Lógica Computacional 2017-2
Boletín 1: Introducción a `Haskell`

Lourdes del Carmen González Huesca
Roberto Monroy Argumedo
Fernando A. Galicia Mendoza

Facultad de ciencias, UNAM

Miércolesm 1 de Febrero del 2017

El siguiente documento indican los ejercicios realizados en la sesión 1 del laboratorio de lógica computacional.

1. Funciones no recursivas

1. En matemáticas podemos definir la función que devuelva el binomio cuadrado de dos enteros de la siguiente forma:

$$binomio : \mathbb{Z}^2 \to \mathbb{Z} \tag{1}$$
$$binomio(x, y) = x^2 + 2xy + y^2 \tag{2}$$

   Define la función *binomio* en tu archivo, la firma de la función es:
   `binomio :: Int -> Int -> Int`

2. Utilizando tuplas haz el ejercicio anterior.

3. Define un tipo de datos que represente el plano real y define una función que obtenga la distancia entre dos puntos.

4. Utilizando la función anterior, define las funciones que devuelvan el perímetro y área de un triángulo.
   Sugerencia: Definir un tipo de datos para representar el triángulo.

5. Para otras figuras geométricas, realiza el ejercicio anterior.

2. Tipos de datos recursivos

1. Dada la siguiente especificación indica que gramática formal le corresponde.

   Un número natural es el cero o bien si n es un número natural, entonces $S(n)$ es un número natural.

   Teniendo la gramática, define en `Haskell` el tipo que corresponde a la construcción anterior.

2. Dada la siguiente especificación indica que gramática formal le corresponde.

   Un número natural es el cero o bien si n es un número natural, entonces $2n$ es un número natural o bien $2n + 1$ es un número natural.

   Teniendo la gramática, define en `Haskell` el tipo que corresponde a la construcción anterior.

3. Dada la siguiente especificación indica que gramática formal le corresponde.

   Sea A un tipo cualquiera, una lista es: la lista vacía o bien si x es un elemento de tipo A y l es una lista de tipo A, entonces $(x : l)$ es una lista de tipo A

   Teniendo la gramática, define en `Haskell` el tipo que corresponde a la construcción anterior.

4. Dada la siguiente especificación indica que gramática formal le corresponde.

   Sea A un tipo cualquiera, una listaSnoc es: la listaSnoc vacía o bien si x es un elemento de tipo A y l es una listaSnoc de tipo A, entonces $(l : x)$ es una listaSnoc de tipo A

   Teniendo la gramática, define en `Haskell` el tipo que corresponde a la construcción anterior.

5. Dada la siguiente especificación indica que gramática formal le corresponde.

   Sea A un tipo cualquiera, una árbol binario es: el árbol binario vacío o bien si x es un elemento de tipo A e i, d son árboles binarios de tipo A, entonces (i, x, d) es un árbol binario de tipo A

   Teniendo la gramática, define en `Haskell` el tipo que corresponde a la construcción anterior.

6. Utilizando los ejercicios 2 y 3, define una gramática para árboles $n-$arios y defínelos en `Haskell`.
   Sugerencia: Se necesitará el uso de la estructura de datos lista.

3. Funciones recursivas

1. Define de manera recursiva (en caso de ser necesario) los operadores de sucesor, suma, predecesor, resta positiva y producto para ambas representaciones de los números naturales.

2. (*)Define de manera recursiva los operadores de sucesor, suma, predecesor, resta positiva y producto para ambas representaciones de los números naturales cuya construcción es pares e impares.

3. (**)Define una función recursiva que normalice los números naturales cuya construcción es pares e impares, es decir, que elimine todas las representaciones del cero distintas a cero. Por ejemplo 2(0) debe ser eliminada.
Sugerencia: El caso de $2x$ hacer un análisis sobre los resultados de la función.

4. Define de manera recursiva (en caso de ser necesario) los operadores de cabeza, último, cola, longitud y concatenación de listas.

5. (*)Define de manera recursiva (en caso de ser necesario) los operadores de cabeza, último, cola, longitud y concatenación de listasSnoc.

6. Define la función recursiva de mapeo para listas y listasSnoc.

7. Define la función recursiva de mapeo para árboles binarios.

8. Define una función recursiva que obtenga el número de nodos de un árbol binario.

9. Define una función recursiva que obtenga la suma de los elementos de una lista de números enteros.

10. (**) Define una función recursiva que obtenga el número de nodos de un árbol $n-$ario.
Sugerencia: Utiliza la función anterior.

11. Define las funciones recursivas que hagan los recorridos de preorden, postorden e inorden de árboles binarios.

12. (**) Define una función recursiva que obtenga la lista de los elementos de un árbol $n-$ario.
Sugerencia: Utiliza uno de los recorridos.