Programación Funcional

Instituto de Computación, Facultad de Ingeniería
Universidad de la República, Uruguay

Entrada y Salida

Entrada/Salida y Programación Funcional

Haskell es un **Lenguaje Puro**

- El resultado de una función se determina completamente a partir de sus argumentos.
- Las funciones no pueden tener efectos laterales.
- Funciones como las siguientes romperían el modelo.
  - $getLine :: String$
  - $putStrLn :: String \rightarrow ()$
- No podríamos distinguir entre funciones puras e impuras.

El tipo (*IO a*), de Acciones de entrada/salida

En Haskell tenemos el tipo (*IO a*), que funciona como ventana con el Mundo Real.

- Representa a una **acción de entrada/salida** (I/O), que al ser ejecutada retorna un valor de tipo *a*.
- Es un tipo abstracto, con una interfaz que nos provee una especie de mini lenguaje imperativo.
- Permite separar lo puro de lo impuro.

Entrada en Haskell

La función *getLine* retorna una acción de entrada/salida que al ser ejecutada lee una línea de la entrada y la retorna como un *String*.

$$getLine :: IO\ String$$

Como *String* no es lo mismo que *IO String*, separamos valores puros de impuros. Esto no es posible:

$$prefijoIn :: String \rightarrow IO\ String$$
$$prefijoIn\ str = getLine +\!\!+ str$$

do-notation

Podemos componer acciones *IO* con do-**notation**.

$$
\begin{aligned}
&\mathit{prefijoIn} :: \mathit{String} \to \mathit{IO}\ \mathit{String} \\
&\mathit{prefijoIn}\ \mathit{str} = \mathrm{do}\ \mathit{pre} \leftarrow \mathit{getLine} \\
&\qquad\qquad\qquad\qquad\quad \mathit{return}\ (\mathit{pre} +\!\!+ \mathit{str})
\end{aligned}
$$

- *off-side rule*: componentes de un bloque do indentados al mismo nivel.
- el bloque define una secuencia de acciones *IO*.
- con $\leftarrow$ asignamos un nombre al resultado de evaluar una acción *IO*.
- con $\mathit{return} :: a \to \mathit{IO}\ a$ generamos una acción *IO* a partir de un valor puro.

Entrada y *read*

La función *read*

$$read :: Read\ a \Rightarrow String \rightarrow a$$

se puede combinar con *getLine* para leer valores de un tipo dado.

Por ejemplo, la siguiente función lee enteros:

$$\begin{aligned} &getInt :: IO\ Int \\ &getInt = \text{do } line \leftarrow getLine \\ &\qquad\qquad return\ (read\ line) \end{aligned}$$

Salida: "Hola Mundo" en Haskell

main = do *putStrLn* `"Hola Mundo!"`

- La función *main* :: *IO* (), consiste de una secuencia de acciones de entrada/salida. Determina el comienzo de la ejecución del programa.
- La función *putStrLn* :: *String* $\rightarrow$ *IO* (), toma un *String* y genera una acción *IO* que lo imprime en la salida estándar.

Ejemplos

```
putStrs []       = return ()
putStrs (x : xs) = do putStrLn $ show x
                      putStrs xs
```

```
getInts = do i ← getInt
             if i == 0
               then return []
               else do is ← getInts
                       return (i : is)
```

```
main = do putStrLn "<=="
          is ← getInts
          putStrLn "==>"
          putStrs $ reverse is
```

IO y Evaluación Perezosa

Si cambio el programa para no utilizar el resultado de la lectura, ¿es posible que *getInts* no se ejecute?

```
main = do putStrLn "<=="
          getInts
          putStrLn "==>"
```

No, porque se define una secuencia de acciones.
Tampoco es posible que los llamados se reordenen.

Y en este caso ¿Qué se imprime?

```
main = do let ps = [putStrLn "A", putStrLn "B", putStrLn "C"]
          ps !! 1
```

La evaluación perezosa juega su rol, dado que tenemos una lista de acciones *IO*, pero sólo ejecutamos una de ellas.

Manejo de Archivos

type $FilePath = String$

$$
\begin{aligned}
&readFile &&:: FilePath \to IO\ String \\
&writeFile &&:: FilePath \to String \to IO\ () \\
&appendFile &&:: FilePath \to String \to IO\ ()
\end{aligned}
$$

Variables Mutables (*Data.IORef*)

$$
\begin{aligned}
&\mathit{newIORef} &&:: a \to \mathit{IO}\ (\mathit{IORef}\ a)\\
&\mathit{readIORef} &&:: \mathit{IORef}\ a \to \mathit{IO}\ a\\
&\mathit{writeIORef} &&:: \mathit{IORef}\ a \to a \to \mathit{IO}\ ()\\
&\mathit{modifyIORef} &&:: \mathit{IORef}\ a \to (a \to a) \to \mathit{IO}\ ()
\end{aligned}
$$

Ejemplo:

$$
\begin{aligned}
\mathit{main} = \mathrm{do}\ & v \leftarrow \mathit{newIORef}\ 10\\
& \mathit{modifyIORef}\ v\ (+5)\\
& x \leftarrow \mathit{readIORef}\ v\\
& \mathit{print}\ x
\end{aligned}
$$

Imprime 15