El $\forall$ mira todos al mismo tiempo

-> HECHO/BIEN
-> HECHO/DUDA
-> NO HICE/MAL

2.1. Funciones auxiliares

Ejercicio 1. ★ Escriba los siguientes predicados sobre números enteros en lenguaje de especificación:

a) *pred esCuadrado* $(x : \mathbb{Z})$ que sea verdadero si y sólo si x es un numero cuadrado.

b) *pred esPrimo* $(x : \mathbb{Z})$ que sea verdadero si y sólo si x es primo.

c) *pred sonCoprimos* $(x, y : \mathbb{Z})$ que sea verdadero si y sólo si x e y son coprimos.

d) *pred mayorPrimoQueDivide* $(x : \mathbb{Z}, y : \mathbb{Z})$ que sea verdadero si y es el mayor primo que divide a x.

PREDICADO: DEVUELVE TRUE O FALSE.
NO DEVUELVE TIPO DE DATO

a) PRED ESCUADRADO$(x:\mathbb{Z})\{$

$$(\exists i:\mathbb{Z})(x = i^2)$$

$\}$

b) PRED esPrimo$(x:\mathbb{Z})\{$

$$x > 1 \land (\forall y:\mathbb{Z})(\overbrace{2 \leq y < x}^{\text{AUNQUE ESTO LABURO CON TODO } \mathbb{Z}} \rightarrow_L \neg(x \bmod y = 0))$$ ✓

$2 \leq y < 0$ necesito cortar la ejecución

$\}$

c) PRED SONCOPRIMOS$(x, y:\mathbb{Z})\{$

$$\neg(\exists a:\mathbb{Z})(a > 1 \land (x \bmod (a) = 0 \land y \bmod (a) = 0))$$ ✓

}

$(10:3) = (2.5:3) = 1$

PRED MAYORPRIMOQUEDIVIDE$(x:\mathbb{Z}, y:\mathbb{Z})\{$
$(esPrimo(y) \wedge x \bmod y = 0) \wedge$
$\neg(\exists i)(esPrimo(i) \wedge x \bmod i = 0 \wedge i > y)$
}

Ej: $x: 10 \quad y: 2 \Rightarrow$ FALSO $\quad i = 5$

$(V \wedge V) \wedge (V \wedge V \wedge V) = V \wedge \neg V = F$

$x: 10 \quad y: 5 \Rightarrow$ VERDADERO $\quad i = 2$

$(V \wedge V) \wedge (V \wedge V \wedge F) = V \wedge \neg(F) = V$

Ejercicio 2. ★ Escriba los siguientes predicados auxiliares sobre secuencias de enteros, aclarando los tipos de los parámetros que recibe:

a) *esPrefijo*, que determina si una secuencia es prefijo de otra.

b) *estáOrdenada*, que determina si la secuencia está ordenada de menor a mayor.

c) *hayUnoParQueDivideAlResto*, que determina si hay un elemento par en la secuencia que divide a todos los otros elementos de la secuencia.

d) *sinRepetidos*, que determina si la secuencia no tiene repetidos.

e) *enTresPartes*, que determina si en la secuencia aparecen (de izquierda a derecha) primero 0s, después 1s y por último 2s. Por ejemplo $\langle 0,0,1,1,1,1,2\rangle$ cumple con *enTresPartes*, pero $\langle 0,1,3,0\rangle$ o $\langle 0,0,0,1,1\rangle$ no. ¿Cómo modificaría la expresión para que se admitan cero apariciones de 0s, 1s y 2s (es decir, para que por ejemplo $\langle 0,0,0,1,1\rangle$ o $\langle\rangle$ sí cumplan *enTresPartes*)?

a) PRED ESPREFIJO$(x: seq\langle\mathbb{Z}\rangle, y: seq\langle\mathbb{Z}\rangle)\{$

$(\forall i : \mathbb{Z})(0 \le i < |X| \to_L (i < |Y| \land_L (X[i] = Y[i]))$ ✓

}

PARA QUE i NO SE PASE SI: $|Y| > |X|$.

Ej: $X: [1,2,3]$, $Y: [1,2]$ V, $X: [1,2,3]$, $Y: []$ F

b) PRED ESTAORDENADA (X : seq<$\mathbb{Z}$>) {

$(\forall i : \mathbb{Z})(0 \le i < (|X|-1) \to_L (X[i+1] > X[i]))$ ✓

}

Ej: $[1, 2, 3, 5, 4] = F$

Ej: $[1] = 0 \le i < -1 = F$

Ej: $[3, 1] : 0 \le i < 1 = 1 > 3 = F$

Ej: $[1, 2, 3] = 0 \le i < 3 = \underset{X[1]\ X[0]}{2 > 1} \land \underset{X[2] > X[1]}{3 > 2} = V$

C) PRED HAYUNOPARQUEDIVIDEALRESTO (X : seq<$\mathbb{Z}$>) {

$(\exists i : \mathbb{Z})$

$(0 \le i < |X| \land_L X[i] \bmod 2 = 0 \land_L (\forall j : \mathbb{Z})(0 \le j < |X| \to_L X[j] \bmod X[i] = 0))$ ✓

}

Ej: $X = [4, 2, 6, 8] = i = 0, X[0] = 4, V \land \begin{cases} V \\ F \end{cases} = F$

D) PRED SINREPETIDOS (X : seq<$\mathbb{Z}$>) {

$(\forall i:\mathbb{Z})(0 \le i < |x| \to_L$

$((\exists j:\mathbb{Z})((0 \le j < |x| \land_L x[j] = x[i]) \to i = j))$

$\}$

✓

Ejercicio 3. Sea s una secuencia de elementos de tipo $\mathbb{Z}$. Escribir una expresión (utilizando sumatoria y productoria) tal que:

a) Cuente la cantidad de veces que aparece el elemento e de tipo $\mathbb{Z}$ en la secuencia s.

b) Sume los elementos en las posiciones impares de la secuencia s.

c) Sume los elementos mayores a 0 contenidos en la secuencia s.

d) Sume los inversos multiplicativos ($\frac{1}{x}$) de los elementos contenidos en la secuencia s distintos a 0.

a) $\sum_{i=0}^{|s|-1} \text{if } (s[i] = e) \text{ THEN } 1 \text{ ELSE } 0$

b) $\sum_{i=2i+1}^{|s|-1} s[i]$ $\quad\Big|\quad$ $\sum_{i=1}^{|s|} \text{if } i \bmod 2 = 0 \text{ THEN } 0 \text{ ELSE } s[i]$ ✓

↳ $s[i]$ no se puede indefinir?

c) $\sum_{i=0}^{|s|-1} \text{if } (s[i] > 0) \text{ THEN } s[i] \text{ ELSE } 0$

↳ $s[i]$ NO se puede INDEFINIR?

d) $\sum_{i=0}^{|s|-1} \text{if}(s[i] > 0) \text{ THEN } \frac{1}{s[i]} \text{ ELSE } 0$

$\sum_{i=0}^{|s|-1} (s[i] > 0, \frac{1}{s[i]}, 0)$ $\Rightarrow$ ¿PUEDO ASÍ?

2.2. Análisis de especificación

Ejercicio 4. Las siguientes especificaciones no son correctas. Indicar por qué y corregirlas para que describan correctamente el problema.

a) `progresionGeometricaFactor2`: Indica si la secuencia l representa una progresión geométrica factor 2. Es decir, si cada elemento de la secuencia es el doble del elemento anterior.

proc progresionGeometricaFactor2 (in l: $seq\langle\mathbb{Z}\rangle$) : Bool

requiere $\{True\}$

$1 \leq i < 1 \rightarrow_L$

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

$1 \leq i < |l|$

b) `minimo`: Devuelve en res el menor elemento de l. → $res \in l$

proc minimo (in l: $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z}$

requiere $\{True\}$

¿x?

asegura $\{(\forall y : \mathbb{Z})((y \in l \wedge \cancel{y \neq x}) \rightarrow y \geq res)\}$

$y \geq res \wedge res \in l$

1

a) $l : [\,] = 0 \leq i < 0$ F = F

$l : [1] = 0 \leq i < 1 \Rightarrow l[0] = 2 * \underbrace{l[-1]}$

OUT OF BOUNDS. !

El problema es q cuando $|l| > 0$, siempre buscará $l[0] = 2 \cdot l[-1]$ y ¡NO ∃ índice neg!

Una solución es que para validar esto la lista tenga más de 1 elemento o hacer algún iF o empezar a contar desde 1.

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

b) $(\forall y : \mathbb{Z})((y \in l \wedge y \in res) \rightarrow res < y)$

¿Cuándo ocurre esto?

E [

Ej: [1,2,3,4] res: 1

EN 2024 CAMBIÓ UN POCO LA GUIA

2.1. Funciones auxiliares

Ejercicio 1. ★ Nombrar los siguientes predicados sobre enteros:

a) pred ???? (x: $\mathbb{Z}$) {
$(\exists c : \mathbb{Z})(c > 0 \wedge (c * c = x))$
}

b) pred ???? (x: $\mathbb{Z}$) {
$x>1 \wedge (\forall n : \mathbb{Z})((1 < n < x) \to_L (x \mod n \neq 0))$
}

→ MOD SE INDEFINE CON 0

⇒ LO HICE BIEN EN CUATRI PASADO

a) CUADRADO ⇒ se busca un num q multiplicado de x. ✓

Ej: $x=4$, la res sería ser $c=2$ pues $2*2=4$

b) ESPRIMO: VALIDA SI DADO UN X, ∃ MÍN que sean >1 y $<M$. ✓

Ej: $x=5 \Rightarrow 1<m<5 = 5 \bmod 2 \neq 0$
$5 \bmod 3 \neq 0$
$5 \bmod 4 \neq 0$ ✓

Ejercicio 2. ★ Escriba los siguientes predicados sobre números enteros en lenguaje de especificación:

a) pred *sonCoprimos* $(x, y : \mathbb{Z})$ que sea verdadero si y sólo si x e y son coprimos.

b) pred *mayorPrimoQueDivide* $(x : \mathbb{Z}, y : \mathbb{Z})$ que sea verdadero si y es el mayor primo que divide a x.

TOP DOWN: MODULARIZACIÓN

a) PRED SONCOPRIMOS $(X, Y : \mathbb{Z})$ {

$$\neg(\exists i : \mathbb{Z})\ (i > 1 \wedge_L (X \bmod i = 0 \wedge Y \bmod i = 0))$$ ✓

}

b) Necesito ASEGURAR Q NO EXISTE OTRO PRIMO Q DIVIDA A X MAS Q Y.

Osea puedo decir que si ∃ uno entonces es menor que Y

PRED MAYORPRIMOQUEDIVIDE $(X : \mathbb{Z}, Y : \mathbb{Z})$ {

$$\star\ (\forall i : \mathbb{Z})\ ((esPrimo(i) \wedge x \bmod i = 0) \to (i \leq Y))$$

}

si no pongo $i \leq Y$ sino $i < Y$ y $i \neq Y$.

¿Está bien? OMITI VER Q Y ES PRIMO, Y DIVIDE A X PORQUE ESO NO SERIA UN LABURO DE "PRECONDICIÓN"? => SÍ PERO LAS PRECOND ∄ EN PREDS

OP 2: CONSIDERARIA Q Y ES PRIMO E Y DIVIDE A X

$$(\forall i : \mathbb{Z})\ ((esPrimo(i) \wedge x \bmod i = 0) \wedge (esPrimo(Y) \wedge x \bmod Y = 0) \to (i \leq Y))$$ ✓

Ejercicio 3. ★ Nombre los siguientes predicados auxiliares sobre secuencias de enteros:

a) **pred** ???? (s: $seq\langle\mathbb{Z}\rangle$) {

$(\forall i : \mathbb{Z})((0 \leq i < |s|) \to_L s[i] \geq 0)$

}

si $i \neq j \to s[i] \neq s[j]$

b) **pred** ???? (s: $seq\langle\mathbb{Z}\rangle$) {

$(\forall i : \mathbb{Z})((0 \leq i < |s|) \to_L (\forall j : \mathbb{Z})((0 \leq j < |s| \wedge i \neq j) \to_L (s[i] \neq s[j])))$

}

a) SON TODOS POSITIVOS: se chequea
q en la lista haya nros positivos,
con el 0 incluido

b) SIN REPETIDOS: se compara q un valor
de i y j si son distintos entonces no
tengan el mismo valor.

Ej: x: [1, 2, 2]
$i = 1$ $j = 2$ ⇒ acá no cumple
xq $i \neq j$ pero $s[i] = s[j]$

4) En el 2 de la guía del cuatri
pasado

5) " " 3 " " " " "
"

6) " " 4 " " " " "
"

Ejercicio 7. Para los siguientes problemas, dar todas las soluciones posibles a las entradas dadas:

a) proc indiceDelMaximo (in l: $seq\langle\mathbb{R}\rangle$) : $\mathbb{Z}$

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

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

I) $l = \langle 1, 2, 3, 4 \rangle$

II) $l = \langle 15.5, -18, 4.215, 15.5, -1 \rangle$

III) $l = \langle 0, 0, 0, 0, 0, 0 \rangle$

b) proc indiceDelPrimerMaximo (in l:$seq\langle\mathbb{R}\rangle$) : $\mathbb{Z}$

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

asegura $\{0 \leq res < |l| \wedge_L ((\forall i : \mathbb{Z})(0 \leq i < |l| \rightarrow_L (l[i] < l[res] \vee (l[i] = l[res] \wedge i \geq res))))\}$

I) $l = \langle 1, 2, 3, 4 \rangle$

II) $l = \langle 15.5, -18, 4.215, 15.5, -1 \rangle$

III) $l = \langle 0, 0, 0, 0, 0, 0 \rangle$

c) ¿Para qué valores de entrada indiceDelPrimerMaximo y indiceDelMaximo tienen necesariamente la misma salida?

a) i) res es un indice entre 0 y 3.

Salida: 3

ii) 0 o 3. No se especifica cual funciona.

iii) 0 a 6. " " " " "

b) i) 3

b) 0

c) 0

c) $l = \langle 1,2,3,4 \rangle$

los otros dos casos al tener repetidos el a podría no coincidir con el b

Ejercicio 8. Sea $f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ definida como:

$$f(a, b) = \begin{cases} 2b & \text{si } a < 0 \\ \ldots \end{cases}$$

$\quad\quad b-1 \quad$ en otro caso

Indicar cuáles de las siguientes especificaciones son correctas para el problema de calcular $f(a,b)$. Para aquellas que no lo son, indicar por qué.

a) **proc f** (in a, b: $\mathbb{R}$) : $\mathbb{R}$
   requiere $\{True\}$
   asegura $\{(a<0 \land res = 2*b) \land (a \geq 0 \land res = b-1)\}$ ✗

b) **proc f** (in a, b: $\mathbb{R}$) : $\mathbb{R}$
   requiere $\{True\}$
   asegura $\{(a<0 \land res = 2*b) \lor (a \geq 0 \land res = b-1)\}$ ✓

c) **proc f** (in a, b: $\mathbb{R}$) : $\mathbb{R}$
   requiere $\{True\}$
   asegura $\{(a<0 \rightarrow res = 2*b) \lor (a \geq 0 \rightarrow res = b-1)\}$ ✗

d) **proc f** (in a, b: $\mathbb{R}$) : $\mathbb{R}$
   requiere $\{True\}$
   asegura $\{res = (\text{if } a<0 \text{ then } 2*b \text{ else } b-1 \text{ fi})\}$ ✓

a) a no puede ser dos valores distintos tiempo
$a = -1$
$V \land F = F$ ✓

c) $a = -1$
$(V \lor (F \rightarrow V)) = (V \lor V) = V$ ✓

NO PUEDEN SER AMBAS V $\Rightarrow$ ESTA ES LA JUST?

Ejercicio 9. Considerar la siguiente especificación, junto con un algoritmo que dado x devuelve x^2.

proc unoMasGrande (in x: $\mathbb{R}$) : $\mathbb{R}$
   requiere $\{True\}$
   asegura $\{res > x\}$

a) ¿Qué devuelve el algoritmo si recibe $x = 3$? ¿El resultado hace verdadera la postcondición de unoMasGrande?

b) ¿Qué sucede para las entradas $x = 0{,}5$, $x = 1$, $x = -0{,}2$ y $x = -7$?

c) Teniendo en cuenta lo respondido en los puntos anteriores, escribir una **precondición** para unoMasGrande, de manera tal que el algoritmo cumpla con la especificación

a) 9 xq $x^2 = 3^2 = 9$. Si xq $9 > 3$

b) solo se cumple en $x = 1$ y $x = -7$.

c) requiere $\{x \leq -1 \vee x \geq 1\}$

2.3. Relación de fuerza

Ejercicio 10. Sean x y r variables de tipo $\mathbb{R}$. Considerar los siguientes predicados:

P1: $\{x \leq 0\}$ 2
P2: $\{x \leq 10\}$ 3
P3: $\{x \leq -10\}$ 1

Q1: $\{r \geq x^2\}$ 1, 4, 9
Q2: $\{r \geq 0\}$ 1, 2, 3, 4
Q3: $\{r = x^2\}$ => $2 = 2^2 = 4$

a) Indicar la relación de fuerza entre P1, P2 y P3

$Q_3 > Q_1 > Q_2$

b) Indicar la relación de fuerza enrte Q1, Q2 y Q3

c) Escribir 2 programas que cumplan con la siguiente especificación:

```
proc hagoAlgo (in x: ℝ) : ℝ
    requiere {x ≤ 0}
    asegura {res ≥ x²}
```

d) Sea A un algoritmo que cumple con la especificación del ítem anterior. Decidir si necesariamente cumple las siguientes especificaciones:

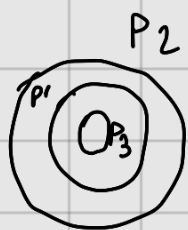
a) **requiere** $\{x \leq -10\}$, **asegura** $\{r \geq x^2\}$ ✓
b) **requiere** $\{x \leq 10\}$, **asegura** $\{r \geq x^2\}$ X
c) **requiere** $\{x \leq 0\}$, **asegura** $\{r \geq 0\}$ ✓
d) **requiere** $\{x \leq 0\}$, **asegura** $\{r = x^2\}$
e) **requiere** $\{x \leq -10\}$, **asegura** $\{r \geq 0\}$
f) **requiere** $\{x \leq 10\}$, **asegura** $\{r = x^2\}$

leyes ? ver si meto la hagoAlgo fuerza en lista

e) ¿Qué conclusión pueden sacar? ¿Qué debe cumplirse con respecto a las precondiciones y postcondiciones para que sea seguro **reemplazar la especificación**?

Decimos que es más fuerte quien nos da más info (→ ÚTIL) o es más restrictivo

a) P3 > P1 > P2 PODEMOS VERLO ASÍ ✓



b) Q > Q > Q ✓

b) $Q_3 > Q_1 > Q_2$

c) PROC A(x: $\mathbb{R}$){

REQUIERE: {$x \leq 0$}

ASEGURA: {res $= x^3$}

}

PROC B(x: $\mathbb{R}$){

REQUIERE: {$x \leq 0$}

ASEGURA: {res $= x^4$}

}

REVI

d) A CUMPLE $x \leq 0 \Rightarrow res \geq x^2$

e) Modificar la PRECONDICION es peligroso. Si no se quiere romper el funcionamiento se debería acotar con cosas q antes valían.

Ej: si $x > 2$ valía y ahora $x > 0$ es mejor y no dañé a nadie donde $x > 2$. Solo agg dos valores.

Con respecto al asegura, debemos [illegible]

V o lintione

Ej: si $res \geq x^2$ es mejor que $res = x^2$.

y podría tener problema pero si pongo que $res = x^2$ o $res \geq x^2$ es mejor.

REVI

Ejercicio 11. Considerar las siguientes dos especificaciones, junto con un algoritmo a que satisface la especificación de p2.

proc p1 (in x: $\mathbb{R}$, in n: $\mathbb{Z}$) : $\mathbb{Z}$
    requiere $\{x \neq 0\}$ ⟶ MAS FUERTE que el REQ de P2. P1 es MAS RESTRICTIVO.
    asegura $\{x^n - 1 < res \leq x^n\}$

P2 PUEDE SER V MÁS FÁCIL

proc p2 (in x: $\mathbb{R}$, in n: $\mathbb{Z}$) : $\mathbb{Z}$
    requiere $\{n \leq 0 \rightarrow x \neq 0\}$
    asegura $\{res = \lfloor x^n \rfloor\}$

TABLA DE VERDAD.

a) Dados valores de x y n que hacen verdadera la precondición de p1, demostrar que hacen también verdadera la precondición de p2.

b) Ahora, dados estos valores de x y n, supongamos que se ejecuta a: llegamos a un valor de res que hace verdadera la postcondición de p2. ¿Será también verdadera la postcondición de p1 con este valor de res?

c) ¿Podemos concluir que a satisface la especificación de p1?

$X = 1 \quad M = 2$

a)

$P_1: \ x \neq 0$ ✓

$P_2: \ m \leq 0 \rightarrow x \neq 0$ ✓

b) $P_2: \ 1^2 = 1$ ← res

$P_1: \ 1^2 - 1 < 1 \leq 1^2 = 0 < 1 \leq 1$ ✓ (1 ← res)

c) sí, xq si tuviés hacer F en P1, en P2 sería F

m = -1 x = 0 => V -> F = F

pero en P1 es F directo.

VA A PASAR

REQUIERE DEBIL -> REQUIERE FUERTE

ASEGURA ~> ASEGURA

2.4. Especificación de problemas

Ejercicio 12. Especificar los siguientes problemas:

a) Dado un entero, decidir si es par

b) Dado un entero n y otro m, decidir si n es un múltiplo de m

c) Dado un entero, listar todos sus divisores positivos (sin duplicados)

d) Dado un entero positivo, obtener su descomposición en factores primos. Devolver una secuencia de tuplas (p, e), donde p es un factor primo y e es su exponente, ordenada en forma creciente con respecto a p

12) a)

PROC ESPAR(x: ℤ): Bool {

REQUIERE: {true}

ASEGURA: {res = true <=> x MOD 2 = 0}

}

b) 4 2 => 4 es MULT xq 4 MOD 2 = 0 => T

10 5 => 10 es MULT xq 10 MOD 5 = 0 => T

-1 0 => NO CUMPLE REQ

-1 2 => F

1 1 => T

PROC ESMULTIPLO(n: ℤ, m: ℤ): Bool {

REQUIERE: $\{ m > 1 \}$

ASEGURA: $\{ res = True \iff n \bmod m = 0 \}$

}

c) PROC DivisoresPos(x: $\mathbb{Z}$): seq<$\mathbb{Z}$> {

REQUIERE: {True}

ASEGURA: {ListaNoNula(res)}

ASEGURA: {DivisoresEnRes(x)}

ASEGURA: {SinRepetidos(res)}

ASEGURA: {TodosDividen(res, x)}

ASEGURA: {TodosPositivos(res)}

}

DEBO VALIDAR que si i DIVIDE a x

entonces i está en res

Y los elementos de res DIVIDEN a x.

Es algo BIDIRECCIONAL

1) La lista es VACÍA o no

2) Si un valor divide a x, estará en res.

3) Sin repetidos

4) Todos los valores de res dividen a X

5) Todos los valores de res son positivos

PRED LISTANONULA.(x: seq <ℤ>){|res| ≥ 0}

PRED DIVISORESENRES(x: ℤ){

$(\forall i:\mathbb{Z})((0 \le i < x \wedge x \bmod i = 0) \to i \in res)$

}

PRED SINREPETIDOS(res: seq <ℤ>){

$(\forall i:\mathbb{Z})((0 \le i < |res| \to_L (\forall j:\mathbb{Z})((0 \le j < |res| \wedge i \ne j) \to_L (res[i] \ne res[j])))$ }

}

PRED TODOSDIVIDEN(res: seq <ℤ>, x: ℤ){

$(\forall i:\mathbb{Z})(0 \le i < res \to x \bmod res = 0)$

}

pred todosPositivos(x: seq<$\mathbb{Z}$>){

$(\forall i:\mathbb{Z})(0 \leq i < |res| \rightarrow_L res[i] > 0)$

}

d)

proc descomposicionPrimos(x: $\mathbb{Z}$): seq<$\mathbb{Z}$, $\mathbb{Z}$> {

requiere: $\{x \geq 2\}$

asegura: {enPrimos(x)}

asegura: {ordenadaCreciente(res)}

asegura: {x = multFactores(res)}

asegura: {sinFactoresRepetidos(res)}

asegura: {todosDividen(res, x)}

ASEGURA: {TodosPrimosEnRes(res)}

ASEGURA: {TodosPositivos(res)}

}

AUX EnPrimos(x: $\mathbb{Z}$): seq<$\mathbb{Z}$, $\mathbb{Z}$> {

$(\forall i : \mathbb{Z})((0 < i \le |x| \wedge esPrimo(i) \wedge_L x \bmod i = 0 \rightarrow ($

Pertenece((i, VecesQDivide(i, x)), res)))

}

FALTA

AUX VecesQueDivide(x, y: $\mathbb{Z}$) {

$\sum$

}

AUX MultFactores(res: seq<$\mathbb{Z}$>): $\mathbb{Z}$ {

$$\prod_{i=0}^{|res|-1} res[0]^{res[1]}$$

}

PRED OrdenadaCreciente(x: seq<$\mathbb{Z}$>) {

$(\forall i : \mathbb{Z})(0 \le i < (|x| - 1) \rightarrow_L x[i] < x[i+1])$

}

)

PRED TODOSDIVIDEN (res: seq⟨ℤ, ℤ⟩, x: ℤ){

$(\forall i:\mathbb{Z})(0 \le i < |x| \rightarrow_L res[i][0] \overset{res[i][1]}{} \ \text{MOD}\ x = 0)$

}

PRED TODOSPRIMOSEUDES(res: seq⟨ℤ, ℤ⟩){

$(\forall i:\mathbb{Z})(0 \le i < |seq| \rightarrow_L esPrimo(res[i][0]))$

}

PRED SINFACTORESREPETIDOS(res: seq⟨ℤ, ℤ⟩){

$(\forall i:\mathbb{Z})(0 \le i < |res| \rightarrow \neg(\exists j:\mathbb{Z})(0 \le j < |res| \wedge res[i][0] = res[j][0]))$

}

PRED TODOSPOSITIVOS(res: seq⟨ℤ, ℤ⟩){

$(\forall i:\mathbb{Z})(0 \le i < |res| \rightarrow_L res[i][0] > 0$

}

2.5. Especificación de problemas usando **inout**

Ejercicio 14. Dados dos enteros a y b, se necesita calcular su suma y retornarla en un entero c. ¿Cúales de las siguientes especificaciones son correctas para este problema? Para las que no lo son, indicar por qué.

a) `proc sumar` (inout a, b, c: $\mathbb{Z}$)
   `requiere` $\{True\}$
   `asegura` $\{a + b = c\}$

X a,b NO SON OUT. NO SE VAN A MODIFICAR. NO NOS INTERESA

→ SOLO OUT AUNQUE SERIA MEJOR

b) `proc sumar` (in a, b: $\mathbb{Z}$, ~~in~~out c: $\mathbb{Z}$)
   `requiere` $\{True\}$
   `asegura` $\{c = a + b\}$

✓

SUMAR(in a,b: $\mathbb{Z}$): $\mathbb{Z}$ {
asegura {res = a + b}
}

c) `proc sumar` (inout a, b: $\mathbb{Z}$, inout c: $\mathbb{Z}$)
   `requiere` $\{a = A_0 \land b = B_0\}$
   `asegura` $\{a = A_0 \land b = B_0 \land c = a + b\}$

X → NO NECESITO ESTADO Y MISMO ERROR 1

Ejercicio 13. Especificar los siguientes problemas sobre secuencias:

a) Dadas dos secuencias s y t, decidir si s está *incluida* en t, es decir, si todos los elementos de s aparecen en t en igual o mayor cantidad

b) Dadas dos secuencias s y t, devolver su *intersección*, es decir, una secuencia con todos los elementos que aparecen en ambas. Si un mismo elemento tiene repetidos, la secuencia retornada debe contener la cantidad mínima de apariciones del elemento en s y en t

c) Dada una secuencia de números enteros, devolver aquel que divida a más elementos de la secuencia. El elemento tiene que pertenecer a la secuencia original. Si existe más de un elemento que cumple esta propiedad, devolver alguno de ellos

d) Dada una secuencia de secuencias de enteros l, devolver una secuencia de l que contenga el máximo valor. Por ejemplo, si $l = \langle\langle 2,3,5\rangle, \langle 8,1\rangle, \langle 2,8,4,3\rangle\rangle$, devolver $\langle 8,1\rangle$ o $\langle 2,8,4,3\rangle$

e) Dada una secuencia l con todos sus elementos distintos, devolver la secuencia de *partes*, es decir, la secuencia de todas las secuencias incluidas en l, cada una con sus elementos en el mismo orden en que aparecen en l

a) $s \subseteq t \Rightarrow s[i] \in t$

PRED PERTENECE $(elem : \mathbb{Z}, l : seq\langle\mathbb{Z}\rangle)\{$
$(\exists i : \mathbb{Z})(0 \leq i < |l| \wedge_L l[i] = elem)$
$\}$

PRED TODOSINCLUIDOS $(s, t : seq\langle\mathbb{Z}\rangle)\{$
$(\forall i : \mathbb{Z})(\underline{0 \leq i < |t|} \wedge_L$ PERTENECE$(s[i], t))$
$\}$

me olvidé de recorrer t, $|s|$ no debería ser $> |t|$

PROC SUBSEQ(in $s, t : seq\langle\mathbb{Z}\rangle$) : Bool $\{$
REQUIERE: $\{|s| \leq |t|\}$
ASEGURA: $\{res = True \iff$ TODOSINCLUIDOS$(s, t)\}$
$\}$

b) $s = [1, 2, 3, 3]$, $t = [1, 1, 2, 4, 3, 3, 3]$

1 de 1 compara con 1. 1 < 2 => 2

¡REPETIDO!

Debo considerar que 1 NO esté en res.

AUX #CANTAPARICIONES(in s: seq<T>, e: T){

$$\sum_{i=0}^{|s|-1} \text{if}(s[i]=e) \text{ THEN } 1 \text{ ELSE } 0 \text{ ENDIF}$$

}

¿Qué le falta al Aux de limitación?

→ CANT MIN APARICIONES

AUX #MINAP(in e: ℤ, r, t: seq<ℤ>): ℤ{

if(#MINAP(e, r) > #MINAP(e, t)) THEN

#MINAP(e, t) ELSE

#MINAP(e, r)

ENDIF

}

VA en PREDS?

PRED ESTAENAMBASLISTAS(in e: ℤ, r, t: seq<ℤ>){

(PERTENECE(res[i], r) ∧ PERTENECE(res[i], t))

}

Devuelve Verdadero si un elemento está la MENOR CANT en 2 listas

PRED CANTAPMINLISTAS(in e: ℤ, res, l_1, l_2: seq<ℤ>){

$(\#\text{CANTAP}(\ell, \text{res}) = (\#\text{MINAP}(\ell, \ell_1, \ell_2))$

}

PROC INTERSECCION(in s, t: seq<$\mathbb{Z}$>) : seq<$\mathbb{Z}$>{

REQUIERE: {$|s| > 0 \wedge |t| > 0$}

$\neg 0 \wedge |t| > |s|$? *

ASEGURA: {$(\forall i: \mathbb{Z})(0 \leq i < |s| \wedge_L \neg$PERTENECE$(s[i], \text{res}) \rightarrow$ CANTAPMINLISTAS$(s[i], \text{res}, s, t))$

ASEGURA: {$(\forall i: \mathbb{Z})(0 \leq i < |\text{res}| \rightarrow_L$ ESTAENAMBASLISTAS$(\text{res}[i], s, t))$}

ASEGURA: {$(\forall i: \mathbb{Z})(0 \leq i < |\text{res}|) \rightarrow_L$ CANTAPMINLISTAS$(\text{res}[i], \text{res}, s, t))$}

}

IF$(|s| > |t|)$ THEN $|s|$ ELSE $|t|$ ENDIF

Ej: s: [1, 2, 2] t: [1, 1, 2, 3]

res: [1, 2, 3] pero si recorro $|s|^{-1}$ veces pierdo el 3.

c) s: [2, 4, 6, 8] = 2 y 2 ∈ s

2 DIVIDE A 3, 4 DIVIDE A 1, 6 DIVIDE A 1, 8 DIVIDE A 2.

s: [0, 2, 3, 4] SE ROMPE

s: [1, 2, 3, 4, 1] : 1 y 1 ∈ s

s: [2, 3, 4, 6, 8, 9, 15]: 2 o 3 y ambos ∈ s

2 DIVIDE A 3, 3 DIVIDE A 3, 4 DIVIDE A 1, 6 DIVIDE A 2, ...,

s: [2, 3] ¿Qué devuelve este caso? ¿1? xq $1 \notin s$.

PROC DIVIDEAMAYORCANT(in n: $seq\langle\mathbb{Z}\rangle$): $\mathbb{Z}$ {

REQUIERE: $\{|n|>0\}$

ASEGURA: $\{(\exists i:\mathbb{Z})(0\leq i<|n| \wedge (\forall j:\mathbb{Z})(0\leq j<|n| \rightarrow (n[j] \bmod n[i] = 0))\}$

ASEGURA: {PERTENECE(res, n)}

X NO. Acá digo que un num divide a todos.

AUX DIVIDEAXCANT(in e: $\mathbb{Z}$, n: $seq\langle\mathbb{Z}\rangle$): $\mathbb{Z}$ {

$\sum_{i=0}^{|n|-1}$ if(n[i] mod e) = 0 THEN 1 ELSE 0 ENDIF

}

PROC DIVIDEMAYORCANT(in n: $seq\langle\mathbb{Z}\rangle$): $\mathbb{Z}$ {

ASEGURA: $\{(\forall i:\mathbb{Z})(0\leq i<|n| \rightarrow ((\text{DIVIDEAXCANT}(n[i], n) = \text{DIVIDEAXCANT}(res, n)) \rightarrow res = n[i]))$

ASEGURA: {PERTENECE(res, n)}

}

d) Debo asegurar que el resultado es una de las secuencias que tiene el máximo valor.
Esa secuencia res está en la original

PROC OBTENERSECMAXVALOR (in l: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$): $seq\langle\mathbb{Z}\rangle$ {

REQUIERE: $\{|l|>0\}$

ASEGURA: $\{(\exists i:\mathbb{Z})(0\leq i<|l| \wedge_L$ SECUENCIACONMAX(l)}

ASEGURA: {PERTENECE(res, l)}

ASEGURA: $\{|res| > 0\}$
}

PRED SECUENCIACONMAX (in l: seq<seq<$\mathbb{Z}$>>) {
$(\exists j: \mathbb{Z})(0 \leq j < |l[i]| \wedge res[\text{INDICEDELMAXIMO}(res)] = l[i][j] \rightarrow_L res = l[i])$

¿PUEDO HACER ESO?

}

Ej: [[2,1,3], [1,4,2], [4,1,2]]

. E.R = [1,4,2] o [4,1,2]

1. [2,1,3], } en [2,1,3] el 4? => V ∧ F →$_L$ res = $l[i]$
=> TRUE.. MAL?

e) $\#P(a) = 2^m$

$[1] = 2^1 = [[\,],[1]]$

$[1,2] = 2^2 = [[\,],[1],[2],[1,2]]$

PROC PARTES(in l: seq<$\mathbb{Z}$>): seq<seq<$\mathbb{Z}$>> {

REQUIERE: {SINREPETIDOS(l)}
ASEGURA: $\{|res| = 2^{|l|}\}$
ASEGURA: {ELEMDERESENL(l)}
ASEGURA: {LONGITUDELEMSRES(l)}
ASEGURA: $\{(\forall i,j: \mathbb{Z})((0 \leq i < |res| \wedge 0 \leq j < |res[i]|-1 \rightarrow_L (\exists \bar{\imath}: \mathbb{Z})($
$0 \leq \bar{\imath} < |l|-1 \wedge_L res[i][j] = l[i][\bar{\imath}] \rightarrow_L res[i][j+1] =$
$l[i][\bar{\imath}+1]\}$

ORDENADOS: Ej: l: [1,2] → [1],[1,2] pero no [1],[2,1]

}

FALLA

PRED ELEMDERESENL $(l: seq\langle\mathbb{Z}\rangle)\{$

$(\forall i,j:\mathbb{Z})((0 \le i < |res| \wedge 0 \le j < |res[i]|) \to res[i][j] \in l)$

}

PRED LONGITUDELEMSRES$(l: seq\langle\mathbb{Z}\rangle)\{$

$(\forall i,j:\mathbb{Z})((0 \le i < |res| \wedge 0 \le j < |res[i]|) \to |res[i][j]| \le |l|)$

}

DEBO GARANTIZAR ADEMAS Q SI LA LISTA ES [1, 2]

EL CONJ PARES NO PUEDE TENER [2,1]

OSEA SI MI SEQ[i] = 1 ENT EL PRIMER VALOR EN CHOCAR EN L DEBE SER 1.

→ head

Ejercicio 15. Dada una secuencia l, se desea sacar su primer elemento y devolverlo. Decidir cúales de estas especificaciones son correctas. Para las que no lo son, indicar por qué y justificar con ejemplos.

a) **proc** tomarPrimero (inout l: $seq\langle\mathbb{R}\rangle$) : $\mathbb{R}$
  requiere $\{|l| > 0\}$
  asegura $\{res = \text{head}(l)\}$

✓

b) **proc** tomarPrimero (inout l: $seq\langle\mathbb{R}\rangle$) : $\mathbb{R}$
  requiere $\{|l| > 0 \wedge l = L_0\}$
  asegura $\{res = \text{head}(L_0)\}$

✓ → ¿Está bien inicializar Lo con l, no?

c) **proc** tomarPrimero (inout l: $seq\langle\mathbb{R}\rangle$) : $\mathbb{R}$
  requiere $\{|l| > 0\}$
  asegura $\{res = \text{head}(L_0) \wedge |l| = |L_0| - 1\}$

X

d) **proc** tomarPrimero (inout l: $seq\langle\mathbb{R}\rangle$) : $\mathbb{R}$
  requiere $\{|l| > 0 \wedge l = L_0\}$

X

asegura $\{res = \text{head}(L_0) \wedge l = \text{tail}(L_0)\}$

↳ no hace falta. l se modifica al usar head.

a) l: es solo IN. La respuesta es un elemento de la lista, si bien l se modifica NO nos interesa su versión MODIFICADA.

Ej: [1,2,3] -> 1, no me interesa cómo quedó l.

c) No se indica qué es l_0, con que se inicializa.

d) l: [1,2,3] => l_0 = [1,2,3] -> head(l_0) = 1
l_0 = [2,3]
tail(l_0) = [3] -> PIERDO el ELEM 2.

PNE6

Ejercicio 16. Dada una secuencia de enteros, se requiere multiplicar por 2 aquéllos valores que se encuentran en posiciones pares. Indicar por qué son incorrectas las siguientes especificaciones y proponer una alternativa correcta.

a) **proc duplicarPares** (inout l: $seq\langle\mathbb{Z}\rangle$)
   requiere $\{l = L_0\}$
   asegura {
   $|l| = |L_0| \wedge$
   $(\forall i : \mathbb{Z})((0 \leq i < |l| \wedge i \mod 2 = 0) \rightarrow_L l[i] = 2 * L_0[i])$
   }

está fuera del AnB del CUANTIFICADOR. ↗ En AZUL los paréntesis q faltaban

b) **proc duplicarPares** (inout l: $seq\langle\mathbb{Z}\rangle$)
   requiere $\{l = L_0\}$
   asegura {
   ~~$(\forall i : \mathbb{Z})((0 \leq i < |l| \wedge i \mod 2 \neq 0) \rightarrow_L l[i] = L_0[i]) \wedge$~~
   $(\forall i : \mathbb{Z})((0 \leq i < |l| \wedge i \mod 2 = 0) \rightarrow_L l[i] = 2 * L_0[i])$
   }

↗ No lo necesito. solo MODIF IND pares.

c) **proc duplicarPares** (in~~out~~ l: $seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{Z}\rangle$
   requiere $\{True\}$
   asegura {
   $|l| = |res| \wedge$
   $(\forall i : \mathbb{Z})((0 \leq i < |l| \wedge i \mod 2 \neq 0) \rightarrow_L res[i] = l[i]) \wedge$
   $(\forall i : \mathbb{Z})((0 \leq i < |l| \wedge i \mod 2 = 0) \rightarrow_L res[i] = 2 * l[i])$
   }

↗ l NO es INOUT xq la lista res NO es l. res es otra seq.

a) Faltan paréntesis. -> l[i] = 2 * l_0[i] NO está dentro del CUANTIFICADOR.

b) Como l es INOUT no tiene sentido presion sobre los IMPARES xq quedan igual.

c) l NO es inout xq la INFO sale en una nueva VARIABLE.

Ejercicio 17. Especificar los siguientes problemas de modificación de secuencias:

a) proc primosHermanos(inout l : $seq\langle\mathbb{Z}\rangle$), que dada una secuencia de enteros mayores a dos, reemplaza dichos valores por el número primo menor más cercano. Por ejemplo, si $l = \langle 6, 5, 9, 14\rangle$, luego de aplicar primosHermanos(l), $l = \langle 5, 3, 7, 13\rangle$

b) proc reemplazar(inout l : $seq\langle\text{Char}\rangle$, in a, b : Char), que reemplaza todas las apariciones de a en l por b

c) proc limpiarDuplicados(inout l : $seq\langle\text{Char}\rangle$, out dup : $seq\langle\text{Char}\rangle$), que elimina los elementos duplicados de l dejando sólo su primera aparición (en el orden original). Devuelve además, dup una secuencia con todas las apariciones eliminadas (en cualquier orden)

a) PROC PRIMOSHERMANOS(inout l: $seq\langle\mathbb{Z}\rangle$){

REQUIERE: $\{l_0 = l\}$
REQUIERE: $\{|l| > 0\}$
REQUIERE: {MAYORESADOS(l)}
ASEGURA: $\{|res| = |l|\}$
ASEGURA: {PRIMOSCERCANOS(l_0)}
ASEGURA: {TODOSPRIMOS(l)}

}

PRED MAYORESADOS(in l: $seq\langle\mathbb{Z}\rangle$){

$(\forall i : \mathbb{Z})(0 \leq i < |l| \to_L l[i] > 2)$

}

AUX PRIMOSCERCANOS(INOUT: l){

$(\forall i:\mathbb{Z})(0\leq i<|l| \rightarrow_L l[i] = \text{PRIMOCERCANO}(l_0[i]))$

}

AUX PRIMOCERCANO(x: $\mathbb{Z}$){

$(\exists i,j:\mathbb{Z})(2\leq i<x \wedge_L \text{esPrimo}(i) \wedge_L x \text{ MOD } i = 0 \wedge i>j)$

}

PRED TODOSPRIMOS(x: seq⟨$\mathbb{Z}$⟩){

$(\forall i:\mathbb{Z})(0\leq i<|x| \rightarrow_L \text{esPrimo}(x[i]))$

}

b) PROC REEMPLAZAR(INOUT l: seq⟨CHAR⟩, IN a,b: CHAR){

REQUIERE: $\{|l|>0 \wedge l_0 = l\}$ ¿ESTA BIEN?

ASEGURA: $\{(\forall i:\mathbb{Z})(0\leq i<|l| \wedge l_0[i] = a \rightarrow l[i] = b)$

ASEGURA: $\{(\forall i:\mathbb{Z})(0\leq i<|l| \wedge l_0[i] \neq a \rightarrow l[i] = l_0[i]\}$

ASEGURA: $\{|l| = |l_0|\}$

ASEGURA: $\{\#\text{CANTAPARICIONES}(a,l) = 0\}$

}

C)

Ejercicio 17. Especificar los siguientes problemas de modificación de secuencias:

a) proc primosHermanos(inout l : $seq\langle\mathbb{Z}\rangle$), que dada una secuencia de enteros mayores a dos, reemplaza dichos valores por el número primo menor más cercano. Por ejemplo, si $l = \langle 6,5,9,14\rangle$, luego de aplicar primosHermanos(l), $l = \langle 5,3,7,13\rangle$

b) proc reemplazar(inout l : $seq\langle\text{Char}\rangle$, in a,b : Char), que reemplaza todas las apariciones de a en l por b

c) proc limpiarDuplicados(inout l : $seq\langle \mathsf{Char}\rangle$, out dup : $seq\langle \mathsf{Char}\rangle$), que elimina los elementos duplicados de l dejando sólo su primera aparición (en el orden original). Devuelve además, dup una secuencia con todas las apariciones eliminadas (en cualquier orden)

PROC LIMPIARDUPLICADOS (INOUT l: $seq\langle CHAR\rangle$, OUT dup: $seq\langle CHAR\rangle$) {

REQUIERE: $\{ l_0 = l \}$

ASEGURA: $\{ |l| \leq |l_0| \wedge |dup| \geq |l| \}$

ASEGURA: $\{ SINREPETIDOS(l) \}$

ASEGURA: $\{ PRIMERAPOSICION(l) \}$

ASEGURA: $\{ DUPENLISTA(l, dup) \}$

ASEGURA: $\{ ESTANENDUP(l, dup) \}$

}

PRED PRIMERAPOSICION (l: $seq\langle CHAR\rangle$) {

$(\forall i : \mathbb{Z})\,(0 \leq i < |l| \rightarrow_L (\forall j : \mathbb{Z})(0 \leq j < |l_0| \wedge l[i] = l_0[j] \rightarrow i < j\,)\,)$

}

PRED DUPENLISTA (l, dup : $seq\langle CHAR\rangle$) {

$(\forall i : \mathbb{Z})(0 \leq i < |dup| \rightarrow_L PERTENECE(dup[i], l_0) \wedge PERTENECE(dup[i], l)\,)$

}

PRED ESTANENDUP (l, dup : $seq\langle CHARS\rangle$) {

$(\forall i : \mathbb{Z})(0 \leq i < |l_0| \rightarrow (\exists j : \mathbb{Z})(0 \leq j < |l_0| \wedge l_0[i] = l_0[j] \wedge i \neq j \rightarrow l_0[j] \in dup))$

}

2.4. Especificación de problemas

Ejercicio 12. Especificar los siguientes problemas:

a) Dado un entero, decidir si es par

b) Dado un entero n y otro m, decidir si n es un múltiplo de m

c) Dado un entero, listar todos sus divisores positivos (sin duplicados)

d) Dado un entero positivo, obtener su descomposición en factores primos. Devolver una secuencia de tuplas (p, e), donde p es un factor primo y e es su exponente, ordenada en forma creciente con respecto a p

a) PROC ESPAR(IN $m: \mathbb{Z}$) : bool {

REQUIERE: {true}

ASEGURA: {res = true <=> m MOD $2 = 0$}

}

b) PROC ESMULTIPLO(IN $n: \mathbb{Z}$, IN $m: \mathbb{Z}$) : bool {

REQUIERE: {true}

ASEGURA: {res = true <=> n MOD $m = 0$}

}

c) res: los sus divisores. Digo que ∀ res divide a M.
Después busco la CANT DIVISORES = |res|.

PROC DIVISORESPOSITIVOS(IN $n: \mathbb{Z}$) : seq<$\mathbb{Z}$> {

REQUIERE: {n != 0}

ASEGURA: {$(\forall i: \mathbb{Z})(0 \leq i < |res| \rightarrow_L n \text{ MOD } res[i] = 0$}

ASEGURA: {$|res| = \sum_{i=1}^{n}$ IF(n MOD $i = 0$) THEN 1 ELSE 0}

ASEGURA: {$(\forall m: \mathbb{Z})((1 \leq m \leq n \wedge n \text{ MOD } m = 0) \rightarrow_L m \in res)$}

ASEGURA: {$(\forall x: \mathbb{Z})(1 \leq x < |res| \rightarrow_L res[x-1] < res[x])$}

}

d) Dado un entero positivo, obtener su descomposición en factores primos. Devolver una secuencia de tuplas (p, e), donde p es un factor primo y e es su exponente, ordenada en forma creciente con respecto a p

$[(\overset{\rightarrow \text{MAL}}{1,3})] = [\underbrace{(2, \overbrace{3), (3,}^{\geq 1} 4)}_{\text{ORDENADOS}}]$

PROC DESCOMPONERENPRIMOS(IN $m: \mathbb{Z}$): seq<$\mathbb{Z} \times \mathbb{Z}$> {

REQUIERE: {$m > 1$}

ASEGURA: {$m = \prod_{j=0}^{|res|-1} res[j]_0^{\,res[j]_1}$}

ASEGURA: $\{(\forall i:\mathbb{Z})(0 \leq i < |res| \rightarrow_L res[i]_0 > 1)\}$ Divisores (POS)

ASEGURA: $\{(\forall i:\mathbb{Z})(0 \leq i < |res| \rightarrow_L res[i]_1 \geq 1)\}$ VALORES ≥ 1

ASEGURA: $\{(\forall i:\mathbb{Z})(1 \leq i < |res| \rightarrow_L res[i-1]_0 < res[i]_0)\}$ ORD

ASEGURA: $\{(\forall i:\mathbb{Z})(0 \leq i < |res| \rightarrow_L esPrimo(res[i]_0))\}$ FACT PRIMOS

ASEGURA: $\{(\forall i,j:\mathbb{Z})((2 \leq i,j < |res| \wedge j > res[i]_1) \rightarrow_L m \text{ MOD } res[i]_0^{res[i]_1+1} \neq 0)\}$

}

PRED EsPrimo(m){

$m>1 \wedge (\forall j:\mathbb{Z})(2 \leq j < m \rightarrow_L m \text{ MOD } j \neq 0)$

}

a) proc primosHermanos(inout l : $seq\langle\mathbb{Z}\rangle$), que dada una secuencia de enteros mayores a dos, reemplaza dichos valores por el número primo menor más cercano. Por ejemplo, si $l = \langle 6, 5, 9, 14\rangle$, luego de aplicar primosHermanos(l), $l = \langle 5, 3, 7, 13\rangle$

$[6,5,9,14] \Rightarrow [5,3,7,13]$

PROC PrimosHermanos(inout $l : seq\langle\mathbb{Z}\rangle$){

REQUIERE: $\{l = l_0\}$

REQUIERE: $\{|l_0| > 0\}$

REQUIERE: $\{(\forall i:\mathbb{Z})(0 \leq i < |l| \rightarrow_L l[i] > 2)\}$

ASEGURA: $\{(\forall i:\mathbb{Z})(0 \leq i < |l| \rightarrow_L \neg(\exists j:\mathbb{Z})((l[i] \leq j < l_0[i] \wedge esPrimo(j)))\}$

ASEGURA: $\{(\forall i:\mathbb{Z})(0 \leq i < |l| \rightarrow_L esPrimo(l[i]))\}$

ASEGURA: $\{|l| = |l_0|\}$

}