PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1988,
edited by ICOT. © ICOT, 1988

SEMANTICS OF LOGIC PROGRAMS OVER SEQUENCE DOMAINS

Susumu Yamasaki

Department of Information Technology
Okayama University, Tsushimanaka, Okayama 700, Japan

ABSTRACT

In this paper we establish a semantics of a logic program over a sequence domain. The domain is the set of all finite and infinite sequences of ground atoms. The motivation is to construct a recursion equation set involving variables over a sequence domain, which is regarded as a dataflow program. The dataflow program constructible from a logic program denotes the formation of the minimal Herbrand model of the original logic program by means of sequence variables. It contains functions corresponding to the inferences caused by definite clauses, and fair merge functions necessary for the purpose of taking set unions in terms of variables denoting ground atom sequences. The functions in accordance with the inferences concerning definite clauses are obtained by eliminating nondeterminism usually involved in such inferences. It is shown that the dataflow program defines a continuous function from a direct product of a sequence domain to itself, therefore there exists a least fixpoint of the function. The fixpoint completely represents the minimal Herbrand model of the original logic program, which is essential for its finite computation. Finally the fixpoint is interpreted as a semantics of the logic program.

1 INTRODUCTION

The semantics of logic programs have been investigated from various points of view since van Emden and Kowalski defined it from model-theoretic, fixpoint and operational approaches (Abdallah 1984, Apt etal. 1982, van Emden et al. 1976, Fitting 1985, Frauden 1985, Lassez et al. 1984, Lassez et al. 1985, and Yamasaki et al. 1987) . There is a way to define the semantics in (Fitting 1985) distinguishable from others in the sense that it is defined over a sequence domain.

In this paper we demonstrate another semantics of a logic program over a sequence domain. It is legal in order to express the denotations of logic programs and to realize their computation mechanism based on dataflow networks. Also it is necessary to establish a method of transforming logic programs into functional programs through dataflows over sequence domains. Because it is rather difficult to find a direct way to get a functional program computing a given logic program, which contains nondeterministic procedures and is regarded as computing relations. It is easier to construct, as an intermediate form, a dataflow program involving variables over a sequence domain such that a logic program is transformable to the dataflow program and any functional program can be generated from it. It is observed from (Apt et al. 1982 and van Emden et al. 1976) that the denotation of a logic program, that is, its minimal Herbrand model is obtained by the limit of the following concecutive procedures: First infer a set of (ground) atoms by means of each definite clause from an already acquired set of atoms. (At the beginning, the already acquired set is empty.) Next unite such newly infered sets per a predicate symbol and take the union of the united sets each of which is in accordance with a predicate symbol. Then we regard the (whole) union as an already acquired set. Repeat this procedure until the acquired set can be no more expanded by the next procedure. In addition, it is notable that a set of atoms with a predicate symbol can be represented by a variable denoting a finite or infinite sequence from the Herbrand base.

In order to represent the denotation of a logic program by means of variables over a sequence domain, we first have a relation among variables, in accordance with the inference caused by each definite clause for an already given set of atoms. Next we prepare for a satisfactory device of taking the union of sets of atoms per a predicate symbol on condition that the denotation of each set of atoms is assumed to be expressed by a variable. Then we make up the relations as to the inferences of definite clauses and the devices of uniting sets of atoms, into a recursion relation set among variables. The recursion relation set will realize the above mentioned step-by-step procedure to get a newly acquired set of atoms from an already obtained set. A relation as to the inference of each definite clause will be constructed such that an output variable representing the conclusion-part of a definite clause may be infered from input variables representing its premise-part.

The relation contains unbounded nondeterminism as a function to define an output variable from input variables. For the relation to be equational, we will have a kind of oracle to eliminate such nondeterminism. Also we shall make use of the fair merge (with an adequate oracle) in (Park 1983) as a desirable tool to represent the union of sets of atoms by means of variables over a sequence domain.

Finally we will have a set of recursion equations as a dataflow over a sequence domain, transformed from a given logic program such that there exists a (least) fixpoint of the recursion equation set. It will be shown that any ground atom is in the minimal Herbrand model of the logic program iff it is contained in the denotation of the least fixpoint of the corresponding recursion equation set. In this sense, we will come up with a conclusion that the least fixpoint is a semantics of a given logic program.

2 BASIC NOTATIONS

In this paper, a logic program means a set of definite (Horn) clauses. A definite clause is a clause of the form $A \leftarrow B_1 \dots B_n$ $(n \geq 0)$, where A, $B_1, \dots,$ and B_n are atoms.

For a definite clause C, $Head(C)$ means the conclusion-part (head) of C, that is, the left-hand side of $\leftarrow$ in C. $Body(C)$ denotes the set of atoms in the premise-part (procedure body) of C, that is, the set of atoms in the right-hand side of $\leftarrow$ in C .

A substitution is a finite set of the form $\{x_1 \mid t_1, \dots, x_n \mid t_n\}$, where each x_i is a variable and each t_i is a term such that x_i does not occur in t_i. For a substitution σ and an atom A, $A\sigma$ is an atom obtained by substituting terms in σ for all the corresponding variables of σ occurring in A simultaneously.

The Herbrand universe of a logic program L is the set of all variable-free terms constructible from constant symbols and function symbols in L. The Herbrand base H_L of L is the set of all variable-free atoms constructible from symbols in L. A ground atom is an atom in the Herbrand base.

By $At(P)$, $At_1(P)$, $\dots$, $At_i(P)$, $\dots$, we mean the sets of ground atoms with the predicate symbol P. For $I \subset H_L$, $PRED(I)$ means the set of predicate symbols in I. For $A \in H_L$, $Pr(A)$ denotes the predicate symbol in A. For $I \subset H_L$ and the predicate symbol P, we define $[I]_P = \{A \in I \mid Pr(A) = P\}$.

A logic program will be transformed into a set of recursion equations involving sequence variables. Each sequence variable denotes a finite or infinite sequence of elements from a base domain. As a base domain, we have a domain $D_L = H_L \cup \{\tau\}$, where H_L is the Herbrand base of a logic program L and τ is a special symbol not in H_L. Intuitively speaking, τ denotes a time delay occurring in a sequence from H_L, and is similar to the hiaton introduced in (Park 1983 and Wadge 1979).

For a set F, $\#F$ denotes the cardinal number of F. F^∞ denotes the set of functions (from ω to F) such that if $u \in F^\infty$ and $u(p)$ is defined, then $u(q)$ is always defined for $q \leq p \in \omega$. Intuitively F^∞ is regarded as denoting the set of all finite and infinite sequences from F. $nil \in F^\infty$ is the function such that $nil\,(p)$ is undefined for any $p \in \omega$.

For $u \in F^\infty$, let $\mid u \mid = \#\{k \mid u(k)$ is defined $\}$. $\mid u \mid$ is interpreted as the length of the sequence denoted by u. Note that $\mid nil \mid = 0$.

Now let $u[p] \in F^\infty$ be defined by:

$u[p](q) = u(q)$ if $p \geq q$ and
$u[p](q) = nil(q)$ otherwise.

$u[p]$ is regarded as an initial part of the sequence for u, truncated up to length $p + 1$.

A partial order $\prec$ on D_L is defined by:

$\tau \prec A$ and $A \prec A$ for any $A \in H_L$.

A partial order $\sqsubset$ on D_L^∞ is defined by:

$u \sqsubset v$ for u, v in D_L^∞ iff $u(p) \prec v(p)$ for $p \in \omega$ whenever $u(p)$ is defined.

The partial order $\sqsubset$ is extended to act on $(D_L^\infty)^m$:

$(u_1, \dots, u_m) \sqsubset (v_1, \dots, v_m)$ iff $u_p \sqsubset v_p$ for $1 \leq p \leq m$.

The least upper bound of $G \subset (D_L^\infty)^m$ is denoted by $\sqcup G$. The partial order $\sqsubset$ is sequentially complete in the sense that any sequence $w_0 \sqsubset w_1 \sqsubset \dots$ has a least upper bound $\sqcup_{i \in \omega} w_i$.

For further discussions, we have some notations:

ω means the set of natural numbers. Let I_m: $\omega \to \omega^m$ $(m \geq 1)$ be a bijection such that if $I_m(p) = (p_1, \dots, p_m)$ then $p_i \leq p$ for $1 \leq i \leq m$. I_m is necessary to indicate an m-tuple by a natural number such that each element of the m-tuple is not greater than the number. Also let a projection $J_{m,i}$: $\omega^m \to \omega$ be defined by $J_{m,i}(p_1, \dots, p_m) = p_i$. $J_{m,i}$ provides the i-th element of an m-tuple.

first: $F^\infty \to F^\infty$ is the function satisfying

first $(u)(p) = u(0)$ for $u \in F^\infty$, and $p \in \omega$.

next: $F^\infty \to F^\infty$ is the function satisfying

next $(u)(p) = u(p + 1)$ for $u \in F^\infty$, and $p \in \omega$.

Note that $p + 1$ means the successor of p. For *first* and *next*, refer to (Ashcroft et al. 1976).

eq: $(F^\infty)^2 \to \{$ *true, false* $\}$ is defined by

$eq\ (a,b) = true$ if $a = b$, and $eq\ (a,b) = false$ otherwise.

if-then-else: $\{$ *true,false* $\} \times (F^\infty)^2 \to F^\infty$ is defined by

if-then-else $(t, f_1, f_2) =$

$$\begin{cases} f_1 & \text{if } t = true \text{ and } f_1 \in F^\infty, \\ f_2 & \text{if } t = false \text{ and } f_2 \in F^\infty, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

if-then-else (t, x, y) is expressed by $(t \to x, y)$.

3 REPRESENTATION OF SET OF ATOMS BY SEQUENCE VARIABLE

First we have semantics of logic programs, which are concerned with their finite computations.

Definition 1. Given a logic program L, TR_L: $L \times 2^{H_L} \to 2^{H_L}$ is defined by

$TR_L(A \leftarrow B_1 \dots B_m, I) =$

$\{A\sigma \in H_L \mid \exists\sigma($ substitution $)$: $B_1\sigma, \dots, B_m\sigma \in I\}$.

The semantics of L is $Sem(L) = \cap\{I \subset H_L \mid \cup_{C \in L} TR_L(C, I) \subset I\}$.

Note that $TR_L(C, I)$ denotes the set of all ground atoms deduced from $I \cup \{C\}$. Now let

$Trans_L(I) = \cup_{C \in L}\ TR_L(C, I) =$

$\cup_{P \in PRED(H_L)} \cup_{P = Pr(Head(C))} TR_L(C, I)$.

We have to find a method of expressing (i) TR_L, and (ii) $\cup_{P = Pr(Head(C))}$, in addition to $\cup_{P \in PRED(H_L)}$, in terms of the sequence domain D_L^∞.

To reach the method, we need the definition of the representation of $At(P) \subset H_L$ by $u_P \in D_L^\infty$.

Definition 2. We say that $u_P \in D_L^\infty$ represents $At(P)$ if (1) for any $A \in At(P)$ there exists $k \in \omega$ such that $u_P(k) = A$, and (2) for any $i \in \omega$ either $u_P(i)$ is undefined or $u_P(i) \in At(P) \cup \{\tau\}$.

It is meant by $u_P \Rightarrow At(P)$ that u_P represents $At(P)$. For the purpose of expressing ' $\cup_{P \in PRED(H_L)}$ ' over the sequence domain, the following definition will be satisfactory.

Definition 3. We say $U \subset D_L^\infty$ represents $I \subset H_L$ if (1) for any $P \in PRED(I)$ there exists $u_P \in U$ such that $u_P \Rightarrow [I]_P = \{A \in I \mid Pr(A) = P\}$, and (2) $\cup_{P \in PRED(I)}\ u_P = U$. By $U \Rightarrow I$, it is meant that U represents I.

Now we investigate the relation among variables over D_L^∞, which is concerning an inference ' TR_L ' caused by each clause of L. Assume that $Pr(B_{i,r}) = Q_{i,r}$ $(1 \le r \le n_i)$ for each $C_i \equiv A_i \leftarrow B_{i,1}, \dots, B_{i,n_i}$ in $L = \{C_1, \dots, C_k\}$. The first subgoal we will reach is to construct $u_i \in D_L^\infty$ such that if $v_{i,r} \Rightarrow At(Q_{i,r})$ for $1 \le r \le n_i$, then $u_i \Rightarrow TR_L(C_i, \cup_r At(Q_{i,r}))$. Then $v_{i,1}, \dots, v_{i,n_i}$, which are input sequence variables, are related with u_i, as an output sequence variable, through the clause C_i. To do so, for C_i, we define the set the member of which is the expression of the form $A\sigma \in H_L$ such that each $B_{i,r}\sigma$ matches the $q_{i,r}$-th denotation of $v_{i,r}$ for $q_{i,r} = J_{n_i,r}(I_{n_i}(q)) \le q$.
That is, let

(3.1) $Out_i(q) =$
$\{A\sigma \in H_L \mid \exists\sigma : B_{i,r}\sigma = v_{i,r}(J_{n_i,r}(I_{n_i}(q))), 1 \le r \le n_i\}$
for $C_i \equiv A_i \leftarrow B_{i,1}, \dots, B_{i,n_i} \in L$, where $v_{i,r} \Rightarrow At(Pr(B_{i,r}))$ for $1 \le r \le n_i$.

Note that $Out_i(q) = \{A\sigma \in H_L\}$ for any $q \in \omega$ if $C_i \equiv A_i \leftarrow$. It is easy to have the following lemma.

Lemma 1. Let $Out_i(q)$ be the set defined by (3.1). Then $\cup_{q \in \omega}\ Out_i(q) = TR_L(C_i, \cup_{1 \le r \le n_i} At(Pr(B_{i,r}))$.

We assume that the q-th denotation of u_i depends on $v_{i,1}[q], \dots, v_{i,n_i}[q]$, that is, $u_i(q)$ depends on the finite part obtained by truncating $v_{i,1}, \dots, v_{i,n_i}$ up to length $q+1$. The assumption is taken for a simple treatment of the relation caused by 'TR_L' in terms of sequence variables.

Then we define

$$(3.2) \begin{cases} u_i(q) \in \ \cup_{p \le q} Out_i(p) \\ \qquad \text{if } \cup_{p \le q}\ Out_i(p) \text{ is not empty,} \\ u_i(q) = \ \tau \\ \qquad \text{if } \cup_{p \le q}\ Out_i(p) \text{ is empty.} \end{cases}$$

Note that $Out_i(p)$ depends on $v_{i,1}, \dots, v_{i,n_i}$. It is also notable that this definition is not absolutely unique, but is enough for u_i to satisfy $u_i \Rightarrow TR_L(C_i, \cup_r At(Pr(B_{i,r})))$. We shall show it later.

In order to select one as $u_i(q)$ from $\cup_{p \le q}\ Out_i(p)$, we first choose $Out_i(p)$ for some $p \le q$, and next pick up a ground atom from $Out_i(p)$.

By Lemma 1, $\cup_{p \in \omega}\ Out_i(p) = TR_L(C_i, \cup_r At(Pr(B_{i,r})))$. Thus each element in $\cup_{p \in \omega}\ Out_i(p)$ should be chosen as $u_i(q)$ $(q \in \omega)$ in order that $u_i \Rightarrow TR_L(C_i, \cup_r At(Pr(B_{i,r})))$. This means in general that $p \in \omega$ should be taken to indicate $Out_i(p)$. In addition, to cope with the case that $\#Out_i(p)$ is ω, p should be selected an arbitrary number of times for each element in $Out_i(p)$ to be chosen. To satisfy the above conditions, we make use of a function (in ω^ω), where any natural number occurs an arbitrary number of times.

Definition 4. We say that a function f in ω^ω is fair if $\#\{p \mid f(p) = q\} = \omega$ for any $q \in \omega$.

Note that ω^ω is the set of functions from ω to ω, and is regarded as the set of infinite sequences from ω.

To provide fair functions in ω^ω, we utilize the *dmerge* function as below. It was investigated in (Park 1983).

dmerge : $F^\infty \times F^\infty \times \{0,1\}^\infty \to F^\infty$ is defined by:

$$
\begin{aligned}
&(3.3)\ \ dmerge(u,v,\delta)(p) = \\
&(\ eq(\delta, nil) \to nil(p), \\
&\qquad (eq(p,0) \to (\ eq(\delta(p),0) \to u(0), v(0)\ ), \\
&\qquad (\ eq(\delta(p),0\ ) \to (\ eq(u,nil) \to nil(p), \\
&\qquad\qquad dmerge(next(u), v, next(\delta))(p-1)), \\
&\qquad (\ eq(v,nil) \to nil(p), \\
&\qquad\qquad dmerge(u, next(v), next(\delta))(p-1)))))
\end{aligned}
$$

for $p \in \omega$.

Now $FM_\delta : F^\infty \times F^\infty \to F^\infty$ is defined by $FM_\delta(u,v) = dmerge(u,v,\delta)$. FM_δ is said a fair merge function if $\#\{k \mid \delta(k) = 0\} = \#\{k \mid \delta(k) = 1\} = \omega$.

Using *dmerge*, we define a recursion equation by:

$$(3.4)\ \ w = dmerge(Succ(w), 0^\omega, \delta) = FM_\delta(Succ(w), 0^\omega),$$

where (1) $Succ : \omega^\infty \to \omega^\infty$ is defined by $Succ(u)(q) = u(q)+1$ for $u \in \omega^\infty$ and $q \in \omega$, and (2) δ is a function from ω to ω such that $\delta(0) = 1$ and $\#\{k \mid \delta(k) = 0\} = \#\{k \mid \delta(k) = 1\} = \omega$.

It is not difficult to see that the recursion equation (3.4) has the fixpoint, because of properties of *dmerge*. Let $Fs^\delta \in \omega^\omega$ be the fixpoint of (3.4).

Lemma 2.

(1) Fs^δ is a fair function in ω^ω.

(2) $Fs^\delta(j) \le j$ for $j \in \omega$.

Proof. (1) $Fs^\delta(0) = 0$, since $\delta(0) = 1$. Now let $Occur(k,h)$ mean that $\#\{p \mid Fs^\delta(p) = k\} \ge h$.

(i) It is seen that $Occur(0,h)$ for any $h \in \omega$, because $Fs^\delta = dmerge(Succ(Fs^\delta), 0^\omega, \delta)$ and $\#\{k \mid \delta(k) = 1\} = \omega$.

(ii) Assume that for some h, $Occur(n,h)$ for $n \le k$. Note that $Fs^\delta(q) = k+1$ for some $q > p$ if $Fs^\delta(p) = k$, since Fs^δ is applied to an argument of *dmerge*. Thus $Occur(k+1,h)$. By mathematical induction, for some $h \in \omega$, $Occur(n,h)$ for any n.

(iii) Since $Occur(0,1)$ from (i), it follows from (ii) that $Occur(n,1)$ for any $n \in \omega$. Suppose that $Occur(k,h)$ for any k and $h \le m$. $Fs^\delta(p) = k$ $(k > 0)$ means that $Succ(Fs^\delta)(r) = k$ for some $r < p$. That is, $Fs^\delta(r) = k-1$. Thus $Occur(k-1,m)$. Finally $Occur(0,m)$. It follows from (i) that $Occur(0, m+1)$ holds. Thus $Occur(k, m+1)$ must hold. That is, $Occur(k,h)$ for any k and any h.

(2) If $j = 0$, then the lemma holds, since $Fs^\delta(0) = 0$. Assume that $Fs^\delta(h) \le h$ for $h \le k$. Since $dmerge(Succ(Fs^\delta), 0^\omega, \delta)(k+1) = Fs^\delta(k+1)$ is either $Succ(Fs^\delta)(k) \le k+1$ or 0, $Fs^\delta(k+1) \le k+1$. This completes the proof. Q.E.D.

4 RECURSION EQUATION SET FOR LOGIC PROGRAM

In this section, we first have recursion equations as to sequence variables, based on the relation (3.2). Next we have a satisfactory device of taking unions of sets in terms of sequence variables. Then we compile them into a set of recursion equations.

4.1 Equation Derived from Definite Clause

By lemma 2, $Fs^\delta(q) \le q$ and any $p \in \omega$ occurs in Fs^δ an arbitrary number of times. Thus, Fs^δ is feasible to indicate $Fs^\delta(q) = p$ and the set $Out_i(p)$ when we identify $u_i(q)$ by (3.2). Here note that we have to select any atom in $Out_i(Fs^\delta(q))$ at least once in order that a variable, say u, may represent $\cup_{q\in\omega} Out_i(Fs^\delta(q)) = \cup_{p\in\omega} Out_i(p)$. To have a correspondence of $q \in \omega$ with the set $Out_i(q)$ and enumerate all the members in the set, we assume the function

$$(4.1)\ \ R_i : \omega \to (\omega \to 2^{H_L})$$

such that $R_i(q)$ is a bijection from $\{0,1,\ldots,\#Out_i(q)-1\}$ to $Out_i(q)$.

Suppose that for $p_0 < p_1 < \ldots \in \omega$, $Fs^{\delta_i}(p_0) = Fs^{\delta_i}(p_1) = \ldots \ne Fs^{\delta_i}(q)$ $(q \ne p_i$ for $i \in \omega)$. Then $Out_i(Fs^{\delta_i}(p_0)) = Out_i(Fs^{\delta_i}(p_1)) = \ldots$. To get all of $Out_i(Fs^{\delta_i}(p_0))$, it is sufficient to enumerate its members by $R_i(Fs^{\delta_i}(p_0))$: $\omega \to Out_i(Fs^{\delta_i}(p_0))$ and by $Fs^{\gamma_i}(0)$, $Fs^{\gamma_i}(1)$, $\ldots$, on the basis of some fair function Fs^{γ_i}. (Note any $r \in \omega$ occurs in Fs^{γ_i}.)

To get t from Fs^{δ_i} and $Fs^{\delta_i}(p_t)$, it is sufficient to define $Ord : \omega^\omega \times \omega \to \omega$ by

$$(4.2)\ \ Ord(Fs^{\delta_i}, q) = \#\{r \mid Fs^{\delta_i}(q) = Fs^{\delta_i}(r), r \le q\} - 1.$$

Then $Ord(Fs^{\delta_i}, p_t) = t$, and

$$R_i(Fs^{\delta_i}(p_0))(Fs^{\gamma_i}(t)) = R_i(Fs^{\delta_i}(p_t))(Fs^{\gamma_i}(Ord\,(Fs^{\delta_i}, p_t))).$$

Finally, on the basis of (3.1)-(3.4) and (4.1)-(4.2), we define

(4.3) for $q \in \omega$

$$\begin{cases} u_i(q) = & R_i(Fs^{\delta_i}(q))(Fs^{\gamma_i}(Ord(Fs^{\delta_i}, q))) \\ & \text{if } Fs^{\gamma_i}(Ord(Fs^{\delta_i}, q)) \leq \#Out_i(Fs^{\delta_i}(q)) - 1, \\ u_i(q) = & \tau \\ & \text{if } Fs^{\gamma_i}(Ord(Fs^{\delta_i}, q)) \geq \#Out_i(Fs^{\delta_i}(q)). \end{cases}$$

We have the following theorem which shows legality of (4.3) to express the inference concerning each definite clause by means of sequence variables.

Theorem 1. Let $L = \{C_1, \ldots, C_k\}$ be a logic program such that $C_i \equiv A_i \leftarrow B_{i,1} \ldots B_{i,n_i}, 1 \leq i \leq k$. Assume that u_i is defined by (4.3), on condition that $v_{i,r} \Rightarrow At(Pr(B_{i,r}))$ for $1 \leq r \leq n_i$. Then $u_i \Rightarrow TR_L(C_i, \cup_r At(Pr(B_{i,r})))$ and $| u_i | = \omega$.

Proof. By Lemma 1, it is sufficient to show that $u_i \Rightarrow \cup_{q \in \omega} Out_i(q)$. Now take any $A \in Out_i(p)$ $(p \in \omega)$. By Lemma 2, $p = Fs^{\delta_i}(t)$ for some $t \in \omega$. Since $R_i(p)$ is a bijection from $\{0, 1, \ldots, \#Out_i(p) - 1\}$ to $Out_i(p)$ from (4.1), and Fs^{γ_i} is a fair function in ω^ω, it follows from (4.2) that there exists t' such that $A = R_i(p)(Fs^{\gamma_i}(Ord(Fs^{\delta_i}, t'))) = u_i(t')$, where $p = Fs^{\delta_i}(t')$. On the other hand, when for $1 \leq r \leq n_i$ $v_{i,r} \Rightarrow At(Pr(B_{i,r}))$, it follows from (4.3) that given $p \in \omega$, $u_i(p) \in Out_i(q) \cup \{\tau\}$ for some $q \leq p$. This completes the proof of $u_i \Rightarrow \cup_{q \in \omega} Out_i(q)$ and $| u_i | = \omega$. Q.E.D.

4.2 General Fair Merge Function

We have a representation for the union of sets of atoms with a predicate symbol by means of general fair merge functions of sequence variables. The definition of general fair merge functions is given as follows.

Definition 5. Let $\alpha = (\alpha_1, \ldots, \alpha_{n-1})$ $(n \geq 2)$, where each α_i is a function in ω^ω such that $\#\{k \mid \alpha_i(k) = 0\} = \#\{k \mid \alpha_i(k) = 1\} = \omega$. Then $FM^n_\alpha : (F^\infty)^n \to F^\infty$ is defined recursively as follows:

(1) $FM^2_\alpha(u_1, u_2) = FM_\alpha(u_1, u_2)$.

(2) $FM^n_\alpha(u_1, \ldots, u_n) = FM^2_{\alpha_1}(u_1, FM^{n-1}_{\alpha'}(u_2, \ldots, u_n))$, where $n > 2$ and $\alpha' = (\alpha_2, \ldots, \alpha_{n-1})$.

FM^n_α is called a general fair merge function.

Lemma 3. Assume that $u_i \Rightarrow At_i(P)$, $| u_i | = \omega$ $(1 \leq i \leq n)$. Then, for a general fair merge function FM^n_α, $FM^n_\alpha(u_1, \ldots, u_n) \Rightarrow \cup_i At_i(P)$.

Proof. Let $v = FM^n_\alpha(u_1, \ldots, u_n)$. Then, it follows from the definition of general fair merge functions that for any $q \in \omega$, there exists u_i and $p \in \omega$ such that $v(q) = u_i(p)$. On the other hand, for any $1 \leq i \leq n$ and $p \in \omega$, there exists $q \in \omega$ such that $v(q) = u_i(p)$. These are sufficient to see that $v \Rightarrow \cup_i At_i(P)$. Q.E.D.

4.3 Recursion Equation Set

Now assume that $L = \{C_1, \ldots, C_k\}$ is a logic program and each C_i takes the form $A_i \to B_{i,1} \ldots B_{i,n_i}$. Suppose that $PRED(H_L) = \{P_1, \ldots, P_h\}$. That is, the set of predicate symbols in L is $\{P_1, \ldots, P_h\}$. Let $Pred(j) = \#\{A_i \mid Pr(A_i) = P_j\}$ for $1 \leq j \leq h$. $Pred(j)$ means the number of definite clauses whose heads have the predicate symbol P_j. Then let

$S_i = TR_L(C_i, Sem(L)), 1 \leq i \leq k$, and
$T_j = [Sem(L)]_{P_j} = \{A \in Sem(L) \mid Pr(A) = P_j\}$,
$1 \leq j \leq h$.

Note $Sem(L) = \cup_{C_i \in L} S_i$ and S_i denotes the set of all ground atoms to be infered from $Sem(L) \cup \{C_i\}$. T_j is the set of all ground atoms in $Sem(L)$, with the predicate symbol P_j.

It is assumed that

for $1 \leq j \leq h$,
$Pr(Head(C_{j_s})) = Pr(A_{j_s}) = P_j, 1 \leq s \leq Pred(j)$, and
for $1 \leq i \leq k$,
$Pr(B_{i,r}) = P_{i_r}, 1 \leq r \leq n_i$.

Using the above notations, we have:

Lemma 4.

(1) $T_j = \cup_{1 \leq s \leq Pred(j)} S_{j_s}$.

(2) $S_i = TR_L(C_i, \cup_{1 \leq r \leq n_i} T_{i_r})$.

Proof. (1) Since $S_{j_s} \subset \cup_{C_i \in L} S_i = Sem(L), \cup_{1 \leq s \leq Pred(j)} S_{j_s} \subset Sem(L)$. If $A \in \cup_{1 \leq s \leq Pred(j)} S_{j_s}$, then $Pr(A) = P_j$. Thus, $\cup_{1 \leq s \leq Pred(j)} S_{j_s} \subset T_j = [Sem(L)]_{P_j}$. On the other hand, $A \in T_j = [Sem(L)]_{P_j}$ implies that $A \in \cup_{C_i \in L} S_i$ and $Pr(A) = P_j$. That is, if $A \in T_j$ then $A \in \cup_{Pr(Head(C_i)) = P_j} S_i = \cup_{1 \leq s \leq Pred(j)} S_{j_s}$. This completes the proof.

(2) $TR_L(C_i, Sem(L)) = TR_L(C_i, \cup_{1 \leq r \leq n_i} [Sem(L)]_{P_{i_r}})$, by the definition of '$TR_L$'. It follows from $S_i = TR_L(C_i, Sem(L))$ that $S_i = TR_L(C_i, \cup_{1 \leq r \leq n_i} T_{i_r})$. Q.E.D.

Now we need $U_i, 1 \leq i \leq k$ and $V_j, 1 \leq j \leq h$ such that $U_i \Rightarrow S_i$ and $V_j \Rightarrow T_j$.

By Lemmas 3 and 4, for each j,

$FM_{\beta_j}^{Pred(j)}(U_{j_1},\ldots,U_{j_{Pred(j)}}) \Rightarrow T_j$

if $U_{j_s} \Rightarrow S_{j_s}, | U_{j_s} |= \omega, 1 \le s \le Pred(j)$.

If we define u_i by means of (4.3) for $v_{i,r} = V_{i_r}, 1 \le r \le n_i$, then it follows from Theorem 1 that $V_{i_r} \Rightarrow T_{i_r}, 1 \le r \le n_i$ implies $u_i \Rightarrow S_i = TR_L(C_i, \cup_{1\le r\le n_i} T_{i_r})$.

Therfore we have a set of recursion equations for $U_i, 1 \le i \le k$ and $V_j, 1 \le j \le h$:

(4.4) $V_j = g_j(U_{j_1},\ldots,U_{j_{Pred(j)}}), 1 \le j \le h,$
$U_i = f_i(V_{i_1},\ldots,V_{i_{n_i}}), 1 \le i \le k,$

where (1) g_j is a general fair merge function $FM_{\beta_j}^{Pred(j)}$, and (2) f_i is a function from $(D_L^\infty)^{n_i} \rightarrow D_L^\infty$, defined by (4.3).

The set of recursion equations by (4.4) is rewritten as

(4.5)
$(U_1,\ldots,U_k,V_1,\ldots,V_h) = f_L(U_1,\ldots,U_k,V_1,\ldots,V_h)$.

5 SEMANTICS OF LOGIC PROGRAM

In this section we assume the set of recursion equations constructed as in (4.4) and/or (4.5). First we see the (least) fixpoint of f_L in (4.5).

$f : (D_L^\infty)^m \rightarrow D_L^\infty$ is continuous if for any chain $\{w_0 \sqsubset w_1 \sqsubset \ldots\}$, $f(\sqcup\{w_i \mid i \in \omega\}) = \sqcup\{f(w_i) \mid i \in \omega\}$.

Lemma 5. In (4.4), $f_i, 1 \le i \le k$ and $g_j, 1 \le j \le h$ are continuous.

Proof. Note in (4.3) that $R_i(Fs^{\delta_i}(q))$ is a bijection from $\{0,1,2,\ldots,\#Out_i(Fs^{\delta_i}(q))-1\}$ to $Out_i(Fs^{\delta_i}(q))$, where $Out_i(Fs^{\delta_i}(q))$ depends on $v_{i_1}[p],\ldots,v_{i_{n_i}}[p]$ for $p \le Fs^{\delta_i}(q) \le q$. Note that $\tau \prec A$ for any $A \in H_L$. Thus, for any $p \in \omega$, $f_i(v_{i_1}[p],\ldots v_{i_{n_i}}[p]) \sqsubset u_i = f_i(v_{i_1},\ldots,v_{i_{n_i}})$. Therefore $\sqcup\{f_i(v_{i_1}[p],\ldots,v_{i_{n_i}}[p]) \mid p \in \omega\} \sqsubset f_i(v_{i_1},\ldots,v_{i_{n_i}}) = f_i(\sqcup\{ (v_{i_1}[p],\ldots,v_{i_{n_i}}[p]) \mid p \in \omega\})$. On the other hand, for any $q \in \omega$, $u_i[q] = f_i(v_{i_1},\ldots,v_{i_{n_i}})[q] \sqsubset f_i(v_{i_1}[p],\ldots,v_{i_{n_i}}[p])$ (for some $p \ge q$), since $u_i(q)$ is determined by $v_{i_1}[q],\ldots,v_{i_{n_i}}[q]$. Thus $f_i(v_{i_1},\ldots,v_{i_{n_i}}) = \sqcup\{u_i[q] \mid q \in \omega\} \sqsubset \sqcup\{f_i(v_{i_1}[p],\ldots, v_{i_{n_i}}[p]) \mid p \in \omega\}$.
This completes the proof for the continuity of f_i.

To prove the continuity of general fair merge functions, it is sufficient to show the continuity of fair merge functions, since a general fair merge function is composed of fair merge functions by Definition 5. Note a fair merge function FM_δ is defined by using *dmerge*: $FM_\delta(u,v) = dmerge(u,v,\delta)$. Since $dmerge(nil,v,\delta_1) = nil$ if $\delta_1(0) = 0$, and $dmerge(u,nil,\delta_2) = nil$ if $\delta_2(0) = 1$, for any chain $\{(u_0,v_0) \sqsubset (u_1,v_1) \sqsubset \ldots\}$, whose least upper bound is (u,v), $FM_\delta(u_i,v_i) \sqsubset FM_\delta(u,v)$. Thus $\sqcup\{FM_\delta(u_i,v_i) \mid i \in \omega\} \sqsubset FM_\delta(u,v) = FM_\delta(\sqcup\{(u_i,v_i) \mid i \in \omega\})$. On the other hand, because of the property of FM_δ, for any $p \in \omega$ there exists $i \in \omega$ such that $FM_\delta(u,v)[p] \sqsubset FM_\delta(u_i,v_i)$. Therefore, $FM_\delta(\sqcup\{(u_i,v_i) \mid i \in \omega\}) = FM_\delta(u,v) = \sqcup\{FM_\delta(u,v)[p] \mid p \in \omega\} \sqsubset \sqcup\{FM_\delta(u_i,v_i) \mid i \in \omega\}$. This completes the proof for the continuity of FM_δ. Q.E.D.

Thus, there exists a (least) fixpoint of the recursion equation set (4.5). Indeed, it is $\sqcup_p\{f_L^p(nil,\ldots,nil) \mid p \in \omega\}$, where $f_L^0(nil,\ldots,nil) = (nil,\ldots,nil)$ and $f_L^p(nil,\ldots,nil) = f_L(f_L^{p-1}(nil,\ldots,nil))$ for $p \ge 1$.

From now on, let $f_L^p(nil,\ldots,nil) = (U_1^p,\ldots,U_k^p, V_1^p,\ldots, V_h^p)$. It follows from Theorem 1 and (4.4) that $| U_i^p | = \omega$ $(1 \le i \le k)$ if $p \ge 1$ and $| V_j^p | = \omega$ $(1 \le j \le h)$ if $p \ge 2$.

Now we have the primary theorem, which states that the recursion equation set expresses the denotation of a given logic program.

Theorem 2. Let $(U_1^f,\ldots,U_k^f, V_1^f,\ldots,V_h^f)$ be a (least) fixpoint of the recurtion equation set (4.5). Then $\{V_1^f,\ldots,V_h^f\} \Rightarrow Sem(L)$.

Proof. For $Trans_L(I) = \cup_{C_i \in L} TR_L(C_i, L)$, we show by induction on p that $V_j^{p+1} \Rightarrow [Trans_L^p(\Phi)]_{P_j}, 1 \le j \le h$, where Φ is the empty set, and $Trans_L^p(\Phi)$ is defined recursively: $Trans_L^0(\Phi) = \Phi$; $Trans_L^p(\Phi) = Trans_L(Trans_L^{p-1}(\Phi))$ for $p \ge 1$.

(i) In case $p = 0$:
$V_j^1 = nil$ $(1 \le j \le h)$, since $U_i^0 = nil$ $(1 \le i \le k)$. On the other hand, $[Trans_L^0(\Phi)]_{P_j} = \Phi$ (the empty set). By the meaning of '$\Rightarrow$', $nil \Rightarrow \Phi$. Thus this step holds.

(ii) Assume that $V_j^{p+1} \Rightarrow [Trans_L^p(\Phi)]_{P_j}, 1 \le j \le h$, for $p \le p'$:
It is easy to see $Trans_L^{p'}(\Phi) = \cup_{1\le i\le k}[Trans_L^{p'}(\Phi)]_{P_i}$. Since $\{V_1^{p'+1},\ldots,V_h^{p'+1}\} \Rightarrow Trans_L^{p'}(\Phi)$, it follows from Theorem 1 that $U_i^{p'+1} \Rightarrow TR_L(C_i, Trans_L^{p'}(\Phi)), 1 \le i \le k$. Because $| U_i^p | = \omega$ $(1 \le i \le k)$ for $p \ge 1$ and $V_j^{p'+2} = g_j(U_{j_1}^{p'+1}, \ldots, U_{j_{Pred(j)}}^{p'+1})$ for $1 \le j \le h$, we can see that $V_j^{p'+2} \Rightarrow \cup_{Pr(Head(C_i))=P_j} TR_L(C_i, Trans_L^{p'}(\Phi)) = [Trans_L^{p'+1}(\Phi)]_{P_j}$.
This completes the induction step.

Now assume that $A \in [Sem(L)]_{P_j}$ for some j. Then there exists $m \in \omega$ such that $A \in [Trans_L^m(\Phi)]_{P_j}$. This is because $Sem(L) = \cup_{m\ge1} Trans_L^m(\Phi)$ (Apt et al. 1982 and van Emden et al. 1976). Since $V_j^{m+1} \Rightarrow [Trans_L^m(\Phi)]_{P_j}$, there exists $q \in \omega$ such that $V_j^{m+1}(q) = A$. It follows from $V_j^{m+1} \sqsubset V_j^f$ that $V_j^f(q) = A$. On the other hand, we see that $V_j^f(q)$ is defined for any $q \in \omega$. Then there exists $m+1 \in \omega$ such that $V_j^{m+1}(q) = V_j^f(q)$. It follows from

$V_j^{m+1} \Rightarrow [Trans_L^m(\Phi)]_{P_j}$ that $V_j^{m+1}(g)$ is in $[Trans_L^m(\Phi)]_{P_j} \cup \{\tau\} \subset [\cup_{m\geq 1} Trans_L^m(\Phi)]_{P_j} \cup \{\tau\}$. This means that $V_j^f \Rightarrow [\cup_{m\geq 1} Trans_L^m(\Phi)]_{P_j}$. Finally we come up with the conclusion tjat $\{V_1^f, \ldots, V_h^f\} \Rightarrow \cup_{m\geq 1} Trans_L^m(\Phi) = Sem(L)$. Q.E.D.

To delete τ in a sequence from $D_L = H_L \cup \{\tau\}$ and to get a sequence from H_L, the following function is useful.
E: $D_L^\infty \to H_L^\infty$ is the function satisfying

$$E(u)(p) = (eq(u(0), \tau) \to E(next(u))(p), E(next(u))(p \dot{-} 1))$$
for $u \in D_L^\infty$ and $p \in \omega$.

Since $\{V_1^f, \ldots, V_h^f\} \Rightarrow Sem(L)$, $(E(V_1^f), \ldots, E(V_h^f))$ can be regarded as a semantics of L.

6 CONCLUDING REMARKS

In this paper, a semantics of a logic program was defined over a sequence domain. It is a least fixpoint of a recursion equation set constructed from a given logic program. To have the recursion equation set, we begin with the interpretation of the inference caused by each definite clause as a relation among sequence variables. The relation is reduced to a function by introducing oracles based on fair functions in ω^ω. Also fair merge functions are made use of , to realize a sequence variable whose denotation is the union of the denotations of other sequence variables. The newly defined semantics represents the minimal Herbrand model of the original logic program in terms of sequence variables based on the Herbrand base. Thus, the recursion equation set is a dataflow program computing the original logic program. This is the primary aspect of the present paper.

The method of transforming the recursion equation set into a functional program is worth while studying, in order to give an insight of establishing the way how the transformation of logic programs into functional programs can be performed through recursion equation sets. It is left for the next work to construct a sequence domain based on substitutions for semantics of a given logic program, combining the result of this paper with the method in (Yamasaki et al. 1987).

ACKNOWLEDGEMENT

The author is grateful to Prof. D.M.R. Park and Dr. S.G. Matthews for their comments on dataflow during the author's visits to University of Warwick and University of Edinburgh, U.K.

REFERENCES

1. Nait.M.A.Abdallah, On the interpretation of infinite computations in logic programming, Lecture Notes in Computer Science 172 (1984) 358-370.

2. K.P.Apt and M.H.van Emden, Contributions to the theory of logic programming, J. ACM 29 (1982) 841-864.

3. E.A.Ashcroft and W.W.Wadge, Lucid–A formal system for writing and proving programs, SIAM J. Computing 5 (1976) 336-354.

4. M.H.van Emden and R.A.Kowalski, The semantics of predicate logic as a programming language, J. ACM 23 (1976) 733-742.

5. M.Fitting, A deterministic Prolog fixpoint semantics, J. of Logic Programming 2 (1985) 111-118.

6. G.Frauden, Logic programming and substitutions, Lecture Notes in Computer Science 199 (1985) 146-158.

7. G.Kahn, The semantics of a simple language for parallel programming, Proc. IFIP 74 (1974) 471-475.

8. J.L.Lassez and M.J.Maher, Closures and fairness in the semantics of programming logic, Theoretical Computer Science 29 (1984) 167-184.

9. J.L.Lassez and M.J.Maher, Maximal fixpoints of logic programs, Theoretical Computer Science 39 (1985) 15-25.

10. D.Park, The 'fairness' problem and nondeterministic computing networks, in: de Bakker and van Leeuwen, eds., Foundations of Computer Science IV (Mathematisch Centrum, Amsterdam, 1983) 133-161.

11. W.W.Wadge, An extensional treatment of dataflow deadlock, Lecture Notes in Computer Science 70 (1979) 285-299.

12. S.Yamasaki et al., A fixpoint semantics of Horn sentences based on substitution sets, Theoretical Computer Science 51 (1987) 309-324.