Learning of Elementary Formal Systems with Two Clauses using Queries and Their Languages

加藤 浩央
Hirotaka Kato
東海大学大学院 理学研究科 数理科学専攻
Graduate School of Science, Tokai University, Hiratsuka 259-1292, Japan

松本 哲志
Satoshi Matsumoto
東海大学 理学部 情報数理科
Department of Mathematical Sciences, Tokai University, Hiratsuka 259-1292, Japan
{hirotaka, matumoto}@ss.u-tokai.ac.jp

宮原 哲浩
Tetsuhiro Miyahara
広島市立大学 情報科学部 知能情報システム工学科
Faculty of Information Sciences, Hiroshima City University, Hiroshima 731-3194, Japan
miyahara@its.hiroshima-cu.ac.jp

Abstract

An elementary formal system, EFS for short, is a kind of logic program over strings, and regarded as a set of rules to generate a language. For an EFS $\Gamma$, the language $L(\Gamma)$ denotes the set of all strings generated by $\Gamma$. Many researchers studied the learnability of EFSs in various learning models. In this paper, we introduce a subclass of EFSs, denoted by $rEFS$, and study the learnability of $rEFS$ in the exact learning model. The class $rEFS$ contains the class of regular patterns, which is extensively studied in Learning Theory.

Let $\Gamma_*$ be a target EFS of learning in $rEFS$. In the exact learning model, an oracle for superset queries answers "yes" for an input EFS $\Gamma$ in $rEFS$ if $L(\Gamma)$ is a superset of $L(\Gamma_*)$, and outputs a string in $L(\Gamma_*) - L(\Gamma)$, otherwise. An oracle for membership queries answers "yes" for an input string $w$ if $w$ is included in $L(\Gamma_*)$, and answers "no", otherwise.

We show that any EFS in $rEFS$ is exactly identifiable in polynomial time using membership and superset queries. Moreover, for other types of queries, we show that there exists no polynomial time learning algorithm for $rEFS$ by using the queries. This result indicates the hardness of learning the class $rEFS$ in the exact learning model, in general.

1 Introduction

An elementary formal system, EFS for short, is a kind of logic program which directly manipulates strings, and is regarded as a set of rules to generate a language. A pattern is a nonempty finite string of constant symbols and variables. In EFSs, patterns are used as terms in a logic program. A rule (or definite clause) in EFSs is a clause of the form $A \leftarrow B_1, \ldots, B_m$ ($m \geq 0$), where $A, B_1, \ldots, B_m$ are atoms. Learning of rules from string data is important in machine learning [1] and it can be applied to learning of rules from HTML files since HTML files are considered to be string data. Learning of EFSs has been long studied in Algorithmic/Computational Learning Theory [4, 8, 10, 11]. The purpose of this work is to give a new learnability of EFSs.

Consider examples of EFSs defined as follows. Let $p$ be a unary predicate symbol, $a$ and $b$ constant symbols, $x$ and $y$ variables. $p(ab) \leftarrow$ and $p(axb) \leftarrow p(x)$ are examples of rules. $\Gamma_1 = \{p(ab) \leftarrow, p(axb) \leftarrow p(x)\}$ is an example of EFS consisting of the above two rules. EFSs
$P$ is the set of all constant strings by substituting non-empty constant symbols for variables and applying Modus Ponens to rules in $\Gamma$. Let $\Sigma = \{a, b, c\}$ be a finite alphabet. In the above examples, $L(\Gamma_1) = \{a^n b^n \mid n \geq 1\}$, $L(\Gamma_2) = \{a^n w b^n \mid w \in \Sigma^+, n \geq 1\}$, and $L(\Gamma_3) = \{aab, abb, abcb, aabbaabc, aabbbabc, \ldots\}.$

In this paper, we give a polynomial time learning algorithm for a subclass of EFSs in the exact learning model. The framework of EFSs for studying formal language theory was established by [3] and the unifying framework of language learning using EFS was originated by [4]. A pattern is regular if each variable appears in the pattern at most once. The target class of learning $\text{rEFS}$ in this paper is defined as the set of EFSs $\Gamma$ which satisfy the following two conditions. (1) All patterns in the heads of all definite clauses in $\Gamma$ are regular. (2) $\Gamma$ consists of one or two definite clauses of the form $p(\pi) \leftarrow$, or exactly two definite clauses of the forms $p(\pi') \leftarrow p(x_1), \ldots, p(x_n)$, where $p$ is a unary predicate symbol, $x_1, \ldots, x_n$ are all of the variables appearing in $\tau$, and $\pi'$ contains at least one variable. By the definition, the classes of regular patterns and unions of two regular patterns are included in $\text{rEFS}$. In the above examples, $\Gamma_1$ is not in $\text{rEFS}$ since the pattern $ab$ contains no variable. $\Gamma_2$ and $\Gamma_3$ are in $\text{rEFS}$.

Let $\Gamma_*$ be an EFS in $\text{rEFS}$ to be identified by a learning algorithm, and we say that the EFS $\Gamma_*$ is a target. We introduce the exact learning model via queries due to Angluin [2]. In this model, learning algorithms can access to oracles that answer specific kinds of queries about the unknown language $L(\Gamma_*).$ We mainly consider the following two oracles in this paper. (1) Superset oracle: The input is an EFS $\Gamma$ in $\text{rEFS}$. If $L(\Gamma) \supseteq L(\Gamma_*), then the output is "yes". Otherwise, it returns a counterexample $t \in L(\Gamma_*) - L(\Gamma).$ The query is called a superset query. (2) Membership oracle: The input is a string $t$ in $\Sigma^+$. The output is "yes" if $t \in L(\Gamma_*),$ and "no" otherwise. The query is called a membership query. A learning algorithm $A$ collects information about $L(\Gamma_*)$ by using queries and outputs an EFS $\Gamma$ in $\text{rEFS}$. We say that a learning algorithm $A$ exactly identifies a target $\Gamma_*$ in polynomial time using a certain type of queries if $A$ halts in polynomial time and outputs an EFS $\Gamma \in \text{rEFS}$ such that $L(\Gamma) = L(\Gamma_*)$ using queries of the specified type.

We discuss the related works of this work. A pattern $\pi$ is regarded as a very restricted form of EFS $\{p(\pi) \leftarrow\}$ in $\text{rEFS}$. Angluin [1] originated the research of pattern learning under another learning model of inductive inference, which is an infinite process of learning. Angluin also showed that patterns are exactly learnable in polynomial time using restricted superset queries [2]. We showed that regular patterns are exactly learnable in polynomial time using membership queries and a positive example [5]. We showed that finite unions of subsequences are exactly learnable in polynomial time using membership and equivalence queries [6]. Moreover, we showed that finite unions of tree patterns are exactly learnable in polynomial time using restricted subset and equivalence queries [7].

The paper [10] deals with a class of restricted EFSs (called primitive EFSs), which is similar but incomparable to $\text{rEFS}$, under the learning model of inductive inference of positive examples without allowing empty string to be substituted for variables. The paper [11] extended this learnability by allowing empty string to be substituted for variables. The work [8] deals with a class of EFSs under the exact learning model using equivalence and extensions of membership queries. The work [8] is known so far about the learnability of EFSs under exact learning model.

2 Preliminaries

Let $S$ be a finite set. We denote by $|S|$ the number of elements in $S$. Let $\Sigma$ be a finite alphabet, $X$ a countable set of variables, and $\Pi$ a set of predicate symbols. We assume that $|\Sigma| \geq 2$ and these sets $\Sigma, X$ and $\Pi$ are mutually distinct. Each predicate symbol is associated with a positive integer called arity. Let $w$ be a string. We denote by $|w|$ the length of $w$. We denote by $w[i]$ the $i$-th symbol in string $w$, and by $w[i : j]$ the substring $w[i] \cdots w[j]$ of $w$. We define $w[i : j] = \varepsilon$ (empty string) if $i > j$. For convenience, a prefix $w[1 : i]$ is abbreviated as $w[i]$, and a suffix $w[i : |w|]$ as $w[i : |w|]$, where $1 \leq i \leq |w|$. For a nonempty set $\Delta$, let $\Delta^+$ denote the set of all nonempty strings.

A pattern is a nonempty string over $\Sigma \cup X$. In particular, we say that a pattern $\pi$ is regular if each variable in $\pi$ appears at most once. An atom is an expression of the form $p(\pi_1, \ldots, \pi_n)$, where $p$ is a predicate symbol with arity $n$ and $\pi_1, \ldots, \pi_n$ are patterns. A definite clause is a clause of the form $A \leftarrow B_1, \ldots, B_m \ (m \geq 0)$, where $A, B_1, \ldots, B_m$ are atoms. The atom $A$ is called the head and the part $B_1, \ldots, B_m$ the body of the definite clause.
Definition 1 An elementary formal system, EFS for short, is a finite set of definite clauses. For an EFS $\Gamma$, each definite clause in $\Gamma$ is called an axiom of $\Gamma$.

A substitution $\theta$ is a homomorphism from patterns to patterns such that $\theta(a) = a$ for each $a \in \Sigma$ and each variable is replaced with any patterns. By $\sigma \theta$, we denote the image of a pattern $\pi$ by a substitution $\theta$. For an atom $A = p(\pi_1, \ldots, \pi_n)$ and a clause $C = A \leftarrow B_1, \ldots, B_m$, we define $A \theta = p(\pi_1 \theta, \ldots, \pi_n \theta)$ and $C \theta = A \theta \leftarrow B_1 \theta, \ldots, B_m \theta$.

For patterns $\pi$ and $\tau$, we introduce binary relations $\preceq$ and $\equiv$ as follows: $\pi \preceq \tau$ if $\pi = \tau \theta$ for some substitution $\theta$, and $\pi \equiv \tau$ if $\pi \preceq \tau$ and $\tau \preceq \pi$. If $\pi \preceq \tau$ and $\tau \not\preceq \pi$, then we write $\tau \prec \pi$.

Let $\pi$ be a pattern, $i$ (for $1 \leq i \leq |\pi|$) a positive integer, and $\alpha$ a symbol in $\Sigma$. We denote by $\pi_{i, \alpha}$ the string obtained from $\pi$ by replacing $\pi[i]$ with $\alpha$, that is, $\pi_{i, \alpha}[i] = \pi[i] - 1 + i$. For a pattern $\pi$, we denote by $S_1(\pi)$ the set of all strings which are obtained from $\pi$ by replacing all variables with a string of length 1. For a nonempty set $P$ of patterns, we define $S_1(P) = \bigcup_{\pi \in P} S_1(\pi)$.

Let $T, T'$ be nonempty sets of patterns. We write $T \subseteq T'$ if for any pattern $\pi \in T$, there is a pattern $\pi' \in T'$ such that $\pi \preceq \pi'$. If $T \subseteq T'$ and $T \not\subseteq T'$, then we write $T \subset T'$.

A definite clause $C$ is provable from an EFS $\Gamma$, denoted by $\Gamma \vdash C$, if $C$ is obtained by finitely many applications of substitutions and Modus Ponens in the way of usual logic programming. We define the language $L(\Gamma, p) = \{ w \in \Sigma^+ \mid \Gamma \vdash p(w) \}$, where $p$ is a unary predicate symbol.

Definition 2 We denote by $rEFS$ the set of EFSs $\Gamma$ which satisfy the following conditions:

1. All patterns in the heads of all clauses in $\Gamma$ are regular.

2. $\Gamma$ consists of one or two clauses of the form $p(\pi) \leftarrow$ and exactly two clauses of the forms $p(\pi') \leftarrow$ and $p(\pi) \leftarrow p(\pi_1), \ldots, p(\pi_n)$, where $p$ is a unary predicate symbol, $x_1, \ldots, x_n$ (for $n \geq 1$) are all the variables appearing in $\pi$, and $\pi'$ contains at least one variable.

By the definition, the class of regular patterns and unions of two regular patterns are included in $rEFS$. We define the size of $\Gamma$, denoted by $|\Gamma|$, as follows: (1) $|\Gamma| = |\pi|$ if $\Gamma = \{ p(\pi) \leftarrow \}$, (2) $|\Gamma| = |\pi_1| + |\pi_2|$ if $\Gamma = \{ p(\pi_1) \leftarrow, p(\pi_2) \leftarrow \}$, (3) $|\Gamma| = |\pi| + |\tau|$ if $\Gamma = \{ p(\pi) \leftarrow, p(\tau) \leftarrow p(x_1), \ldots, p(x_n) \}$.

A language $L$ is an EFS language if $L = L(\Gamma, p)$ for some EFS $\Gamma$ and some unary predicate symbol $p$ in $\Gamma$. In particular, a language $L$ is a regular pattern language if $L = L(\Gamma, p)$ for some EFS $\Gamma = \{ p(\pi) \leftarrow \}$, where $p$ is a regular pattern.

Let $R$ be the set of all regular languages. Let $rEFS$ be the set of all EFS languages by EFSs in $rEFS$.

Example 1 Let $\Gamma = \{ p(axa) \leftarrow, p(byb) \leftarrow p(y) \}$. By the definition of $rEFS$, $\Gamma$ is in $rEFS$. But, $L(\Gamma)$ is not a regular language.

Lemma 1 $rEFS \not\subseteq R$.

Example 2 Let $a$ be a constant symbol and $L = \{ a, aa, aaa \}$. Then, it is clear that $L$ is a regular language. But, there exists no EFS $\Gamma$ in $rEFS$ with $L(\Gamma) = L$.

Lemma 2 $R \not\subseteq rEFS$.

The above lemmas show that $R$ and $rEFS$ are incomparable.

Definition 3 Let $\Gamma$ be an EFS in $rEFS$ and $p$ a unary predicate symbol appearing in $\Gamma$. $\Gamma$ is reduced if $L(\Gamma, p) \subseteq L(\Gamma, p)$ for any $\Gamma' \subseteq \Gamma$.

In this paper, since we deal with the class $rEFS$, we fix a unary predicate symbol, say $p$, and denote $L(\Gamma, p)$ by $L(\Gamma)$ simply. We denote by $\Gamma = (\pi, \tau)$ (resp., $\Gamma = \{ \pi \}$, $\Gamma = \{ \pi, \tau \}$) an EFS $\Gamma = \{ p(\pi) \leftarrow, p(\tau) \leftarrow p(x_1), \ldots, p(x_n) \}$ (resp., $\Gamma = \{ p(\pi) \leftarrow \}$, $\Gamma = \{ p(\pi) \leftarrow, p(\tau) \leftarrow \}$). Moreover, by $L(\{ \pi \})$ (resp., $L(\{ \pi, \tau \})$) we denote $L(\{ p(\pi) \leftarrow \}$, $L(\{ p(\pi) \leftarrow, p(\tau) \leftarrow p(x_1), \ldots, p(x_n) \})$ (resp., $L(\{ p(\pi) \leftarrow \}$, $L(\{ p(\pi) \leftarrow, p(\tau) \leftarrow \}$).

For $\Gamma = (\pi, \tau)$, we define the following particular pattern $\tau_\pi = \tau[x := x_i \mid x$ appears in $\tau]$, where all variables substituted to the variables in $\tau$ are taken to be distinct, so $\tau_\pi$ is always regular. It is clear that $|\pi| \leq |\tau_\pi| \leq |\tau|$. $\tau_\pi$ is defined in a similar way.

Let $\Gamma = (\pi, \tau)$ be an EFS in $rEFS$, $x_1, \ldots, x_n$ all of the variables appearing in $\tau$. $\Gamma[\tau]$ is recursively defined as follows: $\Gamma[\tau] = \{ \tau_\pi \}$ and for any positive integer $t \geq 2$, $\Gamma[t] = \Gamma[t-1] \cup \{ \tau_{x_1 := \zeta_1, \ldots, x_n := \zeta_n} \mid \zeta_i \in \Gamma[t-1] \cup \{ \pi \}, i = 1, \ldots, n \}$. We define $\Gamma_\tau = \bigcup_{t \geq 1} \Gamma[t]$. Note that $\pi$ is not included in $\Gamma_\tau$. Thus, $L(\Gamma_\tau) \subseteq L(\{ \pi, \tau \})$.

A primitive EFS $\Gamma$, a PFS for short, is defined in [11] as follows:

1. All patterns in heads of all clauses in $\Gamma$ are regular.

2. $\Gamma$ consists of exactly two clauses of the forms $p(\pi) \leftarrow$ and $p(\tau) \leftarrow p(x_1), \ldots, p(x_n)$, where $p$ is a unary predicate symbol, and $x_1, \ldots, x_n$ are all of the variables appearing in $\tau$. 

An EFS in $r\mathcal{E}FS$ is different from a PFS. In case of erasing patterns, Uemura et al. showed the following theorem in [11].

**Theorem 1** [11] Let $\Gamma = (\pi, \tau)$ be a PFS. The following statements are equivalent: (i) $\Gamma$ is reduced. (ii) $L(\pi) \cap L(\Gamma_{\tau}) = \emptyset$, where $L(\Gamma_{\tau}) = \bigcup_{\zeta \in \Gamma_{\tau}} L(\zeta)$.

The theorem holds for any EFS $\Gamma = (\pi, \tau)$ in $r\mathcal{E}FS$ in case of nonerasing patterns.

**Corollary 1** Let $\Gamma = (\pi, \tau)$ be an EFS in $r\mathcal{E}FS$. The following statements are equivalent: (i) $\Gamma$ is reduced. (ii) $L(\pi) \cap L(\Gamma_{\tau}) = \emptyset$, where $L(\Gamma_{\tau}) = \bigcup_{\zeta \in \Gamma_{\tau}} L(\zeta)$.

## 3 Learning model

In this paper, let $\Gamma_{*}$ be an EFS in $r\mathcal{E}FS$ to be identified, and we say that the EFS $\Gamma_{*}$ is a target. Non-reduced EFSs have redundant axioms. Even if we consider only reduced EFSs, the expressive power of EFSs is same. So we assume that target EFSs are reduced.

We introduce the exact learning model via queries due to Angluin [2]. In this model, learning algorithms can access to *oracles* that answer specific kinds of queries about the unknown language $L(\Gamma_{*})$. We consider the following oracles.

(1). *Super set oracle* $\text{Sup}_{\Gamma}$: The input is an EFS $\Gamma$ in $r\mathcal{E}FS$. If $L(\Gamma) \supseteq L(\Gamma_{*})$, then the output is "yes". Otherwise, it returns a *counterexample* $t \in L(\Gamma_{*}) - L(\Gamma)$. The query is called a *superset query*. (2). *Subset oracle* $\text{Sub}_{\Gamma}$: The input is an EFS $\Gamma$ in $r\mathcal{E}FS$. If $L(\Gamma) \subseteq L(\Gamma_{*})$, then the output is "yes". Otherwise, it returns a *counterexample* $t \in L(\Gamma) - L(\Gamma_{*})$. The query is called a *subset query*. (3). *Membership oracle* $\text{Mem}_{\Gamma}$: The input is a string $t$ in $\Sigma^{*}$. The output is "yes" if $t \in L(\Gamma_{*})$, and "no" otherwise. The query is called a *membership query*. (4). *Equivalence oracle* $\text{Equiv}_{\Gamma}$: The input is an EFS $\Gamma$ in $r\mathcal{E}FS$. The output is "yes" if $L(\Gamma) = L(\Gamma_{*})$. Otherwise, it returns a *counterexample* $t \in (L(\Gamma) - L(\Gamma_{*})) \cup (L(\Gamma_{*}) - L(\Gamma))$. The query is called an *equivalence query*.

A learning algorithm $A$ collects information about $L(\Gamma_{*})$ by using queries and output an EFS $\Gamma$ in $r\mathcal{E}FS$. We say that a learning algorithm $A$ *exactly identifies* a target $\Gamma_{*}$ in polynomial time using a certain type of queries if $A$ halts in polynomial time with respect to $|\Gamma_{*}|$ and outputs an EFS $\Gamma \in r\mathcal{E}FS$ such that $L(\Gamma) = L(\Gamma_{*})$ using queries of the specified type.

### Figure 1: Procedure LENGTH1 and LEARN.PI

#### Procedure LENGTH1
*Given:* An oracle $\text{Sup}_{\Gamma}$ for the target $\Gamma_{*}$.

Begin

1. $\ell := 1$;
2. // $x_{1} \cdots x_{\ell+1}$ is a regular pattern
3. while $\text{Sup}_{\Gamma_{*}}((x_{1} \cdots x_{\ell+1}))$ = "yes" do
   4. $\ell := \ell + 1$;
   5. output $\ell$;
End.

#### Procedure LEARN.PI($\ell$)
*Input:* A positive integer $\ell$ with $\ell = |\pi_{*}|$;

*Given:* An oracle $\text{Sup}_{\Gamma}$ for the target $\Gamma_{*}$.

Begin

1. // $x_{1}x_{2} \cdots x_{\ell}$ is a regular pattern
2. $\pi := x_{1}x_{2} \cdots x_{\ell}$;
3. for $i := 1$ to $\ell$ do begin
   4. foreach $\alpha \in \Sigma$ do begin
      5. $\pi' := \pi_{i,\alpha}$;
      6. if $\text{Sup}_{\Gamma_{*}}((\pi', x_{1} \cdots x_{\ell+1}))$ = "yes" then begin
         7. $\pi := \pi'$; break;
      8. end;
   9. end;
10. output $\pi$;
end.

End.

## 4 Learning of restricted EFSs using Queries

Let $\Gamma_{*}$ be a target EFS in $r\mathcal{E}FS$. Then we consider the following cases: (i). $\Gamma_{*} = \{\pi_{*}\}$. (ii). $\Gamma_{*} = \{\pi_{*}, \tau_{*}\}$ and the length of $\pi_{*}$ is the same as $\tau_{*}$, that is, $|\pi_{*}| = |\tau_{*}|$. (iii). $\Gamma_{*} = \{\pi_{*}, \tau_{*}\}$ and the length of $\pi_{*}$ is not the same as $\tau_{*}$, that is, $|\pi_{*}| \neq |\tau_{*}|$. Without loss of generality, we assume $|\pi_{*}| < |\tau_{*}|$. (iv). $\Gamma_{*} = (\pi_{*}, \tau_{*})$.

When $\Gamma_{*}$ is in the cases (i), (ii) or (iii), we can regard $\Gamma_{*}$ as a set of at most two regular patterns. Since we use Theorem 2 for some lemmas and theorems, we assume $|\Sigma| \geq 5$ in this paper.

**Theorem 2** [9] Suppose $|\Sigma| \geq 2k + 1$. Let $P$ be a nonempty finite set of regular patterns, $Q$ a set of at most $k$ regular patterns. Then the following three statements are equivalent: (1) $P \subseteq Q$, (2) $L(P) \subseteq L(Q)$, (3) $S_{1}(P) \subseteq L(Q)$. 
Procedure LEARN.PL.TAU1($\pi$)
Input: A regular pattern $\pi$ satisfying
Condition A and $L(\Gamma_*) \subseteq L(\{\pi\})$;
Given: An oracle $\text{Sup}_{\Gamma_*}$ for the target $\Gamma_*$;
begin
\[ t_\pi := 1; \]
while \((t_\pi \leq |\pi|)\) and \((\pi[t_\pi] \in \Sigma)\) do
\[ t_\pi := t_\pi + 1; \]
while \(i \leq |\pi|\) do begin
\begin{enumerate}
\item foreach $\alpha \in \Sigma$ do begin
\end{enumerate}
\begin{enumerate}
\item \begin{enumerate}
\item \begin{enumerate}
\item while \(i \leq |\pi|\) do begin
\item \begin{enumerate}
\item \begin{enumerate}
\item while \((i \leq |\pi|)\) and \((\pi[i] \in \Sigma)\) do
\end{enumerate}
\end{enumerate}
end;
\item \begin{enumerate}
\item while \((i \leq |\pi|)\) and \((\pi[i] \in \Sigma)\) do
\end{enumerate}
end;
\end{enumerate}
\begin{enumerate}
\item output "no";
\end{enumerate}
end.
end.

Condition A:
\begin{enumerate}
\item A-1 $\pi$ satisfies \(|\pi| = |\pi_\pi|\) and $L(\Gamma_*) \subseteq L(\{\pi, x_1 x_2 \cdots x_{|\pi|+1}\})$,
\end{enumerate}
\begin{enumerate}
\item A-2 There is no regular pattern $\pi'$ such that \(|\pi'| = |\pi|\), $\pi' \prec \pi$ and $L(\Gamma_*) \subseteq L(\{\pi', x_1 x_2 \cdots x_{|\pi|+1}\})$ for $\pi$.
\end{enumerate}

Condition B:
\begin{enumerate}
\item B-1 $\pi$ satisfies Condition A and $L(\Gamma_*) \subseteq L(\{\pi\})$,
\end{enumerate}
\begin{enumerate}
\item B-2 There are regular patterns $\pi'$ and $\tau'$ such that \(|\pi'| = |\tau'| = |\pi|\), $\pi' \prec \pi$, $\tau' \prec \pi$ and $L(\Gamma_*) \subseteq L(\{\pi', \tau'\})$ for $\pi$.
\end{enumerate}

Condition C:

Procedure LEARN.PL.TAU2($\pi'$, $\tau'$)
Input: Regular patterns $\pi'$ and $\tau'$ satisfying Condition B-2 for $\pi$
Given: An oracle $\text{Sup}_{\Gamma_*}$ for the target $\Gamma_*$
begin
\[ \pi'' := \pi_\pi'; i := 1; \]
while \((i \leq |\pi'|)\) and \((\pi''[i] \in \Sigma)\) do
\[ i := i + 1; \]
while \(i \leq |\pi'|\) do begin
\begin{enumerate}
\item foreach $\alpha \in \Sigma$ do begin
\end{enumerate}
\begin{enumerate}
\item \begin{enumerate}
\item while \((i \leq |\pi'|)\) and \((\pi''[i] \in \Sigma)\) do
\end{enumerate}
end;
\item \begin{enumerate}
\item while \((i \leq |\pi'|)\) and \((\pi''[i] \in \Sigma)\) do
\end{enumerate}
end;
\begin{enumerate}
\item output $\pi''$, $\tau''$;
\end{enumerate}
end.

Condition C-1 $\pi$ satisfies Condition A and $L(\Gamma_*) \not\subseteq L(\{\pi\})$, and
\begin{enumerate}
\item C-2 There is a regular pattern $\tau$ satisfying the following conditions for $\pi$:
\end{enumerate}
\begin{enumerate}
\item C-2-1 There is a shortest string $w \in L(\Gamma_*) - L(\{\pi\})$ such that \(|w| = |\tau|\) and $w \prec \tau$.
\item C-2-2 $L(\{\tau\}) \subseteq L(\Gamma_*)$.
\item C-2-3 There is no regular pattern $\tau'$ such that \(|\tau'| = |\tau|\), $\tau \prec \tau'$ and $L(\{\tau'\}) \subseteq L(\Gamma_*)$.
\end{enumerate}

By using the above conditions, Corollary 1 and Theorem 2, we can show the following theorem.
Procedure LENGTH2(\(\pi\))
Input: A regular pattern \(\pi\) satisfying
Condition A and \(L(\Gamma_{*}) \not\subseteq L(\pi)\);
Given: An oracle \(\text{Sup}_{\Gamma_{*}}\) for the target \(\Gamma_{*}\);
begin
\(\ell := |\pi| + 1;\)
// \(x_{1} \cdots x_{\ell+1}\) is a regular pattern.
while \(\text{Sup}_{\Gamma_{*}}(\{\pi, x_{1} \cdots x_{\ell+1}\}) = \text{"yes"}\) do
\(\ell := \ell + 1;\)
Let \(w\) be a counterexample obtained
by the oracle \(\text{Sup}_{\Gamma_{*}}\);
output \(w;\)
end.

Procedure LEARN.TAU2(\(\pi, \tau\))
Input: Regular patterns \(\pi\) and \(\tau\)
such that \(\pi\) satisfies Condition C,
\(\tau\) satisfies Condition C-2 for \(\pi\),
and \(L(\Gamma_{*}) \not\subseteq L(\{\pi, \tau\})\);
begin
\(\tau' := \epsilon; i := 1; j := 1;\)
while \(j \leq |\pi| - |\pi| + 1\) do begin
\(\pi' := \pi[j : j + |\pi| - 1];\)
if \(\pi \equiv \pi'\) then begin
\(\tau' := \tau'[i : j-1]x_{j};\)
\(i := j + |\pi|;\)
\(j := j + |\pi|;\)
end else begin
\(j := j + 1;\)
end;
end;
end;
output \(\tau'\);
end.

Figure 5: Procedure LEARN.TAU2

each of the hypotheses \(L_{i}\) using equivalence, membership, and subset queries must at least \(N - 1\) queries in the worst case.

Theorem 4 Any learning algorithm that exactly identifies all strings of length \(n\) using equivalence, membership and subset queries must make at least \(5^{n} - 1\) queries in the worst case.

6 Conclusions

In this paper, we have investigated exact identification of an EFS in \(rEFS\) using queries. We have shown that any EFS in \(rEFS\) is exactly identifiable in \(O(|\Gamma_{*}|^{4})\) time using \(O(|\Gamma_{*}|^{2})\) membership queries and \(O(|\Gamma_{*}|^{2})\) superset queries, where \(|\Sigma| \geq 5\).

Moreover, we have shown that there exists no polynomial time learning algorithm which identifies any EFS in \(rEFS\) using membership, equivalence and subset queries. As future works, we will consider the learnability of \(rEFS\) in the framework of inductive inference from positive data.

References

Algorithm LEARN_REFS

Given: An oracle $Sup_{\Gamma}$, for the target $\Gamma$;

begin
  $\ell := LENGTH1$;
  $\pi := LEARN.PI(\ell)$;
  if $Sup_{\Gamma}((\{\pi\}) = "yes"$ then begin
    if $LEARN.PI.TAU1(\pi) = "no"$ then
      $H := \{\pi\}$
    else begin
      Let $\pi'$ and $\tau'$ be regular patterns
      output by $LEARN.PI.TAU1(\pi)$;
      $\{\pi'', \tau''\} := LEARN.PI.TAU2(\pi', \tau')$;
      $H := \{\pi'', \tau''\}$
    end;
  else begin
    $w := LENGTH2(\pi)$;
    $\tau := LEARN.TAU1(\pi, w)$;
    if $Sup_{\Gamma}((\{\pi, \tau\}) = "yes"$ then
      $H := \{\pi, \tau\}$
    else begin
      $\tau' := LEARN.TAU2(\pi, \tau)$;
      $H := (\pi, \tau')$
    end;
  end;
end;
output $H$;
end.

Figure 6: Algorithm LEARN_REFS


