

Lesson2 Lambda Calculus Basics

1/10/02
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 tricks

Basic ideas

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

$x + 1$

$\lambda x. x + 1$

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

1/10/02

Lesson 2: Lambda Calculus

3

Abstract syntax

Pure lambda calculus: start with *nothing* but *variables*.

Lambda terms

$t ::=$

x

variable

$\lambda x. t$

abstraction

$t t$

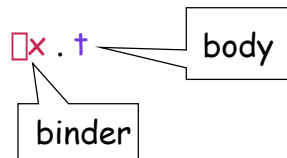
application

1/10/02

Lesson 2: Lambda Calculus

4

Scope, free and bound occurrences



Occurrences 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**:

$$(\lambda x. t_1) t_2 \rightarrow [x \mapsto t_2]t_1$$

where $[x \mapsto t_2]t_1$ denotes the result of **substituting** t_2 for all free occurrences of x in t_1 .

A term of the form $(\lambda x. t_1) t_2$ is called a **beta-redex** (or **λ -redex**).

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

Beta reduction: Examples

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

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

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

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

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

1/10/02

Lesson 2: Lambda Calculus

7

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)

1/10/02

Lesson 2: Lambda Calculus

8

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

1/10/02

Lesson 2: Lambda Calculus

9

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 **λ -redex**).

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

1/10/02

Lesson 2: Lambda Calculus

10

Symbols

Symbols λ μ ν \Rightarrow \rightarrow \rightarrow \rightarrow

1/10/02 Lesson 2: Lambda Calculus 11