Metodología de la Programación

Grado en Ingeniería Informática de Gestión y Sistemas de Información
Escuela de Ingeniería de Bilbao (UPV/EHU)
Departamento de Lenguajes y Sistemas Informáticos
Curso: 1^o
Curso académico: 2021-2022
Grupo 01
Tema 5: Especificación ecuacional de los tipos abstractos de datos (TAD)
2,5 puntos

05-05-2022

Solución

Índice

Lista de figuras

Lista de tablas

****

1 Especificación ecuacional — Listas (*0,750 puntos*)

1.1 Especificación ecuacional de la función $sustituir$ (0,550 puntos)

Especificar ecuacionalmente la función $sustituir$ que, dados un entero p, un valor x de tipo t y una lista cuyos componentes son de tipo t, devuelve la lista que se obtiene sustituyendo el elemento de la posición p por x.

Las posiciones se cuentan empezando desde la izquierda y se considera que el elemento que está más a la izquierda ocupa la posición 1.

Si p es menor o igual que 0 o p es mayor que la longitud de la lista, entonces se ha de mostrar un mensaje de error. Por tanto, si la lista inicial es vacía, se ha de mostrar un mensaje de error.

Además de las operaciones constructoras [ ] y (:), se puede utilizar la siguiente función auxiliar sin necesidad de definirla:

- $longitud$, que dada una lista, devuelve el número de elementos de la lista.

Ejemplos

- $sustituir(3, 10, [5, 8, \underline{7}, 20, 4, 11]) = [5, 8, \underline{10}, 20, 4, 11]$
  El elemento de la posición 3, empezando por la izquierda, es decir, 7, ha sido sustituido por 10.
- $sustituir(6, 10, [5, 8, 7, 20, 4, \underline{11}]) = [5, 8, 7, 20, 4, \underline{10}]$
  El elemento de la posición 6, empezando por la izquierda, es decir, 11, ha sido sustituido por 10.
- $sustituir(1, 18, [\underline{5}, 8, 7, 20, 4, 11]) = [\underline{18}, 8, 7, 20, 4, 11]$
  El elemento de la posición 1, empezando por la izquierda, es decir, 5, ha sido sustituido por 18.
- $sustituir(1, 18, [\underline{5}]) = [\underline{18}]$
  El elemento de la posición 1, empezando por la izquierda, es decir, 5, ha sido sustituido por 18.

Solución

$$sustituir :: (Int, t, [t]) -> \ [t]$$

$$sustituir(p, x, [\,]) \qquad = error \text{ ”Lista vacía.”} \qquad (1^{\text{a}}.\ ec.)$$

$$\begin{array}{lll} sustituir(p, x, y : s) & & \\ \quad | \ p < 1 \ || \ p > longitud(y : s) & = error \text{ ”La posición no es adecuada.”} & (2^{\text{a}}.\ ec.) \\ \quad | \ p == 1 & = x : s & (3^{\text{a}}.\ ec.) \\ \quad | \ otherwise & = y : sustituir(p - 1, x, s) & (4^{\text{a}}.\ ec.) \end{array}$$

1.2 Desarrollo de un ejemplo para la función $sustituir$ (0,200 puntos)

Una vez dadas las ecuaciones, desarrollar paso a paso el siguiente ejemplo. En cada paso hay que indicar qué ecuación se ha utilizado:

$$sustituir(3, 10, [5, 8, 7, 20, 4, 11])$$

Solución

$$sustituir(3, 10, [5, 8, 7, 20, 4, 11]) =$$

$$= sustituir(\underbrace{3}_{p}, \underbrace{10}_{x}, \underbrace{5}_{y} : \underbrace{8 : 7 : 20 : 4 : 11 : [\,]}_{s}) = (4^{\text{a}}.\ ec.)$$

$$= 5 : sustituir \underbrace{2}_{p}, \underbrace{10}_{x}, \underbrace{8}_{y} : \underbrace{7 : 20 : 4 : 11 : [\,]}_{s}) = (4^{\text{a}}.\ ec.)$$

$$= 5 : 8 : sustituir(\underbrace{1}_{p}, \underbrace{10}_{x}, \underbrace{7}_{y} : \underbrace{20 : 4 : 11 : [\,]}_{s}) = (3^{\text{a}}.\ ec.)$$

$$= 5 : 8 : 10 : 20 : 4 : 11 : [\,] =$$

$$= [5, 8, 10, 20, 4, 11]$$

2 Inducción sobre listas (*1,000 puntos*)

2.1 Especificación ecuacional de la función $suma$ (0,050 puntos)

Especificar ecuacionalmente la función $suma$ que, dada una lista de enteros, devuelve la suma de los elementos de la lista.

$$suma :: ([Int]) \to Int$$

Ejemplos

▷ $suma([8,5,9,5]) = 27$ ▷ $suma([80]) = 80$ ▷ $suma([\,]) = 0$

Solución

$$\begin{array}{lll} suma :: ([Int]) \texttt{ -> } Int & & \\ suma([\,]) & = 0 & (1^{\text{a}}\ ec.) \\ suma(a : b) & = a + suma(b) & (2^{\text{a}}\ ec.) \end{array}$$

2.2 Especificación ecuacional de la función $incr$ (0,100 puntos)

Especificar ecuacionalmente la función $incr$ que, dada una lista de elementos de tipo Int, devuelve la lista que se obtiene incrementando en 1 el valor de cada elemento de la lista de entrada.

$$incr :: ([Int]) \to [Int]$$

Ejemplos

▷ $incr([5,7,5,9]) = [6,8,6,10]$ ▷ $incr([0,0,0]) = [1,1,1]$
▷ $incr([4]) = [5]$ ▷ $incr([\,]) = [\,]$

Solución

$$\begin{array}{lll} incr :: ([Int]) \texttt{ -> } [Int] & & \\ incr([\,]) & = [\,] & (3^{\text{a}}\ ec.) \\ incr(a : b) & = (a+1) : incr(b) & (4^{\text{a}}\ ec.) \end{array}$$

2.3 Especificación ecuacional de la función $longitud$ (0,050 puntos)

Especificar ecuacionalmente la función $longitud$ que, dada una lista de elementos de tipo t, calcula el número de elementos de la lista.

$$longitud :: ([t]) \to Int$$

Ejemplos

▷ $longitud([8,5,9,5]) = 4$ ▷ $longitud([80]) = 1$ ▷ $longitud([\,]) = 0$

Solución

$$\begin{array}{lll} longitud :: ([t]) \texttt{ -> } Int & & \\ longitud([\,]) & = 0 & (5^{\text{a}}\ ec.) \\ longitud(a : b) & = 1 + longitud(b) & (6^{\text{a}}\ ec.) \end{array}$$

2.4 Prueba por inducción (0,150 + 0,650 puntos)

Probar por inducción que para cualquier lista de enteros s, se cumple la siguiente propiedad.

$$suma(incr(s)) = suma(s) + longitud(s)$$

La inducción debe realizarse sobre la lista s considerando el caso básico $s = [\,]$ y el caso inductivo $s = x : w$. La hipótesis de la inducción consistirá en suponer que para la sublista w se cumple la propiedad que se está probando.

2.5 Propiedades a utilizar

Se necesitan las siguientes propiedades:

- ⬡ $(Prop\ 1)$: 0 es elemento neutro para la suma: $i + 0 = 0 + i = i$.
- ⬡ $(Prop\ 2)$: La suma es asociativa: $(i + j) + k = i + (j + k)$.
- ⬡ $(Prop\ 3)$: La suma es conmutativa: $i + j = j + i$.

Solución

2.5.1 Caso simple: $s = [\,]$

Cuando se cumple $s = [\,]$, la pregunta es la siguiente:

$$¿suma(incr([\,])) = suma([\,]) + longitud([\,])?$$

2.5.1.1 Comprobación de la igualdad Se ha de comprobar que los dos lados de la igualdad son realmente iguales.

$$¿\underbrace{suma(incr([\,]))}_{\alpha} = \underbrace{suma([\,]) + longitud([\,])}_{\beta}?$$

Desarrollo de α y β: En cada paso, se indicará qué ecuación o propiedad se va a utilizar, a qué parte de la expresión se le aplicará la ecuación o la propiedad, y con qué elementos de la ecuación o la propiedad se asocian los elementos de la expresión que se tiene en ese paso.

- **Desarrollo de α:**

$$\alpha =$$

$$= suma(\underbrace{incr([\,])}_{3^{\text{a}}\ ec.}) =$$

$$= \underbrace{suma([\,])}_{1^{\text{a}}\ ec.} =$$

$$= 0$$

- **Desarrollo de β:**

$$\beta =$$

$$= \underbrace{suma([\,])}_{1^{\text{a}}\ ec.} + longitud([\,]) =$$

$$= 0 + \underbrace{\underbrace{longitud([\,])}_{i}}_{Prop\ 1} =$$

$$= \underbrace{longitud([\,])}_{5^{\text{a}}\ ec.} =$$

$$= 0$$

2.5.1.2 Conclusión En ambos lados (α y β), se ha obtenido el mismo valor. Por tanto, en el caso $s = [\,]$, se cumple la propiedad que estamos probando.

2.5.2 Caso inductivo: $s = x : w$

Cuando $s = x : w$, la pregunta es la siguiente:

$$¿suma(incr(x : w)) = suma(x : w) + longitud(x : w)?$$

2.5.2.1 Hipótesis de la inducción (h.i.): Se ha de considerar que la ecuación se cumple para la sublista w de s:

$$\underbrace{suma(incr(w))}_{\lambda} = \underbrace{suma(w) + longitud(w)}_{\pi}$$

2.5.2.2 Comprobación de la igualdad Se ha de comprobar que los dos lados de la igualdad son iguales en el caso inductivo.

$$¿\underbrace{suma(incr(x : w))}_{\gamma} = \underbrace{suma(x : w) + longitud(x : w)}_{\delta}?$$

Desarrollo de γ y δ: En cada paso, se indicará qué ecuación o propiedad se va a utilizar, a qué parte de la expresión se le aplicará la ecuación o la propiedad, y con qué elementos de la ecuación o la propiedad se asocian los elementos de la expresión que se tiene en ese paso.

- **Desarrollo de γ:**

$$\gamma =$$

$$= \underbrace{suma(incr(\underbrace{x}_{a} : \underbrace{w}_{b}))}_{4^{a}\ ec.} =$$

$$= \underbrace{suma(\underbrace{(x+1)}_{a} : \underbrace{incr(w)}_{b})}_{2^{a}\ ec.} =$$

$$= (x+1) + \underbrace{\underbrace{suma(incr(w))}_{\lambda}}_{h.i.} =$$

$$= (x+1) + (\underbrace{suma(w) + longitud(w)}_{\pi})$$

- **Desarrollo de δ:**

$$\delta =$$

$$= \underbrace{suma(\underbrace{x}_{a} : \underbrace{w}_{b})}_{2^{\text{a}}\ ec.} + longitud(x : w) =$$

$$= (x + suma(w)) + \underbrace{longitud(\underbrace{x}_{a} : \underbrace{w}_{b})}_{6^{\text{a}}\ ec.} =$$

$$= \underbrace{\underbrace{(x + suma(w))}_{i} + (\underbrace{1}_{j} + \underbrace{suma(w)}_{k})}_{Prop\ 2} =$$

$$= \underbrace{((\underbrace{x}_{i} + \underbrace{suma(w)}_{j}) + \underbrace{1}_{k})}_{Prop\ 2} + longitud(w) =$$

$$= (x + \underbrace{(\underbrace{suma(w)}_{i} + \underbrace{1}_{j})}_{Prop\ 3}) + longitud(w) =$$

$$= \underbrace{(\underbrace{x}_{i} + (\underbrace{1}_{j} + \underbrace{suma(w)}_{k}))}_{Prop\ 2} + longitud(w) =$$

$$= \underbrace{(\underbrace{(x + 1)}_{i} + \underbrace{suma(w)}_{j}) + \underbrace{longitud(w)}_{k}}_{Prop\ 2} =$$

$$= (x + 1) + (\underbrace{suma(w) + longitud(w)}_{\pi})$$

2.5.2.3 Conclusión En el caso inductivo, hemos desarrollado γ y δ y hemos comprobado que son iguales. Por tanto, en el caso $s = x : w$, se cumple la propiedad que estamos probando.

3 Especificación ecuacional — Pilas (*0,200 puntos*)

3.1 Especificación ecuacional de la función cdp (0,200 puntos)

Especificar ecuacionalmente la función cdp que, dadas dos pilas de elementos de tipo t, devuelve la pila formada por los elementos de la primera pila que aparecen también en la segunda pila y a la misma altura. Por tanto, esos elementos son las coincidencias de las dos pilas. Si alguna de las dos pilas es vacía, se ha de devolver la pila vacía.

Además de las operaciones constructoras $Pvacia$ y $Apilar$, se pueden utilizar las siguientes funciones auxiliares sin necesidad de definirlas:

- es_pvacia, que dada una pila, devuelve $True$ si la pila es vacía y $False$ en caso contrario.
- $altura$, que dada una pila, devuelve el número de elementos de la pila.
- $cima$, que dada una pila, devuelve el elemento que está más arriba.
- $desapilar$, que dada una pila, devuelve la pila que se obtiene al quitar el elemento de la cima (el que está más arriba).

Ejemplos

- En la figura 1 de la página 10, se muestran cuatro ejemplos de aplicación de la función cdp. En cada ejemplo, las coincidencias de las dos pilas se han marcado en amarillo.
- En la tabla 1 de la página 10, se muestra otro ejemplo de aplicación de la función cdp. Las dos pilas de ese ejemplo coinciden solo en una posición o altura. Los elementos de esa altura están marcados en amarillo. Se han utilizado las operaciones constructoras para representar las pilas.

Solución

$$cdp :: (Pila\ t, Pila\ t) -> \ Pila\ t$$

$$cdp(Pvacia, r) = Pvacia \quad (1^{a}\ ec.)$$

$$\begin{array}{lll} cdp(Apilar(x,s),r) & & \\ \quad | \ altura(Apilar(x,s)) > altura(r) & = cdp(s,r) & (2^{a}\ ec.) \\ \quad | \ altura(Apilar(x,s)) < altura(r) & = cdp(Apilar(x,s), desapilar(r)) & (3^{a}\ ec.) \\ \quad | \ x == cima(r) & = Apilar(x, cdp(s, desapilar(r))) & (4^{a}\ ec.) \\ \quad | \ otherwise & = cdp(s, desapilar(r)) & (5^{a}\ ec.) \end{array}$$

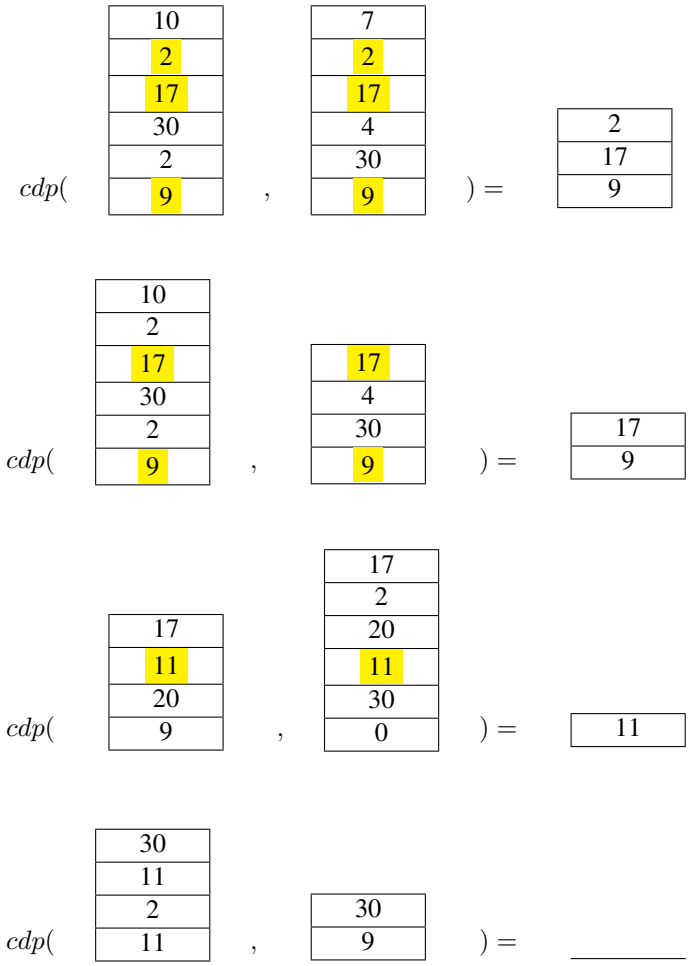

Figura 1: Cuatro ejemplos de aplicación de la función cdp. En cada ejemplo, las coincidencias de las dos pilas se han marcado en amarillo.

$$cdp(Apilar(3, Apilar(9, Apilar(3, Pvacia))), Apilar(9, Apilar(5, Pvacia)))$$
$$= Apilar(9, Pvacia)$$

Tabla 1: Ejemplo de aplicación de la función cdp. Las pilas se han representado mediante las operaciones constructoras. Los elementos de la altura en la que coinciden las dos pilas, están marcados en amarillo.

4 Especificación ecuacional — Colas (*0,150 puntos*)

4.1 Especificación ecuacional de la función $eliminar$ (0,150 puntos)

Especificar ecuacionalmente la función $eliminar$ que, dados un número entero x y una cola de enteros, devuelve la cola que se obtiene eliminando de la cola de entrada las apariciones de x. Si la cola de entrada es vacía, se ha de devolver la cola vacía.

Ejemplos

- $eliminar(7, << 5, 7, 8, 8, 7, 0 >>) =<< 5, 8, 8, 0 >>$
- $eliminar(5, << 5, 7, 8, 8, 7, 0 >>) =<< 7, 8, 8, 7, 0 >>$
- $eliminar(9, << 5, 7, 8, 8, 7, 0 >>) =<< 5, 7, 8, 8, 7, 0 >>$
- $eliminar(9, Poner(Poner(Poner(Cvacia, 2), 9), 2)) = Poner(Poner(Cvacia, 2), 2)$
- $eliminar(7, Cvacia) = Cvacia$

Solución

$$eliminar :: (Int, Cola\ Int) -> \ Cola\ Int$$

$$eliminar(x, Cvacia) \qquad = Cvacia \qquad (1^{a}\ ec.)$$

$$\begin{array}{lll} eliminar(x, Poner(s, y)) & & \\ \quad \mid x == y & = eliminar(x, s) & (2^{a}\ ec.) \\ \quad \mid x /= y & = Poner(eliminar(x, s), y) & (3^{a}\ ec.) \end{array}$$

5 Especificación ecuacional — Árboles binarios (*0,400 puntos*)

5.1 Especificación ecuacional de la función smp (0,400 puntos)

Especificar ecuacionalmente la función smp que, dados dos árboles binarios cuyos elementos son de tipo Int, devuelve el árbol binario formado por las sumas de los nodos situados en la misma posición. Si en uno de los árboles en una posición se tiene un nodo y en el otro árbol no, se pondrá ese valor —es decir, como si en el árbol que no tiene nodo en esa posición, hubiese un 0. Si los dos árboles de entrada son vacíos, la función ha de devolver un árbol vacío.

Además de las operaciones constructoras $Avacio$ y $Crear$, se pueden utilizar las siguientes funciones auxiliares sin necesidad de definirlas:

- es_avacio, que dado un árbol binario, devuelve $True$ si el árbol binario es vacío y $False$ en caso contrario.
- $raiz$, que dado un árbol binario no vacío, devuelve el valor de la raíz.
- $izqdo$, que dado un árbol binario no vacío, devuelve el subárbol izquierdo.
- $drcho$, que dado un árbol binario no vacío, devuelve el subárbol derecho.

Ejemplos

- Por un lado, en la figura 2 de la página 13, tenemos el árbol binario a. Por otro lado, en la figura 3 de la página 13, tenemos el árbol binario b. En la figura 4 de la página 13 se muestra el árbol binario que se obtiene al aplicar la función smp a los árboles binarios a y b. Tal como se puede apreciar en esa figura, se ha realizado la suma cuando en ambos árboles se tiene un nodo en una determinada posición, mientras que en aquellas posiciones en las que en uno de los árboles se tiene un nodo y en el otro árbol no se tiene nodo, se ha puesto el valor del nodo.
- En la figura 5 de la página 13, tenemos el árbol binario c. En la figura 6 de la página 14, tenemos el árbol binario d. En la figura 7 de la página 14 se muestra el árbol binario que se obtiene al aplicar la función smp a los árboles binarios c y d. Cuando en ambos árboles se tiene un nodo en una determinada posición, se ha realizado la suma. En cambio, en aquellas posiciones en las que en uno de los árboles se tiene un nodo y en el otro árbol no se tiene nodo, se ha puesto el valor del nodo.
- En la tabla 2 de la página 14, se muestra la aplicación de la función smp a los árboles binarios c y d. Pero en esta ocasión, el árbol binario c —figura 5 de la página 13—, el árbol binario d —figura 6 de la página 14— y el árbol binario $smp(c, d)$ —figura 7 de la página 14— han sido representados mediante las operaciones constructoras $Avacio$ y $Crear$.

Solución

$$
\begin{array}{lll}
smp :: (Arbin\ Int, Arbin\ Int) & -> \ Arbin\ Int & \\
 & & \\
smp(Avacio, g) & = g & (1^{a}\ ec.) \\
 & & \\
smp(Crear(x, e, f), g) & & \\
\quad \mid es\_avacio(g) & = Crear(x, e, f) & (2^{a}\ ec.) \\
\quad \mid otherwise & = Crear((x + raiz(g)), smp(e, izqdo(g)), smp(f, drcho(g))) & (3^{a}\ ec.)
\end{array}
$$

Los elementos e, f y g son árboles binarios cuyos componentes son de tipo Int. Por su parte, el elemento x es un valor de tipo Int.

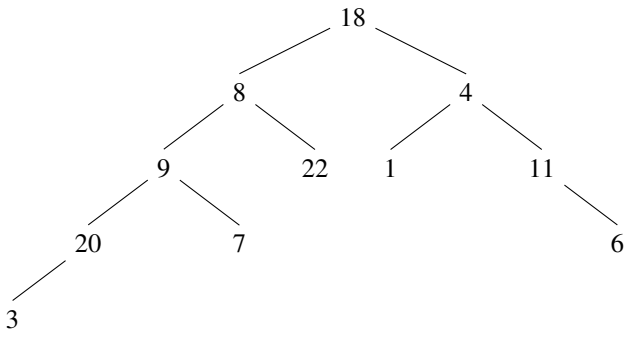

Figura 2: Árbol binario a.

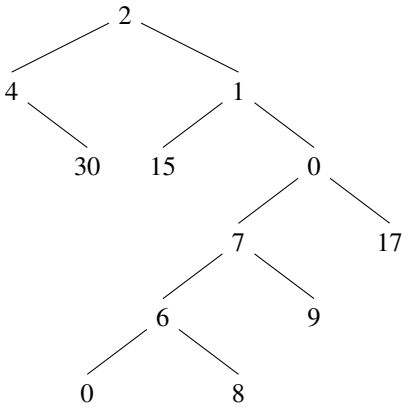

Figura 3: Árbol binario b.

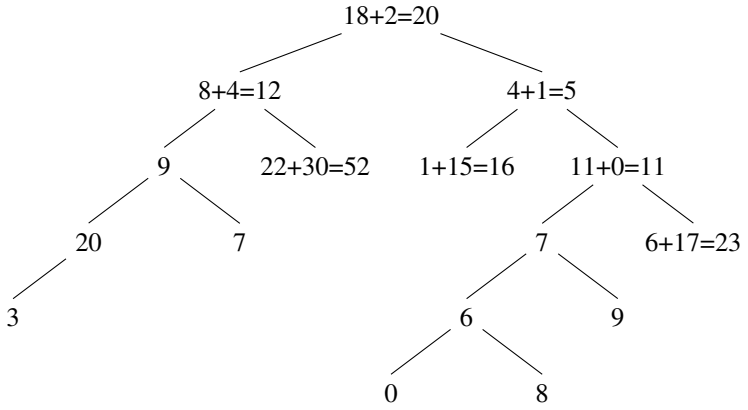

Figura 4: Árbol binario $smp(a, b)$. El árbol a está en la figura 2 de la página 13 y el árbol b está en la figura 3 de la página 13.

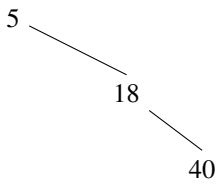

Figura 5: Árbol binario c.

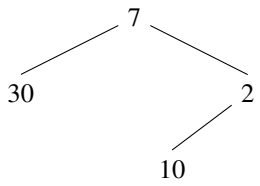

Figura 6: Árbol binario d.

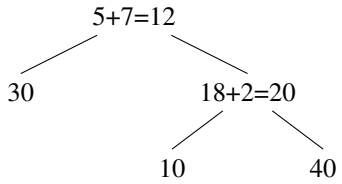

Figura 7: Árbol binario $smp(c, d)$. El árbol c está en la figura 5 de la página 13 y el árbol d está en la figura 6 de la página 14.

$$
\begin{array}{l}
smp(Crear(5, Avacio, Crear(18, Avacio, Crear(40, Avacio, Avacio))),\\
\quad Crear(7, Crear(30, Avacio, Avacio), Crear(2, Crear(10, Avacio, Avacio), Avacio)\\
\qquad)\\
\quad)\\
=\\
\quad Crear(12, Crear(30, Avacio, Avacio),\\
\qquad\quad Crear(20, Crear(10, Avacio, Avacio), Crear(40, Avacio, Avacio))\\
\qquad\quad)
\end{array}
$$

Tabla 2: Ejemplo de aplicación de la función smp. En concreto, se han expresado mediante las operaciones constructoras $Avacio$ y $Crear$, el árbol c que está en la figura 5 de la página 13, el árbol d que está en la figura 6 de la página 14 y el árbol $smp(c, d)$ que está en la figura 7 de la página 14.