Elementary Formal System as a Logic Programming Language

Yamamoto, Akihiro
Department of Information Systems Kyushu University

RIFIS-TR-CS-12

RIFIS Technical Report

Elementary Formal System as a Logic Programming Language

Akihiro Yamamoto

March 5, 1989

Research Institute of Fundamental Information Science
Kyushu University 33
Fukuoka 812, Japan
E-mail: arikawa@rifis.sci.kyushu-u.ac.jp Phone: 092(641)1101 Ext.4470

Elementary Formal System as a Logic Programming Language

Akihiro YAMAMOTO
Department of Information Systems
Kyushu University 39, Kasuga 816, JAPAN

Mailing address : Research Institute of Fundamental Information Science
Kyushu University 33, Fukuoka 812, JAPAN

Abstract

In this paper, we give a theoretical foundation of EFS (elementary formal system) as a logic programming language. We show that the set of all the unifiers of two atoms is finite and computable by restricting the form of axioms and goals without losing generality. The restriction makes the negation as failure rule complete. We give two conditions of EFS's such that the negation as failure rule is identical to the closed world assumption. We also give a subclass of EFS's where a procedure of CWA is given as bounding the length of derivations We compare these classes with the Chomsky hierarchy.

1. Introduction

In this paper we give a theoretical foundation of EFS (elementary formal system) as a logic programming language.

EFS was first introduced by Smullyan [13] to develop his recursive function theory. Both EFS and logic programming use definite clauses as axioms, but the structures of their terms are different. Logic programming uses the first order terms, but the terms of EFS are patterns in $(\Sigma \cup X)^+$ as terms. Arikawa [1] showed EFS is suitable to generate languages.

We formalize a derivation procedure for EFS, and give the same semantics to EFS as that of logic programming. Our motivation for this is to give a unifying framework of inductive inference of languages by combining MIS [12] with EFS. The framework is given as MIEFS [2]. Thus we need a complete refutation procedure to accept languages defined by EFS's, nearly in the same way as that of logic programming.

Our theory is based on that of *logic programming schema* given by Jaffar et al. [7]. In order to show the completeness of refutation for EFS, we give the declarative semantics of EFS's by introducing the associative law as an equational theory to represent the unification of patterns. It would seem that our theory is an instance of the schema such that the Herbrand universe is Σ^+. However, there still remains a big problem in the operational semantics, that is, there are infinitely many maximally general unifiers of two atoms. No method for avoiding this problem was given in the schema, and so the NF(negation as failure rule) is incomplete in general.

We give a solution of the problem by simply assuming that EFS's are variable-bounded and goals are ground. The assumption does not lose generality, because it is known that every recursively enumerable language is definable by a variable-bounded EFS [2]. We show that the set of all unifiers of two atoms is finite and computable when we use variable-bounded EFS's and ground goals. Thus NF is complete under the assumption.

Moreover, we give two procedures of the CWA (closed world assumption) for variable-bounded EFS's. One is to make use of NF by giving some restrictions to the axioms. We show that NF is identical to CWA for hierarchical EFS's and reducing EFS's. The other is to bound the length of derivations with the size of atoms in the goal. We show that this procedure realizes CWA for weakly reducing EFS's. Since EFS combines logic programming and formal language theory, we can compare these classes with Chomsky hierarchy. We show that every context-free language is definable by a reducing EFS. It is shown in [2] that every context-sensitive language is definable by a weakly reducing EFS.

The paper is organized as follows. In Section 2 we give the fundamental definitions of EFS and introduce the variable-bounded EFS. In Section 3, we give the derivation procedure for EFS and show that all unifiers can be effectively computed for variable-bounded EFS's and ground goals. In Section 4 we give the same semantics to EFS as that of logic programming. In Section 5 we prove the completeness of NF for variable-bounded EFS. In Section 6 we discuss CWA mainly for weakly reducing EFS.

2. Elementary Formal System

We start with recalling the definitions of EFS.

Let Σ, X, and Π be mutually disjoint sets. We assume that Σ and Π are finite. We refer to Σ as the *alphabet*, to each element of it as a *symbol*, which will be denoted by $a, b, c, \ldots$, to each element of X as a *variable*, denoted by $x, y, z, x_1, x_2, \ldots$, and to each element of Π as a *predicate symbol*, denoted by $p, q, \ldots$, where each of them has an *arity*.

Definition. A *word* over a set A is a finite sequence of elements of A. A^+ denotes the set of all non-empty words over the set A, and $A^* = A^+ \cup \{\lambda\}$ where λ is the empty word.

Definition. A *term* of S 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) of S is an expression of the form $p(\tau_1, \ldots, \tau_n)$, where p is a predicate symbol in Π with arity n and $\tau_1, \ldots, \tau_n$ are terms of S. The atom is *ground* if $\tau_1, \ldots, \tau_n$ are all ground.

Notation.

(1) For a term π, $|\pi|$ denotes the length of π, that is, the number of all occurrences of symbols and variables in π. For an atom $p(\pi_1, \ldots, \pi_n)$, let

$$|p(\pi_1, \ldots, \pi_n)| = |\pi_1| + \cdots + |\pi_n|.$$

(2) For a term π and variable x, $o(x, \pi)$ is the number of all occurrences of x in π. For an atom $p(\pi_1, \ldots, \pi_n)$, let

$$o(x, p(\pi_1, \ldots, \pi_n)) = o(x, \pi_1) + \cdots + o(x, \pi_n).$$

(3) $v(\alpha)$ denotes the set of all variables in a term or an atom α.

Example 1. Let $A = p(ax, bycx)$. Then $|A| = 6$, $o(x, A) = 2$, and $v(A) = \{x, y\}$.

A *well-formed formula* is defined in the same way as in first-order predicate logic.

Definition. A *clause* is a well-formed formula of the form

$$\forall(A_1 \vee \ldots \vee A_n \vee (\neg B_1) \vee \ldots \vee (\neg B_m)),$$

where $A_1, \ldots, A_n, B_1, \ldots, B_m$ are atoms of S and $n, m \geq 0$. The formula with $n = m = 0$ is assumed to denote $\Box$, and called *the empty clause*. We denote the above clause by

$$A_1, \ldots, A_n \leftarrow B_1, \ldots, B_m.$$

The clause is *ground* if the atoms $A_1, \ldots, A_n$, $B_1, \ldots, B_m$ are all ground.

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

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

Definition (Smullyan [13]) . An *elementary formal system* (*EFS* for short) S is a triplet (Σ, Π, Γ), where Γ is a finite set of definite clauses. The definite clauses in Γ are called *axioms* of S.

Now we explain the provability of atoms in the theory of EFS. Arikawa [1] gave his theory of formal languages using this provability.

Definition. A *substitution* θ is a (semi-group) homomorphism from $(\Sigma \cup X)^+$ to itself such that $a\theta = a$ for every $a \in \Sigma$ and the set $\{x \in X | x\theta \neq x\}$, denoted by $D(\theta)$, is finite. The substitution is *ground* if $x\theta$ is ground for every $x \in D(\theta)$.

Let θ be a substitution. If $D(\theta) = \{x_1, \ldots, x_n\}$ and $x_i\theta = \tau_i$, then θ is denoted by

$$\{x_1 := \tau_1, \ldots, x_n := \tau_n\}.$$

We also define

$$p(\tau_1, \ldots, \tau_n)\theta = p(\tau_1\theta, \ldots, \tau_n\theta)$$

and

$$(A \leftarrow B_1, \ldots, B_m)\theta = A\theta \leftarrow B_1\theta, \ldots, B_m\theta.$$

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

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:

(2.1) If $\Gamma \ni C$, then $\Gamma \vdash C$.

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

(2.3) If $\Gamma \vdash A \leftarrow B_1, \ldots, B_n$ and $\Gamma \vdash B_n \leftarrow$, then $\Gamma \vdash A \leftarrow B_1, \ldots, B_{n-1}$.

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

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

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

We introduce an important subclass of EFS's.

Definition. A definite clause $A \leftarrow B_1, \dots, B_n$ is *variable-bounded* if $v(A) \supset v(B_i)$ $(i = 1, ..., n)$. An EFS is *variable-bounded* if its axioms are all variable-bounded.

Example 2. An EFS $S = (\{a,b,c\}, \{p,q\}, \Gamma)$ with

$$\Gamma = \left\{ \begin{array}{l} p(a,b,c) \leftarrow, \\ p(ax,by,cz) \leftarrow p(x,y,z), \\ q(xyz) \leftarrow p(x,y,z) \end{array} \right\}$$

is variable-bounded and defines a language

$$L(S,q) = \{a^n b^n c^n \mid n \geq 1\}.$$

3. Derivation Procedure

First we give a derivation procedure for an EFS with unification defined as follows:

Definition. Let α and β be a pair of terms or atoms. Then a substitution θ is a *unifier* of α and β, or θ *unifies* α and β if $\alpha\theta = \beta\theta$. α and β are *unifiable* if there exists a unifier of α and β. $U(\alpha, \beta)$ denotes the set of all unifiers θ of α and β such that $D(\theta) \subset v(\alpha) \cup v(\beta)$.

It is often the case that there are infinitely many maximally general unifiers.

Example 3 (Plotkin[10]). Let $S = (\{a,b\}, \{p\}, \Gamma)$. Then $\{x := a^i\}$ for every i is the unifier of $p(ax)$ and $p(xa)$. All the unifiers are maximally general.

We overcome the problem with the following proposition.

Lemma 1. *Let α and β be a pair of terms or atoms. If one of them is ground, then every unifier of α and β is ground and $U(\alpha, \beta)$ is finite and computable.*

Proof. First we show that the result holds in case α and β are terms. Assume that α is ground, $v(\beta) = \{x_1, \dots, x_n\}$ and $\theta = \{x_1 := \pi_1, \dots, x_k := \pi_k\}$ unifies α and β where $k \leq n$. Then $k = n$ and $\pi_1, \dots, \pi_k$ are all ground; that is, θ is ground by the definitions of ground terms and substitutions. By comparing $|\alpha|$ with $|\beta\theta|$, it holds that $|\pi_i| \leq |\alpha|$. Since the number of ground terms π with $|\pi| \leq |\alpha|$ is finite, and the set

$$T(\alpha, \beta) = \{\sigma \mid D(\sigma) = v(\beta) \text{ and } |x\sigma| \leq |\alpha| \text{ for every } x \in D(\sigma)\}$$

is finite and computable, we can compute $U(\alpha, \beta)$ by testing every element σ of $T(\alpha, \beta)$ to see whether $\alpha = \beta\sigma$ or not.

Now let α be a ground atom and β be an atom. If the predicate symbols of α and β are different, it is clear that α and β are not unifiable. Thus we assume $\alpha = p(\tau_1, \dots, \tau_n)$ and $\beta = p(\pi_1, \dots, \pi_n)$. A unifier of α and β is also a unier of τ_i and π_i for $i = 1, \dots, n$. Then $U(\alpha, \beta)$ can be computed by testing every tuple of $(\sigma_1, \dots, \sigma_n)$ where $\sigma_i \in U(\tau_i, \pi_i)$ $(i = 1, \dots, n)$ to see whether $\alpha = \beta\sigma_1 \cdots \sigma_n$ or not.

Next, we formalize the derivation for an EFS with no requirement that every unifier should be most general.

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

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

Definition. Let C and D be two clauses. Then C is a *variant* of D if $C = D\theta$ and $C\theta' = D$ for some substitutions θ and θ'. Similarly we define variants of a clause.

Definition. A *computation rule* is a rule which selects an atom from every goal clause.

Definition. Let S be an EFS, G be a goal of S, and R be a computation rule. A *derivation from* G is a (finite or infinite) sequence of triplets (G_i, θ_i, C_i) $(i = 0, 1, \ldots)$ which satisfies the following conditions:

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

(3.2) $v(C_i) \cap v(C_j) = \phi \quad (i \neq j)$, and $v(C_i) \cap v(G) = \phi$ for every i.

(3.3) If $G_i = \leftarrow A_1, \ldots, A_k$ and A_m is the atom selected by R, then $C_i = A \leftarrow B_1, \ldots, B_q$, θ_i is a unifier of A and A_m, and

$$G_{i+1} = (\leftarrow A_1, \ldots, A_{m-1}, B_1, \ldots, B_q, A_{m+1}, \ldots, A_k)\theta_i.$$

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. For a finite derivation (G_i, θ_i, C_i) $(i = 0, \ldots n)$, we define its *length* as n.

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

Example 4. Let EFS $S = (\{a, b\}, \{p\}, \Gamma)$ with

$$\Gamma = \left\{ \begin{array}{l} p(a) \leftarrow, \\ p(bxy) \leftarrow p(x), p(y) \end{array} \right\}.$$

Then a refutation from $\leftarrow p(babaa)$ is illustrated in Figure 1, where the computation rule selects the leftmost atom from every goal.

The aim of our formalization of derivation is to give a procedure accepting languages definable by EFS's. Thus we assume every derivation starts from a ground goal. Then we get the following lemma by Lemma 1 and the definition of variable-bounded clauses.

Lemma 2. *Let S be a variable-bounded EFS, and G be a ground goal. Then every resolvent of G is ground, and the set of all the resolvents of G is finite and computable.*

The power of variable-bounded EFS's is shown by using the derivation procedure in [2].

Theorem 1 ([2]). *Let Σ be an alphabet with at least two symbols. Then a language $L \subset \Sigma^+$ is definable by a variable-bounded EFS if and only if L is recursively enumerable.*

Thus the variable-bounded EFS's are powerful enough, and we make the following assumption.

Assumption. Every EFS is variable-bounded and every derivation starts from a ground goal.

Moreover, by Lemma 2, we can implement the derivation in nearly the same way as for the traditional logic programming languages under the assumption. If we don't have the assumption, we need another formalization, such as given by Yamamoto[14], in order to control the nondeterministic algorithm of unification.

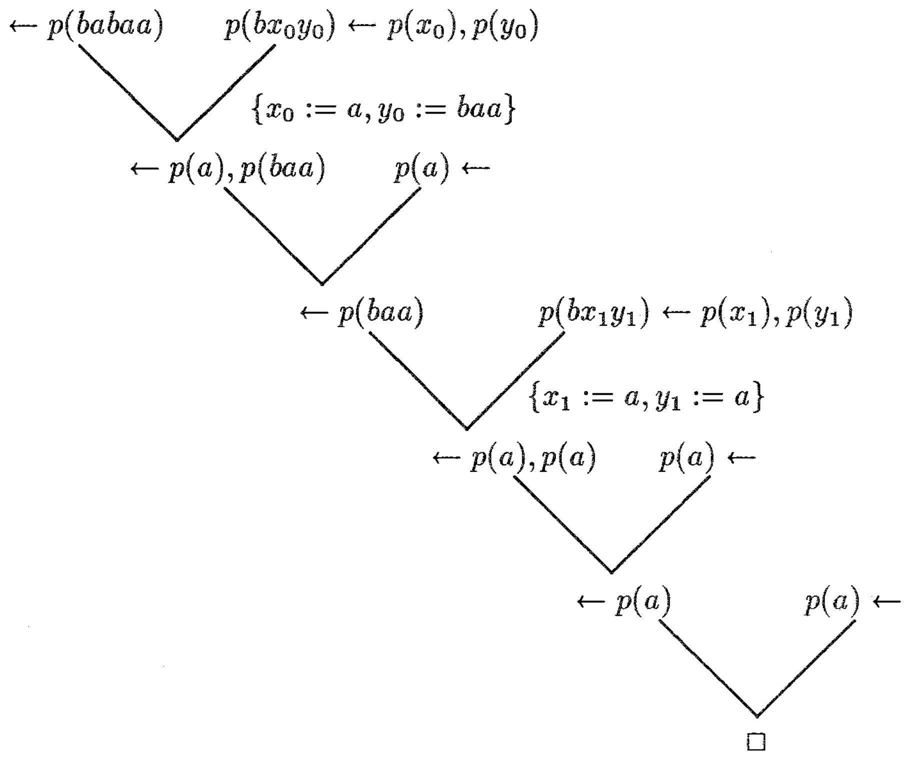

Figure 1: A refutation

4. Completeness of Refutation

We describe the semantics of EFS's according to Jaffar, et al.[7]. They have given a general framework of various logic programming languages by representing their unification algorithm as an equality theory. To represent unification in derivations for EFS's we use the equality theory

$$E = \{cons(cons(x,y),z) = cons(x, cons(y,z))\},$$

where *cons* is to be interpreted as the catenation of terms.

The first semantics for an EFS $S = (\Sigma, \Pi, \Gamma)$ is its model. To interpret well-formed formulas of S we can restrict the domains to the models of E. Then a model of S is an interpretation which makes every axiom in Γ true. We use Σ^+ as the *Herbrand universe* and the set

$$B(S) = \{p(\tau_1, \ldots, \tau_n) \mid p \in \Pi \text{ and } \tau_1, \ldots, \tau_n \in \Sigma^+ \}$$

as the *Herbrand base*. A subset I of $B(S)$ is called an *Herbrand interpretation* in the sense that $A \in I$ means A is true and $A \notin I$ means A is false for $A \in B(S)$. Then

$$M(S) = \cap\{M | M \text{ is an Herbrand model of } S\ \}$$

is an Herbrand model of S, and every ground atom in $M(S)$ is true in any model of S.

The second semantics is the least fixpoint $lfp(T_S)$ of the function $T_S : 2^{B(S)} \longrightarrow 2^{B(S)}$ defined by

$$T_S(I) = \left\{ A \in B(S) \,\middle|\, \begin{array}{l} \text{there is a ground instance} \\ \qquad A \leftarrow B_1, \ldots, B_n \\ \text{of an axiom in } \Gamma \text{ such that } B_k \in I \text{ for } k = 1, \ldots, n. \end{array} \right\}.$$

T_S is shown to be continuous in [7]. We use the following sets defined by T_S:

$$\begin{array}{lll} T_S \uparrow 0 & = \phi, & \\ T_S \uparrow \alpha & = T_S(T_S \uparrow (\alpha - 1)), & \text{if } \alpha \text{ is a successor ordinal,} \\ T_S \uparrow \alpha & = \bigcup\{T_S \uparrow \beta \mid \beta < \alpha\}, & \text{if } \alpha \text{ is a limit ordinal,} \end{array}$$

$$\begin{array}{lll} T_S \downarrow 0 & = B(S), & \\ T_S \downarrow \alpha & = T_S(T_S \downarrow (\alpha - 1)), & \text{if } \alpha \text{ is a successor ordinal,} \\ T_S \downarrow \alpha & = \bigcap\{T_S \downarrow \beta \mid \beta < \alpha\}, & \text{if } \alpha \text{ is a limit ordinal.} \end{array}$$

$lfp(T_S)$ is characterized by the fact that

$$lfp(T_S) = T_S \uparrow \omega.$$

The third semantics, using refutation, is defined by

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

These three semantics were shown to be identical:

Theorem 2 (Jaffar et al.[7]). *For any EFS S,*

$$M(S) = lfp(T_S) = SS(S).$$

Now we give another semantics of EFS using the provability as the set

$$PS(S) = \{A \in B(S) \mid \Gamma \vdash A\}.$$

Theorem 3. *For any EFS S,*

$$PS(S) = SS(S).$$

This theorem is important from the viewpoint of language theory because the refutation is complete as *accepting* EFS languages. The proof of the theorem is clear from the definition.

5. Negation as Failure Rule

Now we discuss the inference of negation.

Definition. A derivation is *finitely failed with length n* if its length is n and there is no axiom which satisfies the condition (3.3) for the selected atom of the last goal.

Example 5. Let S be the EFS in Example 4. Then the derivation illustrated in Figure 2 is finitely failed with length 2.

Definition. A derivation (G_i, θ_i, C_i) $(i = 0, 1, \ldots)$ is *fair* if it is finitely failed or, for each atom A in G_i, there is a $k \geq i$ such that $A\theta_i \cdots \theta_{k-1}$ is the selected atom of G_k. A computation rule is *fair* if it makes all derivations *fair*.

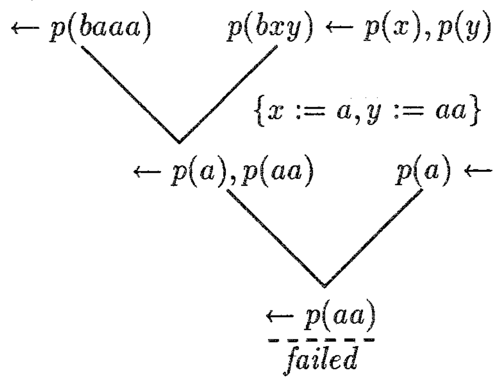

Figure 2: A derivation finitely failed with length 2

The *negation as failure rule* (NF for short) is the rule that infers $\neg A$ when a ground atom A is in the set

$$FF(S) = \left\{ A \in B(S) \;\middle|\; \begin{array}{l} \text{for any fair computation rule, there is an } n \text{ such that} \\ \text{all derivations from } \leftarrow A \text{ are finitely failed within length } n \end{array} \right\}.$$

$FF(S)$ is characterized by the fact that

$$FF(S) = T_S \downarrow \omega.$$

Note that $FF(S)$ is not always identical to the set

$$GF(S) = \left\{ A \in B(S) \;\middle|\; \begin{array}{l} \text{for any fair computation rule, all derivations} \\ \text{from } \leftarrow A \text{ are finitely failed} \end{array} \right\}.$$

The following example was pointed out in [7].

Example 6. Let an EFS $S = (\{a, b\}, \{p, q, r\}, \Gamma)$ with

$$\Gamma = \left\{ \begin{array}{l} p(a) \leftarrow q(ax, xa), \\ q(x, x) \leftarrow r(x), \\ r(ax) \leftarrow r(x) \end{array} \right\}.$$

Then $p(a) \in GF(S)$ but $p(a) \notin FF(S)$ because there are infinitely many unifiers of ax and xa, as was shown in Example 3.

We put $ecj(\theta) = (x_1 = \tau_1 \wedge \ldots \wedge x_n = \tau_n)$ for a substitution $\theta = \{x_1 := \tau_1, \ldots, x_n := \tau_n\}$, and $ecj(\theta) = true$ for an empty θ. From the discussions in Jaffar, et al. [7], NF for EFS is complete if the following two conditions are satisfied:

(5.1) There is a theory E^* that logically implies

$$(\pi = \tau) \rightarrow \vee_{i=1}^{k} ecj(\theta_i)$$

for every two terms π and τ, where $\theta_1, \ldots, \theta_k$ are all the unifiers of π and τ.

(5.2) $FF(S) = GF(S)$.

In general, NF for EFS is not complete, by Example 6, and there is no E^* of (5.1) for an EFS, because there are infinitely many maximal unifiers.

Now we prove that NF for variable-bounded EFS's is complete. We prove that $FF(S)$ is also identical to the set

$$GGF(S) = \left\{ A \in B(S) \;\middle|\; \begin{array}{l} \text{for any fair computation rule, all derivations} \\ \text{from } \leftarrow A \text{ such that all goals in them} \\ \text{are ground are finitely failed} \end{array} \right\}.$$

The inference rule that infers $\neg A$ for a ground atom A if A is in $GGF(S)$ is called the *Herbrand rule*[8].

Theorem 4. *For any variable-bounded EFS S,*

$$FF(S) = GF(S) = GGF(S).$$

Proof. Since S is variable-bounded, $GF(S) = GGF(S)$ by Lemma 2. It is also shown that $FF(S) = GF(S)$ by Lemma 2 and König's Lemma.

By Lemma 2 we use the following equational theory for a variable-bounded EFS instead of (5.1):

$$E^* = \left\{ \tau = \pi \rightarrow \vee_{i=1}^{k} ecj(\theta_i) \;\middle|\; \begin{array}{l} \tau \text{ is a term, } \pi \text{ is a ground term,} \\ \text{and } \theta_1, \ldots, \theta_k \text{ are all the unifiers of } \pi \text{ and } \tau \end{array} \right\}.$$

This E^* exists by Lemma 1. Thus NF is complete and identical to the Herbrand rule for variable-bounded EFS's.

6. Closed World Assumption

In this section we discuss the *closed world assumption* (CWA for short) for EFS. For an EFS $S = (\Sigma, \Pi, \Gamma)$ and $A \in B(S)$, CWA infers $\neg A$ if A is not a logical consequence of Γ[11]. When we use the refutation procedure to show that A is a logical consequence of Γ, CWA infers $\neg A$ if $A \notin SS(S)$. When we treat the refutation as a procedure to accept languages, CWA is very natural because $w \notin L(S, p)$ if $p(w) \notin SS(S)$.

However, since the complement of a recursively enumerable set is not always recursively enumerable, there is no general procedure of CWA by Theorem 1. Moreover, $w \notin L(S, p)$ does not always imply $p(w) \in FF(S)$ even if $L(S, p)$ is recursive, as shown in the following example.

Example 7. Let $S = (\{a, b\}, \{p\}, \Gamma)$ with

$$\Gamma = \left\{ \begin{array}{l} p(a) \leftarrow, \\ p(b) \leftarrow p(b) \end{array} \right\}.$$

Then $SS(S) = \{p(a)\}$, $L(S, p) = \{a\}$ and $b \notin L(S, p)$, but $FF(S) = GGF(S) = \phi$.

In the following, we give two procedures of CWA and introduce some subclasses of EFS's.

6.1. Termination property of EFS

First we give some conditions for variable-bounded EFS's so that CWA is equivalent to NF, that is, $SS(S) = B(S) - FF(S)$. By Theorem 4 and the definition of $GF(S)$ it suffices to show that there are no infinite derivations from $\leftarrow A$ for every $A \in B(S)$. Noting that every derivation from $\leftarrow A$ is ground, we treat only ground goals. From now on we identify every ground goal $\leftarrow A_1, \ldots A_k$ with a sequence of ground atoms $A_1, \ldots A_k$. Then we consider the partial order $\succ$ of $B(S)^*$.

Definition. Let D be a set with a partial order $\succ_D$. Then $\succ_D$ is *well-founded* if there is no infinite sequence $d_1, d_2, d_3 \ldots$ in D such that

$$d_1 \succ_D d_2 \succ_D d_3 \succ_D \cdots.$$

Proposition 1. *Let $S = (\Sigma, \Pi, \Gamma)$ be a variable bounded EFS. Then there is no infinite derivation from $\leftarrow A$ for every $A \in B(S)$ if there is a well-founded partial order $\succ$ of $B(S)^*$ which satisfies the following two conditions:*

(6.1) For every two sequences $A_1, \ldots, A_k$ and $B_1, \ldots, B_q$ such that $q \geq 0, k > 0$,

$$\begin{array}{ll} A_m \succ B_1, \ldots, B_q & \\ \Rightarrow \quad A_1, \ldots, A_k \succ A_1, \ldots, A_{m-1}, B_1, \ldots, B_q, A_{m+1}, \ldots, A_k. \end{array}$$

(6.2) For every ground instance$A \leftarrow B_1, \ldots, B_q$ of an axiom in Γ,

$$A \succ B_1, \ldots, B_q.$$

Proof. Let G_1 and G_2 be two goals. Suppose G_2 is a resolvent of G_1 and the selected atom is A_m. Then there is a ground instance $A_m \leftarrow B_1, \ldots, B_q$ of an axiom. By the condition (6.2), $A_m \succ B_1, \ldots, B_q$ and thus $G_1 \succ G_2$ by the condition (6.1).

Now we show two examples of the order $\succ$. In the first example we use the method introduced into traditional logic programming by Lloyd [8]. We write $pred(p(\pi_1, \ldots, \pi_n)) = p$ for every atom $p(\pi_1, \ldots, \pi_n)$.

Definition. An EFS $S = (\Sigma, \Pi, \Gamma)$ is *hierarchical* if there is a mapping φ from Π to a set of natural numbers such that $\varphi(pred(A)) > \varphi(pred(B_i))$ for every axiom $A \leftarrow B_1, \ldots, B_n$ and $i = 1, \ldots, n$.

Proposition 2. *If a variable-bounded EFS S is hierarchical, then*

$$B(S) - FF(S) = SS(S).$$

Proof. By Proposition 1, it is sufficient to show that there is a well-founded partial order $\succ$ of $B(S)^*$. Let φ be a mapping given in the definition of hierarchical EFS. Then we define $\succ$ as follows:

$A_1, \ldots, A_n \succ B_1, \ldots, B_k$
$\Leftrightarrow$
there is an atom A_m and a sequence $C_1, \ldots, C_q$ of atoms such that

$$B_1, \ldots, B_k = A_1, \ldots, A_{m-1}, C_1, \ldots, C_q, A_{m+1}, \ldots, A_n$$

and

$$\varphi(pred(A_m)) > \varphi(pred(C_i))$$

for every $i = 1, \ldots, q$.

It is easily shown that this order satisfies the conditions (6.1) and (6.2). Moreover the order is well-founded because it is subsumed by the multiset-ordering, which was shown to be well-founded by Dershowitz and Manna [4].

Example 8. An EFS $S = (\{a, b\}, \{p, q\}, \Gamma)$ with

$$\Gamma = \left\{ \begin{array}{l} p(x) \leftarrow q(abx, xab) \\ q(y, y) \leftarrow \end{array} \right\}$$

is hierarchical and it defines a language $L(S, p) = \{(ab)^n, | \, n \geq 1\}$.

We define an important class of EFS to give the second example of $\succ$ making use of the length of atoms.

Definition. A clause $A \leftarrow B_1, \ldots, B_n$ is *reducing* if

$$|A\theta| > |B_i\theta|$$

for any substitution θ and $i = 1, \ldots, n$. An EFS $S = (\Sigma, \Pi, \Gamma)$ is *reducing* if axioms in Γ are all reducing.

The following lemma, which is a modification of a lemma in [2], is useful to characterize the concepts we are introducing.

Lemma 3. *Let $A, B_1, \ldots, B_n$ be atoms. Then*

$$|A\theta| > |B_1\theta| + \cdots + |B_n\theta| \qquad (|A\theta| \geq |B_1\theta| + \cdots + |B_n\theta|)$$

for any substitution θ if and only if

$$|A| > |B_1| + \cdots + |B_n| \qquad (|A| \geq |B_1| + \cdots + |B_n|)$$

and

$$o(x, A) \geq o(x, B_1) + \cdots + o(x, B_n)$$

for any variable x.

For example the concept of reducing is characterized as follows:

Proposition 3. *A clause $A \leftarrow B_1, \ldots, B_n$ is reducing if and only if*

$$\begin{array}{rl} |A| & > |B_i|, \\ o(x, A) & \geq o(x, B_i) \end{array}$$

for any variable x and $i = 1, \ldots, n$.

The EFS in Example 4 is reducing.

Theorem 5. *For every reducing EFS S,*

$$B(S) - FF(S) = SS(S).$$

Proof. The order $\succ$ defined in the same way as Proposition 2 satisfies the conditions (6.1) and (6.2).

Theorem 6. *Every context-free language $L \subseteq \Sigma^+$ is definable by a reducing EFS.*

Proof. Let G be a context-free grammar in Chomsky's normal form representing L. Then we can construct a reducing EFS S by using every non-terminal symbol of G as a predicate symbol of S, and by transforming the rule in G of the form $A \to B, C$ into a clause

$$A(xy) \leftarrow B(x), C(y)$$

and the rule of the form $A \to a$ into

$$A(a) \leftarrow .$$

The following example shows that the converse of Theorem 6 does not hold.

Example 9. An EFS $S = (\{a, b, c\}, \{p, q\}, \Gamma)$ with

$$\Gamma = \left\{ \begin{array}{l} p(a, bb, cc) \leftarrow, \\ p(ax, by, cz) \leftarrow p(x, y, z), \\ q(abc) \leftarrow, \\ q(axyz) \leftarrow p(x, y, z) \end{array} \right\}$$

is reducing and defines a language $L(S, q) = \{a^n b^n c^n \mid n \geq 1\}$, which is not context-free.

6.2. Bounding the length of derivations

We give a class of EFS's where we give another procedure of CWA. Roughly speaking, the procedure we are introducing is to bound the length of derivations even if $SS(S) \neq B(S) - FF(S)$.

Definition. A clause $A \leftarrow B_1, \ldots, B_n$ is *weakly reducing* if

$$|A\theta| \geq |B_i\theta|$$

for any substitution θ and $i = 1, \ldots, n$. An EFS $S = (\Sigma, \Pi, \Gamma)$ is *weakly reducing* if axioms in Γ are all weakly reducing.

The concept of weakly reducing is also characterized by Lemma 3.

Proposition 4. *A clause $A \leftarrow B_1, \ldots, B_n$ is weakly reducing if and only if*

$$\begin{aligned} |A| &\geq |B_i|, \\ o(x, A) &\geq o(x, B_i) \end{aligned}$$

for any variable x and $i = 1, \ldots, n$.

For every subset U of $B(S)$, we put $U|_n = \{A \in U \mid |A| \leq n\}$. Note that $U|_n$ is a finite set for every U.

The following is the main theorem to get the procedure.

Theorem 7. *Let $S = (\Sigma, \Pi, \Gamma)$ be a weakly reducing EFS. Then*

$$A \in T_S \uparrow \omega \quad \Leftrightarrow \quad A \in T_S \uparrow \sharp(B(S)|_{|A|}).$$

Proof. It suffices to prove the $\Rightarrow$ part. First we show

$$(6.3) \qquad T_S((T_S \uparrow k)|_n)|_n = (T_S \uparrow k+1)|_n \qquad (k \geq 0).$$

The $\subset$ part is proved directly from the definition. Thus we prove the $\supset$ part. Let $B \in (T_S \uparrow k+1)|_n$. Then there is a ground instance $B \leftarrow B_1, \ldots, B_q$ of an axiom. Since S is weakly reducing,

$$\{B_1, \ldots, B_q\} \subset (T_S \uparrow k)|_n,$$

and thus

$$B \in T_S((T_S \uparrow k)|_n)|_n.$$

By (6.3) and the monotonicity of T_S,

$$(6.4) \qquad (T_S \uparrow k)|_n \subset (T_S \uparrow k+1)|_n \qquad (k \geq 0).$$

Moreover, if $(T_S \uparrow K)|_n = (T_S \uparrow K+1)|_n$ for some K then

$$(T_S \uparrow K)|_n = (T_S \uparrow k)|_n \qquad (k \geq K).$$

Since $(T_S \uparrow k)|_n \subset B(S)|_n$ and $B(S)|_n$ is finite,

$$(6.5) \qquad (T_S \uparrow k)|_n = (T_S \uparrow \sharp(B(S)|_n))|_n \qquad (k \geq \sharp(B(S)|_n)).$$

From (6.4) and (6.5) we get

$$(6.6) \qquad (T_S \uparrow k)|_n \subset (T_S \uparrow \sharp(B(S)|_n))|_n \qquad (k \geq 0)$$

Now let $A \in T_S \uparrow \omega$. Then $A \in T_S \uparrow k$ for some k. By (6.6) we get

$$A \in (T_S \uparrow k)|_{|A|} \subset (T_S \uparrow \sharp(B(S)|_{|A|}))|_{|A|} \subset T_S \uparrow \sharp(B(S)|_{|A|}).$$

We define $lob(A \leftarrow B_1, \ldots, B_m) = m$ for a definite clause $A \leftarrow B_1, \ldots, B_m$. *lob* means the length of the body. The following lemma gives the upper-bound of the length of the shortest refutation from $\leftarrow A$ such that $A \in T_S \uparrow n$.

Lemma 4. *Let $S = (\Sigma, \Pi, \Gamma)$ be an EFS, and $m = \max_{C \in \Gamma}(lob(C))$. If $A \in T_S \uparrow n$ then there is a refutation with length less than or equal to $f(m)$ where*

$$f(m,n) = \begin{cases} 1 & \textit{if } m = 0 \\ \sum_{i=0}^{n-1} m^i & \textit{otherwise.} \end{cases}$$

Proof. The result is clear in case $m = 0$. In case $m > 0$, we prove the result by induction on n .

First suppose $n = 1$. Then there exists an axiom of the form $A' \leftarrow$ in Γ. Thus the results holds because there is a refutation from $\leftarrow A$ with length 1.

Now we suppose the result holds for n and $A \in T_S \uparrow (n+1)$. Then there is a ground instance $A \leftarrow B_1, \ldots, B_k$ $(k \leq m)$ of the axiom where $\{B_1, \ldots, B_k\} \subset T_S \uparrow n$. By induction hypothesis there is a refutation from each B_i with length less than or equal to $\sum_{i=0}^{n-1} m^i$. Then a refutation can be constructed by combining these refutations, and its length is less than or equal to $1 + m \sum_{i=0}^{n-1} m^i = \sum_{i=0}^{n} m^i$.

By combining Theorem 7 and Lemma 4 we get the following theorem.

Theorem 8. *Let $S = (\Sigma, \Pi, \Gamma)$ be a weakly reducing EFS, and $m = \max_{C \in \Gamma}(lob(C))$. If $A \in SS(S)$, then there is a refutation from $\leftarrow A$ with length less than or equal to $f(m, \sharp(B(S)|_{|A|}))$.*

Now we get a procedure of CWA. For a weakly reducing EFS S, we can conclude $A \notin SS(S)$ if there is no refutation from $\leftarrow A$ with length less than or equal to $f(m, \sharp(B(S)|_{|A|}))$. Arimura [3] pointed out the same procedure for traditional logic programming by observing derivation trees.

Example 10. Let S be the EFS in Example 7. There is no refutation from $\leftarrow p(b)$ with length less than or equal to $f(1,2) = \sum_{i=0}^{1} 1^i = 1 + 1 = 2$. Thus we can decide $p(b) \notin SS(S)$.

Now we compare the class of weakly reducing EFS's and Chomsky hierarchy. We use the following class of EFS's.

Definition. A variable-bounded EFS $S = (\Sigma, \Pi, \Gamma)$ is *length-bounded* if

$$|A\theta| \geq |B_1\theta| + \cdots + |B_n\theta|$$

for every axiom $A \leftarrow B_1, \ldots, B_n \quad (n \geq 1)$ in Γ.

Clearly any length-bounded EFS is weakly reducing. The concept of length-boundness is also characterized by Lemma 3 [2].

Every EFS S in Examples 2, 4, and 7 is length-bounded. The relation between EFS and CSG is shown as follows in [2].

Theorem 9 ([2]).

(1) *Any length-bounded EFS language is context-sensitive.*

(2) *For every context-sensitive language $L \subseteq \Sigma^+$, there exist a superset Σ_0 of Σ, a length-bounded EFS $S = (\Sigma_0, \Pi, \Gamma)$ and $p \in \Pi$ such that $L = L(S,p) \cap \Sigma^+$.*

7. Concluding Remarks

The main problem of derivation procedure for EFS is unification. We have got an algorithm of unification because our aim is to get a procedure to accept languages and apply it to MIEFS. The computational complexity of the algorithm is described in [2].

There are other formalizations of derivations for EFS. A famous one is CLP(X) [6]. CLP(Σ^+) could be got if we could give an algorithm to test the unifiability of two patterns. Makanin [9] showed the existence of the algorithm.

Fitting [5] also formalized EFS as a logic programming language. In the formalization, terms are elements of $\Sigma^+ \cup X$, not $(\Sigma \cup X)^+$, and the procedural semantics is out of consideration.

The original theory of EFS given by Smullyan [13] uses the elements of $(\Sigma \cup X)^*$ as terms. The derivation procedure and semantics as logic programming for the original EFS can be given in the same way as that of this paper by putting

$$E = \left\{ \begin{array}{l} cons(cons(x,y),z) = cons(x, cons(y,z)), \\ cons(\lambda, x) = cons(x, \lambda) = x \end{array} \right\}.$$

However, the results about CWA do not always hold because the empty word may be substituted for variables, and thus Lemma 3 does not hold.

The discussions on CWA can be applied to traditional logic programming (which is based on first order terms), if we introduce a proper size function of first order terms so that Lemma 3 holds [3].

Acknowledgements

The author would like to express his thanks to the reviewers for many helpful comments and suggestions on the original version of this paper.

References

[1] Arikawa, S. , Elementary Formal Systems and Formal Languages - Simple Formal Systems. *Memoirs of Fac. Sci., Kyushu University Ser. A.* Math. 24:47–75 (1970).

[2] Arikawa, S. , Shinohara, T. , and Yamamoto, A. , Elementary Formal System as a Unifying Framework for Language Learning, in Rivest, R. , Haussler, D. , and Warmuth, M. K. (eds.), *Proc. COLT'89*, 312–327, Morgan-Kaufmann, 1989.

[3] Arimura, H. , Completeness of Depth-Bounded Resolution in Logic Programming, Internal report, Research Institute of Fundamental Information Science, Kyushu University, 1989, to appear in 6th Conf. Proc. of JSSST.

[4] Dershowitz, N. and Manna, Z. , Proving Termination with Multiset Orderings. *CACM* 8(22):465–476 (1979).

[5] Fitting, M. , *Computability Theory, Semantics, and Logic Programming,* Oxford University Press, 1987.

[6] Jaffar, J. and Lassez, J.-L. , Constraint Logic Programming, in *Proc. Conference on Principle of Programming Languages*, 1987.

[7] Jaffar, J. , Lassez, J.-L. , and Maher, M. J. , Logic Programming Scheme, in DeGroot, D. and Lindstrom, G. (eds.), *Logic Programming: Functions, Relations, and Equations*, 211–233, Prentice-Hall, 1986.

[8] Lloyd, J. W. , *Foundations of Logic Programming Second, Extended Edition*, Springer - Verlag, 1987.

[9] Makanin, G. S. , The Problem of Solvability of Equations in a Free Semigroup. *Soviet Math. Dokl.* 18(2):330–335 (1977).

[10] Plotkin, G. D. , Building in Equational Theories, in *Machine Intelligence 7*, 132–147, Edinburgh University Press, 1972.

[11] Reiter, R. , On Closed World Data Bases, in Gallaire, H. and Minker, J. (eds.), *Logic and Data Bases*, 55–76, Plenum Press, 1978.

[12] Shapiro, E. Y. . Inductive Inference of Theories From Facts. Research Report 192, Yale University, 1981.

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

[14] Yamamoto, A. , A Theoretical Combination of SLD-Resolution and Narrowing, in Lassez, J.-L. (ed.), *Proc. 4th ICLP*, 470–487, The MIT Press, 1987.