Ejercicio Semanal 4

Emiliano Galeana Araujo

Facultad de Ciencias, UNAM

Fecha de entrega: 15 de Noviembre de 2018

1. Descripción del programa

En este semanal implementamos máquinas K para evaluar expresiones EAB.

1.1. máquina K

Recordemos el lenguaje de EAB y agregamos el tipo Error. También agregamos nuevos datos, para poder regresentar las evaluaciones de los marcos, al cual llamamos Frame.

```
data Exp = V identifier | I Int | B Bool
  | Add Exp Exp | Mul Exp Exp | Succ Exp | Pred Exp
  | And Exp Exp | Or Exp Exp  | Not Exp
  | Lt Exp Exp  | Gt Exp Exp  | Eq Exp Exp
  | If Exp Exp Exp
  | Let Identifier Exp Exp
  | Error

data Frame = AddL Pending Expr
           | AddR Expr Pending
           | MulL Pending Expr
           | MulR Expr Pending
           | SuccF Pending
           | PredF Pending
```

```
          | AndL Pending Expr
          | AndR Expr Pending
          | OrL Pending Expr
          | OrR Expr Pending
          | NotF Pending
          | LtL Pending Expr
          | LtR Expr Pending
          | GtL Pending Expr
          | GtR Expr Pending
          | EqL Pending Expr
          | EqR Expr Pending
          | IfF Pending Expr Expr
          | LetF Identifier Pending Expr
```

Un ejemplo de un programa en una máquina K es el siguiente, el cuál regresa un error.

```
eval (Not (I 3))
```

Agregamos un nuevo tipo para poder representar un marco, y su evaluación.

```
type Pending ()
```

Las siguientes son funciones que implementamos para poder representar las máquinas K.

```
-- | frVars. Obtiene el conjunto de variables libres de una expresion.
frVars :: Exp → [Identifier]

-- | subst. Realiza la substitucion de una expresion de EAB.
subst :: Exp → Substitution → Exp

-- | eval1. Recibe un estado de la maquina K, y devuelve un paso de la
-- |        transicion.
eval1 :: (Mem, Exp) → (Mem, Exp)

-- | evals. Recibe un estado de la maquina K y devuelve un estado derivado de
-- |        evaluar varias veces hasta obtener la pila vacia.
evals :: (Mem, Exp) → (Mem, Exp)

-- | eval. Recibe una expresion EAB, la evalua con la maquina K, y devuelve un
-- |       valor, iniciando con la pila vacia esta devuelve un valor a la pila.
eval :: Exp → Exp
```

2. Entrada y ejecución

El programa es interpretado por `GCHI` de la siguiente forma

```
~:ghci EjerSem04.hs
```

2.1. Funciones

Ya en el programa, los siguientes son ejemplos de ejecuciones de las funciones antes mencionadas.

```
*EjerSem04> frVars (Add (V ‘‘x’’’) (I 5))
[‘‘x’’’]
*EjerSem04> frVars (Let ‘‘x’’’ (I 1) (V ‘‘x’’’))
[]
*EjerSem04> subst (Add (V ‘‘x’’’) (I 5)) (‘‘x’’’, I 10)
Add (I 10) (I 5)
*EjerSem04> subst (Let ‘‘x’’’ (I 1) (V ‘‘x’’’)) (‘‘y’’’, Add (V ‘‘x’’’) (I 5))
***Exceotion: Could not apply the substitucion.
*EjerSem04> eval1 (E ([], Add (I 1) (I 2)))
*EjerSem04> eval1 (R ([OrR () (B True), AndL () (B False)], B False))
*EjerSem04> evals (E ([], Let ‘‘x’’’ (I 2)
                                    (Mul (Add (I 1) (V ‘‘x’’’)) (V ‘‘x’’’))))
R ([], I 6)
*EjerSem04> evals (E ([], Let ‘‘x’’’ (B True)
                                    (If (V ‘‘x’’’) (V ‘‘x’’’) (B False))))
R ([], B True)
*EjerSem04> evals (E ([], Add (B True) (I 1)))
R ([], Error)
*EjerSem04> eval (Let ‘‘x’’’ (I 1)
                             (If (Gt (V ‘‘x’’’) (I 0))
                                 (Eq (V ‘‘x’’’) (I 0)) (B True)))
B False
*EjerSem04> eval (Not (I 3))
*** Exception: Error.
```

3. Conclusiones

Fue sencillo realizar el semanal después de haber hecho los anteriores y ver como se comportaban eval1, evals, eval, frVars, subst, la parte complicada fue

teníamos que tener algo de la forma R R, E E, E R, R E, y en las expresiones binarias creía que me faltaba uno, pero no iba, así que una vez entendido eso, la implementación fue sencilla, aparte de que el evals y eval fueron más sencillos.

Referencias

[1] Archivero, curso de Estructuras Discretas 2017-1