More about Learning Elementary Formal Systems

Arikawa, Setsuo
Research Institute of Fundamental Information Science Kyushu University

Shinohara, Takeshi
Department of Artificial Intelligence Kyushu Institute of Technology

Miyano, Satoru
Research Institute of Fundamental Information Science Kyushu University

Shinohara, Ayumi
Research Institute of Fundamental Information Science Kyushu University

RIFIS-TR-CS-56

RIFIS Technical Report

More about Learning Elementary Formal Systems

Setsuo Arikawa
Takeshi Shinohara
Satoru Miyano
Ayumi Shinohara

March 26, 1992

Research Institute of Fundamental Information Science
Kyushu University 33
Fukuoka 812, Japan

E-mail: miyano@rifis.sci.kyushu-u.ac.jp Phone: 092 (641)1101 Ex. 4471

More about Learning Elementary Formal Systems*

Setsuo ARIKAWA[†]
arikawa@rifis.sci.kyushu-u.ac.jp

Takeshi SHINOHARA[‡]
shino@ai.kyutech.ac.jp

Satoru MIYANO[†]
miyano@rifis.sci.kyushu-u.ac.jp

Ayumi SHINOHARA[†]
ayumi@rifis.sci.kyushu-u.ac.jp

† Research Institute of Fundamental Information Science, Kyushu University 33, Fukuoka 812, Japan.

‡ Department of Artificial Intelligence, Kyushu Institute of Technology, Iizuka 820, Japan.

Abstract

Elementary formal system (EFS for short) is a kind of logic program directly dealing with character strings. In 1989, we proposed the class of variable-bounded EFS's as a unifying framework for language learning. Responding to the proposal, several works have been developed. In this paper, a brief summary of these works on learning elementary formal systems, Shapiro's model inference approach, inductive inference from positive data, Valiant's PAC (probably approximately correct) learning approach, and applications to Molecular Biology, is presented.

* This work is partly supported by Grant-in-Aid for Scientific Research on Priority Areas (No. 03245104), the Ministry of Education, Science and Culture, Japan.

Elementary Formal Systems

The elementary formal systems (EFS's for short) were introduced by Smullyan to develop his recursion theory [24], and it is known that they can be used to define languages as *generators* [4]. Since EFS's can be considered as logic programs [11] over character strings [27], they are also used as *acceptors* for languages. In the field of language learning, many contributions have been made [3], using various frameworks such as regular grammars, context-free and context-sensitive grammars, finite automata, pushdown automata, etc. However, with EFS's we can discuss both generators and acceptors of formal languages, taking full advantage of results obtained in the theory of formal languages and automata as well as those in the theory of logic programs. Thus, we have proposed EFS's as a unifying framework for language learning [5].

The following is an example of EFS which consists of three definite clauses, where a, b, c are constant symbols from an alphabet Σ, x, y, z are variables, and p, q are predicate symbols.

$$\Gamma = \left\{ \begin{array}{l} p(a, b, c) \leftarrow, \\ p(ax, by, cz) \leftarrow p(x, y, z), \\ q(xyz) \leftarrow p(x, y, z) \end{array} \right\}$$

In EFS's we use two inference rules, the substitution of nonempty words for variables, and the modus ponens. The language defined by Γ and q is

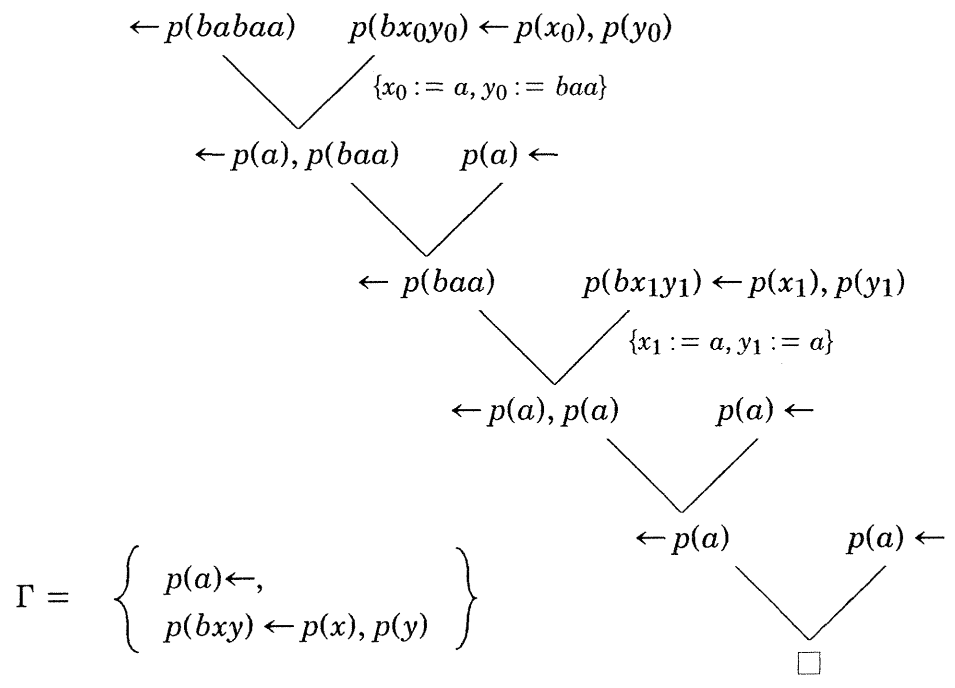
$$L(\Gamma, q) = \{ w \in \Sigma^+ \mid p(w) \text{ is provable from } \Gamma \} = \{ a^n b^n c^n \mid n \geqq 1 \}.$$

In this paper, we demonstrate how efficiently our framework of EFS's works for language learning, by presenting results obtained within the framework.

Variable-Bounded EFS's and Model Inference

A definite clause $A \leftarrow B_1, \ldots, B_n$ is said to be *variable-bounded* if variables in $B_1, \ldots, B_n$ also appear in A. A variable-bounded EFS is a finite set of variable-bounded clauses. The restriction on variable-bounded EFS's does not essentially affect the descriptive power of EFS in the sense that every recursively enumerable language is definable by a variable-bounded EFS. The EFS Γ in the example above is variable-bounded.

For variable-bounded EFS's, derivations based on resolution principle provide a complete refutation procedure [27]. The Figure 1 illustrates a refutation. Note



$\leftarrow p(babaa)$
$p(bx_0y_0) \leftarrow p(x_0), p(y_0)$
$\{x_0 := a, y_0 := baa\}$
$\leftarrow p(a), p(baa)$
$p(a) \leftarrow$
$\leftarrow p(baa)$
$p(bx_1y_1) \leftarrow p(x_1), p(y_1)$
$\{x_1 := a, y_1 := a\}$
$\leftarrow p(a), p(a)$
$p(a) \leftarrow$
$\leftarrow p(a)$
$p(a) \leftarrow$
$\square$
$\Gamma = \left\{ \begin{array}{l} p(a) \leftarrow, \\ p(bxy) \leftarrow p(x), p(y) \end{array} \right\}$

Figure 1. An EFS Γ and a refutation

that the number of unifiers for an atom and a ground atom is finite while it may be infinite for two atoms in general. To decide whether a word w is in the language defined by a variable-bounded EFS, only ground goals of the form $\leftarrow p(w)$ should be refuted. As far as variable-bounded EFS's are concerned, any subgoal appearing in a derivation from a ground goal is always ground.

Shapiro's Model Inference System (MIS for short) [16] synthesizes logic programs form their models. One of the most important features of MIS is that it fully utilizes the tight relation between syntax and semantics of logic programs. Since variable-bounded EFS's have nearly the same properties as usual logic programs, we can naturally consider a learning algorithm based on Shapiro's theory of model inference.

By using the following procedure MIEFS (Model Inference for EFS), variable-bounded EFS's can be synthesized from their models in a similar manner to MIS. The hypothesis H is *too strong* if H proves some negative example. H is *too weak* if H cannot prove some positive example.

To guarantee MIFES correctly infers EFS models, it is sufficient that CBA works for EFS's and a complete refinement operator is available. CBA detects a false clause from an EFS using a refutation of a false fact in the model. Refinement operator provides an effective enumeration of clauses in the direction from general to specific. For more details the reader should be referred to [5].

```
Procedure MIEFS;
begin
    $H := \{\square\}$;
    repeat
        read next example;
        while $H$ is too strong or too weak do begin
            while $H$ is too strong do begin
                apply CBA to $H$ and detect a false clause $C$ in $H$;
                delete $C$ from $H$;
            end;
            while $H$ is too weak do
                add a refinement of a clause deleted so far to $H$;
        end;
        output $H$;
    forever;
end;
```

Length-Bounded EFS's and Inferability from Positive Data

A definite clause $A \leftarrow B_1, \dots, B_n$ is called *length-bounded* if the total length of $B_1\theta, \dots, B_n\theta$ does not exceed the length of $A\theta$ for any substitution θ, where the length of an atom is total length of terms in it. Both of two examples above are length-bounded. The class of languages definable by length-bounded EFS's coincides with the class of context-sensitive languages [5].

Inductive inference is a process to guess an unknown language or rule from its positive and negative examples, and it is said to be *successful* if the sequence of guesses produced by the process converges to a correct representation of the target. Examples contained in the language or explained by the rule is called *positive*, and the others *negative*.

Gold [9] indicated that inductive inference from positive data is strictly less powerful than that from positive and negative data, by proving any *super-finite* class, that includes all finite languages and at least one infinite language, is not possible to be inferred from positive data. Angluin studied inductive inference of languages from positive data and gave a useful necessary and sufficient condition for languages to be inferable from positive data [1, 2]. The class of pattern languages is one of the most important classes shown by her to be inferable from positive data. A *pattern* is a string consisting of constant symbols and variables. The language of a pattern π is a set of constant strings obtained by substituting nonempty constant strings for variables. For example, $\pi = axbx$ is a pattern and defines $L(\pi) = \{ awbw \mid w \in \Sigma^+ \}$. Obviously, any pattern language can be defined by such an EFS as $\{ p(axbx) \leftarrow \}$. Thus, the class of languages defined by EFS's is a natural extension of pattern languages.

Shinohara showed that the class of unions of two pattern languages and the class of languages defined by EFS's with just two clauses are inferable from positive data [20, 21]. Wright proved that the unions of three or more pattern languages are also inferable by using a notion of finite elasticity, which is a sufficient condition for inferability from positive data [26, 13].

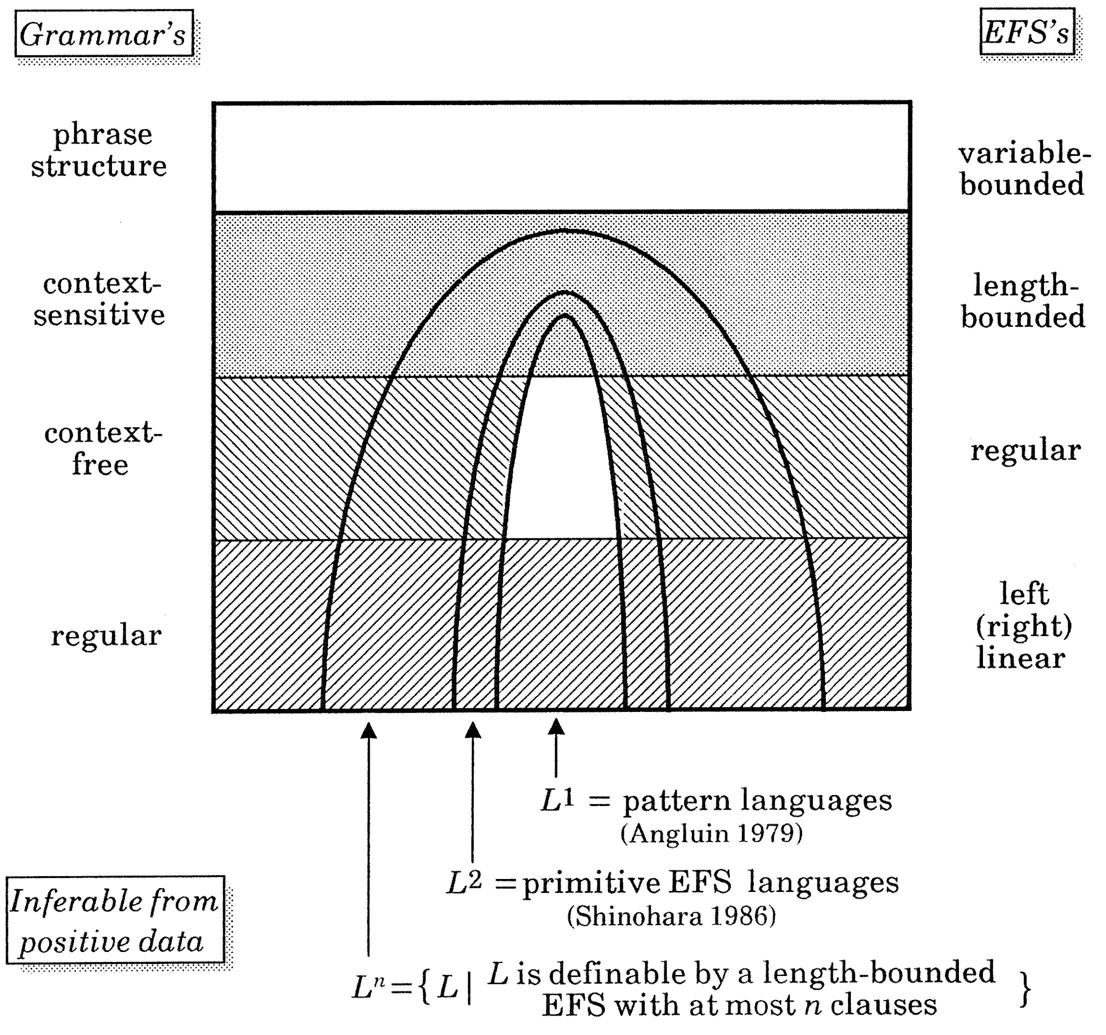
Using the framework of EFS's more explicitly, we can show that any class of minimal models or languages defined by length-bounded EFS's consisting of at most m clauses is inferable from positive data, for an arbitrarily fixed m [22]. One of the most important properties of length-bounded EFS's is that there are only finitely many inequivalent EFS's that explain a given finite set and do not contain any redundant clauses. This property is also used to prove PAC learnability, which will be discussed later. It should be noted that preventing substitutions from erasing variables is essential for our discussion on length-bounded EFS's. If we allow erasing substitutions, we need another discussion as in [19].

From this result, we can show the same results for grammars and Prolog programs [23] as for EFS's. That is, the class of languages defined by context-sensitive grammars with at most m production rules is inferable from positive data, and so is the class of linear Prolog programs [17] (or reducing programs [7]) with at most m clauses.

Figure 2 illustrates the Chomsky hierarchy and EFS languages. A clause is called *regular* if it is of the form $p(\pi) \leftarrow q_1(x_1), \dots , q_n(x_n)$, where $p, q_1, \dots , q_n$ are unary predicates, π is a regular pattern containing variables $x_1, \dots , x_n$. A regular clause is called *left* (*right*) *linear* if the pattern π in the head is of the form wx (xw), where x is a variable and w is a string.

Hereditary EFS's and PAC Learnability

We can also discuss Valiant's PAC (probably approximately correct) [25] learnability of languages by using the framework of EFS's. We call a subset of a universe U a *concept*. When we take the set Σ^* as the universe, a concept is a



Grammar's
EFS's
phrase structure
context-sensitive
context-free
regular
variable-bounded
length-bounded
regular
left (right) linear
L^1 = pattern languages (Angluin 1979)
L^2 = primitive EFS languages (Shinohara 1986)
$L^n = \{ L \mid L$ is definable by a length-bounded EFS with at most n clauses $\}$
Inferable from positive data

Figure 2. Chomsky hierarchy and EFS languages

language. By $U^{\leqq n}$, we denote the set of elements in the universe U whose sizes are at most n. A class of concepts is *polynomial time learnable* if there exists an algorithm A that satisfies the following conditions:

(1) A runs in polynomial time with respect to the length of the input.

(2) There exists a polynomial $p(\cdot, \cdot, \cdot)$ such that for any integer $n \geqq 0$, any concept c, any real number ε, δ $(0 < \varepsilon, \delta < 1)$, and any probability distribution P on $U^{\leqq n}$, whenever A takes $p(n, 1/\varepsilon, 1/\delta)$ examples randomly chosen according to P, A produces a hypothesis h such that $P(c \oplus h) < \varepsilon$ with probability at least $1 - \delta$, where $\oplus$ means a symmetric difference.

For a class C and an integer n, we define $\dim C_n = \log_2 |C_n|$, where $C_n = \{ c \cap U^{\leqq n} \mid c \in C \}$. A class C is of *polynomial dimension* if there exists a polynomial $d(n)$ such that $\dim C_n \leqq d(n)$ for all n. It is known that a class C is polynomial time learnable only if C is of polynomial dimension [14]. As stated above, the class of languages defined by length-bounded EFS's with at most m clauses has the property that there are only finitely many languages containing a finite set of strings. More precisely, it can be easily shown the class is of polynomial dimension. If we do not restrict the number of clauses in EFS's, the dimension of the class clearly exceeds polynomial, because any finite set can be defined.

A pattern is *regular* [18] if each variable appears at most once in it. In [12], we showed that even a class of regular pattern languages, which is the simplest class of EFS languages, is not polynomial time learnable under the assumption NP $\neq$ RP. This result indicates that it is natural to put the bound on the number of variable occurrences. In fact, for any (possibly not regular) pattern with at most k variable occurrences, we can construct a polynomial time learning algorithm in a straightforward way. When we restrict the number of variable occurrences, we can show that more general classes of EFS's are polynomial time learnable.

A clause is *hereditary* if any pattern in the body contains at least one variable and is a subword of a pattern in the head. The examples we have shown so far are all hereditary. It should be noticed that in hereditary EFS's any proof or refutation of a ground goal $\leftarrow p(w)$ does not contain words other than subwords of w. We can prove that the class LB-H-EFS(m, k) of languages defined by length-bounded hereditary EFS's consisting of at most m clauses such that the number of variable occurrences does not exceed k is polynomial time learnable for arbitrarily fixed m and k.

The class LB-H-EFS(m, k) is not very large but even LB-H-EFS(2, 2) contains infinitely many languages such as $\{ a^{2^n} \mid n \geqq 1 \}$, $\{ a^n b^n \mid n \geqq 1 \}$, unions of two pattern languages with at most 2 variable occurrences, and so on. Any context-free language is contained in LB-H-EFS(m, 2) for some m. Hence, our result shows that the range of polynomial time learnable is large enough if m and k are appropriately chosen.

Applications to Molecular Biology

One of the most important objectives in learning theory is to develop applications of theoretical results to practical problems. Recently, we have proposed learning algorithms for EFS's as methods for analyzing amino acid sequences and nucleotide sequences in Molecular Biology [6]. We made experiments on identification of transmembrane domain in amino acid sequences with EFS's. Because of the limitations on computational resources, we restricted candidate hypotheses to EFS's consisting of several atoms with regular patterns.

That is, we tried to identify the feature of transmembrane domains by a union of several regular pattern languages. From not so many positive and negative examples randomly chosen from PIR database [15], our algorithm found a reasonable hypothesis in the sense that it explains more than 90% of all examples from PIR.

Strictly speaking, however, the algorithm used in the experiments is not based on the theoretical results discussed above, but on the theory of Occum algorithms [8]. The following is a sketch of the implemented algorithm:

```
Procedure FIND_UNION;
input: $Pos$, $Neg$; { positive and negative examples }
begin
    $S := \varnothing$;
    $I := Pos$;
    $E := Neg$;
    foreach pattern $\pi$ with $\pi\theta = w$ for some $w \in I$ and $\theta$ do
        if $\pi$ excludes almost all examples in $E$ then $S := S \cup \{\pi\}$;
    Find a minimal subset $\Gamma$ of $S$ covering $I$;
    output $\Gamma$;
end;
```

The problem of finding minimal set cover is NP-complete, but can be solved in polynomial time when an approximation is allowed. According to [10], we can find a set cover of size at most $M \log M$ in polynomial time, when M is the size of minimum set cover.

The identification of transmembrane domains is one of the most important protein classification problems. Most approaches in Molecular Biology have been dealing with only positive examples and have not achieved success rates more than 80%. Figure 3 presents an example of amino acid sequence containing four transmembrane domains, which are indicated by bold face letters.

Based on the fact that the lengths of transmembrane domains vary from 20 to 30, we use sequences consisting of transmembrane domains as positive examples and choose sequences of length around 30 randomly cut from the other parts as negative examples. The numbers of positive and negative examples from PIR are 689 and 19276, respectively. To reduce the hypotheses space, we transform sequences according to *hydropathy plot* symbol by symbol, as shown in Table 1. Figure 4. shows the transformed sequence. Further, we restrict the forms of patterns to xw_1z, xw_1yw_2z and $xw_1y_1w_2y_2w_3z$, where w_1, w_2, w_3 are nonempty strings over {*, +, -}, x, y, y_1, y_2, z are distinct variables, and x and z are allowed to be substituted by empty string. Table 2 gives a collection of regular patterns

```
MDVVNQLVAGGQFRVVKE(PLGFVKVLQWVFAIFAFATCGSY)TGELRLSVECANKTESALNIEVEFEYPFRLHQVYFDA
PSCVKGGTTKIFLVGDYSSSAE(FFVTVAVFAFLYSMGALATYIFL)QNKYRENNK(GPMMDFLATAVFAFMWLVSSSAW
A)KGLSDVKMATDPENIIKEMPMCRQTGNTCKELRDPVTS(GLNTSVVFGFLNLVLWVGNLWFVF)KETGWAAPFMRAPP
GAPEKQPAPGDAYGDAGYGQGPGGYGPQDSYGPQGGYQPDYGQPASGGGGYGPQGDYGQQGYGQQGAPTSFSNQM
```

Figure 3. An amino acid sequence from a membrane protein

Amino Acids	Hydropathy	New Symbol
A M C F L V I	1.8 ~ 4.5	*
P Y W S T G	−1.6 ~ −0.4	+
R K D E N Q H	−4.5 ~ −3.2	-

Table 1. Transformation rules

```
*-**--***++-*-**--(+*+**-**-+********+*+++)++-*-*+*-**--+-+**-*-*-*-++*-*--*+*-*
++**-++++-****+-++++*-(***+*******++*+***++***)---+-----(++**-***+******+**+++*+
*)-+*+-*-**+-+--**--*+**--++-+*--*--+*++(+*-++***+**-***+*+-*+***)--+++**+**-*++
+*+---+*++-*++-*+++-+++++++--++++-+++-+-++-+*++++++++-+-++--+++--+*+++*+--*
```

Figure 4. The sequence obtained by the transformation

Patterns	Positive		Negative	
X******Z	403	(58.5%)	1050	(5.5%)
$X{*}{+}{*}{*}Y_1{+}{*}Y_2{*}{*}{+}Z$	273	(39.6%)	1209	(6.3%)
X++***Y**+Z	153	(22.2%)	700	(3.6%)
$X{+}{*}{+}Y_1{*}{+}{*}Y_2{+}{*}{*}Z$	130	(18.9%)	1019	(5.3%)
$X{*}{*}{-}{*}{*}Y_1{*}Y_2{*}{*}{+}Z$	58	(8.4%)	729	(3.8%)
X*-Y***+**Z	75	(10.9%)	674	(3.5%)
$X{*}{*}Y_1{*}{+}{*}{+}{*}Y_2{*}{+}Z$	94	(13.6%)	687	(3.6%)
total	625	(90.7%)	4367	(22.7%)

Table 2. Collections of regular patterns

Patterns	Positive		Negative	
$X{*}{-}Y_1{-}{-}Y_2{-}Z$	12	(1.7%)	13669	(71.0%)
X++Y+--+Z	21	(3.0%)	6094	(31.6%)
$X{*}{*}Y_1{+}{-}Y_2{+}{-}Z$	21	(3.0%)	9482	(49.2%)
total	43	(6.2%)	17340	(90.0%)

Table 3. Collections of regular patterns for non-transmembrane domains

found by our learning algorithm FIND_UNION from 70 positive examples and 100 negative examples, and their success rates evaluated by all positive and negative examples.

The results from experiments have some problems that they achieve high success rates for positive data (90.7%) but relatively low for negative (77.3%). However, when we try to find patterns from the reverse side, that is, find patterns explaining negative examples, our learning algorithm can find more reasonable patterns. The result for non-transmembrane domains in Table 3 shows that both positive and negative examples are explained by only three patterns in high success ratio (93.8% and 90.0%).

References

[1] Angluin, D.: Finding common patterns to a set of strings, Proc. 11th Annual ACM Symp. Theory of Computing, 130 - 141, 1979.

[2] Angluin, D.: Inductive inference of formal languages from positive data, Inform. Contr., **45**, 117 - 135, 1980.

[3] Angluin, D. and Smith, C. H.: Inductive inference: Theory and methods, Computing Surveys, **15**, 237 - 269, 1983.

[4] Arikawa, S.: Elementary formal systems and formal languages - simple formal systems, Memoirs of Fac. Sci., Kyushu Univ. Ser. A, Math., **24**, 47 - 75, 1970.

[5] Arikawa, S., Shinohara, T. and Yamamoto, A.: Elementary formal system as a unifying framework for language learning, Proc. 2nd Workshop Comput. Learning Theory, 312 - 327, 1989.

[6] 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, Proc. 25th Hawaii International Conference on System Sciences).

[7] Arimura, H.: Completeness of depth-bounded resolution in logic programming, Proc. 6th Conf, Japan Soc. Software Sci. Tech., 61 - 64, 1989.

[8] Blumer, A., Ehrenfeucht, A., Haussler, D. and Warmuth, M.: Occam's razor, Information Processing Letters, **24**, 377 - 380, 1987.

[9] Gold, E.M.: Language Identification in the Limit, Inf. & Contr., **10**, 447 - 474, 1967.

[10] Johnson, D.S.: Approximation algorithms for combinatorial problems, JCSS **9**, 256 - 278, 1974.

[11] Lloyd, J.W.: *Foundations of Logic Programming*, Springer - Verlag, 1984.

[12] Miyano, S., Shinohara, A. and Shinohara, T.: Which classes of elementary formal systems are polynomial-time learnable ?, Proc. Algorithmic Learning Theory, 139 - 150, 1991.

[13] Motoki, T., Shinohara, T. and Wright, K.:The correct definition of finite elasticity: Corrigendum to identification of unions, Proc. 4th Workshop Comput. Learning Theory, 375, 1991.

[14] Natarajan, B.K.: On learning sets and functions, Machine Learning, 4, 67 - 97, 1989.

[15] Protein Identification Resource, National Biomedical Research Foundation.

[16] Shapiro, E.Y.: Inductive Inference of Theories From Facts, YALEU / DCS / TR -192, 1981.

[17] Shapiro, E.Y.: Alternation and the computational complexity of logic programs, J. Logic Program., 1, 19 - 33, 1984.

[18] Shinohara, T.: Polynomial time inference of pattern languages and its applications, Proc. 7th IBM Symposium on Mathematical Foundations of Computer Science, 191 - 209, 1982.

[19] Shinohara, T.: Polynomial time inference of extended regular pattern languages, LNCS, **147**, 115 - 127, 1983.

[20] Shinohara, T.: Inferring unions of two pattern languages, Bull. Inf. Cybern., **20**, 83 - 88, 1983.

[21] Shinohara, T.: Inductive inference of formal systems from positive data, Bull. Inf. Cybern., **22**, 9 - 18, 1986.

[22] Shinohara, T.: Inductive inference from positive data is powerful, Proc. 3rd Workshop Comput. Learning Theory, 97 - 110, 1990 (to appear in Information and Computation as "Rich classes inferable from positive data: length-bounded elementary formal systems").

[23] Shinohara, T.: Inductive inference of monotonic formal systems from positive data, New Gener. Comput., **8**, 371 - 384, 1991.

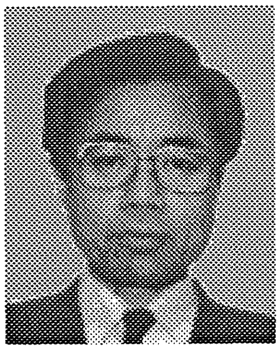
[24] Smullyan, R. M.: *Theory of Formal Systems,* Princeton Univ. Press, 1961.

[25] Valiant, L.G.: A theory of learnable, CACM, **27**, 1134 - 1142, 1984.

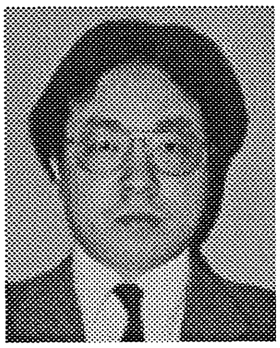
[26] Wright, K.: Identification of unions of languages drawn from an identifiable class, Proc. 2nd Workshop Comput. Learning Theory, 328 - 333, 1989.

[27] Yamamoto, A.: Elementary formal system as a logic programming language, Proc. Logic Program. Conf. '89, ICOT, 123 - 132, 1989.

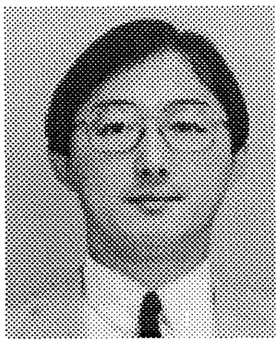
About the Authors



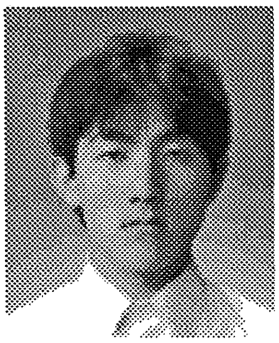
Setsuo Arikawa (有川　節夫) was born in Kagoshima on April 29, 1941. He received the B.S. degree in 1964, the M.S. degree in 1966 and the Dr.Sci. degree in 1969 all in Mathematics from Kyushu University. Presently, he is Professor of Research Institute of Fundamental Inforamtion Science, Kyushu University. His research interests include algorithmic learning theory, logic and inference in AI, and information retrieval systems.



Takeshi Shinohara (篠原　武) was born in Fukuoka on January 23, 1955. He received the B.S. in 1980 from Kyoto University, and the M.S. degree and the Dr. Sci. from Kyushu University in 1982, 1986, respectively. Currently, he is an Associate Professor of Department of Artificial Intelligence, Kyushu Institute of Technology. His present interests include information retrieval, string pattern matching algorithms and computational learning theory.



Satoru Miyano (宮野　悟) was born in Oita on December 5, 1954. He received the B.S. in 1977, the M.S. degree in 1979 and the Dr. Sci. in 1984 all in Mathematics from Kyushu University. Presently, he is an Associate Professor of Research Institute of Fundamental Information Science, Kyushu University. His present interests include parallel algorithms, computational complexity and computational learning theory.



Ayumi Shinohara (篠原　歩) was born in Fukuoka on July 18, 1965. He received the B.S. degree in 1988 in Mathematics and the M.S degree in 1990 in Information Systems from Kyushu University. Presently, he is an Assistant of Research Institute of Fundamental Information Science, Kyushu University. His research interests are computational learning theory and algorithms.

Research Institute of Fundamental Information Scinece, Kyushu University 33, Fukuoka 812, Japan.