ICOT Technical Report: TR-472

TR-472

On Learning Elementary Formal Systems: Towards an Efficient Learning for Context-Sensitive Languages

by
Y. Sakakibara (Fujitsu)

April, 1989

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

On Learning Elementary Formal Systems: Towards an Efficient Learning for Context-Sensitive Languages

Yasubumi SAKAKIBARA

International Institute for Advanced Study of Social Information Science (IIAS-SIS)
FUJITSU LIMITED
140, Miyamoto, Numazu, Shizuoka 410-03, Japan
E-mail : yasu%iias.fujitsu.junet@uunet.uu.net

Abstract

In this paper, we introduce a new class of expressions for learning formal languages defined by Smullyan's elementary formal systems. The class of expressions is a natural extension of context-free grammars and the languages defined by them lie between context-free languages and context-sensitive languages. We demonstrate an efficient algorithm to learn them in the framework of learning by using queries to a teacher modeled on Angluin's approach to learning k-bounded context-free grammars. This algorithm may be viewed as a natural and powerful extension of the Angluin's algorithm.

1 Introduction

We consider the problem of learning formal languages by using queries to a teacher. In [6] Angluin devises an elegant formulation of a teacher and learner paradigm and models a learning situation in which a teacher is available to answer some queries about the material to be learned. Several materials have been investigated to find an algorithm which can quickly and reliably learn them from "reasonable" kinds of queries [2,3,4,5,8,13,14].

In this paper we are interested in formal languages for the target domains to be learned. Formal languages are typically represented as regular expressions, finite-state automata, context-free grammars or transformation rules. Angluin [4] shows that the regular sets can be learned by an algorithm using equivalence and membership queries, a combination termed a *minimally adequate teacher*, in time polynomial. It is still an open question whether there is a polynomial time algorithm for learning the full class of context-free languages using membership and equivalence queries. Recently Angluin [2] gives a polynomial time algorithm to learn k-bounded context-free grammars using equivalence and non-terminal membership queries and Sakakibara [14] gives a polynomial time algorithm to learn deterministic skeletal automata using membership and equivalence queries and applies it to the problem of learning context-free grammars. Now there seems to be two directions to investigate. One direction is to study the problem of learning context-free languages using membership and equivalence queries. The other direction is to study the problem of learning a larger class of formal languages in which more than membership and equivalence queries are available to the learning algorithm. In this paper we take the latter direction.

We introduce a new class of expressions for the problem of learning formal languages defined by Smullyan's elementary formal systems [19]. The class of expressions is a natural extension of context-free grammars and the languages defined by them lie between context-free languages and context-sensitive languages. We demonstrate a polynomial time algorithm to learn them using queries analogous to the ones introduced in Angluin's approach [2] to learning k-bounded context-free grammars. This implies that there exists a larger class of formal languages than context-free languages which is efficiently learnable by using some "acceptable" queries. This algorithm may also be viewed as a natural and powerful extension of the Angluin's algorithm.

2 Preliminaries

2.1 Phrase-Structure Grammars and Languages

An *alphabet* is a finite non-empty set of symbols. The set of all finite strings of symbols from an alphabet A is denoted A^*. The empty string is denoted ϵ. The set of all finite non-null strings of symbols from A is denoted A^+. The length of the string w is denoted $|w|$. If A is a finite set, $|A|$ denotes the cardinality of A.

A *phrase-structure grammar* is denoted $G = (N, \Sigma, \Pi, S)$, where N and Σ are alphabets of *nonterminal symbols* and *terminal symbols* respectively such that $N \cap \Sigma = \phi$. Π is a finite set of productions; each production is of the form $\alpha \rightarrow \beta$, where $\alpha \in (N \cup \Sigma)^* N (N \cup \Sigma)^*$ and $\beta \in (N \cup \Sigma)^*$. Finally, S is a special nonterminal called the *start symbol*. If $\alpha \rightarrow \beta$ is a production of P, then for any strings γ and δ in $(N \cup \Sigma)^*$, we define $\gamma\alpha\delta \Rightarrow \gamma\beta\delta$. $\overset{*}{\Rightarrow}$ is the reflexive and transitive closure of $\Rightarrow$. The *language generated* by G, denoted $L(G)$, is $\{w \mid w \text{ is in } \Sigma^* \text{ and } S \overset{*}{\Rightarrow} w\}$. A phrase-structure grammar $G = (N, \Sigma, \Pi, S)$ is *context-sensitive* if each production is of the form $\alpha A \gamma \rightarrow \alpha\beta\gamma$, where $A \in N$, $\alpha, \gamma \in (N \cup \Sigma)^*$, and $\beta \in (N \cup \Sigma)^+$. A phrase-structure grammar $G = (N, \Sigma, \Pi, S)$ is *context-free* if each production is of the form $A \rightarrow \alpha$, where $A \in N$ and $\alpha \in (N \cup \Sigma)^*$.

2.2 Elementary Formal Systems and Languages

We give the notion of Smullyan's elementary formal systems [19] and define their languages. Let Σ be the alphabet of terminal symbols and elements in it be denoted by $a, b, c, \ldots$. Let V and D be alphabets, where Σ, V and D are mutually disjoint. Elements in V are called *variables* and denoted by $x, x_1, x_2, \ldots$, and elements in D are called *predicates* and denoted by $P, Q, P_1, P_2, \ldots, Q_1, Q_2, \ldots$, each of which is assigned a unique positive integer called its *degree*.

An *elementary formal system* (EFS, for short) over an alphabet Σ is a quadruple $E = (D, V, \Sigma, M)$, where M is a finite set of expressions called (*well-formed*) *formulas* defined below, which are called *axioms* of E.

1. A *term* t of E is a string in $(\Sigma \cup V)^*$, and by $t(x_1, x_2, \ldots, x_r)$ we denote a term in which the occurring variables are precisely $x_1, x_2, \ldots, x_r$ (not necessarily distinct).
2. An *atomic formula* of E is an expression of the form $P(t_1, t_2, \ldots, t_m)$, where P is a predicate in D with degree m and $t_1, t_2, \ldots, t_m$ are terms of E. If $t_1, t_2, \ldots, t_m$ are terminal strings in Σ^*, then $P(t_1, t_2, \ldots, t_m)$ is called *ground*.
3. A (*well-formed*) *formula* of E is an expression of the form

$$R \leftarrow R_1 \& R_2 \& \cdots \& R_n \qquad (n \geq 0)$$

where $R, R_1, R_2, \ldots, R_n$ are atomic formulas of E, and $R_1, R_2, \ldots, R_n$ are called the *premises* of the formula and R is called the *conclusion* of the formula.

In the following definitions, we assume that all predicates are monadic, that is, the degrees of predicates are all one, because predicates with degree one are enough for us.

Let θ be any homomorphism (with respect to concatenation) from terms to terms. We denote the image of a term t by $t\theta$. If θ maps any terminal symbols a in Σ to itself, then θ is called a *substitution*. For a formula $F = P(t) \leftarrow P_1(t_1)\& \cdots \& P_n(t_n)$, we define $F\theta = P(t\theta) \leftarrow P_1(t_1\theta)\& \cdots \& P_n(t_n\theta)$.

We say that a formula $F = P(t) \leftarrow P_1(t_1)\& \cdots \& P_{n-1}(t_{n-1})$ is *provable from* E if F satisfies one of the following conditions:

1. F is in M,
2. $F = F'\theta$ for some formula F' provable from E and some substitution θ, and
3. two formulas of the forms $F = P(t) \leftarrow P_1(t_1)\& \cdots \& P_n(t_n)$ and $P_n(t_n) \leftarrow$ are provable from E.

We say that a formula F is *provable from* E *with* n *steps* if F is provable from E by applying the above three rules n times. We say that an atomic formula $P(t)$ is *provable from* E if the formula of the form $P(t) \leftarrow$ is provable from E. The *language* defined by an EFS E and a predicate P, denoted $L(E, P)$, is the set $\{w \mid w \text{ is in } \Sigma^* \text{ and } P(w) \text{ is provable from } E\}$. For a predicate P, two EFSs E and E' are said to be P-*equivalent* iff $L(E, P) = L(E', P)$.

Proposition 1 ([7]) *The languages generated by phrase-structure grammars (type 0 grammars) are defined by elementary formal systems.*

3 Extended Simple Formal Systems

In this section, we introduce three restricted forms of EFSs, state their relations to other classes of formal languages and show some closure and nonclosure properties of them.

3.1 Restrictions of EFS

We first define a class of restricted EFSs which precisely define context-free languages. An EFS $E = (D, V, \Sigma, M)$ is called a *context-free form* if every predicate symbol in D is monadic and each axiom of E is of the form

$$P(t) \leftarrow P_1(x_1) \& \cdots \& P_n(x_n)$$

where $P, P_1, \ldots, P_n$ are predicates in D, (i) $x_1, \ldots, x_n$ are distinct variables, and (ii) t is a term $t(x_1, \ldots, x_n)$ in which each of the variables $x_1, \ldots, x_n$ occurs precisely once.

As we will show later, the languages defined by context-free forms of EFSs are precisely the context-free languages.

In the definition of context-free forms of EFSs, the axioms are restricted to the form $P(t) \leftarrow P_1(x_1) \& \cdots \& P_n(x_n)$, where t is a term containing the variables $x_1, \ldots, x_n$, and there are two conditions (i) and (ii) of syntactic restrictions on the form. By releasing the condition (ii), we get the following class of restricted EFSs.

An EFS $E = (D, V, \Sigma, M)$ is called a *simple formal system* (SFS, for short) if every predicate symbol in D is monadic and each axiom of E is of the form

$$P(t(x_1, \ldots, x_n)) \leftarrow P_1(x_1) \& \cdots \& P_n(x_n)$$

where $P, P_1, \ldots, P_n$ are predicates in D and $x_1, \ldots, x_n$ are distinct variables.

By releasing the both conditions (i) and (ii), we get the following class of restricted EFSs.

An EFS $E = (D, V, \Sigma, M)$ is called an *extended simple formal system* (ESFS, for short) if every predicate symbol in D is monadic and each axiom of E is of the form

$$P(t(x_1, \ldots, x_n)) \leftarrow P_1(x_1) \& \cdots \& P_n(x_n)$$

where $P, P_1, \ldots, P_n$ are predicates in D.

In other words, the ESFS is an EFS in which all predicates are monadic and the terms of the premises are variables. Note that from the definitions of context-free forms, SFSs and ESFSs, the conclusion of an axiom which has no premise becomes ground.

The class of simple formal systems is firstly introduced by Arikawa [7]. The class of languages defined by SFSs properly contains the class of context-free languages and is properly contained in the class of context-sensitive languages.

Example 1 Suppose $\Sigma = \{a, b, c\}$, $D = \{P, P_1, P_2, P_3, Q_1, Q_2, Q_3\}$ and $V = \{x, x_1, x_2, \ldots\}$.

1. Let $E_1 = (D, V, \Sigma, M_1)$ be an ESFS such that M_1 is the set of the following axioms

$$\begin{aligned}
&P(x) \leftarrow P_1(x) \& Q_1(x),\\
&P_1(x_1x_2) \leftarrow P_2(x_1) \& P_3(x_2),\\
&P_2(axb) \leftarrow P_2(x),\\
&P_2(ab) \leftarrow,\\
&P_3(cx) \leftarrow P_3(x),\\
&P_3(c) \leftarrow,\\
&Q_1(x_1x_2) \leftarrow Q_2(x_1) \& Q_3(x_2),\\
&Q_2(ax) \leftarrow Q_2(x),\\
&Q_2(a) \leftarrow,\\
&Q_3(bxc) \leftarrow Q_3(x),\\
&Q_3(bc) \leftarrow .
\end{aligned}$$

   Then $L(E_1, P) = \{a^nb^nc^n \mid n \geq 1\}$, which cannot be generated by any context-free grammar.

2. Let M_2 be the set of axioms

$$\begin{aligned}
&P(x_1x_2x_1) \leftarrow P_1(x_1) \& P_1(x_2),\\
&P_1(ax) \leftarrow P_1(x),\\
&P_1(bx) \leftarrow P_1(x),\\
&P_1(cx) \leftarrow P_1(x),\\
&P_1(a) \leftarrow,\\
&P_1(b) \leftarrow,\\
&P_1(c) \leftarrow .
\end{aligned}$$

Let $E_2 = (D, V, \Sigma, M_2)$. Then $L(E_2, P) = \{xyx \mid x, y \in \Sigma^*\}$, which is a *pattern language* (the language generated by the pattern xyx) in the sense of Angluin [1].

3. Let M_3 be the set of axioms

$$P(xx) \leftarrow P(x),$$
$$P(a) \leftarrow .$$

Let $E_3 = (D, V, \Sigma, M_3)$. Then $L(E_3, P) = \{a^{2^n} \mid n \geq 0\}$.

The *size* of the ESFS $E = (D, V, \Sigma, M)$, denoted $size(E)$, is the sum of $|D|$, $|\Sigma|$, $|M|$, and the sum of the lengths of the terms in the conclusions of all the axioms in M.

3.2 Relations with Other Formal Languages

We first show that the class of languages defined by ESFSs contains some important classes of formal languages.

Proposition 2 *If L is a context-free languages, then there exists a context-free form of EFS E and a predicate P in E such that $L(E, P) = L$. Conversely, if E is a context-free form of EFS and P is a predicate in E, then $L(E, P)$ is a context-free language.*

Proof. Let $G = (N, \Sigma, \Pi, S)$ be any context-free grammar. We define the corresponding context-free form $E(G) = (D, V, \Sigma, M)$ of EFS as follows.

$$\begin{aligned} D &= N, \\ M &= \{A(y_0x_0y_1 \cdots x_ny_n) \leftarrow B_1(x_1) \& \cdots \& B_n(x_n) \mid \\ &\qquad\qquad A \rightarrow y_0B_0y_1 \cdots B_ny_n \in \Pi \text{ with } B_1, \ldots, B_n \in N \text{ and } y_0, \ldots, y_n \in \Sigma^*\} \end{aligned}$$

We omit the simple inductive proof that $L(E(G), S) = L(G)$.

Conversely let $E = (D, V, \Sigma, M)$ be any context-free form of EFS. We define the corresponding context-free grammar $G(E) = (N, \Sigma, \Pi, S)$ as follows.

$$\begin{aligned} N &= D, \\ S &= P, \\ \Pi &= \{Q \rightarrow t(Q_1, \ldots, Q_n) \mid Q(t(x_1, \ldots, x_n)) \leftarrow Q_1(x_1) \& \cdots \& Q_n(x_n) \in M\} \end{aligned}$$

We omit the simple inductive proof that $L(G(E)) = L(E, P)$. Q.E.D.

Thus the class of languages defined by ESFSs contains the class of context-free languages.

A *pattern* introduced by Angluin in [1] is a non-null finite string of terminal and variable symbols. The *language* of a pattern is all strings obtained by substituting non-null terminal strings for the variables of the pattern.

Proposition 3 *Any pattern language is defined by an ESFS.*

Proof. Let L be the language of any pattern over Σ. Any pattern is a term in the terminology of this paper. Let $t(x_1, \ldots, x_n)$ be a pattern (term) which generates L. We construct the corresponding ESFS $E = (D, V, \Sigma, M)$ as follows.

$$\begin{aligned} D &= \{P, Q\}, \\ M &= \{P(t(x_1, \ldots, x_n)) \leftarrow Q(x_1) \& \cdots \& Q(x_n)\} \\ &\quad \cup \{Q(ax) \leftarrow Q(x) \mid a \in \Sigma\} \\ &\quad \cup \{Q(a) \leftarrow \mid a \in \Sigma\} \end{aligned}$$

Then $L(E, P) = L$. Q.E.D.

Next we show that languages defined by ESFSs are recursive sets.

Proposition 4 *There is an algorithm to determine, for an ESFS E, a predicate P and a string $w \in \Sigma^*$, whether $w \in L(E, P)$.*

Proof. We prove it by exhibiting an algorithm which takes a terminal string w, the ESFS $E = (D, V, \Sigma, M)$ and a predicate P and determines whether $P(w)$ is provable from E.

Let

$$L = \{y \mid y \text{ is a substring of } w\}.$$

There are at most $(|w| + 1)^2$ elements of L, and L is easily computed in time polynomial in $|w|$. Let

$$I = \{Q(y) \mid y \in L,\ Q \in D, \text{ and } Q(y) \text{ is provable from } E\}.$$

There are at most $|D|(|w|+1)^2$ elements of I. Clearly, $P(w)$ is provable from E iff $P(w)$ is in I. We compute I as follows.

Initially, for each axiom of the form $Q(y) \leftarrow$ in M such that y is a substring of w, the algorithm puts $Q(y)$ in I. Then the following process is iterated until the first iteration in which no new elements are added to I.

For each axiom of the form

$$Q(t(x_1,\ldots,x_n)) \leftarrow Q_1(x_1)\&\cdots\&Q_n(x_n)$$

and for every n-tuple

$$\langle Q_1(y_1),\ldots,Q_n(y_n)\rangle$$

of elements of I such that $y_i = y_j$ if $x_i = x_j$ for $1 \leq i,j \leq n$, if $t(y_1,\ldots,y_n)$ is a substring of w, then put $Q(t(y_1,\ldots,y_n))$ in I if it is not already there.

Since I is finite and the set M of axioms is also finite, this algorithm terminates, so we prove that this algorithm computes I. Let I' be the set of ground atomic formulas computed by the algorithm on input w. It is clear that $I' \subseteq I$. Suppose $Q(y) \in I$. Since y is of finite length and M is finite, $Q(y)$ is provable from E with a finite number of steps. We prove $Q(y) \in I'$ by induction on the step n. When $Q(y)$ is provable from E with one step, the formula $Q(y) \leftarrow$ is in M, so the algorithm puts $Q(y)$ in I' in the initial stage.

Next suppose that $Q(y) \in I'$ for $Q(y)$ which is provable from E with n steps. Let $Q(y)$ be provable from E with $n+1$ steps. Then either $Q(y) = F\theta$ for some atomic formula F which is provable from E with n steps and some substitution θ or there are an axiom $Q(t(x_1,\ldots,x_n)) \leftarrow Q_1(x_1)\&\cdots\&Q_n(x_n)$ in M and atomic formulas $Q_1(y_1),\ldots,Q_n(y_n)$ which are provable from E with n steps such that $t(y_1,\ldots,y_n) = y$. Since all atomic formulas provable from an ESFS are ground, $Q(y) = F\theta = F$. Thus $Q(y) \in I'$ by the induction hypothesis. Since $y_1,\ldots,y_n$ are substrings of w, $Q_1(y_1),\ldots,Q_n(y_n) \in I'$ by the induction hypothesis, and since $t(y_1,\ldots,y_n) = y$ is a substring of w, the algorithm puts $Q(y)$ in I'. This completes the induction and the proof of Proposition 4.   Q.E.D.

Further we show that languages defined by ESFSs are context-sensitive.

Proposition 5 *If $L = L(E, P)$ for an ESFS $E = (D, V, \Sigma, M)$ and a predicate P, then L is a context-sensitive language.*

Proof. Without loss of generality, we assume that each axiom of E is either of the form $Q(t(x_1, \ldots, x_n)) \leftarrow Q_1(x_1)\& \cdots \& Q_n(x_n)$ where the variables $x_1, \ldots, x_n$ are distinct or of the form $Q(x) \leftarrow Q_1(x)\&Q_2(x)$. Let k be the maximum number of premises of any axiom in M. We define the corresponding phrase-structure grammar $GS(E) = (N, \Sigma, \Pi, S)$ as follows.

$$
\begin{aligned}
N &= D \\
&\quad \cup \{A_{a_n} \mid a \in \Sigma,\ 1 \le n \le k\} \\
&\quad \cup \{A_a \mid a \in \Sigma\} \\
&\quad \cup \{X_0, X_1, X_2, \ldots, X_k, X_{\#}, X_{\#\#}, Y_1, Y_2, \ldots, Y_k, Z_1, Z_2, Z_3\}, \\
S &= P, \\
\Pi &= \{Q \to X_0 Q_1 X_1 \cdots Q_n X_n t(Y_1, \ldots, Y_n) X_{\#} \mid \\
&\qquad Q(t(x_1, \ldots, x_n)) \leftarrow Q_1(x_1)\& \cdots \& Q_n(x_n) \in M\} && (1) \\
&\quad \cup \{a X_n \to X_n A_{a_n} \mid a \in \Sigma,\ 1 \le n \le k\} && (2) \\
&\quad \cup \{A_{a_n} B \to B A_{a_n} \mid a \in \Sigma,\ 1 \le n \le k, \\
&\qquad B \in \Sigma \cup \{X_{n+1}, \ldots, X_k\} \cup \{Y_1, \ldots, Y_{n-1}, Y_{n+1}, \ldots, Y_k\}\} && (3) \\
&\quad \cup \{A_{a_n} Y_n \to Y_n a A_{a_n} \mid a \in \Sigma,\ 1 \le n \le k\} && (4) \\
&\quad \cup \{A_{a_n} X_{\#} \to X_{\#} \mid a \in \Sigma,\ 1 \le n \le k\} && (5) \\
&\quad \cup \{X_0 X_1 \cdots X_n \to X_{\#\#} \mid 1 \le n \le k\} && (6) \\
&\quad \cup \{X_0 \to X_{\#\#}\} && (7) \\
&\quad \cup \{X_{\#\#} a \to a X_{\#\#} \mid a \in \Sigma\} && (8) \\
&\quad \cup \{X_{\#\#} Y_n \to X_{\#\#} \mid 1 \le n \le k\} && (9) \\
&\quad \cup \{X_{\#\#} X_{\#} \to \epsilon\} && (10) \\
&\quad \cup \{Q \to Z_1 Q_1 Z_2 Q_2 Z_3 \mid Q(x) \leftarrow Q_1(x)\&Q_2(x) \in M\} && (11) \\
&\quad \cup \{Z_2 a \to A_a Z_2 \mid a \in \Sigma\} && (12) \\
&\quad \cup \{b A_a \to A_a b \mid a, b \in \Sigma\} && (13)
\end{aligned}
$$

$$\cup\{Z_1 A_a a \to a Z_1 \mid a \in \Sigma\} \tag{14}$$

$$\cup\{Z_1 Z_2 Z_3 \to \epsilon\}, \tag{15}$$

where for $a \in \Sigma$ and $1 \leq n \leq k$, the elements A_{a_n}, A_a, X_n and Y_n are new nonterminals which are mutually distinct, X_0, $X_{\#}$, $X_{\#\#}$, Z_1, Z_2 and Z_3 are additional nonterminals.

We can prove that $L(GS(E)) = L(E, P)$ inductively.

Using (1), all strings of the form

$$X_0 w_1 X_1 \cdots w_n X_n t(Y_1, \ldots, Y_n) X_{\#} \tag{16}$$

can be derived from Q, where $w_i \in \Sigma^*$ such that $Q_i(w_i)$ is provable from E $(1 \leq i \leq n)$. By (2), (3), (4) and (5), the string

$$X_0 X_1 \cdots X_n t(Y_1 w_1, \ldots, Y_n w_n) X_{\#} \tag{17}$$

can be derived from (16). By (6), (7), (8), (9) and (10), the terminal string $t(w_1, \ldots, w_n)$ can be derived from (17).

Using (11), all strings of the form

$$Z_1 w_1 Z_2 w_2 Z_3 \tag{18}$$

can be derived from Q, where $w_1, w_2 \in \Sigma^*$ such that $Q_1(w_1)$ and $Q_2(w_2)$ are provable from E. By (12), (13) and (14), the string

$$w_1 Z_1 Z_2 Z_3 \tag{19}$$

can be derived from (18) iff $w_1 = w_2$. By (15), the terminal string w_1 can be derived from (19).

Because these are only cases where a derivation leads from Q to a terminal string, we conclude that $L(GS(E)) = L(E, P)$.

Further by the workspace theorem in [15], $L(GS(E))$ is context-sensitive. Q.E.D.

It is an open problem whether or not there are ϵ-free context-sensitive languages which cannot be defined by any ESFS.

Lastly we show the relation among the classes defined by three restricted forms of EFSs.

Proposition 6 *The class C_{CF} of languages defined by context-free forms of EFSs is properly contained in the class C_{SFS} of languages defined by SFSs, and C_{SFS} is properly contained in the class C_{ESFS} of languages defined by ESFSs.*

Proof. From the definitions of context-free forms, SFSs and ESFSs, it is clear that $C_{CF} \subseteq C_{SFS} \subseteq C_{ESFS}$. From the results in [7], $C_{CF} \subsetneqq C_{SFS}$ and the language $\{a^n b^n c^n \mid n \geq 1\}$ cannot be defined by any SFS. Thus $C_{SFS} \subsetneqq C_{ESFS}$. Q.E.D.

3.3 Closure Properties

We show some closure and nonclosure properties for the languages defined by ESFSs.

Proposition 7 *The class of languages defined by ESFSs is closed under the operations of union, concatenation, Kleene closure, intersection and reversal.*

Proof. Let $E_1 = (D_1, V, \Sigma, M_1)$ and $E_2 = (D_2, V, \Sigma, M_2)$ be ESFSs. We may assume that D_1 and D_2 are disjoint. Let $L_1 = L(E_1, P_1)$ and $L_2 = L(E_2, P_2)$. Assume also that predicates P_3, P_4, P_5 and P_6 are not in D_1 or D_2.

For $L_1 \cup L_2$, we construct the ESFS $E_3 = (D_1 \cup D_2 \cup \{P_3\}, V, \Sigma, M_3)$, where M_3 is $M_1 \cup M_2$ plus the axioms

$$P_3(x) \leftarrow P_1(x),$$

$$P_3(x) \leftarrow P_2(x).$$

Then $L(E_3, P_3) = L_1 \cup L_2$.

For concatenation, we construct the ESFS $E_4 = (D_1 \cup D_2 \cup \{P_4\}, V, \Sigma, M_4)$, where M_4 is $M_1 \cup M_2$ plus the axiom

$$P_4(x_1 x_2) \leftarrow P_1(x_1) \& P_2(x_2).$$

Then $L(E_4, P_4) = \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$.

For Kleene closure, we construct the ESFS $E_5 = (D_1 \cup \{P_5\}, V, \Sigma, M_5)$, where M_5 is M_1 plus the axioms

$$P_5(x_1 x_2) \leftarrow P_1(x_1) \& P_5(x_2),$$

$$P_5(\epsilon).$$

Then $L(E_5, P_5) = L_1^*$.

For intersection, we construct the ESFS $E_6 = (D_1 \cup D_2 \cup \{P_6\}, V, \Sigma, M_6)$, where M_6 is $M_1 \cup M_2$ plus the axiom

$$P_6(x) \leftarrow P_1(x)\&P_2(x).$$

Then $L(E_6, P_6) = L_1 \cap L_2$.

For a term t, let t^R be t written backward. For reversal, we construct the ESFS $E_7 = (D_1, V, \Sigma, M_7)$, where

$$\begin{aligned} & Q(t(x_1, \ldots, x_n)^R) \leftarrow Q_1(x_1)\& \cdots \&Q_n(x_n) \in M_7 \\ \text{iff} \quad & Q(t(x_1, \ldots, x_n)) \leftarrow Q_1(x_1)\& \cdots \&Q_n(x_n) \in M_1. \end{aligned}$$

Then $L(E_7, P_1) = \{w^R \mid w \in L_1\}$. Q.E.D.

Thus the class of languages defined by ESFSs (properly) contains an infinite hierarchy of intersections of context-free languages [10].

In order to obtain a nonclosure property, we quote an important result of a representation theorem in [9].

Proposition 8 ([9]) *For each recursively enumerable set $L \subseteq \Sigma^*$, there exist deterministic context-free languages L_1, L_2 and a homomorphism φ such that $L = \varphi(L_1 \cap L_2)$.*

Using this result, we can show the following.

Proposition 9 *The class of languages defined by ESFSs is not closed under homomorphism.*

Proof. By Propositions 2 and 7, there is an ESFS E such that $L(E, P) = L_1 \cap L_2$ for any two deterministic context-free languages L_1 and L_2. If the class of languages defined by ESFSs were closed under homomorphism, any recursively enumerable set can be a language defined by an ESFS, by Proposition 8. This contradicts Proposition 4. Q.E.D.

3.4 Proof-DAGs

We introduce a similar notion of a parse-DAG in [2], called *proof-DAG*, for ESFSs. Let the ESFS $E = (D, V, \Sigma, M)$ be fixed.

A proof-DAG for E is a finite directed acyclic graph that has a number of special properties. At each node there is a fixed linear ordering, called *left-to-right*, of the edges directed out of that node. There is exactly one node with in-degree zero, called the *root*. Each node with out-degree zero is called a *leaf* and the other nodes are called *internal nodes*. Every node has a label consisting of a ground atomic formula of E. For each node d, the nodes d' such that there is an edge from d to d' are called the *children* of d. Finally, for each internal node d, if its label is R and the labels of its children in left-to-right order are $R_1, \ldots, R_n$, then there is an axiom F in M and some substitution θ such that $F\theta = R \leftarrow R_1 \& \cdots \& R_n$, and for each leaf node d, if its label is R, then there is an axiom of the form $R \leftarrow$ in M.

The *size* of the proof-DAG T, denoted $size(T)$, is defined to be the sum of the number of nodes in T, the number of edges in T, and the sum of the lengths of the labels on all the nodes of T. The *depth* of a proof-DAG T is the maximum number of nodes in any directed path in T.

For any node of a proof-DAG T, the *sub-DAG rooted at* d is the induced subgraph of T on all the nodes reachable from d by a directed path in T. Note that this is also a proof DAG for the ESFS E.

A proof-DAG has the advantages similar to the ones which a parse-DAG has.

4 The Learning Algorithm

In this section, we demonstrate an efficient algorithm to learn ESFSs in the framework of learning by using queries to a teacher modeled on Angluin's approach to learning k-bounded context-free grammars. The learning algorithm may be viewed as a natural extension of the Angluin's algorithm in [2]. The lemmas and theorems that follow are analogous to Angluin's results.

Let k be any non-negative integer. An ESFS E is *k-bounded* iff every axiom of E has at most k occurrences of variables in the term of the conclusion and at most k premises. In this section, we show that there is a polynomial time algorithm to learn any k-bounded ESFS using queries.

Suppose $E_U = (D, V, \Sigma, M_U)$ is the unknown k-bounded ESFS and P is a predicate in E_U (that is the predicate which defines the language). This is the ESFS that is to be learned (up to P-equivalence) by the learning algorithm. We assume that k, D, Σ, and P are known to the learning algorithm, but M_U, the set of axioms, is unknown. The assumption that the predicate alphabet is known to the learning algorithm is the same as Shapiro's of incorporating theoretical predicates into the model inference problem [17].

4.1 Types of Queries

A *membership query* proposes a terminal string w and asks whether it is in $L(E_U, P)$. The reply is either *yes* or *no*.

A *predicate provable query* proposes a ground atomic formula $Q(w)$ of E_U and asks whether $Q(w)$ is provable from E_U. The reply is either *yes* or *no*. A membership query with w can be accomplished by a predicate provable query with $P(w)$. Predicate provable queries are analogous to non-terminal membership queries in [2].

An *equivalence query* proposes an ESFS E and asks whether $L(E_U, P) = L(E, P)$. The answer is either *yes* or *no*. If it is *no*, then a *counter-example* is also provided, that is, a terminal string w in the symmetric difference of $L(E_U, P)$ and $L(E, P)$. If $w \in L(E_U, P) - L(E, P)$, w is called a *positive* counter-example, and if $w \in L(E, P) - L(E_U, P)$, it is called a *negative* counter-example.

4.2 Correctness, Incorrectness

We define a formula $Q(t(x_1, \ldots, x_n)) \leftarrow Q_1(x_1) \& \cdots \& Q_n(x_n)$ to be *incorrect for* an ESFS $E = (D, V, \Sigma, M)$ iff there are n terminal strings $y_1, \ldots, y_n$ in Σ^* such that for $1 \leq i \leq n$, $Q_i(y_i)$ is provable from E, but $Q(t(y_1, \ldots, y_n))$ is not provable from E. A formula is *correct for* E iff it is not incorrect for E. Clearly, every axiom in M is correct for E.

4.3 The Learning Algorithm for ESFSs

The form $P(t(x_1, \ldots, x_n)) \leftarrow P_1(x_1) \& \cdots \& P_n(x_n)$ of axioms of ESFSs ensures an efficient learning for ESFSs. The learning algorithm LEE for ESFSs is given in Figure 1.

```
Algorithm $LEE$
Input : the teacher who knows the unknown $k$-bounded ESFS $E_U$ and
        can reply to predicate provable queries and equivalence queries;
Output : an ESFS $E$ such that $L(E,P) = L(E_U,P)$;
Main procedure :
%% Initialization
The set $M$ of axioms is initialized to the empty set.
Then the algorithm iterates the following loop.
%% Main loop
An equivalence query is made, proposing $E = (D, V, \Sigma, M)$.
If the reply is yes, the algorithm outputs $E$ and halts.
Otherwise, a counter-example $w$ is returned, and there are two cases.
    Case (a)
    If $w$ is in $L(E,P)$, then a proof-DAG for $E$ is found with root label $P(w)$.
    The proof-DAG is then diagnosed to find an axiom that is incorrect for $E_U$.
    This axiom is removed from $M$.
    Case (b)
    If $w$ is not in $L(E,P)$, then the set $C(w)$ of all candidate axioms is computed from $w$.
    All of them are added to $M$.
```

Figure 1: The learning algorithm LEE

The algorithm relies on three sub-procedures: proof, diagnosis, and the computation of candidate axioms. These are described in the following three subsections.

4.4 Proof Procedure

We use the algorithm described in the proof of Proposition 4 for the proof sub-procedure. We modify it so that on inputs a terminal string w, an ESFS E and a predicate P, if $P(w)$ is provable from E, a proof-DAG for E with root node labelled $P(w)$ is also returned.

The modification of the algorithm records information for a proof-DAG. The nodes of the proof-DAG will be elements of I. Suppose $Q(y)$ is added to I because the algorithm finds the axiom of the form

$$Q(t(x_1, \ldots, x_n)) \leftarrow Q_1(x_1)\& \cdots \&Q_n(x_n)$$

and the n-tuple of elements

$$\langle Q_1(y_1), \ldots, Q_n(y_n)\rangle$$

of I such that $y = t(y_1, \ldots, y_n)$ is a substring of w. Then for the element $Q(y)$ the algorithm adds an ordered list of n edges from $Q(y)$ to the elements $Q_1(y_1), \ldots, Q_n(y_n)$ of I.

At the end of the algorithm, if $P(w)$ is in I, a proof-DAG is constructed as follows. Discard from I all elements not reachable from $P(w)$ by a directed path of edges. Then the label of each node $Q(y)$ is just $Q(y)$. The root node is $P(w)$.

It is assumed that E is k-bounded.

Lemma 10 *There are a non-decreasing polynomial $p_1(x, y)$ such that the time used by the proof algorithm on input E, P and w is bounded by $p_1(size(E), |w|)$ and a non-decreasing polynomial $p_2(x, y)$ such that the proof-DAG returned by the proof algorithm on inputs E, w and P is of size bounded by $p_2(|D|, |w|)$.*

Proof. Every iteration of the loop except the last adds at least one element to I, so there are at most $|D|(|w| + 1)^2$ iterations of the loop. Each iteration of the loop considers at most $|M|$ axioms, and for each axiom with m premises, considers at most all m-tuples of elements of I. Since E is k-bounded, $m \leq k$. Hence the basic operation of applying a substitution to the term in the argument of the conclusion of an axiom and testing whether it is a substring of w is done no more than

$$|M|(|D|(|w|+1)^2)^{k+1}$$

times in all. The time for proving is bounded by a polynomial in the length of w and the size of E (but exponential in k).

Next clearly the proof-DAG has at most $|I|$ nodes, which is bounded by $|D|(|w|+1)^2$. Each node has a label of length bounded by $|w|+4$, and each node has at most k edges directed out of it. Thus the total size of the resulting proof-DAG is bounded by

$$|D|(|w|+1)^2(|w|+k+4).$$

This prove Lemma 10 Q.E.D.

4.5 Diagnosis Procedure

The diagnosis sub-procedure finds an axiom that is incorrect for E_U of the current ESFS $E=(D,V,\Sigma,M)$ rejected by an equivalence query with a negative counter-example. We can use Shapiro's diagnosis algorithm for Prolog programs [16] as the diagnosis procedure for ESFSs, because the ESFSs can be regarded as logic programs over strings. Thus the diagnosis procedure is essentially a special case of his algorithm for diagnosing an incorrect output. The input to the diagnosis procedure is a correct proof-DAG T for E such that $P(y)$ is the label of the root and $P(y)$ is not provable from E_U. The output of the diagnosis procedure is an axiom of E that is incorrect for E_U.

If the root of T has no child (that is, T consists of one node), then the diagnosis procedure returns the formula $P(y)\leftarrow$. Otherwise the diagnosis procedure considers in turn each child of the root of T. If the child is labelled with $Q(z)$, then the diagnosis procedure makes a predicate provable query with $Q(z)$. If the reply is *no*, then it calls itself recursively with the sub-DAG rooted at the child, and returns the resulting axiom. If the reply is *yes*, then it goes on to the next child of the root of T.

If all the queries are answered *yes*, then the diagnosis procedure finds an axiom F of E such that $F\theta = P(y)\leftarrow P_1(y_1)\&\cdots\&P_n(y_n)$ for some substitution θ, where $P_1(y_1),\ldots,P_n(y_n)$

are labels of the children of the root of T in left-to-right order, and returns F.

Lemma 11 *When the diagnosis procedure is given as input a proof-DAG T for E that has the root labelled $P(y)$ such that $P(y)$ is not provable from E_U, it returns an axiom of E that is incorrect for E_U. Further, there is a non-decreasing polynomial $p_3(x, y)$ such that the time required by the diagnosis procedure on input T is bounded by $p_3(size(T), |M|)$.*

Proof. If the input conditions are met on the initial call, then each recursive call preserves the input conditions. Since each recursive call is with a proper sub-DAG of its input DAG, the procedure must eventually terminate, and since the original DAG is a correct proof-DAG for E (as is every sub-DAG), the diagnosis procedure always finds an axiom of E and returns it.

If the formula $P(y) \leftarrow$ is returned, then it is clearly an axiom of E and incorrect for E_U. If the axiom of the form $P(t(x_1, \ldots, x_n)) \leftarrow P_1(x_1)\& \cdots \&P_n(x_n)$ is returned, then the queries have witnessed that there are n terminal strings $y_1, \ldots, y_n$ such that for $1 \leq i \leq n$, $P_i(y_i)$ is provable from E, but $P(t(y_1, \ldots, y_n)) = P(y)$ is not provable from E, that is, the axiom $P(t(x_1, \ldots, x_n)) \leftarrow P_1(x_1)\& \cdots \&P_n(x_n)$ is incorrect for E_U.

Next the number of queries made by the diagnosis procedure is at most k times the depth of the proof-DAG. The diagnosis procedure considers at most $|M|$ axioms to find the axiom F. It is clear that a straightforward implementation of the diagnosis procedure runs in time polynomial in the size of the input proof-DAG and $|M|$. This proves Lemma 11. Q.E.D.

4.6 Candidate Axioms

The input to the candidate axioms procedure is a terminal string w such that w is in $L(E_U, P)$ but not in $L(E, P)$, where $E = (D, V, \Sigma, M)$ is the current ESFS rejected by an equivalence query with a positive counter-example w, and the output is a set $C(w)$ of axioms added to E such that at least one element of $C(w)$ is in M_U but not in M.

The algorithm considers in turn every substring y of w. For every $m \leq k$ and every factorization of y into $2m + 1$ substrings,

$$y = u_0 v_1 u_1 v_2 u_2 \cdots v_m u_m,$$

and for every $1 \leq n \leq k$ and every $n+1$-tuple of predicates $Q, Q_1, \ldots, Q_n$ from D, the formula of the form

$$Q(u_0 x_1 u_1 x_1 u_2 \cdots x_m u_m) \leftarrow Q_1(z_1) \& \cdots \& Q_n(z_n)$$

is added to $C(w)$, where $x_1, \ldots, x_m$ are variables which are not necessarily distinct, $z_1, \ldots, z_n \in \{x_1, \ldots, x_m\}$ and $\{x_1, \ldots, x_m\} \subseteq \{z_1, \ldots, z_n\}$, and the formula of the form

$$Q(y) \leftarrow$$

is added to $C(w)$.

Lemma 12 *Let the candidate axioms procedure have input $w \in L(E_U, P) - L(E, P)$. Then $C(w)$ contains some axiom in M_U but not in M. Further, there are non-decreasing polynomials $p_4(x,y)$ and $p_5(x,y)$ such that on input w the candidate axioms procedure runs in time bounded by $p_4(|D|, |w|)$ and produces an output set $C(w)$ with at most $p_5(|D|, |w|)$ elements. Moreover, every axiom in $C(w)$ has a term of length at most $|w| + k$ with at most k occurrences of variables in the conclusion and at most k premises.*

Proof. Since w is in $L(E_U, P)$, there is a proof-DAG T for E_U with root labelled $P(w)$. We show that every axiom used in T is in $C(w)$.

Consider any sub-DAG T' of T rooted at a node with label $Q(y)$. Suppose the axiom used at the root of T' is

$$Q(t(x_1, \ldots, x_m)) \leftarrow Q_1(x_1) \& \cdots \& Q_m(x_m).$$

There are m substrings $y_1, \ldots, y_m$ of y such that

$$y = t(y_1, \ldots, y_m).$$

Since y is a substring of w and $m \leq k$, the axiom

$$Q(t(x_1, \ldots, x_m)) \leftarrow Q_1(x_1) \& \cdots \& Q_m(x_m).$$

will be generated from y using the above factorization and choices of predicates.

Thus every axiom used in T is in $C(w)$. If every axiom in $C(w) \cap M_U$ were in M, T would be a proof-DAG for the ESFS E witnessing $w \in L(E,P)$, a contradiction. Thus, $C(w)$ contains some axiom in $M_U - M$.

Next for each $m \leq k$, there are no more than $(|w|+1)^{2m}$ factorization of the string w into $2m+1$ substrings, and no more than m^m choices of m variables for the term of the conclusion and $k(m|D|)^k$ choices of atomic formulas for the premises. Thus, the total number of axioms placed in $C(w)$ is at most

$$1 + k(|w|+1)^{2k+2}k^k|D|k(k|D|)^k.$$

Computing these axioms takes time bounded by a polynomial in $|D|$ and $|w|$. The term in the conclusion of each axiom consists of a subsequence of the terminals in w and at most k variables, for a total length bounded by $|w|+k$, which completes the proof of Lemma 12. Q.E.D.

4.7 Correctness and Time Complexity

Since M is initially empty and is only augmented by axioms output by the candidate axioms procedure, E is k-bounded at all times, by Lemma 12. Clearly if the learning algorithm ever terminates, its output is an ESFS E which is P-equivalent to E_U. This shows the partial correctness of the learning algorithm, so we bound the number of iterations of the main loop as follows.

Lemma 13 *There are at most $|M_U|$ iterations of the case* (b) *of the main loop with positive counter-examples, and if Max_p is the maximum length of any positive counter-example, then there are at most $|M_U|p_5(|D|, Max_p)$ iterations of the case* (a) *of the main loop with negative counter-examples.*

Proof. By Lemma 12, each iteration of the case (b) with a positive counter-example must add to M at least one axiom in $M_U - M$. This axiom is correct for E_U and therefore cannot be removed from M, because the only elements removed from M are incorrect for E_U, by Lemma 11. Hence there are at most $|M_U|$ iterations of the case (b) with positive counter-examples.

Thus there are at most $|M_U|$ positive counter-examples. say

$$w_1, w_2, \ldots, w_r,$$

where $r \leq |M_U|$. Let Max_p be the maximum value of $|w_i|$ for $i = 1, 2, \ldots, r$.

The total number of axioms added to M is bounded by

$$|C(w_1)| + |C(w_2)| + \cdots + |C(w_r)|,$$

which by Lemma 12 is bounded by

$$p_5(|D|, |w_1|) + p_5(|D|, |w_2|) + \cdots + p_5(|D|, |w_r|).$$

Since $p_5(x, y)$ is non-decreasing we have a bound of

$$|M_U| p_5(|D|, Max_p)$$

on the total number of axioms ever added to M.

Each iteration of the case (a) with a negative counter-example removes one axiom from M. Hence this can happen at most as many times as there are axioms ever added to M, which is bounded by

$$|M_U| p_5(|D|, Max_p).$$

This proves Lemma 13. Q.E.D.

Thus the learning algorithm must terminate after at most

$$|M_U| + |M_U| p_5(|D|, Max_p)$$

iterations of the main loop.

Let Max_n be the maximum length of any negative counter-example, and let Max be the maximum of Max_p and Max_n. Max is the maximum length of any counter-example encountered before termination.

We bound the time used by the learning algorithm as follows.

Lemma 14 *The time required by the proof procedure at each iteration of the main loop of the learning algorithm is bounded by a polynomial in the size of E_U and the length of the longest counter-example.*

Proof. By Lemma 10 the time required to prove $P(w)$ for each counter-example w is bounded by $p_1(size(E), |w|)$. Clearly $|w| \leq Max$, but we need to establish a bound on $size(E)$.

We have

$$size(E) = |D| + |\Sigma| + |M| + L,$$

where L is the sum of the lengths of the terms in the conclusions of the axioms in M. By Lemma 13,

$$|M| \leq |M_U| p_5(|D|, Max_p).$$

Each axiom in M has a term of length at most $Max_p + k$ in the conclusion, by Lemma 12.

Hence,

$$size(E) \leq |D| + |\Sigma| + |M_U| p_5(|D|, Max_p)(1 + Max_p + k).$$

So the size of E can be bounded by a non decreasing polynomial in the size of E_U and the length of the longest positive counter-example,

$$size(E) \leq p_6(size(E_U), Max_p).$$

Thus at each iteration the time to prove $P(w)$ for a counter-example w is bounded by

$$p_1(p_6(size(E_U), Max_p), Max),$$

which proves Lemma 14. Q.E.D.

Lemma 15 *The time required by the diagnosis procedure at each iteration in which it is called is bounded by a polynomial in the size of E_U and the length of the longest counter-example.*

Proof. By Lemmas 10, 11 and 13, the time required by the diagnosis procedure at each iteration in which it is called is bounded by

$$p_3(p_2(|D|, Max), |M_U| p_5(|D|, Max_p)),$$

which is bounded by

$$p_3(p_2(size(E_U), Max), size(E_U) p_5(size(E_U), Max_p)).$$

Q.E.D.

Lemma 16 *The time required by the candidate axioms procedure at each iteration in which it is called is bounded by a polynomial in the size of E_U and the length of the longest counter-example.*

Proof. By Lemma 12, the time required by the candidate axioms procedure with input w is bounded by $p_4(|D|, |w|) \leq p_4(size(E_U), Max)$. Q.E.D.

Now we have the main theorem.

Theorem 17 *There is an algorithm that learns an ESFS P-equivalent to any k-bounded ESFS E_U using equivalence and predicate provable queries that runs in time polynomial in the size of E_U and the length of the longest counter-example.*

Proof. Putting the bounds from the above three lemmas together with the bound of

$$|M_U| + |M_U| p_5(|D|, Max_p)$$

on the number of iterations of the main loop of the learning algorithm, we conclude that the total time used by the learning algorithm is bounded by a polynomial in the size of E_U and the length of the longest counter-example. Q.E.D.

In the case that an ESFS E_U has only one predicate P, predicate provable queries reduce to membership queries, so we have the following.

Corollary 18 *There is an algorithm that learns an ESFS P-equivalent to any k-bounded ESFS E_U which has only one predicate P using equivalence and membership queries that runs in time polynomial in the size of E_U and the length of the longest counter-example.*

In the terminology of [4], languages defined by k-bounded ESFSs which have only one predicate can be learned efficiently from a *minimally adequate teacher*. Note that this sub-class of ESFSs contains a class of EFSs introduced in [18], called *primitive formal systems*, which is learnable from positive data.

It is clear that there is no m-bounded standard form of ESFSs for any m, that is, no m exists such that for any ESFS E there is a m-bounded ESFS E' which defines the language $L(E, P)$. Thus for any k the k-bounded ESFSs are strictly less powerful than the $k+1$-bounded ESFSs, that is,

$$C_{1\text{-}ESFS} \subsetneqq C_{2\text{-}ESFS} \subsetneqq \cdots \subsetneqq C_{k\text{-}ESFS} \subsetneqq C_{k+1\text{-}ESFS} \subsetneqq \cdots,$$

where $C_{k\text{-}ESFS}$ denotes the class of languages defined by k-bounded ESFSs.

5 Concluding Remarks

There has been no paper to investigate a computational learning problem for more than context-free languages. In this paper we have proposed a method for efficient learning of a larger class of formal languages than context-free languages. This is the first attempt to consider the problem to learn a large class of formal languages (e.g. context-sensitive languages) in a reasonable amount of time in a teacher and learner paradigm, which is one of central issues in concept learning. We have introduced a new class of expressions for learning of formal languages defined in Smullyan's elementary formal systems and shown that this class has a desirable feature that it contains some important classes of formal languages like context-free languages and pattern languages, which enables us to take an unified view of learning formal languages.

The characterization of the class of languages defined by ESFSs must be investigated. Especially it is important to analyze the relation between the context-sensitive languages and the languages defined by ESFSs. We can also regard formulas in elementary formal systems as Horn formulas in logic programming [11] over strings in which only the concatenation is used as the function. It is interesting to compare ESFSs with *Definite Clause Grammars* [12].

Acknowledgements

The author would like to thank Dr. T.Kitagawa, the president of IIAS-SIS, Dr. H.Enomoto, the director of IIAS-SIS, for giving him the opportunity to pursue this work and warm encouragement. Discussions with the colleagues T.Yokomori and Y.Takada were also very fruitful.

This is part of the work in the major R&D of the Fifth Generation Computer Project, conducted under program set up by MITI.

References

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

[2] D. Angluin. *Learning k-bounded context-free grammars.* RR 557, YALEU/DCS, 1987.

[3] D. Angluin. *Learning k-term DNF formulas using queries and counterexamples.* RR 559, YALEU/DCS, 1987.

[4] D. Angluin. Learning regular sets from queries and counter-examples. *Information and Computation*, 75:87–106, 1987.

[5] D. Angluin. Learning with hints. In *Proceedings of 1st Workshop on Computational Learning Theory*, pages 167–181, 1988.

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

[7] S. Arikawa. Elementary formal systems and formal languages - simple formal systems. *Memoirs of the faculty of science, Kyushu university, Series A*, 24:47–75, 1970.

[8] P. Berman and R. Roos. Learning one-counter languages in polynomial time. In *Proceedings of IEEE FOCS '87*, pages 61–67, 1987.

[9] M. A. Harrison. *Introduction to Formal Language Theory.* Addison-Wesley, 1978.

[10] L. Y. Liu and P. Weiner. An infinite hierarchy of intersections of context-free languages. *Mathematical Systems Theory*, 7:185–192, 1973.

[11] J. W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 1984.

[12] F. C. N. Pereira and D. H. D. Warren. Definite clause grammars for language analysis - a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13:231–278, 1980.

[13] Y. Sakakibara. *Inductive inference of logic programs based on algebraic semantics.* Research Report 79, IIAS-SIS, FUJITSU LIMITED, 1987.

[14] Y. Sakakibara. Learning context-free grammars from structural data in polynomial time. In *Proceedings of 1st Workshop on Computational Learning Theory*, pages 296–310, 1988. To appear in *Theoretical Computer Science.*

[15] A. Salomaa. *Formal Languages.* Academic Press, Inc., 1973.

[16] E. Y. Shapiro. *Algorithmic program debugging.* PhD thesis, Yale University Computer Science Dept., 1982. Published by MIT Press, 1983.

[17] E. Y. Shapiro. *Inductive inference of theories from facts.* Technical Report 192, Yale University Computer Science Dept., 1981.

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

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