

PRÁCTICA 2

1)

- a) $\text{pred esPrimo}(n:\mathbb{Z}) \{ \neg(\exists x:\mathbb{Z})(1 < x < n) \wedge \text{el resto de dividir a } n \text{ por } x \text{ es } 0 \}$
- b) $\text{pred enPosicionValida} (i:\mathbb{Z}, l:\text{seg}(\mathbb{Z})) \{ 0 \leq i < |l| \}$
- c) $\text{esMinimo} (l:\text{seg}(\mathbb{Z}), \text{elem}:\mathbb{Z}) \{ (\forall x:\mathbb{Z})(0 \leq x < |l| \rightarrow x \geq \text{elem}) \wedge \text{elem} \in l \}$
- d) $\text{esMaximo} (l:\text{seg}(\mathbb{Z}), \text{elem}:\mathbb{Z}) \{ (\forall x:\mathbb{Z})(0 \leq x < |l| \rightarrow \text{elem} \geq x) \wedge \text{elem} \in l \}$

2) a) problema $\text{min} (a:\mathbb{Z}, b:\mathbb{Z}) \mathbb{Z} \{$

requiere $\{ a \neq b \}$

asegura $\{ (\text{res} = a \Leftrightarrow a < b) \vee (\text{res} = b \Leftrightarrow b < a) \}$

b) problema $\text{max} (a:\mathbb{Z}, b:\mathbb{Z}) \mathbb{Z} \{$

requiere $\{ a \neq b \}$

asegura $\{ (\text{res} = a \Leftrightarrow a > b) \vee (\text{res} = b \Leftrightarrow b > a) \}$

c) problema $\text{elMayorPrimo} (a:\mathbb{Z}, b:\mathbb{Z}) \mathbb{Z} \{$

requiere $\{ \text{esPrimo}(a) \wedge \text{esPrimo}(b) \}$

asegura $\{ \text{res} = \max(a, b) \}$

d) problema $\text{buscar} (l:\text{seg}(\mathbb{Z}), \text{elem}:\mathbb{Z}) : \mathbb{Z} \{$

requiere $\{ \text{elem} \in l \}$

asegura $\{ 0 \leq \text{res} < |l| \wedge (\text{elem} = l[\text{res}]) \}$

e) problema $\text{buscarMinimo} (l:\text{seg}(\mathbb{Z})) : \mathbb{Z} \{$

requiere $\{ \text{True} \}$

asegura $\{ 0 \leq \text{res} < |l| \}$

asegura $\{ (\forall i:\mathbb{Z})(0 \leq i < |l| \rightarrow l[i] \geq \text{res}) \}$

f) problema #Apariciones ($n: \mathbb{Z}$, $l: \text{seq}(\mathbb{Z})$): \mathbb{Z} {

requiere $\{|l| > 0\}$

asegura $\{ \text{res} = \sum_{i=0}^{|l|-1} (\text{if } l[i] = n \text{ then } 1 \text{ else } 0) \}$

g) problema ordenado crecientemente ($l: \text{seq}(\mathbb{Z})$): \mathbb{B} {

requiere $\{ (\forall x: \mathbb{Z})(0 \leq x < |l| \rightarrow \#Apariciones(s[i], l) = 1) \}$

asegura $\{ (\forall i: \mathbb{Z})(0 \leq i < |l|-1 \rightarrow \text{rec} = (\max(l[i], l[i+1]) = l[i+1])) \}$

requiere $\{|l| > 0\}$

h) problema elMásRepetido ($l: \text{seq}(\mathbb{Z})$): \mathbb{Z} {

requiere $\{|l| > 0\}$

asegura $\{ (\forall x: \mathbb{Z})(x \in l \rightarrow \#Apariciones(l, \text{rec}) \geq \#Apariciones(l, x)) \}$

i) problema borrar ($l: \text{seq}(\mathbb{Z})$, elem: \mathbb{Z}): $\text{seq}(\mathbb{Z})$ {

requiere $\{ (\forall x: \mathbb{Z})(x \in l \rightarrow \#Apariciones(x, l) = 1) \}$

asegura $\{ \text{elem} \in l \rightarrow \text{rec} = \text{Concat}(\text{Subseq}(l, 0, \text{busr(elem, l)}),$
 $\text{tail}(\text{Subseq}(l, \text{busr(elem, l)}, |l|)) \}$

3) a) faltaría reponer la siguiente, el asegura debería ser:

asegura $\{ (\forall x: \mathbb{Z})(0 \leq x < |l| \wedge l[x] = \text{elem} \rightarrow \text{res} = x) \}$

b) la variable x está de más, faltó un require

requiere $\{|l| > 0\}$

asegura $\{ (\forall y: \mathbb{Z})(y \in l \rightarrow y > \text{result}) \}$

c) el rango del cuantificador está mal:

$$\text{asegure}\{ \text{res} = \text{True} \rightarrow ((\forall i : \mathbb{Z})(0 \leq i < |l| \rightarrow l[i] = 2 * l[i-1])$$

- 4) la diferencia entre la especificación semiformal y la formal es que la primera tiende a ser más ambigua que la segunda, ya que al especificar semiformalmente lo hacemos con oraciones lo cual podría ser más fácil de leer pero puede tener ciertas ambigüedades. Por otro lado la especificación formal al usarse un lenguaje matemático (\sim lógico) elimina ambigüedades, haciendo más "rigurose" la especificación.

- 5) si $s = \langle 1, 2, 3 \rangle$ y $\text{sum}_s = 10$, cumple con el requiere, pero el asegura nunca se cumple ya que es imposible sumar 10 con los números 1, 2 y 3.

- b) si tomamos $s = \langle -4, 4 \rangle$ y $\text{sum}_s = 1$, $\text{min_sum}(s) = -4$ y $\text{max_sum}(s) = 4$, cumpliría con el requiere ya que $-4 \leq 1 \leq 4$, pero no existe un result, ya que la suma de algunos de sus elementos no de 1

- c) requiere { Existe una secuencia r la cual la suma de todos sus elementos es sum_s }
requiere { todos elementos de r pertenece a l }

6) a) I) 0 II) 1 III) $\sqrt{27}$

c) I) $\text{res} = 3$

II) $\text{res} = 0 \circ 3$

III) $\text{res} = 0 \circ 1 \circ 2 \circ 3 \circ 4 \circ 5$

c) I) $\text{res} = 3$ II) $\text{res} = 0$ III) $\text{res} = 0$

6) Cuando el máximo de la secuencia l no se repite mas de una vez.

7) a) no es correcta ya que el asegurado siempre devolverá falso, porque a no puede ser $a < 0$ y el mismo tiempo tener $a \geq 0$

b) no es correcta ya que el asegurado no contempla el caso $a=0$

c) está bien

d) está bien

e) está mal ya que tanto $(a < 0 \rightarrow \text{res} = 2 * b)$ y $(a \geq 0 \rightarrow \text{res} = b - 1)$ tienen valor de verdad verdadero, entonces el res del asegurado podría devolver $\text{res} = 2 * b$ o $\text{res} = b - 1$ sin importar el valor de a.

f) está bien

8) el algoritmo devuelve 9, resultado que cumple con la especificación.

9)

X	Algoritmo
0,5	0,125
1	1
0,2	0,04
-1	49

} estos 3 valores no cumplen con la especificación.

c) requiere $\{ x > 1 \vee x < 0 \}$

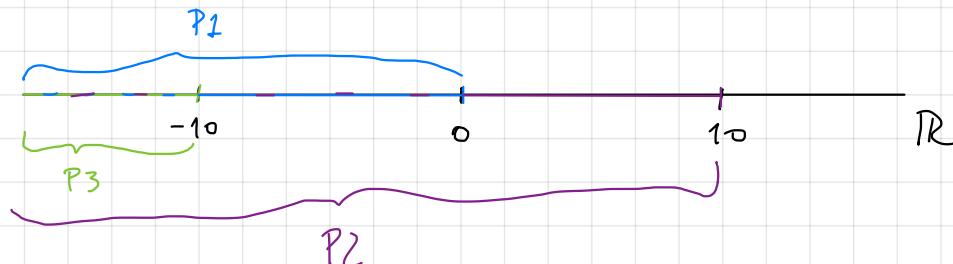
01) a) recordar:
 se dice que A es más fuerte que B si
 $A \rightarrow B$ es una tautología

$$P_1: \{x \leq 0\}$$

$$P_2: \{x \leq 10\}$$

$$P_3: \{x \leq 10\}$$

Gráficamente podríamos representarlos

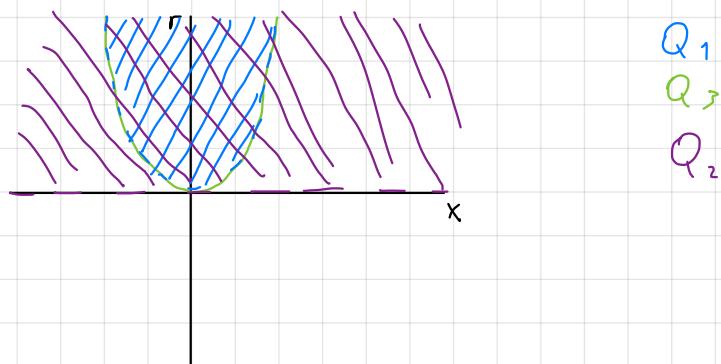


Cómo se ve representado gráficamente, si se cumple P_3 , $P_1 \rightarrow P_2$ son verdaderos también.

Por otro lado, si se cumple P_1 y no se cumple P_3 , se cumple P_2

podemos decir que P_3 es más fuerte que $P_1 \rightarrow P_2$ y además P_1 es más fuerte que P_2 ya que $P_3 \rightarrow P_2 \rightarrow P_1$

b) $Q_1: \{r \geq x^2\}$ $Q_2: \{r \geq 0\}$ $Q_3: r = x^2$



Cómo se puede ver gráficamente si se cumple Q_2 entonces se cumplen Q_1 y Q_3

Por otros lados si se cumple Q_1 se cumple Q_2

la relación de fuerza sería: $Q_2 \rightarrow Q_1 \rightarrow Q_3$

c) en Haskell:

hagoAlgo :: float → float

hagoAlgo n = $n^2 + 1$

en Python:

```
def hagoAlgo(x: float) → float:  
    return  $x^2 + 1$ 
```

- | | | | |
|----|-------|-------|-------|
| d) | a) sí | e) sí | h) No |
| | b) sí | f) sí | |
| | c) sí | g) No | |
| | d) No | | |

e)

10) a) si Dados valores de $x > n$ cumplen P_1
entonces $x \neq 0$ e) verdadero

Analizando el requiere de P_2 $\{ n \leq 0 \rightarrow x \neq 0 \}$
Sabemos que $x \neq 0$ es verdadero veamos que
para sí $n \leq 0$ es verdadero o falso

$n < 0$	$x \neq 0$	$x \leq 0 \rightarrow x \neq 0$
V	V	V
F	V	V

Como se ve en la tabla, el requiere de P_2 es
verdadero sin importar si $n \leq 0$ es falso
o verdadero. Por lo tanto se puede concluir que
si se cumple la precondition P_1 , se
cumple también la precondition de P_2

c) la post condición de P_1 es

$$\underbrace{x^n - 1}_{\mathbb{R}} \leq \underbrace{\text{res}}_{\mathbb{Z}} \leq \underbrace{x^n}_{\mathbb{R}}$$

lo que hace es redondear hacia abajo x^n

por otro lado sabemos que la post condición de P_2 se cumple la cual es la siguiente

$$\text{res} = \lfloor x^n \rfloor$$

lo que hace es aplicarle la función floor a x^n (redondear hacia abajo)

Como las dos postcondiciones hacen lo mismo, se puede decir que si se cumple la post condición de P_2 , se cumple la postcondición de P_1 .

c) Sí

11) nota: Asumo que no hay repetidos en l

¿ Todo algoritmo que cumple n-esimo 1 cumple con n-esimo 2?

Análizo primero si un l y n que hacen verdadera la precondition de n-esimo 1, tambien hacen verdadera la precondition de n-esimo 2.

Precondiciones n-esimo 1

requiere { la longitud de la secuencia sea mayor a 0 }

requiere { n es mayor o igual a 0 y menor que la longitud del }

requiere { los elementos estén ordenados crecientemente }

Precondiciones n-esimo 2

requiere { la longitud de la secuencia sea mayor a 0 }

requiere { n es mayor o igual a 0 y menor que la longitud del }

requiere { los elementos son distintos entre si }

los dos primeros requiere son iguales, por lo tanto
son verdaderos en n-ésimo 2.

por otro lado si tengo una secuencia l que esté
ordenada crecientemente y no tiene repetidos cumple con la condición de
que todos los elementos de l sean distintos

Por lo tanto se puede concluir que si un l
cumple la precondition de n-ésimo 1, también cumple
la precondition de n-ésimo 2

Ahora analizo si el resultado que nos da el algoritmo
que cumple con la especificación de n-ésimo 1,
hace verdadera la postcondición de n-ésimo 2

el algoritmo nos da: $\text{res} = l[n]$

la postcondición de n-ésimo 2 es:

La cantidad de elementos menores a res es igual a n
esto es, lo mismo que $l[n]$ ya que al estar
ordenada la secuencia, el único elemento que tiene n elementos
menores que él es $l[n]$. por lo tanto el
resultado del algoritmo que cumple n-ésimo 1
hace verdadera la precondition n-ésimo 2.

En conclusión todo algoritmo que cumple n-ésimo 1
cumple la especificación de n-ésimo 2.

c) Es cierto que todo algoritmo que cumple con n-ésimo 2
cumple también con n-ésimo 1?

• Si se cumple el require de n-ésimo 2 se cumple
el de n-ésimo 1? No, ya que si l tiene
todos elementos distintos no garantiza que estén
todos ordenados, por ejemplo $l = \langle 1, 4, 3, 2 \rangle$ cumple
con la precondition de n-ésimo 2 pero no con la
de n-ésimo 1.

Se puede concluir que al no cumplir con la precondición el Algoritmo cumple con Número 2 no garantiza que cumpla para Número 1.

12) a)

problema factoresPrimos ($n: \mathbb{Z}$) seq($\mathbb{Z} \times \mathbb{Z}$) {

requiere { $n \geq 2$ }

asegura { esDescomposicion(res, n) }

asegura { primoEnRes(res) }

asegura { ordenadoPorP(res) }

}

pred esDescomposicion (res: seq $\mathbb{Z} \times \mathbb{Z}$, $n: \mathbb{Z}$) {

$$n = \prod_{i=0}^{|res|-1} res_i \quad \left. \begin{array}{l} res_0 \\ res_1 \\ \vdots \end{array} \right\}$$

pred primosEnRes (res: seq $\mathbb{Z} \times \mathbb{Z}$) {

($\forall x: \mathbb{Z}$) ($0 \leq x < |res| \rightarrow \text{esPrimo}(res_x)$) }

pred OrdenadoPorP (res: seq $\mathbb{Z} \times \mathbb{Z}$) {

($\forall x: \mathbb{Z}$) ($0 \leq x < |res|-1 \rightarrow res_0[x] < res_0[x+1]$) }

b) problema diferenciaMaxima ($l: \text{seq}(\mathbb{Z})$): \mathbb{Z} {

requiere { $|l| > 1$ }

asegura { $(\forall i, j: \mathbb{Z})(0 \leq i, j < |l| \wedge i \neq j \rightarrow res \geq \text{diferencia}(i, j))$

}

problema diferencia ($a: \mathbb{Z}, b: \mathbb{Z}$): \mathbb{Z} {

requiere { true }

asegura { if $a \geq b$ then $res = a - b$ else $res = b - a$ }

}

13)

a) problema esSubcadena ($l : \text{seq}(\mathbb{Z})$, $s : \text{seq}(\mathbb{Z})$) : B {
requiere { True } }

asegura { $(\exists x, i : \mathbb{Z})(0 \leq x, i < |l| \wedge x \leq i \rightarrow s = \text{subcadena}(l, x, i))$ }
}

problema subcadena ($l : \text{seq}(\mathbb{Z})$, desde : \mathbb{Z} , hasta : \mathbb{Z}) : $\text{seq}(\mathbb{Z})$ {
requiere { True } }

asegura { res = subseq $(l, \text{desde}, \text{hasta})$ }

b) problema estaIncluida ($t : \text{seq}(\mathbb{Z})$, $s : \text{seq}(\mathbb{Z})$) : IB {
requiere { $|t| > 0 \wedge |s| > 0$ } }

asegura { $(\forall x : \mathbb{Z})(x \in s \leftrightarrow \#\text{apariciones}(s, x) \leq \#\text{apariciones}(t, x))$ }

c) problema mezclarOrdenado ($s, t : \text{Seq}(\mathbb{Z})$) : $\text{Seq}(\mathbb{Z})$ {

requiere { ordenado (crecientemente) (s) }

requiere { ordenado (crecientemente) (t) }

asegura { ordenado (res) }

asegura { estaIncluida (res, concat(s, t)) \wedge estaIncluida (concat(s, t), res) }
}

pred ordenado ($l : \text{seq}(\mathbb{Z})$) {

$|l| > 0 \wedge (\forall x : \mathbb{Z})(0 < x < |l| \rightarrow l[i] \geq l[i-1])$

d) problema multiplicar pares por n ($l: \text{seq}(\mathbb{Z})$, $n: \mathbb{Z}$): $\text{seq}(\mathbb{Z})$ {

requiere { True }

asegura { $|res| = |l|$ }

asegura { $(\forall x: \mathbb{Z})(0 \leq x < |l| \wedge \text{Par}(x) \rightarrow res[x] = l[x] * n)$ }

asegura { $(\forall x: \mathbb{Z})(0 \leq x < |l| \wedge \neg \text{Par}(x) \rightarrow res[x] = l[x])$ }

e) problema borrarMultiplosDe3 ($l: \text{seq}(\mathbb{Z})$): $\text{seq}(\mathbb{Z})$ {

requiere { True }

asegura { $(\forall i: \mathbb{Z})(0 \leq i < |l| \wedge \neg(x \bmod 3 = 0) \rightarrow res[i] = l[i - \text{contmult3}(\text{subseq}(l, 0, i))])$ }

problema contmult3 ($s: \text{seq}(\mathbb{Z})$): \mathbb{Z} {

requiere { True }

asegura { $res = \sum_{i=0}^{|s|-1} (\text{if } (i \bmod 3 = 0) \text{ then 1 else 0 s:})$ }