Comp 411
Principles of Programming Languages
Lecture 14
Eliminating Lambda Using Combinators

Corky Cartwright

February 17, 2014

OO Code Samples

- Show selections from solution to Assignment 2
  - Class hierarchy for Binding union
  - Sample visitor method code
- Discuss some OO design tradeoffs
  - Use of `instanceof`
  - With composite, visitor implementation of methods is not always mandated. Good idea to “build in” some core operations of a composite using the interpreter pattern. Why? Leaner (in terms of lines of code). Easier to read.

Good Commenting Conventions

- Javadoc description for every class, field, non-trivial method.
- Method descriptions are informal contracts. Contracts should be as precise as possible. In some cases (e.g., GUI libraries), complete precision may not be feasible.
- Sample solutions could be better commented.

How to Eliminate `lambda`

Goal: devise a few combinators (functions expressed in lambda-notation with no free variables) that enable us to express all λ-expressions without explicitly using λ.

Notation: let `λ* x.M` denote `λx.M` converted to a form that eliminates the starred λ. Then

- `λ* x.x → I` (where `I = λx.x`)

  `λ* x.y → K y` (where `K = λy.λx.y`)
- `λ* x. M N → S (λ* x.M) (λ* x.N)`

  (where `S = λx.λy.λz.(y x)(z x)`)

Strategy: eliminate λ-abstractions from inside out, one-at a time. Any order works. Transformation can cause exponential blow-up.

Note: `I` is technically unnecessary since `SKK = I`