Práctico 2: Especificación, Derivación y Verificación de Programas Funcionales

Algoritmos y Estructuras de Datos I
2^{do} cuatrimestre 2022

Esta guía tiene como objetivo obtener las habilidades necesarias para llevar adelante un proceso de derivación o verificación de programas recursivos a partir de especificaciones formales.

Los ejercicios de cálculo de programas tienen una dificultad creciente: en los primeros, la derivación o verificación se obtiene de manera *directa* a través de una demostración inductiva. Los ejercicios sucesivos son más complejos y requieren el uso de modularización.

Es importante que recuerdes, al finalizar la derivación, escribir la definición de la función con su tipo y todas sus claúsulas, como en el Ejercio 1.

1. Dado el **programa**

$$\begin{array}{|l}
\text{sum} : [Num] \to Num \\
\hline
\text{sum}.[\,] \doteq 0 \\
\text{sum}.(x \triangleright xs) \doteq x + sum.xs
\end{array}$$

   a) ¿Qué hace esta función? Escriba en lenguaje natural el **problema** que resuelve.
   b) Escriba una **especificación** de la función con una expresión cuantificada.
   c) **Verifique** que esta especificación vale para toda lista xs.
   d) Ahora **derive** la definición de la función a partir de su especificación.
   ¿Esta derivación es parecida a la demostración en el punto $1c$?

2. A partir de las siguientes especificaciones, dar el tipo de cada función y *derivar* las soluciones algorítmicas correspondientes.

   a) $\text{sum\_cuad}.xs = \langle \sum i \,:\, 0 \leq i < \#xs \,:\, xs\,!\,i * xs\,!\,i \rangle$
   b) $\text{iga}.e.xs = \langle \forall\, i \,:\, 0 \leq i < \#xs \,:\, xs\,!\,i = e \,\rangle$
   c) $\text{exp}.x.n = x^n$
   d) $\text{sum\_par}.n = \langle \sum i \,:\, 0 \leq i \leq n \wedge par.i \,:\, i \rangle$
   donde $par.i \doteq i \bmod 2 = 0$.
   e) $\text{cuántos}.p.xs = \langle \text{N}\ i \,:\, 0 \leq i < \#xs \,:\, p.(xs\,!\,i) \rangle$
   f) $\text{busca}.e.xs = \langle \text{Min}\ i \,:\, 0 \leq i < \#xs \wedge xs\,!\,i = e \,:\, i \,\rangle$

3. Para todos los ítems del ejercicio anterior, dar un ejemplo de uso de la función, es decir: elegir valores concretos para los parámetros y calcular el resultado usando la solución algorítmica obtenida. Las listas deben tener por lo menos tres elementos.

4. Derivar las siguientes funciones.

   a) $\text{sum\_pot} : Num \to Nat \to Num$ computa la suma de potencias de un número, esto es

$$\text{sum\_pot}.x.n = \left\langle \sum i \,:\, 0 \leq i < n \,:\, x^i \right\rangle \;.$$

   b) cos' : $Nat \to Num \to Num$ computa la aproximación del coseno del segundo argumento.

$$cos'.n.x \doteq \left\langle \sum i \,:\, 0 \leq i \leq n \,:\, (-1)^i * \frac{x^{2*i}}{(2*i)!} \right\rangle$$

   Ayuda: Modularizar dos veces. La segunda con la función *exp* del ejercicio $2c$ y factorial.

c) cubo : $Nat \to Nat$ computa el cubo ($\text{cubo}.x = x^3$) de un número natural x utilizando únicamente sumas.

Ayuda: Usar inducción y modularización una o más veces.

d) prod_suf: $[Num] \to Bool$ decide si existe un elemento igual al producto de los elementos que le siguen:

$$\text{prod\_suf}.xs = \left\langle \exists i\ : 0 < i \leq \#xs :\ \left\langle \prod j\ : 0 \leq j < \#(xs{\downarrow}i) :\ (xs{\downarrow}i)\,!\,j \right\rangle = xs\,!\,(i-1) \right\rangle$$

5. Especificar formalmente utilizando cuantificadores cada una de las siguientes funciones descriptas informalmente. Luego, *derivar* soluciones algorítmicas para cada una.

   a) iguales : $[A] \to Bool$, que determina si los elementos de una lista de tipo A son todos iguales entre sí. Suponga que el operador $=$ es la igualdad para el tipo A.

   b) minimo : $[Int] \to Int$, que calcula el mínimo elemento de una lista **no vacía** de enteros.

   Nota: 1 *La función no debe devolver* $\pm\infty$.

   c) creciente : $[Int] \to Bool$, que determina si los elementos de una lista de enteros están ordenados en forma creciente.

   d) prod : $[Num] \to [Num] \to Num$, que calcula el producto entre pares de elementos en iguales posiciones de las listas y suma estos resultados (producto punto). Si las listas tienen distinto tamaño se opera hasta la última posición de las más chica.

6. Para complementar el ejercicio 4*b*. Considerando que $Punto = (Num, Num)$ y $Seg = (Punto, Punto)$, defina las siguientes funciones:

   a) punto: $(Num \to Num) \to Num \to Punto$ que dada una función y un valor nos da el punto correspondiente a $(x, f\,x)$.

   b) dist: $(Num \to Num) \to (Num \to Num) \to Num \to Seg$ que dadas dos funciones f, g y un valor nos da el segmento de recta dados por los puntos en f y g.

   c) curva: $(Num \to Num) \to [Num] \to [Punto]$ que dada una función f y una lista xs de valores devuelve la lista de puntos $(x, f\,x)$ para cada elemento x de la lista xs.

   d) angulos: $Nat \to [Num]$ que en el argumento n devuelve la lista de $2n+1$ ángulos entre $-2*\pi$ y $2*\pi$: $[-2\pi, -2\pi\frac{n-1}{n}, \ldots, 0, 2\pi\frac{1}{n}, \ldots, 2\pi\frac{n-1}{n}, 2\pi]$.

Ayuda: No hay nada difícil y son más del estilo de cosas que hicieron en IntroAlg.