Pusan-Kyŏngnam Math. J. 7(1991), No. 1, pp. 37–43

THE EXTENDED SEMATICS FOR LOGIC PROGRAMMING IN THE HIGHER ORDER CALCULUS

HYANG IL YI

1. Introduction

We require the concept of ordinal powers. First we recall some elementary properties of ordinal numbers, simply ordinals. The first ordinal 0 is defined to be $\emptyset$. Then we define $1 = \{\emptyset\} = \{0\}$, $2 = \{\emptyset, \{\emptyset\}\} = \{0, 1\}$, $3 = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\{\emptyset\}\}\}\} = \{0, 1, 2\}$, and so on. These are the finite ordinals, the non-negative integers. The first infinite ordinal is $\omega = \{0, 1, 2, \cdots\}$, the set of all non-negative integers. We can specify an ordering $<$ on the collection of all ordinals by defining $\alpha < \beta$ if $\alpha \in \beta$. We will normally write $n \in \omega$ rather than $n < \omega$. If α is an ordinal, the successor of α is the ordinal $\alpha + 1 = \alpha \cup \{\alpha\}$, which is the least ordinal greater than α. $\alpha + 1$ is then said to be a successor ordinal. If α is a successor ordinal, $\alpha = \beta + 1$, we denote β by $\alpha - 1$.

An ordinal α is said to be a limit ordinal if it is not the successor of any ordinal. The smallest limit ordinal (apart from 0) is ω. After ω comes $\omega + 1 = \omega \cup \{\omega\}$, $\omega + 2 = (\omega + 1) + 1$, $\omega + 3$, and so on. The next limit ordinal is $\omega 2$, which is the set consisting of all n and all $\omega + n$ where $n \in \omega$. Then comes $\omega 2 + 1$, $\omega 2 + 2$, $\cdots, \omega 3$, $\omega 3 + 1, \cdots, \omega 4, \cdots, \cdots, \omega n, \cdots$.

2. Unification algorithms

Resolutions play a fundamental role in problem solving based on clausal forms as Prolog. In fact, resolvent must be obtained at every execution step, and the efficiency of this operation affects crucially the efficiency of the whole procedural interpreter.

We are interested in substitutions which unify a set of expression, that is, make each expression in the set syntactically identical. The

Received March 5, 1991.

concept of unification goes back to Herbrand in 1930. It was rediscovered in 1960 by Prawitz and further exploited by Robinson for use in the resolution rule.

Colmerauer(1983) has proposed a novel theoretical model involving occur checks, and has presented convincing models of the usefulness of such semantics. He presents also corret algorithm for unification with infinite terms of occur checks.

David(1984) presented a method for preprocessing Prolog programs so that their operational sementics would be given by the first order predicate calculus. The Prolog program can be preprocessed to add tests at these places to cause subgoals to fail if they return terms with loops.

Proposals have been recently put forward to extend logic programming so as to deal with infinite terms efficiently. In this paper we present extended semantics of the higher calculus for logic programmings to get rid of occur checks.

3. Formal Semantics

In this section we shell give some semantics of occur checks based on the higher-order calculus. We introduce class functional of fixed objects in view of the theory of types.

DEFINITION 2.1. A *FORMULA* is defined recursively as an expression of propositions satisfying the following :

(1) If s is a term of type n and t is a term of type $n+1$, then $s \in t$ is a formula, where n is a positive integer.
(2) If A is a formula, so is $\neg(A)$.
(3) If A and B are formulas,then $(A \rightarrow B)$ is a formula.
(4) If A is a formula and x is a variable, then $\forall x(A)$ is a formula.
(5) A formula is contructed by finite applications of (1)–(4) above.

Among the logical laws of the higher-order calculus the concept of *IDENTITY* is the one which has the greatest importance. Leibniz first stated the concept of identity as follows:

(LEIBNIZ'S LAW). *$x = y$ if and only if x has every property which y has, and y has every property which x has.*

In the following formulation we employ the term "class" instead of "property".

(LEIBNIZ's LAW). *$x = y$ if and only if every class which contains any one of the things x and y as an element also the other as an element. Now we may set up the following two axioms:*

AXIOM 2.1.

$$\forall x \quad (x \in y \rightleftarrows x \in z \to y = z).$$

AXIOM 2.2. *For every formula $F(x)$,*

$$\exists y \quad \forall x \quad (x \in y \rightleftarrows F(x)).$$

We have a fundamental semantics of occur checks from the following theorem.

THEOREM 2.1. *For every mapping f respect to member x of type n,*

$$\exists 1 y \quad \forall x \; . (x \in y \rightleftarrows x = f(x)).$$

Proof. If follows from the transitivity of identity and axiom 2.1. It remains to show the uniqueness,

$$\forall x \quad (x \in y \rightleftarrows x = f(x)) \quad \text{and} \quad \forall x \quad (x \in z \rightleftarrows x = f(x))$$
$$\to \forall x \quad (x \in y \rightleftarrows x \in z \to y = z)$$

By theorem 2.1, we get a term $\{\, x \,|x = f(x)\,\}$ of type $n+1$ whenever x is of type n. Let us define the special members of type $n+1$ associated with an occur checks.

DEFINITION 2.2. For every mapping f of type n,

(1) $\{\, x \,|x = f(x) = (x)\,\} = \exists 1 y \quad \forall x \quad (z \in y \rightleftarrows x = f(x))$.
(2) $\{\, x \,|x = f(x) = (x, a)\,\} = \exists 1 y \quad \forall x \quad (x \in y \rightleftarrows x = f(x, a))$.

We have, by definition 2.2.,the following corollaries.

COROLLARY 2.1. *For every mapping f of type n,*

$$a \in \{\, x \,|x = f(x)\,\} \rightleftarrows a = f(a)$$

COROLLARY 2.2. *For every mapping f and g of type n,*

$$\begin{aligned}\{\, x \,|x = f(x)\,\} = \{\, x \,|x = g(x)\,\} \\ \rightleftarrows \forall x \quad (x = f(x) \rightleftarrows x = g(x)) \\ \rightleftarrows \forall x \quad (f(x) = g(x))\end{aligned}$$

Proof. Let $y = \{\, x \,|x = f(x)\,\}$. then we have

$y \rightleftarrows \forall x \quad (x \in y \rightleftarrows x = f(x))$.
Replacing y by $\{\, x \,|x = g(x)\,\}$, we have
$\{\, x \,|x = f(x)\,\} = \{\, x \,|x = g(x)\,\}$
$\rightleftarrows \{\, x \,|x = g(x) \rightleftarrows x = f(x)\,\}$
$\rightleftarrows \forall x \quad (x = g(x) \rightleftarrows x = f(x))$.

COROLLARY 2.3. *For every mapping f of type n,*

$$\{\, f(x) \,|x = f(x)\,\} = \{\, x \,|x = f(x)\,\}.$$

COROLLARY 2.4. *For every mapping f of type n,*

$$\forall x \quad (x = f(x) \to \forall x \quad (x = f(f(f(\cdots(f(x))\cdots))))).$$

Now let L be a set of type n. Then set inclusion is easily seen to be a partial order on 2^L and 2^L under the set inclusion is a complete lattice. Moreover, the Cartesian product $L = L_1 \times L_2 \times \cdots \times L_k$, where L_i's $(i = 1, 2, \cdots, k)$ are complete lattices of type n, with the inclusion relation defined as

$$(a_1, a_2, \cdots, a_k) \leq (b_1, b_2, \cdots, b_k) \text{ iff } a_i \leq b_i (i = 1, 2, \cdots, k) \text{ in } L_i$$

is a complete lattice. Next we shall consider the complete lattice L defined just above.

DEFINITION 2.3. Let L be a complete lattice and $f : L \to L$ be a mapping. We say f is *MONOTONIC* if $f(x) \leq f(y)$, whenever $x \leq y$.

DEFINITION 2.4. Let L be a complete lattice and $f : L \to L$ be a mapping. We say f is *CONTINUOUS* if $f(\text{lub}(X)) = \text{lub}(f(X))$, for every subset X of L in which every finite subset of X has an upper bound.

Now we can define the ordinal powers of f.

DEFINITION 2.5. *Let L be a complete lattice and $f : L \to L$ be monotonic. Then we define*

(1) $f \uparrow 0 = \perp$.
(2) $f \uparrow \alpha = f(f \uparrow (\alpha - 1))$, *if α is a successor ordinal.*
(3) $f \uparrow \alpha = \text{lub}\{\, f \uparrow \beta : \beta < \alpha \,\}$, *if α is a limit ordinal.*
(4) $f \downarrow 0 = \top$.
(5) $f \downarrow \alpha = f(f \downarrow (\alpha - 1))$, *if α is a successor ordinal.*
(6) $f \downarrow \alpha = \text{glb}\{\, f \downarrow \beta : \beta < \alpha \,\}$, *if α is a limit ordinal.*

LEMMA 2.1. *Let L be a complete lattice and $f : L \to L$ be monotonic. Then f has a least fixpoint lfp(f) in L and a greatest fixpoint gfp(f) in L.*

Next we give a characterization of lfp(f) and gfp(f) in terms of ordinal powers of f.

LEMMA 2.2. *Let L be a complete lattice and $f : L \to L$ be monotonic. Then for any ordinal α, $f \uparrow \alpha \leq$ lfp(f) and $f \downarrow \alpha \geq$ gfp(f).*

LEMMA 2.3. *Let L be a complete lattice and $f : L \to L$ be continuous. then lfp(f) $= f \uparrow \omega$.*

Proof. By Lemma 2.2, we have $f \uparrow \omega$ is a fixpoint. Note that $\{\, f \uparrow n : n \in \omega \,\}$ is directed, since f is monotonic. Then

$$f(f \uparrow \omega) = f(\text{lub}\{f \uparrow n : n \in \omega\}) = \text{lub}\{\, f(f \uparrow n) : n \in \omega \,\} = f \uparrow \omega,$$

using the continuity of f.

Let p be a program. Then 2^{B_p}, which is the set of all Herbrand interpretations of p, is a complete lattice under the partial order of set inclusion $\subseteq$. The top element of this lattice is B_p and the bottom element is $\emptyset$.

THEOREM 2.2. *Let p be a program. Let $f_p : 2^{B_p} \to 2^{B_p}$ be a mapping which has the property $f_p(I) = \{ A \in B_p : A \leftarrow A_1, A_2, \cdots, A_m$ is a ground instance of a clause in p and $\{A_1, A_2, \cdots, A_m\} \subseteq I$, where I is a Herbrand interpretation $\}$. Then the least Herbrand model M_p is the least fixpoint lfp(f_p) and M_p is $f_p \uparrow \omega$.*

Proof. First we must show the fact that the mapping f_p is continuous, and hence monotonic. Let x be a directed subset of 2^{B_p}. Note that $\{A_1, A_2, \cdots, A_m\} \subseteq \text{lub}(x)$ iff $\{A_1, A_2, \cdots, A_m\} \subseteq I$ for some $I \in X$. In order to show f_p is continuous, we have to show $f_p(\text{lub}(x) = \text{lub}(f_p(x))$. Now we have
$A \in f_p(\text{lub}(x))$
iff $A \leftarrow A_1, A_2, \cdots, A_m$ is a ground instance of a clause in p and $\{A_1, A_2, \cdots, A_m\} \subseteq \text{lub}(x)$
iff $A \leftarrow A_1, A_2, \cdots, A_m$ is a ground instance of a clause in p and $\{A_1, A_2, \cdots, A_m\} \subseteq$ for some $I \in X$
iff $A \in f_p(I)$ for some $I \in X$
iff $A \in \text{lub}(f_p(x))$.
And we have $\{A_1, A_2, \cdots, A_m\} \subseteq I$ implies $A \in I$ iff $f_p(x) \subseteq I$. Then

$$\begin{aligned} M_p &= \text{glb}\{ I : I \text{ is a Herbrand model for } p \} \\ &= \text{glb}\{ I : f_p(I) \subseteq I \} \\ &= \text{lfp}(f_p) \\ &= f_p \uparrow \omega. \quad \text{(by Lemma 2.3)} \end{aligned}$$

The mapping f_p of Theorem 2.2 provides the link between the declarative and procedural semantics of a program p. Now we consider the efficient algorithms of occur checks, whenever a variable x of type n occurs in a term and x is fixed by a function f, we may regard f as a fixed function which is pairwise monotonic. Then the unification algorithm can replace the fix point x by an undetermined object of type $n+1$. We suggest that logic programmings must take the semantics of programs in the higher order language to run an efficient algorithms.

References

1. W. F. Clocksin and C. S. Mellish, *Programming in prolog*, Springer–Verlag, 1981.

2. A. Colmerauer, *Prolog and infinite trees ; Logic programming* , Academic Press, 1982.
3. M. Filguiras, *A prolog interpreter working with infinite terms; in implementation of PROLOG*, J. Wiley and Sons, 1984.
4. S. Haridi and D. Sahlin, *Efficient implementation of unification of cyclic structures; in implementation of PROLOG*, J. Wiley and Sons, 1984.
5. J. W. Liod, *Foundations of logic programming*, Springer- Verlag, 1984.
6. A. Martelli and G. Rossi, *Efficient unification with infinite terms in logic programming*, Proc. of the Int. Conf on 5th Generation Computer System, 1984.
7. K. Mukai, *A unification algorithm for infinite trees*, Proc. of the 8th Int. Conf. on Artifical Intelligence, 1983.
8. D. A. Plaisted, *The occur-check problem in prolog*, Proc. of the Int. Symp. on Logic Programming, 1984.
9. A. Tarski, *Introduction to logic*, Oxford Univ Press, 1941.
10. M. H. van Emden and R. A. Kowalski, *The semantics of predicate logic as a programming language*, JACM, 1976.

Department of Applied Mathematics
National Fisheries University of Pusan
Pusan 608–737, Korea