Lesson2
Lambda Calculus Basics

1/6/05

Chapter 5.1, 5.2

Outline

- Syntax of the lambda calculus
  - abstraction over variables
- Operational semantics
  - beta reduction
  - substitution
- Programming in the lambda calculus
  - representation hacks

Basic ideas

- introduce variables ranging over values
- define functions by (lambda-) abstracting over variables
- apply functions to values

$x + 1$

$\lambda x.\ x + 1$

$(\lambda x.\ x + 1)\ 2$

Abstract syntax

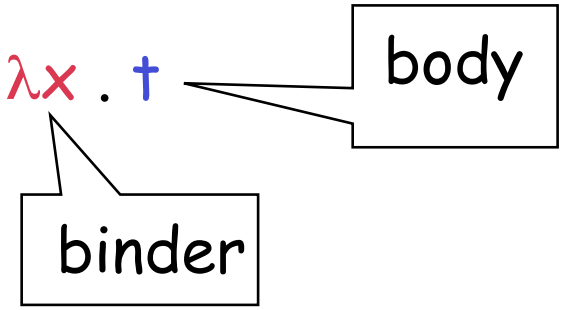
Pure lambda calculus: start with nothing but variables.

Lambda terms

t ::=

x	variable
λx . t	abstraction
t t	application

Scope, free and bound occurences



Occurences of x in the body t are bound.
Nonbound variable occurrences are called free.

$(\lambda x . \lambda y. zx(yx))x$

Beta reduction

Computation in the lambda calculus takes the form of beta-reduction:

$$(\lambda x.\ t1)\ t2 \rightarrow [x \Rightarrow t2]t1$$

where $[x \Rightarrow t2]t1$ denotes the result of substituting t2 for all free occurrences of x in t1.

A term of the form $(\lambda x.\ t1)\ t2$ is called a beta-redex (or β-redex).

A (beta) normal form is a term containing no beta-redexes.

Beta reduction: Examples

$(\lambda x.\lambda y.y\ x)(\lambda z.u) \rightarrow \lambda y.y(\lambda z.u)$

$(\lambda x.\ x\ x)(\lambda z.u) \rightarrow (\lambda z.u)\ (\lambda z.u)$

$(\lambda y.y\ a)((\lambda x.\ x)(\lambda z.(\lambda u.u)\ z)) \rightarrow (\lambda y.y\ a)(\lambda z.(\lambda u.u)\ z)$

$(\lambda y.y\ a)((\lambda x.\ x)(\lambda z.(\lambda u.u)\ z)) \rightarrow (\lambda y.y\ a)((\lambda x.\ x)(\lambda z.\ z))$

$(\lambda y.y\ a)((\lambda x.\ x)(\lambda z.(\lambda u.u)\ z)) \rightarrow ((\lambda x.\ x)(\lambda z.(\lambda u.u)\ z))\ a$

Evaluation strategies

- Full beta-reduction
  - any beta-redex can be reduced
- Normal order
  - reduce the leftmost-outermost redex
- Call by name
  - reduce the leftmost-outermost redex, but not inside abstractions
  - abstractions are normal forms
- Call by value
  - reduce leftmost-outermost redex where argument is a value
  - no reduction inside abstractions (abstractions are values)

Programming in the lambda calculus

- multiple parameters through currying
- booleans
- pairs
- Church numerals and arithmetic
- lists
- recursion
  - call by name and call by value versions

Symbols

Symbols $\lambda \rightarrow \beta \alpha \Rightarrow \rightarrow \rightarrow \rightarrow$