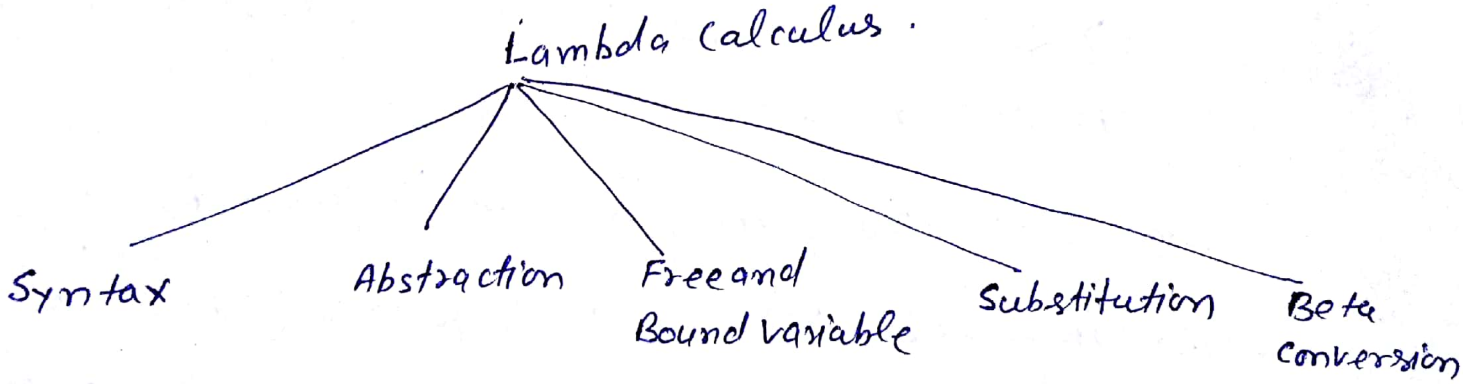
LAMBDA CALCULUS

The lambda (λ) Calculus can be called the smallest universal programming language of the world.

The Lambda (λ) calculus consist single transformation and single function definition scheme.

The Lambda Calculus was introduced in the 1930s by Alonzo Church to express computations with functions.

It is equivalent to Turing Machine and many functional languages like ML, LISP, Haskell are based on Lambda Calculus.



1. Syntax of Lambda Calculus.

```
<expr> ::= (λ<variable>.<expr>)   → Lambda Abstration
         | (<expr> <expr>)         → function application (e.g. prefix, postfix, infix)
         | <variable>              → variable (e.g. a, b, c, x, y, z ...)
         | <constant>
              ↓
         constant (eg: 0, 1, 2 .... or predefined function such as +, * ....)
```

example: $(\lambda x . x+1) (3)$

Head with variable x

Body of expression

another expression used for function Application

$$(\lambda x . x+1) (3)$$

$$\Downarrow$$

$$(3)+1$$

$$\Downarrow$$

$$4$$

2. Lambda Abstraction : The Lambda Abstraction is a way of defining a new function the rule $(\lambda \langle variable \rangle . \langle expr \rangle)$ is called data Abstration it represent nameless function. eg.

$(\lambda x . x+1) (3)$ or $(\lambda x . (x+1)) (3)$ is a nameless function in which x is a variable and $(x+1)$ is Lambda Abstraction. the function is evaluated by substituting 3 for the variable x.

The parenthesis rule in lambad abstration as follows:

(i) E1 E2 E3 can be written as ((E1 E2) E3)

(ii) $\lambda x . E1 E2 E3$ means $\lambda x . (E1 E2 E3)$ and not $(\lambda x . E1 E2) E3$

(iii) $\lambda xyz . E$ means $(\lambda x . (\lambda y . (\lambda z . E)))$

3. Free and Bound Variable:

In Lambda Calculus all names are local to ~~identi~~ definitions. in function $\lambda x.x$ we can say x is 'bound' since x is preceded by λx.

A name not preceded by λ is called 'free' variable.

e.g. $$(\lambda x.xy)$$

$x \rightarrow$ Bound variable

$y \rightarrow$ free variable.

Case for 'free' variable:

(i) $\langle name \rangle$ is free in $\langle name \rangle$

(ii) $\langle name \rangle$ is free in $\lambda \langle name_1 \rangle . \langle expr \rangle$ if $\langle name \rangle \neq \langle name_1 \rangle$ and $\langle name \rangle$ is free in $\langle expr \rangle$

(iii) $\langle name \rangle$ is free in $E_1 E_2$ if $\langle name \rangle$ is free in E_1 or E_2

Case for Bound variable:

(i) $\langle name \rangle$ is bound in $\lambda \langle name_1 \rangle . \langle expr \rangle$ if $\langle name \rangle = \langle name_1 \rangle$ or if $\langle name \rangle$ is bound in $\langle expr \rangle$.

(ii) $\langle name \rangle$ is bound $E_1 E_2$ if $\langle name \rangle$ is bound in E_1 or E_2.

example: $$(\lambda y.y)(\lambda x.xy)$$

4. Substitution:

In the lambda calculus for expression $(\lambda x.E)T$ the argument T will substituted for x in an expression E.

The substitution of Term T for a variable x in E is written as $\{T/x\}E$.

There are two rule that are allowed while performing the substitution.

1. $\{t/x\}\,x = ?$

Replace x in E by t

$\{t/x\}\,x = t$

2. $\{t/x\}\,(* x x) = ?$

Replace x in E by t

$\{t/x\}\,(* x x) = (* t t)$

example: Simplify following using substitution.

(i) $(\lambda x.x)\,z \quad = z \quad$ or $\quad (\lambda x.x)\,z$
$(\lambda x.E)\,T$

$T = z$
$E = x$
$x = x$

$\Rightarrow \{T/x\}\cdot E$
$\Rightarrow \{z/x\}\,x$
$= z$

(ii) $(\lambda x y.x)\,z \Rightarrow \lambda y.z$

(iii) $(\lambda x y.x)\,z u \quad = (\lambda y.z)\,u$

(iv) $(\lambda x.x x)\,(\lambda y.y) \quad = (\lambda y.y)(\lambda y.y)$
$= (\lambda y.y)$

Scanned by CamScanner

5. Beta Conversion: The Beta Conversion is a process of simplifying the expression and this removing the lambda.

e.g. $(\lambda x.\, x+1)(3)$ is equal to $3+1$. Thus we are substituting the value 3 for variable x and throwing the lambda away. this is called beta conversion.

There are two approaches for applying Beta Conversion.

(i) Pass by value.

$$((\lambda x.\,(+\,x\,x))\ (*\,3\,4)) \underset{\beta}{\Rightarrow}$$

$$(\lambda x.\,(+\,x\,x))\ 12 \underset{\beta}{\Rightarrow}$$

$$(+\,12\,12) \underset{\beta}{\Rightarrow} 24$$

(ii) Pass by Name. (delay evaluation)

$$((\lambda x.\,(+\,x\,x))\ (*\,3\,4)) \underset{\beta}{\Rightarrow}$$

$$((\lambda x.\,(+(*\,3\,4)\ (*\,3\,4)))) \underset{\beta}{\Rightarrow}$$

$$((+(*\,3\,4)\ (*\,3\,4))) \underset{\beta}{\Rightarrow}$$

$$((+\ 12\ \ 12)) \underset{\beta}{\Rightarrow}$$

$$(+\,12\ 12) \underset{\beta}{=} 24$$