Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Escuela de Ciencias Exactas y Naturales
Departamento de Ciencia de la Computación
Análisis de Lenguajes de Programación

Práctica 5

Functores

1. Demostrar que los siguientes tipos de datos son functores. Es decir, dar su instancia de la clase Functor correspondiente y probar que se cumplen las leyes de los functores.

a) **data** Pair a = P (a, a)

b) **data** Tree a = Empty | Branch a (Tree a) (Tree a)

c) **data** GenTree a = Gen a [GenTree a]

d) **data** Cont a = C $((a \to \mathsf{Int}) \to \mathsf{Int})$

2. Probar que las siguientes instancias no son correctas (no cumplen las leyes de los functores).

a) **data** Func a = Func $(a \to a)$

```
instance Functor Func where
  fmap g (Func h) = Func id
```

b) **data** Br b a = B b (a, a)

```
instance Functor (Br b) where
  fmap f (B x (y, z)) = B x (f z, f y)
```

Mónadas como substitución

3. Consideremos el siguiente (AST de un) lenguaje

data A a = Num Int | Sum (A a) (A a) | Mul (A a) (A a) | Res (A a) (A a) | Var_A a

para el cual implementamos sustitución simultánea:

$$
\begin{aligned}
&(\gg\!=_A) && :: \mathsf{A}\ a \to (a \to \mathsf{A}\ b) \to \mathsf{A}\ b \\
&\mathsf{Num}\ n \ \gg\!=_A v &&= \mathsf{Num}\ n \\
&\mathsf{Sum}\ \mathsf{t}\ u \gg\!=_A v &&= \mathsf{Sum}\ (\mathsf{t} \gg\!=_A v)\ (u \gg\!=_A v) \\
&\mathsf{Mul}\ \mathsf{t}\ u \ \gg\!=_A v &&= \mathsf{Mul}\ (\mathsf{t} \gg\!=_A v)\ (u \gg\!=_A v) \\
&\mathsf{Res}\ \mathsf{t}\ u \ \gg\!=_A v &&= \mathsf{Res}\ (\mathsf{t} \gg\!=_A v)\ (u \gg\!=_A v) \\
&\mathsf{Var}_\mathsf{A}\ a \ \gg\!=_A v &&= v\ a
\end{aligned}
$$

Dados los términos x = Var `"x"`, y = Var `"y"`, y z = Var `"z"`, dar el tipo y definición de g y h tal que:

Sum x (Mul y z) $\gg\!=_A$ g = Sum (Var 1) (Mul (Res (Var 3) (Var 1)) (Mul (Var 2) (Var 2)))

Sum x (Mul y z) $\gg\!=_A$ h = Sum (Var 1) (Res (Var 2) (Var 3))

¿Cuántas soluciones existen en cada caso?

4. Dado el tipo de datos

```
data BT a = IfBoton (Bool → BT a)
          | Beep (BT a)
          | Var_BT a
```

donde IfBoton detecta la pulsación de un botón, y Beep produce un beep, escribir un programa que haga un beep cada dos pulsaciones de botón.

5. Definimos el AST de un lenguaje entrada/salida de la siguente manera:

$$\textbf{data}\ \mathsf{ES}_\emptyset = \mathsf{Read}\ (\mathsf{Char} \to \mathsf{ES}_\emptyset) \mid \mathsf{Write}\ \mathsf{Char}\ (\mathsf{ES}_\emptyset)$$

Si el lenguaje lee un caracter de entrada y en base a eso decide como seguir o escribe un caracter y continua la ejecución. ¿Qué hacen los siguientes programas?

- $\mathsf{t}_1 = \mathsf{Read}\ (\lambda c \to \mathsf{Write}\ c\ (\mathsf{Write}\ c\ \mathsf{t}_1))$
- $\mathsf{t}_2 = \mathsf{Read}\ (\lambda c \to \mathsf{Write}\ \texttt{"("}\ (\mathsf{Write}\ c\ (\mathsf{Write}\ \texttt{")"}\ \mathsf{t}_2)))$
- $\mathsf{t}_3 = \mathsf{Write}\ \texttt{"("}\ (\mathsf{Read}\ (\lambda c \to \mathsf{Write}\ c\ (\mathsf{Write}\ \texttt{")"}\ \mathsf{t}_3)))$
- $\mathsf{t}_4 = \mathsf{Read}\ (\backslash\_ \to \mathsf{t}_4)$

6. Teniendo en cuenta los siguientes tipos:

$$\textbf{data}\ \mathsf{ES}\ a = \mathsf{Read}\ (\mathsf{Char} \to \mathsf{ES}\ a) \mid \mathsf{Write}\ \mathsf{Char}\ (\mathsf{Es}\ a) \mid \mathsf{Var}_{ES}\ a$$

Escribir un programa:

a) $\mathsf{writeChar} :: \mathsf{Char} \to \mathsf{ES}\ ()$ que escriba su argumento y finalice con una variable ().

b) $\mathsf{readChar} :: \mathsf{ES}\ \mathsf{Char}$ que lea un caracter y finalice con la variable cuyo nombre es el caracter leído.

c) Usando writeChar, escribir un programa $\mathsf{writeStr} :: \mathsf{String} \to \mathsf{ES}\ ()$ que imprima la cadena que se pasa como argumento.

7. Dada la siguiente implementación de la sustitución para términos:

$$\begin{aligned}
&(\gg\!=_{ES}) :: \mathsf{ES}\ a \to (a \to \mathsf{ES}\ b) \to \mathsf{ES}\ b\\
&\mathsf{Read}\ k \quad \gg\!=_{ES} v = \mathsf{Read}\ (\lambda c \to k\ c \gg\!=_{ES} v)\\
&\mathsf{Write}\ c\ \mathsf{t} \gg\!=_{ES} v = \mathsf{Write}\ c\ (\mathsf{t} \gg\!=_{ES} v)\\
&\mathsf{Var}_{ES}\ a \gg\!=_{ES} v = v\ a
\end{aligned}$$

¿Qué hace el siguiente programa?

$$\begin{aligned}
&f :: \mathsf{ES}\ \mathsf{String}\\
&f = \mathsf{readChar} \gg\!=_{ES} g\ \textbf{where}\ g\ \texttt{'\textbackslash n'} = \mathsf{Var}\ [\,]\\
&\qquad\qquad\qquad\qquad\quad g\ c \quad\ = f \gg\!=_{ES} \lambda xs \to \mathsf{Var}\ (c : xs)
\end{aligned}$$

Uso de Mónadas y notación do

8. Probar que toda mónada es un functor, es decir, proveer una instancia

```
instance Monad m ⇒ Functor m where
  fmap ...
```

y probar que las leyes de los functores se cumplen para su definición de fmap.

9. Definir las siguientes funciones:

a) $\mathsf{mapM} :: \mathsf{Monad}\ m \Rightarrow (a \to m\ b) \to [a] \to m\ [b]$, tal que $\mathsf{mapM}\ f\ xs$ aplique la función monádica f a cada elemento de la lista xs, retornando la lista de resultados encapsulada en la mónada.

b) $\mathsf{foldM} :: \mathsf{Monad}\ m \Rightarrow (a \to b \to m\ a) \to a \to [b] \to m\ a$, análogamente a foldl para listas, pero con su resultado encapsulado en la mónada. *Ejemplo:*

$$\begin{aligned}
\mathsf{foldM}\ f\ e_1\ [x_1, x_2, x_3] = \textbf{do}\ &e_2 \leftarrow f\ e_1\ x_1\\
&e_3 \leftarrow f\ e_2\ x_2\\
&f\ e_3\ x_3
\end{aligned}$$

10. Escribir el siguiente fragmento de programa monádico usando notación **do**.

$(m \gg\!= \lambda x \to h\ x) \gg\!= \lambda y \to f\ y \gg\!= \lambda z \to \mathsf{return}\ (g\ z)$

11. Escribir el siguiente fragmento de programa en términos de $\gg\!=$ y return.

```
do x ← (do z ← y
           w ← f z
           return (g w z))
   y ← h x 3
   if y then return 7
        else do z ← h x 2
                return (k z)
```

12. Escribir las leyes de las mónadas usando la notación **do**.

I/O Monádico

13. Escribir y **compilar** un programa (usando `ghc` en lugar de `ghci`) que imprima en pantalla la cadena `"Hola mundo!"`.

14. Escribir un programa interactivo que implemente un juego en el que hay que adivinar un número secreto predefinido. El jugador ingresa por teclado un número y la computadora le dice si el número ingresado es menor o mayor que el número secreto o si el jugador adivinó, en cuyo caso el juego termina. Ayuda: para convertir una String en Int puede usar la función read :: String → Int.

15. El juego nim consiste en un tablero de 5 filas numeradas de asteriscos. El tablero inicial es el siguiente:

1 : * * * * *
2 : * * * *
3 : * * *
4 : * *
5 : *

Dos jugadores se turnan para sacar una o mas estrellas de alguna fila. El ganador es el jugador que saca la última estrella. Implementar el juego en Haskell. Ayuda: para convertir una String en Int puede usar la función read :: String → Int.

16. Un programa pasa todos los caracteres de un archivo de entrada a mayúsculas y los guarda en un archivo de salida. Hacer un programa compilado que lo implemente tomando dos argumentos en la línea de comandos, el nombre de un archivo de entrada y el nombre de un archivo de salida.