20 Ayudas a Ejercicios Seleccionados

20.1. Introducción a Haskell

Ayuda al Ejercicio 2.14 (pág. 33).– Observar que se puede utilizar la cabeza de la lista como acumulador y las ecuaciones a completar son:

$$\begin{array}{ll} aEntero\ [d] & = \ldots \\ aEntero\ (d : m : r) & = \ldots \end{array}$$

Ayuda al Ejercicio 2.32 (pág. 47).– Completar la definición:

$$\begin{array}{lll} resto\ x\ y & \mid x\ <\ y & = \ldots \\ & \mid otherwise & = \ldots \end{array}$$

Ayuda al Ejercicio 2.33 (pág. 47).– Completar la definición:

$$\begin{array}{lll} cociente\ x\ y & \mid x\ <\ y & = \ldots \\ & \mid otherwise & = \ldots \end{array}$$

Ayuda al Ejercicio 2.44 (pág. 48).– Usar la solución del Ejercicio 2.32 y del Ejercicio 2.33.

Ayuda al Ejercicio 2.45 (pág. 48).– Se pueden usar las funciones *trocear* y *aLaDerechaDe*.

20.2. Definiciones de tipo

Ayuda al Ejercicio 4.4 (pág. 78).–

Para la primera completar la siguiente definición:

$$\begin{array}{ll} \acute{a}rea & :: \ Figura \ \rightarrow \ Float \\ \acute{a}rea \ (C\acute{\imath}rculo \ r) & = \ldots \\ \acute{a}rea \ (Cuadrado \ l) & = \ldots \\ \acute{a}rea \ (Rect\acute{a}ngulo \ b \ h) & = \ldots \\ \acute{a}rea \ Punto & = \ldots \end{array}$$

La segunda se define de modo similar.

Solución al Ejercicio 4.5 (pág. 78).– Completar la siguiente definición:

$$\begin{array}{ll} ra\acute{\imath}ces & :: \ Float \ \rightarrow \ Float \ \rightarrow \ Float \ \rightarrow \ Resultado \\ ra\acute{\imath}ces \ a \ b \ c & \\ \quad \mid \ a == 0 & = error \ \text{"Ecuación de primer grado"} \\ \quad \mid \ disc == 0 & = UnaReal \ x \\ \quad \mid \ disc \ > \ 0 & = \ldots \\ \quad \mid \ disc \ < \ 0 & = \ldots \\ \quad \textbf{where} & \\ \qquad \ldots & \end{array}$$

Ayuda al Ejercicio 4.25 (pág. 103).– Utiliza la simetría de max:

$$max \ x \ y \ \equiv \ max \ y \ x$$

Ayuda al Ejercicio 4.26 (pág. 103).– Habrá que definir las cuatro combinaciones posibles para cada operación. Puede ser útil definir una función de conversión

$$conver \ :: \ Complejo \ \rightarrow \ Complejo$$

entre las representaciones polares y cartesianas.

Ayuda al Ejercicio 4.28 (pág. 103).– Completar las ecuaciones:

$$\begin{array}{ll} total \ (D\acute{\imath}gito \ n) & = \ \ldots \\ total \ (i \ :\uparrow: \ d) & = \ \ldots \end{array}$$

20.3. EL SISTEMA DE CLASES DE HASKELL

Ayuda al Ejercicio 5.5 (pág. 129).– Completar la definición:

$$\begin{array}{ll} \multicolumn{2}{l}{\textbf{instance} \ Ord \ Nat \ \textbf{where}} \\ \quad Cero \ \leq _ & = \ldots \\ \quad Suc \ _ \leq Cero & = \ldots \\ \quad Suc \ n \leq Suc \ m & = \ldots \end{array}$$

© ITES-Paraninfo

Ayuda al Ejercicio 5.6 (pág. 129).– Considerar que la igualdad es la estructural y el orden el de enumeración.

Ayuda al Ejercicio 5.7 (pág. 129).– Puede ser útil definir un tipo $Marcador$

$$\textbf{data}\ Marcador\ =\ M\ (Int, Int, Int, Int, Int, Int, Int)$$

donde cada entero describe cuántas veces aparece un determinado color; definir instancias de éste para las clases Num y Eq, de modo que dos colores son iguales si lo son sus marcadores asociados. Por último, definir una función

$$creaMarcador\ ::\ Color\ \rightarrow\ Marcador$$

que a cada color le asocie su marcador.

Ayuda al Ejercicio 5.8 (pág. 130).– Una mochila puede representarse por una lista de pares donde cada par consta del elemento en cuestión y el número de veces que dicho elemento aparece.

$$\textbf{data}\ Mochila\ a\ =\ M\ [(a,\ Int)]$$

Ayuda al Ejercicio 5.9 (pág. 130).– Considerar por separado la igualdad de colores simples y compuestos. En el caso de los colores compuestos los colores serán iguales cuando sus mochilas de colores simples asociadas lo sean.

20.4. PROGRAMACIÓN CON LISTAS

Ayuda al Ejercicio 6.28 (pág. 156).– Para diferenciar los casos $(>)$ y $(\geq)$ considerar el caso de elementos repetidos.

Ayuda al Ejercicio 6.29 (pág. 156).– Considerar el predicado

$$x\ <<\ ys \quad \equiv \quad \langle x \text{ es menor o igual que todos los de la lista } ys \rangle$$

y demostrar previamente por inducción estructural

$$
\begin{array}{llcl}
(1) & ordenadaAsc\ (x : xs) & = & ordenadaAsc\ xs \wedge\ x\ <<\ xs \\
(2) & x\ <<\ ys \wedge\ x\ \leq\ y & \Rightarrow & x\ <<\ (insertar\ y\ ys)
\end{array}
$$

Ayuda al Ejercicio 6.33 (pág. 164).– Dividir el argumento sucesivamente por 2, representando tanto el resto como la división entera; observar que se puede evitar el uso de la concatenación $(+\!\!+)$ introduciendo una función auxiliar con un parámetro adicional que vaya acumulando el resultado.

Ayuda al Ejercicio 6.34 (pág. 164).–

© ITES-Paraninfo

1. Se puede probar

$$(*) \qquad inv2\ xs\ ys \quad = \quad inv2\ xs\ [\,] +\!\!+ ys$$

por inducción estructural sobre xs. En el paso de inducción habrá que utilizar la asociatividad de $(+\!\!+)$ y que la igualdad es sustitutiva (is).

2. Se puede probar

$$inv\ (u +\!\!+ (x : v)) \quad = \quad inv\ v +\!\!+ (x : inv\ u)$$

por inducción estructural sobre u. En el paso de inducción habrá que utilizar la asociatividad de $(+\!\!+)$.

Ayuda al Ejercicio 6.36 (pág. 165).– Completar la definición:

$$\begin{array}{lll} 1 \quad `de`\ (x : _) & = & \ldots \\ (n+1)\ `de`\ (_ : xs) & = & \ldots \end{array}$$

e intentar probar

$$\forall\, m \centerdot m \geq 1 \centerdot \quad \forall\, n \centerdot 1 \leq n \leq m \centerdot \\ n\ `de`\ [x_1,\ \ldots,\ x_m] \quad = \quad x_n$$

Ayuda al Ejercicio 6.37 (pág. 165).– Se trata de probar primero directamente

$$\forall\, xs :: [Int] \centerdot \quad \forall\, x :: Int \centerdot \\ par\ x\ \wedge\ todosPares\ xs \quad = \quad todosPares\ (x : xs)$$

y de aquí, por inducción sobre n, $\forall\, n \centerdot n \geq 0 \centerdot$

$$todos_pares\ [x_n,\ x_{n-1},\ \ldots,\ x_0] = \forall\, k \centerdot 0 \leq k \leq n \centerdot par\ x_k$$

Ayuda al Ejercicio 6.39 (pág. 165).– Puede ser útil utilizar la función:

$$\begin{array}{l} \acute{u}ltimoYResto\ [x] \qquad = (x,\ [\,]) \\ \acute{u}ltimoYResto\ (x : xs) = (y,\ x : r) \\ \quad \textbf{where}\ (y,\ r) = \acute{u}ltimoYResto\ xs \end{array}$$

Ayuda al Ejercicio 6.40 (pág. 165).– Completar la función

$$\begin{array}{lll} [\,] \quad `esMenorQue`\ _ & = & \ldots \\ _ \quad `esMenorQue`\ [\,] & = & \ldots \\ (x : xs)\ `esMenorQue`\ (y : ys) & = & \ldots \end{array}$$

© ITES-Paraninfo

Ayuda al Ejercicio 6.42 (pág. 165).– Completar la definición:

$$\begin{array}{lll} \mathit{carácter}\ _ & [\,] & = \ldots \\ \mathit{carácter}\ 0 & (x : _) & = \ldots \\ \mathit{carácter}\ (n+1) & (x : xs) & = \ldots \end{array}$$

Ayuda al Ejercicio 6.44 (pág. 166).– Utilizar la función *carácter* del Ejercicio 6.42 y *pos* del Ejercicio 6.43.

La definición del tipo *Empleado* puede ser:

$$\mathbf{data}\ \mathit{Empleado}\ =\ \mathit{Emp}\ [\mathit{Char}]\ \mathit{Int}\ \mathbf{deriving}\ \mathit{Show}$$

Ayuda al Ejercicio 6.47 (pág. 166).– Se pueden utilizar las funciones *min* y *minimum* si creamos instancias para el tipo *Emp* de las clases *Eq* y *Ord*.

Ayuda al Ejercicio 6.48 (pág. 166).– Mientras el caso base es muy fácil, para el paso inductivo téngase en cuenta la siguiente identidad:

$$h(\mathbf{if}\ \ b\ \ \mathbf{then}\ \ u\ \ \mathbf{else}\ \ v)\ =\ \mathbf{if}\ \ b\ \ \mathbf{then}\ \ h\ u\ \ \mathbf{else}\ \ h\ v$$

Ayuda al Ejercicio 6.51 (pág. 166).– Si se intenta a través de inducción estructural, para probar el paso inductivo de (a) es útil la siguiente identidad

$$\mathit{map}\ f\ (u +\!\!+\ v)\ =\ \mathit{map}\ f\ u +\!\!+\ \mathit{map}\ f\ v$$

que se prueba también por inducción estructural; para (b) considérese solamente listas no vacías.

Ayuda al Ejercicio 6.52 (pág. 166).– Planteamos una única observación que es válida para todos los apartados; siendo g la constante que representa al conjunto de datos, estúdiese, por ejemplo, la lista:

$$[\ a *:\ b == b *:\ a \mid a \leftarrow\ g,\ b \leftarrow\ g\ ]$$

Ayuda al Ejercicio 6.54 (pág. 167).– Utilizar (y tratar de probar, por inducción sobre xs),

$$\mathit{length}\ (\ \mathit{map}\ f\ xs\ )\ =\ \mathit{length}\ xs$$

Ayuda al Ejercicio 6.55 (pág. 167).– Basta demostrar por inducción sobre n:

$$\begin{array}{l} \forall n \centerdot n \geq\ 0 \centerdot \\ \quad \mathit{foldr}\ f\ z\ [x_1,\ \ldots,\ x_n]\ =\ \mathit{foldl}\ (\mathit{flipf})\ z\ [x_n,\ \ldots,\ x_1] \end{array}$$

© ITES-Paraninfo

y para ello es interesante utilizar la propiedad

$$\forall\, n \,\centerdot\, n \geq 1 \,\centerdot\, \\ \quad \mathit{foldr}\ f\ (f x_n\ z)\ [x_1,\ \ldots,\ x_{n-1}] \ =\ \mathit{foldr}\ f\ z\ [x_1,\ \ldots,\ x_n]$$

que se demuestra, así mismo, por inducción sobre n.

Ayuda al Ejercicio 6.56 (pág. 167).– Al intentar probar

$$\forall\, xs \,\centerdot\, xs :: [a] \,\centerdot\, \\ \quad (\forall\, f, z \,\centerdot\, \mathit{foldl}\ f\ z\ xs \ =\ \mathit{foldr}\ f\ z\ xs)$$

por inducción sobre xs, nos encontramos con que sería interesante utilizar la propiedad

$$f\ x\ (\mathit{foldr}\ f\ z\ yz) \ =\ \mathit{foldl}\ f\ (f\ z\ x)\ ys$$

que se prueba así mismo por inducción sobre ys.

Ayuda al Ejercicio 6.58 (pág. 167).– Compruébese que tenemos

$$\begin{array}{l} \quad \mathit{copia}\ [\,] \\ \equiv \\ \quad \lambda\, y\ \rightarrow\ [\,] \end{array}$$

$$\begin{array}{l} \quad \mathit{copia}\ (x : xs) \\ \equiv \\ \quad \lambda y\ \rightarrow\ y\ :\ \mathit{copia}\ xs \end{array}$$

y dedúzcase una forma más simple.

Ayuda al Ejercicio 6.59 (pág. 167).– Piénsese la reducción

$$\begin{array}{l} \quad \mathit{alFinal}\ 7\ [4,5] \\ \Longrightarrow \\ \quad 4 : \mathit{alFinal}\ 7\ [5] \\ \Longrightarrow \\ \quad 4 : 5 : \mathit{alFinal}\ 7\ [\,] \\ \Longrightarrow \\ \quad 4 : 5 : [7] \end{array}$$

Ayuda al Ejercicio 6.60 (pág. 168).– Observar que el caso $\mathit{listaASec}\ [\,]$ no puede definirse.

© ITES-Paraninfo

20.5. EVALUACIÓN PEREZOSA. REDES DE PROCESOS

Ayuda al Ejercicio 8.4 (pág. 191).– Se pueden probar los dos primeros predicados por inducción sobre n. En el paso inductivo del segundo habrá que probar:

$$\forall\, n \,\centerdot\, n > 0 \,\centerdot\, \forall\, u \,\centerdot\, u :: [a] \,\centerdot\, \\ \quad aprox\ n\ (map\ f\ u)\ =\ map\ f\ (aprox\ n\ u)$$

para lo cual se puede razonar según sea la lista $u \equiv [\,]$ o $u \equiv x : u$. Finalmente se puede aplicar el lema de la Sección 8.2.2 y la transitividad de $=$.

Ayuda al Ejercicio 8.6 (pág. 191).– Piense en una función que devuelve los subconjuntos en el siguiente orden

$$\begin{array}{l} potencias\ [1..] \\ \Longrightarrow \\ [\,[\,], [1], [2], [1,2], [3], [1,3], [2,3], [1,2,3], \ldots\,] \end{array}$$

Ayuda al Ejercicio 8.9 (pág. 197).– Si generamos una lista infinita de pares en los que la primera componente sea el factorial y la segunda la lista (infinita) de los factoriales que quedan por generar, podemos después aplicar *first* a todos los elementos de la lista de pares (siendo *first* la función que extrae la primera componente de un par) para obtener la lista (infinita) de los factoriales.

Ayuda al Ejercicio 8.12 (pág. 204).– Partir de la función

$$iterate\ f\ x\ =\ x\ :\ iterate\ f\ (f\ x)$$

Ayuda al Ejercicio 8.13 (pág. 205).– Completar las definiciones:

$$\begin{array}{l} autom\ =\ [(1,1,0),(2,1,1)] +\!\!+ (sigm\ [2]\ 2\ 1\ 1) \\ sigm\ (x : xs)\ u\ c1\ c2\ =\ \ldots \end{array}$$

donde $c1$ y $c2$ son dos contadores de unos y doses.

Ayuda al Ejercicio 8.14 (pág. 206).– Utilizar una función auxiliar $contad$ con dos acumuladores (uno para llevar el valor actual que se está contando y otro para el número actual de repeticiones del valor actual).

Ayuda al Ejercicio 8.10 (pág. 199).– Puede utilizarse la función *fibs* que genera la lista infinita de números de Fibonacci y la función:

$$pares\ =\ [\,(x,y) \mid x\ \leftarrow\ [2..],\ y\ \leftarrow\ [1..(x-1)]\,]$$

© ITES-Paraninfo

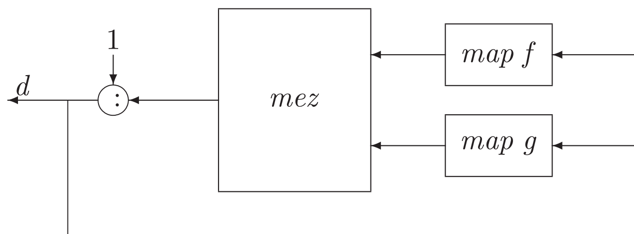

Figura 20.1: Primera versión para el conjunto de Dijkstra..

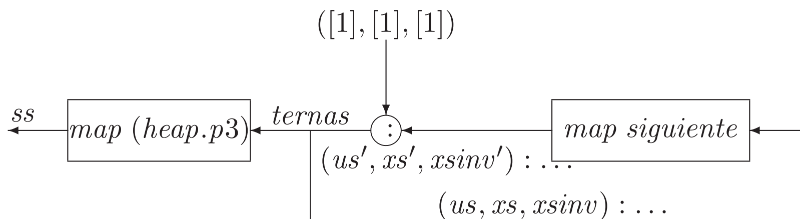

Figura 20.2: Segunda versión para el conjunto de Dijkstra..

Ayuda al Ejercicio 8.15 (pág. 207).– Considerar la red de la figura 20.1. Obsérvese que el conjunto $\mathcal{D}'$ de los elementos de la lista ordenada $dij\ f\ g$ verifica los axiomas $Ax1$ y $Ax2$, por lo que $\mathcal{D} \subseteq \mathcal{D}'$; para demostrar que $\mathcal{D}' = \mathcal{D}$ hay que probar la otra inclusión (i.e., $\mathcal{D}' \subseteq \mathcal{D}$), lo cual puede hacerse por inducción.

Ayuda al Ejercicio 8.16 (pág. 207).– Considerar la red de la figura 20.2

Ayuda al Ejercicio 8.17 (pág. 207).– Considerar la red de la figura 20.3

Ayuda al Ejercicio 8.21 (pág. 209).– Para definir $nombra$ se puede utilizar una función auxiliar $nombra'$ con dos acumuladores (uno para los enteros y otro para las listas) que lleve por separado la cabeza y el resto del argumento que se le pasa a $nombra$.

Para los apartados 2 y 3 utiliza las redes de las figuras 20.4 y 20.5 respectivamente.

Para el apartado 4 utilizar el mismo esquema del apartado 3 definiendo adecuadamente $test$, p y q.

Ayuda al Ejercicio 8.25 (pág. 210).– Obsérvese que la lista a calcular es:

$$[\,[1^2, 2^2, 3^2, \ldots],\ [1^3, 2^3, 3^3, \ldots], [1^4, 2^4, 3^4, \ldots],\ \ldots\,]$$

20.6. Programación con árboles y grafos

© ITES-Paraninfo

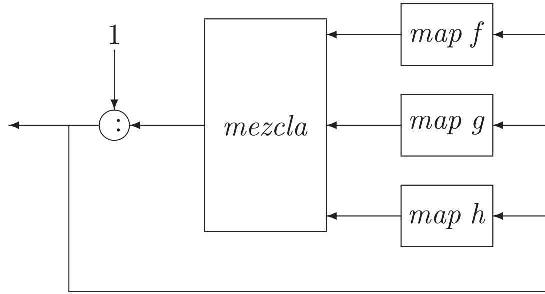

Figura 20.3: Tercera versión para el conjunto de Dijkstra..

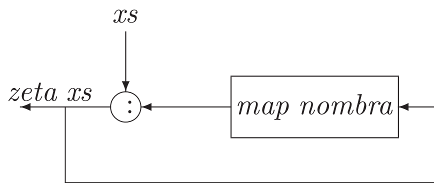

Figura 20.4: Red para computar $zeta$..

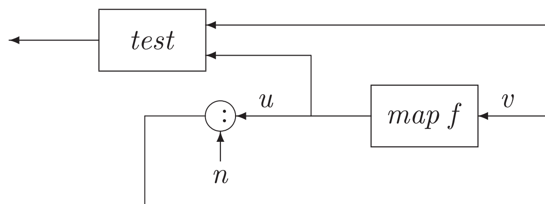

Figura 20.5: Red para computar $ap3$..

© ITES-Paraninfo

Ayuda al Ejercicio 9.16 (pág. 237).– Considérese la lista de reducciones de los *hijos* de cierto nodo $N\ c\ ds$

$$[\ reduce\ f\ d\ z \mid d \leftarrow\ ds\ ]$$

Para el apartado segundo considérense expresiones de la forma

$$N\ (f\ c)\ [\ aplica\ f\ a \mid a\ \leftarrow\ as\ ] \qquad\qquad [\ visita\ a \mid a \leftarrow\ as\ ]$$

20.7. ALGORITMOS NUMÉRICOS PROGRAMADOS FUNCIONALMENTE

Ayuda al Ejercicio 12.1 (pág. 305).– Demuéstrese

$$[a,\ f\ a,\ f^2\ a,\ \ldots,\ f^n\ a] \equiv\ a : map\ f\ [a,\ f\ a,\ \ldots,\ f^{n-1}\ a]$$

Otra posibilidad es comparar las expresiones

$$itera\ (n+1)\ f\ a \qquad\qquad itera\ n\ f\ (f\ a)$$

Ayuda al Ejercicio 12.2 (pág. 311).– Modifica las función tal y la llamada.

Ayuda al Ejercicio 12.3 (pág. 312).– Basta con cambiar la cabeza del resultado devuelto por la función tal.

Ayuda al Ejercicio 12.4 (pág. 312).– Como $\mathbf{Ax} = \mathbf{LRx} = \mathbf{b}$, bastará con saber resolver sistemas con matriz de coeficientes triangular. Usa la perezosidad de los arrays HASKELL para evitar determinar el orden de obtención de las incógnitas a la hora de resolver dichos sistemas.

Ayuda al Ejercicio 12.5 (pág. 312).– Usa la función $conjugate$ de la librería $Complex$ para modificar ligeramente la función $tras$.

Ayuda al Ejercicio 12.6 (pág. 316).– Puede ser útil definir las funciones:

$$\begin{array}{ll} normaliza & :: Double\ \to\ NumReal \\ denormaliza & :: NumReal\ \to\ Double \\ construye & :: Double\ \to\ Integer\ \to\ NumReal \end{array}$$

donde

$$normaliza\ x\ =\ construye\ x\ 4$$

Una solución se obtiene utilizando directamente $normaliza$ y $denormaliza$, aunque una solución más eficiente consiste en definir las operaciones usando $construye$.

© ITES-Paraninfo

Ayuda al Ejercicio 12.7 (pág. 316).– Utiliza $foldr$.

Ayuda al Ejercicio 12.8 (pág. 317).– Tanto $componer$ como la función generadora para enumerar árboles ordenados son traducción directa de las condiciones del problema.

Ayuda al Ejercicio 12.9 (pág. 317).– La función $integralDe$ se puede dar en base a una función auxiliar $integrar$:

$$integralDe\ as\ =\ 0\ :\ (integrar\ as\ 1)\ \mathbf{where}\ \ldots$$

Además, al integrar la ecuación diferencial que verifica $\exp(x)$ con la condición inicial correspondiente se tiene $y = 1 + \int_0^x y(t)dt$.

Ayuda al Ejercicio 12.10 (pág. 317).– La función $derivadaDe$ se puede dar en base a una función auxiliar $derivar$:

$$derivadaDe\ (_\ :\ as)\ =\ derivar\ as\ 1\ \mathbf{where}\ \ldots$$

Ayuda al Ejercicio 12.11 (pág. 317).– Se recomienda estudiar la demostración del Ejercicio 12.8 y la comprobación del Ejercicio 12.10.

20.8. PUZZLES Y SOLITARIOS

Ayuda al Ejercicio 13.1 (pág. 322).– Si incluimos el tamaño de las vasijas en la configuración

$$vasijas\ mx\ my\ n\ =\ caminoDesde\ (V\ 0\ 0\ mx\ my)\ test\ [\,]\ \mathbf{where}\ \ldots$$

se podría definir un predicado

$$aislables\ ::\ Int\ \rightarrow\ Int\ \rightarrow\ Bool$$
$$aislables\ mx\ my\ =\ \ldots$$
– – es posible aislar cualquier múltiplo de mcd(mx,my)

para después calcular el test

$$and\ [\ aislables\ mx\ my\ |\ (mx, my)\ \leftarrow\ pares\ ]$$

donde la función $pares$ devuelve la lista de posibles pares

$$(2,1),(3,1),(4,1),(3,2),\ldots,(n,1),(n-1,2),\ldots$$

Ayuda al Ejercicio 13.2 (pág. 322).– Siguiendo las notaciones de la Sección 13.1.2, ahora existen más movimientos

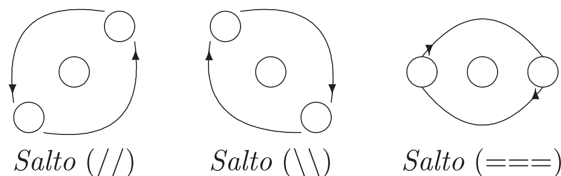

Figura 20.6: Tipos de saltos en tableros triangulares de Abreu..

$$
\begin{array}{l}
ops \ = \ [voXY, voYZ, \ldots] \\
voXY \ (V \ z \ y \ x \ mz \ my \ mx) \ = \quad \text{-- volcar X sobre Y} \\
\quad \textbf{if} \ s \leq my \ \textbf{then} \ V \ z \ s \quad 0 \qquad\qquad mz \ my \ mx \quad \text{-- cabe todo} \\
\qquad\qquad\quad \textbf{else} \ \ V \ z \ my \ (s - my) \ mz \ my \ mx \quad \text{-- sobra algo} \\
\qquad \textbf{where} \ s \ = \ x + y \\
\ldots
\end{array}
$$

y bastará completar el resto de funciones.

Ayuda al Ejercicio 13.3 (pág. 324).– Sólo hay que evitar que dos caníbales viajen en la barca con un misionero, por lo que sólo hay que modificar la función *pares*

$$
pares \ = \ [\,(x, y) \mid x \ \leftarrow \ [0..3], \ y \ \leftarrow \ [0..3], \ \ldots\,]
$$

Ayuda al Ejercicio 13.4 (pág. 326).– Podemos incluir la estructura del tablero en la configuración

$$
\begin{array}{l}
abreu \ :: \ ([(Casilla, Casilla, Casilla)], Int) \ \rightarrow \ Casilla \ \rightarrow \ \ldots \\
abreu \ (tab, n) \ h \ = \ caminoDesde \ (C \ (hueco \ h) \ tab) \ test \ [\,] \\
\quad \textbf{where} \ \ldots \\
resoluble \ (tab, n) \ = \ noVacía \ . \ (abreu \ (tab, n))
\end{array}
$$

donde el predicado $resoluble\ (tab, n)\ h$ comprueba si es posible resolver el solitario para un tablero triangular tab de n filas con el hueco inicial en la posición h, y ahora aplicar un razonamiento parecido al del Ejercicio 13.1. Para generar el tablero téngase en cuenta la Figura 20.6 y la Figura 20.7.

Ayuda al Ejercicio 13.5 (pág. 326).– Podemos considerar la siguiente codificación de las casillas

$$
\begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{array}
$$

y describimos una configuración como una lista *fs* de nueve enteros

© ITES-Paraninfo

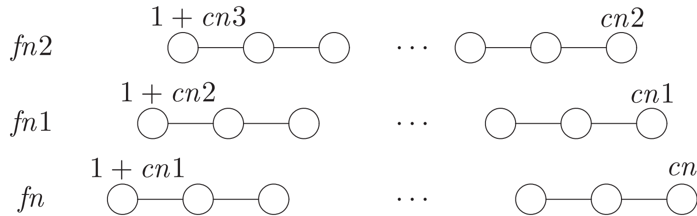

Figura 20.7: Generación recursiva de tableros triangulares de Abreu..

$$\textbf{data}\ Config\ =\ C\ [Int]$$

donde el hueco lo representaremos con un 0 (cero); los movimientos pueden generarse a través de una función

```
movs c = case c of
           { 1 → [2,4] ; 2 → [1,3,5] ; ... }
instance Grafo Config where
  suc (C c) = [ ... | x ← movs v ]
    where v = ...
              -- posición del hueco en la configuración c
```

Ayuda al Ejercicio 13.6 (pág. 327).– Podemos considerar la configuración

```
data RelojX  = X Int Int
data RelojY  = Y Int Int
type Config  = (RelojX, RelojY, Int)
```

(la tercera componente de una configuración mide el tiempo transcurrido) y como únicos movimientos $vaciar$ y $girar$:

```
vaciar (X x x', Y y y', t)
  | y≥ x && x > 0 = [ (X 0 tx, Y (y − x) (y' + x), t + x),
                      (X 0 tx, Y 0        ty ,      t + y) ]
  | ...
girar (X 0 x', Y y y', t) = [ (X tx 0, Y y y', t),
                              (X tx 0, Y y y', t),
                              (X 0 x', Y y' y, t) ]
girar ...
```

donde $girar$ no consume tiempo, mientras que $vaciar$ consume el tiempo necesario hasta que alguno de los relojes quede vacío.

Ayuda al Ejercicio 13.7 (pág. 338).– Si cada torre se representa con una lista ascendente

© ITES-Paraninfo

de enteros (los radios de sus discos), una configuración está formada por tres torres (con una instancia adecuada de la igualdad) y es posible definir tres movimientos $m1$, $m2$, $m3$

$$m1\ (T\ (x : xs)\ ys\ zs)\ =\ \ldots$$

– – la lista de configuraciones obtenidas al
– – mover el primer disco de la primera torre

$m2\ \ldots$
instance *Grafo Torre* **where**
  $suc\ t\ =\ [\ \ldots\ |\ m\ \leftarrow\ [m1, m2, m3],\ \ldots\ ]$

Ayuda al Ejercicio 13.8 (pág. 339).– La función ($) :: $(a\ \rightarrow\ b)\ \rightarrow\ a\ \rightarrow\ b$ del PRELUDE se define mediante $f\ \$\ x\ =\ f\ x$.

Ayuda al Ejercicio 13.9 (pág. 339).– Utiliza directamente lo demostrado en Ejercicio 13.8 y la definición de ($).

Ayuda al Ejercicio 13.10 (pág. 339).– Piensa en las expresiones de *map* y de *concat* en términos de un plegado conocido, e intenta expresar todo como un único plegado.

Ayuda al Ejercicio 13.11 (pág. 339).– Considera las estructuras

type *Ficha* $= (Int, Int)$
type *Fichas* $= [Ficha]$

para representar las fichas de dominó, la representación de las casillas

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

y un par de números para indicar la posición de una ficha (dos casillas contiguas); una lista de éstos indica los huecos disponibles para colocar fichas

type *Posic* $= (Int, Int)$
type *Disp* $= [Posic]$
type *Solu* $= (Disp, Fichas)$

Diseña entonces funciones para generar los posibles huecos iniciales, colocar las fichas, y comprobar si cierta configuración es un cuadrado mágico:

$solus\ ::\ TamDom\ \rightarrow\ Disp\ \rightarrow\ ConMag\ \rightarrow\ [Solu]$
$solus\ tdom\ disp\ cmag\ =\ domino\ [\,]\ [\,]\ disp$ **where**
  $fichas\ =\ [\ (x, y)\ |\ x \leftarrow [0..tdom],\ \ldots\ ]$
  $domino\ \ldots$

© ITES-Paraninfo

	1	2	3	4	5	6
1						
2			X		X	
3		X				X
4				C		
5		X				X
6			X		X	

Figura 20.8: Salto de un caballo en un tablero 6×6..

donde la función *domino* tiene como argumentos una lista *os* de posiciones ocupadas, una lista *us* de fichas colocadas y una lista *ls* de posiciones libres, mientras que el primer argumento de *solus* indica el tamaño del dominó a considerar.

Ayuda al Ejercicio 13.13 (pág. 342).– Para una localización de las casillas del tablero en base a un número de fila y un número de columna (ver Figura 20.8), el caballo puede saltar desde cierta casilla (por ejemplo, $(4, 4)$) a las casillas dadas por la lista

$$[\ (x + dx, y + dy) \mid (dx, dy) \leftarrow\ [(-2, -1), (-2,\ 1), \ldots],\ \ldots]$$

A partir de esta idea es fácil construir una instancia adecuada de la clase *Grafo*.

Ayuda al Ejercicio 13.14 (pág. 342).– La codificación de casillas

$$\begin{array}{ccc} 0 & 3 & 6 \\ 5 & & 1 \\ 2 & 7 & 4 \end{array}$$

tiene un clara ventaja; por ejemplo, desde la casilla 3 se puede saltar a la casilla 2$(= 3-1)$ o a la casilla 4$(= 3 + 1)$; una lista de 4 enteros ($T\ [b, b', n, n']$) permite representar una determinada situación, por lo que una instancia adecuada para la igualdad de situaciones y listas de la forma

$$\begin{array}{l} mueve\ 1\ (T\ [b, b', n, n'])\ =\ [\ T\ [b'', b', n, n'] \mid b''\ \leftarrow\ \ldots] \\ mueve\ 2\ \ldots \end{array}$$

permiten construir una instancia de la clase *Grafo*.

Ayuda al Ejercicio 13.15 (pág. 342).– Si el número de vagones es arbitrario, una lista de objetos tal como $[A, A, L, B]$ permite representar cada tren (donde L representa la *locomotora*); ya todo consiste en *saber partir* tales listas en dos trenes de forma que el tren que se separa (para unirse a otro) deberá contener a la locomotora. Finalmente, tres trenes permiten representar cada situación:

$$\begin{array}{lll} \textbf{data}\ \mathit{Pieza} & = L \mid A \mid B\ \textbf{deriving}\ \ldots \\ \textbf{type}\ \mathit{Tren} & = [\mathit{Pieza}] \\ \textbf{data}\ \mathit{Config} & = C\ \mathit{Tren}\ \mathit{Tren}\ \mathit{Tren}\ \textbf{deriving}\ \ldots \end{array}$$

© ITES-Paraninfo

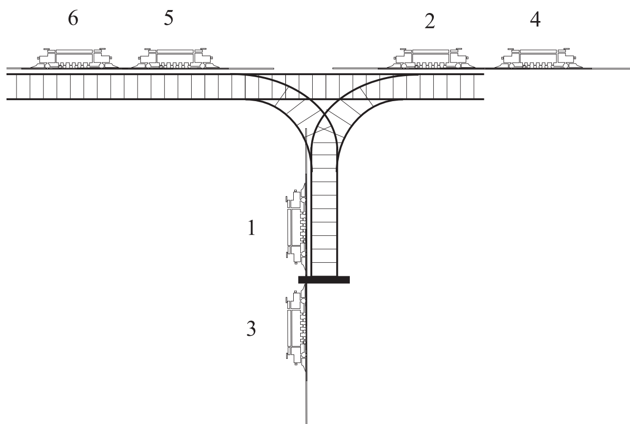

Figura 20.9: El estado $T[5..n][1,3][2,4]$..

y la función sucesor que captura los movimientos podrá describirse a partir de expresiones de la forma

$$[\ c' \mid m \leftarrow\ [girarIzda, salirVía, \dots],\ c' \leftarrow\ m\ c\ ]$$

Ayuda al Ejercicio 13.16 (pág. 343).– Numera los vértices con enteros, considera un par de enteros como descriptor de un arco, y modifica la función $caminoDesde$ ya que la condición de *final de búsqueda* depende de la longitud de la lista de trazos ya realizados. Otra observación: todas las soluciones son "igual de buenas".

Ayuda al Ejercicio 13.17 (pág. 343).– Tres listas permiten describir cada configuración, como aparece en la Figura 20.9. Los posibles movimientos son tres

$direc$ $\equiv$ ⟨trasladar la cabeza de la primera lista a la tercera⟩
$aPila$ $\equiv$ ⟨idem de la primera a la segunda⟩
$dPila$ $\equiv$ ⟨idem de la segunda a la tercera⟩

Descríbanse pues las tres funciones; para la función sucesor de la clase $Grafo$ podemos intentar expresiones tales como

$$[\ mov\ c \mid mov\ \leftarrow\ [aPila\ , dPila\ , direc]\ ]$$

pero hay un problema: por ejemplo, $aPila\ (T\ [\,]\ [1,3][2,4])$ no tendría sentido. Otra observación: $direc$ es la composición $dPila\ .\ aPila$, con lo que podemos simplificar algo la solución. Y finalmente, también podemos observar que si llamamos $a_{n,k}$ al número de nodos finales (formas posibles de extraer las locomotoras) desde un nodo con n elementos en la primera lista y k elementos en la segunda, el juego de movimientos conduce a la recurrencia

$$(B1) \qquad a_{n,k}\ = a_{n-1,k+1} + a_{n,k-1}, \qquad n,k\ \geq\ 1$$

© ITES-Paraninfo

$$(B2) \qquad a_{0,p} = 1$$

cuyas soluciones son los *números de Catalán generalizados*; inténtese resolver la recurrencia para valores sucesivos de k y conjeture la forma de la solución.

20.9. ANALIZADORES

Ayuda al Ejercicio 14.1 (pág. 355).– Piensa en una solución recursiva de la forma

$$\begin{array}{lll} rString\ [\,] & = \ldots & \\ rString\ (c:cs) & = \ldots & \ggg \ldots \rightarrow \\ & \quad rString\ cs & \ggg \ \ldots \rightarrow \\ & \quad \ldots & \end{array}$$

Ayuda al Ejercicio 14.2 (pág. 357).– ¿Qué devuelve $r\acute{A}rbol$ "< 2,¡3,4>>"?

Ayuda al Ejercicio 14.4 (pág. 366).– Prueba en primer lugar

$$\begin{array}{lll} aplica\ (iter\ m)\ y & = & aplica\ (\textbf{do}\ a \leftarrow\ m;\ x \leftarrow\ iter\ m;\ return\ (a:x))\ y +\!\!+ [([\,],y)] \\ k\ a\ y' & = & \textbf{do}\ (a',y'') \leftarrow\ aplica\ (iter\ m)\ y';\ return\ (a:a',y'') \end{array}$$

donde

$$k \qquad \equiv \qquad \lambda\ a\ \rightarrow\ iter\ m\ >>=\ \lambda\ x\ \rightarrow return\ (a:x)$$

Ayuda al Ejercicio 14.5 (pág. 366).– A partir del tipo $Analiz$ del texto y de sus instancias para $Monad$ y $MonadPlus$ se pueden construir fácilmente analizadores para cada categoría, por ejemplo

$$\begin{array}{l} anaVar, anaCon, anaTer\ ::\ Analiz\ (Ter\ Idv) \\ anaVar\ =\ \textbf{do}\ c\ \leftarrow\ elemento\ !>\ isUpper;\ r\ \leftarrow anaIden;\ return\ (Var\ (c:r)) \\ anaCon\ =\ \textbf{do}\ c\ \leftarrow\ elemento!>\ isLower;\ r\ \leftarrow\ anaIden;\ return\ (Con\ (c:r)) \\ \ldots \\ \\ anaReg\ ::\ Analiz\ Regla \\ anaReg\ = \quad \textbf{do}\ oc\ \leftarrow\ anaObj;\ _\ \leftarrow literal\ '.';\ return\ (oc\ :-[\,]) \\ \qquad\qquad !+\ \textbf{do}\ oc\ \leftarrow\ anaObj;\ \ldots;\ return\ (oc\ :-(o:os)) \end{array}$$

donde $elemento$, $literal$, $!>$ y $!+$ también están definidos en el texto.

Ayuda al Ejercicio 14.6 (pág. 367).– Una vez que se disponga de un analizador $identif$ para identificadores (el primer carácter en minúsculas y el resto de caracteres alfanuméricos), el analizador requerido será una traducción directa de la sintaxis del λ-cálculo, por ejemplo

© ITES-Paraninfo

$$
\begin{aligned}
lam \ = \ & \mathbf{do} \\
& \quad literal\ '\lambda' \\
& \quad x \ \leftarrow \ identif \\
& \quad literal\ '.' \\
& \quad t \ \leftarrow \ term \\
& \quad return\ (Lam\ x\ t)
\end{aligned}
$$

siempre y cuando utilicemos un analizador de secuencias que permita resolver la recursión por la izquierda en la definición de los λ-términos:

$$
\begin{aligned}
& chainl1 \ :: Analiz\ a \ \rightarrow \ Analiz\ (a \ \rightarrow \ a \ \rightarrow \ a) \ \rightarrow \ Analiz\ a \\
& p`chainl1`\ op \ = \ \mathbf{do} \\
& \qquad x \leftarrow \ p \\
& \qquad s \leftarrow \ iter\ (\mathbf{do}\{f \ \leftarrow \ op;\ y \ \leftarrow \ p;\ return\ (f, y)\}) \\
& \qquad return\ (foldl(\lambda\ a\ (f, y) \ \rightarrow \ f\ a\ y)\ x\ s)
\end{aligned}
$$

Ayuda al Ejercicio 14.7 (pág. 367).– Usa el transformador de estados definido en el Capítulo 11 para mantener un estado global que sea un número entero inicialmente nulo que se incrementará cada vez que se "invente" un nombre nuevo a partir del último inventado:

$$
\begin{aligned}
newName \ & = \ \mathbf{do} \\
& \qquad n \ \leftarrow \ newVar \\
& \qquad return\ (mkName\ n) \\
newVar \ & = \ T\ (\lambda\ e \ \rightarrow \ (e+1, e)) \\
mkName\ n \ & = \ \text{"x"} +\!\!+\ show\ n
\end{aligned}
$$

20.10. SIMULACIÓN

Ayuda al Ejercicio 15.1 (pág. 372).– Si llamamos sección a cada trozo de la función f se puede pensar en una solución de la forma

$$
\begin{aligned}
& buscaProyecci\acute{o}n(a', b')\ (a, b) \ = \ pintasecciones(secciona\ vs) \\
& \quad \mathbf{where}\ vs = \ [\ (t,\ transformaEntreIntervalos\ (a', b')\ (a, b)\ t) \mid t \leftarrow [a'..b']] \\
& \qquad\quad secciona\ \ldots \\
& \qquad\quad pintasecciones\ \ldots
\end{aligned}
$$

Ayuda al Ejercicio 15.2 (pág. 386).– Usa un mismo tipo para el resultado de un escrutinio y para el resultado de la simulación de la primitiva.

Ayuda al Ejercicio 15.3 (pág. 388).– Utiliza campos etiquetados.

Ayuda al Ejercicio 15.4 (pág. 391).– Observa que $f\ .\ g\ x$ equivale a $f\ .\ (g\ x)$.

20.11. TÉCNICAS DE PROGRAMACIÓN Y TRANSFORMACIONES DE PROGRAMAS

Ayuda al Ejercicio 16.4 (pág. 404).– Demuestra las siguientes implicaciones:

(a) $(*)$ distributiva a derecha $\Rightarrow$ $(*)$ conmutativa

(b) $(*)$ distrib. a derecha $\wedge$ $(*)$ conmut. $\Rightarrow$ $(*)$ distrib. a izquierda

(c) $(*)$ distributiva a izquierda $\Rightarrow$ $(*)$ asociativa

con lo cual sólo sería necesario demostrar la propiedad distributiva a la derecha.

Ayuda al Ejercicio 16.5 (pág. 405).– Definiendo

$$\begin{array}{lll} (def1) & I & = Suc\ O \\ (def-) & -\,x & = O\ -\ x \end{array}$$

demostrar algunas propiedades útiles, como

$$\begin{array}{lll} (Suc(b) = 1 + b) & Suc\ b & = I + b \\ (Pre(b) = b - 1) & Pre\ b & = b - I \\ (Pre(0) = -1) & -\,I & = Pre\ O \\ (cs1) & Pre\ (-x) & = -\ Suc\ x \\ (cs2) & -\ Pre\ x & = Suc\ (-x) \\ (iden0) & -\,O & = O \\ (idenx) & -\,(-x) & = x \end{array}$$

Para obtener ciertas propiedades de las nuevas suma y diferencia ($(+)$ y $(-)$ para Ent) es necesario considerar algunos axiomas. Es suficiente considerar el axioma

$$(Ax1) \qquad Pre\ I = O$$

del cual se deducen fácilmente

$$\begin{array}{lll} (i1) & Pre\ (Suc\ x) & = x \\ (i2) & Suc\ (Pre\ x) & = x \\ (c1) & Pre\ a\ +\ b & = a\ +\ Pre\ b \\ (c2) & Suc\ a\ +\ b & = a\ + Suc\ b \\ (r1) & x\ -\ Suc\ y & = Pre\ x\ -\ y \\ (r2) & x\ -\ Pre\ y & = Suc\ x\ -\ y \end{array}$$

La demostración de la propiedad asociativa, o de la propiedad

$$x\ -\ (y\ +\ z) \qquad = \qquad (x\ -\ y)\ -\ z$$

son fáciles, así como la conmutatividad de $(+)$ para Ent a partir de la propiedad

© ITES-Paraninfo

$$x - y = O \quad \Rightarrow \quad x = y$$

Ayuda al Ejercicio 16.9 (pág. 410).– Introduce un tercer parámetro acumulador.

Ayuda al Ejercicio 16.14 (pág. 418).– Procédase por inducción estructural sobre xs y utilizar la distributividad de $(+)$ sobre $(*)$ en el paso inductivo.

Ayuda al Ejercicio 16.18 (pág. 425).– Parte de la función

$$\begin{aligned} &lult\ [x] &&= [x] \\ &lult\ u@(_ : _ : _) &&= reemplaza\ (ultl\ u)\ u \end{aligned}$$

que cumple los requisitos, pero da dos pasadas.

Ayuda al Ejercicio 16.19 (pág. 425).– Considérese una función:

$$fult\ ::\ [a] \rightarrow (a,\ a \rightarrow [a])$$

de forma que $fult\ xs$ devuelve un par $(u,\ f)$ donde u es el último de la lista y f tiene el mismo comportamiento que la función $reemplaza$, de modo que la siguiente función devuelve la lista de últimos

$$lult'\ xs\ =\ f\ u\ \mathbf{where}\ (u, f)\ =\ fult\ xs$$

Ayuda al Ejercicio 16.20 (pág. 425).– Parte de la función

$$\begin{aligned} &f\ Vacío &&= \lambda\ _\ \rightarrow\ Vacío \\ &f\ (Nodo\ i\ x\ d) &&= \lambda\ y\ \rightarrow\ Nodo\ (f\ i\ y)\ y\ (f\ d\ y) \end{aligned}$$

que reemplaza todos los elementos de un árbol por uno dado.

Ayuda al Ejercicio 16.21 (pág. 425).– Generaliza la solución del Ejercicio 16.20.

Ayuda al Ejercicio 16.23 (pág. 443).– Se puede proceder por inducción estructural sobre la primera lista utilizando las siguientes propiedades en el paso inductivo:

$$long\ (a : b)\ \geq\ 1 \qquad min\ (1 + a)\ (1 + b)\ =\ 1 +\ min\ a\ b$$

Ayuda al Ejercicio 16.24 (pág. 443).– Procédase por inducción sobre u.

Ayuda al Ejercicio 16.26 (pág. 444).– Introduce una función con un parámetro acumulador:

$$\begin{aligned} &inv2\ (x : xs)\ w &&= \ldots \\ &inv2\ \ldots \\ &inv\ w &&= inv2\ w\ [\,] \end{aligned}$$

© ITES-Paraninfo

Ayuda al Ejercicio 16.27 (pág. 444).– Para la demostración proceda de la forma siguiente: partiendo de $parFib (n + 2)$, desplegar dos veces la ecuación y simplifique los cualificadores para después utilizar la función $(+ :)$.

Ayuda al Ejercicio 16.28 (pág. 444).– Si lo hacemos por inducción sobre n, habrá que utilizar $f_0 = 0, f_1 = 1$ en el caso base; para el caso inductivo obsérvese que $f_{(n+1)+m}$ puede escribirse en la forma $f_{n+(m+1)}$.

Ayuda al Ejercicio 16.32 (pág. 445).– Se trata de probar por inducción sobre a:

$$\forall\, a, r \,\centerdot\, a :: Arbol\ [\alpha],\ r :: [\alpha] \,\centerdot \\ aplanaCon\ a\ r \quad = \quad aplana\ a +\!\!+\ r$$

Ayuda al Ejercicio 16.33 (pág. 446).– Ténganse en cuenta como ecuaciones las siguientes:

$$\begin{array}{ll} nClaves\ Vacío & = 0 \\ nClaves\ (Nodo\ i\ x\ d) & = nClaves\ i\ +\ nClaves\ d\ +\ 1 \\ \text{-- -- o, equivalentemente} & \\ nClaves\ (Nodo\ i\ x\ d) & = (\lambda\ _\ ni\ nd\ \rightarrow\ ni + nd + 1) \\ & \quad (nClaves\ i)\ (nClaves\ d) \end{array}$$

para deducir el primer apartado. Para el tercer apartado considérese una función auxiliar

$$equiAux\ a\ =\ (nClaves\ a,\ equia)$$

y trátese de transformar las ecuaciones (utilizando D/P)

$$\begin{array}{ll} equiAux\ Vacío & = \ldots \\ equiAux\ (Nodo\ i\ x\ d) & = \ldots \\ \quad \mathbf{where}\ (ni, fi) & = equiAux\ i \\ \qquad\qquad (nd, fd) & = equiAux\ d \end{array}$$

Una vez completada la segunda ecuación, razónese igual que para el apartado 2. Para el apartado 5 basta completar las ecuaciones

$$\begin{array}{ll} a\acute{A}rbol\ [\,] & = \ldots \\ a\acute{A}rbol\ (x : xs) & = Nodo\ \ldots\ x\ \ldots \\ \quad \mathbf{where}\ (ys, zs) & =\ splitAt\ (length\ xs\ `div`\ 2)\ xs \end{array}$$

ya que la función $aLista$ tiene por ecuaciones:

$$\begin{array}{ll} aLista\ Vacío & = [\,] \\ aLista\ (Nodo\ i\ x\ d) & = x\ :\ aLista\ i +\!\!+ aLista\ d \end{array}$$

Ayuda al Ejercicio 16.35 (pág. 447).– Para los apartados 1 y 2 sígase el esquema de las ayudas del Ejercicio 16.33; para el apartado 3 podemos tomar la función auxiliar

© ITES-Paraninfo

$$sumYnEle\ p\ =\ (suma\ p,\ nEle\ p)$$

para probar después

$$\begin{array}{ll} sumYnEle\ Vacia & = (0,\ 0) \\ sumYnEle\ (Cima\ x\ p) & = (x + s, 1 + p)\ \mathbf{where}\ (s, p)\ =\ sumYnEle\ p \end{array}$$

Para el apartado 4 interesa una función con dos parámetros, uno de los cuales actúa de acumulador

$$\begin{array}{ll} aPila2\ ::\ [a]\ \to\ Pila\ a\ \to\ Pila\ a & \\ aPila2\ [\,]\qquad p & = p \\ aPila2\ (x : xs)\ p & = aPila2\ xs\ (Cima\ x\ p) \end{array}$$

para después demostrar y utilizar la propiedad

$$\begin{array}{lll} & \forall xs, p\ \bullet & xs :: [a], p :: Pila\ a\ \bullet \\ (*) & & aLista\ (aPila2\ xsp)\ =\ inv\ xs \mathbin{+\!\!+} aLista\ p \end{array}$$

Ayuda al Ejercicio 16.39 (pág. 449).– En la primera equivalencia se exige que x no aparezca en e', pues si x apareciera en e' no se tendría la equivalencia. En la segunda equivalencia hay que imponer que $m\ \not\equiv\ y$, pues en el caso de que $m\ \equiv y$ las sentencias no serían equivalentes.

20.12. INTRODUCCIÓN AL λ-CÁLCULO

Ayuda al Ejercicio 17.48 (pág. 497).– Obsérvese que tenemos

$$(par\ <|)\ ::\ [Int]\ \to\ [Int]$$

Por otro lado, inténtese evaluar las expresiones

$$((inc\#).(par\ <|))\ u \qquad\qquad ((par\ <|).(inc\ \#))\ u$$

siendo u cierta lista de enteros.

© ITES-Paraninfo