CS:314 Fall 2024

Section **04**

Recitation **11**

chris.tu@rutgers.edu

Office hours: 2-3pm @ Thursday CoRE 335

Topics Covered

- **Lambda Calculus**
  - β-Reduction
  - α-Reduction
  - Programming in Lambda Calculus

Overview

- A **unified** language to manipulate and reason about **functions**.

multiple aspects in
a single system

Function definition
using λ-terms.

Variable binding and substitution
to evaluate expressions.

Function application
to arguments.

What are λ-terms?

- The fundamental building blocks of lambda calculus. Used to make expressions.

***λ*-terms** are:

- **<u>Variables</u>**: x,y,z
- **<u>Function Abstractions</u>**: **λx.M**, where M is another lambda term
  - defines a function with parameter x
- **<u>Function Applications:</u>** **M N**, where M and N are lambda terms
  - applies the function M to the argument N
    - M is a **function** expressed as a lambda term (e.g., λx.x+1).
    - N is the **input** or argument to that function (e.g., 3).
    - Applying **M** to **N** means **substituting N into M's body** for every free occurrence of the variable

(λx.x+1)3

Precedence

"Need to eat **Apples** before you get **Abs**."

- **Function Application** has the highest precedence and is left-associative.
  - Left associative: (f g z) is ((f g) z)
    - apply f to g, then apply that to z and onward
  - Precedence:
    - λx.yz is grouped as λx.(yz)
      - λx.yz represents a single lambda abstraction with a body (y z)
      - everything after an abstraction is considered the "body" unless explicitly grouped with parentheses.
      - to remain syntactically correct, we are applying y to z
    - **λx.λy.xy is grouped as λx.(λy.(xy))**. Multiple args: **(λxy.z) is (λx.(λy.z)).**

Functions to Lambda Calculus Example

- Consider the function $\mathbf{f(x) = x + 4}$.
- This can be expressed in lambda as $\boldsymbol{\lambda x.(+\ x\ 4)}$
- By applying a value to the function, we can evaluate the function.
- Suppose we want to apply the value **2** to each expression of this function:
  - $f(\mathbf{2}) = 2 + 4 = 6$
  - $((\lambda x.(+\ x\ 4))\ \mathbf{2}) = (+\ 2\ 4) = 6$
    - apply function to 2, meaning substitute 2 for every free occurrence of x.

Free and Bound Variables

- If we have an expression as follows:

  $(\lambda x.M)$

- Then we say that x is **bound** in expression M.
- All other variable occurrences in M for this particular function are **free**.

Free and Bound Variables Trivia

- Consider the following expression:

  $(\lambda xy.((x\ y)\ (x\ w)))$

- What variables are bound in this expression?

- What variables are free in this expression?

Free and Bound Variables Answer

- Consider the following expression:

  $(\lambda xy.((x\ y)\ (x\ w)))$

- What variables are bound in this expression?

  Variables **x** and **y** are bound in this expression.

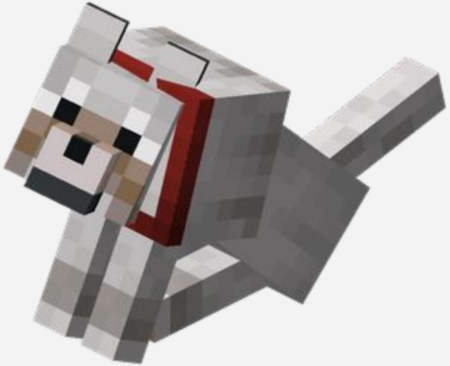
- What variables are free in this expression?

  Variable **w** is free in this expression.

Computation is based on Simplification

Reduce until you can’t anymore.

Alpha-Reduction and Beta-Reduction



Beta-Reduction

- β -Reduction is the technique of applying functions to their arguments.

$$((\lambda x.M)\ v) = [v\ /\ x]M$$

- The notation "[v / x]M" means replacing all free occurrences of x in M with v.

- Examples:
  - $((\lambda x.(+\ x\ 1))\ 2) = [2\ /\ x]\ (+\ x\ 1) = (\ +\ 2\ 1) = 3$
  - $((\lambda x.(+\ x\ x))\ 2) = [2\ /\ x]\ (+\ x\ x) = (\ +\ 2\ 2) = 4$
  - $((\lambda x.3)\ 2) = [2\ /\ x]\ 3 = 3$

More Beta Reduction Examples

- $((\lambda x.\lambda y.(+\ x\ y))\ 2) = [2\ /\ x](\lambda y.(+\ x\ y))$
- $((\lambda x.(x\ y))\ (\lambda z.z)) = ?$
- $((\lambda x.\lambda y.xy)\ (\lambda z.(z\ z))\ x) = ?$

More Beta Reduction Examples

- $((\lambda x.\lambda y.(+\ x\ y))\ 2) = [2\ /\ x](\lambda y.(+\ x\ y))$
  $= (\lambda y.(+\ 2\ y))$
  $= 2 + y$
- $((\lambda x.(x\ y))\ (\lambda z.z)) = ?$
- $((\lambda x.\lambda y.xy)\ (\lambda z.(z\ z))\ x) = ?$

We apply outermost λ first!

More Beta Reduction Examples

- $((\lambda x.\lambda y.(+\ x\ y))\ 2) = [2\ /\ x](\lambda y.(+\ x\ y))$
  $= (\lambda y.(+\ 2\ y))$
  $= 2 + y$
- $((\lambda x.(x\ y))\ (\lambda z.z)) = ((\lambda z.z)\ y)$
  $= y$

- $((\lambda x.\lambda y.xy)\ (\lambda z.(z\ z))\ x) = ?$

More Beta Reduction Examples (answer)

- $((\lambda x.\lambda y.(+\ x\ y))\ 2) = [2\ /\ x](\lambda y.(+\ x\ y))$
  $= (\lambda y.(+\ 2\ y))$
  $= 2 + y$
- $((\lambda x.(x\ y))\ (\lambda z.z)) = ((\lambda z.z)\ y)$
  $= y$

- $((\lambda x.\lambda y.xy)\ (\lambda z.(z\ z))\ x) = ((\lambda y.((\lambda z.(z\ z))\ y)\ x)$
  $= ((\lambda z.(z\ z))\ x)$
  $= (x\ x)$

Alpha-Reduction

- Consider the following expression:
  - $((\lambda x.\lambda y.(x\ y))\ y\ w)$
- Applying Beta-Reduction here gives us:
  - $((\lambda y.(y\ y))\ w) = (w\ w)$
- That's wrong: the y marked in red was free, but accidentally became bound to the inner abstraction **λy**.
- **α-Reduction is the act of renaming a bound variable to avoid this problem.**
  - $((\lambda x.\lambda z.(x\ z))\ y\ w) = ((\lambda z.(y\ z))\ w) = (y\ w)$

Example of Alpha-Reduction

Let's use to α-Reduction and β-Reduction to reduce the following expression:

$$((\lambda x.\lambda y.(x\ y))\ (\lambda y.y)\ w) = ((\lambda x.\lambda z.(x\ z))\ (\lambda y.y)\ w) \qquad \text{via } \alpha\text{-Reduction}$$

$$= ?$$

Example of Alpha-Reduction

Let's use to α-Reduction and β-Reduction to reduce the following expression:

$$
\begin{aligned}
((\lambda x.\lambda y.(x\ y))\ (\lambda y.y)\ w) &= ((\lambda x.\lambda z.(x\ z))\ (\lambda y.y)\ w) && \text{via } \alpha\text{-Reduction} \\
&= ((\lambda z.((\lambda y.\ y)\ z))\ w) && \text{via } \beta\text{-Reduction} \\
&= ?
\end{aligned}
$$

Example of Alpha-Reduction

Let's use to α-Reduction and β-Reduction to reduce the following expression:

$$
\begin{aligned}
((\lambda x.\lambda y.(x\ y))\ (\lambda y.y)\ w) &= ((\lambda x.\lambda z.(x\ z))\ (\lambda y.y)\ w) && \text{via } \alpha\text{-Reduction} \\
&= ((\lambda z.((\lambda y.\ y)\ z))\ w) && \text{via } \beta\text{-Reduction} \\
&= ((\lambda y.\ y)\ w) && \text{via } \beta\text{-Reduction} \\
&= ?
\end{aligned}
$$

Example of Alpha-Reduction (answer)

Let's use to α-Reduction and β-Reduction to reduce the following expression:

$$
\begin{aligned}
((\lambda x.\lambda y.(x\ y))\ (\lambda y.y)\ w) &= ((\lambda x.\lambda z.(x\ z))\ (\lambda y.y)\ w) && \text{via } \alpha\text{-Reduction} \\
&= ((\lambda z.((\lambda y.\ y)\ z))\ w) && \text{via } \beta\text{-Reduction} \\
&= ((\lambda y.\ y)\ w) && \text{via } \beta\text{-Reduction} \\
&= w && \text{via } \beta\text{-Reduction}
\end{aligned}
$$

Programming in Lambda Calculus

- We define **True** = (λxy.x)
    - a.k.a. select-first, since it selects the first of two arguments.
- We define **False** = (λxy.y)
    - a.k.a. select-second, since it selects the second of two arguments

- We can then define **not** = (λx.((x False) True))
- If we apply not to True, it will select the first argument: False.
- If we apply not to False, it will select the second argument: True.
- Let's show more formally why (not False) = True.

Programming in Lambda Calculus

We want to show that **(not False) = True**.

Recall that **not** = (λx.((x False) True)) and **False** = (λxy.y) = (λx.λy.y)

Therefore (not False) = ((λx.((x False) True)) False) by definition
= ((False False) True) by Beta-Reduction
= ((λx.λy.y) False) True) by definition (λx got "eaten up")
= ((λy.y) True) by Beta Reduction
= True by Beta Reduction