数理解析研究所講究録
556巻 1985年 282-293

On the Semantics of Infinite Computations in Logic Programs

Yasubumi Sakakibara (榊原 康文)
Department of Information Science
TOKYO Institute of Technology

Abstract In the classical semantics, successful finite computations in logic programs are characterized by the least fixedpoint. We characterize infinite computations by the greatest fixedpoint on the complete Herbrand universe.

1. Logic programs

Firstly we give some basic definitions.

Definition A term is a variable, a constant or $f(t_1,\dots,t_n)$, where f is an n-ary function symbol and $t_1,\dots,t_n$ are terms. An atom is $p(t_1,\dots,t_n)$, where p is an n-ary predicate symbol and $t_1,\dots,t_n$ are terms. A clause is a pair of sets of atoms of the form: $A_1,\dots,A_m \leftarrow B_1,\dots,B_n$ $(m \geqq 0, n \geqq 0)$, which is to be understood as: for all $x_1,\dots,x_k$, $A_1 \vee \dots \vee A_m \vee \neg B_1 \vee \dots \vee \neg B_n$, where $x_1,\dots,x_k$ are all variables in the clause. A definite clause is a clause where m=1. A goal clause is a clause where m=0. An empty clause is a clause where m=0 and n=0 which is to be understood as a contradiction. A logic program is a finite set of definite clauses.

Definition A substitution is an operation, say θ, which replaces simultaneously each occurrence of the variable in an atom A by a term. The result, denoted by $A\theta$, is called an instance of A by θ. A substitution θ is called a unifier for atoms $A_1,\dots,A_n$ if $A_1\theta=\dots=A_n\theta$. A unifier θ for given atoms is called a most general unifier (m.g.u. for short) if for each unifier σ of those atoms, there is a substitution γ such that $\sigma = \theta\gamma$.

Definition Let P be a logic program. The Herbrand universe HU for P is the set of all ground terms which can be formed out of the constants and functions in P. The Herbrand base HB for P is the set of all ground atoms which can be formed out of predicates, functions and constants in P.

A Herbrand interpretation is an interpretation whose domain is the Herbrand universe HU and in which constants and functions are literally interpreted. We can identify it with a subset of the Herbrand base.

Next we introduce SLD-resolution which is a special-purpose resolution used as interpreter of logic programs. The procedural semantics of logic programs is defined using this SLD-resolution.

Definition A computation rule is a rule to choose an atom, called the selected atom, out of any nonempty goal.

Let G be $<-A_1,\ldots,A_m,\ldots,A_k$ and C be $A<-B_1,\ldots,B_q$ and R be a computation rule. Then G' is said to be derived from G and C with θ via R if the following conditions hold:

(1) A_m is the selected atom by R.

(2) θ is a m.g.u. of A and A_m.

(3) G' is the goal $<-(A_1,\ldots,A_{m-1},B_1,\ldots,B_q,A_{m+1},\ldots,A_k)\theta$.

Let P be a logic program, G be a goal and R be a computation rule. An SLD-derivation (simply derivation) of $P \cup \{G\}$ via R consists of a finite or infinite sequence $G=G_0,G_1,\ldots$ of goals, a sequence $C_1,C_2,\ldots$ of variants of clauses in P and a sequence $\theta_1,\theta_2,\ldots$ of m.g.u.'s, such that each G_{i+1} is derived from G_i

and C_{i+1} with θ_{i+1} via R. Each variant $C_1, C_2, \ldots$ is called an input clause of the derivation.

A ground derivation is same as the derivation except that the substitutions θ_i are unifiers (not necessary to be m.g.u.) and goals G_i are ground (i.e. variable-free).

Definition An SLD-refutation (simply refutation) of $P \cup \{G\}$ is a finite derivation of $P \cup \{G\}$ which has the empty clause $\square$ as the last goal in the derivation.

The success set of P = $\{A \in HB : P \cup \{\leftarrow A\}$ has a refutation$\}$.

Theorem 1.1 (Soundness and completeness of refutations [1],[4]) Let P be a logic program and G a goal. Then there exists a refutation of $P \cup \{G\}$ iff $P \cup \{G\}$ is unsatisfiable.

Definition Let P be a logic program and G a goal. We say a derivation of $P \cup \{G\}$ is finitely failed if the derivation is finite and ends with a goal where the selected atom does not unify with the head of any clause in P.

The finite-failure set of P = $\{A \in HB :$ all derivations of $P \cup \{\leftarrow A\}$ via a computation rule are finitely failed$\}$.

From the "negation as failure" point of view, we infer $\neg A$ if A is in the finite-failure set of P.

Definition (fairness [5]) A derivation is fair if, for every atom B in the derivation, some instantiated copy is selected within a finite number of steps. Note that a fair derivation is finite iff it is a refutation.

Theorem 1.2 ([5]) Let P be a logic program and G a goal. If all derivations of $P \cup \{G\}$ via a computation rule are finitely failed, then $P \cup \{G\}$ has no fair derivation.

2. Least fixedpoint semantics

In the fixedpoint semantics for conventional programs, the semantics of a recursively defined procedure is defined to be the least fixedpoint of the transformation associated with the procedure definition. Here we give a similar definition of fixedpoint semantics for logic programs.

Firstly, with a logic program P we associate a mapping T_p from the set 2^{HB} of all Herbrand interpretations of P to itself. It provides the link between the declarative and procedural semantics of P.

Definition Let P be a logic program. The mapping $T_p:2^{HB} \to 2^{HB}$ is defined as

$$T_p(I) = \{A \in HB: \text{there is a clause } B_0 \leftarrow B_1,\dots,B_n\ (n \geq 0) \text{ in } P \text{ such that } A=B_0\theta \text{ and } B_1\theta,\dots,B_n\theta \in I \text{ for some ground substitution } \theta\}.$$

Lemma 2.1 The mapping T_p is monotonic in the sense that $I_1 \subseteq I_2$ implies that $T_p(I_1) \subseteq T_p(I_2)$, for any I_1 and I_2 in 2^{HB}.

Theorem 2.2 (the Knaster-Tarski fixedpoint theorem [7]) Let L be a complete lattice and $T:L \to L$ a mapping. Then a monotonic mapping T has a greatest fixedpoint (gfp(T), for short) and a least fixedpoint (lfp(T), for short). Furthermore,

$$gfp(T)=\cup\{I:I=T(I)\}=\cup\{I:I\subseteq T(I)\},$$

$$lfp(T)=\cap\{I:I=T(I)\}=\cap\{I:T(I)\subseteq I\}.$$

We define the following ordinal powers of T_p:

$T_p \uparrow 0 = \phi$

$T_p \uparrow n = T_p(T_p \uparrow (n-1))$ if n is a successor ordinal,

$= \cup \{T_p \uparrow k : k<n\}$ if n is a limit ordinal;

$T_p \downarrow 0 = HB$

$T_p \downarrow n = T_p(T_p \downarrow (n-1))$ if n is a successor ordinal,

$= \cap \{T_p \downarrow k : k<n\}$ if n is a limit ordinal.

We have that $T_p \uparrow \omega \subseteq lfp(T_p) \subseteq gfp(T_p) \subseteq T_p \downarrow \omega$

Lemma 2.3 The mapping T_p is continuous in the sense that for every increasing chain $I_1 \subseteq I_2 \subseteq \ldots$ of elements of 2^{HB}, $T(\cup \{I_i : i < \omega\}) = \cup \{T(I_i) : i < \omega\}$.

Now we have a theorem which provides a fixedpoint characterization of the success set of a logic program.

Theorem 2.4 (Fixedpoint characterization of success set [4])

Let P be a logic program. Then

the success set of P = $lfp(T_p) = T_p \uparrow \omega$.

As for the greatest fixedpoint, however we may not have $gfp(T_p) = T \downarrow \omega$. Indeed, $T_p \downarrow \omega$ may not be a fixedpoint of T_p. The following example is such a T_p in [1].

Example Consider the logic program P

```
P = {p(a)<-p(x),q(x).
     p(s(x))<-p(x).
     q(b)<-.
     q(s(x))<-q(x).   }.
```

For all finite n we have

$$T_p \downarrow n = HB - \{q(a), \ldots, q(s^{n-1}(a)), p(b), \ldots, p(s^{n-1}(b))\}.$$

Hence $T_p \downarrow \omega = \{p(s^n(a)):n<\omega\} \cup \{q(s^n(b)):n<\omega\}$. Now $p(a) \notin T_p(T_p \downarrow \omega)$, therefore $T_p \downarrow \omega \neq gfp(T_p)$. In fact, $T_p{}^n(T_p \downarrow \omega) = T_p \downarrow \omega - \{p(s^i(a)):i<n\}$ for finite n, and we have $T_p \downarrow (\omega+\omega) = \{q(s^n(b)):n<\omega\} = gfp(T_p) = lfp(T_p)$.

Theorem 2.5 ([1],[5]) Let P be a logic program. The finite-failure set of P is the complement in HB of $T_p \downarrow \omega$.

3. Greatest fixedpoint semantics

Consider the following logic program P and the goal G:

$$P = \{p(f(x)) \leftarrow p(x).\} \qquad G = \leftarrow p(x).$$

The SLD-resolution for this $P \cup \{G\}$ yields infinite derivation and one might expect its answer to be $\{f(f(\ldots))/x\}$. However in the classical fixedpoint semantics, $lfp(T_p)=gfp(T_p)=\phi$, that is, ϕ is the only fixedpoint, and infinite computations are not taken into account. In this section, we extend the Herbrand universe so that it includes infinite terms, and we will characterize infinite computations by the greatest fixedpoint.

Let P be a logic program, F be the set of functions and constants in P and V be a (finite or infinite) set of variables. First we need the definitions of infinite term and infinite atom to make ready for discussions of infinite computations. We denote the set of all (possibly infinite) trees over a ranked alphabet X by $M^\infty(X)$ (in [3]).

Definition An infinite term is an element of $M^\infty(F \cup V)$. An infinite atom A is $p(t_1,\ldots,t_n)$, where p is an n-ary predicate symbol in P and $t_i \in M^\infty(F \cup V)$ $(1 \leq i \leq n)$. The complete Herbrand universe HU' for P is the set of all ground infinite terms. The complete Herbrand base HB' for P is the set of all ground atoms.

We extend a substitution, a unifier and a m.g.u. to those for infinite atoms. Note that for unifiable infinite atoms, a m.g.u. always exists as in the case of finite atoms. See [3] for details.

In the sequel we adopt the convention that "term" and "atom" will always mean possibly infinite term and atom. If a term or an atom is finite, this will always be explicitly stated.

A <u>complete Herbrand interpretation</u> for P is an interpretation whose domain is the complete Herbrand universe and in which constants and functions are literally interpreted. In an analogous way to HB, the set of all complete Herbrand interpretations for P can be identified with $2^{HB'}$. We also define the mapping $T_p':2^{HB'}\to 2^{HB'}$ as

$T_p'(I)=\{A\in HB'$: there is a clause $B_0\leftarrow B_1,\ldots,B_n$ $(n\geqq 0)$ in P such that $A=B_0\theta$ and $B_1\theta,\ldots,B_n\theta\in I$ for some ground substitution θ $\}$.

Then what we want to show is the important property of T_p' that $gfp(T_p')=T_p'\downarrow\omega$. For this goal, we present some properties of T_p and T_p'.

<u>Lemma 3.1</u> Let P be a logic program. For $A\in gfp(T_p)$, $P\cup\{\leftarrow A\}$ has a (possibly infinite) ground fair derivation.

Proof: For $A\in gfp(T_p)$, we construct a ground fair derivation of $P\cup\{\leftarrow A\}$, $\leftarrow A=\leftarrow N_0,\leftarrow N_1,\leftarrow N_2,\ldots$ inductively. Since $gfp(T_p)$ is the fixedpoint of T_p and a set of ground atoms in HB, it is enough for us to get N_n such that $\{N_n\}\subseteq gfp(T_p)$ $(n\geqq 0)$.

For n=0, since $A\in gfp(T_p)$, clearly $\{N_0\}\subseteq gfp(T_p)$. Now suppose that we get $N_{n-1}=(C_1,\ldots,C_m)$, $m\geqq 1$, and its selected atom (by a fair computation rule) is C_i. By the inductive hypothesis, $C_i\in gfp(T_p)$ and so $C_i\in T_p(gfp(T_p))$. Then there exists a clause $D_0\leftarrow D_1,\ldots,D_q$ in P such that $C_i=D_0\theta$ and $D_1\theta,\ldots,D_q\theta\in gfp(T_p)$

for some ground substitution θ. We define

$N_n=(C_1,\ldots,C_{i-1},D_1,\ldots,D_q,C_{i+1},\ldots,C_m)\theta$.

Then clearly $\{N_n\}\subseteq gfp(T_p)$. Thus $P\cup\{<-A\}$ has a ground fair derivation. □

Lemma 3.2 Let P be a logic program. For $A\in HB$, if $P\cup\{<-A\}$ has a (possibly infinite) ground fair derivation, then $A\in gfp(T_p)$.

Proof: For a ground fair derivation $<-A=<-N_0,<-N_1,<-N_2,\ldots$ of $P\cup\{<-A\}$, let $J=\cup\{N_i:i<\omega\}$ (If the derivation is finite of length n, then $J=\cup\{N_i:i\leqq n\}$). Since the derivation is fair, it is clear that $J\subseteq T_p(J)$. Thus $J\subseteq\cup\{I:I\subseteq T_p(I)\}=gfp(T_p)$. Hence $A\in J\subseteq gfp(T_p)$. □

Example Consider the following logic program P

$P = \{p(s(x))<-p(x).$

$q(0)<-p(x). \quad \}.$

We have the following infinite fair derivation of $P\cup\{<-q(0)\}$

$<-q(0),<-p(x_1),<-p(x_2),<-p(x_3),\ldots$

but no ground fair derivation. We only have the finitely failed ground derivations

$<-q(0),<-p(n),<-p(n-1),\ldots,<-p(0)$.

For a sequence $\theta_1,\theta_2,\theta_3,\ldots$ of substitutions in a derivation, when the sequence θ_1, $\theta_1\theta_2$, $\theta_1\theta_2\theta_3,\ldots$ of substitution compositions converges to a substitution θ, we denote θ by $\lim_n(\theta_1\ldots\theta_n)$.

Lemma 3.3 (m.g.u. lemma) Let P be a logic program and A an atom. Suppose that $P\cup\{<-A\}$ has an (possibly infinite) ground fair derivation. Then $P\cup\{<-A\}$ has a (possibly infinite) fair derivation.

Now we come to the main result of this chapter that $T_P' \downarrow \omega = gfp(T_P')$. The results of lemmas 3.1, 3.2, theorems 2.5 and 1.2 can easily be extended to T_P' and we use the extended versions of them in the proofs of next theorems.

Theorem 3.4 Let P be a logic program. For $A \in HB'$, there is a (possibly infinite) fair derivation of $P \cup \{\leftarrow A\}$ iff $A \in gfp(T_P')$.

Proof:

(only-if part) Suppose that $P \cup \{\leftarrow A\}$ has a (possibly infinite) fair derivation $\leftarrow A = \leftarrow N_0, \leftarrow N_1, \leftarrow N_2, \ldots$ with the sequence $\theta_1, \theta_2, \theta_3, \ldots$ of most general unifiers and the sequence $C_1, C_2, \ldots$ of input clauses. By lemma 3.2, it suffices to show that $P \cup \{\leftarrow A\}$ has the (possibly infinite) ground fair derivation.

Without loss of generality, we can assume that $\lim_m(\theta_{i+1} \cdots \theta_{i+m})$ $(i \geqq 0)$ exists. Take a ground substitution σ and define $N'_i = N_i \lim_m(\theta_{i+1} \cdots \theta_{i+m})\sigma$ for each $i \geqq 0$. Then it is clear that $N'_i \subseteq HB'$ $(i \geqq 0)$. Next we confirm that $\leftarrow N'_0, \leftarrow N'_1, \leftarrow N'_2, \ldots$ is a ground fair derivation from $\leftarrow A$ with unifiers $\lim_m(\theta_1 \cdots \theta_m)\sigma, \lim_m(\theta_2 \cdots \theta_m)\sigma, \ldots$ and input clauses $C_1, C_2, \ldots$ by induction on the derivation steps.

First note that $N'_0 = N_0 = A$. Suppose that we get $N'_{n-1} = N_{n-1} \lim_m(\theta_n \cdots \theta_{n+m})\sigma$, where $N_{n-1} = (B_1, \ldots, B_k)$, B_i is its selected atom, C_n is $D_0 \leftarrow D_1, \ldots, D_q$ and $N_n = (B_1, \ldots, B_{i-1}, D_1, \ldots, D_q, B_{i+1}, \ldots, B_k)\theta_n$ for the m.g.u. θ_n such that $B_i \theta_n = D_0 \theta_n$. Then $\lim_m(\theta_n \cdots \theta_{n+m})\sigma$ is also a unifier of B_i and D_0, and from $N'_{n-1} = (B_1, \ldots, B_k)(\lim_m(\theta_n \cdots \theta_{n+m})\sigma)$, we can derive

$$(B_1, \ldots, B_{i-1}, D_1, \ldots, D_q, B_{i+1}, \ldots, B_k)(\lim_m(\theta_n \theta_{n+1} \cdots \theta_{n+m})\sigma)$$
$$= (B_1, \ldots, B_{i-1}, D_1, \ldots, D_q, B_{i+1}, \ldots, B_k)\theta_n(\lim_m(\theta_{n+1} \cdots \theta_{n+m})\sigma)$$
$$= N_n(\lim_m(\theta_{n+1} \cdots \theta_{n+m})\sigma)$$

$=N'_n$.

Hence we can get a (possibly infinite) ground fair derivation $\leftarrow A = \leftarrow N'_0, \leftarrow N'_1, \leftarrow N'_2, \ldots$ of $P \cup \{\leftarrow A\}$.

(if part) It is clear from lemma 3.1 and m.g.u. lemma. □

Corollary 3.5 Let P be a logic program. Then we have

$T_P' \downarrow \omega = gfp(T_P')$.

Proof: Straightforward by theorems 3.4, 2.5 and 1.2. □

Corollary 3.5 is also obtained by a topological approach in [6].

4. Answer substitutions of infinite computations

Let P be a logic program, G a goal and $\theta_1, \theta_2, \ldots, \theta_n, \ldots$ the sequence of m.g.u.'s in a derivation of $P \cup \{G\}$. An answer substitution for $P \cup \{G\}$ is the substitution obtained by restricting the composition $\lim_n(\theta_1 \ldots \theta_n)$ (or $\theta_1 \ldots \theta_n$ if the sequence is finite of length n) to the variables of G [1]. Then we have the following result, a stronger version of theorem 3.4.

Theorem 4.1 Let P be a logic program and A an atom.

(1) If θ is the answer substitution of a fair derivation of $P \cup \{\leftarrow A\}$, then $A\theta\sigma \in gfp(T_P')$ for a ground substitution σ.

(2) For every substitution θ such that $A\theta \in gfp(T_P')$, there is a fair derivation of $P \cup \{\leftarrow A\}$ with the answer substitution θ' such that $\theta = \theta'\sigma$ for some substitution σ.

Proof: (1) is shown exactly as theorem 3.4.

(2) By theorem 3.4, $P \cup \{\leftarrow A\theta\}$ has a fair derivation. Then, by lifting lemma (see, for example, [2]), we get the conclusion. □

Example Consider the following logic program P which computes the infinite "factorial" sequence from 1.

```
P = {fact(z)<-integer(1,n),fact1(n,1.z).
     fact1(x.n,y.v.w)<-mult(x,y,v),fact1(n,v.w).
     integer(n,n.x)<-plus(n,1,y),integer(y,x).  }.
```

where the argument of fact is the factorial sequence.
(Note the convention that plus(x,y,z) means x+y=z and mult(x,y,z) means x*y=z).
For the goal <-fact(z), we get the answer substitution z=1.2.6.24.120.... through a fair derivation. Then clearly fact(1.2.6.24.120....) $\in gfp(T_p')$.

5. Concluding remarks

As we have seen, taking the complete Herbrand universe as the domain seems to be the simplest and most natural way of providing a semantics for infinite computations. Then $gfp(T_p')$ is the analogue of the semantics $lfp(T_p)$ for (ordinary) terminating logic programs and a fair derivation is the analogue of a refutation.

The characterization of infinite computations remains an open question. The problem is to find the appropriate sense of an infinite computation being "useful". Then finding a satisfactory semantics for useful infinite computations is also a research problem. The greatest fixedpoint semantics gives a nonempty denotation not only to non-terminating logic programs which compute infinite terms, but also to "bad (or not useful)" non-terminating logic programs (for example, "loop" programs). It is a further work to give a good fixedpoint semantics for infinite computations.

Acknowledgement The author would like to thank Prof. Masako Takahashi Horai and Mr. Hideki Yamasaki for their useful comments and encouragement. Thanks are due also to Prof. J.-L.Lassez for his helpful comments.

References

[1] Apt,K.R. and van Emden,M.H.: Contributions to the theory of logic programming, JACM, 29, 3(1982), 841-862.

[2] Chang,C.L. and Lee,R.C.T.: Symbolic logic and mechanical theorem proving, Academic Press, New York, 1973.

[3] Courcelle,B.: Fundamental properties of infinite trees, TCS, 25(1983), 95-169.

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

[5] Lassez,J.-L. and Maher,M.J.: Closures and fairness in the semantics of programming logic, TCS, 29(1984), 167-184.

[6] Lloyd,J.W.: Foundations of logic programming, Technical report (revised) 82/7, Univ. of Melbourne, 1984.

[7] Tarski,A.: A lattice-theoretical fixpoint theorem and its applications, Pacific J. Math., 5(1955), 285-309.