Calculo Lambda

* Variables
* Aplicaciones
* Abstracciones

<expresión> ::= <variable>

| <expresión> <expresión> // Combinación o aplicación

| lambda <variable> <expresión> //abstracción, definir funciones

| <constante>

Ejemplo de función

(lambda x . x)

(lambda f. (lambda x ( f ( f (x))))) sqr

Reducción Beta (hacer la aplicación de constantes o expresiones sobre otras expresiones)

lambda x (sqr ( sqr (x)) 3 )

sqr(sqr(3))

Currying -> aplicar un solo parámetro sobre una función

Función de orden superior -> función que recibe como parámetro otras función

Ejercicios

1. ((lambda x. (x y)) N )

* (N y)

1. ((lambda y. x) N)

* x

1. (lambda x. (Lambda y . y x ) z ) v

* (lambda y . y z ) v
* v z

1. ((lambda x. x x )(lambda x. x x))

* ((lambda x. x x )(lambda x. x x))
* ((lambda x. x x )(lambda x. x x))
* ((lambda x. x x )(lambda x. x x))

….

1. ((lambda x. x x y )(lambda x. x x y))

x x -> variables acotadas

y - > variable libre

* (lambda x. x x y)(lambda x. x x y)y
* (lambda x. x x y)(lambda x. x x y)y y

…

1. (lambda u . v)(lambda x. x y z)

* v

1. (lambda x. x y) (lambda u. v u u)

* (lambda u. v u u)y
* v y y

1. (lambda x . x y) u v

* u y v

1. (lambda x. lambda y. y x) u v

* (lambda y. y u)v
* v u

1. (lambda x. x (x (y (z x)))) (lambda u. u v)

* (lambda u. u v) ((lambda u. u v) (y (z (lambda u. u v) )))
* ((lambda u. u v) (y (z (lambda u. u v) ))) v
* y (z (lambda u. u v)) v v

1. (lambda x. lambda y. x y y) (lambda q. q y z) -> Hay que cambiar de nombre a la variable fija. Ese procedimiento se llama alpha reduction [ y -> a] (Mejor hacerlo en las variables que están acotadas

* (lambda x. lambda a. x a a) (lambda q. q y x)
* lambda a. (lambda q. q y x) a a
* ( lambda a. a y x ) a
* a y x

j) ( lambda x y z. x z ( y z )) (lambda x. x) (lambda x y. x)

=> (lambda y z. (lambda x. x) z ( y z)) (lambda x y. x)

=> (lambda y z. z (y z)) (lambda x y. x)

=> lambda z. z ((lambda x y. x) z)

=> lambda z. z (lambda y. z)