Práctica 3

314032324 Galeana Araujo, Emiliano
314011163 Miranda Sánchez, Kevin Ricardo

Facultad de Ciencias, UNAM

Fecha de entrega: Miercoles 3 de Octubre 2018

1 Descripción del programa

Expresiones del Calculo Lambda

```
-- | Identifier. Tipo que define un nombre de variables como
una cadena de texto.
type Identifier = String

-- | Expr. Tipo que representa una expresion lambda sin tipos.
data Expr = Var Identifier
| Lam Identifier Expr
| App Expr Expr deriving(Eq)

-- | Substitution. Tipo que representa la sustitucion.
type Substitution = ( Identifier , Expr )
```

Los cuales representan las expresiones del calculo lambda sin tipos y el reducto para la funcion de sustitucion.

Se realizaron las siguiente funciones que representan la semántica operacional en en calculo lambda.

```
--frVars. Obtiene el conjunto de variables libres de una expresion.
frVars :: Expr -> [Identifier]


--lkVars. Obtiene el conjunto de variables ligadas de una expresion.
lkVars :: Expr -> [Identifier]

--incrVar. Dado un identificador, si este no termina en numero
--le agrega el sufijo 1, en caso contrario toma el valor del numero y lo
```

```
--incrementa en 1.
incrVar :: Identifier -> Identifier


--alphaExpr. Toma una expresion lambda y devuelve una alpha-equivalente
--utilizando la funcion incrVar hasta encontrar un nombre que no aparezca
-- en el cuerpo.
alphaExpr :: Expr -> Expr

-- subst. Aplica la sustitucion a la expresion dada.
subst :: Expr -> Substitution -> Expr

-- beta. Aplica un paso de la beta reduccion.
beta :: Expr -> Expr

-- locked. Determina si una expresion esta bloqueada, es decir, no se pueden hac
locked :: Expr -> Bool

--eval. Evalua una expresion lambda aplicando beta reducciones hasta quedar bloq
eval :: Expr -> Expr
```

2 Entrada y ejecución

Para correr el programa, se debe estar ubicado en la carpeta src, abrir una terminal e invocar el comando ghci Practica3.
En el programa puede probar algunos ejemplos de ejecucion, escribiendo simplemente el nombre del ejemplo que se quiere ejecutar.
En el programa se encuentran las lineas de codigo.

```
---FRVARS----------
ejemplo = frVars (App (Lam "x" (App ( Var "x" ) ( Var "y" ) ) ) (Lam "z" ( Var "
ejemplo2 = frVars (Lam "f" (App (App (Var "f")(Lam "x"(App(App(Var "f")(Var "x")
---LKVARS----------
ejemplo3 = lkVars (App (Lam "x" (App ( Var "x" ) ( Var "y" ))) (Lam "z" ( Var "z
ejemplo4 = lkVars (Lam "f"(App(App(Var "f" )(Lam "x"(App(App(Var "f")(Var "x" ))
-----INCRVAR-----
ejemplo5 = incrVar "elem"
ejemplo6 = incrVar "x97"

-------ALPHAEXPR----------------------------
ejemplo7 = alphaExpr (Lam "x" (Lam "y" (App (Var "x" ) (Var "y" ))))
ejemplo8 = alphaExpr (Lam "x" (Lam "x1" (App ( Var "x" ) ( Var "x1"))))

----------SUBST-------
ejemplo9 = subst (Lam "x" (App ( Var "x" ) ( Var "y" ) ) ) ( "y" , Lam "z" ( Var
```

```
ejemplo10 = subst (Lam "x" ( Var "y" )) ( "y" , Var "x" )

-----BETA-----
ejemplo11 = beta (App (Lam "x" (App ( Var "x" ) ( Var "y" ))) (Lam "z" ( Var "z"
ejemplo20 = beta (App (Lam "n" (Lam "s" (Lam "z" (App ( Var "s" ) (App (App ( Va
----------LOCKED---------
ejemplo12 = locked (Lam "s" (Lam "z" ( Var "z" ) ) )
ejemplo13 = locked (Lam "x" (App (Lam "x" ( Var "x" ))( Var "z" )))

ejemplo14 = eval (App (Lam "n" (Lam "s" (Lam "z" (App ( Var "s" ) (App (App ( Va
ejemplo15 = eval (App (Lam "n"(Lam "s"(Lam "z" (App (Var "s")(App (App (Var "n")

cero = Lam "s" (Lam "z" (Var "z"))
uno = Lam "s1" (Lam "z1" (App (Var "s1") (Var "z1")))
suc = Lam "n" (Lam "s2" (Lam "z2" (App (Var "s2") (App (App (Var "n") (Var "s2")
ejemplo16 = eval (App suc cero)
ejemplo17 = eval (App suc uno)
```

Entonces, para ejecutar algun de los ejemplos, basta escribir el nombre de la siguiente manera:

```
*Practica3> ejemplo15
\s ->\ z -> (s (s z))
*Practica3> ejemplo12
True
*Practica3> ejemplo3
["x","z"]
*Practica3> ejemplo9
\x -> (x \z -> z)
```

3 Conclusiones

Estuvo perra

References

[1] Leslie Lamport, *LaTeX: a document preparation system*, Addison Wesley, Massachusetts, 2nd edition, 1994.