ソフトウェア基礎論　３５－３
(1990. 5. 24)

論理プログラムに対する等価な置換集合の操作機構としてのデータフロー

山崎　進　（岡山大学工学部）

Dataflow for Logic Program as Manipulator of Equivalent Substitutions Sets

Susumu Yamasaki

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

Abstract

This paper shows a method of constructing a dataflow, which denotes the deductions of a logic program, by means of a sequence domain based on equivalence classes of substitutions. The dataflow involves fair merge functions to represent unions of atom subsets over a sequence domain, as well as functions as manipulations of unifiers for the deductions of clauses. A continuous functional is associated with the dataflow on condition that the dataflow completely and soundly denotes the atom generation in terms of equivalent substitutions sets. Its least fixpoint is interpreted as denoting the whole atom generation based on manipulations of equivalent substitutions sets.

1. Introduction

Many researches as to the semantics of logic programs have been made since van Emden and Kowalski defined a semantics for finite computations/deductions of logic programs [1,2,4,5,7, 8,9,10,11,12]. Among them there are distinguishable approaches to grasp the semantics in the sense that they are concerned with sequence domains. M.Fitting defined a deterministic Prolog fixpoint semantics, which represents all the answer sequences with the goals for a given logic program. M.Baudinet proposed a method of transforming a logic program into a recursive program which represents manipulations of substitutions caused by the computation of the logic program and denotes a fixpoint semantics of the given logic program [4,7]. Her approach is regarded as an advanced version of defining Fitting's semantics in that its sequence domain is not based on the Herbrand base but on a substitution set. Different from those approaches to define semantics of logic programs over sequence domains, this paper deals with semantics of a logic program, based on dataflow construction approach.

The semantics proposed in this paper reflects the behaviour of dataflow for a logic program as a manipulator of substitutions. It will be defined over a sequence domain based on the quotient set for a substitution set. The quotient set is obtained on the basis of an equivalence relation on the substitution set. The reason to adopt the quotient set is to eliminate nondeterminism occurring in selecting one substitution among equivalent substitutions in accordance with a resolution deduction. We will construct a dataflow to realize manipulations on the quotient set for unit deduction. At the same time, it defines a continuous functional based on a sequence domain. The sequence domain is constructed as the set of all finite and infinite sequences from the quotient set with a symbol representing time delay or the hiaton in [13]. The dataflow for a logic program will be defined as a recursion equation set as to sequence variables and a continuous functional is to be associated with the recursion equation set. We will see that any item (equivalent substitutions set) is denoted in some sequence iff the original logic program generates a corresponding atom subset with the item. Therefore the least fixpoint of the functional is closely related with a semantics for the given logic program to represent all the items by which deductive atom subsets are denoted.

2. Basic Notations and Fundamentals

In this paper a logic program means a set of definite clauses. A definite clause takes the form such as $A \leftarrow B_1 \ldots B_n$ $(n \geq 0)$, where $A, B_1, \ldots$, and B_n are atoms. An atom is an expression of the form $P(t_1, \ldots, t_m)$, where P is a predicate symbol and

t_i are terms. A term is recursively defined as: (i) a variable is a term, and (ii) $f(t_1, \ldots, t_k)$ $(k \geq 0)$ is a term if f is a k-place symbol and t_j are terms.

In this section we have technical terms and fundamentals concerning substitutions for unifications in the deductions of logic programs.

Let $Term$ be a set of terms and Var a set of variables. Note $Var \subset Term$. A substitution is a function from Var to $Term$. For a substitution θ, Var_θ denotes the set $\{ x \in Var \mid \theta(x) \neq x \}$, that is, the domain of θ.

For the treatment with the substitution as unification of variables, a substitution θ is assumed to satisfy the condition that $Var_\theta \cap \{ y \mid y$ occurs in some $\theta(z)$ for $z \in Var_\theta \}$ is empty. The condition guarantees that the terms substituted for the variables do not involve any corresponding variables on which a given substitution operates. That is, the condition means the idempotence of the substitution. From now on, Sub means the set of all idempotent substitutions. The substitution θ is especially denoted by ε if Var_θ is empty. That is, $\varepsilon(x) = x$ for any $x \in Var$. The effect of a substitution for the term or atom, and the composition of substitutions are defined by the following formalism.

Let $Exp = Term \cup Atom$, where $Atom$ is the set of atoms. For $\theta \in Sub$ and $E \in Exp$, $E\theta$ is recursively defined:

$$E\theta = \begin{cases} \theta(x) & \text{if } E = x \text{ for } x \in Var, \\ f(t_1\theta, \ldots, t_n\theta) & \text{if } E = f(t_1, \ldots, t_n) \in Term, \\ P(t_1\theta, \ldots, t_m\theta) & \text{if } E = P(t_1, \ldots, t_m) \in Atom. \end{cases}$$

For $\theta, \varphi \in Sub$, the composition of θ and φ, denoted by $\varphi\theta$, is defined: $\varphi\theta(x) = \theta(x)\varphi$ for $x \in Var$, as long as $\varphi\theta \in Sub$. Also we see that $(E\theta)\varphi = E(\varphi\theta)$ for $E \in Exp$ and $\theta, \varphi \in Sub$, if $\varphi\theta \in Sub$.

Definition 2.1. We say that θ is more general than φ, in another word, φ is less general than θ for $\theta, \varphi, \psi \in Sub$ if $\varphi = \psi\theta$. By $\varphi < \theta$ it is meant that θ is more general than φ.

Note that $<$ is not a partial order.

Definition 2.2. A relation $\sim$ on Sub is defined: $\theta \sim \varphi$ iff $\varphi < \theta$ and $\theta < \varphi$.

It is seen that $\sim$ is an equivalence relation.

$Sub/\sim$ denotes the quotient set of Sub by means of the equivalence relation $\sim$, which is refered to just as the quotient set from now on. Any element in SUB is regarded as a subset of Sub.

To restrict the domain of the substitution to some appropriate set, the following definition is exploited.

For a substitution σ and a set of atoms $\{A_1, \ldots, A_m\}$ $(m \geq 1)$, a restriction of σ with respect to $\{A_1, \ldots, A_m\}$, that is, $[\sigma]_{\{A_1,\ldots,A_m\}}$: $Var \to Term$ is defined as follows.

$$[\sigma]_{\{A_1,\ldots,A_m\}}(x) = \begin{cases} \sigma(x) & \text{if } x \text{ occurs in either } A_1, \ldots, \text{ or } A_m, \\ x & \text{otherwise,} \end{cases}$$

for $x \in Var$.

It follows that $[\varphi]_{\{A_1,\ldots,A_m\}} \sim [\theta]_{\{A_1,\ldots,A_m\}}$ if $\varphi \sim \theta$. The following is an extension of the restriction of a substitution.

Definition 2.3. Assume that $\Theta \subset Sub$, and $\{A_1, \ldots, A_m\} \subset Atom$ $(m \geq 1)$. Then we define

$$[\Theta]_{\{A_1,\ldots,A_m\}} = \begin{cases} \{[\theta]_{\{A_1,\ldots,A_m\}} \mid \theta \in \Theta\} & \text{if } \Theta \text{ is nonempty,} \\ \text{empty} & \text{if } \Theta \text{ is empty.} \end{cases}$$

As a restriction of the equivalence class (in SUB), which is regarded as a subset of Sub, we have the following definition.

Definition 2.4. Let $\Phi \in SUB$ and $\{A_1, \ldots, A_m\} \subset Atom$. A restriction of Φ with respect to $\{A_1, \ldots, A_m\}$ is $[[\Phi]]_{\{A_1,\ldots,A_m\}} = \Psi$, where $\Psi \in SUB$ such that $[\Phi]_{\{A_1,\ldots,A_m\}} \subset \Psi$.

3. Description of Deduced Atoms by Means of Quotient Set SUB

In this section we have a sketch on the deductions of atoms, that is, the computations for a given logic program, in terms of manipulations on SUB introduced in the previous section.

Assume that a set of clauses $\{C \leftarrow D_1 \ldots D_m,\ D_1'\varphi_1 \leftarrow, \ldots, D_m'\varphi_m \leftarrow\}$ is given, where $\varphi_1, \ldots, \varphi_m \in Sub$. Then an atom $C\theta \leftarrow$ may be derivable from the set, and θ can be calculated by means of $\varphi_1, \ldots, \varphi_m$ with the unifications between D_i and D_i' $(1 \leq i \leq m)$. We will see how θ can be decided. On the assumption that $C\theta \leftarrow$ is deductive from the above set, we will see the property that $C\theta' \leftarrow$ is derivable from the set $\{C \leftarrow D_1 \ldots D_m,\ D_1'\varphi_1' \leftarrow, \ldots, D_m'\varphi_m' \leftarrow\}$ if $\varphi_i' \sim \varphi_i$ $(1 \leq i \leq m)$ and $\theta' \sim \theta$. This leads to our observation that the set SUB might be taken into account, in order to represent the deduced atoms.

Firstly we need some definitions concerning unifications, and investigate related properties.

Definition 3.1. Let A be a nonempty subset of $Atom$. We define $unif(A) = \{\theta \in Sub \mid \forall A_1, A_2 \in A\colon A_1\theta = A_2\theta\}$. Also we define $mgu(A) = \{\sigma \in Sub \mid \sigma \in unif(A)$ and σ is most general $\}$. When A is empty, we regard $unif(A)$ as Sub and $mgu(A)$ as $\{ \varepsilon \}$, respectively.

Note that each substitution in $mgu(A)$ is a most general unifier of A in the usual sense. The next lemma is well known. Also for the lemma to state there exists an idempotent unifier which is most general for a unifiable set, see [6].

Lemma 3.2. Assume that $mgu(A)$ is not empty for $A \subset Atom$. If $\theta, \varphi \in mgu(A)$, then $\theta \sim \varphi$, that is, $mgu(A)/\sim$ is a singleton.

Definition 3.3. Let A be a subset of $Atom$. We define $MGU(A) = mgu(A)/\sim$.

Next we observe the deduction for definite clauses as manipulations on Sub/SUB.

Unit resolution is an inference to derive $C\theta \leftarrow D_1\theta \ldots D_{i-1}\theta\ D_{i+1}\theta \ldots D_m\theta$ from two clauses $C \leftarrow D_1 \ldots D_m$ and $E \leftarrow$, where $\theta \in mgu(\{D_i, E\})$. A unit deduction from a set S of definite clauses is a sequence of definite clauses $G_1, \ldots, G_n$, where each G_i is either in S or infered by unit resolution from some G_j and G_k $(j, k < i)$.

The unit deduction is thought of as one of ways for the generations of atoms from a logic program, and is interpreted as computing mechanism. We will see in the following that the unit deduction might be realized by some manipulations on Sub/SUB.

Definition 3.4. $consis : 2^{Sub} \rightarrow 2^{Sub}$ and $comb : 2^{Sub} \rightarrow 2^{Sub}$ are defined:

$$consis(\{\theta_1, \ldots, \theta_n\}) =$$
$$\{\theta \mid \exists\sigma_1, \ldots, \sigma_n\colon \sigma_1\theta_1 = \ldots = \sigma_n\theta_n = \theta\ \},$$
$$comb(\{\theta_1, \ldots, \theta_n\}) =$$
$$\{\rho \mid \rho \in consis(\{\theta_1, \ldots, \theta_n\}) \text{ and } \rho \text{ is most general}\}.$$

We define $consis(\ ) = Sub$ and $comb(\ ) = \{\varepsilon\}$ for the empty set.

Example 3.5. Let θ_1 and θ_2 be defined by:

$$\theta_1(x) = f(y),\ Var_{\theta_1} = \{x\};$$
$$\theta_2(y) = g(z),\ Var_{\theta_2} = \{y\}.$$

Then $\theta \in comb(\{\theta_1, \theta_2\})$, where $\theta(x) = f(g(z))$ and $\theta(y) = g(z)$ for $Var_\theta = \{x, y\}$.

As is the case for $mgu(A)$, $comb(\{\theta_1, \ldots, \theta_n\})$ is either empty or contains substitutions which belong to an equivalence class of SUB. That is, we have the following.

Lemma 3.6. $comb(\{\theta_1, \ldots, \theta_n\})/\sim$ consists of one equivalent class for any $\theta_1, \ldots, \theta_n$.

It is easy to see the following lemma from the definition that the $comb$ function causes the same effect for the equivalent substitutions in the sense '$\sim$' as the initially considered substitutions.

Lemma 3.7. $comb(\{\theta_1, \ldots, \theta_n\}) = comb(\{\theta'_1, \ldots, \theta'_n\})$ if $\theta_i \sim \theta'_i$, $1 \leq i \leq n$.

Based on Lemmas 3.6 and 3.7, we have the $COMB$ functions as follows:

Definition 3.8. $COMB$: $2^{SUB} \rightarrow SUB$ is defined by $COMB(\{\Theta_1, \ldots, \Theta_n\}) = comb(\{\theta_1, \ldots, \theta_n\})/\sim$ for some $\theta_i \in \Theta_i$ $(1 \leq i \leq n)$.

Note $COMB(\{\Theta_1, \ldots, \Theta_n\})$ is an equivalence class (in SUB), and $\theta \in COMB(\{\Theta_1, \ldots, \Theta_n\})$ means $\theta \in comb(\{\theta_1, \ldots, \theta_n\})$ for some $\theta_i \in \Theta_i$ $(1 \leq i \leq n)$, because we regard any class in SUB as a subset of Sub. Before investigating the method of describing the unit deduction in terms of the $COMB$ function, we have the following representation of the set of atoms with attached substitutions.

Definition 3.9. For an atom A and $\Theta \in SUB$, we define $A\Theta = \{A\theta \leftarrow \mid \theta \in \Theta\}$.

At the same time, for each predicate symbol, we adopt a standard form of the atom involving the predicate symbol as follows:

Definition 3.10. For each predicate symbol $P_i \in PRED(L)$, a tuple of variables $\bar{x}_i$ is attached to P_i such that

(i) $\bar{x}_i$ contains no variable occurring in L,
(ii) $P_i(\bar{x}_i)$ is an atom,
(iii) $\bar{x}_i$ and $\bar{x}_j$ have no common variable if $i \neq j$,

where $PRED(L)$ means the set of all predicate symbols appearing in L.
$P_i(\bar{x}_i)$ is refered to as a standard atom (with P_i). For an atom D, $Stand(D)$ means the standard atom with the predicate symbol involved in D.

Now we examine the unit deduction by means of the $comb/COMB$ functions. First of all, a basic lemma for the relation between the mgu and $comb$ functions is given.

Lemma 3.11. Let C and D be atoms, and $\psi \in Sub$, $\theta \in mgu(\{C, D\})$. Then $mgu(\{C, D\psi\}) = [comb(\{\theta, [\psi]_D\})]_{\{C,D\psi\}}$.

Proof. Assume that $\alpha \in mgu(\{C, D\psi\})$. Since $C\theta = D\theta$ and $C\alpha = (D\psi)\alpha$, there exists β such that $\alpha\psi = \beta\theta$. As α is most general, it is conceded that $\alpha \in [comb(\{\theta,\ [\psi]_{\{D\}}\})]_{\{C,D\psi\}}$. Thus $mgu(\{C, D\psi\}) \subset [comb(\{\theta,\ [\psi]_{\{D\}}\})]_{\{C,D\psi\}}$. On the other hand, assume $\alpha \in [comb(\{\theta,\ [\psi]_{\{D\}}\ \})]_{\{C,D\psi\}}$. Then $\alpha = \beta\theta = \gamma[\psi]_{\{D\}}$ for some $\beta, \gamma \in Sub$. Taking the idempotence of γ and the domain of α into account, it is concluded that $\gamma = \beta\theta_1$, $[\psi]_{\{D\}} = \theta_2$, $\theta = \theta_2\theta_1$ and $C\theta_1 = (D\theta_1)\theta_2$ for some adequate θ_1, θ_2. Therefore $C(\beta\theta_1) = (C\theta_1)\beta = ((D\theta_2)\theta_1)\beta = (D\psi)(\beta\theta_1)$. That is, γ is a unifier of C and $D\psi$. γ should be most general, as $\gamma \in [comb(\{\theta, [\psi]_{\{D\}}\})]_{\{C,D\psi\}}$. This completes the proof.

Making use of Lemma 3.11, we have the folllowing theorem, which expresses the unit deduction only in terms of equivalent classes of substitutions, which are interpreted as attached to atoms.

Theorem 3.12. Assume a definite clause $C \leftarrow D_1 \ldots D_m$ in which the predicate symbol of D_i is P_i for $1 \leq i \leq m$. Also let $\Psi_i = [[\Theta_i]]_{\{P_i(\bar{x}_i)\}} \in SUB$, $1 \leq i \leq m$. Then $C' \leftarrow$ is derivable by unit deduction from $\cup_{1 \leq i \leq m} P_i(\bar{x}_i)\Psi_i \cup \{C \leftarrow D_1 \ldots D_m\}$

iff $C' \leftarrow\in C\Theta$ for $\Theta = [[COMB(\{\Delta_1, \Psi_1, \ldots, \Delta_m, \Psi_m\})]]_{\{C\}}$ and $\Delta_i = MGU(\{D_i, P_i(\bar{x}_i)\})$, $1 \leq i \leq m$.

Proof. We can have a proof by induction on m. In case $m = 0$, the theorem holds, since $C\varepsilon \rightarrow$ is derivable and $\Theta = \{\varepsilon\}/\sim$.
Induction Step: Assume the theorem holds for $m \leq k-1$. Let $m = k$ and $\Theta = [[COMB(\{\Delta_1, \Psi_1, \ldots, \Delta_k, \Psi_k\})]]_{\{C\}}$. Assume that $C\varphi \leftarrow \in C\Theta$. Then $\varphi \in [comb(\{\delta_k, \psi_k, \varphi_k\})]_{\{C\}}$ for some $\delta_k \in \Delta_k$, $\psi_k \in \Psi_k$ and $\varphi_k \in [[COMB(\{\Delta_1, \Psi_1, \ldots, \Delta_{k-1}, \Psi_{k-1}\})]]_{\{C, D_k\}}$. By the induction hypothesis, $C\varphi_k \leftarrow D_k\varphi_k$ is derivable. Applying Lemma 3.11, we can conclude that $\varphi \in [mgu(\{D_k\varphi_k, P_k(\bar{x}_k)\ \psi_k\})]_{\{C\}}$. Thus $C\varphi \leftarrow$ is derivable from $\cup_{1\leq i\leq k} P_i(\bar{x}_i) \cup \{C \leftarrow D_1 \ldots D_k\}$. On the other hand, assume $C' \leftarrow$ is derivable from $\cup_{1\leq i\leq k} P_i(\bar{x}_i)\Psi_i \cup \{C \leftarrow D_1 \ldots D_k\}$. By the induction hypothesis, $C' \leftarrow$ is derivable from $P_k(\bar{x}_k)$ $\Psi_k \cup \{C\varphi_k \leftarrow D_k\varphi_k\}$, where $\varphi_k \in [[COMB(\{\Delta_1, \Psi_1, \ldots, \Delta_{k-1}, \Psi_{k-1}\})]]_{\{C, D_k\}}$. Therefore $C\varphi \leftarrow = C' \leftarrow$ is derivable, where $\varphi = [\varphi]_{\{C\}}$ such that $\varphi \in [mgu(\{D_k\varphi_k, P_k(\bar{x}_k)\psi_k\})]_{\{C\}}$ for $\psi_k \in \Psi_k$. By applying Lemma 3.11, we conclude that $\varphi \in [comb(\{\delta_k, \psi_k, \varphi_k\})]_{\{C\}}$. Thus $\varphi \in [[COMB(\{\Delta_1, \Psi_1, \ldots, \Delta_k, \Psi_k\})]]_{\{C\}}$. This completes the proof.

The next theorem is necessary to form $P_i(\bar{x}_i)\Psi$, which is equivalent to the atom set $C\Theta$ such that the standard atom of C is $P_i(\bar{x}_i)$.

Theorem 3.13. Assume that $C\Theta$ is derivable for $\Theta = [[\Psi]]_{\{C\}} \in SUB$. Also assume that $P(\bar{x})$ is the standard atom of C. For $\Psi = [[COMB(\{\Gamma, \Theta\})]]_{\{P(\bar{x})\}}$, $P(\bar{x})\Psi = C\Theta$, where $\Gamma = MGU(\{C, P(\bar{x})\})$.

Proof. For $\theta \in \Theta$, $mgu(\{C\theta, P(\bar{x})\}) = [comb(\{\gamma, \theta\})]_{\{C\theta, P(\bar{x})\}}$ by Lemma 3.11, where $\gamma \in \Gamma$. Note that for any $\varphi \in [comb(\{\gamma, \theta\})]_{\{C\theta, P(\bar{x})\}}$, there exists $\psi \in [comb(\{\gamma, \theta\})]_{\{P(\bar{x})\}}$ such that $P(\bar{x})\psi = P(\bar{x})\varphi$. Thus we see the lemma holds.

4. Dataflow for Deductions of Logic Programs

In this section, we have a dataflow to realize the deductions for a given logic program L.

We take the translations of substitutions based on Theorems 3.12 and 3.13 into account. As shown in Theorem 3.12, the clause $C \leftarrow D_1 \ldots D_m$ can be interpreted as a node emitting $\Theta = [[COMB(\{\Delta_1, \Psi_1, \ldots, \Delta_m, \Psi_m\})]]_{\{C\}}$ for $\{\Psi_1, \ldots, \Psi_m\}$ (regarded as an input set) with a determinate set $\{\Delta_1, \ldots, \Delta_m\}$, where $MGU(\{D_i, Stand(D_i)\}) = \Delta_i$, $1 \leq i \leq m$, and $Stand(D_i)\Psi_i$, $1 \leq i \leq m$ are assumed to be generative. (Note that $Stand(D_i)$ is a standard atom for D_i.) The node for a clause can be extended to a manipulator of input sequences by providing an output item for each tuple of input items. Each item emitted by such a node is transformable into another by means of the relationship between the head of the clause and its standard atom. This transformation is based on Theorem 3.13. For the transformation, there may be more than one head of clauses with the same predicate symbol and thus the same standard atom. Thus transformed items should be gathered as one sequence. The device of the merge in [13] is most adequate to gather sequences and form a sequence. Fairness in the merge is necessary for any item to be transfered to other nodes as an input.

To deal with the sequences, we have a base domain $DOM = SUB \cup \{\tau\}$. τ is a special symbol not in SUB, to denote a time delay or a pause occurring in a sequence. It is exploited to manage the merge as continuous.

DOM^∞ denotes the set of partial functions from the set of natural numbers (denoted by ω) to DOM such that if $u \in DOM^\infty$ and $u(p)$ ($p \in \omega$) is defined then $u(q)$ is always defined for $q \leq p$. That is, DOM^∞ is regarded as the set of all finite and infinite sequences from DOM. nil is the function such that $nil(p)$ is undefined for any $p \in \omega$. Note nil is thought of as the empty sequence.

Now assume a logic program $L = \{Cl_1, \ldots, Cl_k\}$, where Cl_i is $C_i \leftarrow D_{i,1} \ldots D_{i,n_i}$ ($n \geq 0$) for atoms C_i and $D_{i,j}$ ($1 \leq j \leq n_i$). Also let $PRED(L) = \{P_1, \ldots, P_h\}$ and $\{P_1(\bar{x}_1), \ldots, P_h(\bar{x}_h)\}$ be a set of standard atoms. $Stand(D_{i,j}) = P_{i_j}(\bar{x}_{i_j})$ is assumed for $1 \leq i \leq k$ and $1 \leq j \leq n_i$.

To equip sequence variables for the representations of outputs emitted by the nodes for clauses, we assume $\{u_1, \ldots, u_k\}$ in accordance with $\{Cl_1, \ldots, Cl_k\}$, where each u_i denotes a sequence in DOM^∞. Also we prepare for a set of sequence variables $\{v_1, \ldots, v_h\}$ to express the sequences whose items are connected to standard atoms, where each v_j denotes a sequence in DOM^∞.

To identify an input tuple with a natural number, we adopt a bijection $I_m : \omega \rightarrow \omega^m$, and we define a projection $J_{m,i} : \omega^m \rightarrow \omega$ by $J_{m,i}(p_1, \ldots, p_m) = p_i$. $J_{m,i} \circ I_m$, that is, the composition is denoted by $I_{m,i}$.

Then, based on Theorem 3.12, we define

$$(4.1)\ u_i(p) = \begin{cases} [[COMB(\{\ \Delta_{i,1}, v_{i_1}(I_{n_i,1}(p)), \ldots, \\ \qquad \Delta_{i,n_i}, v_{i_{n_i}}(I_{n_i,i_{n_i}}(p))\ \ \})]]_{\{C_i\}}, \\ \qquad \text{if the right-hand side is defined,} \\ \tau \qquad \text{otherwise,} \end{cases}$$

for $1 \leq i \leq k$, where $\Delta_{i,j} = MGU(\{D_{i,j}, Stand(D_{i,j})\})$ for $1 \leq j \leq n_i$.

Next we have a representation of a set of clauses, whose heads have the same standard atom. For $1 \leq j \leq h$, we define $Pred(j) = \{i \mid C_i \leftarrow D_{i,1} \ldots D_{i,n_i} \in L$ and $P_j(\bar{x}_j) = Stand(C_i)\}$. The cardinal number of $Pred(j)$ is denoted by $NO(j)$. To gather the sequences emitted by the nodes for clauses, we need $fairmerge^p$: $(DOM^\infty)^p \rightarrow DOM^\infty$, which denotes one of functions interleaving input sequences without neglecting any part of any input. Based on Theorem 3.13, we define

$$(4.2)\ v_j = fairmerge^{NO(j)}(w_{j_1}, \ldots, w_{j_{NO(j)}}),$$

for $1 \leq j \leq h$, where

(i) $Pred(j) = \{j_1, \ldots, j_{NO(j)}\}$,
(ii) w_{j_i} denotes a sequence in DOM^∞, and is defined by

$$w_{j_i}(q) = \begin{cases} [[COMB(\{\Gamma_{j_i}, u_{j_i}(q)\})]]_{\{P_{j_i}(\bar{x}_{j_i})\}} \\ \qquad \text{if the right-hand side is defined,} \\ \tau \qquad \text{otherwise,} \end{cases}$$

for $\Gamma_{j_i} = MGU(\{C_{j_i}, P_{j_i}(\bar{x}_{j_i})\})$.

(4.1) as well as (4.2) is regarded as defining a dataflow for a logic program L.

5. Semantics of Logic Programs over Sequence Domains

In this section, we will see that (4.1) and (4.2) are complete and sound in representing the substitutions with which the atoms are deduced from a given logic program.

First we have a semantics for (4.1) and (4.2), which is given by introducing a partial order on $(DOM)^\infty$ and by the fixpoint approach.

Definition 5.1. A partial order $\prec$ on DOM is defined by: $\tau \prec \Theta$ and $\Theta \prec \Theta$ for any $\Theta \in SUB$. A partial order $\sqsubset$ on DOM^∞ is defined by: $u \sqsubset v$ for $u, v \in DOM^\infty$ iff $u(p) \prec v(p)$ for any $p \in \omega$ whenever $u(p)$ is defined.

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

$$(u_1, \ldots, u_m) \sqsubset (v_1, \ldots, v_m) \text{ iff } u_i \sqsubset v_i \text{ for } 1 \leq i \leq m.$$

The least upper bound of $T \subset (DOM^\infty)^m$ is denoted by $\sqcup T$. Note that $\sqsubset$ is sequentially complete in the sense that any $\{w^0 \sqsubset w^1 \sqsubset \ldots\}$ has a least upper bound.

(4.1) shows that the finite part of u_i depends on only finite parts of inputs. Also (4.2) consists of mappings from finite parts of inputs into finite parts of outputs and *fairmerge* functions. Since u_i, $1 \leq i \leq k$ and v_j, $1 \leq j \leq h$, which are defined in (4.1) and (4.2), are infinite by means of τ, the following lemma is easily seen.

Lemma 5.2. Assume (4.1) and (4.2) for a given logic program. Let f_L: $(DOM^\infty)^{k+h} \rightarrow (DOM^\infty)^{k+h}$ be a function such that $(u_1, \ldots, u_k, v_1, \ldots, v_h) = f_L(u_1, \ldots, u_k, v_1, \ldots, v_h)$ is defined by (4.1) and (4.2). Then f_L is continuous and there is a least fixpoint of f_L.

Now we define the least fixpoint of f_L in Lemma 5.3 as

(4.3) $(u_1^f, \ldots, u_k^f, v_1^f, \ldots, v_h^f)$.

(4.3) is a semantics for (4.1) and (4.2). (4.1) and (4.2) are based on Theorems 3.12 and 3.13, respectively, which are representations of unit deductions and transformations of substitutions over SUB. Therefore (4.1) and (4.2) are sound in generating atoms from a logic program. Thus the following theorem holds.

Theorem 5.3. Assume the least fixpoint f_L as denoted by (4.3). Then $P_j(\bar{x}_j)\Psi_j$ is generative from L if $\Psi_j = v_j^f(p)$ for any $p \in \omega$ as long as $v_j^f(p) \neq \tau$. Also $C_i\Theta_i$ is deduced from L if $\Theta_i \in u_i^f(q)$ for $q \in \omega$ as long as $u_i^f(q) \neq \tau$.

The completeness of (4.3) in denoting the atoms generative from a logic program L is guaranteed by the completeness of unit deduction in atom generation and the equivalence of the unit deduction with the manipulations by the $COMB$ functions, and by the assurance of the *fairmerge* functions to transfer all the parts of infinite inputs.

Therefore we have:

Theorem 5.4. Assume that $C_i\Theta_i$ is deduced from L for $\Theta_i = [[\Phi]]_{\{C\}} \in SUB$. Then $\Theta_i = u_i^f(p)$ for some $p \in \omega$. Also, when $P_j(\bar{x}_j)\Psi_j$ is assumed to be generative from L for $\Psi_j = [[\Theta]]_{\{C\}} \in SUB$, $\Psi_j = v_j^f(q)$ for some $q \in \omega$.

To remove τ, that is, time delay from a sequence, the following function *delete*: $DOM^\infty \rightarrow SUB^\infty$ is useful.

$$delete(u)(p) = if\ u(0) = \tau\ \ then\ delete(next(u))(p)$$
$$else\ if\ p = 0\ then\ u(0)$$
$$else\ delete(next(u))(p-1)$$

for $u \in DOM^\infty$ and $p \in \omega$.

The *next* function is similar to the function in Lucid (See [3]).

$$next(u)(p) = u(p+1)$$

for $u \in DOM^\infty$ and $p \in \omega$.

Finally $(delete(u_1^f), \ldots, delete(u_k^f), delete(v_1^f), \ldots, delete(v_h^f))$ is interpreted as a semantics of a given logic program over SUB^∞.

6. Concluding Remarks

We have constructed a dataflow for a logic program as a recursion equation set over a sequence domain, which is on the basis of the quotient set SUB for the substitution set Sub. SUB is obtained by means of an equivalence relation, that is, generality relation on Sub.

The dataflow denotes unit deductions for the original logic program, and is regarded as involving the nodes to express deductions through clauses, and the nodes to transform and to merge sequences. We adopted hiaton in [13] as time delay, to represent the timing at which no atom exists by means of deductions.

The dataflow defines a continuous functional from a direct product of sequence domains into itself. The sequence domain is the set of all finite and infinite sequences from the union of SUB and the set consisting of the hiaton. It completely and soundly produces the items in SUB with which atom subsets can be generated from the originally given logic program. It is thus concluded that the least fixpoint of the functional is much concerned with a semantics of the original logic program.

Acknowledgement

The author is grateful to Dr.Stephen G. Matthews for his comments on dataflow during the author's visit to University of Warwick. This work was partially supported by the Royal Society of London.

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.Baudinet, Proving termination properties of PROLOG programs: A semantic approach, Research Report STAN–CS–88–1202, Computer Science Dept., Stanford University (1988).
5. M.H.van Emden and R.A.Kowalski, The semantics of predicate logic as a programming language, J. ACM 23 (1976) 733-742.
6. E.Eder, Properties of substitutions and unifiers, J. Symbolic Computation 1 (1985) 31-46.
7. M.Fitting, A deterministic Prolog fixpoint semantics, J. of Logic Programming 2 (1985) 111-118.
8. M.Fitting, A Kripke-Kleene semantics for logic programs, J. Logic Programming 2 (1985) 295-312.
9. G.Frauden, Logic programming and substitutions, Lecture Notes in Computer Science 199 (1985) 146-158.
10. G.Kahn, The semantics of a simple language for parallel programming, Proc. IFIP 74 (1974) 471-475.
11. J.L.Lassez and M.J.Maher, Closures and fairness in the semantics of programming logic, Theoretical Computer Science 29 (1984) 167-184.
12. J.L.Lassez and M.J.Maher, Optimal fixpoints of logic programs, Theoretical Computer Science 39 (1985) 15-25.
13. 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.
14. W.W.Wadge, An extensional treatment of dataflow deadlock, Lecture Notes in Computer Science 70 (1979) 285-299.
15. S.Yamasaki et al., A fixpoint semantics of Horn sentences based on substitution sets, Theoretical Computer Science 51 (1987) 309-324.