Comp 311
Principles of Programming Languages
Lecture 11
The Semantics of Recursion II

Corky Cartwright
October 1, 2012
Call-by-value Fixed-Point Operators

Given a recursive definition in a call-by-value language

\[ f := E_f \]

where \( E_f \) is an expression constructed from constants in the based data domain \( D \), operations (continuous functions) on \( D \), and \( f \), what does it mean?

Example: let \( D \) be the domain of Scheme values. Then

\[ \text{fact} := \text{map } n \text{ to if } n=0 \text{ then } 1 \text{ else } n \times \text{fact}(n-1) \]

is a program defining a function in \( D \rightarrow D \).

In a call-by-name language, the meaning of \( \text{fact} \) is

\[ Y (\text{map } f \text{ to } E_f) \]

where \( Y := \)

\[ \text{map } F \text{ to } (\text{map } x \text{ to } F (x (x))) \text{ (map } x \text{ to } F (x (x))) \]

but this expression diverges using call-by-value beta-reduction.
Formulating $\mathbf{Y}_v$ (Call-by-Value $\mathbf{Y}$)

Key trick: use $\eta$-conversion to delay evaluation. In the mathematical literature on the $\lambda$-calculus, $\eta$-conversion is often assumed as an axiom. In models of the $\lambda$-calculus, it is typically required to hold.

Definition: $\eta$-conversion is the following equation:

$$M = \lambda x.\ Mx$$

where $x$ is not free in $M$.

Examples:

$$y = \eta\lambda x.\ yx$$

$$\lambda y.y = \lambda x.\ (\lambda y.y)x$$
What Is the Code for $\mathbf{Y}_v$?

$$\lambda F. (\lambda x. \lambda y. F(x x) y) (\lambda x. \lambda y. F(x x) y)$$

- Recall that application associates to the left: $F(x x) y = (F(x x)) y$
- Does this work for Scheme (or Java with an appropriate encoding of functions as anonymous inner classes)? Yes!
- Let $G$ be some functional $G = \lambda f. \lambda n. M_f$ like $\text{FACT}$ for a recursive function definition. $G$ is a value. Then

$$\mathbf{Y}_v G \rightarrow (\lambda x. \lambda y. G(x x) y) (\lambda x. \lambda y. G(x x) y) \rightarrow$$

$$\lambda y. G ((\lambda x. \lambda z. G(x x) z) (\lambda x. \lambda z. G(x x) z)) y$$

is a value.

- Hence, $G(\mathbf{Y}_v G) \rightarrow (\lambda n. M_f) [f := \mathbf{Y}_v G]$ is a value.

- Moreover,

$$\mathbf{Y}_v G = \lambda y. G ((\lambda x. \lambda z. G(x x) z) (\lambda x. \lambda z. G(x x) z)) y =$$

$$\lambda y. G (\mathbf{Y}_v G) y$$

which is the $\eta$-conversion of $G(\mathbf{Y}_v G)$.
Loose Ends

• Meta-errors
• Read the notes!
  • Explains how to implement rec-let more thoroughly