Algoritmos y Estructuras de Datos I - 2º cuatrimestre 2011

Práctico 2: Especificación, derivación y verificación de programas.

Docentes: Javier Blanco, Mariana Badano, Mauricio Tellechea, Demetrio Vilela, Matias Lee

Esta guía tiene como objetivo, por un lado, consolidar el manejo de expresiones cuantificadas, complementando la práctica anterior que se centraba principalmente en el cálculo (manejo sintáctico), con ejercicios en los que el énfasis reside en la comprensión del significado de las expresiones cuantificadas (manejo semántico).

Por otro lado, siendo el objetivo principal, se busca 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 partir de las siguientes especificaciones, dá el tipo de cada función y derivá las soluciones algorítmicas correspondientes.

   a) $sum\_pares.n = \langle \sum i \ : \ 0 \leq i \leq n \wedge par.i \ : \ i \rangle$

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

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

   d) $cuadrado?.n = \langle \exists x \ : \ 0 \leq x \leq n \ : \ x * x = n \rangle$

2. Expresá 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) $\langle \exists i \ : \ 0 \leq i < \#xs \, min \, \#ys \ : \ xs.i \neq ys.i \rangle$

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

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

   d) $sum\_ant : [Num] \to Bool$, que dada una lista de números, determina si algún elemento de la lista es igual a la suma de todos los elementos anteriores dentro de la misma.

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

4. Demostrá que para todo número natural n con $n \geq 4$, se satisface $n! > 2^n$.

5. Especificá 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, *verificá* que las siguientes definiciones recursivas satisfagan esas especificaciones:

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

6. Derive una función f que compute 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. Especificá formalmente utilizando cuantificadores cada una de las siguientes funciones descriptas informalmente. Luego, *derivá* soluciones algorítmicas para cada una.

   a) $divide : Nat- > Nat- > Bool$, que determina si el primer parámetro divide al segundo.

   b) $primo? : Nat- > Bool$, que determina si un numero es primo.

   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) $minout : [Nat] \to Nat$, que dada una lista de naturales, devuelve el menor número natural que **no** está en la misma.

   f) $perm : [A] \to [A] \to Bool$, que dada dos listas decide si una es una permutación de la otra.

8. 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?

9. De manera análoga al ejercicio anterior, el siguiente razonamiento intenta demostrar que si en un grupo cualquiera de personas una de ellas es comunista, entonces todas lo son:

   > La demostración es por inducción en el número de personas del grupo. El caso de 0 personas es trivialmente cierto, puesto que el antecedente de la implicación es falso. Para un grupo de 1 persona también es inmediato dado que si el único integrante del grupo es comunista, todo el grupo lo es. Tomemos ahora un grupo de $n + 2$ personas, asumiendo que al menos una de ellas es comunista. Separamos una persona cualquiera del grupo y si esta persona es comunista, entonces la intercambiamos con cualquier otra del grupo para asegurarnos que entre las restantes $n + 1$ personas exista al menos una persona comunista. Luego, por hipótesis inductiva, tenemos un grupo de $n + 1$ personas que son todas comunistas, y una persona apartada cuyas simpatías políticas no podemos deducir. Intercambiemos entonces nuevamente esta persona con cualquiera de los integrantes del grupo de $n + 1$ personas (que son todas comunistas). Como resultado obtenemos por un lado un comunista, y por otro, un grupo de $n + 1$ personas de las cuales con toda seguridad n son comunistas. Por hipótesis inductiva deducimos que los $n + 1$ integrantes de este grupo son comunistas, luego las $n + 2$ personas son comunistas.

¿Este razonamiento es correcto? en caso negativo, ¿estamos en presencia del mismo error que en el ejercicio anterior?

10. 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:

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

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

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

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

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

12. Expresá utilizando cuantificadores las siguientes sentencias del lenguaje natural:

    a) El elemento x ocurre un número par de veces en la lista xs.
    b) El elemento x ocurre en las posiciones pares de la lista xs.
    c) El elemento x ocurre únicamente en las posiciones pares de la lista xs.
    d) Si x ocurre en la lista xs, entonces y 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.

13. Derivá funciones recursivas a partir de cada una de las especificaciones que escribiste para los ítems a, c, g del ejercicio anterior.

14. Siendo $\mathbb{P}(A)$ el conjunto de todos los subconjuntos de A, y $\#(A)$ el cardinal del conjunto A, demuestre que para todo conjunto finito A se satisface $\#(\mathbb{P}(A)) = 2^{\#(A)}$.

15. Expresá utilizando cuantificadores las siguientes sentencias del lenguaje natural:

    a) El número natural n es un *cubo perfecto*.
    b) La lista xs es un segmento inicial de la lista ys.
    c) La lista xs es un segmento final de la lista ys.
    d) La lista xs es un segmento de la lista ys.
    e) Las listas xs y ys tienen en común un segmento.
    f) La lista xs posee un segmento **no** inicial y **no** final cuyos valores son mayores a los valores del resto de la misma.
    g) La lista xs de numeros enteros tienen la misma cantidad de elementos pares e impares.

16. Derivá funciones recursivas a partir de cada una de las especificaciones que escribiste para los ítems a, b y g del ejercicio anterior.

17. La función de *Fibonacci*, $fib : Num \to Num$, puede definirse recursivamente como:

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

Demostrar que se satisface $fib.n = \frac{A^n - B^n}{\sqrt{5}}$, donde $A \doteq \frac{1+\sqrt{5}}{2}$ y $B \doteq \frac{1-\sqrt{5}}{2}$.

Ayuda: Demostrar $1 + A = A^2$ y $1 + B = B^2$

18. Describí en lenguaje natural y especificá el *tipo* de cada una de las siguientes funciones especificadas formalmente:

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

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

19. *[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?

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

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

21. Mejorá la eficiencia de la función derivada en el ejercicio 6 utilizando la técnica de tuplas.

22. Sea *hanoi* la función que definiste en el ejercicio 19. Derivá 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? Pensá en la eficiencia de *hanoi*.

    Ayuda: Puede ser útil calcular manualmente $t.a.b.c,0$, $t.a.b.c,1$ y $t.a.b.c.(n+2)$

23. Derivá 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:

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

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

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

25. Tomá la definición de la función *perm* del ejercicio 7, y *rev* definida en el teórico, y demostrá $perm.xs.(rev.xs)$