数理解析研究所講究録
992 巻 1997 年 207-214

The Computational Complexity of Hereditary Elementary Formal Systems

池田 大輔 (Daisuke Ikeda) 有村 博紀 (Hiroki Arimura)

daisuke@i.kyushu-u.ac.jp arim@i.kyushu-u.ac.jp

Department of Informatics
Kyushu University 39, Kasuga 816, Japan

Abstract

An elementary formal system (EFS) is a logical system that generates a language. In this paper, we consider a subclass of EFS, called hereditary EFS (HEFS). First, we compare HEFS with the following extensions of pattern languages: multi-pattern languages, languages defined by pattern grammars, and non-synchronized pattern languages. Particularly, we show that a subclass of HEFS called simple EFS (SEFS) precisely generates non-synchronized pattern languages, and the subclass of SEFS with only one predicate symbol precisely generates the languages defined by pattern grammars. Next, we analyze the complexity of the languages definable by HEFS. We show that HEFS exactly defines the complexity class P, the class of languages accepted by deterministic Turing machines in polynomial time. This seems to be the first result to characterize the class P by grammars, while various characterization results by automata, logic, recursive functions, algebraic systems, lambda calculus are shown in literatures. We also show that a subclass of HEFS, called linear HEFS, exactly defines NSPACE($\log n$). Finally, we consider the membership problem for HEFS, that is, the problem of, given a string w and an HEFS Γ, determining whether Γ generates w. We prove that the membership problem is EXPTIME-complete for HEFS, and NP-complete for SEFS.

1 Introduction

In this paper, we consider elementary formal systems (EFS's) which are logical systems introduced by Smullyan for the development of recursive function theory over strings [12]. An EFS is a set of definite clauses, called *axioms*, whose arguments have patterns. EFS's are extensively studied in formal language theory [2], logic programming [14], and computational learning theory [3, 10, 11]. A hereditary EFS (HEFS) is originally introduced by Miyano *et al.* to investigate the polynomial-time learnability of formal languages from examples [9]. An EFS is hereditary if, for each axiom $A \leftarrow B_1, \ldots, B_m$, every pattern in $B_1, \ldots, B_m$ must appear as a subword of some pattern in A.

First, we study the simulation capability of a subclass of HEFS, called simple EFS (SEFS) defined by Arikawa [2]. The only relation among SEFS and other language classes known so far is the inclusion CFL $\subseteq$ SEFS $\subseteq$ CSL [3]. On the other hand, many extensions of CFL are proposed in literatures [8, 13]. These grammars generalize CFL, for example, by iterative substitutions for variables, or by parallel rewriting with productions. Non-synchronized pattern languages (NSPL) [8] are an example of the former type and extended OL-systems (EOL) [13] are an example of the latter type. Then, we show that the class NSPL is exactly the class of languages definable by SEFS. An interesting consequence of this result is that CFL, EOL, and NSPL have almost same space complexity modulo logspace reduction.

Next, we consider the computational complexity of the HEFS languages. Miyano *et al.* [9] showed that any HEFS language is decidable in polynomial time. Therefore, the class HEFS is included in the complexity class P. We show that the converse is also true. Thus, P is the class of languages definable by HEFS. To prove this theorem, we simulate a two-way multihead alternating finite automaton [7] with an HEFS. In a sense, this result gives a framework for describing formal languages for which efficient parsers exist. As a consequence of this result, we have another subclass of HEFS, called linear HEFS, that precisely defines the class NSPACE($\log n$).

Finally, we investigate the computational complexity of the membership problem, which is the problem to, given a grammar G and a string w, decide whether G generates w. We show that the membership problem is EXPTIME-complete for HEFS and NP-complete for SEFS.

2 Preliminaries

For a finite set Δ, $|\Delta|$ denotes the cardinality of Δ. Let Σ be a finite alphabet and X be a set of variables. We assume that Σ and X are mutually disjoint. We denote by Σ^* the set of all words over Σ and by ε the empty word. A *pattern* is an element of $(\Sigma \cup X)^*$. For a pattern π, $var(\pi)$ denote the set of all variables in π. The *length* of a pattern π is denoted by $|\pi|$. Let Π be a finite alphabet of *predicate symbols* associated with a mapping $r : \Pi \to \mathbb{N}$, called *arity*. An *atom* is an expression of the form $p(\tau_1, \ldots, \tau_{r(p)})$, where $p \in \Pi$ and $\tau_1, \ldots, \tau_{r(p)}$ are patterns. For an atom $A = p(\pi_1, \ldots, \pi_n)$, the length of A is defined by $|A| = |\pi_1| + \cdots + |\pi_n|$. An *axiom* C is an expression of the form $A \leftarrow B_1, \ldots, B_m$, where $m \geq 0$ and $A, B_1, \ldots, B_m$ are atoms. The parts A and $B_1, \ldots, B_m$ are called the *head* and the *body* of C, respectively. In case $m = 0$, the axiom C is called a *unit* axiom. We write A for a unit axiom $A \leftarrow$.

An *elementary formal system* (EFS) is a quadruple $S = (\Sigma, \Pi, \Gamma, p_0)$, where Γ is a finite set of axioms and $p_0 \in \Pi$ is the distinguished predicate symbol. For convention, we often identify Γ with $(\Sigma, \Pi, \Gamma, p_0)$ if Σ, Π, and p_0 are understood from context.

A *substitution* θ is a homomorphism $\theta : (\Sigma \cup X)^* \to (\Sigma \cup X)^*$ such that $\theta(a) = a$ for any $a \in \Sigma$. The substitution that maps x_1 to t_1, ..., x_m to t_m is denoted by $\{x_1 := t_1, \ldots, x_m := t_m\}$. A substitution θ is *erasing* if $\theta(x) = \varepsilon$ for some variable x, and *nonerasing* otherwise. In this paper, erasing substitutions are allowed unlike [1]. For a pattern π and a substitution θ, we denote by $\pi\theta$ the image of π with θ. For an atom $A = p(\pi_1, \ldots, \pi_n)$ and an axiom $C = A \leftarrow B_1, \ldots, B_m$, we extend θ by defining $A\theta = p(\pi_1\theta, \ldots, \pi_n\theta)$ and $C\theta = A\theta \leftarrow B_1\theta, \ldots, B_m\theta$.

Definition 1. Let $S = (\Sigma, \Pi, \Gamma, p_0)$ be an EFS. We define a binary relation $\vdash$ inductively as follows:

(i) If C is an axiom in Γ then $\Gamma \vdash C$.

(ii) If $\Gamma \vdash C$ then $\Gamma \vdash C\theta$ for any substitution θ.

(iii) If $\Gamma \vdash A \leftarrow B_1, \ldots, B_m, B_{m+1}$ and $\Gamma \vdash B_{m+1}$ then $\Gamma \vdash A \leftarrow B_1, \ldots, B_m$.

If $\Gamma \vdash C$ then we say C is *provable* from Γ. We define $L(S) = \{w \in \Sigma^* \mid \Gamma \vdash p_0(w)\}$. A language $L \subseteq \Sigma^*$ is *definable* by EFS or an *EFS language* if such S exists.

Now, we introduce some constraints on the patterns of EFS's. An axiom

$$q(\pi_1, \ldots, \pi_n) \leftarrow q_1(\tau_1, \ldots, \tau_{t_1}), q_2(\tau_{t_1+1}, \ldots, \tau_{t_2}), \ldots, q_l(\tau_{t_{l-1}+1}, \ldots, \tau_{t_l})$$

is *hereditary* if, for each $1 \leq j \leq t_l$, a pattern τ_j is a subword of some π_i [9, 10]. By definition, a unit axiom is hereditary. A hereditary axiom $A \leftarrow B_1, \ldots, B_m$ is *length-bounded* if $|A\theta| \geq |B_1\theta| + \cdots + |B_m\theta|$ for any substitution θ [3, 11]. A hereditary axiom is *simple* if it is of the form $p(\pi) \leftarrow q_1(x_1), \ldots, q_m(x_m)$, where π is a pattern and $x_1, \ldots, x_m$ are mutually distinct variables in π [2, 3].

An EFS is *hereditary* (resp. *length-bounded* and *simple*) if all axioms are hereditary (resp. length-bounded and simple). We denote by HEFS (resp. LB-HEFS and SEFS) the class of hereditary EFS's (resp. length-bounded EFS's and simple EFS's) and the class of corresponding languages.

By definition, SEFS and LB-HEFS are subclasses of HEFS. From Arikawa *et al.* [3], we have the hierarchy CFL $\subseteq$ SEFS $\subseteq$ LB-HEFS $\subseteq$ CSL, where CFL and CSL are the classes of context-free languages and context-sensitive languages, respectively.

3 The simulating capacity of SEFS's

In this section, we demonstrate that SEFS's can simulate languages generated by various grammatical devices based on patterns. Thus, SEFS's provide a uniform framework to study these devices.

First, we examine *pattern languages* (PL) [1] and *multi-pattern languages* (MPL) which are extensions of pattern languages to unions [6]. For a pattern π and an alphabet Σ, the pattern language $L_{E,\Sigma}(\pi)$ is the set $\{\pi\theta \in \Sigma^* \mid \theta \text{ is any substitution}\}$. The language is definable by the SEFS $\Gamma = \{p_0(\pi)\}$. For a set of patterns $\{\pi_1, \ldots, \pi_n\}$ and an alphabet Σ, a multi-pattern language $L_{E,\Sigma}(\pi_1, \ldots, \pi_n)$ is the set $\cup_{i=1}^n L_{E,\Sigma}(\pi_i)$. The language is also definable by the SEFS $\Gamma = \{p_0(\pi_i) \mid 1 \leq i \leq n\}$.

Next, we consider grammars that produce strings by iteratively substituting strings for a set of patterns in a non-synchronous way. A *pattern grammar* (GPL) [5] is a pair $G = (P, A)$ of a finite set P of patterns and a finite set $A \subseteq \Sigma^*$ of strings. The language defined by G is $L(G) = \bigcup_{i \geq 0} D_i$, where $D_0 = A$ and $D_{i+1} = \bigcup_{\pi \in P} \{\pi\theta \in \Sigma^* \mid x\theta \in D_0 \cup \cdots \cup D_i \text{ for all } x \in var(\pi)\}$.

A *pattern system* [8] is a quadruple $G = (\Sigma, V, p, t)$, where $n \geq 0$, $V = \{X_1, \ldots, X_n\}$ is a set of variables, and p and t are mappings from V into nonempty finite sets of patterns in $(\Sigma \cup V)^* V (\Sigma \cup V)^*$ and strings in Σ^*, respectively.

Definition 2. (**Mitrana *et al.*** [8]) Let $G = (\Sigma, V, p, t)$ be a pattern system and $n = |V|$. Then, for some fixed j, a *non-synchronized pattern language* (NSPL) is the language defined by $L_{NS}(G, j) = \cup_{i \geq 0} D_j^{(i)}(G)$, where $D_j^{(i)}(G)$ is defined recursively as follows: $D_j^{(0)}(G) = t(X_j)$, and

$$D_j^{(i+1)}(G) = \bigcup_{k \leq i} D_j^{(k)}(G) \cup \{\pi_j\theta \mid \pi_j \in p(X_j), X_l\theta \in D_l^{(i)}(G), 1 \leq l \leq n\}.$$

We denote by PL, MPL, GPL and NSPL the corresponding classes of languages defined by the grammatical devices introduced above. From Mitrana *et al.* [8], we know that two inclusions PL $\subset$ MPL $\subset$ NSPL and GPL $\subset$ NSPL hold.

The only relation among SEFS and Chomsky hierarchy known before is the inclusion CFL $\subseteq$ SEFS $\subseteq$ CSL. Theorem 1 precisely locates SEFS in the hierarchy consisting of grammatical devices based on patterns.

Theorem 1. NSPL is precisely the class of languages definable by SEFS.

Proof: Let $S = (\Sigma, \Pi, \Gamma, p_0)$ be an SEFS. It is easy to transform S into an equivalent SEFS $S' = (\Sigma, \Pi', \Gamma', q_0)$ such that for any axiom in S', all variables in the head appear in the body and all predicate symbols in the body are different from each other.

Now, we built a pattern system $G = (\Sigma, V, p, t)$ as follows. V is the set $\{X_q \mid q \in \Pi'\}$ of variables. For each unit axiom $q(w) \in \Gamma'$, the set $t(X_p)$ contains w. Note that w contains no variables. For each non-unit axiom $q(\pi) \leftarrow q_1(x_1), \ldots, q_m(x_m) \in \Gamma'$, the set $p(X_q)$ contains the pattern $\pi' = \pi\{x_1 := X_{q_1}, \ldots, x_m := X_{q_m}\} \in (\Sigma \cup V)^*$. It immediately follows that $L(S) = L_{NS}(G, q_0)$.

Conversely, it is also easy to construct an SEFS S from a pattern system G using the correspondence between predicate symbols of S and variables of G. □

Mitrana *et al.* raised a question whether NSPL $\subseteq$ EOL holds [8]. Concerning with EOL, Sudborough proved that EOL is LOGCFL-complete [13]. On the other hand, we know that SEFS = NSPL is also LOGCFL-complete from the inclusion CFL $\subseteq$ SEFS $\subseteq$ LOGCFL shown by Miyano *et al.* [9]. From these observations and Theorem 1 above, we know both NSPL and EOL belong to LOGCFL, and they are complete for the class. Thus, we know that NSPL and EOL are somewhat similar in computational complexity.

Corollary 2. NSPL and EOL are equivalent under many-one logspace reduction.

The following theorem is straightforward from the proof of Theorem 1.

Theorem 3. GPL is precisely the class of languages definable by SEFS whose predicate symbol is only p_0.

4 The expressive power of HEFS's

In the previous section, we considered restricted HEFS's, called SEFS's. SEFS's are less powerful than HEFS's since $L = \{a^n b^n c^n \mid n \geq 1\} \in$ HEFS but $L \notin$ SEFS = NSPL [8]. In this section, we examine the expressive power of non-restricted HEFS's and show that HEFS is exactly the class of languages accepted by deterministic Turing machines in polynomial time.

Miyano *et al.* showed the following lemma using the property that the number of subwords of any input is bounded by some polynomial in the length of the input.

Lemma 1. (Miyano *et al.* [10]) Any language definable by HEFS is accepted by some deterministic Turing machine in polynomial time.

To show the converse of Lemma 1, we use that any language in P is accepted by some two-way alternating finite automaton with k heads (2AFA(k)) [7]. A 2AFA(k) M is a finite automaton which has the single input tape, two-way access to the tape, and k read-only heads. An input on the tape is enclosed with the left endmarker ¢ and right endmarker \$. A state of M is either *existential* or *universal*. A *configuration* of M on input $w \in \Sigma^*$ is a $(k+1)$-tuple $(q, h_1, \ldots, h_k)$, where q is a state of M and h_j is the position of the jth head $(0 \leq h_j \leq |w| + 1)$ for each $1 \leq j \leq k$. The configuration is existential (resp. universal) if q is existential (resp. universal).

Let K be the set of states of M. A *transition function* is a mapping from $K \times (\Sigma \cup \{¢, \$\})^k$ into the subsets of $K \times \{L, N, R\}^k$. Let C be a configuration of M and $a_j \in \Sigma \cup \{¢, \$\}$ be the jth symbol of the tape of C. Then a *transition* from C is defined by $\delta(p, a_1, \ldots, a_k) \ni (q, d_1, \ldots, d_k)$, which means that M changes the state into q and moves the head toward the direction $d_j \in \{L, N, R\}$, where L, N and R stand for left, neutral, and right, respectively. A configuration followed from C is called a *successor* of C. Let $D_1, \ldots, D_m$ be all successors of C. When C is existential (resp. universal), C leads to acceptance if and only if D_i leads to acceptance some (resp. all) i.

For a configuration C on input $w \in \Sigma^*$, we define an atom conf(C) as follows. The $(2k+1)$-ary predicate symbol of conf(C) is p subscripted with $e \in \{0,1\}^k$. We denote by $e[j]$ the jth bit of e. In the rest of this section, we write $p_e(\pi_1, \ldots, \pi_{2k}; \pi_{2k+1})$ for the atom $p_e(\pi_1, \ldots, \pi_{2k}, \pi_{2k+1})$. Then conf($C$) is defined by conf($C$) $= p_e(u_1, v_1, \ldots, u_k, v_k; w)$, where for each $1 \le j \le k$, u_j and v_j are subwords of w such that $u_j v_j = w$, if $h_j \ge 1$ then $|u_j| = h_j - 1$ and $e[j] = 1$, and if $h_j = 0$ then $|u_j| = 0$ and $e[j] = 0$. Each triplet $(e[j], u_j, v_j)$ denotes the position of the jth head.

Now we prove the converse of Lemma 1.

Theorem 4. Any language L in P is definable by HEFS.

Proof: Let M be a 2AFA(k) that accepts L. We construct an EFS $S = (\Sigma, \Pi_M, \Gamma_M, p_0)$. The set Π_M is defined as $\Pi_M = \{p_0\} \cup \{p_e \mid p \text{ is a state of } M \text{ and } e \in \{0,1\}^k\}$, where $r(p_0) = 1$ and $r(p_e) = 2k+1$. The set Γ_M defined as follows.

Let $(q, d_1, \ldots, d_k) \in \delta(p, a_1, \ldots, a_k)$ be a transition of M and p be an existential state. We can assume that M moves at most one head, say the sth head. Let E be the set of all pairs $(e, e') \in \{0,1\}^k \times \{0,1\}^k$ such that $e[j] = e'[j]$ for all $j \ne s$ and if $d_s = R$ (resp. $d_s = L$) then $e'[s] = 1$ (resp. $e[s] = 1$). For all $(e, e') \in E$, we add the axioms

$$p_e(\pi_1, \tau_1, \ldots, \pi_k, \tau_k; \pi_s\tau_s) \leftarrow q_{e'}(\pi'_1, \tau'_1, \ldots, \pi'_k, \tau'_k; \pi_s\tau_s), \tag{1a}$$

into Γ_M, where each (π_j, τ_j) and (π'_j, τ'_j) are defined as follows: If $d_s = R$ then

$$(\pi_s, \tau_s, \pi'_s, \tau'_s) = \begin{cases} (x_s, a_s y_s, x_s a_s, y_s) & \text{if } e[s] = 1, \\ (\varepsilon, y_s, \varepsilon, y_s) & \text{if } e[s] = 0. \end{cases} \tag{1b}$$

If $d_s = L$ then

$$(\pi_s, \tau_s, \pi'_s, \tau'_s) = \begin{cases} (x_s b, a_s y_s, x_s, b a_s y_s) & \text{if } e'[s] = 1, a_s \ne \$, \\ (x_s b, \varepsilon, x_s, b) & \text{if } e'[s] = 1, a_s = \$, \\ (\varepsilon, a_s y_s, \varepsilon, a_s y_s) & \text{if } e'[s] = 0. \end{cases} \tag{1c}$$

For all $1 \le j \le k$ such that $d_j = N$,

$$(\pi_j, \tau_j, \pi'_j, \tau'_j) = \begin{cases} (x_j, a_j y_j, x_j, a_j y_j) & \text{if } e[j] = 1, a_j \ne \$, \\ (x_j, \varepsilon, x_j, \varepsilon) & \text{if } e[j] = 1, a_j = \$, \\ (\varepsilon, y_j, \varepsilon, y_j) & \text{if } e[j] = 0. \end{cases} \tag{1d}$$

Let p be a universal state. We can assume that M does not move its heads but change its state p into some states $q^{(1)}, \ldots, q^{(m)}$ universally. This transition is translated into the following axioms for all $e \in \{0,1\}^k$:

$$p_e(\pi_1, \tau_1, \ldots, \pi_k, \tau_k; z) \leftarrow q_e^{(1)}(\pi_1, \tau_1, \ldots, \pi_k, \tau_k; z), \ldots, q_e^{(m)}(\pi_1, \tau_1, \ldots, \pi_k, \tau_k; z), \tag{2}$$

where z is a variable and (π_j, τ_j) is defined as same as those in (1d) for all $1 \le j \le k$.

Finally, for the initial state q of M, we add the axiom with $e = 1^k$

$$p_0(x) \leftarrow q_e(\varepsilon, x, \ldots, \varepsilon, x; x), \tag{3}$$

and for all accepting states p and for all $e \in \{0,1\}^k$, we add axioms

$$p_e(x_1, y_1, \ldots, x_k, y_k; z). \tag{4}$$

Claim: A configuration $C = (p, h_1, \ldots, h_k)$ of M on input $w \in \Sigma^*$ leads to acceptance if and only if $\Gamma_M \vdash \mathrm{conf}(C)$.

Proof of Claim: Assume that C leads to acceptance. If C is an accepting or a universal configuration then it is easy to show $\Gamma_M \vdash \mathrm{conf}(C)$. If C is existential then there exists a successor D of C such that D leads to acceptance. It is sufficient to show $\mathrm{conf}(C) \leftarrow \mathrm{conf}(D)$ is an instance of some axiom in Γ_M since $\Gamma_M \vdash \mathrm{conf}(D)$ by the induction hypothesis. There exists an axiom $A \leftarrow B \in \Gamma_M$ such that $\mathrm{conf}(C) = A\theta$ for some θ. Therefore we show $B\theta = \mathrm{conf}(D)$. From the construction of (1), the pairs (π_j, τ_j) and (π'_j, τ'_j) simulate the move of the jth head, and (π_j, τ_j) and (π'_j, τ'_j) simulate one of the sth head. Thus, $B\theta = \mathrm{conf}(D)$. The converse direction is proved in a similar way by using induction on the construction of the proof for $\Gamma_M \vdash \mathrm{conf}(C)$.

From the above claim, the initial configuration C_0 leads to acceptance if and only if $\Gamma_M \vdash \mathrm{conf}(C_0)$. Hence, $w \in L$ if and only if $w \in L(S)$. The above axioms are obviously hereditary. This completes the proof. □

From Lemma 1 and Theorem 4, we have the main theorem.

Theorem 5. P is exactly the class of languages definable by HEFS.

A *linear HEFS* is an HEFS such that each axiom has at most one atom in the body. By a similar proof for Theorem 4, we have the following corollary.

Corollary 6. NSPACE($\log n$) is exactly the class of languages definable by linear HEFS.

5 Complexity of the membership problem

In this section, we investigate the complexity of the membership problems for HEFS and SEFS. The *membership problem* for a class C of grammars is the problem of, given a string $w \in \Sigma^*$ and a grammar $G \in C$, deciding whether $w \in L(G)$. Let EXPTIME be the class of languages accepted by deterministic Turing machines in time $O(2^{n^c})$ for some $c > 0$.

Theorem 7. The membership problem for HEFS is EXPTIME-complete.

Proof: For one direction, it is easy to give an *alternating Turing machine* (ATM) M that solves the membership problem in space $O(n^c)$ for some $c > 0$. This can be done in space $O(r \log n)$ by using pointers on input to represent an atom and by using alternations to simulate a top down proof for $\Gamma \vdash p_0(w)$, where r is the maximum arity of predicates.

For the converse direction, let $L \subseteq \Sigma^*$ be a language in EXPTIME. An idea is to encode a configuration $(a_1 \cdots a_{i-1} p\, a_i \cdots a_n)$ of an ATM M, where M is scanning the ith cell in state p and $a_i \in \{0, 1, B\}$ for all $1 \le i \le n$, by an atom $p(a_1, \ldots, a_{i-1}, \dagger, a_i, \ldots, a_n, 0, 1, B)$ of arity $n^c + 4$.

It is easy to transform axioms from transitions in similar way to Theorem 4. This transformation can be done in space $O(\log n)$. Combining these results, we prove the theorem. □

As a corollary, we can easily show that the membership problem for linear HEFS is PSPACE-complete. For every $k \ge 1$, let LB-HEFS(k) be the subclass of LB-HEFS for which the arity of predicates are bounded by k.

Theorem 8. For every $k \geq 1$, the membership problems for LB-HEFS(k) and SEFS are both NP-complete.

Proof: Since the membership problem for PL is NP-complete [1], it is reducible to that for LB-HEFS(k) for every $k \geq 1$. This is also the case for SEFS.

Since an LB-HEFS Γ is hereditary and the arities of predicates are bounded by constant k, there are at most polynomially many distinct atoms whose arguments are subwords of an input. Therefore, a nondeterministic Turing machine M can decide whether $\Gamma \vdash p(w)$ in polynomial time. Since SEFS $\subseteq$ LB-HEFS(1), the result immediately follows. □

The membership problem for CFL (CFG as representation) is known to be P-complete. Hence, it is interesting that the complexity of the membership problems for CFL and SEFS are quite different in Theorem 8 above, while the languages of CFL and SEFS have almost same complexity in Corollary 2 of Section 3.

6 Conclusion

In this paper, we studied the HEFS languages. We show that HEFS captures the complexity class P and linear HEFS captures NSPACE(log n). We also show that SEFS and the subclass of SEFS with only one predicate symbol precisely generate NSPL and GPL, respectively. Finally, we investigate the complexity of the membership problems for HEFS and for SEFS.

For a pattern system, Mitrana *et al.* defined two languages, one is NSPL and the other is a *strongly synchronized pattern languages* (SSPL) [8]. Both languages are generated by iterative substitutions. At some step of iterations, all substituted words are generated in the previous step for SSPL, while they are generated in before steps for NSPL. The classes SSPL and NSPL are incomparable [8]. Thus, it is an interesting task to find a subclass of HEFS that corresponds to SSPL. Since SSPL is closely related with some OL system, it is also interesting to compare HEFS with them.

References

[1] D. Angluin. Finding Patterns Common to a Set of Strings. *Journal of Computer and System Sciences*, 21(1):46–62, 1980.

[2] S. Arikawa. Elementary Formal Systems and Formal Languages — Simple Formal Systems. *Memoirs of Faculty of Science, Kyushu University, Ser. A. , Math.*, 24:47–75, 1970.

[3] S. Arikawa, T. Shinohara, and A. Yamamoto. Learning Elementary Formal Systems. *Theoretical Comput. Sci.*, 95(1):97–113, 1992.

[4] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133, 1981.

[5] J. Dassow, G. Păun, and A. Salomaa. Grammars Based on Patterns. *Internatinal Journal of Foundations of Computer Science*, 4(1):1–14, 1993.

[6] L. Kari, A. Mateescu, G. Păun, and A. Salomaa. Multi-Pattern Languages. *Theoretical Comput. Sci.*, 141(1–2):253–268, 1995.

[7] K. N. King. Alternating Multihead Finite Automata. *Theoretical Comput. Sci.*, 61(2–3):149–174, 1988.

[8] V. Mitrana, G. Păun, G. Rozenberg, and A. Salomaa. Pattern Systems. *Theoretical Comput. Sci.*, 154(2):183–201, 1996.

[9] S. Miyano, A. Shinohara, and T. Shinohara. Which Classes of Elementary Formal Systems are Polynomial-Time Learnable? In *Proc. 2nd Workshop on Algorithmic Learning Theory*, pages 139–150, 1991.

[10] S. Miyano, A. Shinohara, and T. Shinohara. Polynomial-Time Learning of Elementary Formal Systems and an Application to Discovering Motifs in Proteins. Technical Report RIFIS-TR-CS-37, Research Institute of Fundamental Information Science, Kyushu University, 1995.

[11] T. Shinohara. Rich Classes Inferable from Positive Data: Length-Bounded Elementary Formal Systems. *Inf. Comput*, 108(2):175–86, 1994.

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

[13] I. H. Sudborough. The Complexity of the Membership Problem for Some Extensions of Context-Free Languages. *Int. J. Comput. Math.*, 6(3):191–215, 1977.

[14] A. Yamamoto. Procedural Semantics and Negative Information of Elementary Formal System. *J. Logic Programming*, 13(1):89–97, 1992.