We discuss procedural semantics and inference of negated ground atoms in elementary formal system (EFS). EFS is now a logic programming with associative unification. There are two problems on the SLD-resolution when we infer negated atoms. One is existence of infinitely many unifiers for two atoms, even maximally general. This prevents us finding a proper completed definitions of EFS's corresponding to the negation as failure rule. The other problem is existence of infinite derivations. They make it difficult to reject atoms not in the least Herbrand model. In the note, we give solutions for these problems. When we use the SLD-resolution to accept formal languages defined by EFS's, we assume that every refutation begins from a ground goal. Under the assumption we solve the first problem by introducing the variable-bounded EFS, which is powerful enough to define languages. The solution for the second problem is to bound the length of every SLD-derivation. We present it as an algorithm to decide whether a ground atom is in the least Herbrand model or not. We introduce the weakly reducing EFS as a class of EFS's where our algorithm is a complete realization of the closed world assumption.

1. INTRODUCTION

We discuss procedural semantics and inference of negated ground atoms in elementary formal system. The elementary formal system (EFS for short) was first introduced by Smullyan [12] to develop his recursive function theory. Arikawa [2] showed that EFS is useful to define formal languages. By regarding the SLD-resolution for EFS as a
procedure to accept the languages, we give a framework of inductive inference of formal languages [3].

An EFS is a triplet consisting of a set $\Sigma$ of symbols, a set $\Pi$ of predicate symbols and a set $\Gamma$ of definite clauses. The clauses in $\Gamma$ use patterns in $(\Sigma \cup X)^+$ as terms, where $X$ is a set of variables. By introducing a new function symbol $\text{cons}$ and assuming its associativity with an equality theory

$$E_{\text{assoc}} = \{ \text{cons}(\text{cons}(x, y), z) = \text{cons}(x, \text{cons}(y, z)) \},$$

EFS is now in the framework of the logic programming schema given by Jaffar et al. [8]. By using unification under $E_{\text{assoc}}$, words in $\Sigma^+$ are treated as arguments of atoms without translating them into first order terms.

There are two problems on the SLD-resolution when we infer negated atoms. One is existence of infinitely many unifiers for two atoms, even maximally general. This prevents us finding a proper completed definitions of EFS's to treat the negation as failure rule. The other problem is existence of infinite derivations. They make it difficult to reject atoms in the least Herbrand model.

In the note we give solutions for these problems. When we use the SLD-resolution to accept languages defined by EFS's, we assume that every refutation begins from a ground goal. Under the assumption we solve the first problem by introducing the variable bounded EFS. The variable-bounded EFS is powerful enough because it has been shown, by simulating Turing Machines, that every recursively enumerable languages is definable by a variable-bounded EFS [3]. The solution for the second problem is to bound the length of every SLD-derivation. We present it an algorithm to decide whether a ground atom is in the least Herbrand model or not. We introduce the weakly reducing EFS as a class of EFS's where our algorithm is a complete realization of the closed world assumption.

2. ELEMENTARY FORMAL SYSTEM

We start with recalling the definitions of EFS.

Let $\Sigma$, $X$, and $\Pi$ be mutually disjoint sets. We assume that $\Sigma$ and $\Pi$ are finite. We refer to $\Sigma$ 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 $x, y, z, \ldots$ and to each element of $\Pi$ as 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 words over the set $A$ without the empty word.

Definition. A term of $S$ is an element of $(\Sigma \cup X)^+$. A ground term of $S$ is an element of $\Sigma^+$. 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 $\Pi$ 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.

A well-formed formula, a clause, the empty clause, a definite clause, a goal clause, a substitution, and a variant of a clause are defined in the same way as in the
first order predicate logic. A substitution $\theta$ is denoted by a set \( \{ x_i := \tau_i, \ldots, x_n := \tau_n \} \) where $x_1, \ldots, x_n$ are distinct variables and $\tau_1, \ldots, \tau_n$ are terms. We put $\text{dom}(\theta) = \{ x_1, \ldots, x_n \}$. The substitution is ground if every $\tau_i$ is ground.

**Definition** (Smullyan [12]). An elementary formal system (EFS for short) $S$ is a triplet $(\Sigma, \Pi, \Gamma)$, where $\Gamma$ is a finite set of definite clauses. The definite clauses in $\Gamma$ are called axioms of $S$.

**Notation.**

1. For a term $\pi$, $|\pi|$ denotes the length of $\pi$, that is, the number of all occurrences of symbols and variables in $\pi$. For an atom $p(\pi_1, \ldots, \pi_n)$, let $|p(\pi_1, \ldots, \pi_n)| = |\pi_1| + \cdots + |\pi_n|$.
2. For every set $U$ of atoms, we put $U \mid_n = \{ A \in U \mid |A| \leq n \}$.
3. For an atom $A$ and variable $x$, $o(x, A)$ is the number of all occurrences of $x$ in $A$.
4. $\nu(\alpha)$ denotes the set of all variables in a term or an atom $\alpha$.
5. We define $\text{lob}(A \leftarrow B_1, \ldots, B_m) = m$ for a definite clause $A \leftarrow B_1, \ldots, B_m$, and $\text{lob}(S) = \max_{C \in \Pi} \text{lob}(C)$ for an EFS $S = (\Sigma, \Pi, \Gamma)$.
6. $\#(U)$ denotes the cardinality of a set $U$.

**Example 1.** Let $A = p(ax, by, cx)$. Then $|A| = 6$, $o(x, A) = 2$, and $\nu(p(ax, by, cx)) = \{ x, y \}$.

For an EFS $S = (\Sigma, \Pi, \Gamma)$, $\Sigma^+$ is the Herbrand universe and the set $B(S)$ of all ground atoms is the Herbrand base. A mapping $\mathcal{T}_S : 2^{B(S)} \rightarrow 2^{B(S)}$ is defined as

\[
\mathcal{T}_S(I) = \begin{cases} 
A \in B(S) & \text{there is a ground instance} \\
A \leftarrow B_1, \ldots, B_m & \text{of an axiom in } \Gamma \text{ such that } B_k \in I \text{ for } k = 1, \ldots, n.
\end{cases}
\]

and

\[
\begin{align*}
\mathcal{T}_S \uparrow 0 &= \phi, \\
\mathcal{T}_S \downarrow 0 &= B(S), \\
\mathcal{T}_S \uparrow n &= \mathcal{T}_S(\mathcal{T}_S \uparrow (n - 1)), \\
\mathcal{T}_S \downarrow n &= \mathcal{T}_S(\mathcal{T}_S \downarrow (n - 1)) \quad (n \geq 1), \\
\mathcal{T}_S \uparrow \omega &= \bigcup_{n \geq 0} \mathcal{T}_S \uparrow n, \\
\mathcal{T}_S \downarrow \omega &= \bigcap_{n \geq 0} \mathcal{T}_S \downarrow n.
\end{align*}
\]

$\mathcal{T} \uparrow \omega$ coincides with the least Herbrand model. Arikawa [2] defined formal languages in $\Sigma^+$ with the set $\mathcal{T}_S \uparrow \omega$.\(^1\)

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

\[
L(S, p) = \{ (\tau_1, \ldots, \tau_n) \in (\Sigma^+)^n : p(\tau_1, \ldots, \tau_n) \in \mathcal{T}_S \uparrow \omega \}.
\]

\(^1\)Precisely speaking, he used the provability in Smullyan [12].
In case \( n = 1 \), \( L(S, p) \) is a language over \( \Sigma \). A language \( L \subseteq \Sigma^+ \) is definable by an EFS or an EFS language if such \( S \) and \( p \) exist.

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

\[
\begin{align*}
p(a, b, c) &\leftarrow , \\
p(ax, by, cz) &\leftarrow p(x, y, z), \\
q(xyz) &\leftarrow p(x, y, z)
\end{align*}
\]

defines a language

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

3. **REFUTATION AND NEGATION AS FAILURE RULE**

An SLD-derivation for an EFS \( S = (\Sigma, \Pi, \Gamma) \) is the \( (\Gamma, E_{assoc}) \)-derivation in [8], which is an SLD-derivation with unification based on \( E_{assoc} \).

**Definition.** Let \( \alpha \) and \( \beta \) be a pair of term or atoms. Then a substitution \( \theta \) is a unifier of \( \alpha \) and \( \beta \), or \( \theta \) unifies \( \alpha \) and \( \beta \) if \( \alpha \theta = \beta \theta \). \( \alpha \) and \( \beta \) is unifiable if there exists a unifier of \( \alpha \) and \( \beta \). \( U(\alpha, \beta) \) denotes the set of all unifiers \( \theta \) of \( \alpha \) and \( \beta \) such that \( \text{dom}(\theta) \subseteq \nu(\alpha) \cup \nu(\beta) \).

Note that the definition of unifiers is declarative, that is, we have not introduce any unification algorithm yet.

An SLD-derivation for EFS is formally defined with a computation rule, which selects an atom from every goal clause such as in Lloyd [9].

**Definition.** Let \( S \) be an EFS, \( G \) be a goal of \( S \), and \( R \) be a computation rule. An SLD-derivation from \( G \) is a sequence of triplets \( (G_i, \theta_i, C_i) \) \( (i = 0, 1, \ldots) \) which satisfies the following conditions:

1. \( G_i \) is a goal, \( \theta_i \) is a substitution, \( C_i \) is a variant of an axiom of \( S \), and \( G_0 = G \).
2. \( \nu(C_i) \cap \nu(C_j) = \phi \) \( (i \neq j) \), and \( \nu(C_i) \cap \nu(G) = \phi \) for every \( i \).
3. If \( G_i = \leftarrow A_1, \ldots, A_k \) and \( A_m \) is the atom selected by \( R \), then \( C_i = A \leftarrow B_1, \ldots, B_q, \theta_i \) is a unifier of \( A \) and \( A_m \), and

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

\( A_m \) is a selected atom of \( G_i \), and \( G_{i+1} \) is a resolvent of \( G_i \) and \( C_i \) by \( \theta_i \).

We call an SLD-derivation simply a derivation. For a finite derivation \( (G_i, \theta_i, C_i) \) \( (i = 0, \ldots, n) \), we define its length as \( n \). A finite derivation ending with the empty goal \( \Box \) is called an SLD-refutation, or refutation for short. A derivation is finitely failed with depth \( n \) if its length is \( n \) and there is no axiom which satisfies the condition 3 for the selected atom of the last goal.

**Definition.** A derivation \( (G_i, \theta_i, C_i) \) \( (i = 0, 1, \ldots) \) is fair if it is finitely failed or, for each atom \( A \) in \( G_i \), there is a \( k \geq i \) such that \( A \theta_1, \ldots, \theta_{k-1} \) is the selected atom of \( G_k \). A computation rule is fair if it makes all derivations fair.
Definition. The negation as failure rule is the rule that infers \( \neg A \) if there is a number \( k \) such that any fair derivation from \( \neg A \) is finitely failed with its depth less than or equal to \( k \).

\( B(S) - T_\omega \) is identical to the set
\[
\{ A \in B(S) ; \neg A \text{ is inferred under the negation as failure rule} \}
\]
but is not characterized with the completion of \( S \) because there may be infinitely maximal unifiers for a pair of atoms \([8, 11]\).

4. VARIABLE-BOUNDED EFS

When we use the derivations to accept the languages defined by an EFS, we can assume that every derivation starts from a ground goal.

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

For variable-bounded EFS there is a simple unification algorithm useful to construct derivations from ground goals.

Lemma 1. Let \( \alpha \) and \( \beta \) be a pair of atoms and \( \alpha \) be ground. Then every unifier of \( \alpha \) and \( \beta \) is ground and \( U(\alpha, \beta) \) is finite and computable.

Proof. It is sufficient to show the result in the case that \( \alpha = p(\tau_1, \ldots, \tau_n) \) and \( \beta = p(\pi_1, \ldots, \pi_n) \). At first we show that \( U(\tau_i, \pi_i) \) is finite and computable. Suppose a substitution \( \theta \) is in \( U(\tau_i, \pi_i) \). Then \( \text{dom}(\theta) = \nu(\pi_i) \), \( x\theta \) is ground, and \( |x\theta| \leq |\tau_i| \) for every \( x \in \text{dom}(\theta) \). Thus \( U(\tau_i, \pi_i) \) is obtained by generating a finite and computable set
\[
S(\tau_i, \pi_i) = \{ \theta \mid \text{dom}(\theta) = \nu(\pi_i), x\theta \text{ is ground}, |x\theta| \leq |\tau_i| \text{ for every } x \in \text{dom}(\theta) \},
\]
and testing every element \( \theta \) of \( S(\tau_i, \pi_i) \) to see whether \( \tau_i = \pi_i \theta \) or not.

Since a unifier of \( \alpha \) and \( \beta \) is also a unifier of \( \tau_i \) and \( \pi_i \) for \( i = 1, \ldots, n \), \( U(\alpha, \beta) \) can be computed by testing every tuple of \( (\sigma_1, \ldots, \sigma_n) \) where \( \sigma_i \in U(\tau_i, \pi_i) \) to see whether \( \alpha = \beta \sigma_1 \ldots \sigma_n \) or not.

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 EFS defined in Example 2 is variable-bounded.

From Lemma 1 the completion of an EFS can be defined in the same way of the ordinal logic programming with an equality theory
\[
E_{assoc}^* = \{ \tau = \pi \leftrightarrow \bigwedge_{i=1}^k \text{eqn}(\theta_i) \mid \tau \text{ is a ground term, } \pi \text{ is a term, and } \theta_1, \ldots, \theta_k \text{ are all unifiers of } \pi \text{ and } \tau \},
\]
where \( \text{eqn}(\theta) = (x_1 = \tau_1 \land \cdots \land x_n = \tau_n) \) for a substitution \( \theta = \{ x_1 := \tau_1, \ldots, x_n := \tau_n \} \), and \( \text{eqn}(\varepsilon) = \text{true} \) for the empty substitution \( \varepsilon \). By Lemma 2 and König's Lemma, the negation as failure rule is complete with respect to the completion of EFS.
Theorem 1. For any variable-bounded EFS $S$ and $A \in B(S)$, following three are equivalent:

1. $\neg A$ is inferred under the negation as failure rule.
2. $\neg A$ is a logical consequence of the completion of $S$.
3. there is no infinite fair derivation from $\leftarrow A$.

5. WEAKLY REDUCING EFS

In this section we give a method of avoiding infinite derivations as an algorithm to decide whether a ground atom $A$ is in $T_S^\omega$ or not. We make use of a recursive function $f_S$ such that

$$A \in T_S^\omega \Leftrightarrow A \in T_S^\omega \uparrow f_S(A).$$

**Example 3.** Let $S = (\{a, b\}, \{p\}, \Gamma)$ with

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

and $f_S(A) = 1$ for every $A \in B(S)$. Then

$$A \in T_S^\omega \Leftrightarrow A \in T_S^\omega \uparrow f_S(A)$$

for every $A \in B(S)$.

Since every recursively enumerable language in $\Sigma^+$ is definable by a variable-bounded EFS [3], the $f_S$ does not exist for every $S$. We give a class of EFS’s in which the $f_S$ always exists and computable from the syntax of $S$.

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

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

for any substitution $\theta$ and $i = 1, \ldots, n$. An EFS is weakly reducing if its axioms are all weakly reducing.

It is decidable whether an EFS is weakly reducing or not by the following lemma.

**Lemma 3.** Let $A$ and $B$ be a pair of atoms. Then $A \theta \geq (>) B \theta$ for any substitution $\theta$ if and only if

$$| A | \geq (>) | B |,$$

$$\sigma(x, A) \geq \sigma(x, B)$$

for any variable $x$.

The proof of Lemma 3 is similar to Lemma 2.1 in [3]. Both EFS’s in Example 2 and in Example 3 are weakly reducing.

Now we give the main theorem.

**Theorem 2.** Let $S = (\Sigma, \Pi, \Gamma)$ be a weakly reducing EFS. Then

$$A \in T_S^\omega \Leftrightarrow A \in T_S^\omega \uparrow \# \left( B(S) \mid _{\mid A} \right).$$
PROOF. It suffices to prove the $\Rightarrow$ part. First we show that
\[
T_S\left(\left( T_S \uparrow k \right) \n \right)_{n} = \left( T_S \uparrow k + 1 \right) \n_{n} \quad (k \geq 0).
\]
(1)
The $\subseteq$ part is proved directly from the definition. Suppose $B \in (T_S \uparrow k + 1) \n_{n}$. Then there is an ground instance $B \leftarrow B_1, \ldots, B_q$ of a clause in $\Gamma$. Since $S$ is weakly reducing,
\[
\{ B_1, \ldots, B_q \} \subset (T_S \uparrow k) \n_{n},
\]
and thus
\[
B \in T_S\left(\left( T_S \uparrow k \right) \n \right)_{n}.
\]
Since $T_S$ is monotone,
\[
( T_S \uparrow k ) \n_{n} \subset ( T_S \uparrow k + 1 ) \n_{n} \quad (k \geq 0).
\]
(2)
Moreover, if $( T_S \uparrow K ) \n_{n} = ( T_S \uparrow K + 1 ) \n_{n}$ for some $K$ then
\[
( T_S \uparrow K ) \n_{n} = ( T_S \uparrow k ) \n_{n} \quad (k \geq K)
\]
from equation (1). Since $( T_S \uparrow k ) \n_{n} \subset B(S) \n_{n}$ and $B(S) \n_{n}$ is finite,
\[
( T_S \uparrow k ) \n_{n} = ( T_S \uparrow \#(B(S) \n_{n}) ) \n_{n} \quad (k \geq \#(B(S) \n_{n})).
\]
(3)
By the equations (2) and (3), we get
\[
( T_S \uparrow k ) \n_{n} \subset ( T_S \uparrow \#(B(S) \n_{n}) ) \n_{n} \quad (k \geq 0).
\]
Now let $A \in T_S \uparrow k$ for some $k$. Then
\[
A \in (T_S \uparrow k) \mid A \mid_{n} \subset ( T_S \uparrow \#(B(S) \mid A \mid) ) \mid A \mid_{n} \subset T_S \uparrow \#(B(S) \mid A \mid),
\]
and this shows the $\Rightarrow$ part of the theorem.

From the above theorem, we get the intended algorithm by bounding the length of derivation from $\leftarrow A$.

**Theorem 3.** Let $S$ be a weakly reducing EFS. If $A \in T_S \uparrow \omega$, then there is a refutation from $\leftarrow A$ with length less than or equal to $\text{dep}(\text{lob}(S), \#(B(S) \mid A \mid))$ where
\[
\text{dep}(m, n) = \begin{cases} 
1 & \text{if } m = 0, \\
\sum_{i=0}^{n-1} m^i & \text{otherwise.}
\end{cases}
\]

**Example 4.** Let $S$ be the EFS in Example 3. There is no refutation from $\leftarrow p(b)$ with length less than or equal to $\text{dep}(1, 2) = \sum_{i=0}^{1} 1^i = 1 + 1 = 2$. Thus we can decide $p(b) \notin T_S \uparrow \omega$.

There is a subclass of weakly reducing EFS's where constructing a derivation from $\leftarrow A$ implies bounding its own length.

**Definition.** A clause $A \leftarrow B_1, \ldots, B_n$ is reducing if
\[
| A \theta | > | B_\theta |$

for any substitution \( \theta \) and \( i = 1, \ldots, n \). An EFS is *reducing* if its axioms are all reducing.

It is also decidable from Lemma 3 whether an EFS is weakly reducing or not. If \( S \) is reducing, there is no infinite derivation from \( \leftarrow A \) and thus \( T_S \uparrow \omega = B(S) - T_S \downarrow \omega \).

It is shown in [3], by simulating linear bounded automata, that every context-sensitive language \( L \subseteq \Sigma^+ \) is definable by a weakly reducing EFS. We give a result on the class of languages defined by reducing EFS's.

**Theorem 4.** 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 obtain a reducing EFS which defines \( L \), by using every non-terminal symbol of \( G \) as a predicate symbol of \( S \), and transforming every rule in \( G \) of the form \( A \rightarrow B, C \) into a clause

\[
A(x) \leftarrow B(x), C(y)
\]

and every rule of the form \( A \rightarrow a \) into

\[
A(a) \leftarrow .
\]

The reverse of Theorem 4 does not hold.

**Example 5.** A reducing EFS \( S = (\{a, b, c\}, \{p, q\}, \Gamma) \) with

\[
\Gamma = \begin{cases}
  p(a, bb, cc) \leftarrow, \\
  p(ax, by, cz) \leftarrow p(x, y, z), \\
  q(abc) \leftarrow, \\
  q(axyz) \leftarrow p(x, y, z)
\end{cases}
\]

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

### 6. RELATION TO OTHER WORKS

The unification of a ground term and a term possibly with variables is NP-complete [5] (see also [1]). More precise results were discussed in [3].

CLP \((\Sigma^+)\), which is an instance of CLP \((X)\) [7], would be another formalization of EFS as logic programming if we could give an algorithm to test the unifiability of two patterns. Makanin [10] showed the existence of the algorithm. Fitting [6] formalized EFS as logic programming in the case that terms are elements of \( \Sigma^+ \cup X \), not \( (\Sigma \cup X)^+ \). In the formalization the procedural semantics is out of consideration.

The original theory of EFS given by Smullyan [12] uses the elements of \( (\Sigma \cup X)^+ \cup \{\lambda\} \) as terms, where \( \lambda \) represents the empty word. In this case, a derivation for EFS can be formalized in the same way as that in this paper. However, the results on weakly reducing EFS do not always hold because the empty word may be substituted for variables and thus Lemma 3 does not hold.

For traditional logic programming, an algorithm same as in Section 5 has been pointed out by Arimura [4]. He has formalized the weakly reducing programs in the ordinal logic programming possibly with negated atoms in the bodies. He has shown the completeness of the algorithm with respect to the perfect model semantics.
REFERENCES


