Learning of Elementary Formal Systems with Two Clauses Using Queries*

Hirotaka KATO†, Nonmember, Satoshi MATSUMOTO‡, and Tetsuhiro MIYAHARA§, Members

SUMMARY 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 $\Gamma$. We consider a new form of EFS, called a restricted two-clause EFS, and denote by $rEFS$ the set of all restricted two-clause EFSs. Then we 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 in $rEFS$ of learning. 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.

key words: computational learning theory, query learning, learning of logic programs

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 a definite clause) in EFS 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. 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 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 $p(xby) \leftarrow p(x)$ are examples of rules. $\Gamma_1 = \{p(ab) \leftarrow, p(xby) \leftarrow p(x)\}$ 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 $L(\Gamma, p)$ generated by an EFS $\Gamma$ and a unary predicate symbol $p$ is the set of all constant strings $w$ such that $p(w)$ is provable 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, $L(\Gamma_1, p) = \{a^ib^n \mid n \geq 1\}$, $L(\Gamma_2, p) = \{a^ib^j \mid w \in \Sigma^*, n \geq 1\}$, and $L(\Gamma_3, p) = \{aab, abb, acb, aabbaabc, aabbbaabc, aacbbbaabc, \cdots\}$.

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 at most once in the pattern. In this paper, we define a restricted two-clause EFS $EFS$ as follows: $\Gamma$ consists of one or two definite clauses of the form $p(\pi) \leftarrow$, or $\Gamma$ consists of two definite clauses of the forms $p(\pi') \leftarrow$ and $p(\tau) \leftarrow p(x_1), \cdots, p(x_n)$, where $\pi$, $\pi'$ and $\tau$ are regular patterns, $p$ is a unary predicate symbol, $\pi'$ contains at least one variable, and $x_1, \cdots, x_n$ ($n \geq 1$) are all of the variables appearing in $\tau$. The target class of learning in this paper is $rEFS$, which is the set of all restricted two-clause EFSs.

Since we fix a unary predicate symbol $p$, we can identify $p(\pi) \leftarrow$ and $\pi$. Thus, the class of regular patterns and the class of sets of two regular patterns are subclasses of $rEFS$. Regular patterns are used in practical fields such as genome informatics. Since $rEFS$ contains regular patterns and EFSs with recursive rules, we consider the learnability of $rEFS$.

In the above examples, $\Gamma_1$ is not in $rEFS$ since the pattern $ab$ contains no variable. $\Gamma_2$ and $\Gamma_3$ are in $rEFS$. Since we deal with the class $rEFS$, where a unary predicate symbol, say $p$, is fixed, we denote $L(\Gamma, p)$ by $L(\Gamma)$ simply.

Let $\Gamma_1$ be an EFS in $EFS$ to be identified by a learning algorithm, and we say that the EFS $\Gamma_1$ 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 consider the following two oracles in this paper. (1) Superset oracle: The input is an EFS $\Gamma$ in $EFS$. 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 $\Sigma^*$. The output is “yes” if
We denote by \( \Gamma \) the set of all restricted superset queries [2]. We showed that regular patterns are exactly learnable in polynomial time using membership and equivalence queries [6]. Moreover, we showed that any finite set of tree patterns is 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 \( rEFS \), under the learning model of inductive inference of positive examples without allowing the empty string to be substituted for variables. The paper [11] extended this learnability by allowing the 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.

This paper is organized as follows. In Sect. 2, we explain EFSs and their languages and give the definition of \( rEFS \) which is our target class of learning. In Sect. 3, we give our exact learning model. In Sect. 4, we show that the class \( rEFS \) is exactly identifiable in polynomial time using membership and superset queries. In Sect. 5, we give a hardness result of learning \( rEFS \) in the 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 \( w[i] \) \( j \)-th symbol in string \( w \), and by \( w[i : j] \) the substring \( w[i] \cdots w[j] \) of \( w \). We define \( w[i : j] = \epsilon \) (the empty string) if \( i > j \). For convenience, a prefix \( w[i : i] \) is abbreviated as \( w[i] \), and a suffix \( w[i : |w|] \) as \( w[i : j] \), where \( 1 \leq i < |w| \). We denote by \( \Sigma^* \) the set of all nonempty strings over \( \Sigma \).

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.

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 pattern. A substitution \( \theta \) is denoted by \( \{x_1 := \pi_1, \ldots, x_n := \pi_n\} \), where \( x_1, \ldots, x_n \) are mutually distinct variables and \( \pi_1, \ldots, \pi_n \) are patterns. By \( \pi \theta \), 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 \( C \theta = A \theta \leftarrow B_1 \theta, \ldots, B_m \theta \).

Example 1: Let \( \Sigma = \{a,b,c\} \) and \( X = \{x,y,z\} \). Let \( \pi_1 = axbya, \pi_2 = baxyca \) be patterns and \( \theta = \{x := cc, y := ab\} \) a substitution. \( \pi_1 \theta = acbcabza \) and \( \pi_2 \theta = bccazbbca \).

For patterns \( \pi \) and \( \tau \), we introduce binary relations \( = \) and \( \equiv \) as follows: \( \pi \equiv \pi \) if \( \pi = \pi \theta \) for some substitution \( \theta \), and \( \pi = \pi \theta \) if \( \pi \subseteq \pi \) and \( \pi = \pi \theta \). If \( \pi \subseteq \pi \) and \( \pi 
eq \pi \), then we write \( \pi \neq \pi \).

Example 2: Let \( \pi_1 = axbya, \pi_3 = acbya \) and \( \pi_4 = azbxa \) be regular patterns. Since there is a substitution \( \theta_1 = \{x := c\} \) with \( \pi_1 = \pi_1 \theta_1 \), and there is no substitution \( \theta_2 \) with \( \pi_1 = \pi_3 \theta_2 \), we have \( \pi_3 \neq \pi_1 \). It is clear that \( \pi_1 \equiv \pi_4 \).

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

For a pattern \( \pi \), we denote by \( S_i(\pi) \) 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 \( S(P) = \bigcup_{\pi \in P} S_i(\pi) \). Let \( T, T' \) be nonempty sets of patterns. We write \( T \subseteq T' \) if for any pattern \( \pi \in T \), there exists a pattern \( \pi' \in T' \) such that \( \pi \leq \pi' \). If \( T \subseteq T' \) and \( T' \not\subseteq T \), then we write \( T \prec T' \).

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 as in the way of usual logic programming. We define the language \( L(\Gamma, p) = \{w \in \Sigma^* | \Gamma \vdash p(w)\} \), where \( p \) is a unary predicate symbol. A language \( L \) is an EFS language if \( L = L(\Gamma, p) \) for some EFS \( \Gamma \) and some unary predicate symbol \( p \). In particular, a language \( L \) is a regular pattern language if \( L = L(\Gamma, p) \) for some EFS \( \Gamma \).

Definition 2: We define a restricted two-clause EFS \( \Gamma \) as follows: \( \Gamma \) consists of one or two clauses of the form \( p(\pi') \leftarrow \), or \( \Gamma \) consists of two clauses of the forms \( p(\pi') \leftarrow \) and \( p(\pi) \leftarrow p(x_1), \ldots, p(x_n) \), where \( \pi, \pi' \) and \( \tau \) are regular patterns, \( p \) is a unary predicate symbol, \( \pi' \) contains at least one variable, and \( x_1, \ldots, x_n (n \geq 1) \) are all of the variables appearing in \( \pi \). We denote by \( rEFS \) the set of all restricted two-clause EFSs.
We will explain the reason why we need the condition that $\pi'$ contains at least one variable before the proof of Theorem 2. By the above definition, the class of regular patterns and the class of sets of two regular patterns are subclasses of $rEFS$.

For an EFS $\Gamma$ in $rEFS$, we define the size of $\Gamma$, denoted by $|\Gamma|$, as follows: (1) $|\Gamma| = |\pi|$ if $\Gamma = \{p(\pi) \leftarrow \}$, (2) $|\Gamma| = |\pi| + |\tau|$ if $\Gamma = \{p(\pi_1) \leftarrow, p(\pi_2) \leftarrow \}$, (3) $|\Gamma| = |\pi| + |\tau|$ if $\Gamma = \{p(\pi) \leftarrow, \tau(\pi) \leftarrow (x_1, \ldots, x_n)\}$.

Example 3: Let $\Gamma_1 = \{p(ab) \leftarrow, p(abx) \leftarrow (x), \Gamma_2 = \{p(abx) \leftarrow, p(abx) \leftarrow (y), \Gamma_3 = \{p(abx) \leftarrow, p(abxy) \rightarrow (y), p(z)\}, \Gamma_4 = \{p(abx) \rightarrow \}$ and $\Gamma_5 = \{p(ax) \rightarrow, p(axa) \rightarrow\}$. Since $ab$ has no variable, $\Gamma_1$ is not in $rEFS$. $\Gamma_2, \Gamma_3, \Gamma_4$ and $\Gamma_5$ are EFSs in $rEFS$.

Languages generated by $\Gamma_2, \Gamma_3, \Gamma_4$ and $\Gamma_5$ are as follows: $L(\Gamma_2, p) = \{a^{n}b^{n}w | w \in \Sigma^{*}, n \geq 1\}$, $L(\Gamma_3, p) = \{a^{n}b^{n}c^{n}w | w \in \Sigma^{*}\}$, $L(\Gamma_4, p) = \{a^{n}b^{n}w | w \in \Sigma^{*}\}$ and $L(\Gamma_5, p) = \{a^{n}w_{1}w_{2} | w_{1}, w_{2} \in \Sigma^{*}\}$.

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 $\Gamma'$ in $\Gamma$.

Example 4: Let $\Sigma = \{a, b, c\}$ and $X = \{x, y, z, \ldots\}$. Let $\Gamma_6 = \{p(axa) \leftarrow, p(aya) \leftarrow (y), \Gamma_7 = \{p(axb) \leftarrow, p(byb) \leftarrow (y), \Gamma_8 = \{p(bxya) \leftarrow, p(bazba) \leftarrow (z)\}$ and $\Gamma_9 = \{p(bxya) \leftarrow, p(bzabb) \leftarrow (z)\}$. Since $bacbb \in L(\Gamma_7, p)$ and $bacbb \notin L(\Gamma_7 - \{p(byb) \leftarrow (y), p\}, p)$, $\Gamma_7$ is reduced. Since $baccbab \notin L(\Gamma_8, p)$ and $baccbab \notin L(\Gamma_8 - \{p(azba) \leftarrow (z), p\}, p)$, $\Gamma_8$ is not reduced.

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., $\Gamma = \{\pi\}$) an EFS $\Gamma = \{p(\pi) \leftarrow, p(\tau) \leftarrow (x_1, \ldots, x_n)\}$ (resp., $\Gamma = \{p(\pi) \leftarrow\}$). Moreover, by $L((\pi, \tau))$ (resp., $L(\{\pi\})$) we denote $L((\pi(\pi) \leftarrow, p(\tau) \leftarrow (x_1, \ldots, x_n))$ (resp., $L((\pi(\pi) \leftarrow)$).

Example 5: For EFSs $\Gamma_2 = \{p(abx) \leftarrow, p(ab) \rightarrow (y)\}$, $\Gamma_3 = \{p(abx) \rightarrow, p(abxy) \rightarrow (y), p(z)\}$, $\Gamma_4 = \{p(abx) \rightarrow\}$ and $\Gamma_5 = \{p(ax) \rightarrow, p(axa) \rightarrow\}$, we write $\Gamma_2 = (abx, ayb)$, $\Gamma_3 = (ax, ayb, ayzb)$, $\Gamma_4 = (axb)$ and $\Gamma_5 = (axy, axa)$ simply.

For $\Gamma = (\pi, \tau)$, we denote by $\tau_{\pi}$ the regular pattern which is generated by the following two operations: (1) All variables in $\tau$ are replaced with the pattern $\pi$. (2) All variables in the pattern generated in (1) are replaced with new distinct variables. Thus, $\tau_{\pi}$ is always regular. It is clear that $|\tau_{\pi}| \leq |\tau_{\pi}| \leq |\tau||\pi|$. $\tau_{\pi}$ is defined in a similar way.

Example 6: Let $\Gamma_{10} = (abxb, aybzx)$ be an EFS in $rEFS$, $\pi = abxb$ and $\tau = aybzx$. We have $\tau_{\pi} = aabxb_{1}bbabx_{2}bc$. Note that $\tau_{\pi}$ is a regular pattern.

Let $\Gamma = (\pi, \tau)$ be a reduced EFS in $rEFS$. $x_1, \ldots, x_n$ all of the variables appearing in $\tau$. $\Gamma_{10}$ is recursively defined as follows: $\Gamma_{[1]} = \{\tau_{\pi}\}$ and for any positive integer $t \geq 2$, $\Gamma_{[t]} = \Gamma_{[t-1]} \cup \{\pi(x_1 := \zeta_1, \ldots, x_n := \zeta_n) | \zeta_i \in \Gamma_{[t-1]} \cup \{\pi, i = 1, \ldots, n\}\$. We define $\Gamma_{\pi} = \bigcup_{i=2}^{\infty} \Gamma_{[i]}$. Note that $\pi$ is not included in $\Gamma_{\pi}$. Thus, $L(\Gamma_{\pi}) \subseteq L(\pi, \tau)$.

A primitive EFS $\Gamma$, a PFS for short, is defined in [10] as follows: $\Gamma$ consists of two clauses of the forms $p(\pi) \leftarrow$ and $p(\tau) \leftarrow (x_1, \ldots, x_n)$, where $\pi, \tau$ are regular patterns, $p$ is a unary predicate symbol, and $x_1, \ldots, x_n$ are all of the variables appearing in $\tau$.

A language $\{a^{n}b^{m}w \in \Sigma^{n} | n \geq 1\}$ is generated by a PFS $\Gamma_{10} = \{p(ab) \leftarrow, p(abx) \leftarrow (x), (\{x\})\}$, but there is no restricted two-clause EFS which generates the language.

In case of substitutions where a variable is allowed to be replaced with the empty string, Uemura and Sato showed the following theorem in [11]. The theorem holds for any $\pi$ in $rEFS$ in case of standard substitutions.

Theorem 1: [11] Let $\Gamma = (\pi, \tau)$ be a PFS. The following statements are equivalent: (i) $\Gamma$ is reduced. (ii) $L(\pi) \cap L(\Gamma_{\pi}) = \emptyset$, where $L(\Gamma_{\pi}) = \bigcup_{i=2}^{\infty} L(\Gamma_{[i]})$.

3. Learning Model

In this paper, let $\Gamma$ be an EFS in $rEFS$ to be identified, and we say that the EFS $\Gamma$ is a target. Non-reduced EFSs have redundant definite clauses. Even if we consider only reduced EFSs, the expressive power of EFSs is the 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 $L(\Gamma_{\pi})$. We consider the following oracles. (1) Superset oracle $Sup_{\pi}$: The input is an EFS $\Gamma$ in $rEFS$. If $L(\Gamma) \supseteq L(\Gamma_{\pi})$, then the output is “yes”. Otherwise, it returns a counterexample $t \in L(\Gamma_{\pi}) - L(\Gamma)$. The query is called a superset query. (2) Subset oracle $Sub_{\pi}$: The input is an EFS $\Gamma$ in $rEFS$. If $L(\Gamma) \subseteq L(\Gamma_{\pi})$, then the output is “yes”. Otherwise, it returns a counterexample $t \in L(\Gamma_{\pi}) - L(\Gamma)$. The query is called a subset query. (3) Membership oracle $Mem_{\pi}$: The input is a string $t \in \Sigma^{*}$. The output is “yes” if $t \in L(\Gamma_{\pi})$, and “no” otherwise. The query is called a membership query. (4) Equivalence oracle $Eqv_{\pi}$: The input is an EFS $\Gamma$ in $rEFS$. The output is “yes” if $L(\Gamma) = L(\Gamma_{\pi})$. Otherwise, it returns a counterexample $t \in L(\Gamma_{\pi}) - L(\Gamma)$. The query is called an equivalence query.

A learning algorithm $\mathcal{A}$ collects information about $L(\Gamma_{\pi})$ by using queries and outputs an EFS $\Gamma$ in $rEFS$. We say that a learning algorithm $\mathcal{A}$ exactly identifies a target $\Gamma_{\pi}$ in polynomial time using certain types of queries if $\mathcal{A}$ halts in polynomial time with respect to $|\Gamma_{\pi}|$ and outputs an EFS
\[ \Gamma \in \mathcal{EFS} \text{ such that } L(\Gamma) = L(\Gamma_\ast) \text{ using queries of the specified types.} \]

4. Learning of Restricted EFSs Using Queries

Let \( \Gamma_\ast \) be a target EFS in \( \mathcal{EFS} \). We give the following theorem as one of the main results of this paper. Some lemmas and theorem needed in Theorem 2 are stated after the theorem.

**Theorem 2:** There exists a learning algorithm which identifies any EFS \( \Gamma_\ast \in \mathcal{EFS} \) in polynomial time using \( \mathcal{O}(|\Gamma|, \ell^3) \) membership queries and \( \mathcal{O}(|\Gamma|, \ell^2) \) superset queries, where \( |\Sigma| \geq 5 \).

For the target \( \Gamma_\ast \), we consider the following cases: (i) \( \Gamma_\ast = \{ \pi_\ast \} \). (ii) \( \Gamma_\ast = \{ \pi_\ast, \tau_\ast \} \) and the length of \( \pi_\ast \) is the same as \( \tau_\ast \), that is, \( |\pi_\ast| = |\tau_\ast| \). (iii) \( \Gamma_\ast = \{ \pi_\ast, \tau_\ast \} \) and the length of \( \pi_\ast \) is not the same as \( \tau_\ast \), that is, \( |\pi_\ast| \neq |\tau_\ast| \). Without loss of generality, we assume \( |\pi_\ast| < |\tau_\ast| \). (iv) \( \Gamma_\ast = \{ \pi_\ast, \tau_\ast \} \).

When \( \Gamma_\ast \) is in the cases (i), (ii) or (iii), we can regard \( \Gamma_\ast \) as a set of at most two regular patterns. Since we use Theorem 3 for some lemmas and theorems, we assume that \( |\Sigma| \geq 5 \) in this paper.

**Theorem 3:** [9] Let \( k \) be a positive integer. 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) \( S_1(P) \subseteq L(Q) \).

By using the procedure \( \text{LENGTH}1 \) of Fig. 1 and \( \text{LEARN}_{\text{PI}} \) of Fig. 2, the algorithm \( \text{LEARN}_{\text{REFS}} \) of Fig. 8 decides whether the target is in the case (i) or (ii), or whether the target is in the case (iii) or (iv). This is shown by using Lemma 1, 2, 3 and the next condition.

Let \( \pi \) be a regular pattern. We define the following condition, called \( \text{Condition A} \), as follows:

**A-1** \( \pi \) satisfies \( L(\Gamma_\ast) \subseteq L(\pi, x_1 x_2 \cdots x_{|\pi|+1}) \) and \( |\pi| = |\pi_\ast| \), and

**A-2** For \( \pi \), there is no regular pattern \( \pi' \) such that \( |\pi'| = |\pi| \), \( \pi' \prec \pi \) and \( L(\Gamma_\ast) \subseteq L(\pi', x_1 x_2 \cdots x_{|\pi|+1}) \).

The procedure \( \text{LENGTH}1 \) outputs a positive integer \( \ell \) with \( \ell = \min \{|w| \mid w \in L(\Gamma_\ast)\} \), that is, \( \ell = |\pi_\ast| \). The procedure uses \( \mathcal{O}(|\pi_\ast|, \ell^3) \) superset queries, and runs in \( \mathcal{O}(|\pi_\ast|, \ell^2) \) time.

The procedure \( \text{LEARN}_{\text{PI}} \) takes as input a positive integer \( \ell \) with \( \ell = |\pi_\ast| \), and outputs a regular pattern \( \pi \) such that \( |\pi| = |\pi_\ast| = \ell \) and \( L(\Gamma_\ast) \subseteq L(\pi, x_1 x_2 \cdots x_{\ell+1}) \).

We omit the proofs of some lemmas easy to prove.

**Lemma 1:** Let \( \ell \) be an input positive integer and \( \pi \) an output regular pattern by the procedure \( \text{LEARN}_{\pi} \). Then, there is no regular pattern \( \pi' \) such that \( |\pi'| = |\pi| \), \( \pi' \prec \pi \) and \( L(\Gamma_\ast) \subseteq L(\pi', x_1 x_2 \cdots x_{\ell+1}) \).

By Lemma 1, a regular pattern \( \pi \) output by the procedure \( \text{LEARN}_{\pi} \) satisfies \( \text{Condition A} \). The procedure uses \( \mathcal{O}(|\pi_\ast|, \ell^3) \) superset queries, and runs in \( \mathcal{O}(|\pi_\ast|, \ell^2) \) time.

**Lemma 2:** Let \( \pi \) be a regular pattern satisfying \( \text{Condition A} \). If \( \Gamma_\ast \) is not in the case (ii), then \( \pi \equiv \pi_\ast \).

**Lemma 3:** Let \( \pi \) be a regular pattern satisfying \( \text{Condition A} \). Then, the following statements hold. (1) If \( L(\Gamma_\ast) \subseteq L(|\pi_\ast|), \) then \( \Gamma_\ast \) is in the case (i) or (ii). (2) If \( L(\Gamma_\ast) \not\subseteq L(|\pi_\ast|) \), then \( \Gamma_\ast \) is in the case (iii) or (iv).

**Proof.** We show each case.

(1) We assume that \( \Gamma_\ast \) is in the case (iii). By Lemma 2, \( \pi \equiv \pi_\ast \). Since \( \pi \equiv \pi_\ast \) and \( L(\Gamma_\ast) \subseteq L(|\pi|) \), we have \( L(\pi, x_1 \cdots x_{|\pi|+1}) \) is not in the case (ii). This is a contradiction. Thus, \( \Gamma_\ast \) is not in the case (iii).

(2) We assume that \( \Gamma_\ast \) is in the case (ii). By Condition A-1 and Theorem 3, since \( L(\pi_\ast, x_1 \cdots x_{|\pi|+1}) \), \( \pi \leq \pi_\ast \) or \( \pi_\ast \leq \pi \). Since \( L(\Gamma_\ast) \not\subseteq L(\pi) \), we have \( \pi \neq \pi_\ast \) or \( \pi_\ast \neq \pi \). This is a contradiction. Thus, \( \Gamma_\ast \) is not in the case (ii).

By using the procedure \( \text{LEARN}_{\pi} \) of Fig. 3 and \( \text{LEARN}_{\pi, \tau} \) of Fig. 4, the algorithm \( \text{LEARN}_{\text{REFS}} \) decides whether the target is in the case (i) or whether the target is in the case (ii). This is shown by using Lemma 4, 5, 6 and the next condition.

For a regular pattern \( \pi \), we define the following condition, called \( \text{Condition B} \), as follows:

**B-1** \( \pi \) satisfies \( \text{Condition A} \) and \( L(\Gamma_\ast) \subseteq L(|\pi|) \), and

---

**Procedure \( \text{LENGTH}1 \)**

*Given:* An oracle \( \text{Sup}_{\pi_\ast} \) for the target \( \Gamma_\ast \);

*begin*

\[ \ell := 1; \]

// A pattern \( x_1 x_2 \cdots x_{|\pi_\ast|+1} \) is a regular pattern.

*while* \( \text{Sup}_{\pi_\ast}((x_1 x_2 \cdots x_{|\pi_\ast|+1})) = \text{"yes"} \) *do*

\[ \ell := \ell + 1; \]

*output* \( \ell \);

*end.*

---

**Procedure \( \text{LEARN}_{\pi} \)**

*Input:* A positive integer \( \ell \) with \( \ell = |\pi_\ast| \);

*Given:* An oracle \( \text{Sup}_{\pi_\ast} \) for the target \( \Gamma_\ast \);

*begin*

\[ \pi := x_1 x_2 \cdots x_{|\pi_\ast|}; \]

*for* \( i := 1 \) to \( \ell \) *do*

*foreach* \( \alpha \in \Sigma \) *do*

\[ \pi' := \pi \cdot \alpha; \]

*if* \( \text{Sup}_{\pi_\ast}((\pi', x_1 x_2 \cdots x_{|\pi_\ast|+1})) = \text{"yes"} \) *then*

\[ \pi := \pi'; \]

*break;*

*end;*

*end;*

*output* \( \pi \);

*end.*
Procedure LEARN.PI.TAU1(\pi)
Input: A regular pattern \pi satisfying
Condition A and L(\Gamma_\pi) \subseteq L(\pi);
Given: An oracle Sup_{\pi}, for the target \Gamma_\pi;
begin
\tau := 1;
while ((\tau; \leq |\pi|) and (\pi[\tau]; \in \Sigma)) do
\tau := \tau + 1;
while \tau; \leq |\pi| do begin
\tau := \tau + 1;
end;
end;
output "yes";
end.

---

Procedure LEARN.PI.TAU2(\pi', \tau')
Input: Regular patterns \pi' and \tau' satisfying Condition B-2 for \pi,
where \pi satisfies Condition B;
Given: An oracle Sup_\pi, for the target \Gamma_\pi;
begin
\pi'' := \pi'; \tau := 1;
while ((\tau; \leq |\pi''|) and (\pi''[\tau]; \in \Sigma)) do
\tau := \tau + 1;
while \tau; \leq |\pi''| do begin
\tau' := \tau'; \tau := 1;
while ((\tau; \leq |\pi''|) and (\pi''[\tau]; \in \Sigma)) do
\tau := \tau + 1;
end;
end;
output \pi'', \tau';
end.

---

**Fig. 3** Procedure LEARN.PI.TAU1.

**Fig. 4** Procedure LEARN.PI.TAU2.

B-2 For \pi, there are regular patterns \pi' and \tau' such that

\[
|\pi'| = |\tau'| = |\pi|, \pi' < \pi, \tau' < \pi \quad \text{and} \quad L(\Gamma_\pi) \subseteq L((\pi', \tau')).
\]

The procedure LEARN.PI.TAU1 takes as input a regular pattern \pi satisfying Condition A and L(\Gamma_\pi) \subseteq L(\pi). If there are regular patterns \pi' and \tau' satisfying Condition B-2 for \pi, the procedure outputs such regular patterns. Otherwise, the procedure outputs "no". The procedure uses \(O(|\pi|)^2\) superset queries, and runs in \(O(|\pi|)^2\) time.

**Lemma 4:** Let \pi be a regular pattern satisfying Condition A and L(\Gamma_\pi) \subseteq L(\pi)). Then, the following statements hold.

1. If there are no regular patterns satisfying Condition B-2 for \pi, then \Gamma_\pi is in the case (i).
2. If there are regular patterns satisfying Condition B-2 for \pi, then \Gamma_\pi is in the case (ii).

**Proof:** By Lemma 3, \Gamma_\pi is in the case (i) or (ii). We show each case.

1. We assume that \Gamma_\pi is in the case (ii). By Condition A, we have |\pi| = |\pi| = |\pi|.

2. We assume that \Gamma_\pi is in the case (i). By Condition B-2, there are regular patterns \pi' and \tau' such that |\pi'| = |\tau'| = |\pi|.
\( \tau_i \leq \tau_j \). We assume \( \tau_i \neq \tau_j \). Since \( \tau_i \neq \tau_j \), we have \( \tau_i \neq \tau_j \). Since \( |\tau_i| = |\tau_j| \), \( \tau_i \) satisfies \( |x_i| = |\tau_i| \), \( \tau_i \neq \tau_j \) and \( \Gamma(\tau_i) = L((\tau_i, \tau_j)) \subseteq L((\tau_i, \tau_j)) \). This is a contradiction. We have \( \tau_i \neq \tau_j \). We can show \( \tau_i \equiv \tau_j \) in a similar way. Therefore, \( \tau_i \equiv \tau_j \) and \( \tau_j \equiv \tau_i \). Next, let \( \tau_i \equiv \tau_j \). \( \tau_j \equiv \tau_j \). We can show \( \tau_i \equiv \tau_j \) and \( \tau_j \equiv \tau_i \) in a similar way. □

**Lemma 6:** Let \( \tau'' \) and \( \tau''' \) be regular patterns output by the procedure LEARN.PI.TAU2. There are no regular patterns \( \tau'' \) and \( \tau''' \) such that \( |\tau''| = |\tau'''| \), \( \tau'' < \tau''' \), \( \Gamma(\tau_i) \subseteq L((\tau''', \tau''')) \). By Lemma 5 and Lemma 6, regular patterns \( \tau'' \) and \( \tau''' \) output by the procedure LEARN.PI.TAU2 satisfy \( \tau'' \equiv \tau''' \) and \( \tau_i \equiv \tau'' \). or \( \tau_i \equiv \tau''' \) and \( \tau_j \equiv \tau'' \). The procedure uses \( O(|\tau_i| + |\tau_j|) \) superset queries, and runs in \( O(|\tau_i|^2 + |\tau_j|^2) \) time. Since \( |\tau_i| = |\tau_j| \), it uses \( O(|\tau_i|) \) superset queries, and runs in \( O(|\tau_i|^2) \) time.

By using the procedure LENGTH2 of Fig.5 and LEARN.TAU1 of Fig.6, the algorithm LEARN.REFS decides whether the target is in the case (iii) or whether the target is in the case (iv). This is shown by using Lemma 7–13 and the next condition.

For a regular pattern \( \tau \), we define the following condition, called Condition C, as follows:

**C-1** \( \tau \) satisfies Condition A and \( \Gamma(\tau_i) \not\subseteq L(|\tau|) \), and

\[ \tau \subseteq \tau' \]. We assume \( \tau' \neq \tau \). Since \( \tau' \neq \tau \), we have \( \tau' \neq \tau \). Since \( |\tau'| = |\tau| \), \( \tau \) satisfies \( |\tau| = |\tau'| \), \( \tau' \neq \tau' \) and \( \Gamma(\tau') = L((\tau', \tau')) \subseteq L((\tau', \tau')) \). This is a contradiction. We have \( \tau' \neq \tau \). We can show \( \tau' \equiv \tau \) in a similar way. Therefore, \( \tau' \equiv \tau \) and \( \tau \equiv \tau' \). Next, let \( \tau \equiv \tau' \) and \( \tau' \equiv \tau \). We can show \( \tau' \equiv \tau \) and \( \tau \equiv \tau' \) in a similar way. □

**C-2** For \( \tau \), there is a regular pattern \( \tau' \) satisfying the following conditions:

**C-2-1** There is a shortest string \( \nu \in \Gamma(\tau) \setminus L(|\tau|) \) such that \( |\nu| = |\tau| \) and \( \nu \leq \tau \).

**C-2-2** \( \Gamma(\tau') \subseteq \Gamma(\tau) \).

**C-2-3** There is no regular pattern \( \tau' \) such that \( |\tau'| = |\tau| \), \( \tau < \tau' \) and \( \Gamma(\tau') \subseteq \Gamma(\tau) \).

The procedure LENGTH2 takes as input a regular pattern \( \tau \) satisfying Condition A and \( \Gamma(\tau) \not\subseteq L(|\tau|) \), and outputs a shortest string \( \nu \in \Gamma(\tau) \setminus L(|\tau|) \), where \( \nu \) is a counterexample output by \( Sup_{\Gamma} \). By Lemma 3, \( \Gamma \) is in (ii) or (iv). By Lemma 7, \( |\nu| = |\tau| \) in the case (iii), and \( |\nu| = |\tau| \) in the case (iv). Thus, the procedure uses \( O(|\tau|) \) superset queries, and runs in \( O(|\tau|^2) \) time.

**Lemma 7:** Let \( \tau \) be a regular pattern satisfying Condition A and \( \Gamma(\tau) \not\subseteq L(|\tau|) \). Let \( \nu \) be a shortest string in \( \Gamma(\tau) \setminus L(|\tau|) \). Then, the following statements hold. (1) If \( \Gamma(\tau) \) is in the case (ii), then \( |\nu| = |\tau| \). (2) If \( \Gamma(\tau) \) is in the case (iv), then \( |\nu| = |\tau| \).

**Proof.** By Lemma 3, \( \Gamma \) is in the case (iii) or (iv). By Lemma 2, \( \pi \equiv \pi \). We show each case.

(1) For a shortest string \( \nu \in \Gamma(\tau) \setminus L(|\tau|) \), \( |\nu| \leq |\nu| \). We assume \( |\nu| < |\nu| \). It implies \( S_1(\nu) \subseteq \Gamma(\nu) \). By Theorem 3, \( \sigma \equiv \pi \). Since \( \pi \equiv \pi \), we have \( \tau \equiv \tau \). This contradicts that \( \Gamma(\tau) \) is reduced. Thus, we have \( |\nu| = |\nu| \).

(2) By Theorem 1, since \( \Gamma(\tau) \) is reduced, \( \Gamma(\tau) \cap \Gamma(\tau') = \emptyset \). Since \( \Gamma(\tau) \cap \Gamma(\tau') = \emptyset \) and \( \pi \equiv \pi \), the length of shortest strings in \( \Gamma(\tau) \setminus L(|\tau|) \) is \( |\tau| \).

By using Lemma 8 and Lemma 10, we can show Lemma 11.

**Lemma 8:** Let \( \Gamma(\tau) \) be a reduced EFS in \( \mathcal{KFS} \) and \( \nu \) be a shortest string in \( \Gamma(\tau) \setminus L(|\tau|) \). Then, for a positive integer \( i \) (0 ≤ \( i \) ≤ |\nu|), the following statements are equivalent: (1) There is a symbol \( \alpha \in \Sigma \) such that \( \alpha \neq w[i] \) and \( w[i] \in \Gamma(\tau) \) and \( w[i+1] \in \Gamma(\tau) \). (2) \( \tau \cdot [\tau] \cdot [\nu] = \tau \).

**Proof.** Since \( \nu \) is a shortest string in \( \Gamma(\tau) \setminus L(|\tau|) \), \( |\nu| \geq |\nu| \). We assume \( |\nu| < |\nu| \). It implies \( S_1(\nu) \subseteq \Gamma(\nu) \). Thus, \( \Gamma(\tau) \cap \Gamma(\tau') \neq \emptyset \). By Theorem 1, this contradicts that \( \Gamma(\tau) \) is reduced. Therefore, \( |\nu| = |\nu| \).

(1) \( (1) \Rightarrow (2) \) From \( w \in \Gamma(\tau) \setminus L(|\tau|) \), \( w \in \Gamma(\tau') \). Since \( w \in \Gamma(\tau') \) and \( |\nu| = |\nu| \), \( w \leq |\nu| \). We can show \( w[i] \in \Gamma(\tau) \) in a similar way. Since \( |\nu| = |\nu| \), \( w \leq |\nu| \), and \( w[i] \leq |\nu| \), we have \( |\nu| \).

(2) \( (2) \Rightarrow (1) \) From \( |\nu| \), \( |\nu| \) in \( X \), \( w[i] \) in \( \Gamma(\tau') \) for some symbol \( \alpha \in \Sigma \). It implies that \( w[i] \in \Gamma(\tau) \). Since \( \Gamma(\tau) \) is reduced, we have \( \Gamma(\tau) \cap \Gamma(\tau') = \emptyset \). Thus, there is a string \( w[i] \in \Gamma(\tau) \).

To show Lemma 10, we need the following lemma. Note that we assume \( |\Sigma| \geq 5 \) in this paper.
Lemma 9: [9] Let \(|\Sigma| \geq 3\), \(\pi\) and \(\tau\) regular patterns, and \(x\) a variable in \(\pi\). If \(\pi[x := a] \preceq \tau\), \(\pi[x := b] \preceq \tau\) and \(\pi[x := c] \preceq \tau\) for symbols \(a\), \(b\) and \(c\) in \(\Sigma\) which are mutually distinct, then \(\pi \preceq \tau\).

Lemma 10: Let \(\Gamma = \{\pi_a, \tau\}\) be a reduced EFS in \(rEFS\) with \(|\pi_a| < |\tau|\), and \(w\) a shortest string in \(L(\Gamma) - L(\pi_a)\). Then, for a positive integer \(i\) (\(1 \leq i \leq |w|\)), the following statements are equivalent: (1). There is a symbol \(a \in \Sigma\) with \(\pi \neq w[i]\) and \(w_{ia} \in L(\Gamma) - L(\pi_a)\). (2). \(\tau_{i}[i] \in X\).

Proof. Since \(w \in L(\Gamma) - L(\pi_a)\), \(|w| \geq |\tau|\). We assume \(|w| > |\tau|\). It implies \(S_1(\tau, \pi) \subseteq L(\pi_a)\). By Lemma 3, \(\tau \preceq \pi\). This contradicts that \(\Gamma\) is reduced. Thus, \(|w| = |\tau|\). (1) \(\Rightarrow\) (2) We can show in a similar way as Lemma 8.

((2) \(\Rightarrow\) (1)) From \(\tau, [i] \in X\), \(w_{ia} \in L(\pi_a)\) for any symbol \(a \in \Sigma\). It implies \(w_{ia} \in L(\Gamma)\). We assume \(w_{ia} \in L(\pi_a)\) for any symbol \(a \in \Sigma\) with \(\pi \neq w[i]\). Let \(\pi' = w[i] - 1\). It is clear that \(w \preceq \pi'\). Since \(|\Sigma| \geq 5\), there are symbols \(a\), \(b\) and \(c\) in \(\Sigma\) such that \(a, b, c\) and \(w[i]\) are mutually distinct.

By Lemma 9, \(w_{ia} = \pi'\) is a\(\{x := a, w_{ia} = \pi'\} \preceq \pi\). From \(w_{ia} \in L(\pi_a)\), we have \(\pi' \preceq \pi\). From \(w \preceq \pi'\) and \(\pi' \preceq \pi\), \(w \in L(\pi_a)\). This contradicts \(w \notin L(\pi_a)\).

Therefore, there are positive integers \(i \in \{1, \cdots, |w|\}\) and a symbol \(a \in \Sigma\) with \(w_{ia} \in L(\Gamma) - L(\pi_a)\).

Lemma 11: Let \(\tau\) be a regular pattern output by the procedure \(\text{LEARN}_{\text{TAU}1}\). There is no regular pattern \(\tau'\) such that \(\tau' = |\tau|\), \(\tau \preceq \tau'\) and \(L(\tau') \subseteq L(\tau)\).

Proof. It is clear that \(|w| = |\tau|\). We assume that there is a regular pattern \(\tau'\) such that \(|\tau'| = |\tau|\), \(\tau \preceq \tau'\) and \(L(\tau') \subseteq L(\tau)\). By Lemma 2 and Lemma 3, \(\pi \equiv \pi_{\tau}\), and \(\Gamma_{\tau}\) is in the case (iii) or (iv). At first, we assume that \(\Gamma_{\tau}\) is in the case (iii). If \(\tau' \preceq \tau\), then \(\tau \preceq \tau'\). This contradicts \(w \notin L(\pi)\). It implies \(\tau' \preceq \tau\). Since \(\tau' \preceq \tau\) and \(L(\tau') \subseteq L(\tau)\), we have \(\tau \preceq \tau_{\tau}\). By Lemma 7, \(|\tau'| = |\tau|\). Since \(|\tau'| = |\tau|\) and \(\tau \preceq \tau_{\tau}\), there is a positive integer \(i \in \{1, \cdots, |\tau|\}\) such that \(\tau[i] \in \Sigma\) and \(\tau'[i] \in X\). From \(\tau \preceq \tau'\) and \(|\tau'| = |\tau|\), \(\tau[i] \in \Sigma\) and \(\tau_{\tau}[i] \in X\). By Lemma 10, there is a symbol \(a \in \Sigma\) such that \(\alpha \neq w[i]\) and \(w_{ia} \in L(\Gamma) - L(\pi)\). This contradicts \(\tau[i] \in \Sigma\). Therefore, \(\Gamma_{\tau}\) is not in the case (iii).

In case that \(\Gamma_{\tau}\) is in the case (iv), we can show in a similar way using Lemma 8. Thus, there is no regular pattern \(\tau'\) such that \(|\tau'| = |\tau|\), \(\tau \preceq \tau'\) and \(L(\tau') \subseteq L(\tau)\).

By Lemma 11, the procedure \(\text{LEARN}_{\text{TAU}1}\) outputs a regular pattern \(\tau\) satisfying Condition C-2 for \(\pi\). Since \(|w| \leq |\tau|, |\tau_{\tau}|, |w[i]|\), it uses \(O(|\pi|, |\tau|, |w|)\) membership queries, and runs in \(O(|\pi|, |\tau|, |w|)\). time.

Lemma 12: Let \(\pi\) be a regular pattern satisfying Condition C, and \(\tau\) a regular pattern satisfying Condition C-2 for \(\pi\). Then, the following statements hold. (1). If \(\Gamma_{\tau}\) is in the case (iii), then \(\tau \equiv \tau_{\tau}\) or (ii). If \(\Gamma_{\tau}\) is in the case (iv), then \(\tau \equiv \tau_{\tau}\).

Proof. By Lemma 3, \(\Gamma_{\tau}\) is in the case (iii) or (iv). Moreover, by Lemma 2, \(\pi \equiv \pi_{\tau}\). Let \(w\) be a shortest string in \(L(\Gamma) - L(\pi)\) satisfying Condition C-2-1 for \(\tau\). We show each case.

(1). We assume \(\tau \neq \tau_{\tau}\). By Condition C-2-1 and Lemma 7,
satisfies Condition C-2 for \( \pi \), and \( L(\Gamma_\pi) \not\subseteq L((\pi, \tau)) \). In the case (iv), \( \tau \equiv (\tau_\pi)_\pi \). The reason we impose the condition in Definition 2 that \( \pi_\pi \) contains at least one variable is that this condition guarantees to construct \( \tau_\pi \) from \( \pi \) and \( (\tau_\pi)_\pi \).

By the above lemmas and a theorem, we can prove Theorem 2 as follows.

**Proof of Theorem 2.** We use the algorithm LEARN_ReFS of Fig. 8 as a learning algorithm of Theorem 2. By Lemma 1, the procedure LEARN_PI outputs a regular pattern \( \pi \) satisfying Condition A. For \( \pi \), we consider the following cases:

1. In case that \( L(\Gamma_\pi) \subseteq L((\pi)) \). By Lemma 3, \( \Gamma_\pi \) is in the case (i) or (ii). If the procedure LEARN_PI_TAU1 outputs “no”, then there are no regular patterns \( \pi' \) and \( \tau' \) satisfying Condition B-2 for \( \pi \). By Lemma 2 and Lemma 4, \( \Gamma_\pi \) is in the case (i) and \( \pi \equiv \pi_\pi \). Thus, \( L(\Gamma_\pi) = L((\pi)) \). By Lemma 4, if the procedure LEARN_PI_TAU1 outputs regular patterns, then \( \Gamma_\pi \) is in the case (ii). By Lemma 5 and Lemma 6, the procedure LEARN_PI_TAU2 outputs regular patterns \( \pi'' \) and \( \tau'' \) with \( \pi'' \equiv \pi_\pi \), and \( \tau'' \equiv \pi_\pi \), or \( \pi'' \equiv \pi_\pi \), and \( \tau'' \equiv \pi_\pi \). Thus, \( L(\Gamma_\pi) = L((\pi'', \tau'')) \).

2. In case that \( L(\Gamma_\pi) \not\subseteq L((\pi)) \). By Lemma 3, \( \Gamma_\pi \) is in the case (iii) or (iv). By Lemma 2, we have \( \pi \equiv \pi_\pi \). From Lemma 11, The procedure LEARN_TAU1 outputs a regular pattern \( \tau \) satisfying Condition C-2 for \( \pi \). By Lemma 12 and Lemma 13, if \( L(\Gamma_\pi) \subseteq L((\pi, \tau)) \), then \( \Gamma_\pi \) is in the case (iii) and \( \tau \equiv \tau_\pi \). Thus, \( L(\Gamma_\pi) = L((\pi, \tau)) \). If \( L(\Gamma_\pi) \not\subseteq L((\pi, \tau)) \), then \( \Gamma_\pi \) is in the case (iv) and \( \tau \equiv (\tau_\pi)_\pi \). Then the procedure LEARN_TAU2 outputs a regular pattern \( \tau' \) with \( \tau' \equiv \tau_\pi \). Thus, \( L(\Gamma_\pi) = L((\pi, \tau')) \).

Therefore, the algorithm outputs an EFS \( H \in rEFS \) with \( L(\Gamma_\pi) = L(H) \).

In case that \( \Gamma_\pi \) is in the case (i) or (ii), the algorithm runs in \( O(|\pi, \tau|^3) \) time and uses \( O(|\pi, \tau|^2) \) superset queries. In case that \( \Gamma_\pi \) is in the case (iii) or (iv), it runs in \( O(|\pi, \tau|^2) \) time and uses \( O(|\pi, \tau|^2) \) superset queries and \( O(|\pi, \tau|^2) \) membership queries. Thus, it runs in \( O(|\pi, \tau|^2(|\pi, \tau|)) \) time, and uses \( O(|\pi, \tau|^2) \) membership queries and \( O(|\pi, \tau|^2(|\pi, \tau|)) \) superset queries. Since \( |\pi, \tau| \leq |\Gamma_\pi| \) and \( |\pi, \tau| \leq |\Gamma_\pi| \), it runs in \( O(|\Gamma_\pi|^3) \) time and uses \( O(|\Gamma_\pi|^2) \) membership queries and \( O(|\Gamma_\pi|^2) \) superset queries.

\( \square \)

5. **Hardness Results on Learnability**

In this section, we show the insufficiency of learning \( rEFS \) in the query learning model. By Lemma 14, we have Theorem 4. We omit the proof of Theorem 4.

**Lemma 14:** [2] Suppose the hypothesis space contains a class of distinct sets \( L_1, \ldots, L_N \), and there exists a set \( L_\pi \) which is not a hypothesis, such that for any pair of distinct indices \( i \) and \( j \), \( L_\pi \cap L_\pi = L_\pi \cap L_j \). Then any algorithm that exactly identifies each of the hypotheses \( L_\pi \) using equivalence, membership, and subset queries must make 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 \( \Gamma_\pi \) in \( rEFS \) is exactly identifiable in \( O(|\Gamma_\pi|^4) \) time using \( O(|\Gamma_\pi|^2) \) membership queries and \( O(|\Gamma_\pi|^2) \) superset queries, where the alphabet size is greater than or equal to 5. Moreover, we showed that there exists no polynomial time learning algorithm which identifies any EFS in \( rEFS \) using mem-

| Table 1 | Our results and future works. |
|---------|-------------------------------|
| Polynomial Time | Polynomial Time | Inductive Inference |
| Exact Learning | from Positive Data |
| \( rEFS \) | sufficiency [This work] | Open |
| | superset query & membership query | |
| | insufficiency [This work] | |
| | membership query | equivalence query |
| | subset query | |
| \( P \) | restricted superset query [2] | Open |
| \( RP \) | membership query & one positive example [5] | polynomial time [10] |
bership, equivalence and subset queries. As future works, we will consider the learnability of \( rEFs \) in the framework of polynomial time inductive inference from positive data. We summarize our results and future works in Table 1. We denote by \( \mathcal{P} \) (resp., \( \mathcal{RP} \)) the set of all patterns (resp., regular patterns).

References

[1] D. Angluin, “Finding patterns common to a set of strings,” J. Comput. Syst. Sci., vol.21, pp.46–62, 1980.
[2] D. Angluin, “Queries and concept learning,” Mach. Learn., vol.2, pp.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, vol.24, pp.47–75, 1970.
[4] S. Arikawa, T. Shinohara, and A. Yamamoto, “Learning elementary formal systems,” Theor. Comput. Sci., vol.95, pp.97–113, 1992.
[5] S. Matsumoto and A. Shinohara, “Learning pattern languages using queries,” Proc. EuroCOLT-97, Springer-Verlag, LNAI 1208, pp.185–197, 1997.
[6] S. Matsumoto, A. Shinohara, H. Arimura, and T. Shinohara, “Learning subsequence languages,” in Information Modelling and Knowledge Bases VIII, pp.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, pp.523–534, 2002.
[8] H. Sakamoto, K. Hirata, and H. Arimura, “Learning elementary formal systems with queries,” Theor. Comput. Sci., vol.298, pp.21–50, 2003.
[9] M. Sato, Y. Mukouchi, and D. Zheng, “Characteristic sets for unions of regular pattern languages and compactness,” Proc. ALT-98, Springer-Verlag, LNAI 1501, pp.220–233, Springer, 1998.
[10] T. Shinohara, “Inductive inference of formal systems from positive data,” Bulletin of Informatics and Cybernetics, vol.22, pp.9–18, 1986.
[11] J. Uemura and M. Sato, “Learning of erasing primitive formal systems from positive examples,” Proc. ALT-2003, Springer-Verlag, LNAI 2842, pp.69–83, Springer-Verlag, 2003.

Satoshi Matsumoto is an associate professor of Department of Mathematical Sciences, Tokai University, Kanagawa, Japan. He received the B.S. degree in Mathematics, the M.S. and Dr. Sci. degrees in Information Systems all from Kyushu University, Fukuoka, Japan in 1993, 1995 and 1998, respectively. His research interests include algorithmic learning theory.

Tetsuhiro Miyahara is an associate professor of Graduate School of Information Sciences, Hiroshima City University, Hiroshima, Japan. He received the B.S. degree in Mathematics, the M.S. and Dr. Sci. degrees in Information Systems all from Kyushu University, Fukuoka, Japan in 1984, 1986 and 1996, respectively. His research interests include algorithmic learning theory, knowledge discovery and machine learning.

Hirotaka Kato received the B.S. degree in Mathematical Sciences and the M.S. degrees in Information Mathematics all from Tokai University, Kanagawa, Japan in 2002 and 2004, respectively. His research interests include algorithmic learning theory.