(a,G)} = |a| + |\gamma_3|$ holds. Adding this equation for $S \overset{*}{\underset{G}{\Rightarrow}} x$, it holds that

$$|S| + \sum_{a \in \Sigma, m_{(x,a)} \neq 0} m_{(x,a)} n_{(a,G)} = |x|$$

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

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

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

Thus Eq. (1) holds. □

From this lemma, if $x' \in \Sigma^*$ does not satisfy Eq. (1) then it holds that $x' \notin L(G)$.

It is clear that the class of suSDGs is included in the class of SDGs because of Definition 1.

Theorem 5: The class of suSDLs is a proper subclass of SDLs.

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 $X = \{x_1 = ab, x_2 = aabb\} \subset L(G)$, then $m_{(x_1,a)} = 1$, $m_{(x_1,b)} = 1$, $m_{(x_2,a)} = 2$, $m_{(x_2,b)} = 2$. There is no solution $\vec{n_G} = (n_a, n_b)$ which satisfies

$$m_{(x_1,a)}(n_a - 1) + m_{(x_1,b)}(n_b - 1) = -1$$

and

$$m_{(x_2,a)}(n_a - 1) + m_{(x_2,b)}(n_b - 1) = -1.$$

Thus, $L(G)$ is not an suSDG. □

Definition 6: 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 appear in middle of any words.

Theorem 7: The class of regular languages with an end marker is proper contained in the class of suSDLs.

Proof: Without loss of generality, any regular language with an end marker can be represented by an suSDG $G = (N, \Sigma, P, S)$ 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 an suSDG 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 a regular language. Thus, this theorem holds. □

The class of suSDGs has the following property.

Lemma 8: Let $G = (N, \Sigma, P, S)$ be an suSDG. For any $w \in L(G)$, we can construct the skeleton $sk(t_w)$ from the parameter vector $\vec{n_G}$ and Σ, 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 child nodes according to $\vec{n_G}$.
3. Change the current node based on the depth first search and back to the previous step.

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

The output skeleton sk satisfies followings.

- The yield of sk is w.
- The first child of every internal node is labeled by $a_i \in \Sigma$ $(i = 1, \cdots, n)$.

```
Procedure make_skeleton
INPUT: $w = a_1 a_2 \cdots a_n \in \Sigma^+$, parameter vector $\vec{n_G}$
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 exit
    if ($n_{(a_1,G)} = 0$) then output $sk = \sigma(a_1)$ and $u = a_2 \cdots a_n$, and exit
    for $i = 1, 2, \cdots, n_{(a_1,G)}$ do
    begin
        if ($i == 1$) then
            call make_skeleton with INPUT: $a_2 a_3 \cdots a_n, \vec{n_G}$
            (let OUTPUT: $sk_1, u'_1$)
        else
            call make_skeleton with INPUT: $u'_{i-1}, \vec{n_G}$
            (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.
```

Fig. 1 The skeleton construction algorithm.

- The root of sk and the root of t_w are corresponding nodes.
- If a node v of t_w and a node u of sk are corresponding nodes, then both v and u have $n_{(a,G)} + 1$ child nodes where $a \in \Sigma$ is the label of the first child of u.

Thus, sk is isomorphic to t_w.

When this procedure is called recursively, the input word will be decreased. Thus, the number of recursive call is at most $|w|$, it implies that the time complexity of this procedure is $O(|w|)$. □

Example 9: We show an example run of Fig. 1. Assume an suSDG $G = (\{S, T\}, \{a, b\}, \{S \to aST, S \to b, T \to b\}, S)$ and try to make the skeleton for $w = aabbb \in L(G)$. The parameter vector of G is

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

and we start the procedure "make_skeleton(INPUT: aabbb, $\vec{n_G}$)." Since $n_{(a,G)} = 2$, it makes two times recursive calls of make_skeleton(). The first call is "make_skeleton(INPUT: abbb, $\vec{n_G}$)." This second level procedure also makes two times recursive calls, and the first call is "make_skeleton(INPUT: bbb, $\vec{n_G}$)." The third level procedure returns $\sigma(b)$ and bb. The second level procedure calls "make_skeleton(INPUT: bb, $\vec{n_G}$)" and obtains $\sigma(b)$ and b. Now, the second level procedure returns $\sigma(a, \sigma(b), \sigma(b))$ and b. The first level procedure calls "make_skeleton(INPUT: b, $\vec{n_G}$)" and obtains $\sigma(b)$ and ε. Finally, the first level procedure returns $\sigma(a, \sigma(a, \sigma(b), \sigma(b)), \sigma(b))$ and ε. This is the skeleton of the derivation tree. The time complexity is $O(|w|)$.

Lemma 10: Let $X \subset \Sigma^+$ hold that

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

In other words, every $a \in \Sigma$ appears in some $x \in X$. Then it holds that

$$|\{\vec{n_G} \mid X \subseteq L(G), G \text{ is an suSDG}\}| = O(l^{|\Sigma|})$$

where $l = \max\{|x| \mid x \in X\}$, i.e. the number of possible parameter vectors is finite and $O(l^{|\Sigma|})$.

Proof: Every suSDG G such that $X \subseteq L(G)$ holds Eq. (1) for any $x \in X$. Thus, $|\{\vec{n_G} \mid X \subseteq L(G), G \text{ is an suSDG}\}|$ is bounded by the number of solutions of

$$M_X(\vec{n_G} + \vec{-1}) = \vec{-1} \tag{2}$$

with

$$n_{a_i} \geq 0$$

for $i = 1, 2, \cdots, |\Sigma|$. Here,

$$\vec{n_G} = \begin{pmatrix} n_{a_1} \\ n_{a_2} \\ \vdots \\ n_{a_{|\Sigma|}} \end{pmatrix}, \quad \vec{-1} = \left.\begin{pmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{pmatrix}\right\} |\Sigma|.$$

Let $k = |X|$. For any a_i, there exists $1 \leq h_i \leq k$ such that $m_{(x_{h_i}, a_i)} > 0$. In addition, it holds that $0 \leq n_{a_i} < l$ for any i $(1 \leq i \leq |\Sigma|)$ because of the following. If there exists i such that $n_{a_i} \geq l$ then

$$\begin{aligned} &\sum_{i'=1}^{|\Sigma|} m_{(x_{h_i}, a_{i'})}(n_{a_{i'}} - 1) \\ &= m_{(x_{h_i}, a_i)}(n_{a_i} - 1) + \sum_{i'=1\ (i' \neq i)}^{|\Sigma|} m_{(x_{h_i}, a_{i'})}(n_{a_{i'}} - 1) \\ &\geq 1(l-1) + (l-1) \cdot \min_{i' \neq i}(n_{a_{i'}} - 1) \\ &\geq (l-1) + (l-1)(-1) = 0 > -1. \end{aligned}$$

This is a contradiction to Eq. (1).

Thus, the number of solutions of (2) is bounded by $l^{|\Sigma|}$. □

For an suSDG G and its parameter vector $\vec{n_G}$, we define the following suSDG G_u.

$$G_u = (\{S\}, \Sigma, P_u, S)$$
$$P_u = \{S \to a_i \underbrace{S \cdots S}_{n_{(a_i,G)}} \mid a_i \in \Sigma\}$$

We call this a Universal suSDG (USDG for short). Then, the followings hold.

- Both G and G_u have the same parameter vector, i.e. $\vec{n_G} = \vec{n_{G_u}}$.
- $L(G) \subseteq L(G_u)$.

Similar definition and universal grammars are used in the learning algorithm for linear languages [10].

4. Learning Algorithm When the Parameter Vector is Given

We define the following queries. Here, L_t is the target suSDL.

[Membership query: *Member*]

INPUT: $w \in \Sigma^*$
OUTPUT: *yes* if $w \in L_t$,
no if $w \notin L_t$

[Equivalence query: *Equiv*]

INPUT: a hypothesis suSDG G_h
OUTPUT: *yes* if $L(G_h) = L_t$,
no and $w \in \Sigma^*$ if $L(G_h) \neq L_t$

here $w \in \Sigma^*$ 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 an suSDG such that $L(G_t) = L_t$. With these queries and a parameter vector $\vec{n_{G_t}}$, we can show a polynomial time learning algorithm. This learning algorithm is an application of Angluin [1]'s algorithm to suSDGs.

From Lemma 8, we can make a skeleton *sk* for $w \in L_t$ if $\vec{n_{G_t}}$ is known. In the following of this paper, we denotes the skeleton for $w \in L_t$ by $sk(w)$. Let V be the set of all nodes of $sk(w)$. Since every SDG is unambiguous and $sk(w)$ is isomorphic to the derivation tree t_w, we can use a word $pre(v)$ for $v \in V$ as a representation of a nonterminal. In addition, let $x \in \Sigma^+$ be the shortest suffix of $post(v)$ such that

- $x'x = post(v)$ for some $x' \in \Sigma^*$,
- $pre(v) \cdot word(v) \cdot x \in L_t$.

Such shortest suffix x is denoted by $sspost(v)$. We can find $sspost(v)$ for any node v by $|post(v)|$ times calling of membership queries. Let v be a node of $sk(w)$, u be the node of t_w such that v and u are corresponding nodes. Suppose that $S_t \overset{*}{\Rightarrow} pre(u) \cdot A \cdot \beta \overset{*}{\Rightarrow} pre(u) \cdot word(u) \cdot \beta \overset{*}{\Rightarrow} pre(u) \cdot word(u) \cdot post(u)$, then it also holds that $S_t \overset{*}{\Rightarrow} pre(u) \cdot A \cdot \beta \overset{*}{\Rightarrow} pre(u) \cdot word(u) \cdot \beta \overset{*}{\Rightarrow} pre(u) \cdot word(u) \cdot sspost(u)$. The following lemma holds.

Lemma 11 (Ishizaka [7] Lemma 8): Let v and u are corresponding nodes where v is a node of t_y and u is a node of $sk(y)$ for $y \in \Sigma^*$. In addition, let $A \in N_t$ be the label of v. It holds that

$$\begin{aligned} L_{G_t}(A) = \{w \in \Sigma^+ \\ &\mid Member(pre(u) \cdot w \cdot sspost(u)) = yes, \\ &\forall w' \in \Sigma^+, \forall w'' \in \Sigma^+, w'w'' = w, \\ &Member(pre(u) \cdot w' \cdot sspost(u)) = no\}. \end{aligned}$$

Proof: Since v and u are corresponding nodes, it holds that $pre(u) = pre(v)$ and $sspost(u) = sspost(v)$. From t_y is a derivation tree, it holds that

$$\begin{aligned} L_{G_t}(A) = \{w \in \Sigma^+ \\ &\mid Member(pre(u) \cdot w \cdot sspost(u)) = yes, \\ &\forall w' \in \Sigma^+, \forall w'' \in \Sigma^+, w'w'' = w, \\ &Member(pre(u) \cdot w' \cdot sspost(u)) = no\}. \end{aligned}$$

□

Let V be the set of internal nodes of $sk(w)$. In our algorithm, $\{(pre(v), sspost(v)) | v \in V\}$ is a set of nonterminal candidates of $sk(w)$. These candidates are added to N_0 for all positive counterexamples.

Let $W \subset \Sigma^+$. For a set of nonterminal candidates N_0 and $(u, v) \in N_0$, we define $L_{(u,v)} = \{w \in W | Member(u \cdot w \cdot v) = yes, \forall w' \in \Sigma^+, \forall w'' \in \Sigma^+, w'w'' = w, Member(u \cdot w' \cdot v) = no\}$. In addition, for $(u_1, v_1), (u_2, v_2) \in N_0$, we define $(u_1, v_1) \overset{W}{=} (u_2, v_2)$ iff $L_{(u_1,v_1)} = L_{(u_2,v_2)}$. An equivalence class on N_0 by $\overset{W}{=}$ is a nonterminal of a hypothesis grammar and N_h denotes the set of nonterminals. We denote the equivalence class $A \in N_h$ which contains $(p, q) \in N_0$ by $A(p, q)$. The root of $sk(w)$ corresponds to $(\varepsilon, \varepsilon) \in N_0$. The root of $sk(w')$ also corresponds to $(\varepsilon, \varepsilon) \in N_0$ for any $w' \in \Sigma^+$ such that $w' \neq w$. Thus, all root nodes are in the same equivalence class at every hypothesis guessed by our algorithm.

Let $sk(w)$ be a skeleton obtained from a positive counterexample $w \in L_t$ and $v, u_1, u_2, \cdots, u_n$ are nodes of $sk(w)$ such that v has child nodes $u_1, \cdots, u_n$, respectively. From Lemma 8, the first child node of every internal node in $sk(w)$ is labeled by a terminal in Σ and the other child nodes are internal node if exist. Now, let the label of u_1 be $a \in \Sigma$. In addition, let $A(v), A(u_2), \cdots, A(u_n) \in N_h$ be nonterminals which correspond to $v, u_2, \cdots, u_n$, respectively. Then, a rule $A(v) \to aA(u_2) \cdots A(u_n)$ is added to P_h in our algorithm. If both of $A \to a\beta$ and $A \to a\gamma$ are in P_h then $|\beta| = |\gamma|$ should hold.

If $\beta \neq \gamma$ then the following is processed. Let $\beta = A(p_1, p'_1) \cdots A(p_n, p'_n)$ and $\gamma = A(q_1, q'_1) \cdots A(q_n, q'_n)$ where $(p_1, p'_1), \cdots, (p_n, p'_n), (q_1, q'_1), \cdots, (q_n, q'_n) \in N_0$. If there are no $w \in W$ and $1 \leq i \leq n$ such that $w \in (L_{(p_i,p'_i)} - L_{(q_i,q'_i)}) \cup (L_{(q_i,q'_i)} - L_{(p_i,p'_i)})$, we call the pair $A \to a\beta$ and $A \to a\gamma$ a consistent pair. We call P_h is consistent if every pair of the form $A \to a\beta$ and $A \to a\gamma$ in P_h is a consistent pair. If P_h is consistent then delete one of $A \to a\beta$ or $A \to a\gamma$, arbitrarily.

If P_h is not consistent, i.e. there exist $w \in W$ and i such that $w \in (L_{(p_i,p'_i)} - L_{(q_i,q'_i)}) \cup (L_{(q_i,q'_i)} - L_{(p_i,p'_i)})$ then

$$W := W \cup \{a \cdot s_1 \cdots s_{i-1} \cdot w \cdot s_{i+1} \cdots s_n\} \tag{3}$$

where $s_j \in \Sigma^+$ $(1 \leq j \leq n)$ is the shortest word in W such that $Member(p_j \cdot s_j \cdot p'_j) = yes$. Now, $L_{(p_i,p'_i)} \neq L_{(q_i,q'_i)}$ holds. Thus, we must remake N_h from N_0 by the new W. Applying this process to P_h repeatedly, we can obtain a consistent P_h.

From these process, we can obtain a hypothesis grammar (Fig. 2). The learning algorithm is shown in A_1 (Fig. 3).

Lemma 12: In the algorithm A_1, it holds that $|N_h| \leq |N_t|$.

Proof: For every $(x, y) \in N_0$, there exists a node v of $sk \in SK$ such that $pre(v) = x$, $sspost(v) = y$. From Lemma 11,

$$\begin{aligned} L_{G_t}(A) = \{w \in \Sigma^+ | Member(x \cdot w \cdot y) = yes, \\ \forall w' \in \Sigma^+, \forall w'' \in \Sigma^+, w'w'' = w, \\ Member(x \cdot w' \cdot y) = no\} \end{aligned}$$

for some $A \in N_t$. In addition,

$$\begin{aligned} L_{G_t}(A) = \{w \in \Sigma^+ | Member(u \cdot w \cdot v) = yes, \\ \forall w' \in \Sigma^+, \forall w'' \in \Sigma^+, w'w'' = w, \\ Member(u \cdot w' \cdot v) = no\} \end{aligned}$$

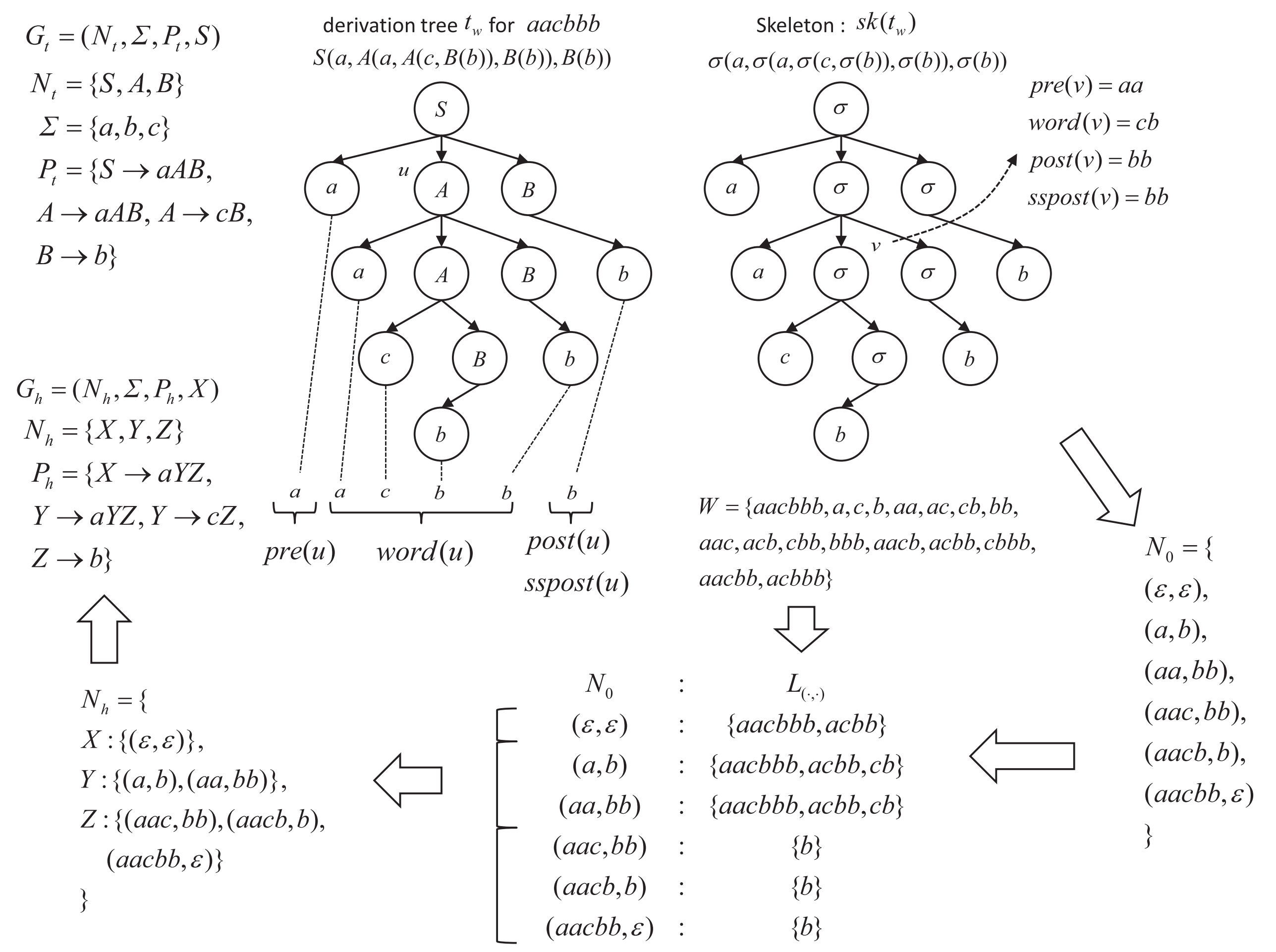
$G_t = (N_t, \Sigma, P_t, S)$
$N_t = \{S, A, B\}$
$\Sigma = \{a, b, c\}$
$P_t = \{S \to aAB,$
$A \to aAB, A \to cB,$
$B \to b\}$
derivation tree t_w for $aacbbb$
$S(a, A(a, A(c, B(b)), B(b)), B(b))$
u
$pre(u)$ $word(u)$ $post(u)$ $sspost(u)$
Skeleton : $sk(t_w)$
$\sigma(a, \sigma(a, \sigma(c, \sigma(b)), \sigma(b)), \sigma(b))$
v
$pre(v) = aa$
$word(v) = cb$
$post(v) = bb$
$sspost(v) = bb$
$W = \{aacbbb, a, c, b, aa, ac, cb, bb, aac, acb, cbb, bbb, aacb, acbb, cbbb, aacbb, acbbb\}$
$N_0 = \{ (\varepsilon,\varepsilon), (a,b), (aa,bb), (aac,bb), (aacb,b), (aacbb,\varepsilon) \}$
N_0 : $L_{(\cdot,\cdot)}$
$(\varepsilon,\varepsilon)$: $\{aacbbb, acbb\}$
(a,b) : $\{aacbbb, acbb, cb\}$
(aa,bb) : $\{aacbbb, acbb, cb\}$
(aac,bb) : $\{b\}$
$(aacb,b)$: $\{b\}$
$(aacbb,\varepsilon)$: $\{b\}$
$N_h = \{ X : \{(\varepsilon,\varepsilon)\}, Y : \{(a,b),(aa,bb)\}, Z : \{(aac,bb),(aacb,b), (aacbb,\varepsilon)\} \}$
$G_h = (N_h, \Sigma, P_h, X)$
$N_h = \{X, Y, Z\}$
$P_h = \{X \to aYZ,$
$Y \to aYZ, Y \to cZ,$
$Z \to b\}$

Fig. 2 An example of trees and a hypothesis grammar.

holds for any $(u,v) \in N_0$ such that $L_{(x,y)} = L_{(u,v)}$. It implies that every $B \in N_h$ corresponds to some $A \in N_t$. Thus, $|N_h| \le |N_t|$ holds. □

Lemma 13: Suppose $(x,y) \in N_0$ and $A \in N_h$ such that $A(x,y) = A$ and $Member(x \cdot w \cdot y) = no$ for $w \in W$. It holds that $w \notin L_{G_h}(A)$.
Proof: We denote the derivation tree on G_t which is isomorphic to $sk \in SK$ by t_{sk}. Let u be a node of sk. v_u denotes the node of t_{sk} such that u and v_u are corresponding nodes. In addition, $B(v_u) \in N_t$ denotes the nonterminal by whom v_u is labeled.

Suppose $w = a \in \Sigma$. For any node u of any $sk \in SK$ such that $A(pre(u), sspost(u)) = A$, then it holds that $a \notin L_{G_t}(B(v_u))$ from the assumption that $Member(x \cdot a \cdot y) = no$. It implies that $B(v_u) \to a$ is not in P_t. Every $sk \in SK$ is a skeleton of a derivation tree of G_t, then $A \to a$ is not in P_h.

Suppose that this lemma holds for any $|w| = n$ and $|aw| = n + 1$ where $a \in \Sigma$. For any node u of any $sk \in SK$ such that $A(pre(u), sspost(u)) = A$, it also holds that $aw \notin L_{G_t}(B(v_u))$ from the assumption. It implies that

- $B(v_u) \to a\beta$ is not in P_t for any $\beta \in N_t^+$, or
- $B(v_u) \to aC_1 \cdots C_m$ is in P_t for $m \le n$, but there exists $1 \le j \le m$ such that $w_j \notin L_{G_t}(C_j)$ for any separation of

$$w = w_1 w_2 \cdots w_m \ (w_i \in \Sigma^+, i = 1, \cdots, m).$$

If the former case holds then $aw \notin L_{G_h}(A)$. If the later case holds, there are two cases.

- u has child nodes $u_0, u_1, \cdots, u_m$ of sk such that the label of u_0 is a and $B(v_{u_i}) = C_i$ $(i = 1, \cdots, m)$.
- u has only one child node whose label is a.

If the later case holds then $aw \notin L_{G_h}(A)$. If the former case holds then $Member(pre(u_j) \cdot w_j \cdot sspost(u_j)) = no$ from $w_j \notin L_{G_t}(C_j)$. Now, $|w_j| \le n$ thus $w_j \notin L_{G_h}(D_j)$ where $D_j \in N_h$ and $D(pre(u_j), sspost(u_j)) = D_j$. It implies that $aw \notin L_{G_h}(A)$. □

Lemma 14: Let $G_h^{(t)} = (N_h^{(t)}, \Sigma, P_h^{(t)}, S_h)$ be the hypothesis grammar such that the t-th equivalence query is guessed by A_1. When a negative counterexample is given and the next hypothesis $G_h^{(t+1)} = (N_h^{(t+1)}, \Sigma, P_h^{(t+1)}, S_h)$ is guessed, it holds that $|N_h^{(t)}| < |N_h^{(t+1)}|$.
Proof: No skeleton is added to SK since the counterexample is negative. In addition, N_0 is not modified, too. Suppose that this lemma does not hold. For any pair of $(x_1, y_1) \in N_0$ and $(x_2, y_2) \in N_0$ such that $A(x_1, y_1), A(x_2, y_2) \in N_h^{(t)}$ and $A(x_1, y_1) \ne A(x_2, y_2)$, it holds that $C(x_1, y_1) \ne C(x_2, y_2)$

```
Learning algorithm $A_1$
INPUT: a parameter vector $\vec{n} = \{n_{a_1}, n_{a_2}, \cdots, n_{a_{|\Sigma|}}\}$
OUTPUT: a hypothesis suSDG $G_h = (N_h, \Sigma, P_h, S \in N_h)$
        such that $L(G_h) = L_t$
begin
    $N_0 := \{(\varepsilon, \varepsilon)\}, W := \emptyset, SK := \emptyset$
    $N_h := \{S\}, P_h := \emptyset$
    if ($Member(\varepsilon) == yes$) then $P_h := \{S \to \varepsilon\}$
    $G_h := (N_h, \Sigma, P_h, S)$
    while ($Equiv(G_h) == no$) begin
        let $w$ be the counterexample
        $W := W \cup \{y \in \Sigma^+ | xyz = w, x, z \in \Sigma^*\}$
        if ($w \in L_t$) then
            $SK := SK \cup \{sk(w)\}$
            let $V$ be the set of all nodes of $sk(w)$
            find $sspost(v)$ for all $v \in V$
            $N_0 := N_0 \cup \{(pre(v), sspost(v)) | v \in V\}$
        fi
        repeat
            make $N_h$ and $P_h$ from $SK$, $N_0$ and $W$
        until ($P_h$ is consistent)
        let $S_h \in N_h$ be the equivalence class which contains $(\varepsilon, \varepsilon) \in N_0$
        $G_h = (N_h, \Sigma, P_h, S_h)$
        if ($Member(\varepsilon) == yes$) then
            $N_h := N_h \cup \{S_\varepsilon\}$
            $P_h := P_h \cup \{S_\varepsilon \to \alpha | S_h \to \alpha \in P_h\} \cup \{S_\varepsilon \to \varepsilon\}$
            $G_h = (N_h, \Sigma, P_h, S_\varepsilon)$
        fi
    end
end.
```

Fig. 3 The learning algorithm A_1.

where $C(x_1, y_1), C(x_2, y_2) \in N_h^{(t+1)}$. Thus, $N_h^{(t)} = N_h^{(t+1)}$ w.r.t. $|N_h|$ is monotone increasing. It holds that $P_h^{(t)} = P_h^{(t+1)}$ from SK is not modified. It implies that $G_h^{(t)} = G_h^{(t+1)}$. The counterexample holds $w \notin L_t$. Thus, $Member(\varepsilon \cdot w \cdot \varepsilon) = no$. From Lemma 13, it implies that $w \notin L_{G_h^{(t+1)}} = L_{G_h^{(t)}}$. This is contradiction. □

Lemma 15: Let $G_h^{(t)} = (N_h^{(t)}, \Sigma, P_h^{(t)}, S)$ be the hypothesis grammar such that the t-th equivalence query is guessed by A_1. When a positive counterexample is given and the next hypothesis $G_h^{(t+1)} = (N_h^{(t+1)}, \Sigma, P_h^{(t+1)}, S)$ is guessed, it holds that

- $|N_h^{(t)}| < |N_h^{(t+1)}|$, or
- $|P_h^{(t)}| < |P_h^{(t+1)}|$.

Proof: Obviously, $|N_h|$ and $|P_h|$ are monotone increasing. Thus, if this lemma does not hold then $G_h^{(t)} = G_h^{(t+1)}$. Let w be the positive counterexample and t_w be its derivation tree on G_t. From $w \notin L_{G_h^{(t)}}$, there is no derivation $S_h \underset{G_h^{(t)}}{\overset{*}{\Rightarrow}} w$. There is also no derivation $S_h \underset{G_h^{(t+1)}}{\overset{*}{\Rightarrow}} w$ from the assumption. On the other hand, $sk(w)$ is in SK when $G_h^{(t+1)}$ is guessed. For every node v in $sk(w)$ and its child nodes $v_1, v_2, \cdots, v_m$, it holds that $A(v) \to aA(v_2) \cdots A(v_m)$ is in $P_h^{(t+1)}$. Thus, it holds that $S_h \underset{G_h^{(t+1)}}{\overset{*}{\Rightarrow}} w$. This is contradiction. □

Theorem 16: An suSDL is polynomial time learnable from membership queries, equivalence queries and the parameter vector $\vec{n}$ of G_t.

Proof: It is clear that if the algorithm A_1 terminates then $Equiv(G_h) = yes$. Thus it confirm the correctness of A_1.

From Lemma 14 and Lemma 12, negative counterexamples are given at most $|N_t|$ times. From Lemma 15 and Lemma 12, positive counterexamples are given at most $|N_t| + |P_t|$ times. Thus, equivalence queries are guessed at most $2|N_t| + |P_t|$ times.

Let l be the maximum length of counterexamples. The increase of $|W|$ by all counterexamples is bounded by $\frac{1}{2}l(l-1)(2|N_t| + |P_t|)$. If P_h is not consistent then W is modified by (3) and $|N_h|$ is increased. From Lemma 12, it happens at most $|N_t|$ times that P_h is not consistent. The increase of $|W|$ by (3) is also at most $|N_t|$. Thus, $|W|$ is bounded by $O(l^2(|N_t| + |P_t|))$ when A_1 is terminated.

Let $k = \max\{|w| \mid w \in W\}$. When (3) is applied, the length of the string added to W is at most $l + k$. Especially, at the first time that (3) is applied, the length of the string added to W is at most $2l$. Thus, k is bounded by $l(|N_t| + 1)$ when A_1 is terminated.

To find not consistent pair of rules takes at most $O((|N_h||\Sigma|)^2|W|)$ time. This is also bounded by a polynomial of $|N_t|, |\Sigma|, |P_t|, l$. Thus, this theorem holds. □

Example 17: We show an example run of A_1. Let $G_t = (N_t, \Sigma, P_t, S)$ be $N_t = \{S, A, B, D, E\}$, $\Sigma = \{a, b, c\}$, $P_t = \{ S \to aAB, S \to cD, A \to aDA, A \to b, B \to aEB, B \to b, D \to cE, D \to b, E \to b \}$ and $L_t = L(G_t)$. The parameter vector of G_t is ${}^t(2, 0, 1)$.

At first, an equivalence query for $G_h = (\{S\}, \Sigma, \emptyset, S)$ is guessed and suppose the counterexample $aabbabb$. Then, $W = \{ a, b, aa, ab, bb, ba, aab, abb, bba, bab, aabb, abba, bbab, babb, aabba, abbab, bbabb, aabbab, abbabb, aabbabb \}$. The skeleton $sk_{aabbabb}$ is $\sigma(a, \sigma(a, \sigma(b), \sigma(b)), \sigma(a, \sigma(b), \sigma(b)))$ and this skeleton is added to SK. Let names of internal nodes of $sk_{aabbabb}$ be $v_1, v_2, \cdots, v_7$ and place them on $v_1(a, v_2(a, v_3(b), v_4(b)), v_5(a, v_6(b), v_7(b)))$. Then, $sspost(v_1) = \varepsilon$, $sspost(v_2) = b$, $sspost(v_3) = bb$, $sspost(v_4) = b$, $sspost(v_5) = \varepsilon$, $sspost(v_6) = b$ and $sspost(v_7) = \varepsilon$. The set of nonterminal candidates is $N_0 = \{ (\varepsilon, \varepsilon), (a, b), (aa, bb), (aab, b), (aabb, \varepsilon), (aabba, b), (aabbab, \varepsilon) \}$.

The learner asks membership queries and finds $L_{(\varepsilon,\varepsilon)} = \{abb, aabbabb\}$, $L_{(a,b)} = L_{(aab,b)} = L_{(aabb,\varepsilon)} = L_{(aabbab,\varepsilon)} = \{b, abb\}$, $L_{(aa,bb)} = L_{(aabba,b)} = \{b\}$. Then, the learner obtains the equivalence class such that $N_h = \{ X : \{(\varepsilon, \varepsilon)\}, Y : \{(a, b), (aab, b), (aabb, \varepsilon), (aabbab, \varepsilon)\}, Z : \{(aa, bb), (aabba, b)\} \}$. The hypothesis grammar is $G_h = (N_h, \Sigma, P_h, X)$, here $P_h = \{ X \to aYY, Y \to aZY, Y \to b, Z \to b \}$ and an equivalence query is asked with G_h.

Suppose the next counterexample is ccb. $W := W \cup \{ c, cc, cb, ccb \}$. The constructed skeleton sk_{ccb} is $\sigma(c, \sigma(c, \sigma(b)))$ and the learner adds this skeleton to SK. Suppose that names of nodes of sk_{ccb} is u_1, u_2, u_3 and place them on $u_1(c, u_2(c, u_3(b)))$. Then, $sspost(u_1) = \varepsilon$, $sspost(u_2) = \varepsilon$ and $sspost(u_3) = \varepsilon$. The set of nonterminal candidates is $N_0 = \{ (\varepsilon, \varepsilon), (a, b), (aa, bb), (aab, b), (aabb, \varepsilon), (aabba, b),$

$(aabbab,\varepsilon)$, (c,ε), (cc,ε) }. Here, both of u_1 and v_1 correspond to $(\varepsilon,\varepsilon)$.

The learner distinguishes (aa,bb) and $(aabba,b)$ by $L_{(aa,bb)} = \{b,cb\}$ but $L_{(aabba,b)} = \{b\}$. Then, $N_h = \{\ X : \{(\varepsilon,\varepsilon)\},\ Y : \{(a,b),(aab,b),(aabb,\varepsilon),(aabbab,\varepsilon)\},\ Z : \{(aa,bb),(c,\varepsilon)\},\ T : \{(aabba,b),(cc,\varepsilon)\}\ \}$ and $P_h = X \to aYY, X \to cZ, Y \to aZY, Y \to aTY, Y \to b, Z \to cT, Z \to b, T \to b$ }. Now, the pair of $Y \to aZY$ and $Y \to aTY$ is not consistent pair. The learner adds $a \cdot cb \cdot b$ to W, then $L_{(a,b)} = L_{(aab,b)} = \{b,aab,acbb\}$ and $L_{(aabb,\varepsilon)} = L_{(aabbab,\varepsilon)} = \{b,abb\}$. Remaking N_h, the learner obtains $N_h = \{\ X : \{(\varepsilon,\varepsilon)\},\ Y : \{(a,b),(aab,b)\},\ Q : \{(aabb,\varepsilon),(aabbab,\varepsilon)\},\ Z : \{(aa,bb),(c,\varepsilon)\},\ T : \{(aabba,b),(cc,\varepsilon)\}\ \}$. The set of hypothesis rules is $P_h = \{\ X \to aYQ, X \to cZ, Y \to aZY, Y \to b, Q \to aTQ, Q \to b, Z \to cT, Z \to b, T \to b\ \}$. The hypothesis grammar $G_h = (N_h,\Sigma,P_h,S)$ is equivalent to G_t.

5. Learning via Example Based Queries

From Lemma 10, if the learner takes a positive counterexample w whose length is $|w| = l$, the number of skeletons whose yields are w is bounded by $O(l^{|\Sigma|})$. Thus, if we can run the algorithm A_1 with $O(l^{|\Sigma|})$ times, we can construct the learning algorithm for suSDLs via membership and equivalence queries. But, this is not a polynomial time learning algorithm.

We define the following superset query and show a polynomial time learning algorithm via membership, equivalence and superset queries.

[Superset query: $Super$]
INPUT: a hypothesis suSDG G_h
OUTPUT: *yes* if $L(G_h) \supseteq L_t$,
no and $w \in \Sigma^*$ if $L(G_h) \not\supseteq L_t$
where $w \in \Sigma^*$ is a positive counterexample such that $w \in (L_t - L(G_h))$.

With superset queries and USDGs (universal suSDGs), we can identify the parameter vector $\vec{n_t}$ of G_t (Fig. 4).

For any $X = \{x_1, x_2, \cdots, x_k\} \subseteq L_t$, a solution of Eq. (2),

$$M_X(\vec{n} + \vec{-1}) = \vec{-1},$$

is a candidate of the parameter vector of L_t.

```
Learning algorithm $A_2$
INPUT: none
OUTPUT: a hypothesis suSDG $G_h = (N_h, \Sigma, P_h, S \in N_h)$
        such that $L(G_h) = L_t$
begin
    $X := \emptyset$
    Let $G_h = (\{S_h\}, \Sigma, \emptyset, S_h)$
    while ($Super(G_h) == no$) begin
        let $w$ be the positive counterexample
        $X := X \cup \{w\}$
        solve $M_X(\vec{n} + \vec{-1}) = \vec{-1}$ for $\vec{n}$
        make a USDG for $\vec{n}$, then let it $G_h$
    end
    call $A_1$ with $\vec{n}$
end.
```

Fig. 4 Algorithm A_2 (learning from example based queries).

For $X \subseteq L_t$, it holds that $1 \leq rank(M_X) \leq |\Sigma|$ because M_X has $|\Sigma|$ columns. Let $r = \max\{rank(M_X) | X \subseteq L_t\}$. Obviously, r is monotone increasing and if $|X| \geq |\Sigma|$ then $rank(M_X) = r$. Now, we can show the following lemma.

Lemma 18: Let $r = \max\{rank(M_X) | X \subseteq L_t\}$ and $X^{(t)}$ be the X in A_2 at t-th superset query is guessed. Suppose $1 \leq |X^{(t)}| < |\Sigma|$ and m ($\leq r$) be the rank of $M_{X^{(t)}}$. If the next superset query guessed by A_2 and a positive counterexample $w \in L_t$ is given, i.e. $X^{(t+1)} = X^{(t)} \cup \{w\}$, then it holds that

- $rank(M_{X^{(t+1)}}) = m + 1$ if $m < r$.

Proof: If $|X| = 1$ then M_X is a row vector and $M_X \neq \vec{0}$, thus $m = 1$. Suppose if this lemma does not hold, then $rank(M_{X^{(t+1)}}) = m$. It implies that all solutions of $M_{X^{(t)}}(\vec{n} + \vec{-1}) = \vec{-1}$ are also solutions of $M_{X^{(t+1)}}(\vec{n} + \vec{-1}) = \vec{-1}$ and the number of both solutions are the same. Thus, $M_{X^{(t)}}(\vec{n} + \vec{-1}) = \vec{-1}$ and $M_{X^{(t+1)}}(\vec{n} + \vec{-1}) = \vec{-1}$ have same solutions. It implies that $G_h^{(t)}$ by whom t-th superset query is guessed is also consistent with $X^{(t+1)}$, i.e. $X^{(t+1)} \subseteq L(G_h^{(t)})$. This is contradiction. □

Theorem 19: An suSDL is polynomial time learnable from membership, equivalence and superset queries.
Proof: From Lemma 18, at most $|\Sigma|$ times superset queries are guessed by the learning algorithm A_2, and the parameter vector will be fixed. Thus, from Theorem 16, we can find the correct hypothesis by the algorithm A_1. The time complexity to solve Eq. (2) is bounded by a polynomial of $|X|$ and $|\Sigma|$. Thus, this theorem holds. □

The time complexity of Ishizaka [7]'s algorithm is, at least, larger than $O(|N_t|^3 l^6)$. On the other hand, the time complexity of our algorithm is bounded by $O(|N_t|^3 l^2 |\Sigma|^4)$. Usually, $|\Sigma| < l$ holds, then the time complexity of our algorithm is less than that of Ishizaka's algorithm.

6. Teachability

Teachability is one of a variation of machine learning problems. There are some different settings [2], [4], [9]. In Goldman and Mathias's setting [2], at first, the teacher makes a teaching set T which is a set of examples for the learner. Then, the adversary adds a set of examples A to T arbitrarily. The learner is given $E = A \cup T$ and try to identify the target language. The learner can not distinguish T from E when E is given. If the teacher can make T in polynomial time of the size of the representation G_t for the target language $L(G_t)$ and the learner can identify in polynomial time of the size of G_t and the total length of E, then such the representation class is polynomially T/L-teachable. If the teacher's complexity is not bounded by polynomial time then we call such the representation class is semi-poly T/L-teachable.

It is known that learnability via queries leads teachability.

Theorem 20 (Goldman and Mathias [2]): A representation class which is polynomial time learnable via example based queries is semi-poly T/L-teachable. □

Here, all of membership, equivalence and superset queries are example based queries. From Theorem 19, an suSDL is polynomial time learnable via example based queries, but the polynomial consists of the size of G_t and the maximum length of counterexamples. On the other hand, in Theorem 20, the polynomial only consists of the size of G_t.

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 SDGs or CFGs are consistency-easy class. In addition, suSDGs are also consistency-easy because we can make an suSDG which only generates T by the following process.

1. Find a parameter vector by a solution of Eq. (2).
2. Make all skeletons of T from the algorithm in Lemma 8.
3. Let all internal nodes of skeletons be different nonterminals and make rules from parent-child relations.

Definition 21 (de la Higuera [4]): 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 sample 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 [5]. Then, following theorems hold.

Theorem 22 (de la Higuera [4]): A consistency-easy class is identifiable in the limit from polynomial time and data iff it is semi-poly T/L-teachable. □

From this theorem, following negative results are known.

Theorem 23 (de la Higuera [4]): A language class whose equivalence is undecidable is not identifiable in the limit from polynomial time and data. □

Equivalence problem of both of linear grammars and context-free grammars are undecidable. Thus, these class of languages are not semi-poly T/L-teachable.

Theorem 24 (de la Higuera [4]): The class of SDGs is not identifiable in the limit from polynomial time and data.

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

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

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 an suSDG. Thus, the class of suSDGs 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 define new teachability which concerns the length of counterexamples.

Definition 25: Let $G_t = (N_t, \Sigma, P_t, S_t)$ be a CFG. 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 β, 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 subclass of CFGs is T/L-teachable in polynomial of the size of G_t and $tck(G_t)$, then we call the subclass of CFGs *teachable in polynomial examples*.

We can claim immediately that the class of suSDGs is teachable in polynomial examples from Theorem 19.

Theorem 26: The class of suSDGs is teachable in polynomial examples. □

7. Conclusions

We have shown the learning algorithm via membership, equivalence and superset queries for suSDLs. It implies that suSDLs are teachable in polynomial examples. It is equivalent that suSDLs are semi-poly T/L-teachable if the thickness is a parameter of the polynomial. Eventhough, it is still open problems for SDLs that teachability in polynomial examples and polynomial time query learnability.

The thickness is a parameter of the time complexity of teaching in polynomial examples. This parameter is also important in the time complexity of the parsing and the equivalence problem of SDLs. An inclusion problem between two SDGs is undecidable, but it is solvable between suSDGs [13]. In the future, if we find another parameter which is important to solve some example based queries, then it should also be important to clear the relation between the parameter and teachability or learnability.

Suppose a CFG which generates an suSDL. If we can solve the equivalence and the inclusion problems between such the CFG and an suSDG in polynomial time, then we can claim that there exists a conversion algorithm from the CFG to the equivalent suSDG. Because, let the CFG be the target of our learning algorithm, then we can obtain the suSDG which is equivalent to the CFG.

References

[1] D. Angluin, "Learning regular sets from queries and counterexamples," Inf. Comput., vol.75, no.2, pp.87–106, 1987.
[2] S.A. Goldman and H.D. Mathias, "Teaching a smart learner," J. Computer and System Sciences, vol.52, pp.255–267, 1996.
[3] M.A. Harrison, Introduction to Formal Language Theory, Addison-Wesley, Reading MA, 1978.
[4] C. de la Higuera, "Characteristic sets for polynomial grammatical

inference," Mach. Learn., vol.27, no.2, pp.125–138, 1997.

[5] 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. ICGI 2002), pp.134–148, 2002.

[6] P. Dupont, B. Lambeau, C. Damas, and A. van Lamsweerde, "The QSM algorithm and its application to software behavior model induction," Applied Artificial Intelligence, vol.22, nos.1-2, pp.77–115, 2008.

[7] H. Ishizaka, "Polynomial time learnability of simple deterministic languages," Mach. Learn., vol.5, no.2, pp.151–164, 1990.

[8] M. Linna, "Two decidability results for deterministic pushdown automata," J. Computer and System Sciences, vol.18, no.1, pp.92–107, Feb. 1979.

[9] A. Shinohara and S. Miyano, "Teachability in computational learning," New Generation Computing, vol.8, pp.337–347, 1991.

[10] Y. Takada, "A hierarchy of language families learnable by regular language learning," Inf. Comput., vol.123, no.2, pp.138–145, 1995.

[11] Y. Tajima, E. Tomita, M. Wakatsuki, and M. Terada, "Polynomial time learning of simple deterministic languages via queries and a representative sample," Theor. Comput. Sci., vol.329, nos.1-3, pp.203–221, 2004.

[12] M. Wakatsuki and E. Tomita, "On the upper bound of the shortest length of input strings to decide the equivalence of simple deterministic pushdown automata," IEICE Trans. Inf. & Syst. (Japanese Edition), vol.J75-D-I, no.10, pp.950–953, Oct. 1992.

[13] R. Yoshinaka, "Polynomial-time identification of an extension of very simple grammars from positive data," LNAI 4201 (Proc. ICGI 2006), pp.45–58. 2006.

Yasuhiro Tajima received his Ph.D. from The University of Electro-Communications in 2001. He had joined Ishikawajima-Harima Heavy Industries Co., Ltd., and Tokyo University of Agriculture and Technology. Currently, he is an associate professor at Okayama Prefectural University. His research interests are in Machine Learning and Text Mining. He is a member of JSAI and IPSJ.