202303150303

Type : #Note
Tags : Lambda Calculus

---

# Lambda Calculus Evaluation Rules

## $\beta$-reductions

The only rule that is really required to evaluate any **Lambda Expression** is $\beta$-reduction, which is related to the second rule of the lambda calculus syntax which is

> 2. $(\lambda x. M)$ - abstraction, here the variable $x$ will be bound in the definition of $M$

$\beta$-reduction is the rewriting of a lambda expression when one of the from $\lambda x. M$ is applied to another lambda term $N$. We replace all instances of $x$ bound with the parameter with $N$ which can also be written as

$$(\lambda x. M)N \longmapsto M[x := N]$$

Left part of the equation $(\lambda x. M)N$ is called a *Redex*(Reducible Expression). While the right part of the equation $M[x := N]$ is called the *Contractum*(Contracted Term?? I think??).
An example of this can be:

$$(\lambda xy. xy)z \longmapsto (\lambda y. xy)[x := z] \longmapsto (\lambda y. zy)$$

When no more $\beta$-reductions can be performed, then the lambda term is said to be in a $\beta$-normal form.

## $\alpha$-conversion

There is one slight problem with $\beta$-reductions, that of **variable collision**, which can be easily fixed using $\alpha$-conversions.
$\alpha$-conversions are just formal change of variables in a lambda expression, like

$$\lambda ab.\, ab \equiv_\alpha \lambda cb.\, cb$$

There two lamba term are considered to be intrinsionally equivalent.
The following example will show the problem with $\beta$-reduction.
It is renaming a *binding variable* and all variables *bound* to it. (see Lambda Calculus Syntax > ^BindingsOfVariables)

Let $N = (\lambda xy.\, xy)$ and $M = Ny$
In this case, applying a beta reduction would result in the expression $M = \lambda y.\, yy$ which is not desired,
The desired goal was, that $N$ would take $2$ inputs, and apply the first one on the second one, but the following lambda expression would ignore the first input provided by $M$ and just take one input, applying it to itself.

This can be fixed in the follwing way
let $N = (\lambda xz.\, xz)$, which is an $\alpha$-conversion of the inner lambda term, replacing $y$ with $z$ and now the $\lambda$ expression $M$ evaluates to $(\lambda z.\, yz)$, which was the desired result.

## $\eta$-reduction

$\eta-$reduction is a method to simplify a lambda expression by eliminating redundent inputs and modifying the defintion such that the function.
$\lambda x.\, N\ x \longmapsto N$ is the simplest example of $\eta-$reduction
an example of a slightly more complex $\eta-$reudction would be

$$\begin{aligned}\lambda mnfx.\ (n\ m(\lambda x.\, fx))\ x &\longmapsto \lambda mnf.\ n\ m\ (\lambda x.\, fx)\\ &\longmapsto \lambda mnf.\ n\ m\ f\\ &\longmapsto \lambda mn.\ n\ m\end{aligned}$$

---

# References

Lambda Calculus Syntax