An Efficient Prover for Elementary Formal Systems

Harada, Naoyuki
Department of Informatics, Faculty of Information Science and Electrical Engineering : Graduate Student

Arikawa, Setsuo
Department Informatics, Faculty of Information Science and Electrical Engineering, Kyushu University

Ishizaka, Hiroki
Department of Artificial Intelligence, Kyushu Institute of Technology

https://doi.org/10.15017/1474990

出版情報：九州大学大学院システム情報科学紀要. 2 (1), pp.1-8, 1997-03-26. 九州大学大学院システム情報科学研究院
バージョン：
権利関係：

九州大学大学院
システム情報科学研究科報告
第2巻 第1号 平成9年3月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.2, No.1, March 1997

An Efficient Prover for Elementary Formal Systems

Naoyuki HARADA*, Setsuo ARIKAWA** and Hiroki ISHIZAKA***

(Received December 24, 1996)

Abstract: The elementary formal system(EFS, for short) is a kind of logic programs over the domain of strings. The EFS's can define various classes of formal languages, and a prover, i.e., a resolution procedure, for EFS's works as a uniform recognizer of such various classes of languages. The basic operation of the procedure is the unification as in the ordinary resolution. This paper deals with a prover for a special subclass of EFS's called regular EFS's that defines exactly the same class as context-free languages in the Chomsky hierarchy. The prover searches derivation trees for refutations. This paper first introduces a subclass of regular EFS's, called variable-separated EFS's, for which the prover becomes efficient, proves that the newly introduced EFS's are equivalent to the regular EFS's in their language defining power, and shows an experimental result on parsing some simple Japanese sentences.

Keywords: Elementary formal systems, Formal languages, Parsing algorithm, Unification algorithm

1. Introduction

The elementary formal system (EFS, for short) Smullyan invented[3] to develop his recursive function theory is a kind of logic programs over the domain of strings of characters. The EFS's can define various classes of formal languages, and a resolution procedure for EFS's works as a uniform recognizer of such various classes of languages.

In the formal language theory, several classes of automata such as finite automata, pushdown automata, linear bounded automata, and Turing machines are used as recognizers of such classes of languages. Their basic mechanisms are different from one to another. Hence, we need to change the automata depending on the classes of languages we are interested in. However, as far as EFS's are concerned, we do not need to change the basic devices. We may just use the resolution principle as a uniform device.

Several subclasses of EFS's have been considered; variable-bounded EFS's, length-bounded EFS's, regular EFS's and linear EFS's and so on[4)5)]. Among them the regular EFS's equivalent to the class of context-free grammars are the most important in discussing programming languages[1)].

We implemented the prover, i.e., the resolution procedure, for regular EFS's. It is a kind of recognizers of the formal languages. Given a regular EFS and a string, the prover determines whether the string is proved by the regular EFS's, or not. In fact, the prover searches for a refutation in a tree, a derivation tree, produced from the EFS and the string.

In case ill-formed EFS's are given, the prover does not work efficiently. The inefficiency is mainly due to *consecutiveness of variables* in clauses. The consecutive variables spread many failed branches which can not be a refutation in a derivation tree. Hence it takes a lot of time to search such trees for refutations.

To get rid of *consecutive variables* in clauses, we propose a new subclass of EFS's called variable-separated EFS's by putting a restriction on the clauses in the regular EFS's. In a variable-separated EFS, any pattern in the heads of clauses must start and end with some constant symbols, and variables in the term must be separated from each other.

In this paper first we recall some basic definitions on EFS's necessary for our discussions. In Section 3 we define derivations for EFS's and derivation trees. In Section 4, we introduce the variable-separated EFS's, and prove that the power of them is the same as that of regular EFS's although they are defined by posing some restrictions on the regular EFS's. Thus our prover for variable-separated EFS's can be applied to all context-free languages. In Section 5 we give an experimental result on parsing some simple Japanese sentences.

* Department of Informatics, Graduate Student
** Department of Informatics
*** Department of Artificial Intelligence, Kyushu Institute of Technology

2. Elementary Formal Systems

We start with recalling the basic notions on EFS's. Let Σ, X, and Π be mutually disjoint sets, and let Σ and Π be finite. We refer to Σ as *alphabet*, and to each element of it as symbol which will be denoted by $a, b, c, \ldots$, to each element of X as *variable*, denoted by $x, y, z, x_1, x_2, \ldots$ and to each element of Π as *predicate symbol*, denoted by $p, q, q_1, q_2, \ldots$, each of which has an arity. A^+ denotes the set of all nonempty words over a set A.

Definition A *term* is an element of $(\Sigma \cup X)^+$. Each term is denoted by $\pi, \tau, \pi_1, \pi_2, \ldots, \tau_1, \tau_2, \ldots$. A *ground term* of S is an element of Σ^+. Terms are also called *patterns*.

Definition An *atomic formula* (or *atom*, for short) is an expression of the form $p(\tau_1, ..., \tau_n)$, where p is a predicate symbol in Π with arity n and $\tau_1, ..., \tau_n$ are terms. The atom is *ground* if all $\tau_1, ..., \tau_n$ are ground.

Clauses, empty clauses ($\square$), *ground clauses and substitutions* are defined in the ordinary way[2].

Definition A *definite clause* is a clause of the form

$$A \leftarrow B_1, ..., B_n \quad (n \geq 0).$$

Definition An *elementary formal system* (EFS, for short) S is a triplet (Σ, Π, Γ), where Γ is a finite set of definite clauses, which are sometimes called *axioms* of S.

We denote a substitution by $\{x_1 := \tau_1, ..., x_n := \tau_n\}$, where x_i are mutually distinct variables. We also define

$$p(\tau_1, ..., \tau_n)\theta = p(\tau_1\theta, ..., \tau_n\theta),$$
$$(A \leftarrow B_1, ..., B_m)\theta = A\theta \leftarrow (B_1\theta, ..., B_m\theta),$$

for substitution θ, an atom $p(\tau_1, ..., \tau_n)$ and a clause $A \leftarrow B_1, ..., B_m$.

Definition Let $S = (\Sigma, \Pi, \Gamma)$ be an EFS. We define the relation $\Gamma \vdash C$ for a clause C of S inductively as follows:

(1) If $\Gamma \ni D$, then $\Gamma \vdash D$.
(2) If $\Gamma \vdash D$, then $\Gamma \vdash D\theta$ for any substitution θ.
(3) If $\Gamma \vdash A \leftarrow B_1, ..., B_n$ and $\Gamma \vdash B_n \leftarrow$, then $\Gamma \vdash A \leftarrow B_1, ..., B_{n-1}$.

C is *provable from* Γ if $\Gamma \vdash C$.

Definition For an EFS $S = (\Sigma, \Pi, \Gamma)$ and $p \in \Pi$ with arity n, we define

$$L(S, p) = \{(\alpha_1, ..., \alpha_n) \in (\Sigma^+)^n \mid \Gamma \vdash p(\alpha_1, ..., \alpha_n) \leftarrow\}$$

In case $n = 1$, $L(S, p)$ is a language over Σ. A language $L \subseteq \Sigma^+$ is *definable by EFS* or an *EFS language* if such S and p exist.

Now we give definition of regular EFS's which can define context-free languages.

Definition Let $o(x, \pi)$ be the number of all occurrences of a variable x in term π. A pattern π is *regular* if $o(x, \pi) \leq 1$ for any variable x in π. A definite clause C is *regular* if C is of the form

$$p(\pi) \leftarrow q_1(x_1), ..., q_n(x_n),$$

and π is regular. An EFS $S = (\Sigma, \Pi, \Gamma)$ is *regular* if each clause in Γ is regular.

Theorem 2.1 (Arikawa et al.[5]) A language L is definable by regular EFS if and only if L is a context-free language.

Example 2.1 An EFS $S = (\{a, b\}, \{p\}, \Gamma)$ with

$$\Gamma = \left\{ \begin{array}{c} p(axb) \leftarrow p(x), \\ p(ab) \leftarrow \end{array} \right\}$$

is regular. It defines a context-free language $L(S, p) = \{a^n b^n \mid n \geq 1\}$.

3. Derivations and Refutations for EFS's

EFS's have an aspect of logic programs. Hence a resolution procedure for EFS's can be considered and works as a device for recognizing formal languages. In this section, we discuss the derivations and refutations for EFS's.

Definition Let α and β be a pair of terms or atoms. Then a substitution θ is a *unifier* of α and β if $\alpha\theta = \beta\theta$.

Definition A *goal clause* (or *goal*, for short) of EFS S is a clause of the form

$$\leftarrow B_1, ..., B_n \quad (n \geq 0).$$

Definition If clauses C and D are identical except renaming of variables, that is, $C = D\theta$ and $C\sigma = D$ for some substitutions θ and σ, we say D is a *variant* of C and write $C \equiv D$.

We assume a *computation rule* R to select an atom from a goal.

Definition Let S be an EFS, G be a goal of S, and $v(C)$ be the set of all variables in a clause C. A *derivation from* G is a (finite or infinite) sequence of triplets (G_i, θ_i, C_i) $(i = 0, 1, ...)$ which satisfies the following conditions:

(1) G_i is a goal, θ_i is a substitution, C_i is a variant of an axiom of S, and $G_0 = G$.

(2) $v(C_i) \cap v(C_j) = \emptyset$ for every i and j ($i \neq j$), and $v(C_i) \cap v(G_i) = \emptyset$ for every i.

(3) If G_i is $\leftarrow A_1, ..., A_k$ and A_m is the atom selected by R, then C_i is $A \leftarrow B_1, ..., B_q$, and θ_i is a unifier of A and A_m, and G_{i+1} is $(\leftarrow A_1, ..., A_{m-1}, B_1, ..., B_q, A_{m+1}, ..., A_k)\,\theta_i$. A_m is a *selected atom* of G_i, and G_{i+1} is a *resolvent* of G_i and C_i by θ_i.

Definition A *refutation* is a finite derivation ending with the empty goal $\square$.

Now we show a theorem, which asserts that a ground atom A is provable from EFS if and only if there is a refutation from $\leftarrow A$. Let $S = (\Sigma, \Pi, \Gamma)$ be an EFS, and $B(S)$ be the set of all ground atoms, i.e., the *Herbrand base* of S. Then we define the following sets.

$$SS(S) = \{A \in B(S) \mid \text{there exists a refutation from } \leftarrow A\},$$
$$PS(S) = \{A \in B(S) \mid \Gamma \vdash A\}.$$

Theorem 3.1 (Arikawa et al.) $SS(S) = PS(S)$ for every EFS S.

Example 3.1 For an EFS $S = (\{a,\ b\},\ \{p\},\ \Gamma)$ with

$$\Gamma = \left\{ \begin{array}{c} p(axy) \leftarrow p(x), p(y) \\ p(ab) \leftarrow \\ p(b) \leftarrow \end{array} \right\},$$

a refutation from $\leftarrow p(aabb)$ is illustrated by **Fig.1**, where the computation rule R selects the leftmost atom from every goal.

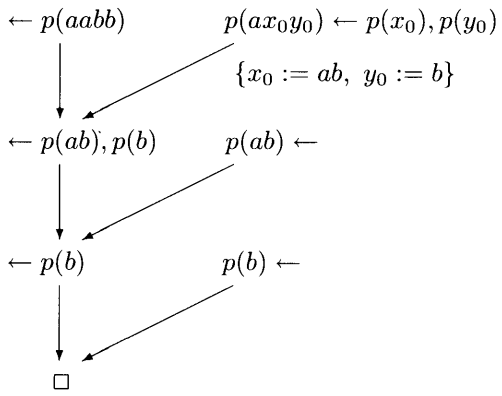

Fig.1 A refutation

There may exist two or more derivations from a goal. In the above example, if we use a substitution $\{x_0 := a,\ y_0 := bb\}$ instead of the substitution $\{x_0 := ab,\ y_0 := b\}$, we can get another derivation. Now we consider the derivation tree which expresses all such derivations together.

Definition Let S be an EFS, and G be a goal of S. Then the *derivation tree from* G is defined as follows:

(1) Each *node* of the tree is a goal.

(2) A *root node* of the tree is G.

(3) Let $\leftarrow A_1, ..., A_m, ..., A_k$ $(k \geq 1)$ be a node, and A_m be a selected atom. If $A \leftarrow B_1, ..., B_q$ is a clause of S and θ is a unifier of A and A_m, then a child of the node is

$$\leftarrow (A_1, ..., A_{m-1}, B_1, ..., B_q, A_{m+1}, ..., A_k)\theta.$$

(4) A node of empty goal does not have any children.

A derivation tree in example 3.1 is illustrated by **Fig.2**.

4. An Efficient Prover for EFS's

Given a set of axioms and a ground goal, the prover outputs "yes" and a refutation from the goal if any, otherwise it outputs "no". The prover searches a derivation tree for the empty goal by a depth-first-search. Hence if we can reduce branchings in the derivation tree, we can make the prover more efficient. A branching is caused by the following steps in derivations:

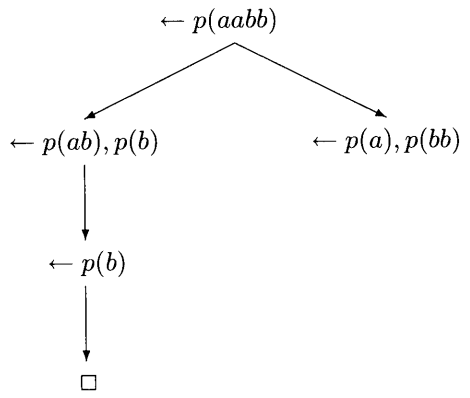

Fig.2 A derivation tree

1. Selecting an axiom whose head can be unified with the goal. Two or more clauses may be selected as in the ordinary logic programming.
2. Selecting an unifier. Unlike the ordinary logic programming two or more unifiers may exist for a pair of two terms, which is due to:
   (a) Consecutiveness of variables in a head of the selected axiom. For example, in unifying two atoms $p(abba)$ and $p(axy)$, there exist two unifiers $\{x := b,\ y := ba\}$ and $\{x := bb,\ y := a\}$.
   (b) Selecting a location of constant strings in the goal which are substrings of the term in the head of the axiom. Some constant strings in an axiom may appear in several locations in a goal. For example, in unifying $p(abba)$ and $p(xby)$, a constant string b in xby locates twice in $abba$, and two unifiers $\{x := a,\ y := ba\}$ and $\{x := ab,\ y := a\}$ are obtained.

The number of branchings depends on the number of axioms in Case 1, and on the length of a term in a goal in Case 2. The number of axioms is fixed, while the lengths of the terms vary depending on the goals. The branchings due to Case 1 are common to all the logic programming. In this paper we try to reduce the branchings due to Case 2. However, we can not cope with the branchings due to Case 2(b), because the number of locations where the constant strings in the head of the selected atom appear depends on the input strings in the given goal which may range all over the Σ^+. Hence only the way we can take is to reduce the branchings due to Case 2(a). We solve this problem by transforming the given EFS's to EFS's with a good property. First we introduce a notion of such good EFS's, and show that they are equivalent to the original ones in language defining power.

Definition A regular EFS $S = (\Sigma, \Pi, \Gamma)$ is *variable-separated* if the term in the head of each clause in Γ is of the form

$$\alpha x_1 \alpha_1 \ldots x_n \alpha_n \quad (\alpha, \alpha_i \in \Sigma^+,\ x_i \in X,\ n \geq 0).$$

In the variable-separated EFS's, every variable must be surrounded by constant strings on both sides, by virtue of which we can reduce the branchings due to Case 2(a). The variable-separated EFS's are a special class of regular EFS's. However, they are equivalent to the unrestricted ones. We prove this in the rest of this section.

By a *p-clause* we mean an axiom with predicate symbol p on its head. Then we can easily prove the following lemma.

Lemma 4.1 Let $S = (\Sigma, \Pi, \Gamma)$ be a regular EFS. Let $p(\alpha_1 x \alpha_2) \leftarrow A_1, q(x), A_2$ be an axiom in Γ and $q(\beta_1) \leftarrow B_1, \ldots, q(\beta_r) \leftarrow B_r$ be the set of all q-clause in Γ, where $\alpha_i \in (\Sigma \cup X)^*$ and $\beta_i \in (\Sigma \cup X)^+$, and A_i, B_i are finite (possibly empty) sequences of atoms. Let $S' = (\Sigma, \Pi, \Gamma')$ be a regular EFS obtained from S by deleting the axiom $p(\alpha_1 x \alpha_2) \leftarrow A_1, q(x), A_2$ from Γ and adding the axioms $p(\alpha_1 \beta_i \alpha_2) \leftarrow A_1, B_i, A_2 (1 \leq i \leq r)$. Then $L(S, p) = L(S', p)$.

We need two more lemmas.

Lemma 4.2 Let $S = (\Sigma, \Pi, \Gamma)$ be a regular EFS, $\{p(x\alpha_j) \leftarrow p(x), A_j \mid 1 \leq j \leq r\}$ be the set of p-clauses such that p is the leftmost predicate symbol in the body of an axiom in S, and $\{p(\beta_i) \leftarrow B_i \mid 1 \leq i \leq s\}$ be the remaining p-clauses in S. Construct an EFS $S' = (\Sigma, \Pi \cup \{q\}, \Gamma')$ by replacing all the p-clauses with the following clauses:

$$
\begin{array}{l}
p(\beta_i) \leftarrow B_i \\
p(\beta_i\alpha_j) \leftarrow B_i, A_j \\
p(\beta_i x\alpha_j) \leftarrow B_i, q(x), A_j \\
q(\alpha_j) \leftarrow A_j \\
q(\alpha_j\alpha_k) \leftarrow A_j, A_k \\
q(\alpha_j x\alpha_k) \leftarrow A_j, q(x), A_k \\
\qquad (1 \leq i \leq s,\ 1 \leq j,k \leq r)
\end{array}
$$

Then $L(S,p) = L(S',p)$.

Proof We first prove that if $w \in L(S,p)$ then $w \in L(S',p)$.

Suppose a goal $\leftarrow p(w)$ is given in S. If $p(\beta_i) \leftarrow B_i$ is selected at the first step, it is also selected in a refutation from $\leftarrow p(w)$ in S'. Next we consider the case that $p(x\alpha_i) \leftarrow p(x), A_i$ is selected. The predicate symbol of leftmost atom in a resolvent is p so that the axiom of this form can be selected many times, say n times, until the axiom of the form $p(\beta_i) \leftarrow B_i$ is selected. Let $\theta_n, \ldots, \theta_1$ be the unifiers used in these n resolutions. Then the leftmost atom with a predicate symbol p is replaced with $B_j\theta$ by selecting $p(\beta_j) \leftarrow B_j$ and a unifier θ. The resolvent, i.e., the goal at this point, is $\leftarrow B_j\theta, A_{i_1}\theta_1, \ldots, A_{i_n}\theta_n$.

These resolutions in S can be traced in S' in the following way. If $n = 1$, $p(\beta_i\alpha_j) \leftarrow B_i, A_j$ is selected in a refutation from $\leftarrow p(w)$ in S'. If k is an even number, $q(\alpha_j) \leftarrow A_j$ is selected after $p(\beta_i x\alpha_j) \leftarrow B_i, q(x), A_j$ is selected, and then axioms of the form $q(\alpha_j x\alpha_k) \leftarrow A_j, q(x), A_k$ are selected $(n-2)/2$ times in the refutation. If n is an odd number, $q(\alpha_j\alpha_k) \leftarrow A_j, A_k$ is selected after $p(\beta_i x\alpha_j) \leftarrow B_i, q(x), A_j$ is selected, and then axioms of the form $q(\alpha_j x\alpha_k) \leftarrow A_j, q(x), A_k$ are selected $(n-3)/2$ times.

The other parts of refutaions in S are exactly the same as those in S'. Thus the refutations in S can be translated as those in S'. Hence we have $L(S,p) \subseteq L(S',p)$. The converse can be proved in a similar way. □

Lemma 4.3 A similar statement to Lemma 4.2 is valid for p-clauses of the form $p(\alpha_j x) \leftarrow A_j, p(x)$.

By using these lemmas we can show the following theorem.

Theorem 4.1 For any regular EFS S and any predicate symbol p, there exists a variable-separated EFS S' and a predicate symbol p' in S' such that $L(S,p) = L(S',p')$.

Proof Let $S = (\Sigma, \Pi, \Gamma)$ be a regular EFS and let $\Pi = \{p_1, \ldots, p_m\}$. First we transform S into an EFS in which clauses of the form

$$p_i(x_j\alpha x_k) \leftarrow p_j(x_j), A, p_k(x_k),$$

are its axioms only if $i < j,k$, where $\alpha \in (\Sigma \cup X)^*$, *and* A is a finite sequence of atoms.

This transformation is carried out from p_1-clauses to p_m-clauses in the following way. Suppose the clauses have been transformed so that for any i $(1 \leq i < l)$, $p_i(x_j\alpha x_k) \leftarrow p_j(x_j), A, p_k(x_k)$ is an axiom only if $i < j,k$. Now let us transform the p_l-clause. If a p_l-clause is either of the forms:

$$
\begin{array}{l}
p_l(x_j\alpha) \leftarrow p_j(x_j), A, \\
p_l(\alpha x_k) \leftarrow A, p_k(x_k) \quad (l > j,k),
\end{array}
$$

we introduce a new set of clauses by replacing $p_j(x_j)$ and $p_k(x_k)$ with bodies of p_j-clauses and p_k-clauses according to Lemma 4.1. By repeating this process at most $2(l-1)$ times, we obtain clauses of the form:

$$p_l(x_{j'}\alpha' x_{k'}) \leftarrow p_{j'}(x_{j'}), A', p_{k'}(x_{k'}) \quad (l \leq j', k'),$$

where both ends of the terms in the head may possibly be constant symbols. If $l = j'$ or $l = k'$, the p_l-clauses are replaced according to Lemma 4.2 and 4.3, introducing a new predicate symbol q_l.

By the above process, we have clauses only of the forms:

$$
\begin{array}{l}
p_i(a_1\gamma a_2) \leftarrow A, \\
p_i(x_j\gamma a_2) \leftarrow p_j(x_j), A, \\
p_i(a_1\gamma x_k) \leftarrow A, p_k(x_k), \\
p_i(x_j\gamma x_k) \leftarrow p_j(x_j), A, p_k(x_k), \\
q_i(\gamma) \leftarrow A, \\
q_i(\gamma x_i\gamma') \leftarrow A, p_i(x_i), A', \\
\qquad (i < j,k,\ a_1, a_2 \in \Sigma,\ \gamma, \gamma' \in (\Sigma \cup X')^*),
\end{array}
$$

where X' consists of variables introduced when Lemma 4.2 and 4.3 are applied, and A, A' are finite sequences of atoms whose predicate symbols are in $\Pi \cup \{q_1, \ldots, q_{i-1}\}$.

Both ends of terms in the heads of p_m-clauses are constant symbols, because m is the largest index of p. By using Lemma 4.1, we transform all the p_i-clauses from $m-1$ down to 1 until constant symbols appear at the both ends of terms in the

heads. Then we obtain clauses of the forms:

$$p_i(a_1\gamma a_2) \leftarrow A,$$
$$q_i(\gamma) \leftarrow A,$$
$$q_i(\gamma x_i \gamma') \leftarrow A, p_i(x_i), A'.$$

Now we transform p_i-clauses with bodies into variable-separated clauses, by introducing new predicate symbol r_i and replacing bodies of p_i-clauses with $r_i(x_i)$. Then all of the p_i-clauses are transformed into variable-separated clauses of the forms:

$$p_i(a_1 x_i a_2) \leftarrow r_i(x_i),$$
$$q_i(\gamma) \leftarrow A,$$
$$q_i(\gamma x_i \gamma') \leftarrow A, p_i(x_i), A',$$
$$r_i(\gamma) \leftarrow A.$$

Finally, we deal with q_i- and r_i-clauses. We can show that indices of predicate symbols in the bodies of any q_i- and r_i-clauses are less than those in their heads. Since all the predicate symbols in the bodies of q_1-clauses are p_i for some i, we can transform q_1-clauses into variable-separated clauses according to lemma 4.1. Applying this process to the clauses with $q_2, \ldots, q_m, r_1, \ldots, r_m$ as heads in this order, we can transform all the clauses into variable-separated clauses. Thus we have a variable-separated EFS S' such that $L(S,p) = L(S',p')$. □

Example 4.1 Let $S = (\{a,b\}, \{p_1,p_2,p_3\}, \Gamma)$ be a regular EFS with the following Γ:

$$p_1(xy) \leftarrow p_2(x), p_3(y),$$
$$p_2(ax) \leftarrow p_2(x),$$
$$p_2(a) \leftarrow,$$
$$p_3(yb) \leftarrow p_3(y),$$
$$p_3(b) \leftarrow .$$

The EFS S defines a regular language $L(S,p_1) = \{a^m b^n \mid m,n \geq 1\}$. Let us transform S into a variable-separated the regular EFS S'. The indices of predicates in the body of the first axiom is greater than the index in its head. Hence we start with the axiom $p_2(ax) \leftarrow p_2(x)$ and $p_3(yb) \leftarrow p_3(y)$, and apply Lemma 4.2 and 4.3. Then we have the following set of clauses:

$p_1(xy) \leftarrow p_2(x), p_3(y)$,
$p_2(a) \leftarrow$, $p_3(b) \leftarrow$,
$p_2(aa) \leftarrow$, $p_3(bb) \leftarrow$,
$p_2(axa) \leftarrow q_2(x)$, $p_3(byb) \leftarrow q_3(y)$,
$q_2(a) \leftarrow$, $q_3(b) \leftarrow$,
$q_2(aa) \leftarrow$, $q_3(bb) \leftarrow$,
$q_2(axa) \leftarrow q_2(x)$, $q_3(byb) \leftarrow q_3(y)$.

We now apply Lemma 4.1 to the p_1-clause. Then we have the desired set of clauses each of which is a variable-separated clause:

$p_1(ab) \leftarrow$, $p_1(aabyb) \leftarrow q_3(y)$,
$p_1(abb) \leftarrow$, $p_1(axab) \leftarrow q_2(x)$,
$p_1(abyb) \leftarrow q_3(y)$, $p_1(axabb) \leftarrow q_2(x)$,
$p_1(aab) \leftarrow$, $p_1(axabyb) \leftarrow q_2(x), q_3(y)$,
$p_1(aabb) \leftarrow$,
$p_2(a) \leftarrow$, $p_3(b) \leftarrow$,
$p_2(aa) \leftarrow$, $p_3(bb) \leftarrow$,
$p_2(axa) \leftarrow q_2(x)$, $p_3(byb) \leftarrow q_3(y)$,
$q_2(a) \leftarrow$, $q_3(b) \leftarrow$,
$q_2(aa) \leftarrow$, $q_3(bb) \leftarrow$,
$q_2(axa) \leftarrow q_2(x)$, $q_3(byb) \leftarrow q_3(y)$.

As a more practical example, we have transformed all the Pascal grammar, which can be described in a regular EFS, into an equivalent variable-separated EFS. Due to limitation of space, we do not go into further detail.

By giving the prover a variable-separated EFS which is equivalent to the original regular EFS, we can reduce not only the number of branchings in the derivation tree but also the height of the tree, i.e., the length of the longest branch from the root to a leaf. Note here that given a ground goal, in general, a derivation tree for a regular EFS may be infinite. In fact, in the derivation for a regular EFS, if an axiom which does not contain any constant symbol is selected, the total length of terms in the resolvent, a new goal, does not decrease at all. In the worst case that an axiom $p(x) \leftarrow q(x)$ is selected and another axiom $q(x) \leftarrow p(x)$ exists, the tree becomes infinite. However, the trees for our variable-separated EFS's are always finite.

In this paper we are mainly interested in the refutations starting with ground goals. In such refutations the heights of trees depend on the lengths of ground goals given to the prover.

Now we compare the heights of derivation trees for regular EFS's and those for variable-separated EFS's equivalent to them. Here we deal with regular EFS's for which every derivation tree from a ground goal is finite, that is, the pathological case mentioned above may not happen.

Let n be the length of an input string, i.e., a term in the ground goal given to the prover. For a regular EFS the length of the term in the derivation tree decreases by at least one at every step of the derivation. Hence, the height of the derivation tree is n in the worst case. On the other hand, for a variable-separated EFS, the length decreases by at least two at every step, because the both ends of the term in each clause are constant symbols. Hence, the height of the tree does not exceed $n/2$. This shows that we can make the prover faster if we use the variable-separated EFS's instead of the regular EFS's.

5. An Experimental Result

In this section we show how the prover for variable-separated EFS's works efficiently by an example. The following are a regular EFS and a variable-separated EFS which define the same simple Japanese sentences.
The "は", "が" and "を", "に" below are Japanese postpositional particles which change (pro)nouns into the subjective cases and the objective cases, respectively. The "あなた", "彼", "彼女" and "彼ら" are Japanese stems of pronouns which make pronouns which correspond to, for example, "you", "he", "she" and "they", respectively, followed by the postpositional particles "は" or "が". The meanings of the other Japanese words are shown in the comments.

1. A regular EFS S with the following 36 axioms.

```
sentence(XYZ)<-subject(X),pred(Y),
                              period(Z)
subject(XYZ)<-noun(X),post(Y),comma(Z)
subject(WXYZ)<-modifier(W),noun(X),
                      post(Y),comma(Z)
pred(X)<-verb(X)
pred(X)<-adj(X)
pred(XY)<-modifier(X),verb(Y)
pred(XY)<-modifier(X),adj(Y)
modifier(X)<-adj(X)
modifier(XY)<-noun(X),post(Y)
noun(あなた)<-
noun(彼)<-
noun(彼女)<-
noun(彼ら)<-
noun(犬)<-        /* noun(dog)<-       */
noun(鳥)<-        /* noun(bird)<-      */
noun(花)<-        /* noun(flower)<-    */
noun(大空)<-      /* noun(sky)<-       */
noun(自由)<-      /* noun(freedom)<-   */
noun(海岸)<-      /* noun(beach)<-     */
post(は)<-
post(が)<-
post(を)<-
post(に)<-
adj(おおきな)<-  /* adj(big)<-         */
adj(ひろい)<-    /* adj(wide)<-        */
adj(あおい)<-    /* adj(blue)<-        */
adj(きれいな)<-  /* adj(beautiful)<-   */
adj(ちいさい)<-  /* adj(little)<-      */
verb(飛ぶ)<-     /* verb(fly)<-        */
verb(走る)<-     /* verb(run)<-        */
verb(植える)<-   /* verb(plant)<-      */
verb(見る)<-     /* verb(see)<-        */
verb(放す)<-     /* verb(release)<-    */
verb(降る)<-     /* verb(fall)<-       */
comma(, )<-
period(. )<-
```

2. A variable-separated EFS S' with 506, some of which are listed below.

```
sentence(あなた X, Y. )<-post(X),pred(Y)
sentence(彼 X, Y. )<-post(X),pred(Y)
sentence(彼女 X, Y. )<-post(X),pred(Y)
sentence(彼ら X, Y. )<-post(X),pred(Y)
sentence(犬 X, Y. )<-post(X),pred(Y)
sentence(鳥 X, Y. )<-post(X),pred(Y)
sentence(花 X, Y. )<-post(X),pred(Y)
sentence(大空 X, Y. )<-post(X),pred(Y)
sentence(自由 X, Y. )<-post(X),pred(Y)
sentence(海岸 X, Y. )<-post(X),pred(Y)
          :
adj(おおきな)<-
adj(ひろい)<-
adj(あおい)<-
adj(きれいな)<-
adj(ちいさい)<-
verb(飛ぶ)<-
verb(走る)<-
verb(植える)<-
verb(見る)<-
verb(放す)<-
verb(降る)<-
comma(, )<-
period(. )<-
```

We made an experiment on Sun SPARC station 10 by giving these two EFS's and following four goals with Japanese sentences to the prover, and obtained

the results in the table.

1. ←sentence(大空を飛ぶ)
   /* ←sentence(Fly in the sky.) */
2. ←sentence(彼女は，鳥を放す．)
   /* ←sentence(She releases a bird.) */
3. ←sentence(ちいさい犬が，海岸を走る．)
   /* ←sentence(A little dog runs on the beach.)*/
4. ←sentence(彼女が植えた赤い花が，
   庭の花壇できれいにさいている．)
   /* ←sentence(Flowers she planted are in full
   bloom in the flower bed in the garden.) */

Sentence	Regular EFS	Var-separated EFS
1	0.17 sec	0.60 sec
2	0.44 sec	1.04 sec
3	2.42 sec	1.62 sec
4	318.89 sec	1.90 sec

The sentences No.2 and No.3 have been accepted by the prover, while the rest have been rejected. From this table we can see that the variable-separated EFS S' makes the prover much faster than the original regular EFS.

6. Concluding Remarks

In this paper, we have introduced a notion of variable-separated EFS's. Then we have proved that these two classes of EFS's are equivalent to each other in expressive power, and shown that the variable-separated EFS's make the prover for regular EFS's at least twice as fast as the original regular EFS's.

We have also given a procedure to transform regular EFS's into equivalent variable-separated EFS's in the process of proving the theorem. However, it is not always efficient. In fact, there has been observed that even a small regular EFS with just five axioms may be transformed into a variable-separated EFS with more than a thousand axioms. To make the procedure efficient is one of the future problems we should solve.

We conclude this paper by noticing that our notions and methods can be directly applied to context-free grammars, and hence the prover also works as a top-down parser for context-free grammars.

Acknowledgments

We would like to thank E. Hirowatari and S. Matsunaga for their valuable comments.

References

1) J.E.Hopcroft and J.D.Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Publishing Company, 1979.
2) J.W.Lloyd. *Foundations of Logic Programming(second edition).* Springer-Verlag, 1987.
3) R.M.Smullyan. *Theory of Formal Systems.* Princeton Univ. Press, Princeton, 1961.
4) S.Arikawa, S.Miyano, A.Shinohara, T.Shinohara, and A.Yamamoto. Algorithmic learning theory with elementary formal systems. *IEICE Transactions on Information and Systems*, Vol. E75-D, No. 4, pp. 405–414, 1992.
5) S.Arikawa, T.Shinohara, and A.Yamamoto. Learning elementary formal systems. *Theoretical Computer Science*, Vol. 95, pp. 97–113, 1992.