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

Corky Cartwright
February 14, 2018
OO Code Samples

• 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 $\lambda$-expressions without explicitly using $\lambda$.

Notation: let $\lambda^* x. M$ denote $\lambda x. M$ converted to a form that eliminates the starred $\lambda$. Then

- $\lambda^* x. x \rightarrow I$ (where $I = \lambda x. x$)
- $\lambda^* x. y \rightarrow K y$ (where $K = \lambda y. \lambda x. y$)
- $\lambda^* x. M N \rightarrow S (\lambda^* x. M) (\lambda^* x. N)$
  (where $S = \lambda x. \lambda y. \lambda z. ((x z)(y z))$)

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

Note: $I$ is technically unnecessary since $SKK = I$
Checking the **App** case

\[
S \ (\lambda x. M) \ (\lambda x. N)
\]

\[
= (\lambda x. \lambda y. \lambda z(x \ z)(y \ z))\ (\lambda x. M) \ (\lambda x. N)
\]

\[
= (\lambda y. \lambda z((\lambda x. M) \ z)(y \ z))\ (\lambda x. N)
\]

\[
= (\lambda y. \lambda z(M_{x \leftarrow z})(y \ z))\ (\lambda x. N)
\]

\[
= (\lambda z. (M_{x \leftarrow z})(\lambda x. N) \ z))
\]

\[
= (\lambda z. (M_{x \leftarrow z} \ N_{x \leftarrow z}))(\lambda x. (M \ N)) \text{ (by } \alpha\text{-conversion)}
\]

- \[\lambda^* x. x \rightarrow I\] (where \(I = \lambda x. x\))
  
  \[\lambda^* x. y \rightarrow K \ y\] (where \(K = \lambda y. \lambda x. y\))

- \[\lambda^* x. M \ N \rightarrow S \ (\lambda^* x. M) \ (\lambda^* x. N)\]
  
  (where \(S = \lambda x. \lambda y. \lambda z. (x \ z)(y \ z)\))

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

Note: \(I\) is technically unnecessary since \(SKK = I\)