A Learning Algorithm for Elementary Formal Systems and its Experiments on Identification of Transmembrane Domains

Arikawa, Setsuo
Research Institute of Fundamental Information Science Kyushu University

Kuhara, Satoru
Graduate School of Genetic Resources Technology, Kyushu University

Miyano, Satoru
Research Institute of Fundamental Information Science Kyushu University

Shinohara, Ayumi
Department of Artificial Intelligence, Kyushu Institute of Technology | Research Institute of Fundamental Information Science Kyushu University

他

出版情報：RIFIS Technical Report. 40, 1991-06-05. Research Institute of Fundamental Information Science, Kyushu University
バージョン：
権利関係：

RIFIS-TR-CS-40

RIFIS Technical Report

A Learning Algorithm for Elementary Formal Systems and its Experiments on Identification of Transmembrane Domains

Setsuo Arikawa
Satoru Kuhara
Satoru Miyano
Ayumi Shinohara
Takeshi Shinohara

June 5, 1991

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

A Learning Algorithm for Elementary Formal Systems and its Experiments on Identification of Transmembrane Domains

Setsuo Arikawa† Satoru Kuhara‡ Satoru Miyano†
Ayumi Shinohara† Takeshi Shinohara*

† Research Institute of Fundamental Information Science, Kyushu University 33, Fukuoka 812, Japan.
‡ Graduate School of Genetic Resources Technology, Kyushu University 46, Fukuoka 812, Japan.
* Department of Artificial Intelligence, Kyushu Institute of Technology Iizuka 820, Japan.

Abstract

We propose a method for algorithmic learning of transmembrane domains based on elementary formal systems. An elementary formal system (EFS, for short) is a kind of a logic program consisting of if-then rules. With this framework, we have implemented the algorithm for identifying transmembrane domains in amino acid sequences. Because of the limitations on computational resources, we restrict candidate hypotheses to EFSs defined by collections of regular patterns. From 70 transmembrane sequences and a similar amount of negative examples which are not transmembrane sequences, our algorithm has produced several reasonable hypotheses of small size. Experiments with the database PIR show that one of them recognizes 95% of 689 transmembrane sequences and 95% of 19256 negative examples which consist of non-transmembrane sequences of length around 30 randomly chosen from PIR.

1 Introduction

In [10] we have shown that some subclasses of elementary formal systems are polynomial-time PAC-learnable [3, 12, 21]. An elementary formal system (EFS, for short) introduced by [20] is a kind of a logic program consisting of if-then rules that can define a set of words. Its descriptive ability is quite large even in some restricted cases [2] and its semantics has been thoroughly studied [23]. EFSs also have received considerable attentions from inductive inference [2, 18].

In this paper we apply the learning algorithm for EFS to identification of transmembrane domains in amino acid sequences. Given positive and negative examples, the algorithm finds a hypothesis represented by an EFS of a specified form. Since an EFS has a structure of logic program, the resulting hypothesis explains the given examples reasonably in the sense that what kind of facts and rules are used to produce given positive examples while excluding negative examples.

Smith and Smith [19] present an efficient method for extracting patterns from amino acid sequences. In contrast, our approach using EFSs has the following features. First, we use both positive and negative data while the approach in [19] is to find information common to positive data. Second, EFSs include patterns or regular expressions as special instances and have much expressive ability. EFSs may give a chance to find unknown motifs which can not be described with regular expressions. Therefore the learnability of EFSs from positive and negative examples is an important problem to be discussed.

We have implemented the basic idea in [10] as a learning algorithm and made some computational experiments using positive and negative data. Based on the accepted assumption that membrane proteins contain transmembrane domains, we consider the prob-

lem of identifying them. The algorithm in [10] runs in polynomial time for some subclasses of EFSs. But, unfortunately, the running time is beyond practical use. However, by restricting a target hypothesis to an EFS consisting of regular patterns [16, 17] of special forms, our implemented algorithm can cope with this problem in reasonable time and space. As positive training examples, we use 70 transmembrane domain sequences. Negative training examples are 100 sequences of length around 30 which are not transmembrane sequences. The strategy of our algorithm is to find a hypothesis that covers positive examples and excludes negative examples. It has found some hypotheses consisting of just at most seven regular patterns with a few variables. We examined these hypotheses by using already known membrane protein data from PIR database [13]. By experiments on 689 positive and 19256 negative examples, they cover 90% transmembrane domain sequences and excludes 75% $\sim$ 77% negative examples. The accuracy rate for positive examples is acceptable but that for negative examples may be unacceptable.

In [1] we pointed out the importance of negative examples. Inspired by the results in [1], we changed the strategy in a way that the algorithm searches a hypothesis which covers negative examples and excludes positive examples. As a result, we have found a single regular pattern that can recognize both positive and negative examples with accuracy more than 95% although transmembrane domains have seemed to be difficult to define as a simple expression when the view point was focussed on positive examples.

2 Elementary formal systems and learnability

Let Σ be a finite alphabet, $X = \{x, y, z, x_1, x_2, \cdots\}$ a set of variables, and $\Pi = \{p, q, r, s, p_1, p_2, \cdots\}$ a set of predicate symbols. We assume that Σ, X, and Π are mutually disjoint. Let Σ^* be the set of all words, Σ^+ the set of all nonempty words, $\Sigma^{\leq n}$ the set of all words of length n or less for $n \geq 0$ over Σ.

A *pattern* is an element of $(\Sigma \cup X)^+$. A pattern π is called *regular* if each variable in π occurs exactly once in π. An *atom* is an expression of the form $p(\tau_1, \ldots, \tau_n)$, where p is a predicate symbol in Π with arity n and $\tau_1, \ldots, \tau_n$ are patterns. A *definite clause* is a clause of the form

$$A \leftarrow B_1, \ldots, B_m,$$

where $m \geq 0$ and $A, B_1, \ldots, B_m$ are atoms. The atom A is called the *head* and the part $B_1, \ldots B_m$ the *body* of the definite clause. An *elementary formal system* (EFS) is a finite set of definite clauses.

Example 1. Consider the following EFS with $\Sigma = \{a, b\}$:

$$\Gamma = \left\{ \begin{array}{l} p(x) \leftarrow q(x), r(x) \\ q(ax) \leftarrow q(bx) \\ q(bbxby) \leftarrow \\ r(xaxb) \leftarrow \end{array} \right\}.$$

The pattern $bbxby$ is regular but $xaxb$ is not. In the definite clause $p(x) \leftarrow q(x), r(x)$, the head is the atom $p(x)$ and the atoms $q(x)$ and $r(x)$ form the body.

A *substitution* θ is a homomorphism from patterns to patterns such that $\theta(a) = a$ for each symbol $a \in \Sigma$. A substitution which maps some variables to the empty string ε is called an *ε-substitution*, and it is not allowed without extra notice. By $\pi\theta$ we denote the image of a pattern π by a substitution θ. For an atom $A = p(\pi_1, \ldots, \pi_n)$ and a definite clause $C = A \leftarrow B_1, \ldots, B_m$ we define $A\theta = p(\pi_1\theta, \ldots, \pi_n\theta)$ and $C\theta = A\theta \leftarrow B_1\theta, \ldots, B_m\theta$.

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, we define the relation $\Gamma \vdash C$ 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 $\Gamma \vdash A \leftarrow B_1, \ldots, B_m, B_{m+1}$ and $\Gamma \vdash B_{m+1}$ then $\Gamma \vdash A \leftarrow B_1, \ldots, B_m$

For $p \in \Pi$ with arity one, we define $L(\Gamma, p) = \{w \in \Sigma^+ \mid \Gamma \vdash p(w) \leftarrow\}$. A language $L \subseteq \Sigma^+$ is *definable by EFS* or an *EFS language* if such Γ and p exist. For a pattern π, the pattern language $L(\pi)$ is the set $\{w \in \Sigma^+ \mid w = \pi\theta$ for some $\theta\}$. It should be noticed that a pattern language $L(\pi)$ is an EFS language $L(\Gamma, p)$ with $\Gamma = \{p(\pi) \leftarrow\}$.

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 $A \leftarrow B_1, \ldots B_m$ is *length-bounded* if $||A\theta|| \geq ||B_1\theta|| + \cdots + ||B_m\theta||$ for any substitution θ. It is known that a definite clause is length-bounded if and only if $||A|| \geq ||B_1|| + \cdots + ||B_m||$ and, for each variable x_i, the number of occurrences of x_i in the body is not greater than that in the head. For example, the definite clause $q(bx) \leftarrow q(ax)$ is length-bounded, while $p(x) \leftarrow q(x), r(x)$ is not. An EFS Γ is *length-bounded* if all definite clauses in Γ are length-bounded.

We say that a definite clause

$$q(\pi_1, ..., \pi_n) \leftarrow q_1(\tau_1, ..., \tau_{t_1}), ..., \ldots, q_l(\tau_{t_{l-1}+1}, ..., \tau_{t_l})$$

is *hereditary* if, for each $j = 1, ..., t_l$, pattern τ_j contains at least one variable and τ_j is a subword of some π_i. An EFS Γ is *hereditary* if each definite clause in Γ is hereditary. For example, the definite clause $q(bx) \leftarrow q(ax)$ is not hereditary. For $m, k \geq 1$, LB-H-EFS(m,k) denotes 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. Obviously the class LB-H-EFS(m,k) contains infinitely many languages for any m and k. Any context-free language is in LB-H-EFS(m,2) for some $m \geq 1$ and any regular language is in LB-H-EFS(m,1) for some $m \geq 1$. Moreover, LB-H-EFS(m,k) contains union of m pattern languages which are definable by patterns with at most k variable occurrences.

The following are examples of languages and length-bounded hereditary EFSs which define them.

Example 2. $\{a^n b^n \mid n \geq 1\} \in$ LB-H-EFS(2,1).

$$\Gamma = \left\{ \begin{array}{l} p(axb) \leftarrow p(x) \\ p(ab) \leftarrow \end{array} \right\}.$$

Example 3. $\{a^n b^n c^n \mid n \geq 1\} \in$ LB-H-EFS(3,3).

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

We call a subset c of Σ^* a *concept*. A concept c can be regarded as a function $c : \Sigma^* \rightarrow \{0, 1\}$, where $c(x) = 1$ implies x is in the concept and $c(x) = 0$ otherwise. A *concept class* is a nonempty set $\mathcal{C} \subseteq 2^{\Sigma^*}$ of concepts. An *example* of a concept c is a pair $\langle x, c(x)\rangle$ for $x \in \Sigma^*$.

A concept class $\mathcal{C}$ is *polynomial-time learnable* [3, 12] if there exists an algorithm $\mathcal{A}$ which satisfies the following conditions:

(1) $\mathcal{A}$ 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 $n \geq 0$, any concept $c \in \mathcal{C}$, any real number ε, δ $(0 < \varepsilon, \delta < 1)$, and any probability distribution P on $\Sigma^{\leq n}$, if $\mathcal{A}$ takes $p(n, \frac{1}{\varepsilon}, \frac{1}{\delta})$ examples which are generated randomly according to

P, then $\mathcal{A}$ outputs a representation of a hypothesis h such that $P(c \oplus h) < \varepsilon$ with probability at least $1 - \delta$.

Theorem 1. [10] LB-H-EFS(m,k) is polynomial-time learnable for any $m, k \geq 1$.

Proof. It has been shown [3, 12] that a concept class $\mathcal{C}$ is polynomial-time learnable if and only if $\mathcal{C}$ is of polynomial dimension [11, 12] and there is a randomized polynomial-time hypothesis finder [3] for $\mathcal{C}$.

First we show that LB-EFS(m) is of polynomial dimension for any $m \geq 1$. Let LB-EFS$(m)_n = \{L \cap \Sigma^{\leq n} \mid L \in \text{LB-EFS}(m)\}$ for $n \geq 0$. We evaluate the cardinality of LB-EFS$(m)_n$. Let Γ and p be a length-bounded EFS and a predicate symbol with arity one defining a language $L(\Gamma, p)$. Let $C = A \leftarrow B_1, \ldots, B_l$ be a definite clause in Γ. Note that if $||A|| > n$ then $L(\Gamma, p) \cap \Sigma^{\leq n} = L(\Gamma - \{C\}, p) \cap \Sigma^{\leq n}$. Therefore we need to consider only length-bounded EFSs consisting of at most m definite clauses whose heads are of length at most n. We also have $l \leq n$. The number of such length-bounded EFSs is roughly bounded by $m^{(n+1)m}(|\Sigma|+n)^{(2n^2+n-1)m}$. Hence dimLB-EFS$(m)_n$ is $O(n^2 \log n)$. Thus the dimension of LB-EFS(m) is polynomial.

Now it suffices to show that there is a polynomial-time hypothesis finder for LB-H-EFS(m, k). Let S be a set of positive examples and T be a set of negative examples. If S is empty, we choose a word $w \in \Sigma^+$ not in T and take a length-bounded hereditary EFS consisting of a single definite clause $p(w) \leftarrow$. Then obviously it is consistent with S and T. Therefore, we can assume that S is not empty. If there is a pair (Γ, p) of a hereditary EFS Γ and a predicate symbol p such that

(1) Γ contains at most m definite clauses each of whose head has at most k occurrences of variables, and

(2) $L(\Gamma, p)$ is consistent with S and T,

then without loss of generality we can add the following condition:

(3) (Γ, p) is reduced with respect to S, i.e., $S \subseteq L(\Gamma, p)$ and $S \not\subseteq L(\Gamma', p)$ for any $\Gamma' \subsetneq \Gamma$.

Claim 1. Let $\mathcal{G}(m, k, S)$ be the set of pairs (Γ, p) which satisfies (1)-(3). Then $|\mathcal{G}(m, k, S)|$ is bounded by some polynomial in $\sum_{s \in S} |s|$

Proof. Let (Γ, p) be a pair in $\mathcal{G}(m, k, S)$ and $q(\pi_1, ..., \pi_n) \leftarrow q_1(\tau_1, ..., \tau_{t_1}),\ q_2(\tau_{t_1+1}, \ ..., \tau_{t_2}),\ \ldots,$ $q_l(\tau_{t_{l-1}+1}, ..., \tau_{t_l})$, be a clause in Γ. Since each τ_j contains at least one variable and the head contains at most k variable occurrences, it follows from the length-boundedness that $t_l \leq k$. Let $\Pi(k, S)$ be the set of patterns π up to renaming of variables such that it contains at most k variable occurrences and $\pi\theta$ is a subword of some $s \in S$ for some substitution θ. Then $|\Pi(k, S)| \leq \sum_{s \in S}((|s|^2)^{k+1}k!)$. Since (Γ, p) is reduced, we can see that for any definite clause $q(\pi_1, ..., \pi_n) \leftarrow q_1(\tau_1, ..., \tau_{t_1}), q_2(\tau_{t_1+1}, ..., \tau_{t_2}), \ldots, q_l(\tau_{t_{l-1}+1}, ..., \tau_{t_l})$, there exists a substitution θ such that all $\pi_1\theta, ..., \pi_n\theta$ are subwords of some $w \in S$. Therefore all patterns π_i and τ_j are in $\Pi(k, S)$. Since Γ is reduced and has at most m clauses, there are at most m distinct predicate symbols in Γ. Thus we have $|\mathcal{G}(m, k, S)| \leq (|\Pi(k, S)|^{2k} m^{k+1})^{m+1}$. Since m and k are constants, $|\mathcal{G}(m, k, S)|$ is bounded by a polynomial.

Claim 2. There is an algorithm that, given a word w in Σ^+ and a length-bounded hereditary EFS Γ satisfying (1), decides whether $w \in L(\Gamma, p)$ in polynomial-time with respect to $|w| + |\Gamma|$, where $|\Gamma|$ represents the length of Γ as an expression and p is a specified predicate symbol in Γ.

Proof. We first consider the family $\Gamma(w)$ of the

definite clauses that can be obtained from the definite clauses in Γ by replacing all variables by nonempty subwords of w. Since the number of variable occurrences is bounded by k and since the number of definite clauses in Γ is also bounded by m, $\Gamma(w)$ contains at most polynomially many definite clauses. In order to check whether w is in $L(\Gamma, p)$, we repeat applications of modus ponens to $\Gamma(w)$ until $p(w) \leftarrow$ is derived. This procedure works correctly since Γ is hereditary. It is not hard to see that the total algorithm can be implemented in polynomial time with respect to $|w|+|\Gamma|$.

The polynomial-time algorithm that finds the required length-bounded hereditary EFS runs as follows: The algorithm enumerates pairs (Γ, p) in $\mathcal{G}(m, k, S)$. Then it checks by using the polynomial-time algorithm of Claim 2 whether $s \in L(\Gamma, p)$ for $s \in S$ and $t \notin L(\Gamma, p)$ for $t \in T$. If such pair is found, the algorithm outputs it as a hypothesis.□

It should be noticed that if the number of definite clauses is not bounded by a constant m then the resulting concept class is not of polynomial dimension since all finite sets can be defined. Moreover, we showed in [10] that the problem of finding a regular pattern which is consistent with given positive and negative examples is **NP**-complete. Therefore it is reasonable to make restrictions on the number m of definite clauses and the number k of variable occurrences in order to achieve polynomial-time learnability with respect to domain dimension.

The learning algorithm sketched in the proof of Theorem 1 requires unavailable amount of time and space if it is applied to LB-H-EFS(m,k) for $m, k \geq 10$. This is because the running time is polynomial but exponential with respect to m and k although they are fixed constants. Therefore it seems necessary for feasible learning to make both the number k of variable occurrences and the number m of definite clauses smaller. Moreover, we do not know in advance how small the number m can be. However, if we deal with an EFS consisting of only patterns with a few variables, there is a way of solving this problem in feasible time.

Given a set Pos of positive examples and a set Neg of negative examples, we first compute a set S of patterns which produce some of positive examples but exclude almost all negative examples. Then we compute a subset Γ of S which covers all positive examples in Pos. It is preferable that the size of Γ is as small as possible. But it is known that the minimum set cover problem is NP-complete [5]. Therefore, instead, we use a minimal subset Γ of S in the sense that no proper subset of Γ covers Pos. The following algorithm sketches our implementation.

input Pos, Neg;
$S := \emptyset$; $I := Pos$; $E := Neg$;
foreach pattern π with $\pi\theta = w$ for some $w \in I$ and θ;
  if π excludes almost all examples in E
    then $S := S \cup \{\pi\}$;
Find a subset Γ of S covering I which is minimal with respect to set-inclusion;
output Γ;

Algorithm 1

The approximation algorithm for the minimum set cover problem by Johnson [7] is useful to find a minimal subset Γ in Algorithm 1. It finds very efficiently a set cover of size at most $M \log M$, where M is the size of a minimum set cover.

3 Experiments on transmembrane domains of proteins

The final purpose of this approach is to establish an algorithm which classifies proteins into the following three categories by simply searching amino acid sequences:

C1. Membrane proteins

C2. Secretary proteins

C3. Cytosolic proteins

Figure 1 shows an example of an amino acid sequence of a membrane protein.

```
MDVVNQLVAGGQFRVVKE(PLGFVKVLQWVFAIFAFATCGSY)
TGELRLSVECANKTESALNIEVEFEYPFRLHQVYFDAPSCVKG
GTTKIFLVGDYSSSAE(FFVTVAVFAFLYSMGALATYIFL)QN
KYRENNK(GPMMDFLATAVFAFMWLVSSSAWA)KGLSDVKMAT
DPENIIKEMPMCRQTGNTCKELRDPVTS(GLNTSVVFGFLNLV
LWVGNLWFVF)KETGWAAPFMRAPPGAPEKQPAPGDAYGDAGY
GQGPGGYGPQDSYGPQGGYQPDYGQPASGGGGYGPQGDYGQQG
YGQQGAPTSFSNQM
```

Figure 1: Protein which contains four transmembrane domains shown by the parenthesized parts.

However, it is computationally rather hard to find an appropriate EFS which describes each category since these sequences are too long to apply the learning algorithm in [10] using reasonable amount of computational resources. There is a tendency to assume that a membrane protein has transmembrane domains each of which constitutes an α-helix structure. The reported length of a transmembrane domain is not large, usually, $20 \sim 30$. If a sequence corresponding to a transmembrane domain is found in a protein, the probability that it is a membrane protein may be larger.

Therefore it is important to give an algorithm which identifies transmembrane domains in an amino acid sequence. Since the length of a transmembrane domain is not so large, our learning algorithm may work for learning EFSs consisting of regular patterns which describe the sequences of transmembrane domains. We used 37 membrane proteins from the database of PIR.

A hydropathy plot [4, 8, 15] has been used generally to predict transmembrane domains from primary sequences. Instead of dealing with twenty symbols of amino acids, we classify these symbols into three classes by the hydropathy indices of amino acids [8]. More precisely, we transform symbols by Table 1.

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

Table 1: Transformation rules

This transformation from 20 symbols to just 3 symbols reduces the search space of hypotheses and may make learning from a small number of examples possible. The sequence in Figure 2 is the result of this transformation from the sequence in Figure 1.

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

Figure 2: The sequence obtained by the transformation

As is seen, this transformation makes the characteristics of a transmembrane domain more vivid. Figure 3 gives some of the sequences obtained by this transformation from the transmembrane domains chosen from our 37 examples.

A *positive example* is a sequence which is already known to be a transmembrane domain. A *negative example* is a sequence of length around 30 which is cut out from the part other than transmembrane domains. Hence, for a protein which is not a membrane protein, all sequences of length around 30 are negative examples. We use the examples in Figure 3 as positive training examples. As negative training examples, we

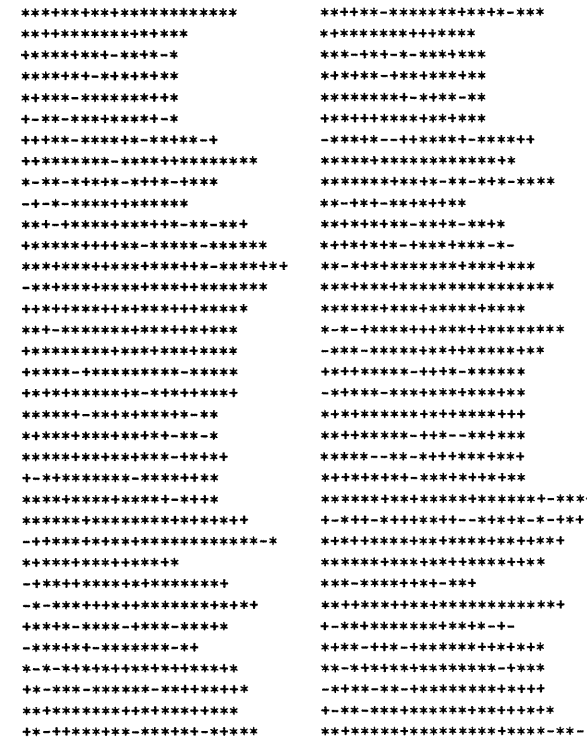


Figure 3: Examples of transformed transmembrane domains

have chosen randomly 100 negative examples shown in Table 4.

The approach we have taken for this problem is to find a collection of regular patterns which covers almost all positive training examples and excludes almost all negative training examples.

We restrict the forms of patterns to the following:

$$x\alpha_1 z$$

$$x\alpha_1 y\alpha_2 z$$

$$x\alpha_1 y_1 \alpha_2 y_2 \alpha_3 z$$

In the above forms, $\alpha_1, \alpha_2, \alpha_3$ are strings in $\{*, +, -\}^+$ constructed from positive training examples and x, y_1, y_2, z are distinct variables, where we allow ε-substitutions to the variables x, z. We consider EFSs

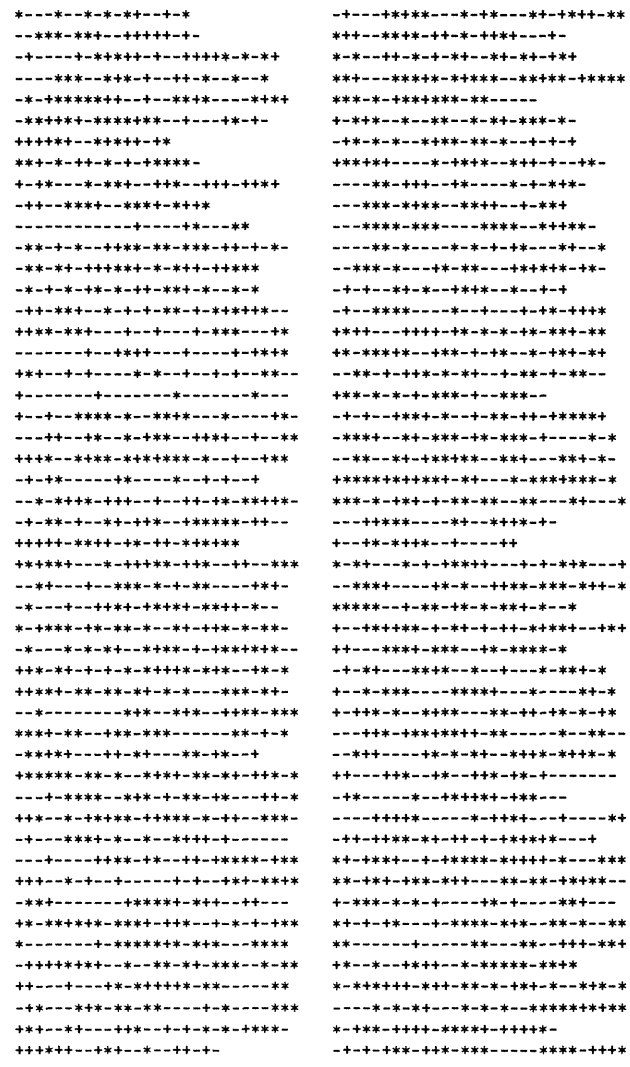


Figure 4: Negative training examples

which consists of a finite number of regular patterns in the above forms.

We have implemented Algorithm 1 so that it can cope with the above number of positive and negative training examples. Our implemented algorithm found 2.44 million patterns that meet our conditions. Then the algorithm produced some reasonable hypotheses from them. The total computation time on Sun SPARC station 2 is about three hours. Table 2 shows the hypotheses and their success rates that our

learning algorithm has produced.

Patterns	Positive	MB	ALL
******	403 (58.5%)	19 (3.0%)	1050 (5.5%)
*+**X+*X**+	273 (39.6%)	32 (5.0%)	1209 (6.3%)
+++**X**+	153 (22.2%)	29 (4.6%)	700 (3.6%)
+*+X**+X+**	130 (18.9%)	24 (3.8%)	1019 (5.3%)
-X*X**+	58 (8.4%)	21 (3.3%)	729 (3.8%)
*-X*****	75 (10.9%)	11 (1.7%)	674 (3.5%)
X**X*+	94 (13.6%)	11 (1.7%)	687 (3.6%)
total	625 (90.7%)	113 (17.8%)	4367 (22.7%)

(P0) Each pattern is consistent with all positive and negative training examples.

Patterns	Positive	MB	ALL
X*X**+	482 (70.0%)	59 (9.3%)	2596 (13.5%)
******	403 (58.5%)	19 (3.0%)	1050 (5.5%)
*+**+X*+	137 (19.9%)	20 (3.2%)	1309 (6.8%)
***+X**+X+**	184 (26.7%)	9 (1.4%)	593 (3.1%)
-X*+X*+	63 (9.1%)	23 (3.6%)	1089 (5.7%)
*****+**	302 (43.8%)	10 (1.6%)	621 (3.2%)
total	632 (91.7%)	101 (15.9%)	4823 (25.0%)

(P1) Each pattern is consistent with all positive training examples and inconsistent with at most one negative training example.

Table 2: Collections of regular patterns produced by our learning algorithm from 70 positive and 100 negative training examples. The leftmost and rightmost variables are extracted for simplicity. The symbol X represents a position where the variable occurs. The second column shows the number of the positive examples from 689 positive examples that the pattern in the first column covers. The third (fourth) column shows the number of the negative examples from MB (ALL) that the pattern in the first column generates.

We verified these hypotheses (P0), (P1) by experiments. As the total space of positive examples, we use the set POS of all transmembrane sequences (689) from PIR database. We consider two kinds of total spaces MB and ALL for negative examples. MB consists of randomly chosen 634 negative examples taken from *membrane* proteins. ALL is the set of negative examples consisting of 19276 negative examples randomly chosen from *all* proteins from PIR. The success rates for positive are more than 90%. The rates rejecting negative examples in MB are not bad (82% $\sim$ 84%). But only 75% $\sim$ 77% of negative examples in ALL can be recognized as negative.

4 Finding patterns from negative training examples

In Section 3, we were interested in collections of regular patterns which cover positive examples and exclude negative examples. In contrast, this section deals with collections of regular patterns which *exclude* positive examples and *cover* negative examples. The strategy is first to generate regular patterns from negative training examples instead of positive training examples and then to apply Algorithm 1 in which statements $I := Pos$; $E := Neg$; are replaced with $I := Neg$; $E := Pos$;. As training examples, we use the same set of positive examples and randomly chosen 50 negative examples in MB.

We made experiments in the same way as in Section 3 using POS for positive examples and MB and ALL for negative examples. Table 3 shows the results. As is seen, hypotheses (N0) and (N1) are very small and the success rates for both positive and negative examples are quite good. From these observations, we

Pattern	Positive	MB	ALL
*-X--X-	12 (1.7%)	444 (70.0%)	13669 (71.0%)
++X+-+	21 (3.0%)	264 (41.6%)	6094 (31.6%)
**X+-X+-	21 (3.0%)	333 (52.5%)	9482 (49.2%)
total	43 (6.2%)	585 (92.3%)	17340 (90.0%)

(N0) Each pattern is consistent with all positive and negative training examples.

Pattern	Positive	MB	ALL
-X*X--	36 (5.2%)	535 (84.4%)	16103 (83.6%)
*+X*X++-	27 (3.9%)	253 (39.9%)	6538 (34.0%)
total	61 (8.9%)	581 (91.6%)	17302 (89.9%)

(N1) Each pattern is consistent with all negative training examples and inconsistent with at most two positive examples.

Table 3: Collections of regular patterns produced by our learning algorithm from 50 negative and 70 positive training examples.

can say that the approach from negative examples is much better than that from positive examples in the last section.

After recognizing the importance of negative examples, we have finally found the following pattern:

-X-X-X-X-

This is an abbreviation of x-y_1-y_2-y_3-y_4-z that generates all sequences containing "-" at least five times. Table 4 (N2) shows the result of the case where ε-substitutions are allowed only to x and z. Table 4 (N3) is the result of the case that all variables allow ε-substitutions where the accuracy of more than 95% is achieved.

Patterns	Positive	MB	ALL
-X-X-X-X-	17 (2.5%)	599 (94.5%)	17555 (91.2%)

(N2) ε-substitutions are not allowed to inside variables.

Patterns	Positive	MB	ALL
-X-X-X-X-	32 (4.6%)	616 (97.2%)	18304 (95.1%)

(N3) ε-substitutions are allowed to all variables.

Table 4: Results for -X-X-X-X-

5 Discussions

In this paper we showed and examined a new framework for acquisition of knowledge from protein data. Even a very restricted class of EFSs is useful for identifying transmembrane domain sequences. Especially, we have shown that a single regular pattern can recognize transmembrane domains with an excellent accuracy. Through these results, we have indicated the importance of negative examples.

Although the learning algorithm [10] for more general class, LB-H-EFS(m,k), is shown to run in polynomial time, it requires enormous amount of time. In order to attack more general situations, we need essential improvements on the algorithm.

In [1] we reported a machine learning system using decision trees over regular patterns which employs the idea of ID3 [14] for the construction of decision trees. We also have good results for the transmembrane domain identification problem. A neural network approach has been taken for prediction of secondary structures of proteins [6]. But the authors have not a chance to examine this method. The comparison with other methods for transmembrane domain identification remains as a future work.

A well-known structure around the membrane integrated domain is the signal-anchor structure that consists of two parts, the hydrophobic part of a membrane-spanning sequence and the charged residues around the hydrophobic part [9, 22]. The pattern x-y_1-y_2-y_3-y_4-z which indicates a cluster of polar amino acid residues may be closely related to the signal-anchor structure.

References

[1] S. Arikawa, S. Kuhara, S. Miyano, A. Shinohara and T. Shinohara, "Identification of transmembrane domains by decision trees over regular patterns," *Technical Report RIFIS-TR-CS-44, Research Institute of Fundamental Information Science, Kyushu University* (to be presented at Second Int. Symp. Artificial Intelligence and Mathematics), 1991.

[2] S. Arikawa, T. Shinohara and A. Yamamoto, "Elementary formal system as a unifying framework for language learning," In *Proc. 2nd Workshop on Computational Learning Theory*, pp. 312-32, 1989 (to appear in *Theoretical Computer Science*).

[3] A. Blumer, A. Ehrenheucht, D. Haussler and M.K. Warmuth, "Learnability and the Vapnik-Chervonenkis dimension," *JACM* Vol. 36, pp. 929-965, 1989.

[4] D.M. Engelman, T.A. Steiz and A. Goldman, "Identifying nonpolar transbilayer helices in amino acid sequences of membrane proteins," *Ann. Rev. Biophys. Biophys. Chem.* Vol. 15, pp. 321-353, 1986.

[5] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* W.H. Freeman, 1979.

[6] L.H. Holley and M. Karplus, "Protein secondary structure prediction with a neural network," In *Proc. Natl. Acad. Sci. USA* Vol. 86, pp. 152-156, 1989.

[7] D.S. Johnson, "Approximation algorithms for combinatorial problems," *JCSS* Vol. 9, pp. 256–278, 1974.

[8] J. Kyte and R.F. Doolittle, "A simple method for displaying the hydropathic character of protein," *J. Mol. Biol.* Vol. 157, pp. 105-132 1982.

[9] J. Lipp, N. Flint, M.T. Haeuptle and B. Dobberstein, "Structural requirements for membrane assembly of proteins spanning the membrane several times," *J. Cell Biol.* Vol. 109, pp 2013-2022, 1989.

[10] S. Miyano, A. Shinohara and T. Shinohara, "Which classes of elementary formal systems are polynomial-time learnable?," *Technical Report RIFIS-TR-CS-37, Research Institute of Fundamental Information Science, Kyushu University,* 1991 (to appear in Proc. ALT'91).

[11] B.K. Natarajan, "On learning boolean functions,,, In *Proc. 19th ACM Symp. Theory of Computing,* pp. 296-304, 1987.

[12] B.K. Natarajan, "On learning sets and functions," *Machine Learning* Vol. 4, pp. 67-97, 1989.

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

[14] J.R. Quinlan, "Induction of decision trees," *Machine Learning* Vol. 1, pp. 81-106, 1986.

[15] J.K.M. Rao and P. Argos, "A conformational preference parameter to predict helices in integral membrane proteins," *Biochim. Biophys. Acta* Vol. 869, pp. 197-214, 1986.

[16] T. Shinohara, "Polynomial time inference of pattern languages and its applications," In *Proc. 7th IBM Symp. on Mathematical Foundations of Computer Science,* pp. 191-209, 1982.

[17] T. Shinohara, "Polynomial time inference of extended regular pattern languages," In *Proc. RIMS Symposia on Software Science and Engineering* (Lecture Notes in Computer Science Vol. 147), pp. 115-127, 1983.

[18] T. Shinohara, "Inductive inference from positive data is powerful," In *Proc. 3rd Workshop on Computational Learning Theory,* pp. 97-110, 1990.

[19] R.F. Smith and T.F. Smith, "Automatic generation of primary sequence patterns from sets of related protein sequences," In *Proc. Natl. Acad. Sci. USA* Vol. 87, pp. 118-122, 1990.

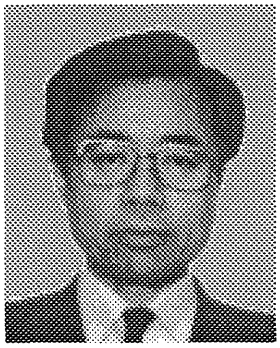
[20] R.M. Smullyan, *Theory of Formal Systems,* Princeton University Press, 1961.

[21] L. Valiant, "A theory of the learnable," *Commun. ACM* Vol. 27, pp. 1134-1142, 1984.

[22] G. von Heijine, "Transcending the impenetrable: how proteins come to terms with membranes," *Biochim. Biophys. Acta* Vol. 947, pp. 307-333, 1988.

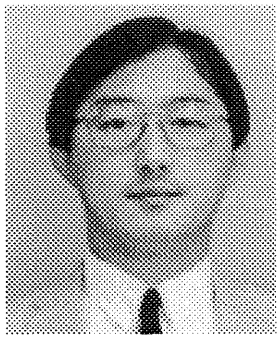
[23] A. Yamamoto, "Elementary formal system as a logic programming language," In *Proc. Logic Programming Conference '89,* pp. 123-132, 1989.

About the Authors

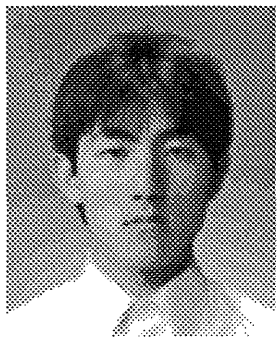


Setsuo Arikawa (有川　節夫) was born in Kagoshima on April 29, 1941. He received the B.S. degree in 1964, the M.S. degree in 1966 and the Dr.Sci. degree in 1969 all in Mathematics from Kyushu University. Presently, he is Professor of Research Institute of Fundamental Inforamtion Science, Kyushu University.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

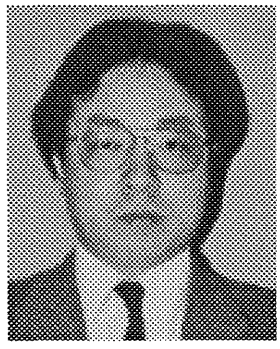
Satoru Kuhara (久原　哲) was born in Fukuoka on April 20, 1950. He recieved the B.A. in 1974, the M.A. degree in 1976 and the Dr. Agr. in 1980 from Kyushu University. Currently, he is an Associate Professor of Graduate School of Genetic Resources Technology, Kyushu University. His present interests include computer analysis of genetic information and protein structure.



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.



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.

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