

Comp 311
Principles of Programming Languages
Lecture 9
Meta-interpreters III

Corky Cartwright
September 13, 2010

Major Challenge

LC does not include a recursive binding operation (like Scheme **letrec** or **local**). How would we define **eval** for such a construct?

- Key problem: the closure structure for a recursive **lambda** must include an environment that refers to itself!
- In imperative Java, how would we construct such an environment. Hint: how did we build “circular” data structures in Comp 211/212? Imperativity is *brute force*.

Minor Challenge

How could we define an environment that refers to itself in functional Scheme?

Key problem: observe that in **let** and **lambda** the expression defining the value of a variable cannot refer to itself.

- Solution: does functional Scheme contain a recursive binding construct?
- How can we use this construct to define a recursive environment?
- What environment representation must we use?

A Bigger Challenge

Assume that we want to write LC in a purely functional language without a recursive binding construct (say functional Scheme without **define** and **letrec**)?

- Key problem: must expand **letrec** into **lambda**
- No simple solution to this problem. We need to invoke syntactic magic or (equivalently) develop some sophisticated mathematical machinery.