UNIT-5

Lambda Calculus:

It is a mathematical method for expressing computation by function.
It is used for studying functional programming language concept.
It can be used to define what a computational function is.

Syntax for Lambda Calculus:

$$\lambda x . M$$

where $\lambda \rightarrow$ lambda calculus
$x \rightarrow$ variable
$. \rightarrow$ operator for seperating variable with fn. body
$M \rightarrow$ expression / function body.

Mathematic function: It is mapping of members of 1 set called domain set to another set called range set.

eg if, $f(x) = x + 2$,
then $\lambda x . x+2$ (in lambda calculus)

eg if $\{\lambda x . x * x\}$ (2)

output = 4

Why Study Lamda Calculus :

i) It has tremendous influence on design and analysis of programming language

ii) It is standard test bed for studying programming language features.

iii) It is the smallest universal programming language

iv) Describes nameless function

1. Definition of Lamda Calculus :

It can be defined inductively as one of the following terms —

i) Identifier / Atom : Lamda Calculus can be a constant variable or function

Constant are functions that give a constant result.

$$E ::= x$$

ii) Abstraction : ~~Lamda Calculus can also be~~ Abstraction is one form that can be represented as follows —

$$(\{\lambda x . E\} \backslash a)$$

eg $(\{\lambda x . x+1\} \backslash 2) = 3$

$(\{\lambda x . x+1\} \backslash 4) = 5$

iii) Application : An application represent more than one lamda expression.

$$E E'$$

eg $\{\lambda x . (x+1)(x+2)\}$

Features of Lamda expression

1) A function of two variables is expressed in lamda Calculus as a function of one argument which returns a function of one argument as curry function.

eg $f(x,y) = x - y$

$\lambda x . \lambda y . x - y$

ii) function application is left associative.

eg fxy

$\Downarrow$

$(fx)y$

iii) Abstraction extends to right associativity.

$\lambda x . x \lambda y . xyz$

$\Downarrow$

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

Q Correctly parenthesise each of the lamda expression

1.) $\lambda x . x \lambda y . yx$

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

2.) $(\lambda x . x)(\lambda y . y) \lambda x . x (\lambda y . y) z$

It is already paranthesized.

: Scope of Variables :

1) Free variable : and : 2) Bound variable :

x is bound in the following expression since its occurance is in the body of the definition preceeded by the λx.

eg) $\lambda x.x$

A name not preeceded by λ is called free variable

eg) $\lambda x.xy$ — y: free; x: bound.

eg) $(\lambda x.x)(\lambda y.yx)$ — x: bound; y: bound; x: free

Thus, an occurance of variable x in λ expression is called bound if it is ~~written~~ within the scope of λx otherwise it is called free.

Q) Find a set of free variable and bound variable in the given λ-expression.

1) $\lambda x.xy\ \lambda z.xz$

Sol

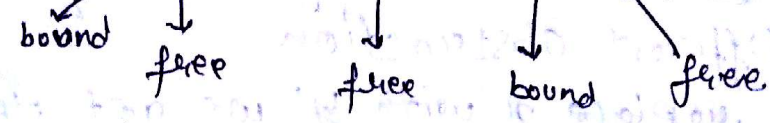
x: bound; y: free; x: free; z: bound.

⇒ bound = x, z

free = x, y

(II) $(\lambda x.xy)\ \lambda z.w\lambda w.wzyx$

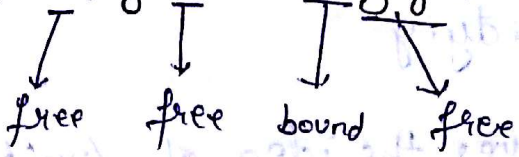
Sol



bound = x, w free = x, y, z, w

(III) $x\lambda z.x\lambda w.wzy$

Sol

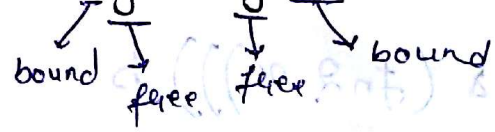


bound = w

free = x, y, z

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

Sol



bound = x.

free = y

Reductions in λ-Calculus :

1) α-reduction
2) β-reduction
3) γ-reduction
4) η-reduction

1) **α-reduction** : It Allows bound variable names to be changed.

Two rules need to be considered —

* the only variable occurance that are renamed are those that are bound to same abstraction

eg $\lambda x.\lambda x.x \Rightarrow \lambda y.\lambda x.x$

or

$\lambda y.\lambda y.y$

* α-conversion is not possible if it would result in a variable getting captured by a different abstraction

eg) if we replace 'x' with 'y' we get the following expression.

eg) $\lambda x.\lambda y.x \not\Rightarrow \lambda y.\lambda y.y$

2) β-reduction: It captures the idea of function application. In this, actual substitution of parameters are carried out by replacing every free occurance of variable 'x' in body 'b'.

eg) $(\lambda x.((fn1\ x)\ \&\ (fn2.x)))\ b$

$\Downarrow$

$((fn1\ b)\ \&\ (fn2\ b))$

3) γ-reduction: It is associated with predefined values & functions

eg) $\gamma add(5,3) \Rightarrow 8$

4) η-reduction: It expresses the idea of extentional Two functions are same if and only if they give same result for all arguments.

eg) $[\lambda x.x*x]\ (x)$

$\eta\ 2*2=4.$

$3*3=9$

Q Using the functions, twice() and successor().
① Write a λ functions that it adds 4 to its argument.

Sol

~~λx.x+4~~

{λx. x+4 / a}

~~λtwice.Twice+4~~

= {λTwice. Twice+4 / successor}

Operations on Lamda Calculus

If fA is a lamda expression and F = λx.M then A may be substituted for all free occurances of x in M.

$$(\lambda x.MA) \Rightarrow M'$$

eg)

* $\lambda x.x(y) \xrightarrow{A} y$
* $(\lambda x.(xy)(y)) \xrightarrow{A} yy$
* $(\lambda x.(xy)\,\lambda x.x) \Rightarrow x.xy \Rightarrow y$

Parameter passing with Lamda Expression

1) Call by name :
* Outermost first
* lazy evaluation.

eg) $(\lambda y.(yy)(\lambda x.(xx)a))$

$\Rightarrow ((\lambda x.(xx)a)(\lambda x.(xx)a))$

$\Rightarrow ((aa)(aa))$