Lenguajes de Programación 2022-1
Proyecto 03: La Máquina $\mathcal{C}$

Javier Enríquez Mendoza Luis Felipe Benítez Lluis Ramón Arenas Ayala

10 de enero de 2022
Fecha de entrega: Viernes 28 de enero de 2022

Para este proyecto se implementará en Haskell la máquina abstracta $\mathcal{C}$ vista en el curso para la evaluación de programas del lenguaje de estudio TinyC .

1 TinyC

Definición 1.1 (Implementación en Haskell de TinyC)**.** Se definen en Haskell los siguientes tipos de datos algebraicos para implementar los programas del lenguaje TinyC .

```
data Expr = Num Int
          | Bo Bool
          | Id Identificador
          | Fun [Identificador] Stmt
          | Suma Expr Expr
          | Resta Expr Expr
          | Gt Expr Expr
          | Lt Expr Expr
          | Funcall Identificador [Expr]
          deriving(Show,Eq)

data Stmt = Vardec Identificador Expr
          | Fundec Identificador [Identificador] Stmt
          | Asig Identificador Expr
          | Secu Stmt Stmt
          | If Expr Stmt Stmt
          | IfO Expr Stmt
          | While Expr Stmt
          | Return Expr
          | MtP
          deriving(Show,Eq)
```

```
type Program = Either Expr Stmt
```

Es importante notar que los constructores del lenguaje se dividen en dos niveles sintácticos, las expresiones (`Expr`) que son aquellos constructores que regresan un valor y los comandos (`Stmt`), que son aquellos que modifican el estado de evaluación del programa.
Por último se define un sinónimo para los programas en TinyC como un **`Either`** de expresiones o comandos.

2 Ambientes

En esta sección se implementan los ambientes de variables que se usan dentro de la máquina $\mathcal{C}$ para almacenar los valores de las variables.

Definición 2.1 (Ambientes)**.** Se definen los ambientes con los siguientes tipos de datos algebraicos en Haskell.

```
data Value = N Int | B Bool | F [Identificador] Stmt

type Var = (Identificador,Value)

data Env = MtEnv | As Var Env | Star Env Env
```

Primero se define un tipo `Value` que son los valores que se pueden guardar en el ambiente.
Se define también un sinónimo de pares de identificador y valor que son la estructura que se almacena en los ambientes.
Por último se definen los ambientes con tres constructores, el ambiente vacío, el ambiente con un tope y el ambiente estrella que modela el símbolo especial en forma de estrella que se usa en la máquina $\mathcal{C}$ para respetar el alcance.

2.1 Ejercicios

1. Implementa la función

   ```
   look :: Env ->Identificador ->Maybe Value
   ```

   Que busca el valor de un identificador en un ambiente, regresando una instancia **`Maybe`** que en caso de no encontrar el identificador regresa **`Nothing`** mientras que si el identificador si se encuentra en el ambiente regresa **`Just`** de el valor.
2. Implementa la función

   ```
   change :: Env ->Identificador ->Value ->Maybe Env
   ```

   Que modifica el valor de un identificador dentro de un ambiente, siguiendo la definición que se encuentra en las notas de clase. En caso de no encontrarse el identificador se regresa **`Nothing`**, si se encuentra se regresa el nuevo ambiente envuelto en un **`Just`**

3 Máquina $\mathcal{C}$

Definición 3.1 (Implementación de la Máquina abstracta)**.** Se define el tipo de dato algebraico en Haskell para los marcos de la pila de control de la Máquina $\mathcal{C}$.

```
data Marco = VardecM Identificador
           | AsigM Identificador
           | SecuM Stmt
           | IfM Stmt Stmt
           | IfOM Stmt
           | ReturnM
           | FuncallM Identificador
```

Con lo que se definen las pilas de control como el siguiente tipo de dato:

```
data Pila = Mt | Top Marco Pila
```

Y por último se definen los estados de la Máquina $\mathcal{C}$ con el tipo de dato que se muestra a continuación.

```
data State = E Pila Env Env Program | R Pila Env Env Program
```

En donde se tienen dos tipos de estados, `E` representa estados de evaluación mientras que `R` son estados de retorno.

3.1 Ejercicios

1. Define una función

   ```
   trans :: State ->State
   ```

   Que modela las transiciones definidas en la nota para los estados de la máquina $\mathcal{C}$. Es importante implementar todas las transiciones vistas en clase.

Entrega

La entrega será por GitHub Classroom en la tarea que se encuentra en la siguiente liga `https://classroom.github.com/a/wUUxyfX4`.

En esta tarea ya se encuentra el código con las definiciones necesarias para el desarrollo del proyecto estructurado de la siguiente forma:

- En el archivo `tinyc.hs` se incluyen las definiciones mostradas en este documento.

Cualquier copia o plagio total o parcial será calificada directamente con cero.