Comp 311
Principles of Programming Languages
Lecture 12
The Semantics of Recursion III

Corky Cartwright

Swarat Chaudhuri

September 19, 2011

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 =
   (lambda (n) (if (zero? n) 1 (* 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 (lambda (f) E_f)
```

where `Y =`

```
(lambda (F) (lambda (x) (F (x x))) (lambda (x) (F (x x)))
```

but this expression diverges using call-by-value beta-reduction.

Formulating $\mathbf{Y}_{\mathbf{v}}$ (Call-by-Value **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:

```
M = λx. Mx
```

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

Examples:

```
y =λx. yx
λy.y = λx. (λy.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 → (λx.λy.G(x x)y)(λx.λy.G(x x)y) →

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

  is a value.
- Hence, G(Y_v G) → (λn.M_f)[f:=Y_v G] 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