Práctica 3

1er cuatrimestre 2022

Algoritmos y Estructuras de Datos 1

Integrante	LU	Correo electrónico
Yago Pajariño	546/21	`ypajarino@dc.uba.ar`

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

Índice

3. Práctica 3

3.1. Ejercicio 1

(a) La postcondición se indefine si $0 \leq result < |l|$

`proc` `buscar` (`in` seq: $seq\langle\mathbb{Z}\rangle$, `in` elem: $\mathbb{R}$, `out` result: $\mathbb{Z}$) {

`Pre` $\{elem \in \mathbb{R}\}$

`Post` $\{0 \leq result < |l| \wedge_L l[result] = elem\}$

}

(b) Se indefine con i = 0

`proc` `progresiónGeométricaFactor2` (`in` l: $seq\langle\mathbb{Z}\rangle$, `out` result: Bool) {

`Pre` {true}

`Post` $\{result = \text{true} \iff ((\forall i : \mathbb{Z})(0 \leq i < |l| - 1 \longrightarrow_L l[i+1] = 2.l[i]))\}$

}

(c) No se define "x"

`proc` `mínimo` (`in` l: $seq\langle\mathbb{Z}\rangle$, `out` result: $\mathbb{Z}$) {

`Pre` {true}

`Post` $\{(\forall y : \mathbb{Z})(y \in l \to y \geq result) \wedge result \in l\}$

}

3.2. Ejercicio 2

(a) $l = seq\langle 1, 2, 3\rangle$ y $suma = 7$

(b) El problema con los límites es que no determina si un valor intermedio resulta producto de la suma de un subconjunto de elementos de l. Ej. $l = seq\langle 1, 3\rangle$ y $suma = 2$

(c) TODO

3.3. Ejercicio 3

(a)
   a) 0
   b) $\{-1, 1\}$
   c) $\{-\sqrt{27}, \sqrt{27}\}$

(b)
   a) 3
   b) $\{0, 3\}$
   c) $\{0, 1, 2, 3, 4, 5\}$

   a) 3
   b) 0
   c) 0

(c) Tienen la misma salida en secuencias sin elementos repetidos.

3.4. Ejercicio 4

(a) Incorrecta. No se pueden cumplir ambas partes de la conjunción

(b) Incorrecta. No contempla el caso $a = 0$

(c) Correcta

(d) Correcta

(e) Incorrecta. La implicación junto con la disjunciódin permite que cualquier valor de result haga verdadera la postcondición.

(f) Correcta

3.5. Ejercicio 5

(a) El algoritmo devuelve el valor 9. Hace verdadera la postcondición.

(b) En $x \in \{0, 1\}$ no cumple la postcondición, en el resto sí.

(c) Pre$\{x > 1\}$

3.6. Ejercicio 6

(a) $P3 \to P1 \to P2$

(b) $Q1 \to Q2$ y $Q3 \to Q2$

(c)
   a) $r = x^2 + 1$
   b) $r = x^2 + 2$

(d)
   a) Si
   b) No
   c) Si
   d) No
   e) Si
   f) No
   g) No
   h) No

(e) Se debe cumplir que las precondiciones sean más fuertes y las postcondiciones más débiles.

3.7. Ejercicio 7

1. $(x \neq 0) \to (\neg(n \leq 0) \vee (x \neq 0))$ Por la regla de la implicación.
2. Sí, la postcondición de P1 es verdadera.
3. No pues $P1 \to P2$ pero no viceversa. P1 podría recebir valores $n > 0$, no implementados por a.

3.8. Ejercicio 8

Es cierto que todo algoritmo que cumpla con n-esimo1 también cumple con n-esimo2 pues pre1 $\to$ pre2 y post1 $\to$ post2, pero no al revés.

3.9. Ejercicio 9

(a) `proc` `esPar` (in x: $\mathbb{Z}$, out result: Bool) {
  `Pre` $\{True\}$
  `Post` $\{result = \text{true} \iff (x \text{ mód } 2 = 0)\}$
}

(b) `proc` `esMultiplo` (in n: $\mathbb{Z}$, in m: $\mathbb{Z}$, out result: Bool) {
  `Pre` $\{True\}$
  `Post` $\{result = \text{true} \iff ((\exists k : \mathbb{Z})(n = m.k))\}$
}

(c) `proc` `inverso` (in x: $\mathbb{R}$, out result: $\mathbb{R}$) {
  `Pre` $\{x \neq 0\}$
  `Post` $\{result = \frac{1}{x}\}$
}

(d) `proc` `numericos` (in l: $seq\langle Char\rangle$, out result: $seq\langle Char\rangle$) {
  `Pre` $\{True\}$
  `Post` $\{(\forall i : \mathbb{Z})(0 \leq i < |result| \longrightarrow_L result[i] \in digitos)\}$
}
digitos $= \langle '0', '1', '2', '3', '4', '5', '6', '7', '8', '9' \rangle$

(e) `proc` `duplicaPosicionesImpares` (in l: $seq\langle \mathbb{R}\rangle$, out result: $seq\langle \mathbb{R}\rangle$) {
  `Pre` $\{True\}$
  `Post` $\{(\forall i : \mathbb{Z})(0 \leq i < |l| \longrightarrow_L ((i \text{ mód } 2 = 1 \wedge result[i] = 2.l[i]) \vee (result[i] = l[i])))\}$
}

(f) `proc` `getDivisores` (in x: $\mathbb{Z}$, out result: $\mathbb{Z}$) {
  `Pre` $\{x \neq 0\}$
  `Post` $\{(\forall i : \mathbb{Z})(0 \leq i < |result| \longrightarrow_L ((x \text{ mód } result[i] = 0) \wedge result[i] > 0 \wedge (\forall j : \mathbb{Z})(0 \leq j < |result| \longrightarrow_L result[i] \neq result[j]))))\}$
}

3.10. Ejercicio 10

(a) Sí tiene sentido pues tanto 4 como 0 son números enteros. La respuesta es que 4 NO es múltiplo de 0 pues $\neg \exists k \in \mathbb{Z} : 4 = 0.k$

(b) Debería ser una entrada válida. En la especificación no lo es.

(c) Ver 9.b

(d) La nueva precondición {true} es más debil que $\{m \neq 0\}$

3.11. Ejercicio 11

(a) No

(b) Sí

(c) Ver 9.e

(d) La nueva postcondición es más fuerte que la anterior.

3.12. Ejercicio 12

proc `getBinario` (in x: $\mathbb{Z}$, out result: $seq\langle\mathbb{Z}\rangle$) {

`Pre` $\{x > 0\}$

`Post` $\{x = \sum_{i=0}^{|result|-1} result[i].2^{|result|-i-1}\}$

}

3.13. Ejercicio 13

Sí, en ambos la precondición es demasiado restrictiva. Se está sobreespecificando.

3.14. Ejercicio 14

(a) `proc sumaDeFactoresPrimos` (in x: $\mathbb{Z}$, out res: $\mathbb{Z}$) {

`Pre` $\{x > 0\}$

`Post` $\{res = \sum_{i=2}^{x-1} \text{if } (esPrimo(i) \land x \bmod i = 0) \text{ then } i \text{ else } 0 \text{ fi}\}$

}

(b) `proc esPerfecto` (in x: $\mathbb{Z}$, out res: Bool) {

`Pre` $\{x > 0\}$

`Post` $\{res = \text{true} \iff x = (\sum_{i=1}^{x-1} \text{if } x \bmod i = 0 \text{ then } i \text{ else } 0 \text{ fi})\}$

}

(c) `pred sonCoprimos` (n,m: $\mathbb{Z}$) {

$1 = \sum_{i=1}^{n+m} \text{if } (n \bmod i = 0 \land m \bmod i = 0) \text{ then } 1 \text{ else } 0 \text{ fi}$

}

`proc menorCoprimo` (in n: $\mathbb{Z}$, out m: $\mathbb{Z}$) {

`Pre` $\{n > 0\}$

`Post` $\{m > 1 \land sonCoprimos(n, m) \land (\forall i : \mathbb{Z})(1 \leq i < m \longrightarrow_L \neg sonCoprimos(n, i))\}$

}

(d) `proc descomposicionEnPrimos` (in x: $\mathbb{Z}$, out res: $seq\langle\mathbb{Z} \times \mathbb{Z}\rangle$) {

`Pre` $\{x > 0\}$

`Post` $\{(x = \sum_{i=0}^{|res|-1} res[i]_0^{res[i]_1})$

$\land$

$(\forall i : \mathbb{Z})(0 \leq i < |res| \longrightarrow_L (esPrimo(res[i]_0) \land res[i]_1 \geq 1))$

$\land$

$(\forall j : \mathbb{Z})(0 \leq j < |res| - 1 \longrightarrow_L res[i]_0 < res[i+1]_0)\}$

}

(e) TODO

(f) `aux cantQueDivide` (x: $\mathbb{Z}$, l: $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z}$ $= \sum_{i=0}^{|l|-1} \text{if } l[i] \bmod x = 0 \text{ then } 1 \text{ else } 0 \text{ fi}$;

`proc divideAMasElementos` (in l: $seq\langle\mathbb{Z}\rangle$, out res: $\mathbb{Z}$) {

`Pre` $\{True\}$

`Post` $\{res \in l \land (\forall i : \mathbb{Z})(0 \leq i < |l| \longrightarrow_L cantQueDivide(l[i], l) < cantQueDivide(res, l))\}$

}