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 $1 = \{0\} = \{0\}$, $2 = \{\emptyset, \{0\}\} = \{0, 1\}$, $3 = \{\emptyset, \{0\}, \{\emptyset, \{0\}\}\} = \{0, 1, 2\}$, and so on. These are the finite ordinals, the non-negative integers. The first infinite ordinal is $\omega = \{0, 1, 2, \ldots\}$, 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 $\alpha$ is an ordinal, the successor of $\alpha$ is the ordinal $\alpha + 1 = \alpha \cup \{\alpha\}$, which is the least ordinal greater than $\alpha$. $\alpha + 1$ is then said to be a successor ordinal. If $\alpha$ is a successor ordinal, $\alpha = \beta + 1$, we denote $\beta$ by $\alpha - 1$.

An ordinal $\alpha$ is said to be a limit ordinal if it is not the successor of any ordinal. The smallest limit ordinal (apart from 0) is $\omega$. After $\omega$ 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 $\omega^2 + 1, \omega^2 + 2, \ldots, \omega^3, \omega^3 + 1, \ldots, \omega^4, \ldots, \omega^n, \ldots$.

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 correct algorithm for unification with infinite terms of occur checks.

David (1984) presented a method for preprocessing Prolog programs so that their operational semantics 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 programs to get rid of occur checks.

3. Formal Semantics

In this section we shall 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 constructed by finite applications of (1)–(4) above.

Among the logical laws of the higher-order calculus the concept of IDENTIFY 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.**

$$\forall x \ (x \in y \iff x \in z \rightarrow y = z).$$

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

$$\exists y \ \forall x \ (x \in y \iff F(x)).$$

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 $n$,

$$\exists y \ \forall x \ (x \in y \iff x = f(x)).$$

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

$$\forall x \ (x \in y \iff x = f(x)) \text{ and } \forall y \ (x \in z \iff x = f(x)) \rightarrow \forall x \ (x \in y \iff x \in z \rightarrow y = z)$$

By theorem 2.1, we get a term $\{ x \mid x = f(x) \}$ 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. $\{ x \mid x = f(x) \} = \exists y \ \forall x \ (z \in y \iff z = f(x)).$
2. $\{ x \mid x = f(x) = (x, a) \} = \exists y \ \forall x \ (z \in y \iff z = f(x; a)).$

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

**Corollary 2.1.** For every mapping $f$ of type $n$,

$$a \in \{ x \mid x = f(x) \} \iff a = f(a)$$
**Corollary 2.2.** For every mapping $f$ and $g$ of type $n$,
\[
\{ x \mid x = f(x) \} = \{ x \mid x = g(x) \}
\]

\[\iff \forall x \ (x = f(x) \Rightarrow x = g(x))\]

\[\iff \forall x \ (f(x) = g(x))\]

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

\[y \Rightarrow \forall x \ (x \in y \Rightarrow x = f(x)).\]

Replacing $y$ by $\{ x \mid x = g(x) \}$, we have

\[\{ x \mid x = f(x) \} = \{ x \mid x = g(x) \}\]

\[\Rightarrow \forall x \ (x = g(x) \Rightarrow x = f(x))\]

\[\Rightarrow \forall x \ (x = g(x) \Rightarrow x = f(x)).\]

**Corollary 2.3.** For every mapping $f$ of type $n$,

\[\{ f(x) \mid x = f(x) \} = \{ x \mid x = f(x) \}.\]

**Corollary 2.4.** For every mapping $f$ of type $n$,

\[\forall x \ (x = f(x) \rightarrow \forall x \ (x = f(f(\cdots (f(x)\cdots))))).\]

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 $f : L \rightarrow L$ 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. \( f \uparrow 0 = \bot \).
2. \( f \uparrow \alpha = f(f \uparrow (\alpha - 1)) \), if $\alpha$ is a successor ordinal.
3. \( f \uparrow \alpha = \text{lub}\{ f \uparrow \beta : \beta < \alpha \} \), if $\alpha$ is a limit ordinal.
4. \( f \downarrow 0 = \top \).
5. \( f \downarrow \alpha = f(f \downarrow (\alpha - 1)) \), if $\alpha$ is a successor ordinal.
6. \( f \downarrow \alpha = \text{glb}\{ f \downarrow \beta : \beta < \alpha \} \), if $\alpha$ 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 $\text{lfp}(f)$ in $L$ and a greatest fixpoint $\text{gfp}(f)$ in $L$.

Next we give a characterization of $\text{lfp}(f)$ and $\text{gfp}(f)$ 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 $\alpha$, $f \uparrow \alpha \leq \text{lfp}(f)$ and $f \downarrow \alpha \geq \text{gfp}(f)$.

LEMMA 2.3. Let $L$ be a complete lattice and $f : L \to L$ be continuous. Then $\text{lfp}(f) = f \uparrow \omega$.

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 $f_p(I) = \{ A \in B_p : A \leftarrow A_1, A_2, \ldots, A_m \}$ is a ground instance of a clause in $p$ and $\{ A_1, A_2, \ldots, A_m \} \subseteq I$, where $I$ is a Herbrand interpretation. Then the least Herbrand model $M_p$ is the least fixpoint $\text{lfp}(f_p)$ 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 $\{ A_1, A_2, \ldots, A_m \} \subseteq \text{lub}(x)$ iff $\{ A_1, A_2, \ldots, A_m \} \subseteq I$ 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

\begin{align*}
A \in f_p(\text{lub}(x)) & \iff A \leftarrow A_1, A_2, \ldots, A_m \text{ is a ground instance of a clause in } p \text{ and } \\
\{ A_1, A_2, \ldots, A_m \} \subseteq \text{lub}(x) & \iff A \leftarrow A_1, A_2, \ldots, A_m \text{ is a ground instance of a clause in } p \text{ and } \\
\{ A_1, A_2, \ldots, A_m \} \subseteq I & \text{ for some } I \in X \\
A \in f_p(I) & \text{ for some } I \in X \\
A \in \text{lub}(f_p(x)) & \text{ iff } A \in I \text{ implies } A \in I \text{ iff } f_p(x) \subseteq I.
\end{align*}

Then

\begin{align*}
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. \text{ (by Lemma 2.3)}
\end{align*}

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


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