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

```
fact := map n to if n=0 then 1 else n*fact(n-1)
```

is a program defining a function in D → D.

In a call-by-name language, the meaning of **fact** is

Y (map f to E_f)

where **Y :=**

```
map F to (map x to F (x (x))) (map x 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 η-conversion to delay evaluation.

In the mathematical literature on the λ-calculus, η-conversion is often assumed as an axiom. In models of the λ-calculus, it is typically required to hold.

Definition: η-conversion is the following equation:

$$\mathbf{M = \lambda x.\ Mx}$$

where $\mathbf{x}$ is not free in $\mathbf{M}$.

Examples:

$$\mathbf{y =}\square\mathbf{\lambda x.\ yx}$$

$$\mathbf{\lambda y.y = \lambda x.\ (\lambda y.y)x}$$

What Is the Code for Y_v?

λF.(λx.λy.F(x x)y)(λx.λy.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 =λf.λn.M_f like FACT for a recursive *function* definition. G is a value. Then

  Y_vG $\rightarrow$ (λx.λy.G(x x)y)(λx.λy.G(x x)y) $\rightarrow$

  λy.G ((λx.λz.G(x x)z)(λx.λz.G(x x)z)) y

  is a value.
- Hence, G(Y_vG) $\rightarrow$ (λn.M_f)[f:=Y_vG] is a value.
- Moreover,

  Y_vG = λy.G ((λx.λz. G(x x)z)(λx.λz.G(x x)z)) y =

  λy. G (Y_vG) y

  which is the η-conversion of G(Y_vG)

Loose Ends

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