New Generation Computing, 18(2000)217-242
Ohmsha, Ltd. and Springer-Verlag

Polynomial-time Learning of Elementary Formal Systems

Satoru MIYANO
Human Genome Center, Institute of Medical Science, University of Tokyo
4-6-1 Shirokanedai, Minato-ku, Tokyo 108-8639, JAPAN
miyano@ims.u-tokyo.ac.jp

Ayumi SHINOHARA
Department of Informatics, Kyushu University 33, Fukuoka 812-8581, JAPAN
ayumi@i.kyushu-u.ac.jp

Takeshi SHINOHARA
Department of Artificial Intelligence, Kyushu Institute of Technology, Iizuka 820-8502, JAPAN
shino@ai.kyutech.ac.jp

Received 08 December 1995
Revised manuscript received 30 December 1998

Abstract An elementary formal system (EFS) is a logic program consisting of definite clauses whose arguments have patterns instead of first-order terms. We investigate EFSs for polynomial-time PAC-learnability. A definite clause of an EFS is hereditary if every pattern in the body is a subword of a pattern in the head. With this new notion, we show that H-EFS(m, k, t, r) is polynomial-time learnable, which is the class of languages definable by EFSs consisting of at most m hereditary definite clauses with predicate symbols of arity at most r, where k and t bound the number of variable occurrences in the head and the number of atoms in the body, respectively. The class defined by all finite unions of EFSs in H-EFS(m, k, t, r) is also polynomial-time learnable. We also show an interesting series of **NC**-learnable classes of EFSs. As hardness results, the class of regular pattern languages is shown not polynomial-time learnable unless **RP**=**NP**. Furthermore, the related problem of deciding whether there is a common subsequence which is consistent with given positive and negative examples is shown **NP**-complete.

Keywords: Elementary Formal System (EFS), Polynomial-time PAC-learning, **NC**-learnable, Pattern Languages, Common Subsequence.

§1 Introduction

An elementary formal system (EFS for short), which was introduced by Smullyan,[26] can be considered as a logic program such as a Prolog program.[31] In EFSs, word concatenation is the only function symbol except constant symbols. Thus, terms in EFSs are words consisting of constant symbols and variables. Such terms are called patterns. Arikawa et al. proposed EFSs as a unifying framework for language learning.[5]

For example, the following set Γ of definite clauses is an EFS:

$$\Gamma = \left\{ \begin{array}{l} p(x_1x_2x_3) \leftarrow q(x_1, x_2, x_3) \\ q(a, b, c) \\ q(ax_1, bx_2, cx_3) \leftarrow q(x_1, x_2, x_3) \end{array} \right\},$$

where x_1, x_2, x_3 are variables, a, b, c are constant symbols taken from a finite alphabet Σ, and p, q are predicate symbols. We use two inference rules: substitution of nonempty patterns for variables and modus ponens. A clause C is said to be provable from an EFS Γ when C can be obtained from Γ by finitely many applications of the inference rules. We define a formal language by specifying an EFS Γ and a unary predicate symbol p as follows:

$$L(\Gamma, p) = \{w \in \Sigma^+ \mid p(w) \text{ is provable from } \Gamma\}$$

By the EFS Γ above we have $L(\Gamma, p) = \{a^n b^n c^n \mid n \geq 1\}$. Pattern languages introduced by Angluin[2] can also be defined by using the simplest EFSs. The language $L(\pi)$ of a pattern π is defined by an EFS with a single unit clause "$p(\pi)$."

Since the proposal by Arikawa et al., studies on learning EFSs have been developed focusing mainly on traditional inductive inference.[15,16,21,25] In this paper we discuss the learnability of EFSs in the sense of Valiant's PAC-learning.[28] In contrast with traditional inductive inference based on "identification in the limit" due to Gold[10] or "learning from minimally adequate teacher (MAT)" due to Angluin,[3] PAC (probably approximately correct) learning is to find an approximation of the target from random sampling, and therefore it has attracted much attention even from the viewpoint of practice. However, it seems that the contributions of studies on theory of PAC-learning are mainly on negative results derived from the theory of computational complexity. The main purpose of the present study is to find learnable classes of languages as general as possible using the framework of EFSs.

Shinohara[25] showed that the class of languages defined by length-bounded EFSs with at most n clauses is identifiable in the limit from positive data for any positive integer n. A clause $C = A \leftarrow A_1, \cdots, A_t$ is called length-bounded if for any substitution θ, the total length of patterns in $A_1\theta, \cdots, A_t\theta$ does not exceed the total length of patterns in $A\theta$. An EFS is called length-bounded if it consists of length-bounded clauses. The EFS Γ in the first example is length-bounded. A language is context-sensitive if and only if it is defined by a length-bounded EFS. One of the most important properties of length-bounded EFSs is that, for any finite set S of words, there exist only finitely many inequivalent length-bounded EFSs that are reduced with respect to S. An EFS Γ is said to be reduced

with respect to a set S if $S \subseteq L(\Gamma, p)$ but $S \not\subseteq L(\Gamma', p)$ for any $\Gamma' \subsetneq \Gamma$. In other words, a reduced EFS has no redundant clauses to cover a set. By counting more precisely the number of these reduced length-bounded EFSs we can show the Vapnik-Chervonenkis dimension (VC-dimension for short)[8] of the class of languages defined by length-bounded EFSs with at most n clauses is of polynomial order. Therefore, the class is PAC-learnable, if we do not care about computing time. Unfortunately we do not know whether this class is polynomial-time PAC-learnable or not. Nevertheless this fact motivates us enough to study the PAC-learnability of languages using the framework of EFSs. Some of the classes of EFSs, which we deal with in this paper, are a natural extension of context-free grammars. Abe discussed some related topics on context-free grammars and ranked node rewriting grammars.[1]

First, we consider the class of pattern languages, which is considered as a class of the simplest EFS languages. Ko and Tzeng[13] showed that the consistency problem for pattern languages is Δ_2^p-complete. The consistency problem for a class is to decide whether there exists a language in the class consistent with given positive and negative examples. If the consistency problem is **NP**-hard, then the class is not polynomial-time PAC-learnable under the assumption **RP** $\neq$ **NP**. Therefore, we cannot expect any efficient learning for pattern languages. Furthermore, Schapire[22] showed a stronger negative result. Such negative results seem to be quite natural, because even the membership problem for pattern languages is **NP**-complete.[2] Therefore, we should consider subclasses of pattern languages for which at least the membership problem is computable in polynomial time.

A pattern π is called regular if every variable of π appears just once in π. For example, a pattern "xy" is regular, but "xx" is not. For any regular pattern π and any word w, whether $w \in L(\pi)$ or not is decidable in $O(|\pi|+|w|)$ time, where $|\cdot|$ is the length function. The class of regular pattern languages is efficiently inferable from positive data.[23] However, as we will see later, the consistency problem is **NP**-complete for regular pattern languages. Also we will show the similar negative results for extended regular pattern languages, introduced by Shinohara,[24] where substitutions of the empty word for variables are allowed, as well as for common sequence languages. The latter result is independently shown by Jiang and Li.[12] From these results we know that even for regular pattern languages polynomial-time PAC-learning algorithms cannot be realized without any additional conditions.

To get another subclass of pattern languages for which the membership problem is computable in polynomial time, we restrict the total number of variable occurrences in patterns. When a pattern π defines a language containing a word w, $|\pi| \leq |w|$ and every subword in π without variables is a subword of w. This property that every constant word appearing in a pattern π whose language contains a word w is a subword of w is called "heredity". Therefore, the number of inequivalent languages that are defined by patterns with at most n variable occurrences and contain w is of polynomial order in $|w|$, if we fix n arbitrarily. Thus we can show the polynomial-time PAC-learnability of this subclass.

From the observation above we get two important facts:

- Even for regular pattern languages, polynomial-time PAC-learning is so hard to realize.
- Once we restrict the number of variable occurrences in patterns, the subclass of pattern languages is polynomial-time PAC-learnable.

Here we should note that the subclass of pattern languages with the restriction on the number of variable occurrences consists of infinitely many languages. Taking these facts as a starting point, we approach to more general classes of languages in the framework of EFSs.

We introduce "hereditary" EFSs that preserve the heredity of pattern languages. A clause $C = A \leftarrow A_1, \cdots, A_t$ is said to be hereditary, if any pattern in the body part $A_1, \cdots, A_t$ is a subword of some pattern in the head A. The EFS Γ in the first example is hereditary. Given a hereditary EFS Γ, a unary predicate symbol p, and a word w, whether $w \in L(\Gamma, p)$ or not is decidable in polynomial time with respect to $|w|$, when we consider the size of Γ as a constant. In this paper, we consider the following four parameters concerning the size of EFSs:

m: the number of clauses
k: the number of variable occurrences in the head
t: the number of atoms in the body
r: the arity of predicate symbols

The main positive result of this paper is that the class H-EFS(m, k, t, r) of languages defined by hereditary EFSs with parameters bounded by some constants m, k, t, r, respectively is polynomial-time PAC-learnable. We have only partial reasons for the restrictions on all the parameters. When we put no restriction on the number of clauses in EFSs, any finite languages can be defined, and therefore the VC-dimension of the class naturally becomes exponential. It seems reasonable to bound the number of clauses by some constant. However, it will turn out that this restriction does not always derive the polynomial VC-dimension of the class. For example, for the most general class of EFSs, called variable-bounded EFSs, by which any recursively enumerable language can be defined, the VC-dimension still remains in exponential order after bounding all the four parameters by constants. Even for hereditary EFSs with bounded number of clauses, when some of the other three parameters are not restricted by constants, the class is also of exponential VC-dimension. As for the number of variable occurrences, when we do not restrict it, we have the negative results for pattern languages.

§2 Preliminaries

In this section, we introduce some notations and definitions, and summarize basic lemmas.

2.1 Patterns and Elementary Formal Systems

Let Σ be a finite alphabet and $X = \{x_1, x_2, \ldots, y_1, y_2, \ldots\}$ be a set of variables. We assume that $\Sigma \cap X = \emptyset$. For an alphabet Δ, let Δ^* denote the set of all words over Δ, Δ^+ the set of all nonempty words, Δ^n the set of all words of length n, and $\Delta^{[n]}$ the set of all words of length n or less for $n \geq 0$.

A *pattern* is a word in $(\Sigma \cup X)^+$. A pattern π is called *regular* if each variable in π occurs exactly once in π. For instance, ax_1bx_2a is a regular pattern, but ax_1bx_1a is not, where a and b are in Σ. An *atom* is an expression of the form $p(\pi_1, \ldots, \pi_r)$, where p is a predicate symbol with arity r and $\pi_1, \ldots, \pi_r$ are patterns. A *definite clause* is a clause of the form

$$A \leftarrow A_1, \ldots, A_t,$$

where $A, A_1, \ldots, A_t$ are atoms and $t \geq 0$. The atom A is called the *head* and the part $A_1, \ldots, A_t$ the *body* of the definite clause. In case $t = 0$, we denote simply A instead of $A \leftarrow$. An *elementary formal system* (EFS for short) is a finite set of definite clauses.

A *substitution* θ is a homomorphism from patterns to patterns such that $\theta(a) = a$ for each $a \in \Sigma$. A substitution which maps some variables to the empty word is called an *ε-substitution*. In this paper, we do not allow any ε-substitutions without extra notice. For a pattern π and a substitution θ, we denote by $\pi\theta$ the image of π by θ. For an atom $A = p(\pi_1, \ldots, \pi_r)$ and a definite clause $C = A \leftarrow A_1, \ldots, A_t$, we define $A\theta = p(\pi_1\theta, \ldots, \pi_r\theta)$ and $C\theta = A\theta \leftarrow A_1\theta, \ldots, A_t\theta$.

A definite clause C is *provable from* an EFS Γ, denoted by $\Gamma \vdash C$, if C is obtained from Γ by finitely many applications of substitutions and modus ponens. That is, the relation $\Gamma \vdash C$ is defined inductively as follows:

(1) If $\Gamma \ni C$, then $\Gamma \vdash C$.
(2) If $\Gamma \vdash C$, then $\Gamma \vdash C\theta$ for any substitution θ.
(3) If $\Gamma \vdash A \leftarrow A_1, \ldots, A_t, A_{t+1}$ and $\Gamma \vdash A_{t+1}$, then $\Gamma \vdash A \leftarrow A_1, \ldots, A_t$.

For a predicate p with arity one, we define $L(\Gamma, p) = \{w \in \Sigma^+ \mid \Gamma \vdash p(w)\}$. A language $L \subseteq \Sigma^+$ is *definable by EFS* if there is an EFS Γ with a predicate symbol p with $L = L(\Gamma, p)$.

For a pattern π, the pattern language $L(\pi)$ is the set $\{w \in \Sigma^+ \mid w = \pi\theta$ for some substitution $\theta\}$.[2] It should be noticed that a pattern language $L(\pi)$ is also defined by an EFS $\Gamma = \{p(\pi)\}$.

Example 2.1
Consider the following EFS with $\Sigma = \{a, b\}$:

$$\Gamma = \left\{ \begin{array}{l} p(x_1x_2) \leftarrow q(x_1), r(x_2) \\ q(ax_1b) \leftarrow q(x_1) \\ q(ab) \\ r(x_1x_1) \end{array} \right\}.$$

The language defined by Γ is $L(\Gamma, p) = \{a^n b^n ww \mid n \geq 1, w \in \{a, b\}^+\}$. In the definite clause $p(x_1x_2) \leftarrow q(x_1), r(x_2)$, the head is the atom $p(x_1x_2)$ and the atoms $q(x_1), r(x_2)$ form the body. A substitution is denoted as a collection of assignments $\{x_1 := \pi_1, \ldots, x_n := \pi_n\}$. We can see $aabbaa \in L(\Gamma, p)$ as follows:

$$
\begin{array}{ll}
C_1 = r(x_1x_1) & (\text{ axiom }) \\
C_2 = r(aa) & (\ C_1\{x_1 := a\}\ ) \\
C_3 = q(ax_1b) \leftarrow q(x_1) & (\text{ axiom }) \\
C_4 = q(aabb) \leftarrow q(ab) & (\ C_3\{x_1 := ab\}\ ) \\
C_5 = q(ab) & (\text{ axiom }) \\
C_6 = q(aabb) & (\ C_4\ \&\ C_5\ ) \\
C_7 = p(x_1x_2) \leftarrow q(x_1), r(x_2) & (\text{ axiom }) \\
C_8 = p(aabbaa) \leftarrow q(aabb), r(aa) & (\ C_7\{x_1 := aabb, x_2 := aa\}\ ) \\
C_9 = p(aabbaa) \leftarrow q(aabb) & (\ C_2\ \&\ C_8\ ) \\
C_{10} = p(aabbaa) & (\ C_6\ \&\ C_9\ )
\end{array}
$$

Example 2.2
The languages $\{a^nb^n \mid n \geq 1\}$ and $\{a^nb^nc^n \mid n \geq 1\}$ are defined by the following EFSs Γ_1 and Γ_2, respectively.

$$\Gamma_1 = \left\{ \begin{array}{l} p(ax_1b) \leftarrow p(x_1) \\ p(ab) \end{array} \right\},$$

$$\Gamma_2 = \left\{ \begin{array}{l} p(x_1x_2x_3) \leftarrow q(x_1, x_2, x_3) \\ q(ax_1, bx_2, cx_3) \leftarrow q(x_1, x_2, x_3) \\ q(a, b, c) \end{array} \right\}.$$

2.2 Polynomial-time Learnability

This section briefly reviews some necessary notions for PAC-learnability due to Valiant.[28]

We call a subset c of Σ^* a *concept*. A concept c can be regarded as a function $c : \Sigma^* \to \{0, 1\}$, where $c(w) = 1$ if w is in the concept and $c(w) = 0$ otherwise. A *concept class* is a nonempty set $\mathcal{C} \subseteq 2^{\Sigma^*}$ of concepts. We use a finite alphabet Λ for representing concepts. A *representation* for a concept class $\mathcal{C}$ is a function $R : \mathcal{C} \to 2^{\Lambda^*}$ such that $R(c)$ is a nonempty subset of Λ^* for any c in $\mathcal{C}$ and $R(c_1) \cap R(c_2) = \emptyset$ for any distinct concepts c_1 and c_2 in $\mathcal{C}$. For each $c \in \mathcal{C}$, $R(c)$ is the set of *names* for c. The length of a name $\nu \in R(c)$ is the word length $|\nu|$ of ν. We denote the length of the shortest name for c by $l_{min}(c, R)$.

An *example* is an element $\langle w, a \rangle$ in $\Sigma^* \times \{0, 1\}$. An *example for a concept* c is a pair $\langle w, c(w) \rangle$ for $w \in \Sigma^*$. For a set $S \subseteq \Sigma^* \times \{0, 1\}$ of examples, we define $S_+ = \{w \mid \langle w, 1 \rangle \in S\}$ and $S_- = \{w \mid \langle w, 0 \rangle \in S\}$. We call a word in S_+ a *positive example* and a word in S_- a *negative example*, respectively. For two sets Y and N with $Y \cap N = \emptyset$, we say that a concept c is *consistent* with positive examples in Y and negative examples in N if $c(w) = 1$ for all $w \in Y$ and $c(w') = 0$ for all $w' \in N$. A concept $c \in \mathcal{C}$ is *consistent* with a set S of examples if c is consistent with positive examples in S_+ and negative examples in S_-. For a set S of examples, $l_{min}(S, R)$ is the length of the shortest name in R of any concept in $\mathcal{C}$ which is consistent with S.

Definition 2.1
A concept class $\mathcal{C}$ is *polynomial-time learnable* in a representation R if there exist an algorithm $\mathcal{A}$ and a polynomial $poly(\cdot, \cdot, \cdot, \cdot)$ which satisfy the following

conditions for any concept c in $\mathcal{C}$, any real numbers ε, δ $(0 < \varepsilon, \delta < 1)$, any integers $n \geq 0$, $s \geq 1$, and any probability distribution P on $\Sigma^{[n]}$:

(a) $\mathcal{A}$ takes ε, δ, n, and s. The real numbers ε and δ are called the *accuracy* and *confidence*, respectively. The integers n and s are called the *length parameter* and the *concept complexity*, respectively.
(b) $\mathcal{A}$ may call EXAMPLE, which generates examples for the concept $c \in \mathcal{C}$, randomly according to the probability distribution P on $\Sigma^{[n]}$.
(c) $\mathcal{A}$ outputs a name $\nu \in R(h)$ for some concept $h \in \mathcal{C}$ satisfying $P(c \cup h - c \cap h) < \varepsilon$ with probability at least $1 - \delta$, when $l_{min}(c, R) \leq s$ is satisfied.
(d) The running time of $\mathcal{A}$ is bounded by $poly(1/\varepsilon, 1/\delta, n, s)$.

When R is clear from the context, we simply say that a concept class $\mathcal{C}$ is polynomial-time learnable.

Remark 2.1
In the textbook due to Natarajan,[18] the input to the learning algorithm does not include the parameter s of concept complexity. This yields slightly different learnability. See the paper[11] for the equivalence and difference among these definitions of learnabilities.

Definition 2.2 (Blumer et al.[8])
Let $\mathcal{C}$ be a concept class. We say that $\mathcal{C}$ *shatters* a set $S \subseteq \Sigma^*$ if the set $\{c \cap S \mid c \in \mathcal{C}\}$ coincides with the set of *all* subsets of S. The *Vapnik-Chervonenkis dimension* (VC-dimension for short) of $\mathcal{C}$, denoted by $D_{VC}\mathcal{C}$, is the greatest integer d such that there exists a set of cardinality d that is shattered by $\mathcal{C}$. For an integer $n \geq 0$, we define $\mathcal{C}^{[n]} = \{c \cap \Sigma^{[n]} \mid c \in \mathcal{C}\}$. We say that $\mathcal{C}$ is of *polynomial dimension* if there exists a polynomial $d(n)$ such that $D_{VC}\mathcal{C}^{[n]} \leq d(n)$ for all $n \geq 0$.

Definition 2.3
A representation R for a concept class $\mathcal{C}$ is *polynomial-time computable* if there exist a deterministic algorithm B and a polynomial *poly* satisfying (a) and (b):

(a) B takes as input a pair (w, ν) of words $w \in \Sigma^*$ and $\nu \in \Lambda^*$.
(b) If $\nu \in R(c)$ for some $c \in \mathcal{C}$, then B halts in time $poly(|w| + |\nu|)$ and outputs $c(w)$.

Definition 2.4 (Natarajan[18])
Let $\mathcal{C}$ be a concept class with representation R, and $S \subseteq \Sigma^* \times \{0, 1\}$ be a finite set of examples. A deterministic algorithm is said to be a *fitting* for $\mathcal{C}$ in R if it takes as input S and outputs a name $\nu \in R(c)$ of a concept $c \in \mathcal{C}$ which is consistent with S if any. A fitting is said to be a *polynomial-time fitting* if it runs in time polynomial in the length of its input and $l_{min}(S, R)$. A *randomized fitting* for $\mathcal{C}$ in R is a randomized algorithm which takes as input S and outputs a name $\nu \in R(c)$ of a concept $c \in \mathcal{C}$ which is consistent with S, if any, with probability greater than $1/2$. A fitting is an *Occam fitting* if there exist a polynomial *poly* and a real number $0 \leq \alpha < 1$ such that for every input S, the output is of length at most $poly(n, l_{min}(S, R))|S|^{\alpha}$, where $|S|$ is the number of examples in S and

$n = \max\{|w| \mid \langle w, a\rangle \in S\}$.

Polynomial-time learnability is characterized as follows:

Lemma 2.1 (Blumer et al.,[8] Haussler et al.,[11] Natarajan[17,18])
Let $\mathcal{C}$ be a concept class and R be a polynomial-time computable representation for $\mathcal{C}$.

(1) $\mathcal{C}$ is polynomial-time learnable in R if $\mathcal{C}$ is of polynomial dimension and there exists a polynomial-time fitting for $\mathcal{C}$ in R.
(2) $\mathcal{C}$ is polynomial-time learnable in R if there exists a polynomial-time Occam fitting for $\mathcal{C}$ in R.
(3) $\mathcal{C}$ is polynomial-time learnable in R only if there exists a randomized polynomial-time fitting for $\mathcal{C}$ in R.

§3 Regular Patterns Are Hard to Learn

For a concept class $\mathcal{C}$ with a representation R, we consider the following problem:

Consistency Problem for $\mathcal{C}$ in R

Instance: A set of examples $S \subseteq \Sigma^* \times \{0,1\}$ with $S_+ \cap S_- = \emptyset$.

Question: Is there a name $\nu \in R(h)$ of a concept $h \in \mathcal{C}$ which is consistent with S?

If the consistency problem for $\mathcal{C}$ in R is shown **NP**-complete, we can say that $\mathcal{C}$ is not polynomial-time learnable in R under the assumption of **RP**≠**NP**, since there is no randomized polynomial-time fitting for $\mathcal{C}$ in R by Lemma 2.1 (3).

In this section, we deal with only regular patterns and abbreviate variables $x_1, x_2, \ldots$ by the same symbol x for simplicity.

Theorem 3.1
The consistency problem for the class of regular pattern languages is **NP**-complete.

Proof
Obviously the problem is in **NP**. We give a polynomial-time reduction from 3SAT to the problem. Let $F = C_1 \cdots C_m$ be a formula in 3-CNF with variables $u_1, u_2, \ldots, u_n$. Without loss of generality, we can assume that C_i does not contain both u_k and $\overline{u_k}$ for any $1 \leq k \leq n$. We define Y of positive examples and N of negative examples so that F is satisfiable if and only if there is a regular pattern π consistent with Y and N.

First, we use $n+1$ positive examples $s_0, \ldots, s_n$ and $n+1$ negative examples $t_0, \ldots, t_n$ over $\Sigma = \{0,1\}$ so that any consistent regular pattern π must be of the form $\pi = \tau_1\tau_2\cdots\tau_n$, where $\tau_i = 0x$ or $x0$ $(1 \leq i \leq n)$. We define

$s_0 = (00)^n$;
$s_i = (00)^{i-1}010(00)^{n-i}$ for $1 \leq i \leq n$;

$t_0 = 0^{2n-1}$;
$t_i = (00)^{i-1}11(00)^{n-i}$ for $1 \leq i \leq n$.

From the positive example s_0 with $|s_0| = 2n$ and the negative example t_0 with $|t_0| = 2n-1$, we see that any pattern π consistent with s_0 and t_0 satisfies $\pi \in \{0, x\}^+$ and $|\pi| = 2n$. Thus we can denote $\pi = \sigma_1\sigma_2\cdots\sigma_{2n}$ with $\sigma_k \in \{0, x\}$ for $1 \leq k \leq 2n$.

For $1 \leq i \leq n$, since $|s_i| = 2n+1$ and ε-substitutions are not allowed, the $(2i)$th character 1 of s_i must match with either σ_{2i-1} or σ_{2i} of pattern π, i.e., either σ_{2i-1} or σ_{2i} is x. On the other hand, both of them are not x since $t_i \notin L(\pi)$. Therefore each $\tau_i = \sigma_{2i-1}\sigma_{2i}$ is $0x$ or $x0$ for $1 \leq i \leq n$.

For $C_1, \ldots, C_m$ of F, we use additional negative examples $d_1, \ldots, d_m$ to forbid that all of the three literals in C_i are assigned to *false* for $1 \leq i \leq m$. For $1 \leq i \leq m$, we define

$$d_i = r_1 r_2 \ldots r_n, \text{ where } r_k = \begin{cases} 01 & \text{if literal } u_k \text{ appears in } C_i \\ 10 & \text{if literal } \overline{u_k} \text{ appears in } C_i \\ 00 & \text{otherwise.} \end{cases}$$

Since we have assumed that u_k and $\overline{u_k}$ do not both appear in any C_i, the above d_i is well-defined. Then let $Y = \{s_0, s_1, \ldots, s_n\}$ and $N = \{t_0, t_1, \ldots, t_n, d_1, \ldots d_m\}$.

Assume that F is satisfiable under a truth assignment $\hat{u}_1, \ldots, \hat{u}_n$. Then we define a regular pattern $\pi = \tau_1 \cdots \tau_n$ by putting $\tau_i = x0$ if $\hat{u}_i = \mathit{true}$ and $\tau_i = 0x$ if $\hat{u}_i = \mathit{false}$ for $1 \leq i \leq n$. It is easy to see from the definitions of Y and N that π is consistent with Y and N. In fact, it is clear that $s_i \in L(\pi)$ and $t_i \notin L(\pi)$ for $0 \leq i \leq n$. Since F is satisfiable by the assumption, each C_i contains either u_k with $\hat{u}_k = \mathit{true}$ or $\overline{u_k}$ with $\hat{u}_k = \mathit{false}$ for some k. In the former case, $\tau_k = x0$ and $r_k = 01$ guarantee $d_i \notin L(\pi)$. In the latter case, $\tau_k = 0x$ and $r_k = 10$ witness $d_i \notin L(\pi)$ similarly.

Conversely, we assume that there exists a regular pattern π consistent with Y and N. Then π must be of the form $\tau_1\tau_2\cdots\tau_n$, where $\tau \in \{0x, x0\}$, since π is consistent with s_i's and t_i's for $0 \leq i \leq n$. We define the truth assignment $\hat{u}_i = \mathit{true}$ if $\tau_i = x0$ and $\hat{u}_i = \mathit{false}$ if $\tau_i = 0x$ for $1 \leq i \leq m$. For each C_i, at least one literal must be assigned to *true* since $L(\pi)$ does not contain the negative example d_i. ■

Example 3.1
For an instance $F = (u_1 + \overline{u_2} + u_3) \cdot (\overline{u_1} + u_3 + u_4)$ of 3SAT, we construct five positive examples { 00000000, 010000000, 000100000, 000001000, 000000010} and seven negative examples { 0000000, 11000000, 00110000, 00001100, 00000011, 01100100, 10000101}. A regular pattern consistent with these examples, say $\pi = x0x00xx0$, corresponds to the truth assignment $\hat{u}_1 = \mathit{true}$, $\hat{u}_2 = \mathit{true}$, $\hat{u}_3 = \mathit{false}$, $\hat{u}_4 = \mathit{true}$, which satisfies the formula F.

Theorem 3.2
Let $m \geq 1$ be an integer. The consistency problem for the class

$$\{L(\pi_1) \cup \cdots \cup L(\pi_m) \mid \pi_i\text{'s are regular patterns}\}$$

is **NP**-complete for any $m \geq 1$.

Proof
The case of $m = 1$ is Theorem 3.1. For $m \geq 2$, we will reduce the problem to the case of $m = 1$. Let Y and N be the sets of positive examples and negative examples given in the proof of Theorem 3.1. Then we define as follows:

$$Y' = Y \cup \{1^i \mid 1 \leq i \leq m-1\},$$
$$N' = N \cup \{1^m\}.$$

We will show that the following two statements are equivalent:

(1) There is a set M consisting of m regular patterns such that $\cup_{\pi \in M} L(\pi)$ is consistent with Y' and N'.
(2) There is a regular pattern π_0 such that $L(\pi_0)$ is consistent with Y and N.

We show that (1) implies (2). For each $i = 1, \ldots, m-1$, the set M contains a regular pattern π with $1^i \in L(\pi)$ since 1^i is a positive example. Then π is in $\{1, x\}^{[i]}$. However, if a variable occurs in π, the negative example 1^m is also in $L(\pi)$. Therefore, $\pi = 1^i$. Thus M contains $m-1$ patterns 1, 11, $\ldots, 1^{m-1}$ without any variables. Therefore M must contain a regular pattern π_0 such that $L(\pi_0)$ is consistent with Y and N. The converse is almost clear. ■

An extended regular pattern language $\tilde{L}(\pi)$ [24] is defined by allowing ε-substitutions to a regular pattern π. The ε-substitutions might change the behavior entirely since the length of the possible patterns can not be bounded. However, any extended regular pattern language can be defined by a regular pattern in *canonical form* $\pi = w_0 x w_1 x \cdots x w_{n-1} x w_n$ with $w_0, w_n \in \Sigma^*$ and $w_i \in \Sigma^+$ for $1 \leq i \leq n-1$, since two consecutive variables are reduced to a single variable.[24]

Theorem 3.3
Let $m \geq 1$ be an integer. The consistency problem for the class

$$\{\tilde{L}(\pi_1) \cup \cdots \cup \tilde{L}(\pi_m) \mid \pi_i\text{'s are regular patterns}\}$$

is **NP**-complete.

Proof
We will show only the case of $m = 1$. For $m \geq 2$, we can reduce the problem to the case of $m = 1$ in the same way as the proof of Theorem 3.2.

The basic idea of the proof is similar to that of Theorem 3.1. For an instance $F = C_1 \cdots C_m$ of 3SAT with variables $u_1, u_2, \ldots, u_n$, we give the following two positive examples over $\Sigma = \{0, 1, \#\}$:

$$s_1 = 0\#0\#\cdots\#0 \text{ with } |s_1| = 2n-1,$$
$$s_2 = 00\#00\#\cdots\#00 \text{ with } |s_2| = 3n-1.$$

We also use $2n-1$ negative examples $\check{t}_i$'s and t_i's:

$\check{t}_i$: the word obtained by deleting the ith # of s_1, $(1 \leq i \leq n-1)$,

t_i : the word obtained by replacing the ith 0 of s_1 with 101, $(1 \leq i \leq n)$.

It can be noticed that a regular pattern in canonical form which is consistent with positive examples s_1, s_2, and negative examples $\check{t}_1, \ldots \check{t}_{n-1}, t_1, \ldots, t_n$ must be of the form:

$$\pi = \tau_1 \# \tau_2 \# \ldots \# \tau_n, \text{ where } \tau_i \in \{0x, x0\} \text{ for } 1 \leq i \leq n.$$

Then for each C_i of F, we define the following additional negative examples :

$$d_i = r_1 \# r_2 \# \ldots \# r_n,$$

$$\text{where } r_k = \begin{cases} 01 & \text{if literal } u_k \text{ appears in } C_i \\ 10 & \text{if literal } \overline{u_k} \text{ appears in } C_i \\ 0 & \text{otherwise.} \end{cases}$$

We can verify that there is a regular pattern π such that $\tilde{L}(\pi)$ is consistent with these examples if and only if F is satisfiable in the same way. ■

The common subsequence problem has been dealt with from independent viewpoints, such as text processing, data compression, or DNA sequences.[14,30] It is known that the longest common subsequence problem is **NP**-complete.[14] The problem of finding a sequence $a_1 \cdots a_n$ which is common to all positive examples but not common to any negative examples is equivalent to finding a regular pattern $\pi = xa_1x \cdots xa_nx$ such that the extended regular pattern language $\tilde{L}(\pi)$ is consistent with the positive and negative examples. Thus we call a regular pattern of the form $\pi = xa_1xa_2x \cdots xa_nx$ with $a_i \in \Sigma$ for $1 \leq i \leq n$ a *common subsequence*.

Theorem 3.4
The consistency problem for the class of common subsequence languages is **NP**-complete.

Proof
Since the basic idea is also similar, details are omitted. Two positive examples

$$s_1 = 01\#01\# \ldots \#01 \text{ with } |s_1| = 3n-1,$$

$$s_2 = 10\#10\# \ldots \#10 \text{ with } |s_2| = 3n-1$$

and $2n-1$ negative examples

$\check{t}_i$: the word obtained by deleting the ith # of s_1, $(1 \leq i \leq n-1)$,

t_i : the word obtained by deleting the ith 01 of s_1, $(1 \leq i \leq n)$,

make a consistent common subsequence restricted to the form:

$$\pi = a_1\#a_2\#\ldots\#a_n, \text{ where } a_i \in \{0,1\} \text{ for } 1 \leq i \leq n.$$

Additional m negative examples

$$d_i = r_1\#r_2\#\ldots\#r_n,$$

$$\text{where } r_k = \begin{cases} 0 & \text{if literal } u_k \text{ appears in } C_i \\ 1 & \text{if literal } \overline{u_k} \text{ appears in } C_i \\ 01 & \text{otherwise,} \end{cases}$$

guarantee the equivalence between the existence of a consistent common subsequence and the satisfiability of a given 3-CNF formula. ■

§4 VC-dimension of EFSs

Consider a definite clause

$$q_0(\pi_1^0,\ldots,\pi_{r_0}^0) \leftarrow q_1(\pi_1^1,\ldots,\pi_{r_1}^1),\ldots,q_t(\pi_1^t,\ldots,\pi_{r_t}^t).$$

An EFS is defined as a finite collection of such clauses. In order to get polynomial-time learnable classes of EFSs, in this section, we investigate the VC-dimensions of EFSs. If a class $\mathcal{C}$ is of polynomial dimension, then a polynomial-time fitting for $\mathcal{C}$ suffices. Otherwise, we need to devise a polynomial-time Occam fitting. Although exponential VC-dimension does not imply the impossibility of polynomial-time learning, it seems reasonable to restrict EFSs to avoid the exponential VC-dimension. We focus our attention on the following features of EFSs.

(1) the relationship between patterns in the head and patterns in the body.
(2) m: the number of clauses in the EFS.
(3) k: the number of variable occurrences in the head.
(4) t: the number of atoms in the body.
(5) r: the arity of a predicate.

It has been shown in Arikawa et al.[5] that any recursively enumerable set is definable by a *variable-bounded* EFS whose clauses do not contain any internal variables, i.e., in each clause all variables in the body also appear in the head. The class of variable-bounded EFSs seems too large to realize efficient learning algorithms. In fact, we can show any finite language can be defined by a variable-bounded EFS with parameters m, k, t, r bounded by constants.

Definition 4.1
For $m, k, t, r \geq 0$, VB-EFS(m, k, t, r) is the class of languages definable by variable-bounded EFSs with at most m definite clauses each of which satisfies the following conditions:

(a) The number of variable occurrences in the head is at most k.
(b) The number of atoms in the body is at most t.
(c) The arity of each predicate symbol is at most r.

To measure dimensions of EFSs, we use the the following lemma:

Lemma 4.1 (Natarajan[17])
A concept class $\mathcal{C}$ is of polynomial dimension if and only if there exists a polynomial $poly(n)$ such that $\log_2 |\mathcal{C}^{[n]}| \leq poly(n)$ for all $n \geq 0$.

We say that a concept class $\mathcal{C}$ is of *exponential dimension* if $\mathcal{C}$ is not of polynomial dimension. First, we show that the VC-dimension of the class of languages definable by variable-bounded EFSs is exponential, even if all the parameters m, k, t, r are bounded by constants. Hereafter we assume that the alphabet Σ contains two or more constant symbols, because any class of languages over a singleton alphabet is always of polynomial dimension.

Theorem 4.1
VB-EFS(m, k, t, r) is of exponential dimension if $m \geq 9$, $k \geq 2$, $t \geq 2$, and $r \geq 1$.

Proof
We assume that the alphabet Σ contains at least two symbols 0 and 1. If a class $\mathcal{C}$ contains any language $Y \subseteq \{0,1\}^n$ for any $n \geq 0$, then $\mathcal{C}^{[n]} = \{c \cap \Sigma^{[n]} \mid c \in \mathcal{C}\} \supseteq \{c \cap \{0,1\}^n \mid c \in \mathcal{C}\} = 2^{\{0,1\}^n}$, and therefore $|\mathcal{C}^{[n]}| \geq 2^{2^n}$ shows the exponential dimension of $\mathcal{C}$.

Let $Y = \{w_1, \ldots, w_i\} \subseteq \{0,1\}^n$ be any set of words of length n. Consider the following variable-bounded EFS Γ:

$$\Gamma = \left\{ \begin{array}{l} q_1(0^n) \\ q_1(x_1x_2) \leftarrow q_1(x_2x_1) \\ q_1(1x_1) \leftarrow q_1(0x_1) \\ q_2(w_1 \cdots w_i) \\ q_2(x_1x_2) \leftarrow q_1(x_2), q_2(x_2x_1) \\ q_3(x_1) \leftarrow q_2(x_1) \\ q_3(x_1) \leftarrow q_3(x_10) \\ q_3(x_1) \leftarrow q_3(x_11) \\ p(x_1) \leftarrow q_3(x_1), q_1(x_1) \end{array} \right\}.$$

In the above EFS, we define three auxiliary predicates q_1, q_2, q_3 such that

1. $\Gamma \vdash q_1(u) \Longleftrightarrow u \in \{0,1\}^n$,
2. $\Gamma \vdash q_2(u) \Longleftrightarrow u = w_j \cdots w_i w_1 \cdots w_{j-1}$ for some $1 \leq j \leq i$, and
3. $\Gamma \vdash q_3(u) \Longleftrightarrow \Gamma \vdash q_2(v)$ and u is a prefix of v.

Thus we have $L(\Gamma, p) = Y$. Note that only two clauses $q_1(0^n)$ and $q_2(w_1 \cdots w_i)$ in Γ depend on n and Y. The number of clauses in Γ is 9, the number of variable occurrences is at most 2, the number of atoms in the bodies is at most 2, and all the predicate symbols are of arity 1. Therefore, if $m \geq 9$, $k \geq 2$, $t \geq 2$, and $r \geq 1$, the VC-dimension of the class VB-EFS(m, k, t, r) is exponential. ■

By strengthening the view on the relationship between patterns in the head and patterns in the body of variable-bounded clauses, we define the following notion that is the key to finding polynomial-time learnable classes.

Definition 4.2
We say that a definite clause is *hereditary* if each pattern in the body is a subword of some pattern in the head. An EFS Γ is *hereditary* if each definite clause in Γ is hereditary.

Example 4.1
The definite clauses $p(ax_1bc) \leftarrow q(ax_1), r(x_1b)$ and $p(ax_1, bx_2, cx_3) \leftarrow q(x_1, x_2, x_3)$ are hereditary. But the definite clause $p(ax_1) \leftarrow q(bx_1)$ is not hereditary.

Definition 4.3
For $m, k, t, r \geq 0$, H-EFS(m, k, t, r) is the class of languages definable by hereditary EFSs with at most m definite clauses each of which satisfies the following conditions:

(a) The number of variable occurrences in the head is at most k.
(b) The number of atoms in the body is at most t.
(c) The arity of each predicate symbol is at most r.

We define H-EFS$(m, *, t, r) = \bigcup_{k \geq 1}$ H-EFS(m, k, t, r) for $m, r, t \geq 0$. That is, H-EFS$(m, *, t, r)$ is the class of languages defined by hereditary EFSs that allow an arbitrary number of variable occurrences in heads in their definite clauses while other restrictions are kept as they are. We also define other classes such as H-EFS$(m, *, *, r)$ in a similar way.

Since any finite language $\{w_1, \ldots, w_i\}$ is definable by a hereditary EFS $\{p(w_1), \ldots, p(w_i)\}$, the class H-EFS$(*, k, t, r)$ is of exponential dimension. Therefore we need to restrict the number of definite clauses in EFSs to get a concept class of polynomial dimension. However, the restriction of the number of clauses does not immediately mean polynomial dimension. In reality, if we can use as many variables in a pattern as we want and predicate symbols of large arity, the dimension of languages definable by hereditary EFSs still remains exponential.

Theorem 4.2
H-EFS$(m, *, t, *)$ is of exponential dimension, if $m \geq 10$, $t \geq 1$.

Proof
We show that for every set $Y = \{w_1, \ldots, w_i\}$ of words over $\Sigma = \{0, 1\}$ of length n, the language $L = \{01w \mid w \in Y\}$ is definable by a hereditary EFS Γ consisting of ten definite clauses with at most one atom in the body.

For simplicity we fix the length n to be 3. First, consider the following EFS:

$$\Gamma_0 = \left\{ \begin{array}{l} (1)\ q_1(01, 01, 0, 0, 0) \\ (2)\ q_2(01, 01, x_1, x_2, x_3) \leftarrow q_1(01, 01, x_1, x_2, x_3) \\ (3)\ q_1(01, x_1, x_2, x_3, 1) \leftarrow q_2(01, x_1, x_2, x_3, 0) \\ (4)\ q_2(01, 0, x_1, x_2, x_3) \leftarrow q_2(01, x_1, x_2, x_3, 1) \\ (5)\ q_1(01, 01, x_1, x_2, x_3) \leftarrow q_2(01, x_1, x_2, x_3, 01) \\ (6)\ q_1(01, x_1, x_2, x_3, x_4) \leftarrow q_1(01, x_4, x_1, x_2, x_3) \\ (7)\ p(01x_1x_2x_3) \leftarrow q_1(01, 01, x_1, x_2, x_3) \end{array} \right\}.$$

The first argument "01" in q_1 and q_2 is just for keeping clauses hereditary. The arguments from the second to the last are used to simulate a circular working tape of a Turing machine that increments a number in binary code, where "01" represents the end marker and the left of the second argument is connected with the last. Using (2) once corresponds to incrementing a number by one. (6) is used to rotate the arguments from the second to the last. From these observations, we can show that

$$\Gamma_0 \vdash p(w) \Longleftrightarrow w = 01v \text{ and } v \in \Sigma^n.$$

To get an EFS that defines $L = \{01w \mid w \in Y\}$, we use 2^n additional arguments. By the following EFS Γ, we define L in case $Y = \{010, 011, 101\}$. In this case, words in Y correspond to integers 2, 3, 5, respectively, when they are considered as binary numbers. We rewrite (1) as

(1) $q_1(01, 01, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0)$,

where the i-th argument in the additional part is set to 1 if and only if the number i is represented by a word in Y. In (2) we rotate the additional arguments to the left.

(2) $q_2(01, 01, x_1, x_2, x_3, y_1, \ldots, y_7, y_0) \leftarrow q_1(01, 01, x_1, x_2, x_3, y_0, y_1, \ldots, y_7)$

In (3) to (6) the additional arguments are passed as they are.

(3) $q_1(01, x_1, x_2, x_3, 1, y_0, \ldots, y_7) \leftarrow q_2(01, x_1, x_2, x_3, 0, y_0, \ldots, y_7)$
(4) $q_2(01, 0, x_1, x_2, x_3, y_0, \ldots, y_7) \leftarrow q_2(01, x_1, x_2, x_3, 1, y_0, \ldots, y_7)$
(5) $q_1(01, 01, x_1, x_2, x_3, y_0, \ldots, y_7) \leftarrow q_2(01, x_1, x_2, x_3, 01, y_0, \ldots, y_7)$
(6) $q_1(01, x_1, x_2, x_3, x_4, y_0, \ldots, y_7) \leftarrow q_1(01, x_4, x_1, x_2, x_3, y_0, \ldots, y_7)$

In (7), we define another predicate q_3 as

(7) $q_3(01x_1x_2x_3, y_1, \ldots, y_7, 0) \leftarrow q_1(01, 01, x_1, x_2, x_3, 1, y_1, \ldots, y_7)$,

where we should note that the first additional argument in the body should be 1. At this point, we can see that

$$\Gamma \vdash q_3(01b_1b_2b_3, b'_0, \ldots, b'_7) \Longleftrightarrow b_1b_2b_3 \in Y,$$

where $b_i, b'_j \in \{0, 1\}$, because the additional arguments are rotated exactly as often as the number is incremented. To extract the first argument "$01b_1b_2b_3$," we clear all the additional arguments by "0."

(8) $q_3(01x_1, y_1, \ldots, y_7, 0) \leftarrow q_3(01x_1, 0, y_1, \ldots, y_7)$
(9) $q_3(01x_1, y_1, \ldots, y_7, 0) \leftarrow q_3(01x_1, 1, y_1, \ldots, y_7)$

After clearing all the additional arguments, the target word is passed to the predicate p.

(10) $p(01x_1) \leftarrow q_3(01x_1, 0, 0, 0, 0, 0, 0, 0, 0)$

■

From Theorem 4.1 and Theorem 4.2, at least in some sense, we need to restrict EFSs to get a class of polynomial dimension. Although we have not fully investigated the necessity of restrictions on the parameters m, k, t, r, we can show that H-EFS$(m, *, *, r)$ is of polynomial dimension. Here we should note that there are infinitely many languages definable by hereditary EFSs even if all the parameters m, k, t, r are bounded by small constants. To measure the dimension of H-EFS$(m, *, *, r)$, we may consider only these EFSs without any redundancy, called reduced EFSs.

Definition 4.4
A pair (Γ, p) of an EFS Γ and a predicate symbol p is said to be *reduced* with respect to a set Y of words if $Y \subseteq L(\Gamma, p)$ and $Y \not\subseteq L(\Gamma', p)$ for any $\Gamma' \subsetneq \Gamma$.

Moreover, hereditary EFSs have the following property that is very well suited for learning from examples. The proof is obvious from the definition.

Lemma 4.2
Let Y be a nonempty set of words in Σ^+ and let (Γ, p) be a pair of a hereditary EFS and its predicate symbol of arity one. If (Γ, p) is reduced with respect to Y, then for each definite clause

$$q_0(\pi_1^0, \ldots, \pi_{r_0}^0) \leftarrow q_1(\pi_1^1, \ldots, \pi_{r_1}^1), \ldots, q_t(\pi_1^t, \ldots, \pi_{r_t}^t)$$

in Γ, there exists a substitution θ such that all $\pi_i^j\theta$'s are subwords of some $w \in Y$.

Theorem 4.3
H-EFS$(m, *, *, r)$ is of polynomial dimension for any $m, r \geq 0$.

Proof
Let H-EFS$(m, *, *, r)^{[n]} = \{L \cap \Sigma^{[n]} \mid L \in \text{H-EFS}(m, *, *, r)\}$ for $n \geq 0$. We evaluate the cardinality of H-EFS$(m, *, *, r)^{[n]}$. Let (Γ, p) be a pair of a hereditary EFS Γ and its predicate symbol p of arity one. Since the number of definite clauses is bounded by m, we need to consider only m predicate symbols, each of whose arity is at most r. Let $C = q_0(\pi_1^0, \ldots, \pi_{r_0}^0) \leftarrow q_1(\pi_1^1, \ldots, \pi_{r_1}^1), \ldots, q_t(\pi_1^t, \ldots, \pi_{r_t}^t)$ be a definite clause in Γ. We may consider only hereditary EFSs that are reduced with respect to some nonempty set $Y \subseteq \Sigma^{[n]}$. Therefore, from Lemma 4.2 we have only to consider definite clauses whose heads contain patterns of length at most n. Since the arity of the predicate symbol of the head is bounded by r, we see that the number of possible heads is at most $m(|\Sigma| + nr)^{nr}$.

Each pattern in the body of a hereditary clause is a subword of some pattern in the head. The number of all the subwords taken from the head is at most $r\frac{n(n-1)}{2}$. Therefore the number of possible atoms in the body is at most $m(r\frac{n(n-1)}{2})^r$. Since we can ignore the order of atoms in the body, the number of possible bodies with the head is at most $2^{m(r\frac{n(n-1)}{2})^r}$. Thus, the number of possible definite clauses is bounded by $m(|\Sigma| + nr)^{nr} 2^{m(r\frac{n(n-1)}{2})^r}$. Hence, the number of such hereditary EFSs is at most $(m(|\Sigma| + nr)^{nr} 2^{m(r\frac{n(n-1)}{2})^r})^m$. Thus,

we see that $\log_2 |\text{H-EFS}(m,*,*,r)^{[n]}|$ is $O(n^{2r})$. By Lemma 4.1, H-EFS$(m,*,*,r)$ is of polynomial dimension. ■

§5 Polynomial-time Learnable Classes of EFSs

In this section, we show a series of polynomial-time learnable classes of EFS languages. In Section 4, we show that the class H-EFS$(m,*,*,r)$ is of polynomial dimension, where m and r are the bounds of the number of clauses and the arity of predicate symbols, respectively. To show a polynomial-time fitting for a class, a polynomial-time algorithm for membership is very useful.

First we investigate the membership problem for hereditary EFS languages. An atom or clause is called *ground* if it contains no variable.

Lemma 5.1
Given a hereditary EFS Γ and a word w, whether $w \in L(\Gamma, p)$ is decidable in $O(m^2|w|^{4k+1}rt)$ time, where Γ consists of m clauses, in each of which the number of variable occurrences in the head, the number of atoms in the body, and the arity of predicate symbols are less than or equal to k, t, and r, respectively.

Proof
We apply a bottom-up algorithm for deciding whether a given word w is in $L(\Gamma, p)$. To derive the provability of $p(w)$ from Γ, we can take a ground clause $C\theta$ by a substitution θ before applying modus ponens whenever we use a clause $C \in \Gamma$, because $p(w)$ is ground and Γ is variable-bounded. Furthermore, the hereditariness guarantees that only substitutions that maps every variable to some subword of w are sufficient to derive $p(w)$. Therefore, we construct the family $\mathcal{G}(w)$ of all ground clauses obtained from Γ by such substitutions. Since the number of subwords of w is at most $\frac{|w|(|w|-1)}{2}$, there are at most $(\frac{|w|(|w|-1)}{2})^k$ substitutions that map all patterns to subwords of w for each clause. Therefore, $|\mathcal{G}(w)| \leq m(\frac{|w|(|w|-1)}{2})^k$. For a clause C and an atom A, we can compute the clause obtained from C and A by modus ponens in $O(|C|+|A|)$ time. The length of any atom in $\mathcal{G}(w)$ is at most $|w|r$. The length of any clause in $\mathcal{G}(w)$ is at most $|w|r(1+t)$. If there is an atom A in $\mathcal{G}(w)$, then we delete all the occurrences of A in the bodies of the clauses in $\mathcal{G}(w)$. This step is done in $O(|\mathcal{G}(w)|(|w|r+|w|r(1+t))) = O(m|w|^{2k+1}rt)$ time. We repeat this process until the atom $p(w)$ is obtained or no new atom is obtained. Since at most $|\mathcal{G}(w)|$ steps are sufficient, the total computing time is $O(m^2|w|^{4k+1}rt)$. ■

Note that the time complexity of the membership problem for all the hereditary EFS languages is **NP**-complete, because the membership problem for pattern languages is **NP**-complete. From the above upper bound of the time complexity, we can see that among the four parameters m,k,t,r of the hereditary EFSs, only the bound k on the number of variable occurrences in the head concerns the hardness of the membership problem. Thus, if we consider k as a constant, the hereditary EFSs as a representation is polynomial-time computable.

Lemma 5.2
There exists a polynomial-time fitting for H-EFS(m,k,t,r) for any $m,k,t,r \geq 0$.

Proof
Let S be a finite set of examples for some concept $c \in$ H-EFS(m,k,t,r). If S_+ is empty, we choose a word $w \in \Sigma^+$ not in S_- and take a hereditary EFS $\Gamma = \{p(w)\}$. Then obviously $L(\Gamma,p)$ is consistent with S. Therefore, we assume that S_+ is not empty.

Let $\mathcal{G}(m,k,t,r,S_+)$ be the set of all pairs (Γ,p) which satisfies the following conditions (1)-(3):

(1) Γ contains at most m hereditary definite clauses such that in each clause the head has at most k variable occurrences, the body has at most t atoms, and each predicate is arity at most r.
(2) For each pattern π in each definite clause in Γ, there is a substitution θ such that $\pi\theta$ is a subword of some positive example in S_+.
(3) Variables are from $\{x_1,\ldots,x_k\}$ and predicate symbols are from $\{p,p_1,\ldots,p_{m-1}\}$. The arity of p is fixed to be one, but we do not fix in advance the arity of p_i for $i=1,\ldots,m-1$. Hence the arity of p_i may differ from one EFS to another.

Claim 1
There exists a pair $(\Gamma,p) \in \mathcal{G}(m,k,t,r,S_+)$ such that the concept $L(\Gamma,p)$ is consistent with S.

Proof
Since S is the set of examples for some concept c in H-EFS(m,k,t,r), there exists a pair (Γ_0,p) with $L(\Gamma_0,p)=c$ which satisfies the condition (1). Without loss of generality, we can assume that Γ_0 is reduced with respect to S_+. Then Γ_0 satisfies the condition (2) by Lemma 4.2. Since Γ_0 is reduced and has at most m clauses, there are at most m distinct predicate symbols in Γ_0. Therefore we can rename the variables and predicate symbols in Γ_0 so that they satisfy the condition (3). ■

Claim 2
$|\mathcal{G}(m,k,t,r,S_+)|$ is bounded by some polynomial in $|S_+|$ and $n = \max\{|w| \mid w \in S_+\}$.

Proof
Let $\Pi(k,S_+)$ be the set of patterns π such that π contains at most k variable occurrences which are from $\{x_1,\ldots,x_k\}$, and $\pi\theta$ is a subword of some $w \in S_+$ for some substitution θ. Then $|\Pi(k,S_+)| \leq \sum_{w\in S_+} ((\frac{|w|(|w|-1)}{2})^{k+1}k!)$. Therefore $|\Pi(k,S_+)|$ is $O(|S_+|n^{2k+2}k!)$. The number of possible heads is at most $m|\Pi(k,S_+)|^r$. The number of possible atoms in the body is at most $m(\frac{n(n-1)}{2}r)^r$, because every pattern in the body is a subword of some pattern in the head. Thus, $|\mathcal{G}(m,k,t,r,S_+)|$ is $O((m(|S_+|n^{2k+2}k!)^r(m(n^2r)^r)^t)^m)$ $= O((m^{t+1}|S_+|^r n^{2kr+2r+2rt} r^{rt}\ (k!)^r)^m)$. When we consider m, k, t, and r as

constants, $|\mathcal{G}(m,k,t,r,S_+)|$ is bounded by a polynomial with respect to $|S_+|$ and n. ■

The polynomial-time algorithm for finding the required hereditary EFS runs as follows: The algorithm enumerates pairs (Γ,p) in $\mathcal{G}(m,k,t,r,S_+)$. Then it checks whether $w \in L(\Gamma,p)$ for $w \in S_+$ and $w' \notin L(\Gamma,p)$ for $w' \in S_-$. By Lemma 5.1 this check is polynomial-time computable. If such a pair is found, the algorithm outputs it as a hypothesis. ■

Theorem 5.1
H-EFS(m,k,t,r) is polynomial-time learnable for any $m,k,t,r \geq 0$.

Proof
From Theorem 4.3, H-EFS(m,k,t,r) is of polynomial dimension. From Lemma 5.2, there exists a polynomial-time fitting for H-EFS(m,k,t,r). Therefore, by Lemma 2.1 (1), H-EFS(m,k,t,r) is polynomial-time learnable. ■

For a concept class $\mathcal{C}$, Blumer et al.[8] discussed the polynomial-time learnability of the *finite union class* of $\mathcal{C}$ that is defined as

$$\mathcal{FU}(\mathcal{C}) = \{c_1 \cup c_2 \cup \cdots \cup c_u \mid c_i \in \mathcal{C}, u \geq 1\}.$$

Though the concept class $\mathcal{FU}$(H-EFS(m,k,t,r)) is not of polynomial dimension, we can prove the following theorem by showing a polynomial-time Occam fitting for this class. The idea of the proof is mostly due to Blumer et al..[8]

From Lemma 5.1, the representation for the concept class H-EFS(m,k,t,r) is polynomial-time computable if we use a conventional representation for EFSs. We can also see that the representation for $\mathcal{FU}$(H-EFS(m,k,t,r)) is polynomial-time computable. In Lemma 5.3, we show that there is a polynomial-time Occam fitting for $\mathcal{FU}$(H-EFS(m,k,t,r)). Then Lemma 2.1 (2) yields polynomial-time learnability.

We use the *weighted set cover problem* and its approximation algorithm *GreedyWSC* due to Chvatal.[9] The weighted set cover problem is, given a collection of finite sets $T_1,\ldots,T_n$ with positive real weights $W_1,\ldots,W_n$, to find $J^* \subseteq I = \{1,\ldots,n\}$ with $\bigcup_{i\in J^*} T_i = \bigcup_{i\in I} T_i$ such that $weight(J^*) = \sum_{i\in J^*} W_i$ is minimized. The algorithm *GreedyWSC* is described in Figure 1:

Lemma 5.3 (Chvatal[9])
For the weighted set cover problem, algorithm *GreedyWSC* runs in polynomial time and produces a set cover $J \subseteq I$ with $weight(J) \leq weight(J^*)\cdot\log|I|$, where J^* is a minimal weighted set cover.

Theorem 5.2
$\mathcal{FU}$(H-EFS(m,k,t,r)) is polynomial-time learnable for any $m,k,t,r \geq 0$.

Proof
The basic idea is due to Blumer et al..[8] By Lemma 2.1 (2), we have only to show a polynomial-time Occam fitting for $\mathcal{FU}$(H-EFS(m,k,t,r)). For a set S of

```
procedure GreedyWSC ( {⟨i, T_i, W_i⟩}_{i∈I} : set of triples ) : subset of I
  begin
    UnCover := ⋃_{i∈I} T_i ;
    J := ∅ ;
    while UnCover ≠ ∅ do
      begin
        Find k ∈ I which minimizes the ratio W_k/|T_k| ;
        J := J ∪ {k} ;
        UnCover := Uncover − T_k ;
        foreach i ∈ I do
          T_i := T_i − T_k
      end
    return J
  end
```

Fig. 1 Algorithm GreedyWSC

```
procedure Occam ( S : set of examples ) : pair of hereditary EFS and predicate
  begin
    J := ∅;    /* instance of the weighted set cover problem */
    foreach (Γ, p) ∈ G(m, k, t, r, S_+)
      if L(Γ, p) ∩ S_− = ∅ then
        J := J ∪ {⟨Γ, L(Γ, p) ∩ S_+, size(Γ)⟩}
    G := GreedyWSC(J) ;
    Γ^∪ = ⋃_{Γ∈G} Γ ;
    return (Γ^∪, p)
  end
```

Fig. 2 Occam Fitting for $\mathcal{FU}$(H-EFS(m, k, t, r))

examples, the algorithm *Occam* in Figure 2 finds in polynomial time a hypothesis which is consistent with S.

Recall that we can generate all pairs in $\mathcal{G}(m, k, t, r, S_+)$ in polynomial time with respect to $|S_+|$ and $\max\{|w| \mid w \in S_+\}$. Moreover, by Lemma 5.3, the size of the output Γ is at most $\log |S|$ times the size of the minimum hereditary EFS which is consistent with S. Therefore the algorithm *Occam* is a polynomial-time Occam fitting for $\mathcal{FU}$(H-EFS(m, k, t, r)). ■

§6 NC-learnability

The notion of **NC**-learnability is introduced by Vitter and Lin.[29] By using **NC** algorithms instead of polynomial-time algorithms, we can develop the same argument as that in Section 2 and obtain a similar result for **NC**-learnability.

The purpose of this section is to show a series of $\mathbf{NC}^2$-learnable subclasses of hereditary EFSs. It is shown in[6] that a **P**-complete set can be described with a hereditary EFS. Therefore, at least, we cannot expect any **NC**-computable representation for H-EFS(m, k, t, r). Hence we are required to get into a new class of EFSs.

Let $|\pi|$ denote the length of a pattern π. For an atom $p(\pi_1, \ldots, \pi_n)$, we define $||p(\pi_1, \ldots, \pi_n)|| = |\pi_1| + \cdots + |\pi_n|$. A definite clause $A \leftarrow A_1, \ldots, A_t$ is called *length-bounded* if $||A\theta|| \geq ||A_1\theta|| + \cdots + ||A_t\theta||$ for any substitution θ.[5] An EFS Γ is *length-bounded* if all definite clauses in Γ are length-bounded. For a length-bounded definite clause, we can easily see that, for each variable x_i in

the clause, the number of occurrences of x_i in the head is not less than that in the body.

Example 6.1
The definite clause $p(ax_1) \leftarrow q(bx_1)$ is length-bounded but not hereditary, while $p(ax_1bc) \leftarrow q(ax_1), r(x_1b)$ is not length-bounded but hereditary. The definite clause $p(ax_1, bx_2, cx_3) \leftarrow q(x_1, x_2, x_3)$ is both length-bounded and hereditary.

Definition 6.1
For $m, k, r \geq 0$, LB-H-EFS(m,k,r) is the class of languages definable by length-bounded hereditary EFSs with at most m definite clauses such that the number of variable occurrences in the head of each clause is bounded by k and the arity of each predicate is at most r.

Obviously the class LB-H-EFS(m,k,r) contains infinitely many languages for any $m, k, r \geq 1$. Any context-free language is in LB-H-EFS(m,2,1) for some $m \geq 1$ and any regular language is in LB-H-EFS(m,1,1) for some $m \geq 1$.

Example 6.2
Example 2.2 shows that the language $\{a^n b^n \mid n \geq 1\}$ is in LB-H-EFS(2,1,1) and that the language $\{a^n b^n c^n \mid n \geq 1\}$ is in LB-H-EFS(3,3,3).

Remark 6.1
Unlike the definition of H-EFS(m, k, t, r), we do not bound the number t of atoms in the body explicitly, when we define the class LB-H-EFS(m,k,r). However, we can assume that the number of atoms in the body is at most k without loss of generality: Let (Γ, p) be a pair of a length-bounded EFS Γ and its predicate symbol of arity one. Assume that the number of variable occurrences in the head of each clause is at most k. For a definite clause $C = A \leftarrow A_1, \ldots, A_t$ in Γ, suppose that an atom A_i does not contain any variables. If A_i is not provable from $\Gamma - \{C\}$, it is not hard to show $\Gamma \not\vdash A_i$ by induction on the number of applications of modus ponens since only modus ponens rules can eliminate A_i from the body of C. Hence the clause C is redundant in Γ, i.e., $L(\Gamma, p) = L(\Gamma - \{C\}, p)$. On the other hand, if A_i is provable from $\Gamma - \{C\}$, then the atom A_i is redundant in the clause C, i.e., $L(\Gamma, p) = L((\Gamma - \{C\}) \cup \{C'\}, p)$, where $C' = A \leftarrow A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_t$. Therefore we can assume that each atom contains at least one variable. From the length-boundedness of Γ, the total number of variable occurrences in the body is bounded by k. Thus the number of atoms in the body is at most k.

Theorem 6.1
LB-H-EFS(m,k,r) is $\mathbf{NC}^2$-learnable for any $m, k, r \geq 0$.

Proof
We first observe that the membership problem for LB-H-EFS(m,k,r) is solvable in $\mathbf{NC}^2$. Consider the following nondeterministic recursive procedure that returns *true* if and only if $\Gamma \vdash q(u_1, \ldots, u_l)$ for a length-bounded hereditary EFS Γ and $u_1, \ldots, u_l \in \Sigma^+$.

We can simulate the procedure *Prove* by a two-way nondeterministic auxil-

```
procedure Prove(q, [u_1, ..., u_l])
begin
  guess a definite clause q(π_1^0, ..., π_n^0) ← q_1(π_1^1, ..., π_{r_1}^1), ..., q_t(π_1^t, ..., π_{r_t}^t) in Γ;
  guess a substitution θ with π_i^0 θ = u_i for all i = 1, ..., l;
  if t = 0 then
    return true
  else if Prove(q_j, [π_1^j θ, ..., π_{r_j}^j θ]) = true for all j = 1, ..., t then
    return true
end
```

Fig. 3 Algorithm for Proving $\Gamma \vdash q(u_1, \ldots, u_l)$

iary pushdown automaton that runs in polynomial time using $O(\log n)$ worktape space.[27] We just state the idea of simulation. We assume that a word w and a pair (Γ, p) are given on the input tape and the total length of the input is n. Since Γ is hereditary, we can assume that $x_h\theta$ is a subword of w for any variable x_h in the guessed clause. Therefore each $x_h\theta$ can be expressed by specifying the start and end positions of $x_h\theta$ in w. This requires only $O(\log n)$ worktape space. Moreover, since π contains at most k variables, we need $O(\log n)$ space to keep θ. Since Γ is on the input tape, guessing nondeterministically a definite clause from Γ requires $O(\log n)$ worktape space. Checking $\pi_i\theta = u_i$ is also possible in $O(\log n)$ worktape space. Recursions are simulated by a pushdown store in a conventional way by pushing $(q_j, [\pi_1^j\theta, \ldots, \pi_{t_j}^j\theta])$ for $j = 1, \ldots, t$, where each $\pi_i^j\theta$ is represented by a pair of binary integers in $O(\log n)$ bits. Since t_j is at most r, $(q_j, [\pi_1^j\theta, \ldots, \pi_{t_j}^j\theta])$ requires $O(\log n)$ space. Then the simulation can be continued using $O(\log n)$ worktape space.

We consider the size of the recursion tree for *Prove*$(p, [w])$, where each node is labeled with some *Prove*$(q, [u_1, \ldots, u_l])$. We can assume that each u_i of *Prove*$(q, [u_1, \ldots, u_l])$ is a subword of w. Since the number of predicates is at most m and the arity of each predicate symbol is bounded by r, the depth of the recursion tree is bounded by some polynomial in $|w|$. Moreover, since Γ is length-bounded, the number of the leaves of this tree is at most $|w|$. Therefore, the number of the nodes in this recursion tree is also bounded by a polynomial in $|w|$. Hence, this nondeterministic algorithm accepts in polynomial time.

It is known in References[19,20] that if a set is accepted by a polynomial-time auxiliary pushdown automaton that uses $O(\log n)$ worktape space then it is in $\mathbf{NC}^2$. Therefore it is possible to check in $\mathbf{NC}^2$ whether $\Gamma \vdash p(w)$. Thus we can decide the membership $w \in L(\Gamma, p)$ in $\mathbf{NC}^2$ when w and (Γ, p) are given as input. Therefore the representation for LB-H-EFS(m,k,r) is computable in $\mathbf{NC}^2$.

We combine this **NC** algorithm for deciding membership with the following **NC** algorithm. We consider how to generate all candidates (Γ, p) consistent with S_+ of positive examples and S_- of negative examples. We define $\mathcal{G}'(m, k, r, S_+)$ in the same way as $\mathcal{G}(m, k, t, r, S_+)$ in Lemma 5.2. From Lemma 4.2, it suffices to deal with $\mathcal{G}'(m, k, r, S_+)$ as the space of candidates. It is not hard to see that we can generate all pairs in $\mathcal{G}'(m, k, r, S_+)$ in $\mathbf{NC}^2$ since we are required to consider only subwords of the words in S_+ as patterns in atoms and m, k, r are constants.

From these $\mathbf{NC}^2$ algorithms, we can see that a pair (Γ, p) in $\mathcal{G}'(m, k, r, S_+)$ which is consistent with examples in S is computable in $\mathbf{NC}^2$. ■

Theorem 6.2
$\mathcal{FU}$(LB-H-EFS(m,k,r)) is **NC** learnable for any $m,k,r \geq 0$.

Proof
We can construct an **NC** Occam fitting for $\mathcal{FU}$(LB-H-EFS(m,k,r)) by using the **NC** approximate algorithm for the weighted set cover problem due to Berger, Rompel and Shor.[7] ■

§7 Conclusion

We showed that the classes H-EFS(m,k,t,r) and $\mathcal{FU}$(H-EFS(m,k,t,r)) are polynomial-time learnable for any constant $m,k,t,r \geq 0$. But if the number k of variable occurrences in the head of each definite clause is not bounded by a fixed constant, even for some small subclasses of H-EFS$(m,*,t,r)$, we can not expect polynomial-time learning. For example, the consistency problem for H-EFS$(1,*,0,1)$, which is the class of pattern languages, is known to be Σ_2^p-complete.[13] We strengthened this observation by showing the **NP**-completeness of the consistency problem for the class defined by taking all unions of m regular pattern languages for any fixed constant $m \geq 1$. As for the number m of the clauses, we know that the class H-EFS$(*,k,t,r)$ is not of polynomial dimension. Note that, however, it does not imply the impossibility of polynomial-time learning, since Occam fitting for the class might exist. Similarly for the arity r of the predicate symbols, the class H-EFS$(m,*,t,*)$ is not of polynomial dimension. We also showed the class H-EFS$(m,*,*,r)$ is of polynomial dimension. For the classes H-EFS$(m,k,*,*)$ and H-EFS$(m,k,t,*)$, it is open whether they are of polynomial dimension.

In our paper,[4] we presented an application of our learning algorithms to protein classification problems, where we used the class of finite unions of regular pattern languages. Unfortunately, the experiments did not have the benefit of hereditariness directly. This is because the running time of our learning algorithm, which is essentially based on an enumeration method, is huge although it remains polynomial. In order to finish the computation in realistic time, we had to restrict the class as described in the paper.[4] Therefore it is an important problem to develop a more efficient learning algorithm which makes good use of the hereditariness.

Acknowledgements

We would like to thank Setsuo Arikawa, Hiroki Arimura, Satoru Kuhara and Shinichi Shimozono for discussions.

References

1) Abe, N., "Polynomial Learnability and Locality of Formal Grammars," in *Proc. 26th Meeting of the Assciation for Computational Linguistics*, 1988.

2) Angluin, D., "Finding Patterns Common to a Set of Strings," *Journal of Computer and System Sciences*, *21*, pp. 46–62, 1980.

3) Angluin, D., "Learning Regular Sets from Queries and Counterexamples," *Information and Computation*, *75*, pp. 87–106, 1987.

4) Arikawa, S., Kuhara, S., Miyano, S., Shinohara, A., and Shinohara, T., "A Learning Algorithm for Elementary Formal Systems and its Experiments on Identification of Transmembrane Domains," in *Proc. 25th Hawaii International Conference on System Sciences, Vol. I*, pp. 675–684, 1992.

5) Arikawa, S., Shinohara, T., and Yamamoto, A., "Learning Elementary Formal Systems," *Theoretical Computer Science*, *95*, pp. 97–113, 1992.

6) Arimura, H., *Personal Communication*, 1993.

7) Berger, B., Rompel, J., and Shor, P., "Efficient NC Algorithms for Set Cover with Application to Learning and Geometry," in *Proc. of the 30th Annual Symposium on Foundations of Computer Science*, pp. 54–59, 1989.

8) Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M., "Learnability and the Vapnik-Chervonenkis Dimension," *JACM*, *36*, *4*, pp. 929–965, 1989.

9) Chvatal, V., "A Greedy Heuristic for the Set Covering Problem," *Mathematics of Operations Research*, *4*, *3*, pp. 233–235, 1979.

10) Gold, E., "Language Identification in the Limit," *Information and Control*, *10*, pp. 447–474, 1967.

11) Haussler, D., Kearns, M., Littlestone, N., and Warmuth, M., "Equivalence of Models for Polynomial Learnability," in *Proc. of the 1st Workshop on Computational Learning Theory*, pp. 34–50, 1988.

12) Jiang, T. and Li, M., "On the Complexity of Learning Strings and Sequences," in *Proc. of the 4th Annual Workshop on Computational Learning Theory*, pp. 367–371, 1991.

13) Ko, K. and Tzeng, W., "Three Σ_2^p-complete Problems in Computational Learning Theory," *Computational Complexity*, *1*, *3*, pp. 269–310, 1991.

14) Maier, D., "The Complexity of Some Problems on Subsequences and Supersequences," *JACM*, *25*, pp. 322–336, 1978.

15) Moriyama, T. and Sato, M., "Properties of Language Classes with Finite Elasticity," in *Proc. 4th International Workshop on Algorithmic Learning Theory (Lecture Notes In Artificial Intelligence 744)*, pp. 187–196, 1993.

16) Mukouchi, Y. and Arikawa, S., "Inductive Inference Machines that can Refute Hypothesis Spaces," in *Proc. 4th International Workshop on Algorithmic Learning Theory (Lecture Notes in Artificial Intelligence 744)*, pp. 123–136, 1993.

17) Natarajan, B., "On Learning Sets and Functions," *Machine Learning*, *4*, *1*, pp. 67–97, 1989.

18) Natarajan, B., *Machine Learning — A Theoretical Approach*, Morgan Kaufmann Publishers, 1991.

19) Ruzzo, W., "Tree-size Bounded Alternation," *Journal of Computer and System Sciences*, *21*, *2*, pp. 218–235, 1980.

20) Ruzzo, W., "On Uniform Circuit Complexity," *Journal of Computer and System Sciences*, *22*, *3*, pp. 365–383, 1981.

21) Sakakibara, Y., "On Learning Smullyan's Elementary Formal Systems: Towards an Efficient Learning for Context-sensitive Languages," *Advances in Software Science and Technology*, *2*, pp. 79–101, 1990.

22) Schapire, R., "Pattern Languages are not Learnable," in *Proceedings of the 3rd Annual Workshop on Computational Learning Theory*, pp. 122–129, 1990.

23) Shinohara, T., "Polynomial Time Inference of Pattern Languages and its Applications," in *Proc. 7th IBM Symp. Math. Found. Comp. Sci.*, pp. 191–209, 1982.

24) Shinohara, T., "Polynomial Time Inference of Extended Regular Pattern Languages," *Lecture Notes in Computer Science*, *147*, pp. 115–127, 1983.

25) Shinohara, T., "Inductive Inference from Positive Data is Powerful," in *Proceedings of the 3rd Annual Workshop on Computational Learning Theory*, pp. 97–110, 1990.

26) Smullyan, R., *Theory of Formal Systems*, Princeton University Press, 1961.

27) Sudborough, I., "On the Tape Complexity of Deterministic Context-free Languages," *JACM*, *25*, *3*, pp. 405–414, 1978.

28) Valiant, L., "A Theory of the Learnable," *CACM*, *27*, *11*, pp. 1134–1142, 1984.

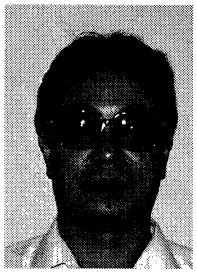
29) Vitter, J. S. and Lin, J.-H., "Learning in Parallel," *Information and Computation*, *96*, pp. 179–202, 1992.

30) Wagner, R. and Fischer, M., "The String-to-string Correction Problem," *JACM*, *21*, pp. 168–73, 1974.

31) Yamamoto, A., "Procedural Semantics and Negative Information of Elementary Formal System," *Journal of Logic Programming*, *13*, *1*, pp. 89–97, 1992.

Satoru Miyano, Dr. Sci.: He is a Professor in Human Genome Center at the University of Tokyo. He obtained B.S. in 1977, M.S. in 1979, and Dr. Sci. degree all in Mathematics from Kyushu University. His current interests include bioinformatics, discovery science, computational complexity, computational learning. He has been organizing Genome Informatics Workshop Series since 1996 and has served for the chair/member of the program committee of many conferences in the area of Computer Science and Bioinformatics. He is on the Editorial Board of Theoretical Computer Science and the Chief Editor of Genome Informatics Series.

Ayumi Shinohara, Dr. Sci.: He is an Associate Professor in the Department of Informatics at Kyushu University. He obtained B.S.in 1988 in Mathematics, M.S. in 1990 in Information Systems, and Dr. Sci. degree in 1994 all from Kyushu University. His current interests include discovery science, bioinformatics, and pattern matching algorithms.



Takeshi Shinohara, Dr. Sci.: He is a Professor in the Department of Artificial Intelligence at Kyushu Institute of Technology. He obtained his B.S. in Mathematics from Kyoto University in 1980, and his Dr. Sci. degree from Kyushu University in 1986. His research interests are in Computational/Algorithmic Learning Theory, Information Retrieval, and Approximate Retrieval of Multimedia Data.