Práctico 2: Especificación, derivación y verificación de programas funcionales

Algoritmos y Estructuras de Datos I
2$^{\text{do}}$ cuatrimestre 2015

El objetivo general de esta guía es desarrollar nuestras capacidades para leer y escribir especificaciones de programas, encontrar programas recursivos a partir de ellas utilizando la técnica de derivación, y ademas comprender qué hacen ciertos programas y verificar propiedades sobre ellos. Las demostraciones por inducción son fundamentales para estas actividades.

1. Retomemos el ejercicio 8 del práctico 1. Una solución al problema de la especificación de la función *sumatoria* es la siguiente:

$$sumatoria.xs = \langle \sum i \ : 0 \leq i < \#xs : \ xs.i \ \rangle$$

   Y un programa recursivo que computa tal función es:

$$sumatoria : [Num] \to Num$$

$$\begin{aligned} sumatoria.[\,] &\doteq 0 \\ sumatoria.(x \triangleright xs) &\doteq x + sumatoria.xs \end{aligned}$$

   Demostrá por inducción que, para una lista arbitraria xs, la definición recursiva de *sumatoria* satisface la especificación dada más arriba.

2. Expresá en lenguaje natural el significado que le das a cada una de las siguientes fórmulas, donde siempre suponemos que x e y son números, y xs y ys son listas de números:

   a) $\langle \exists x :: \langle \forall y :: x \leq y \rangle \rangle$
   b) $\langle \forall y :: \langle \exists x :: x \leq y \rangle \rangle$ ¿Cuál es la diferencia con la anterior?
   c) $\langle \forall x, z : x < z : \langle \exists y :: x < y < z \rangle \rangle$
   d) $\langle \exists i : 0 \leq i < \#xs : xs.i = 0 \rangle$
   e) $\langle \forall i : 0 \leq i < \#xs : xs.i \geq 0 \rangle$
   f) $\langle \exists i : 0 < i < \#xs : xs.(i-1) > xs.i \rangle$
   g) $\#xs \neq \#ys \vee \langle \exists i : 0 \leq i < \#xs \wedge 0 \leq i < \#ys : xs.i \neq ys.i \rangle$

3. Enunciá los teoremas de Separación de Término para el cuantificador Σ. Calculá a mano cada una de las reglas para una valor particular, digamos $n = 5$, para darte una idea de como "funcionan".

4. Estamos en medio del proceso de derivación de un programa recursivo $suma\_hasta : Num \to Num$, especificado como

$$suma\_hasta.n = \langle \sum i \ : 0 \leq i < n : \ i \ \rangle$$

   que suma los primeros n números naturales (sin incluir n). En el caso base encontramos

$$suma\_hasta.0 \doteq 0$$

   Demostrá que esto es correcto. Ahora, en el caso inductivo, nos estancamos después de realizar los siguientes pasos:

$$\begin{aligned} & suma\_hasta.(n+1) \\ \equiv & \ \{ \text{ especificación } \} \\ & \langle \sum i \ : 0 \leq i < n+1 : \ i \ \rangle \\ \equiv & \ \{ \text{ separación de término } \} \\ & 0 + \langle \sum i \ : 0 \leq i < n : \ i+1 \ \rangle \end{aligned}$$

   ¿Está bien aplicada la regla del último paso? ¿Podemos continuar la derivación hasta encontrar una definición recursiva? Si te parece que si, dale para adelante. Si no, propone un camino alternativo.

5. Considerá el siguiente programa recursivo:

$$\begin{aligned} pp.[\,] &\doteq 1 \\ pp.(x \triangleright xs) &\doteq x * x * pp.xs \end{aligned}$$

¿Cuál es el tipo de la función computada por pp? Escribí una especificación para pp y verificá que es correcta, por inducción en xs.

6. Considerá la función $f : A \to [A] \to Bool$, especificada como

$$f.e.xs = \langle\, \forall\, i \;:\; 0 \leq i < \#xs :\; xs.i = e \,\rangle$$

¿Qué hace esta función? Derivá un programa recursivo a partir de su especificación.

7. Considerá la función de exponenciación:

$$exp : Nat \to Nat \to Nat$$

$$exp.x.n \doteq x^n$$

¿Esta es una especificación aceptable? En cualquier caso, encontrá una definición recursiva para exp que sólo utilice la $+$ como operación elemental.

8. Considerá el siguiente programa recursivo:

$$del : A \to [A] \to [A]$$

$$\begin{aligned} del.n.[\,] &\doteq [\,] \\ del.n.(x \triangleright xs) &\doteq (n = x \to del.n.xs \\ &\quad \Box\ n \neq x \to x \triangleright del.n.xs \\ &\quad) \end{aligned}$$

¿Qué hace? Demostrá que

$$\langle\, \forall\, i \;:\; 0 \leq i < \#(del.n.xs) :\; (del.n.xs).i \neq n \,\rangle$$

Ayuda: Planteá dos casos inductivos, de acuerdo a la guarda del caso recursivo de del.

La especificación anterior no caracteriza completamente lo que hace la función. ¿Cómo especificarías que $del.n.xs$ mantiene los valores que ocurren en xs que son diferentes a n? ¿Ahora sí la especificación dice todo lo que hace la función?

9. Considerá el siguiente programa recursivo:

$$cuantos : (A \to Bool) \to [A] \to Num$$

$$\begin{aligned} cuantos.T.[\,] &\doteq 0 \\ cuantos.T.(x \triangleright xs)) &\doteq (T.x \to 1 + cuantos.T.xs \\ &\quad \Box\ \neg T.x \to cuantos.T.xs \\ &\quad) \end{aligned}$$

¿Cómo especificarías esta función? Una vez que creas tener la especificación correcta, verificá el programa anterior respecto a ella.

10. A partir de las siguientes especificaciones, da el tipo de cada función y *derivá* programas recursivos para cada una de ellas:

   a) $suma\_potencias.x.n = \langle \sum i \;:\; 0 \leq i < n :\; x^i \,\rangle$.
   Recordá el ejercicio 7

   b) $suma\_pares.n = \langle \sum i \;:\; 0 \leq i \leq n \wedge par.i \;:\; i \rangle$,
   donde par es una función dada, que satisface $par.n \equiv \langle\, \exists\, i \;::\; n = 2 * i \,\rangle$

   c) $busca.e.xs = \langle \text{Min}\ \ i \;:\; 0 \leq i < \#xs \wedge xs.i = e :\; i \,\rangle$.

11. Dada el siguiente programa recursivo:

$$fib : Nat \to Nat$$

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

demostrá que la siguiente propiedad es válida (para todo $n \in Nat$):

$$fib.n < 2^n$$

12. El siguiente programa recursivo calcula el mínimo elemento de una lista **no vacía** de enteros:

$$minimo : [Num] \to Num$$

$$\begin{aligned} minimo.[x] &\doteq x \\ minimo.(x \triangleright y \triangleright xs)) &\doteq (x < y \to minimo.(x \triangleright xt) \\ &\quad \Box\ x \geq y \to minimo.(y \triangleright xs) \\ &\quad) \end{aligned}$$

¿Cómo podrías simplificarlo para eliminar el uso de análisis por casos? Hay al menos dos estrategias principales para demostrar que tu programa (simplificado) es equivalente al programa anterior ¿se te ocurren cuales son?

13. Considerá una función *iguales* : $[A] \to Bool$, que determina si los elementos de una lista de tipo A son todos iguales entre sí. La siguiente especificación para tal función ¿es correcta?:

$$\langle\, \forall\, i\ : 0 \leq i < \#xs :\ xs.i = xs.(i+1)\, \rangle$$

Si no lo es, corregila. En cualquier caso, derivá un programa recursivo que compute esta función.

14. Especificá formalmente y luego derivá un programa recursivo para la función *creciente* : $[Int] \to Bool$, que determina si los elementos de una lista de enteros están ordenados en forma creciente.

15. Especificá formalmente y luego derivá un programa recursivo para la función *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). Considerá cuidadosamente la situación en que las listas tienen distinto tamaño.

16. Considerá la sentencia "El valor e ocurre en las posiciones pares de la lista xs". ¿Cuál de las siguientes especificaciones formales carateriza tal sentencia?

    a) $\langle\, \exists\, i\ : 0 \leq i < \#xs \wedge par.i :\ xs.i = e\, \rangle$

    b) $\langle\, \forall\, i\ : 0 \leq i < \#xs \wedge par.i :\ xs.i = e\, \rangle$

    Considerá ahora la sentencia "El valor e ocurre únicamente en las posiciones pares de la lista xs". ¿Cómo la especificarías?

17. Considerá el siguiente programa recursivo:

$$rev : [A] \to [A]$$

$$\begin{aligned} rev.[\,] &\doteq [\,] \\ rev.(x \triangleright xs) &\doteq rev.xs +\!\!+ [x] \end{aligned}$$

¿Qué hace este programa? Considerá ahora la siguiente propiedad:

$$rev.(xs +\!\!+ ys) = rev.ys +\!\!+ rev.xs$$

¿Cómo expresarías en lenguaje natural su significado? Demostrá que es válida.

18. La función $promedio : [Num] \to Num$, que calcula el promedio de una lista de números, puede especificarse como sigue:

$$promedio.xs = \langle \sum i \ : 0 \leq i < \#xs : \ xs.i \, \rangle \, / \, \langle \mathrm{N}\, i \ : 0 \leq i < \#xs : \ True \, \rangle$$

El numerador caracteriza una función sobre la que ya hemos trabajado; el denominador una función muy conocida ¿cuál es? ¿Cuál sería un programa que calcule *promedio*? ¿Cómo demostrás que es correcto?

19. Especificá formalmente la función $divide : Nat \to Nat \to Bool$, que determina si el primer parámetro divide al segundo. Luego encontrá un programa que la compute y demostrá su corrección.

20. Utilizando la función anterior, escribí (sin necesidad de derivar) un programa $es\_primo : Nat \to Bool$, que determina si un número es primo. ¿Cómo especificarías el programa *es_primo*?

21. Expresá formalmente las siguientes sentencias:

    a) El elemento e ocurre un número par de veces en la lista xs.

    b) Si e ocurre en la lista xs, entonces l ocurre en alguna posicion anterior en la misma lista.

    c) Existe un elemento de la lista xs que es estrictamente mayor a todos los demás.

    d) Cualquier valor x que anula la funcion f (es decir que $f.x = 0$) ocurre en la lista xs.

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

22. Para cada una de las siguientes especificaciones de funciones, escribí el tipo de la función y derivá un programa recursivo que la compute:

    a) $psum.xs = \langle \forall i \ : 0 \leq i \leq \#xs : \ sumatoria.(xs{\uparrow}i) \geq 0 \, \rangle$

    b) $sum8.xs = \langle \exists i \ : 0 \leq i \leq \#xs : \ sumatoria.(xs{\uparrow}i) = 8 \, \rangle$

23. Especificá formalmente las siguientes funciones, y luego derivá programas recursivos que las computen:

    a) $cuad : Nat \to Bool$, que dado un natural determina si es el cuadrado de un número.

    b) $n8 : [Num] \to Nat$, que cuenta la cantidad de segmentos iniciales de una lista cuya suma es igual a 8.

24. Describí en lenguaje natural el significado de las siguientes especificaciones:

    a) $\langle \exists bs \ :: \ ys = xs +\!\!+ bs \, \rangle$

    b) $\langle \exists as, bs \ :: \ ys = as +\!\!+ xs +\!\!+ bs \, \rangle$

    Luego encontrá programas recursivos para cada una de ellas, y demostrá que son correctos.

25. Especificá formalmente las siguientes sentencias:

    a) La lista xs es un segmento final de la lista ys.

    b) Las listas xs y ys tienen en común un segmento.

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

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

26. Derivá las siguientes funciones:

    a) Mínima sumatoria de un segmento:

$$sumin.xs = \langle \mathrm{Min} \ \ as, bs, cs \ : xs = as +\!\!+ bs +\!\!+ cs : \ sumatoria.bs \, \rangle$$

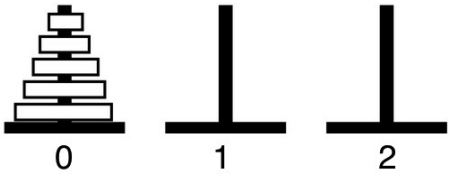
    b) Cantidad de segmentos que suman "eso":

$$cant\_8.xs = \langle \mathrm{N}\, as, bs, cs \ : xs = as +\!\!+ bs +\!\!+ cs : \ 8 = sumatoria.bs \, \rangle$$

    c) Máxima longitud de segmento de iguales a e, donde f es la función del ejercicio 6:

$$max\_iguales.e.xs = \langle \mathrm{Max} \ \ as, bs, cs \ : xs = as +\!\!+ bs +\!\!+ cs \wedge f.e.bs : \ \#bs \, \rangle$$

27. La Torre de Hanoi es un *puzzle* de ingenio, que consta de tres postes (numerados 0, 1 y 2), y n discos, todos de diferente tamaño, ubicados en alguno de los postes. Inicialmente todos los discos están ubicados en el poste 0, ordenados de mayor a menor, con el disco más grande en la base formando una torre:



El problema consiste en trasladar toda la torre al poste 2, siguiendo siempre las siguientes reglas:

a) Sólo se puede mover un disco a la vez.

b) Sólo se puede mover el disco que se encuentra más arriba en algún poste.

c) No se puede colocar un disco sobre otro de menor tamaño.

Así, una solución al problema se puede describir como una lista de *movimientos*. Por las reglas anteriores, cada movimiento se puede caracterizar por una tupla de valores en $\{0, 1, 2\}$; el primer valor indica el poste de donde sale el disco que se encuentra más arriba, y el segundo valor indica el poste donde va a parar.

Por ejemplo, si consideramos una torre con un único disco (vaya torre!), la mejor solución al problema es mover directamente el disco del poste 0 al poste 2, lo que puede caracterizarse como $[(0, 2)]$. Otra solución es $[(0, 1), (1, 2)]$, pero en este caso hacemos un movimiento innecesario.

Si consideramos una torre con dos discos, las reglas nos obligan a, primero mover el disco pequeño al poste 1 (que funciona como poste "auxiliar"), luego mover el disco grande al poste 2, y finalmente mover el disco chico del poste 1 al poste 2. Es decir la solución es $[(0, 1), (0, 2), (1, 2)]$. Por supuesto existen otras soluciones con más movimientos.

a) Encontrá una solución a una Torre de Hanoi con tres discos. Justifica la solución, describiendo gráficamente cada paso intermedio. Tené en cuenta que la mejor solución consta de 7 pasos!

b) Encontrá una solución general a una Torre de Hanoi con n discos, definiendo un programa recursivo

$$hanoi : Nat \to Nat \to Nat \to Nat \to [(Nat, Nat)]$$

tal que $hanoi.a.b.c.n$ calcula la secuencia de *movimientos* para llevar una torre de n discos, desde el poste a hacia el poste c, utilizando (posiblemente) el poste b de forma auxiliar.

c) Demostrá que $\#hanoi.a.b.c.n = 2^n - 1$.

28. Para cada una de las siguientes funciones, escribí su tipo y describí en lenguaje natural el significado de su especificación.

a) $f.xs \doteq \langle \text{N } i, j \ : \ 0 \leq i < j < \#xs \ : \ xs.i = xs.j \rangle$

b) $g.xs \doteq \langle \text{Max } p, q \ : \ 0 \leq p < q < \#xs \ : \ xs.p + xs.q \rangle$

c) $h.xs \doteq \langle \text{N } k \ : \ 0 \leq k < \#xs \ : \ \langle \forall i \ : \ 0 \leq i < k \ : \ xs.i < xs.k \rangle \rangle$

d) $k.xs \doteq \langle \forall i, j \ : \ 0 \leq i \wedge 0 \leq j \wedge i + j = \#xs - 1 \ : \ xs.i = xs.j \rangle$

e) $l.xs \doteq \langle \text{Max } p, q \ : \ 0 \leq p \leq q < \#xs \wedge \langle \forall i \ : \ p \leq i < q \ : \ xs.i \geq 0 \rangle \ : \ q - p \rangle$

29. Derivá la función especificada en el ejercicio 28*b*.

30. Derivá un programa recursivo para la función $f : [Num] \to [Num] \to Num$, que calcula la mínima diferencia entre valores de las listas xs y ys, y cuya especificación es la siguiente:

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