数理解析研究所講究録
1489 巻 2006 年 64-70

[10]

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 情報数理学科
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 Γ, the language $L(\Gamma)$ denotes the set of all strings generated by Γ. Many researchers studied the learnability of EFSs in various learning models. In this paper, we introduce a subclass of EFSs, denoted by $r\mathcal{EFS}$, and study the learnability of $r\mathcal{EFS}$ in the exact learning model. The class $r\mathcal{EFS}$ contains the class of regular patterns, which is extensively studied in Learning Theory.

Let Γ_* be a target EFS of learning in $r\mathcal{EFS}$. In the exact learning model, an oracle for superset queries answers "*yes*" for an input EFS Γ in $r\mathcal{EFS}$ 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 $r\mathcal{EFS}$ 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 $r\mathcal{EFS}$ by using the queries. This result indicates the hardness of learning the class $r\mathcal{EFS}$ 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 $A \leftarrow B_1, \cdots, B_m$ $(m \geq 0)$, where $A, B_1, \cdots, B_m$ 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

$\Gamma_2 = \{p(axb) \leftarrow, \ p(ayb) \leftarrow p(y)\}$ and $\Gamma_3 = \{p(axb) \leftarrow, \ p(aybzc) \leftarrow p(y), p(z)\}$ are defined similarly. The *language* $L(\Gamma)$ generated by an EFS Γ is the set of all constant strings by substituting non-empty constant symbols for variables and applying Modus Ponens to rules in Γ. 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 $L(\Gamma_3) = \{aab, abb, acb, aaabbaabc, aabbbaabc, \ldots\}$.

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 EFSs was originated by [4]. A pattern is *regular* if each variable appears in the pattern at most once. The target class of learning $r\mathcal{EFS}$ in this paper is defined as the set of EFSs Γ which satisfy the following two conditions. (1) All patterns in the heads of all definite clauses in Γ are regular. (2) Γ consists of one or two definite clauses of the form $p(\pi) \leftarrow$, or exactly two definite clauses of the forms $p(\pi') \leftarrow$ and $p(\tau) \leftarrow p(x_1), \cdots, p(x_n)$, where p is a unary predicate symbol, $x_1, \cdots, x_n$ $(n \geq 1)$ are all of the variables appearing in τ, and π' contains at least one variable. By the definition, the classes of regular patterns and unions of two regular patterns are included in $r\mathcal{EFS}$. In the above examples, Γ_1 is not in $r\mathcal{EFS}$ since the pattern ab contains no variable. Γ_2 and Γ_3 are in $r\mathcal{EFS}$.

Let Γ_* be an EFS in $r\mathcal{EFS}$ to be identified by a learning algorithm, and we say that the EFS Γ_* 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 Γ in $r\mathcal{EFS}$. 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 Σ^+. The output is *"yes"* if $t \in L(\Gamma_*)$, and *"no"* otherwise. The query is called a *membership query*. A learning algorithm $\mathcal{A}$ collects information about $L(\Gamma_*)$ by using queries and outputs an EFS Γ in $r\mathcal{EFS}$. We say that a learning algorithm $\mathcal{A}$ *exactly identifies* a target Γ_* in polynomial time using a certain type of queries if $\mathcal{A}$ halts in polynomial time and outputs an EFS $\Gamma \in r\mathcal{EFS}$ such that $L(\Gamma) = L(\Gamma_*)$ using queries of the specified type.

We discuss the related works of this work. A pattern π is regarded as a very restricted form of EFS $\{p(\pi) \leftarrow\}$ in $r\mathcal{EFS}$. 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 $r\mathcal{EFS}$, 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 Σ be a *finite alphabet*, X a countable set of *variables*, and Π a set of *predicate* symbols. We assume that $|\Sigma| \geq 2$ and these sets Σ, X and Π 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 $w[i:]$, where $1 \leq i \leq |w|$. For a nonempty set Δ, let Δ^+ denote the set of all nonempty strings.

A *pattern* is a nonempty string over $\Sigma \cup X$. In particular, we say that a pattern π is *regular* if each variable in π appears at most once. An *atom* is an expression of the form $p(\pi_1, \cdots, \pi_n)$, where p is a predicate symbol with arity n and $\pi_1, \cdots, \pi_n$ are patterns. A *definite clause* is a clause of the form $A \leftarrow B_1, \cdots, B_m$ $(m \geq 0)$, where A, $B_1, \cdots, B_m$ are atoms. The atom A is called the *head* and the part $B_1, \cdots, B_m$ 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 Γ, each definite clause in Γ is called an *axiom* of Γ.

A *substitution* θ 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 $\pi\theta$, we denote the image of a pattern π by a substitution θ. For an atom $A = p(\pi_1, \cdots, \pi_n)$ and a clause $C = A \leftarrow B_1, \cdots, B_m$, we define $A\theta = p(\pi_1\theta, \cdots, \pi_n\theta)$ and $C\theta = A\theta \leftarrow B_1\theta, \cdots, B_m\theta$.

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

Let π be a pattern, i $(1 \leq i \leq |\pi|)$ a positive integer, and α a symbol in Σ. We denote by $\pi_{i,\alpha}$ the string obtained from π by replacing $\pi[i]$ with α, that is, $\pi_{i,\alpha} = \pi[: i-1]\alpha\pi[i+1 :]$.

For a pattern π, we denote by $S_1(\pi)$ the set of all strings which are obtained from π by replacing all variables with a string of length 1. For a nonempty set P of patterns, we define $S_1(P) = \cup_{\pi \in P} S_1(\pi)$. Let T, T' be nonempty sets of patterns. We write $T \sqsubseteq T'$ if for any pattern $\pi \in T$, there is a pattern $\pi' \in T'$ such that $\pi \preceq \pi'$. If $T \sqsubseteq T'$ and $T \not\sqsupseteq T'$, then we write $T \sqsubset T'$.

A definite clause C is *provable from* an EFS Γ, denoted by $\Gamma \vdash C$, if C is obtained by finitely many applications of substitutions and Modus Ponens as 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 $r\mathcal{EFS}$ the set of EFSs Γ which satisfy the following conditions:

1. All patterns in the heads of all clauses in Γ are regular.

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

By the definition, the class of regular patterns and unions of two regular patterns are included in $r\mathcal{EFS}$. We define the *size* of Γ, 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 Γ and some unary predicate symbol p in Γ. In particular, a language L is a *regular pattern language* if $L = L(\Gamma, p)$ for some EFS $\Gamma = \{p(\pi) \leftarrow\}$, where π is a regular pattern.

Let $\mathcal{R}$ be the set of all regular langauges. Let $r\mathcal{LEFS}$ be the set of all EFS languages by EFSs in $r\mathcal{EFS}$.

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

Lemma 1 $r\mathcal{LEFS} \not\subseteq \mathcal{R}$.

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 Γ in $r\mathcal{EFS}$ with $L(\Gamma) = L$.

Lemma 2 $\mathcal{R} \not\subseteq r\mathcal{LEFS}$.

The above lemmas show that $\mathcal{R}$ and $r\mathcal{LEFS}$ are incomparable.

Definition 3 Let Γ be an EFS in $r\mathcal{EFS}$ and p a unary predicate symbol appearing in Γ. Γ is *reduced* if $L(\Gamma', p) \subsetneq L(\Gamma, p)$ for any $\Gamma' \subsetneq \Gamma$.

In this paper, since we deal with the class $r\mathcal{EFS}$, 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 $\Gamma = \{p(\pi) \leftarrow,\ p(\tau) \leftarrow p(x_1), \cdots, p(x_n)\}$ (resp., $\Gamma = \{p(\pi) \leftarrow\}$, $\Gamma = \{p(\pi) \leftarrow,\ p(\tau) \leftarrow\}$). Moreover, by $L((\pi, \tau))$ (resp., $L(\{\pi\})$, $L(\{\pi, \tau\})$) we denote $L(\{p(\pi) \leftarrow,\ p(\tau) \leftarrow p(x_1), \cdots, p(x_n)\})$ (resp., $L(\{p(\pi) \leftarrow\})$, $L(\{p(\pi) \leftarrow,\ p(\tau) \leftarrow\})$).

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

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

A *primitive* EFS Γ, a PFS for short, is defined in [11] as follows:

1. All patterns in heads of all clauses in Γ are regular.

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

An EFS in $r\mathcal{EFS}$ 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) Γ is reduced. (ii) $L(\pi) \cap L(\Gamma_\tau) = \emptyset$, where $L(\Gamma_\tau) = \cup_{\zeta \in \Gamma_\tau} L(\zeta)$.

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

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

3 Learning model

In this paper, let Γ_* be an EFS in $r\mathcal{EFS}$ to be identified, and we say that the EFS Γ_* 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). *Superset oracle* Sup_{Γ_*}: The input is an EFS Γ in $r\mathcal{EFS}$. 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* Sub_{Γ_*}: The input is an EFS Γ in $r\mathcal{EFS}$. 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* Mem_{Γ_*}: The input is a string t in Σ^+. The output is *"yes"* if $t \in L(\Gamma_*)$, and *"no"* otherwise. The query is called a *membership query*. (4). *Equivalence oracle* $Equiv_{\Gamma_*}$: The input is an EFS Γ in $r\mathcal{EFS}$. 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 $\mathcal{A}$ collects information about $L(\Gamma_*)$ by using queries and output an EFS Γ in $r\mathcal{EFS}$. We say that a learning algorithm $\mathcal{A}$ *exactly identifies* a target Γ_* in polynomial time using a certain type of queries if $\mathcal{A}$ halts in polynomial time with respect to $|\Gamma_*|$ and outputs an EFS $\Gamma \in r\mathcal{EFS}$ such that $L(\Gamma) = L(\Gamma_*)$ using queries of the specified type.

Procedure *LENGTH1*
Given: An oracle Sup_{Γ_*} for the target Γ_*;
begin
  $\ell := 1$;
  // $x_1 \cdots x_{\ell+1}$ is a regular pattern
  while $Sup_{\Gamma_*}(\{x_1 \cdots x_{\ell+1}\}) =$ *"yes"* **do**
    $\ell := \ell + 1$;
  output ℓ;
end.

Procedure *LEARN_PI(ℓ)*
Input: A positive integer ℓ with $\ell = |\pi_*|$;
Given: An oracle Sup_{Γ_*} for the target Γ_*;
begin
  // $x_1 x_2 \cdots x_\ell$ is a regular pattern
  $\pi := x_1 x_2 \cdots x_\ell$;
  for $i := 1$ **to** ℓ **do begin**
    foreach $\alpha \in \Sigma$ **do begin**
      $\pi' := \pi_{i,\alpha}$;
      if $Sup_{\Gamma_*}(\{\pi', x_1 \cdots x_{\ell+1}\}) =$ *"yes"*
        then begin
          $\pi := \pi'$; **break**;
        end;
    end;
  end;
  output π;
end.

Figure 1: **Procedure** *LENGTH1* and *LEARN_PI*

4 Learning of restricted EFSs using Queries

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

When Γ_* is in the cases (i), (ii) or (iii), we can regard Γ_* 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) $P \sqsubseteq Q$, (2) $L(P) \subseteq L(Q)$, (3) $S_1(P) \subseteq L(Q)$.

```
Procedure LEARN_PI_TAU1(π)
Input: A regular pattern π satisfying
        Condition A and L(Γ*) ⊆ L({π});
Given: An oracle Sup_Γ* for the target Γ*;
begin
  i_π := 1;
  while ((i_π ≤ |π|) and (π[i_π] ∈ Σ)) do
    i_π := i_π + 1;
  while i_π ≤ |π| do begin
    foreach α ∈ Σ do begin
      i_τ := i_π;
      while i_τ ≤ |π| do begin
        foreach β ∈ Σ do begin
          π' := π_{i_π,α};
          π'' := π_{i_τ,β};
          if Sup_Γ*({π', π''}) = "yes" then
            begin
              output π', π''; halt;
            end;
        end;
        i_τ := i_τ + 1;
        while ((i_τ ≤ |π|) and (π[i_τ] ∈ Σ)) do
          i_τ := i_τ + 1;
      end;
    end;
    i_π := i_π + 1;
    while ((i_π ≤ |π|) and (π[i_π] ∈ Σ)) do
      i_π := i_π + 1;
  end;
  output "no";
end.
```

Figure 2: **Procedure** *LEARN_PI_TAU1*

For a regular pattern π, we define the following three conditions.

Condition A:

A-1 π satisfies $|\pi| = |\pi_*|$ and $L(\Gamma_*) \subseteq L(\{\pi, x_1x_2\cdots x_{|\pi|+1}\})$, and

A-2 There is no regular pattern π' such that $|\pi'| = |\pi|$, $\pi' \prec \pi$ and $L(\Gamma_*) \subseteq L(\{\pi', x_1x_2\cdots x_{|\pi|+1}\})$ for π.

Condition B:

B-1 π satisfies Condition A and $L(\Gamma_*) \subseteq L(\{\pi\})$, and

B-2 There are regular patterns π' and τ' such that $|\pi'| = |\tau'| = |\pi|$, $\pi' \prec \pi$, $\tau' \prec \pi$ and $L(\Gamma_*) \subseteq L(\{\pi', \tau'\})$ for π.

Condition C:

```
Procedure LEARN_PI_TAU2(π', τ')
Input: Regular patterns π' and τ'
        satisfying Condition B-2 for π,
          where π satisfies Condition B;
Given: An oracle Sup_Γ* for the target Γ*;
begin
  π'' := π'; i := 1;
  while (i ≤ |π''|) and (π''[i] ∈ Σ) do
    i := i + 1;
  while i ≤ |π''| do begin
    foreach α ∈ Σ do begin
      if Sup_Γ*({π''_{i,α}, τ'}) = "yes" then
        begin
          π'' := π''_{i,α}; break;
        end;
    end;
    i := i + 1;
    while (i ≤ |π''|) and (π''[i] ∈ Σ) do
      i := i + 1;
  end;
  τ'' := τ'; i := 1;
  while (i ≤ |τ''|) and (τ''[i] ∈ Σ) do
    i := i + 1;
  while i ≤ |τ''| do begin
    foreach α ∈ Σ do begin
      if Sup_Γ*({π', τ''_{i,α}}) = "yes" then
        begin
          τ'' := τ''_{i,α}; break;
        end;
    end;
    i := i + 1;
    while (i ≤ |τ''|) and (τ''[i] ∈ Σ) do
      i := i + 1;
  end;
  output π'', τ'';
end.
```

Figure 3: **Procedure** *LEARN_PI_TAU2*

C-1 π satisfies Condition A and $L(\Gamma_*) \not\subseteq L(\{\pi\})$, and

C-2 There is a regular pattern τ satisfying the following conditions for π:

- **C-2-1** There is a shortest string $w \in L(\Gamma_*) - L(\{\pi\})$ such that $|w| = |\tau|$ and $w \preceq \tau$.
- **C-2-2** $L(\{\tau\}) \subseteq L(\Gamma_*)$.
- **C-2-3** There is no regular pattern τ' such that $|\tau'| = |\tau|$, $\tau \prec \tau'$ and $L(\{\tau'\}) \subseteq L(\Gamma_*)$.

By using the above conditions, Corollary 1 and Theorem 2, we can show the following theorem.

Procedure *LENGTH2*(π)
Input: A regular pattern π satisfying
Condition A and $L(\Gamma_*) \not\subseteq L(\pi)$;
Given: An oracle Sup_{Γ_*} for the target Γ_*
begin
$\ell := |\pi| + 1$;
// $x_1 \cdots x_{\ell+1}$ is a regular pattern.
while $Sup_{\Gamma_*}(\{\pi, x_1 \cdots x_{\ell+1}\})$ = "yes" **do**
$\ell := \ell + 1$;
Let w be a counterexample obtained
by the oracle Sup_{Γ_*};
output w;
end.

Procedure *LEARN_TAU1*(π,w)
Input: A regular pattern π satisfying
Condition A and $L(\Gamma_*) \not\subseteq L(\{\pi\})$, and
a shortest string $w \in L(\Gamma_*) - L(\{\pi\})$;
Given: An oracle Mem_{Γ_*} for the target Γ_*;
begin
$\tau := w$;
for $i := 1$ **to** $|w|$ **do**
foreach $\alpha \in \Sigma$ **do**
if $w[i] \neq \alpha$ **then**
if $Mem_{\Gamma_*}(w_{i,\alpha})$ = "yes" **then**
if $w_{i,\alpha} \notin L(\pi)$ **then begin**
$\tau[i] := x_i$; **break**;
end;
output τ;
end.

Figure 4: **Procedure** *LENGTH2* and *LEARN_TAU1*

Theorem 3 The algorithm *LEARN_REFS* of Fig. 6 identifies any EFS $\Gamma \in r\mathcal{EFS}$ in $O(|\Gamma_*|^4)$ time using $O(|\Gamma_*|^2)$ membership queries and $O(|\Gamma_*|^2)$ superset queries, where $|\Sigma| \geq 5$.

5 Hardness Results on Learnability

In this section, we show the insufficiency of learning $r\mathcal{EFS}$ in the query learning model. By Lemma 3, we can show Theorem 4.

Lemma 3 [2] Suppose the hypothesis space contains a class of distinct sets $L_1, \ldots, L_N$, and there exists a set $L_\cap$ which is not a hypothesis, such that for any pair of distinct indices i and j, $L_\cap = L_i \cap L_j$. Then any algorithm that exactly identifies

Procedure *LEARN_TAU2*(π,τ)
Input: Regular patterns π and τ
such that π satisfies Condition C,
τ satisfies Condition C-2 for π,
and $L(\Gamma_*) \not\subseteq L(\{\pi,\tau\})$;
begin
$\tau' := \varepsilon$; $i := 1$; $j := 1$;
while $j \leq |\tau| - |\pi| + 1$ **do begin**
$\pi' := \tau[j : j + |\pi| - 1]$;
if $\pi \equiv \pi'$ **then begin**
$\tau' := \tau'\tau[i : j-1]x_j$;
$i := j + |\pi|$;
$j := j + |\pi|$;
end else begin
$j := j + 1$;
end;
end;
$\tau' := \tau'\tau[j :]$;
output τ';
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 $r\mathcal{EFS}$ using queries. We have shown that any EFS in $r\mathcal{EFS}$ 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 $r\mathcal{EFS}$ using membership, equivalence and subset queries. As future works, we will consider the learnability of $r\mathcal{EFS}$ in the framework of inductive inference from positive data.

References

[1] D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Science*, 21:46–62, 1980.

Algorithm *LEARN_REFS*
Given: An oracle Sup_{Γ_*} for the target Γ_*;
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 π' and τ' be regular patterns
          output by $LEARN\_PI\_TAU1(\pi)$;
      $\{\pi'', \tau''\} := LEARN\_PI\_TAU2(\pi', \tau')$;
      $H := \{\pi'', \tau''\}$;
    end;
  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;
  output H;
end.

Figure 6: **Algorithm** *LEARN_REFS*

[2] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[3] S. Arikawa. Elementary formal systems and formal languages - simple formal systems. *Memoirs of Faculty of Science, Kyushu University, Series A, Mathematics*, 24:47–75, 1970.

[4] S. Arikawa, T. Shinohara, and A. Yamamoto. Learning elementary formal systems. *Theoretical Computer Science*, 95:97–113, 1992.

[5] S. Matsumoto and A. Shinohara. Learning pattern languages using queries. *Proc. EuroCOLT-97, Springer-Verlag, LNAI 1208*, pages 185–197, 1997.

[6] S. Matsumoto, A. Shinohara, H. Arimura, and T. Shinohara. Learning subsequence languages. In *Information Modelling and Knowledge Bases VIII*, pages 335–344. IOS Press, 1997.

[7] S. Matsumoto, T. Shoudai, T. Miyahara, and T. Uchida. Learning of finite unions of tree patterns with internal structured variables from queries. *Proc. AI-2002, Springer LNAI 2557*, pages 523–534, 2002.

[8] H. Sakamoto, K. Hirata, and H. Arimura. Learning elementary formal systems with queries. *Theoretical Computer Science*, 298:21-50, 2003.

[9] M. Sato, Y. Mukouchi, and D. Zheng. Characteristic sets for unions of regular pattern languages and compactness. In *Proc. ALT-98, Springer-Verlag, LNAI 1501*, pages 220–233. Springer, 1998.

[10] T. Shinohara. Inductive inference of formal systems from positive data. *Bulletin of Informatics and Cybernetics*, 22:9–18, 1986.

[11] J. Uemura and M. Sato. Learning of erasing primitive formal systems from positive examples. In *Proc. ALT-2003, Springer-Verlag, LNAI 2842*, pages 69–83. Springer-Verlag, 2003.