Fundamentos de la Computación
Ejercicios de Listas

Cátedra de Teoría de la Computación

2013

Considere la siguiente definición del tipo (polimórfico) de las listas de elementos de tipo a:

`data` $[a]$ `where` $\{[\,] :: [a], (:) :: a \to [a] \to [a]\}$

?1. Defina las siguientes funciones utilizando recursión estructural sobre listas:

1. *length* $:: [a] \to \mathbb{N}$, que calcula la longitud (cantidad de elementos) de una lista.
2. $(++) :: [a] \to [a] \to [a]$, que concatena dos listas.
3. *reverse* $:: [a] \to [a]$, que invierte una lista.
4. *concat* $:: [[a]] \to [a]$, que concatena una lista de listas.
   Por ejemplo: *concat* $[[1,2],[3],[4,5,6],[\,],[7,8]] = [1,2,3,4,5,6,7,8]$
5. *filter* $:: (a \to \textbf{\textit{Bool}}) \to [a] \to [a]$, que recibe un predicado y una lista y devuelve la lista con los elementos para los cuales el predicado es verdadero.
   Por ejemplo: *filter* $?par\,[2,4,6,7,8,9,0] = [2,4,6,8,0]$
6. *takeWhile* $:: (a \to \textbf{\textit{Bool}}) \to [a] \to [a]$, que recibe un predicado y una lista y devuelve la lista que se obtiene tomando los elementos de la lista argumento mientras el predicado se cumpla.
   Por ejemplo: *takeWhile* $?par\,[2,4,6,7,8,9,0] = [2,4,6]$
7. *dropWhile* $:: (a \to \textbf{\textit{Bool}}) \to [a] \to [a]$, que recibe un predicado y una lista y devuelve la lista que se obtiene descartando los elementos de la

lista argumento mientras el predicado se cumpla.
Por ejemplo: $dropWhile\ ?par\ [2,4,6,7,8,9,0] = [7,8,9,0]$

8. $sum :: [\mathbb{N}] \to \mathbb{N}$, que suma todos los elementos de una llista de naturales.

9. $prod :: [\mathbb{N}] \to \mathbb{N}$, que multiplica todos los elementos de una llista de naturales.

?2. Defina una función *foldr* que generalice el esquema de recursión primitiva. ¿Cuál es el tipo de *foldr*?

1. Utilizando *foldr* redefina las funciones *concat*, *sum* y *prod*

2. Defina la función $and :: [\boldsymbol{Bool}] \to \boldsymbol{Bool}$, que calcula la conjunción de una lista de booleanos utilizando *foldr*.

3. Defina la función $or :: [\boldsymbol{Bool}] \to \boldsymbol{Bool}$, que calcula la disyunción de una lista de booleanos utilizando *foldr*.

?3.

1. Defina la función $map :: a \to b \to [a] \to [b]$ que aplica una función a todos los elementos de una lista.

2. Defina la función $all :: (a \to \boldsymbol{Bool}) \to [a] \to \boldsymbol{Bool}$, que recibe un predicado y una lista y verifica si todos los elementos de la lista cumplen el predicado. No utilice recursión, sino la función *map* combinada con otra de las anteruirmente definidas.

3. Haga lo mismo para definir la función $exists :: (a \to \boldsymbol{Bool}) \to [a] \to \boldsymbol{Bool}$, que recibe un predicado y una lista y verifica si existe algún elemento de la lista para el cual se cumpla el predicado.

4. Defina la función $zipWith :: (a \to b \to c) \to [a] \to [b] \to [c]$, que construye una lista cuyos elementos se calculan a partir de aplicar la función dada a los elementos de las listas de entrada que ocurren en la misma posición en ambas listas. Si una lista es más larga que otra, no se consideran los elementos sobrantes.
Por ejemplo, $zipWith\ (+)\ [1,3,5]\ [2,4,6,8] = [3,7,11]$

?4. Defina las siguientes funciones *parciales*. Especifique la precondición en cada caso:

1. $head :: [a] \to a$ que devuelve el primer elemento de una lista no vacía.

2. *tail* :: $[a] \to [a]$ que devuelve el resultado de quitar el el primer elemento de una lista no vacía.

3. *last* :: $[a] \to a$ que devuelve el último elemento de una lista no vacía.

4. *init* :: $[a] \to [a]$ que devuelve el resultado de quitar el último elemento de una lista no vacía.

5. (!!) :: $[a] \to \mathbb{N} \to a$ que devuelve el n-ésimo elemento de una lista, empezando desde 0.

?5. Defina las siguientes funciones:

1. *take* :: $\mathbb{N} \to [a] \to [a]$, tal que *take* $n\, xs$ devuelve una lista fomada con los primeros n elementos de xs. Si xs tiene menos de n elementos, devuelve todos lod e xs.

2. *drop* :: $\mathbb{N} \to [a] \to [a]$, tal que *drop* $n\, xs$ devuelve la lista resultante de tirar los primeros n elementos de xs. Si xs tiene menos de n elementos, devuelve la lista vacía.

3. *splitAt* :: $\mathbb{N} \to [a] \to ([a], [a])$, tal que *splitAt* $n\, xs$ devuelve un par, conteniendo una lista con los primeros n elementos de xs y otra lista con el resto de los elementos de xs. Dar dos definiciones de esta función, una utilizando las funciones anteriores, y otra recursiva. ¿Cuál es más eficiente de las dos?

?6.

1. Defina las instancia de *Eq* para $[a]$. ¿Qué condición debe cumplir el tipo a?

2. Defina las instancia de *Ord* para $[a]$, de modo tal que el orden de las listas sea el lexicográfico. ¿Qué condición debe cumplir el tipo a?

?7. Defina las siguientes funciones sobre listas de elementos con igualdad:

1. $Eq\, a \Rightarrow$ *member* :: $a \to [a] \to$ ***Bool***, que verifica si un elemento pertenece a una lista.

2. $Eq\, a \Rightarrow$ *deleteOne* :: $a \to [a] \to [a]$, que borra la primera ocurrencia de un elemento de una lista.

3. $Eq\, a \Rightarrow$ *deleteAll* :: $a \to [a] \to [a]$, que borra la todas las ocurrencias de un elemento de una lista.

?8. Defina las siguientes funciones sobre listas de elementos con orden:

1. $Ord\, a \Rightarrow max :: [a] \to a$, que devuelve el máximo elemento de una lista.
2. $Ord\, a \Rightarrow sorted :: [a] \to \textbf{\textit{Bool}}$, que verifica si una lista está ordenada.
3. $Ord\, a \Rightarrow insert :: a \to [a] \to [a]$, que inserta un elemento en una lista, de modo tal que si la lista está ordenada, el resultado tambin lo está.
4. $Ord\, a \Rightarrow sort :: [a] \to [a]$, que ordena una lista, insertando recursivamente cada uno de sus elementos en forma ordenada.

?9. Defina las siguientes funciones:

1. $divisores :: \mathbb{N} \to [\mathbb{N}]$, que devuelve la lista de los divisores de un número natural dado.
2. $primo :: \mathbb{N} \to \textbf{\textit{Bool}}$, que verifica si un número dado es primo (utilice la función anuetior).
3. $factoresPrimos :: \mathbb{N} \to [\mathbb{N}]$, que devuelve la lista de factores primos de de un número natural dado (sin repeticiones).
   Por ejemplo, $factoresPrimos\ 24 = [2, 3]$.
4. $descomponer :: \mathbb{N} \to [\mathbb{N}]$, que devuelve la lista de factores primos de de un número natural dado con todas las repeticiones.
   Por ejemplo, $descomponer\ 24 = [2, 2, 2, 3]$.
5. $descomponer1 :: \mathbb{N} \to [(\mathbb{N}, \mathbb{N})]$, que devuelve una lista de pares conteniendo los factores primos de de un número natural dado con su exponente.
   Por ejemplo, $descomponer1\ 24 = [(2, 3), (3, 1)]$.

?10. Pruebe que las siguientes propiedades de la concatenación de listas:

1. $(\forall xs, ys \Downarrow :: [a])\ length\ (xs +\!\!+ ys) = length\ xs + length\ ys$
2. $(\forall xs \Downarrow :: [a])\ xs +\!\!+ [\,] = xs$
3. $(\forall xs, ys, zs \Downarrow :: [a])\ (xs +\!\!+ ys) +\!\!+ zs = xs +\!\!+ (ys +\!\!+ zs)$

?11. Pruebe que las siguientes propiedades de la función *reverse*:

1. $(\forall x \Downarrow :: a)\ reverse\ [x] = [x]$
2. $(\forall xs, ys \Downarrow :: [a])\ reverse\ (xs +\!\!+ ys) = reverse\ ys +\!\!+ reverse\ xs$
3. $(\forall xs \Downarrow :: [a])\ reverse\ (reverse\ xs) = xs$