Programación Funcional

Trabajo Práctico Nro. 5

Temas: Demostraciones. Propiedades de programas: terminación, equivalencia. Inducción. Recursión.

Bibliografía relacionada:

- Simon Thompson. The craft of Functional Programming. Addison Wesley, 1996.
- L.C. Paulson. ML for the working programmer. Cambridge University Press, 1996.
- Bird, Richard. Introduction to funtional programming using Haskell. Prentice Hall, 1998 (Second Edition).

1. Definir recursivamente las siguientes funciones y dar sus tipos:

`nextDiv`,	toma dos números `x`, `y` y devuelve el primer divisor de `y` mayor que `x`.
`sumDivs`,	toma un número y devuelve la suma de sus divisores.
`power`,	que toma un número y un natural, y devuelve el resultado de elevar el primero a la potencia dada por el segundo.
`dividesTo`,	de la práctica 1.
`sum`,	tal que `sum f i j` $= \sum_{k=i}^{j} f(k)$
`prime`,	que decide si un número es primo (es decir, si tiene sólo 2 divisores positivos 1 y sí mismo).
`phi`,	que toma un entero i y devuelve el i-ésimo número primo.

2. Demostrar que

   a) `flip (curry f) = curry (f . swap)`

   b) Sean i j k tales que $i \leq j \leq k$, vale
   `sum f i j + sum f (j+1) k = sum f i k`

   c) `prime x` *sii* `nextDiv 1 x == x`

3. Enumere las propiedades que tiene un conjunto definido por inducción estructural.

4. Para los casos en que sea posible, demostrar la terminación de las funciones del ejercicio 1. ¿En qué principios justifica sus afirmaciones?

5. Demostrar considerando las definiciones de la práctica 4:

   a) `pairs . squares = squares . pairs`, donde `squares` eleva al cuadrado los elementos de una lista, y `pairs` devuelve sólo los elementos pares de una lista de números.

   b) `((‘mod‘ n) . sum) (remainders n xs) = (sum xs) ‘mod‘ n`, para todo `n` > 0
   Ayuda: Asuma verdadero el siguiente lema:
   `(n + m) ‘mod‘ p = ((n ‘mod‘ p) + (m ‘mod‘ p)) ‘mod‘ p`

6. Demostrar la siguiente propiedad: `sum` $\mathtt{x_s} \leq$ `len` $\mathtt{x_s} *$ `maxl` $\mathtt{x_s}$, siendo `maxl` la función

```
maxl [] = 0
maxl (x:xs) = x ‘max‘ maxl xs
```

   y considerando que $\mathtt{x_s}$ es una lista finita de números naturales.

7. ⊛ Dada la función:

```
hailstone n = if (n<=1)
              then 0
              else if (n ‘mod‘ 2 == 0) then (n ‘div‘ 2)
                                       else (3*n+1)
```

   definir una función `hail`, que toma un entero `n` y devuelve el mínimo i tal que

$$\mathtt{hailstone}^i\ \mathtt{n} = 0$$

$$\underbrace{(\mathtt{hailstone}\ (\ldots(\ \mathtt{hailstone}}_{i\ veces}\ \mathtt{n}))) = 0$$

   ¿Para que valores la evaluación de `hail` termina?

Ejercicios complementarios

8. Recordemos el algoritmo, atribuído a Euclides, para calcular el máximo divisor entre dos números:

   Dados a y b, con $a \geq b$, sabemos que existen únicos enteros q_0 y r_0, con $q_0 \geq 0$ y $0 \leq r_0 < b$, tales que $a = bq_0 + r_0$. Con las mismas condiciones se puede formar la secuencia

$$\begin{array}{rcrcl} a & = & bq_0 & + & r_0 \\ b & = & r_0q_1 & + & r_1 \\ r_0 & = & r_1q_2 & + & r_2 \\ r_1 & = & r_2q_3 & + & r_3 \\ & & \cdots & & \\ r_{n-2} & = & r_{n-1}q_n & & \end{array}$$

   La secuencia termina cuando $r_n = 0$, siendo el mcd de a y b, r_{n-1}.

a) Implementar una función `mcd` que calcule el máximo común divisor entre dos enteros dados (Utilizar el algoritmo de Euclides).

b) Demostrar que la implementación dada termina para todo par de enteros.

9. ⍟ Dadas las funciones `reverse` y `rev` definidas de la siguiente manera:

```
reverse, rev :: [a] -> [a]
reverse  []    = []
reverse (x:xs) = reverse xs ++ [x]
rev xs = fastrev [] xs
         where fastrev xs []     = xs
               fastrev xs (y:ys) = fastrev (y:xs) ys
```

a) Demuestre que se cumple:

Lema 1 `fastrev ys xs = fastrev [] xs ++ ys`, para todas listas `xs,ys`

b) Demuestre que `reverse` = `rev`.

Propiedades útiles:

1. $\forall$ n, m ϵ N, n $\leq$ max (n,m)
2. x es par $\Leftrightarrow$ $x * x$ es par.
3. $(x \bmod n) \bmod n = x \bmod n$