Comp 411
Principles of Programming Languages
Lecture 12
The Semantics of Recursion III & Loose Ends

Corky Cartwright

February 12, 2014

Call-by-name vs. Call-by-value Fixed-Point Operators

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

$\mathbf{f} \stackrel{\text{def}}{=} \mathbf{E}_{\mathbf{f}}$

where $\mathbf{E}_{\mathbf{f}}$ is an expression constructed from constants in the base language and **f**. What does it mean?

Example: let **D** be the domain of Scheme values. Then the base
- operations are continuous functions on **D** and

`fact` $\stackrel{\text{def}}{=}$

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

is a recursive definition of a function on **D** .

In a call-by-name language (`map n to` ... is interpreted using call-by-name), the meaning of `fact` is

`Y(map f to E`$_{\mathbf{f}}$`)`

What if `map` (λ-abstraction) has call-by-value semantics?

Defining **Y** in a Call-by-value Language

We want to define $\mathtt{Y}_\mathtt{v}$, a call-by-value variant of **Y**.

Key trick: use η(eta)-conversion to delay the evaluation. In the mathematical literature on the λ-calculus, η-conversion is often assumed as an axiom. In models of the pure λ-calculus, it typically holds.

Definition: η-conversion is the following equation:

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

where **x** is not free in **M**. If the λ-abstraction used in the definition of **Y** has call-by-value semantics, then given the functional **F** corresponding to recursive function definition, the computation **YF** diverges. We can prevent this from happening by η-converting both occurrences of **F(x x)** within **Y**.

What Is the Code for Y_v?

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

- Does this work for Scheme (or Java with an appropriate encoding of functions as anonymous inner classes)? Yes!
- Let **G** be some functional **λf.M**, like **FACT**, for a recursive *function definition*. **G** and **M** are values (λ-expressions). Then

  $Y_v G = \lambda x.\ (\lambda y.(G(x\ x))y)(\lambda y.(G(x\ x))y) =$

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

  is a value.
- Hence, $G(Y_v G) = (\lambda f.M)(Y_v G) = M[f:=Y_v G]$, which is a value.
- It is straighforward to prove (using conversion rules) that

  $Y_v G = G(Y_v G)$

Loose Ends

- Meta-errors
- Read the notes!
- rec-let (in notes)