202303150203

Type : #Note
Tags : Lambda Calculus

Lambda Calculus Syntax

Syntax

Lambda Calculus has a minimal syntax which captures the intrinsional definition of a function, the syntax is as follows

You have a *countably Infinite set* of **variables**, which can be any character except λ . $($ $)$ because these characters have predefined meanings

Lambda calculus is build from expressions/terms of 3 types:

1. x - variable
2. $(\lambda x.\, M)$ - abstraction, here the variable x will be bound in the definition of M
3. $(M\ N)$ - application, Applying a function M on the argument N, where both of there are lambda terms

The variables that are written just after λ and before . are called the *Binding* variables, the variables whose values are associated with it(possibly affected because of a β-reduction), are called *Bound*
like, in the expression

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

Here x is *unbounded*, while the first occurence of y is *binding* and the second one is *bound*.

Composition of lambda expression is left associated:- $ABC = (AB)C$

Rules for a more shorthand notation is

1. $\lambda x.\,(\lambda y.\,M)$ is abbreviated as $\lambda x.\,\lambda y.\,M$
2. $\lambda x_1.\,(\lambda x_2.\,(\ldots(\lambda x_n.\,M)\ldots)$ is abbreviated as $(\lambda x_1 x_2 \ldots x_n.\,M)$
3. $\lambda x.\,MN$ is the same as $\lambda x.\,(MN)$

Examples

- $\lambda a.\,a$ - Which is the identitiy function
- $\lambda x.\,y$ - Which is the contant function, returning y for any input x
- $(\lambda a.\,b)(\lambda c.\,d)$ - Which is applying the left lambda term on the right one
- x - a variable

References

Lambda Calculus Evaluation Rules