Programación Funcional

Curso 2016/2017. Ejercicios – Sesión práctica (Lote 3)

Esto es un repertorio de actividades sugeridas para la sesión de prácticas

1. Estudia si puedes definir mediante `foldr` o `foldl`, en lugar de mediante recursión explícita, las siguientes funciones: `last`, `reverse`, `all`, `any`, `concat`, `minimum`, `map`, `filter`, `takeWhile`, `(++)`.

2. Programa, indicando los tipos, las siguientes variantes de `foldl` y `foldr`, que operan con listas no vacías y no usan valor acumulado inicial:

   - $foldr1\ \bigoplus\ [x_1,\ldots,x_n] = x_1 \bigoplus x_2 \bigoplus \ldots \bigoplus x_n$ (con $\bigoplus$ asociando por la derecha)
   - $foldl1\ \bigoplus\ [x_1,\ldots,x_n] = x_1 \bigoplus x_2 \bigoplus \ldots \bigoplus x_n$ (con $\bigoplus$ asociando por la izquierda)

3. Revisa ejercicios anteriores para ver en cuáles de ellos pueden ser de utilidad las listas intensionales.

4. Programa las siguientes funciones, teniendo en cuenta listas intensionales por si son de utilidad.

```
prefixes,suffixes,sublists,parts,perms:: [a] -> [[a]]
-- (prefixes xs) devuelve las lista de todos los prefijos de xs.
-- (suffixes xs) devuelve las lista de todos los sufijos de xs.
-- (sublists xs) devuelve las lista de todas las sublistas de xs.
-- (partss xs) devuelve las lista de todos las partes (subconjuntos) de xs.
-- (perms xs) devuelve la lista de todas las permutaciones de xs.
-- (inits xs) devuelve las lista de todos los segmentos iniciales de xs,
--  en orden de longitud creciente.

combinaciones,variaciones:: Int -> [a] -> [[a]]
-- (combinaciones n xs) devuelve la lista de todas las combinaciones de
--                      elementos de xs tomados de n en n
-- Algo similar para variaciones
sumandos:: Int -> [[Int]]
-- sumandos n devuelve la lista de todas las descomposiciones en sumandos positivos de n
-- Ejemplo: sumandos 3 = [[1,1,1],[1,2],[2,1],[3]]
-- O bien, si no queremos resultados que sean permutaciones unos de otros:
--   sumandos 3 = [[1,1,1],[1,2],[3]]
```

5. Programa el producto escalar de vectores y el producto de matrices, suponiento que los vectores se representan como `[Float]` y las matrices como listas de vectores.

6. Elimina las listas intensionales de las siguientes definiciones, usando *map, filter* y *concat*:

```
f n     = [x*x | x <- [1..n], mod x 2 == 0]
g n m   = [x+y | x <- [1..n], y <- [x..m]]
h p n m = [x+y | x <- [1..n], p (n-x), y <- [x..m]]
```

7. Elimina, reemplazándolas por funciones auxiliares no locales, las definiciones locales y la λ-abstracción de la definición siguiente:

```
f x y = map (\u -> (g u,g (u+1))) y
        where z = x * last y
              g u = (x+z)*u
```