Práctico 2: Especificación, derivación y verificación de programas funcionales

Algoritmos y Estructuras de Datos I

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.

Completan esta guía ejercicios algunos ejercicios para demostrar tanto por inducción sobre naturales, como inducción ***estructural*** *sobre listas. Se busca mejorar la habilidad para utilizar esta técnica de prueba que es central para la tarea de cálculo de programas recursivos.*

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. Por el contrario, los ejercicios sucesivos son más complejos y requieren el uso de técnicas avanzadas: tuplas para mejorar la eficiencia, modularización y generalización.*

1. *a*) Dada la definición recursiva de la función

$$\begin{array}{|l} fac : Nat \to Nat \\ \hline fac.0 \doteq 1 \\ fac.(n+1) \doteq (n+1) * fac.n \end{array}$$

demostrar que la siguiente propiedad

$$P.n : fac.n = \langle \sum i \ : 0 \leq i < n : \ i * fac.i \, \rangle + 1$$

vale para todo $n \in Nat$.

b) Dada la siguiente definición recursiva

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

escriba en lenguaje natural que hace la función y demuestre que la propiedad

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

vale para toda lista xs.

c) Dada la definición recursiva de la función

$$\begin{array}{|l} fib : Nat \to Nat \\ \hline fib.0 \doteq 0 \\ fib.1 \doteq 1 \\ fib.(n+2) \doteq fib.n + fib.(n+1) \end{array}$$

demostrar que la siguiente propiedad

$$P.n : fib.n < 2^n$$

vale para todo $n \in Nat$.

2. Especificar utilizando cuantificadores las funciones $abren : [Char] \to Nat$ y $cierran : [Char] \to Nat$, que devuelven la cantidad de paréntesis "(" y ")" respectivamente, que ocurren en una lista de caracteres.

   Luego, *verificar* que la siguiente definición recursiva satisface la especificación:

$$\begin{array}{rcl} abren.[\,] & \doteq & 0 \\ abren.(\text{`(`} \triangleright xs) & \doteq & abren.xs + 1 \\ abren.(\text{`)`} \triangleright xs) & \doteq & abren.xs \end{array}$$

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

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

   b) $sum\_cuad.xs = \langle \sum i \ : \ 0 \leq i < \#xs \ : \ xs.i * xs.i \rangle$

   c) $iga.e.xs = \langle \forall i \ : 0 \leq i < \#xs : \ xs.i = e \ \rangle$

   d) $exp.x.n = x^n$

   e) $sum\_par.n = \langle \sum i \ : \ 0 \leq i \leq n \wedge par.i \ : \ i \rangle$

   f) $cuántos.p.xs = \langle \mathrm{N} \ i \ : \ 0 \leq i < \#xs \ : \ p.(xs.i) \rangle$

   g) $busca.e.xs = \langle \mathrm{Min} \ \ i \ : 0 \leq i < \#xs \wedge xs.i = e : \ i \ \rangle$

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

   a) $\langle \forall x \ : \ x \in Num \ : \ \langle \exists y \ : \ y \in Num \ : \ x < y \rangle \rangle$

   b) $\langle \exists x \ : \ x \in Num \ : \ \langle \forall y \ : \ y \in Num \ : \ x < y \rangle \rangle$ ¿Cuál es la diferencia con la anterior?

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

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

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

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

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

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

   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 a punto). Si las listas tienen distinto tamaño se opera hasta la última posición de las más chica.

   e) dada un función $f : Nat \to Bool$ y suponiendo $\langle \exists n \ : 0 \leq n : \ f.n \ \rangle$ encontrar el mínimo natural x tal que $f.x$.

6. Derivar las siguientes funciones.

   a) $f : Num \to Nat$ computa la suma de potencias de un número, esto es

   $$f.x.n = \langle \sum i \ : 0 \leq i < n : \ x^i \ \rangle \ .$$

   b) $pi : Nat \to Num$ computa la aproximación del número π

   $$pi.n = 4 * \langle \sum i \ : 0 \leq i < n : \ (-1)^i/(2 * i + 1) \ \rangle$$

   Ayuda: Modularizar dos veces. La segunda con la función *exp* del ejercicio 3*d*

   c) $f : Nat \to Nat$ computa el cubo de un número natural x utilizando únicamente sumas. La especificación es muy simple: $f.x = x^3$.

   Ayuda: Usar inducción y modularización varias veces

7. Obtener definiciones con costo lineal para las siguientes funciones:

   a) Función que calcula las potencias del ejercicio 6*a*.

   b) Función *pi* del ejercicio 6*b*.

   c) Función que calcula el número de Fibonacci en el ejercicio 1*c*.

Ayuda: Para obtener costo lineal Usar la técnica de tuplas.

8. A partir de las siguientes especificaciones expresar en lenguaje natural que devuelven las funciones, agregarles su tipo y derivarlas:

   a) $psum.xs = \langle\, \forall\, i \;:\; 0 \le i \le \#xs :\; sum.(xs{\uparrow}i) \ge 0 \,\rangle$, con *sum* la función del ejercicio 3*a*.

   b) $sum\_ant = \langle\, \exists\, i \;:\; 0 \le i < \#xs :\; xs.i = sum.(xs{\uparrow}i) \,\rangle$

   c) $sum8.xs = \langle\, \exists\, i \;:\; 0 \le i \le \#xs :\; sum.(xs{\uparrow}i) = 8 \,\rangle$.

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

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

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

10. Expresar utilizando cuantificadores las siguientes sentencias del lenguaje natural:

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

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

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

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

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

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

11. Derivar funciones recursivas a partir de cada una de las especificaciones que escribiste para los ejercicios 10*a*, 10*b* y 10*e*.

12. Derivar las funciones especificadas como:

    a) $sumin.xs = \langle \text{Min}\ \ as, bs, cs \;:\; xs = as +\!\!+ bs +\!\!+ cs :\; sum.bs \,\rangle$
    (suma mínima de un segmento).

    b) $f.xs = \langle \text{N}\, as, bs, cs \;:\; xs = as +\!\!+ bs +\!\!+ cs :\; 8 = sum.bs \,\rangle$
    (cantidad de segmentos que suman eso).

    c) $maxiga.e.xs = \langle \text{Max}\ \ as, bs, cs \;:\; xs = as +\!\!+ bs +\!\!+ cs \wedge iga.e.bs :\; \#bs \,\rangle$
    (máxima longitud de igules a e)
    donde *iga* es la función del ejercicio 3*c*.

13. Obtener definiciones recursivas con costo lineal para las funciones derivadas en los ejercicios 12*a* y 12*c* utilizando la técnica de tuplas.

14. Describir en lenguaje natural y dar el *tipo* de cada una de las siguientes funciones especificadas formalmente:

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

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

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

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

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

15. Derivar la función definida en el ejercicio 14*b*.

16. Derivar una definición recursiva para la función $f : [Num] \to [Num] \to Num$, que calcula la mínima distancia entre valores de las listas xs y ys, y cuya especificación es la siguiente:

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

17. Derivar una definición recursiva para la función $g : [Char] \to [Char] \to Bool$, especificada como sigue:

$$g.xs.ys = \langle \exists as, bs, c, cs \ : \ xs = as +\!\!+ bs \wedge ys = as +\!\!+ (c \triangleright cs) \ : \ bs = [\,] \vee bs.0 < c \rangle$$

Decir en palabras cuando $g.xs.ys$ es $True$.

Ejercicios extra

18. Dada la siguiente definición recursiva

$$\begin{array}{|l} rev : [A] \to [A] \\ \hline rev.[\,] \doteq [\,] \\ rev.(x \triangleright xs) \doteq rev.xs +\!\!+ [x] \end{array}$$

escriba en lenguaje natural que hace la función y demuestre que la propiedad

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

vale para toda lista xs y ys.

19. Para las funciones $sum : [Num] \to Num$ y $duplica : [Num] \to [Num]$ definidas recursivamente como:

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

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

20. Dada la siguiente definición recursiva para la funcion $repetir : Nat \to Nat \to [Nat]$, que dada una cantidad n y un valor x construye una lista de longitud n con elementos repetidos x:

$$\begin{array}{rcl} repetir.0.x & \doteq & [\,] \\ repetir.(n+1).x & \doteq & x \triangleright repetir.n.x \end{array}$$

Demostrá que es válido $\#repetir.n.x = n$.

21. Expresar utilizando cuantificadores las siguientes sentencias del lenguaje natural:

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

   b) El elemento e ocurre en las posiciones pares de la lista xs.

   c) El elemento e ocurre únicamente en las posiciones pares de la lista xs.

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

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

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

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

22. Derivar funciones recursivas a partir de cada una de las especificaciones que escribió para el ejercicio 21.

23. Sea *fib* la definición recursiva *estándar* para la función de Fibonacci. Calculá la función de Fibolucci, $Fbl : Nat \to Nat$, especificada como:

$$Fbl.n = \langle \sum i \ : \ 0 \leq i < n \ : \ fib.i \times fib.(n-i) \rangle$$

24. Mejorar la eficiencia de la función derivada en el ejercicio 6*c* utilizando la técnica de tuplas.

25. Derivar la función recursiva a partir de la especificacion del ejercicio 10*c*.

26. *[Torres de Hanoi]*. Se tienen tres postes numerados 0, 1 y 2, y n discos de distinto tamaño. Inicialmente se encuentran todos los discos ubicados en el poste 0, ordenados según el tamaño con el disco más grande en la base. El problema consiste en llevar todos los discos al poste 2, con las siguientes restricciones:

    a) Se puede mover sólo un disco a la vez

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

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

    Resolvé los siguientes items:

    a) Sea $B = \{0, 1, 2\}$. Definí la función $hanoi : B \to B \to B \to Nat \to [(B, B)]$ tal que $hanoi.a.b.c.n$ calcula la secuencia de *movimientos* para llevar n discos desde el poste a hacia el poste c, utilizando posiblemente el poste b de forma auxiliar. Un *movimiento* es un par (B, B) cuya primer componente indica el poste de salida, y la segunda el poste de llegada.

    Ayuda: Por ejemplo, *hanoi* para los postes 0, 1 y 2, con dos discos es: $hanoi.0.1.2.2 = [(0, 1),\ (0, 2),\ (1, 2)]$

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

    c) ¿En qué movimiento se cambia de poste por primera vez el disco de mayor tamaño?

27. El siguiente razonamiento intenta demostrar que un grupo cualquiera de gatos está compuesto íntegramente por gatos negros:

    > La demostración es por inducción en el número de gatos del grupo. Para el caso de 0 gatos el resultado es inmediato puesto que cualquier gato del grupo es negro. Consideremos ahora un grupo de $n + 1$ gatos y tomemos un gato cualquiera de ellos. El grupo de los gatos restantes tiene n integrantes y por hipótesis inductiva son todos negros. Luego tenemos n gatos negros y uno apartado cuyo color no podemos predecir. Intercambiemos ahora el gato apartado con uno del grupo de n gatos, el cual sabemos con total seguridad que es negro. Así tenemos por un lado un gato negro y por otro un grupo de n gatos. Nuevamente por hipótesis inductiva, el grupo de n gatos contiene sólo gatos negros, y como el que está apartado también es negro, tenemos que los $n + 1$ gatos son negros.

    ¿Cuál es el error en este intento de demostración?

28. Especificá formalmente utilizando cuantificadores cada una de las siguientes funciones descriptas informalmente. Luego, *derivá* soluciones algorítmicas para cada una.

    a) *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.

    b) *divide* : $Nat \to Nat \to Bool$, que determina si el primer parámetro divide al segundo.

    c) *primo?* : $Nat \to Bool$, que determina si un numero es primo.

29. Derivar la función $maxigual : [A] \to Num$ especificada como

$$maxigual.xs = \langle \text{Max } as, bs, cs : xs = as +\!\!+ bs +\!\!+ cs \wedge iguales.bs : \#bs \rangle$$

    donde *iguales* es la función derivada en el ejercicio 5*c*.

    La función obtenida debe tener costo lineal.

    Ayuda: Para obtener costo lineal puede ser necesario utilizar la técnica de tuplas.

30. Especificar y derivar la función que calcula la cantidad de apariciones de un segmento en una lista.

    Ayuda: Ver cómo se especificó y derivó el ejemplo de la sección 10*b*.

31. Derivar las funciones a partir de las especificaciones en el ejercicio 14.