

Lambda-Calculus Overview

Susan Older

VERSION: February 21, 2007

Abstract

These notes provide a review of some of the terminology from lectures that do not explicitly appear in the textbook but that I expect you to know for assignments and exam.

1 Syntactic Properties

Definition 1:

The set of free variables in E , written $\text{fv}[E]$, is defined inductively as follows:

$$\begin{aligned}\text{fv}[X] &= \{X\} \\ \text{fv}[(E_1 E_2)] &= \text{fv}[E_1] \cup \text{fv}[E_2] \\ \text{fv}[(\text{lambda } (X) E)] &= \text{fv}[E] - \{X\}\end{aligned}$$

Definition 2:

The set of bound variables in E , written $\text{bv}[E]$, is defined inductively as follows:

$$\begin{aligned}\text{bv}[X] &= \emptyset \\ \text{bv}[(E_1 E_2)] &= \text{bv}[E_1] \cup \text{bv}[E_2] \\ \text{bv}[(\text{lambda } (X) E)] &= \begin{cases} \text{bv}[E] \cup \{X\} & \text{if } X \in \text{fv}[E] \\ \text{bv}[E] & \text{otherwise} \end{cases}\end{aligned}$$

Definition 3:

In any abstraction $(\text{lambda } (X) E)$:

- E is the **body** of the abstraction
- The occurrence of X in $(\text{lambda } (X) \dots)$ is called a **binding occurrence**.
- The **extent** of the binding (or declaration) of X is the entire body E .
- The **scope** of the binding of X is that portion of the body in which occurrences of X are (or would be) free. Thus, the scope of the binding of X is the extent of the binding, minus any subsequent declarations of X that occur within the body E .

2 Substitution

Definition 4:

Let M and N be λ -calculus expressions, and let X be a variable. The **substitution of N for X in M** , denoted $M[N/X]$, is defined inductively as follows:

- If M is a variable reference Y :

$$Y[N/X] = \begin{cases} N & \text{if } X \text{ and } Y \text{ are the same variable (i.e., } X = Y) \\ Y & \text{if } X \text{ and } Y \text{ are not the same variable (i.e., } X \neq Y) \end{cases}$$

- If M is an application $(M_1 M_2)$:

$$(M_1 M_2)[N/X] = (M_1[N/X] M_2[N/X])$$

- If M is an abstraction $(\text{lambda } (Y) P)$:

$$\begin{aligned} & (\text{lambda } (Y) P)[N/X] \\ &= \begin{cases} (\text{lambda } (Y) P) & \text{if } X \text{ and } Y \text{ are the same variable} \\ (\text{lambda } (Y) P) & \text{if } X \neq Y \text{ and } X \notin \text{fv}[P] \\ (\text{lambda } (Y) P[N/X]) & \text{if } X \neq Y, X \in \text{fv}[P], \text{ and } Y \notin \text{fv}[N] \\ (\text{lambda } (Z) P[Z/Y][N/X]) & \text{if } X \neq Y, X \in \text{fv}[P], \text{ and } Y \in \text{fv}[N] \\ & \text{where } Z \notin \text{fv}[N] \cup \text{fv}[P] \end{cases} \end{aligned}$$

Note that, in the last case, Z is any fresh variable (i.e., not appearing free in either N or P). ■

Example 1: Here are several examples of substitution:

- $x[(w y)/t] = x$
- $t[(w y)/t] = (w y)$
- $(t x)[(w y)/t] = ((w y) x)$
- $(\text{lambda } (t) (t x))[(w y)/t] = (\text{lambda } (t) (t x))$
- $(\text{lambda } (t) (t x))[(w y)/z] = (\text{lambda } (t) (t x))$
- $(\text{lambda } (t) (t x))[(w y)/x] = (\text{lambda } (t) (t (w y)))$
- $(\text{lambda } (w) (w x))[(w y)/x] = (\text{lambda } (a) (a (w y)))$
- $((t x) (\text{lambda } (w) (w x)))[(w y)/x]$
 $= ((t x)[(w y)/x] (\text{lambda } (w) (w x)))[(w y)/x]$
 $= ((t (w y)) (\text{lambda } (a) (a (w y))))$ ◇

3 α -conversion

Definition 5:

Let X and Y be (distinct) variables, and further suppose that Y does not occur free in the λ -calculus expression E . The expression $(\text{lambda } (X) E)$ α -converts to the expression $(\text{lambda } (Y) E[Y/X])$.

This process of α -conversion is sometimes also called a **change of bound variables**. ■

Example 2: The expression $(\text{lambda } (w) (w x))$ α -converts to each of the following expressions (among others):

$$(\text{lambda } (z) (z x)) \quad (\text{lambda } (t) (t x)). \quad \diamond$$

Example 3: The expression $(\text{lambda } (w) ((\text{lambda } (w) (w y)) (w x)))$ α -converts to

$$(\text{lambda } (z) ((\text{lambda } (w) (w y)) (z x))).$$

However, it does not α -convert to the following expression, because a free occurrence of y would be captured:

$$(\text{lambda } (y) ((\text{lambda } (w) (w y)) (y x))) \quad \diamond$$

4 β -reduction

Definition 6:

A β -redex (short for “ β -reducible expression”) is a λ -calculus expression having the following form:

$$((\text{lambda } (X) M) N)$$

That is, it is an application whose operator (i.e., lefthand-side) is an abstraction. ■

Example 4: Each of the underlined subexpressions is a β -redex:

$$\underline{((\text{lambda } (x) (x y)) (y ((\text{lambda } (t) (t u)) w)))} \quad \diamond$$

Definition 7:

β -reduction is the transformation of a β -redex $((\text{lambda } (X) M) N)$ to the expression $M[N/X]$, and is denoted as follows:

$$((\text{lambda } (X) M) N) \Rightarrow_{\beta} M[N/X] \quad \blacksquare$$

Example 5: The following are examples of β -reduction:

$$\begin{aligned} ((\text{lambda } (x) (x y)) (w u)) &\Rightarrow_{\beta} ((w u) y) \\ ((\text{lambda } (y) (x y)) (w u)) &\Rightarrow_{\beta} (x (w u)) \\ ((\text{lambda } (y) (\text{lambda } (x) (x y))) (w x)) &\Rightarrow_{\beta} (\text{lambda } (q) (q (w x))) \end{aligned} \quad \diamond$$

In fact, β -reduction can occur at any β -redex inside an expression. Thus, the following are also examples of β -reduction, with the involved β -redex underlined:

$$\begin{aligned} &(((\underline{\text{lambda}}(y)(x\ y))(w\ u))(t\ w)) \Rightarrow_{\beta} ((x(w\ u))(t\ w)) \\ &(\text{lambda}(y)(\underline{(\text{lambda}(x)(x\ y))(w\ u)})) \Rightarrow_{\beta} (\text{lambda}(y)((w\ u)\ y)) \end{aligned}$$

Notice that the nonunderlined portions of the original expressions remain unaltered in the final results.

5 Reduction Sequences (Evaluations)

Definition 8:

An expression E is in **normal form** (also called a **normal form**) if it contains no β -redexes. ■

Definition 9:

A **reduction** (or a **reduction sequence**) is a finite sequence of α -conversions and β -reductions. That is, a reduction has form

$$E \Rightarrow E_1 \Rightarrow E_2 \Rightarrow \cdots \Rightarrow E_n,$$

where each \Rightarrow is either \Rightarrow_{α} or \Rightarrow_{β} . ■

Definition 10:

A reduction sequence

$$E \Rightarrow E_1 \Rightarrow E_2 \Rightarrow \cdots \Rightarrow E_n$$

terminates (or is **terminal**) if E_n is a normal form. ■

Theorem 1:

(Church-Rosser) Suppose that

$$E \Rightarrow E_1 \Rightarrow E_2 \Rightarrow \cdots \Rightarrow E_m$$

and

$$E \Rightarrow E'_1 \Rightarrow E'_2 \Rightarrow \cdots \Rightarrow E'_n$$

are both terminal reductions of E . Then E_m and E'_n are α -convertible.

That is, if two different reductions of the same expression terminate, then the resulting terms are equivalent (modulo α -conversion). ■

There are several common reduction strategies (i.e., strategies for choosing the next β -redex to reduce), but here are perhaps the two most familiar:

1. **Normal-order reduction** (also known as *leftmost (outermost) reduction*) says:

Always reduce the leftmost outermost redex.

2. **Applicative-order reduction** says:

Always reduce the leftmost redex $((\text{lambda}(X)\ M)\ N)$ for which N is a value.

An expression E is a *value* provided that all of its β -redexes occur inside the body of λ -abstractions.

Example 6: Consider the following expression, with its β -redexes underlined:

$$\underline{((\lambda (v) (v x)) (w ((\lambda (y) (z y)) x)))}$$

Under normal-order reduction, the reduction proceeds as follows:

$$\begin{aligned} \underline{((\lambda (v) (v x)) (w ((\lambda (y) (z y)) x)))} &\Rightarrow_{\beta} ((w ((\lambda (y) (z y)) x)) x) \\ &\Rightarrow_{\beta} ((w (z x)) x) \end{aligned}$$

Under applicative-order reduction, the reduction proceeds as follows:

$$\begin{aligned} ((\lambda (v) (v x)) (w \underline{((\lambda (y) (z y)) x)})) &\Rightarrow_{\beta} \underline{((\lambda (v) (v x)) (w (z x)))} \\ &\Rightarrow_{\beta} ((w (z x)) x) \end{aligned} \quad \diamond$$