Algoritmos y estructuras de datos I - Recursado - 2010
Práctico 2: Especificaciones, Inducción, Funciones recursivas y Derivación

El primer objetivo de esta guía (ejercicios 1 al 3) es consolidar el manejo de expresiones cuantificadas, el énfasis reside en la comprensión del significado de las expresiones cuantificadas (manejo semántico). El ejercicio 5 aborda la tarea de definir ***informalmente*** *funciones recursivas, con el objetivo de conseguir la habilidad de resolver problemas a través del razonamiento inductivo. Esto ayuda a la comprensión del modelo computacional funcional. Otro objetivo muy importante es la maduración del concepto de inducción (ejercicios 6 y 7), en primera instancia utilizado en propiedades de los números naturales, y extendido sobre listas (inducción* ***estructural****). Se busca mejorar la habilidad para utilizar esta técnica de prueba que es central para la tarea de cálculo de programas recursivos, es decir, la derivación de un programa recursivo a partir de una especificación formal de la misma. Tarea abordada a partir del ejercicio 9.*

1. Exprese en lenguaje natural cada una de las siguientes sentencias formales, donde xs y ys son listas de enteros:

   a) $\langle \forall x, z : x \in Num \wedge z \in Num \wedge x \neq z : \langle \exists y : y \in Num : x < y < z \rangle \rangle$

   b) $\langle \exists i : 0 \leq i < \#xs : xs.i = 0 \rangle$

   c) $\langle \forall i : 0 \leq i < \#xs : xs.i \geq 0 \rangle$

   d) $\langle \exists i : 0 < i < \#xs : xs.(i-1) < xs.i \rangle$

   e) $\langle \exists i : 0 \leq i < \#xsmin\#ys : xs.i \neq ys.i \rangle$

2. Exprese utilizando cuantificadores las siguientes sentencias del lenguaje natural:

   a) En la lista xs solo ocurren valores que anulan la funcion f.

   b) La lista xs es un segmento inicial de la lista ys.

   c) La lista xs posee un segmento **no** inicial y **no** final cuyos valores son mayores a los valores del resto de la misma.

   d) La lista xs de numeros enteros tienen la misma cantidad de elementos pares e impares.

3. Especifique utilizando cuantificadores cada una de las siguientes funciones descriptas en lenguaje natural:

   a) $minimo : [Int] \to Int$, que calcula el mínimo elemento de una lista de enteros.

   b) $iguales : [A] \to Bool$, 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.

   c) $listas\_iguales : [A] \to [A] \to Bool$, que determina si dos listas son iguales, es decir, contienen los mismos elementos en las mismas posiciones respectivamente.

   d) $listas\_iguales? : [A] \to [A] \to Bool$, que determina si dos listas tienen los mismos elementos.

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

   f) $pos\_min : Int \to [Num] \to Bool$, que determina si el n-ésimo elemento de una lista de números contiene al mínimo valor de la misma.

4. Dada la definición recursiva de la función $func$:

$$func.[\,] = 0 \quad func.(x \triangleright xs) = x^2 + func.xs$$

   (a) Explicar en palabras qué calcula.

   (b) Especificarla (usando cuantificadores).

5. Defina recursivamente cada una de las funciones descriptas en el ejercicio 3.

6. (*)Dada la siguiente definición recursiva para la funcion $repetir : Nat \to Nat \to [Nat]$:

$$repetir.0.x \doteq [\,]$$
$$repetir.(n+1).x \doteq x \triangleright repetir.n.x$$

demostrar que se satisface $\#repetir.n.x = n$.

7. (*)Para las funciones $sum : [Num] \to Num$ y $duplica : [Num] \to [Num]$ definidas recursivamente como:

$$\begin{array}{rcl} sum.[\,] & \doteq & 0 \\ sum.(x \triangleright xs) & \doteq & x + sum.xs \end{array} \qquad \begin{array}{rcl} duplica.[\,] & \doteq & [\,] \\ duplica.(x \triangleright xs) & \doteq & (2 * x) \triangleright duplica.xs \end{array}$$

demostrar que se satisface $sum.(duplica.xs) = 2 * sum.xs$.

8. Las funciones *abr* y *cie*, cuentan –respectivamente– la cantidad de paréntesis que abren – o cierran – hay en una lista, digamos en xs y sus especificaciones son las siguientes:

$$\begin{array}{rcl} abr.xs & = & < Ni : 0 \le i < \#xs : xs.i = \text{`(`} > . \\ cie.xs & = & < Ni : 0 \le i < \#xs : xs.i = \text{`)`} > . \end{array}$$

Corroborar que son correctas las siguientes definiciones recursivas:

$$\begin{array}{rcl} abr.[] & = & 0 \\ abr.(\text{`(`} \triangleright xs) & = & 1 + abr.xs \\ abr.(\text{`)`} \triangleright xs) & = & abr.xs \end{array} \qquad \begin{array}{rcl} cie.[] & = & 0 \\ cie.(\text{`(`} \triangleright xs) & = & cie.xs \\ cie.(\text{`)`} \triangleright xs) & = & 1 + cie.xs \end{array}$$

9. Sea $m : [Int] \mapsto Int$ la función que devuelve el mínimo elemento de una lista de enteros. Escribir la especificación y a partir de ésta obtener una definición recursiva para m.

10. Derivar una definición recursiva para cada una de las siguientes funciones:

    (a) $iguales : [A] \mapsto Bool$, definida en el ejercicio 3.

    (b) $creciente : [Int] \mapsto Bool$, definida en el ejercicio 3.

    (c) $P : [Num] \mapsto Bool$, la cual, dada una lista de naturales, determina si algún elemento de la lista es igual a la suma de todos los otros.

11. Calcular la función que eleva un número natural al cubo, usando sólo sumas. La especificación es la obvia: $f.x = x^3$. Sugerencia: usar inducción, modularización varias veces y luego técnica de tuplas para mejorar la eficiencia.

12. *Torres de Hanoi.* Se tienen tres postes -0, 1 y 2- y n discos de distinto tamaño. En la situación inicial se encuentran todos los discos ubicados en el poste 0 en forma decreciente según el tamaño (el más grande en la base).

    El problema consiste en llevar todos los discos al poste 2, con las siguientes restricciones:

    (i) Se puede mover sólo un disco por vez (el que está más arriba en algún poste).

    (ii) No se puede apoyar un disco sobre otro de menor tamaño.

    Sea $B = \{0, 1, 2\}$.

    (a) Definir una función $hanoi : B \mapsto B \mapsto B \mapsto Nat \mapsto [(B, B)]$ tal que $hanoi.a.b.c.n$ calcule la secuencia de movimientos para llevar n discos del poste a al poste c, pasando por el poste b.

    a)

    (b) Derive una función recursiva para *hanoi*.
    Ejemplo: $hanoi.0.1.2.2 = [(0,1),\ (0,2),\ (1,2)]$

    (c) (**)Derive una función recursiva para la función $t : B \to B \to B \to Nat \to ([(B,B)],\ [(B,B)],\ [(B,B)])$ especificada como:

$$t.a.b.c.n = (hanoi.a.b.c.n,\ hanoi.b.c.a.n,\ hanoi.c.b.a.n)$$

    ¿Cuál es el propósito de esta función? Piense en la eficiencia de *hanoi*.
    Ayuda: Puede ser útil calcular manualmente $t.a.b.c.0$, $t.a.b.c.1$ y $t.a.b.c.(n+2)$

13. (**) Derivar una definición recursiva para la función $f : Int \mapsto [Num] \mapsto Bool$, que determina si el k-ésimo elemento de una lista de números aloja el mínimo valor de la misma.

14. Dada la función $f : [Num] \mapsto [Num] \mapsto Num$, especificada como sigue:

$$f.xs.ys =< \text{Min}\, i, j : 0 \leq i < \#xs \wedge 0 \leq j < \#ys : |xs.i - ys.j| > .$$

    (a) Decir en palabras qué calcula.

    (b) Derivar una definición recursiva. Ayuda: Es necesario modularizar.

15. Derivar una definición recursiva para la función $l : [Char] \mapsto [Char] \mapsto Bool$, especificada como sigue:

$$l.xs.ys =< \exists\, as, bs, c, cs : xs = as + +bs \wedge ys = as + +(c \triangleright cs) : bs = [] \vee bs{,}0 < c > .$$

    Decir en palabras cuándo $l.xs.ys$ es $true$.

16. Sea fib la función de Fibonacci, definida (recursivamente) como sigue:

$$\begin{array}{rcl} fib{,}0 & \doteq & 0 \\ fib{,}1 & \doteq & 1 \\ fib.(n+2) & \doteq & fib.n + fib.(n+1) \end{array}$$

    Calcular la función de Fibolucci, $Fbl : Nat \mapsto Nat$, especificada por:

$$Fbl.n =< \sum i : 0 \leq i < n : fib.i * fib.(n-i) > .$$