Lambda Calculus

Lambda Calculus

- Lambda calculus is a framework developed by Alonzo Church in 1930s to study computations with functions.

- **Function creation** − Church introduced the notation **λx.E** to denote a function in which 'x' is a formal argument and 'E' is the functional body. These functions can be of without names and single arguments.(**lambda abstraction)**

- **Function application** − Church used the notation $\mathbf{E_1.E_2}$ to denote the application of function $\mathbf{E_1}$ to actual argument $\mathbf{E_2}$. And all the functions are on single argument.

Syntax of Lambda Calculus

- Lamdba calculus includes three different types of expressions, i.e.,

$E ::= x$ (variables)

$| E_1 E_2$ (function application)

$| \lambda x.E$ (function creation)

Where **λx.E** is called Lambda abstraction and E is known as λ-expressions.

Evaluating Lambda Calculus

- Pure lambda calculus has no built-in functions.

- Eg:

  (+ (* 5 6) (* 8 3))

- Here, we can’t start with '+' because it only operates on numbers. There are two reducible expressions: (* 5 6) and (* 8 3).

- We can reduce either one first. For example −

(+ (* 5 6) (* 8 3))

(+ 30 (* 8 3))

(+ 30 24) = 54

Free and bound variables

- In an expression, each appearance of a variable is either "free" (to λ) or "bound" (to a λ).
- β-reduction of **(λx . E)** y replaces every **x** that occurs free in **E** with **y**
- **Bound Variable**: a variable that is associated with some lambda.
- **Free Variable**: a var that is *not* associated with any lambda.

Α-CONVERSION

Alpha conversions allow us to rename bound variables.

A bound name x in the lambda abstraction (λ x.e) may be substituted by any other name y, as long as there are *no free occurrences of y in e:*

B-REDUCTION RULE

- We need a reduction rule to handle λs:- **applying functions to their arguments**

1. (λx . * 2 x) 4

   (* 2 4) = 8

2.

(λx . + x x) 4

(+ 4 4) = 8

Eta Reduction

η-reduction allows you to convert between $\lambda x.(f\ x)$ and f whenever x is not a free variable in f. That is, $f = \lambda x(f\ x)$.

imagine that we have a simple function $f\ x = g\ x$. Both g and f take the same argument, x, and the function application function results in the same value (specified by the equality symbol). Since both f and g take the same argument and produce the same result, we can simplify the equation to just $f = g$. In lambda calculus, this simplification is called η-reduction.