Ejercicio Semanal 2

Marco Antonio Orduña Avila

Facultad de Ciencias, UNAM

Jueves 19 de Septiembre

1. Descripción del programa

Para la implemenntación del programa lo que se hizo fue que para la de variables libres se fue poniendo las apariciones de las variables y ver cuales estaban ligadas por una abstracción lambda e irlas quitando.
Para la de incrementar las variables fue mas complicado, pues lo que hice fue, primero preguntar sí el identificador terminaba en número o sino, si, si terminaba lo que hice fue obtener el número e incrementarle uno, si no lo que hacia era agragarle un uno.
Para la alpha expresión, aplique la definición de la alpha equivalencia, para la de substitución fue más sencillo pues solo era intercambiar las variables por las que estaban y sí ya estaban solo se incrementaban las que ligaban.
Para la beta reducción solo era aplicar la siguiente formula $(\lambda x.a)y \rightarrow^{\beta} a[x := y]$ pero la regla que es la que ocupe es $((\lambda\alpha.\eta)e_2' \rightarrow^{\beta} \eta[\alpha := e_2']$
Para la función de verificado de si una función esta en normal, solo es checar si la expresion es igual a la expresión aplicada beta, pues sí la expresión se le puede aplicar a lo más una reducción beta, quiere decir que no estaba en su forma normal.
Para la función eval. solo era aplicar la función beta hasta que estuviera en su forma normal.

2. Entrada y ejecución

Para la ejecución del semanal es necesario ubicarse en la raíz del archivo y utilizar el interprete de haskell con el codigo ghci y poner el nombre del semanal, para cada función daremos un ejemplo de la ejecución.

Para la función frVars :: Expr -¿[Identifier]

$frVars(L"x"(L"y"(L"z"(L"w"(L"n"(App(L"v"(App(V"x")(V"y")))
(App(V"n")(App(V"w")(V"g"))))))))$

Debe de regresar lo siguiente ["g"]

Para la función incrVar :: Identifier -¿Identifier

incrVar .ªssdbgsjobrj"

Debe de regresar assdbgsjobrj1

incrVar "jksd98798124"

Debe de regresar jksd98798125

incrVar "lifhlaifliahfpoeqjfpo16439"

Debe regresar "lifhlaifliahfpoeqjfpo16440"

Para la función subst :: Expr -¿Substitution -¿Expr

subst (L "x" (App (V "x") (V "y"))) ("y" ,L " z " (V " z "))

Lo que deberia regresar $/x \to (x/z \to z)$

Para la función beta :: Expr -¿Expr

beta (App (L "x" (App (V "x") (V "y"))) (L"z" (V " z ")))

Debe de regresar $(/z \to zy)$

"tambien se puede probar todos los ejemplos que vienen en las notas de la clase"

Para la función normal podemos utilizar el ejemplo anterior

normal (App (L "x" (App (V "x") (V "y"))) (L"z" (V " z ")))

Debe de regresar false

Para la función eval

Solo se deben de aplicar las betas reducciones hasta que queden en forma normal

eval (App (L "n"(L "s"(L "z"(App (V "s") (App (App (V "n") (V "s")) (V "z")))))) (L"s"(L "z"(App (V "s") (V "z ")))))

Debe de regresar $/s \rightarrow /z \rightarrow (s(sz))$

Para las conclusiones solo puedo concluir que eel callculo lambda sirve para la definición de funcion, la nocción de aplicaciones de funciones y la recursión.

Las complicaciones que tuve al hacer el semanal fueron demasiadas, pues para la función de variables libres la hice como tres veces ya que si funcionaban para los casos de la especificación, pero probaba otros casos y no funconaba, ya que tenía que quitar las variables que aparecian el idenificador de la expresión lambda en la recursión de las variables libres y eso lo entendí hasta el último, la siguiente que más se me complicó fue la de incrementar las variables, pues no tenía ni idea por lo que tuve que importar unas bibliotecas para cambiar de String a Int y viceversa, de ahí fue un poco más fácil, la última que se complico pero no tanto fue la de alpha expresión, pues me costó bastante pensar en como tenia que cambiar las variables si ya estaban ligadas antes por una abstracción lambda.

Referencias

las bibliotecas para cambiar de char a entero y de String a entero las saque de la siquiente pagina
`https://hackage.haskell.org/package/base-4.12.0.0/docs/src/Data.Char.html#digitToInt`
`https://www.haskell.org/tutorial/stdclasses.html#sect8.3`