METODOLOGÍA DE LA PROGRAMACIÓN
Escuela de Ingeniería de Bilbao (UPV/EHU)
Grado en Ingeniería Informática de Gestión y Sistemas de Información
28 de abril de 2017
Examen Parcial – Tema 5 – Especificación ecuacional de tipos abstractos de datos
Grupo 01 – 2,5 puntos
Solución

Ejercicio 1 (Especificación ecuacional – Listas)

a)

cc:: (Int, t, [t], Int) → [t]

```
cc(n, x, [], p)
 | n < 0 || p <= 0 || p > 1        = error "Datos no adecuados."        (#1)
 | n == 0                          = []                                 (#2)
 | otherwise                       = x : cc(n – 1, x, [], p)            (#3)

cc(n, x, y:s, p)
 | n < 0 || p <= 0 || p > longitud(y:s) + 1   = error "Datos no adecuados."   (#4)
 | n == 0                          = y:s                                (#5)
 | p == 1                          = x : cc(n – 1, x, y:s, p)           (#6)
 | otherwise                       = y : cc(n, x, s, p – 1)             (#7)
```

Solución alternativa que utiliza adicionalmente las funciones primero y resto:

Se podrían unificar el caso de lista vacía ([]) y lista no vacía (y:s), considerando un único parámetro ℓ que abarcase el caso de lista vacía y no vacía:

```
cc(n, x, ℓ, p)
 | n < 0 || p <= 0 || p > longitud(ℓ) + 1   = error "Datos no adecuados."   (%1)
 | n == 0                          = ℓ                                  (%2)
 | p == 1                          = x : cc(n – 1, x, ℓ, p)             (%3)
 | otherwise                       = primero(ℓ) : cc(n, x, resto(ℓ), p – 1)   (%4)
```

b) Se utilizarán las ecuaciones de la primera versión, es decir, las escuaciones (#1)-(#7):

```
cc(3, 10, [5, 8, 7], 3)
= cc(3, 10, 5:8:7:[], 3) =                         n = 3, x = 10, y = 5, s = 8:7:[], p = 3
(#7)= 5:cc(3, 10, 8:7:[], 2) =                     n = 3, x = 10, y = 8, s = 7:[], p = 2
(#7)= 5:8:cc(3, 10, 7:[], 1) =                     n = 3, x = 10, y = 7, s = [], p = 1
(#6)= 5:8:10:cc(2, 10, 7:[], 1) =                  n = 2, x = 10, y = 7, s = [], p = 1
(#6)= 5:8:10:10:cc(1, 10, 7:[], 1) =               n = 1, x = 10, y = 7, s = [], p = 1
(#6)= 5:8:10:10:10:cc(0, 10, 7:[], 1) =            n = 0, x = 10, y = 7, s = [], p = 1
(#5)= 5:8:10:10:10:7:[] =
= 5:8:10:10:10:7:[]
```

Al desarrollar el ejemplo, en cada paso se ha coloreado la parte de la fórmula a la que se le aplica la ecuación indicada en la esquina izquierda de la siguiente línea. Se han utilizado dos colores para mejorar la legibilidad.

Ejercicio 2 (Inducción)

a) suma :: ([Int]) → Int

suma([]) = 0 (#1)
suma(*a*:b) = *a* + suma(b) (#2)

b) inversa:: ([t]) → [t]

inversa([]) = [] (#3)
inversa(*a*:b) = inversa(b) ++ (*a*:[]) (#4)

c) Ahora se probará por inducción sobre la lista s la siguiente propiedad, donde s es una lista cualquiera de tipo Int:

suma(s) = suma(inversa(s))

Caso básico: s = []
¿suma([]) = suma(inversa([]))?

✓ suma([]) =
(#1)= 0

✓ suma(inversa([])) =
(#3)= suma([]) =
(#1)= 0

Se cumple porque en ambos lados nos queda 0.

Por tanto en el caso básico se cumple la propiedad.

Caso inductivo: s = z:w

¿suma(z:w) = suma(inversa(z:w))?

Hipótesis de inducción (h.i.): (para la lista w se cumple la propiedad)

suma(w) = suma(inversa(w))

Volviendo a lo que se quiere probar:

¿suma(z:w) = suma(inversa(z:w))?

suma(z:w) =
(#2)= z + suma(w) =
(h.i.)= z + suma(inversa(w))

suma(inversa(z:w)) =
(#4)= suma(inversa(w) ++ (z:[])) =
(Prop)= suma(inversa(w)) + suma(z:[]) =
(#2)= suma(inversa(w)) + (z + suma([])) =
(#1)= suma(inversa(w)) + (z + 0) =
(0 elemento neutro de +)= suma(inversa(w)) + z =
(conmutatividad de +)= z + suma(inversa(w))

Por tanto, también se cumple la propiedad en el caso inductivo.

Nota: Al desarrollar la demostración, en cada paso se ha coloreado la parte de la fórmula a la que se le aplica la ecuación o propiedad indicada en la esquina izquierda de la siguiente línea. Se han utilizado tres colores para mejorar la legibilidad.

Ejercicio 3 (Especificación ecuacional – Pilas)

```
sb:: (t, Pila t) → Pila t

sb(x, Pvacia) = error "Pila vacía."            (#1)

sb(x, Apilar(y, p))
 | es_pvacia(p)       = Apilar(x, Pvacia)       (#2)
 | otherwise          = Apilar(y, sb(x, p))     (#3)
```

Ejercicio 4 (Especificación ecuacional – Colas)

a)

```
estac:: (Int, Cola Int) → Bool

estac(x, Cvacia) = False                        (#1)

estac(x, Poner(s, y))
 | x == y             = True                    (#2)
 | otherwise          = estac(x, s)             (#3)
```

b)

```
cpq:: (Cola Int) → Bool

cpq(Cvacia) = Cvacia                            (#1)

cpq(Poner(s, x))
 | estac(x, s)        = cpq(s)                  (#2)
 | otherwise          = Poner(cpq(s), x)        (#3)
```

Ejercicio 5 (Especificación ecuacional – Árboles binarios)

```
ch:: (Arbin t) → Arbin t

ch(Avacio) = Avacio                                            (#1)

ch(Crear(x, a, b))
 | es_avacio(a) && (not es_avacio(b))     =
          Crear(x, Crear(raiz(b), Avacio, Avacio), ch(b))      (#2)
 | (not es_avacio(a)) && es_avacio(b)     =
          Crear(x, ch(a), Crear(raiz(a), Avacio, Avacio))      (#3)
 | otherwise                              = Crear(x, ch(a), ch(b))   (#4)
```