Especificacion de problemas

Ejercicio 1: Nombrar los siguientes predicados sobre enteros.

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

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

Ejercicio 3: Escriba los siguientes predicados sobre numeros enteros en lenguaje de especificacion.

pred sonCoprimos (x,y : $\mathbb{Z}$) {
$(\forall n : \mathbb{N})\ (1 < n < max\{x, y\} \wedge_L esPrimo(n) \wedge_L \neg((x\ mod\ n\ =\ 0) \iff (y\ mod\ n\ =\ 0)))$
}
pred mayorPrimoQueDivide (x:$\mathbb{Z}$,y:$\mathbb{Z}$) {
$(\exists!\ n : \mathbb{Z})\ (y \leq n < x \wedge_L esPrimo(n) \wedge_L xmodn = 0 \wedge_L n = y)$
}

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

pred todosPositivos (s: $seq\langle\mathbb{Z}\rangle$) {
$(\forall i : \mathbb{Z})\ (0 \leq i < |s| \rightarrow_L s[i] \geq 0)$
}
pred todosDistintos (s: $seq\langle\mathbb{Z}\rangle$) {
$(\forall i : \mathbb{Z})\ ((0 \leq i < |s|) \rightarrow_L (\forall j : \mathbb{Z})\ (0 \leq j < |s| \wedge i \neq j \rightarrow_L (s[i] \neq s[j])))$
}

Ejercicio 4: Escriba los siguientes predicados auxiliares sobre secuencias de enteros, aclarando los tipos de los parametros que recibe.

1. *esPrefijo*, que determina si una secuencia es prefijo de otra.
   pred esPrefijo (s1: $seq\langle\mathbb{Z}\rangle$, s2$seq\langle\mathbb{Z}\rangle$) {
   $|s1| \leq |s2| \wedge_L (\forall i : \mathbb{Z})\ (0 \leq i < |s1| \wedge_L s1[i] = s2[i])$
   }
2. *estaOrdenada*, que determina si la secuencia se encuentra ordenada de menor a mayor.
   pred estáOrdenada (s: $seq\langle\mathbb{Z}\rangle$) {
   $(\forall n : \mathbb{Z})\ (0 \leq n < (|s| - 1) \wedge_L (s[n] \leq s[n + 1]))$
   }
3. *hayUnParQueDivideAlResto*, que determina si hay un elemento par en la secuencia que dividie a todos los otros elementos de la secuencia.
   pred hayUnParQueDivideAlResto (s:$seq\langle\mathbb{Z}\rangle$) {
   $(\exists n : \mathbb{Z})\ ((s[n]\ mod\ 2 = 0 \wedge_L (\forall i : \mathbb{Z})\ (0 \leq i < |s| \wedge_L s[i]\ mod\ s[n] = 0)))$
   }
4. PENDIENTE 4.d se ve divertido

Analisis de especificacion

Ejercicio 6: Las siguientes especificaciones no son correctas. Indicar por que y corregirlas de modo que describan correctamente el problema

`progresionGeometricaFactor2:` Indica si la secuencia l representa una progresion geometrica 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}
`asegura` $\{res = \text{true} \longleftrightarrow ((\forall i : \mathbb{Z})\ (0 \leq i < |l| \longrightarrow_L l[i] = 2 * l[i-1]))\}$

El problema es el rango de la secuencia, en i=0 al hacer l[0]=2*l[0-1] salimos del rango en el que la secuencia esta definida.
Una posible solucion es cambiar el rango a $0 < i \leq |l|$

`proc progresionGeometricaFactor2` (in l: $seq\langle\mathbb{Z}\rangle$) : Bool
`requiere` {true}
`asegura` $\{res = \text{true} \longleftrightarrow ((\forall i : \mathbb{Z})\ (0 < i \leq |l| \longrightarrow_L l[i] = 2 * l[i-1]))\}$

minimo: Devuelve en res el menor elemento de l.

`proc minimo` (in l: $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z}$
`requiere` {true}
`asegura` $\{(\forall y : \mathbb{Z})\ ((y \in l \wedge y \neq x) \longrightarrow y > res)\}$

La variable x no significa nada y tiene que ser reemplazada por res, y falta asegurar que res es un elemen de l, y cambiar el >por $\geq$

`proc minimo` (in l: $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z}$
`requiere` {true}
`asegura` $\{res \in l\}$
`asegura` $\{(\forall y : \mathbb{Z})\ ((y \in l \wedge y \neq res) \longrightarrow y \geq res)\}$

Ejercicio 8: la primera no puede ser porque requiere ambos casos a la vez, la segunda me parece mejor que la tercera y la cuarta, asi que si busco simpleza iria por la segunda si busco legibilidad de la especif iria por la cuarta.

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` $\{res > x\}$

¿Qué devuelve el algoritmo si recibe x = 3? ¿El resultado hace verdadera la postcondición de unoMasGrande?
Con x=3 el algoritmo devuelve 9, y este resultado hace verdadera la postcondiciónpues 3<9

¿Qué sucede para las entradas x = 0,5, x = 1, x = -0,2 y x = -7?
$f(0{,}5) = 0{,}25 \wedge f(1) = 1 \wedge f(-0{,}2) == 0{,}04 \wedge f(-7) = 49$

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

`proc unoMasGrande` (in x: $\mathbb{R}$) : $\mathbb{R}$
`requiere` $\{x < 0 \vee x > 1\}$
`asegura` $\{res > x\}$

Relación de fuerza

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

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

Q1: $\{r \geq x^2\}$
Q2: $\{r \geq 0\}$
Q3: $\{r = x^2\}$

Indicar la relacion de fuerza entre P1, P2 y P3
$P2 \subset P1 \subset P3$

Indicar la relacion de fuerza entre Q1, Q2 y Q3
$Q2 \subset Q1 \subset Q3$

Escribir dos programas que cumplan con la siguiente especificacion:

`proc hagoAlgo` (in x:$\mathbb{R}$) : $\mathbb{R}$
`requiere` $\{x \geq 0\}$
`asegura` $\{res \geq x^2\}$

$result = x * x$
$result = (x * x) + 4$
a- si
b- si
c- si
d- no necesariamente
e- si
f- no necesariamente

Hay requieres y aseguras mas fuertes que otros, para que sea seguro reemplazar una pre y/o postcondición de forma segura creo que hay que hacerlo por alguna mas fuerte que abarque mas casos.

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

`proc p1` (in x:$\mathbb{Z}$, in n:$\mathbb{Z}$) : $\mathbb{Z}$
`requiere` $\{x \neq 0\}$
`asegura` $\{x^n - 1 < res \leq x^n\}$

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

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

$$x \neq 0 \longrightarrow (n \leq 0 \longrightarrow x \neq 0)$$
$$x \neq 0 \longrightarrow (\neg(n \leq 0) \vee (x \neq 0))$$
$$\neg(x \neq 0) \vee (n > 0) \vee (x \neq 0)$$
$$(x = 0) \vee (n > 0) \vee (x \neq 0)$$
$$(n > 0) \vee True$$
$$True$$

Por lo tanto queda demostrado que $x \neq 0 \longrightarrow (n \leq 0 \longrightarrow x \neq 0)$

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?
Si porque la funcion floor esta contenida en $x - 1 < \lfloor x \rfloor \leq x$
¿Podemos concluir que a satisface la especificación de p1?
Podemos concluir que el algoritmo satisface ambas especificaciones.

Especificacion de problemas

Ejercicio 12: Especificar los siguientes problemas.

Dado un entero decidir si es par.

proc esPar (in x:$\mathbb{Z}$) : Bool
requiere {true}
asegura $\{x \ mod \ 2 = 0\}$

Dado un entero n y otro m, decidir si n es un multiplo de m

proc esMultiplo (in n: $\mathbb{Z}$, in m: $\mathbb{Z}$) : Bool
requiere {true}
asegura $\{res = \text{true} \longleftrightarrow (\exists k : \mathbb{Z})\ (k * n = m)\}$

Dado un entero, listar todos sus divisores positivos.

proc listaDivisores (in x: $\mathbb{Z}$) : $seq\langle\mathbb{Z}\rangle$
requiere {true}
asegura $\{(\forall n : \mathbb{Z})\ (1 \leq n \leq x \wedge_L divideA(n, x) \wedge_L n \in res)\}$

pred divideA (n:$\mathbb{Z}$,m:$\mathbb{Z}$) {
$m \ mod \ n = 0$
}

Dado un entero positivo, obtener su descomposici´on 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 pred esPrimo (n: $\mathbb{N}$) {
$\nexists m \in \{2, 3, \cdots, n-2, n-1\} | divideA(m, n) = 0$
}

pred ordenada (s: $seq\langle\mathbb{N} \times \mathbb{N}\rangle$) {
$(\forall i : \mathbb{Z})\ (0 \leq i < |s| - 1 \wedge_L s[i]_1 < s[i+1]_1)$
}

proc descomponerEnPrimos (in n: $\mathbb{N}$) : $seq\langle\mathbb{Z} \times \mathbb{Z}\rangle$
requiere {true}
asegura $\{((\forall i : \mathbb{Z})\ (0 \leq i < |res| \wedge_L esPrimo(res[i]_1) \wedge_L res[i]_2 \geq 0) \wedge_L \prod_{j=0}^{|res-1|} res[j]_1^{res[j]_2} = n \wedge_L estaOrdenada(res))\}$

Ejercicio 13: Especificar los siguientes problemas sobre secuencias.

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.

proc incluidaEn (in s: $seq\langle T\rangle$, in t: $seq\langle T\rangle$) : Bool
requiere {true}
asegura $\{|t| < |s| \wedge_L res = \text{false}\}$
asegura $\{|s| = 0 \wedge_L res = \text{true}\}$
asegura $\{(\forall i : \mathbb{Z})\ (0 \leq i < |t| \wedge_L \#(t, s[i]) \geq \#(s, s[i])) \wedge_L res = \text{true}\}$

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

proc interseccion (in s:$seq\langle T\rangle$, in t:$seq\langle T\rangle$) : $seq\langle T\rangle$
requiere {true}
asegura $\{res \neq \langle\rangle \longleftrightarrow (\exists e : T)\ (e \in s \wedge e \in t)\}$
asegura $\{(\forall i : \mathbb{Z})\ (0 \leq i < |res| \wedge_L \#(res, res[i]) = minimo(\#(s, res[i]), \#(t, res[i])))\}$

Dada una secuencia de numeros enteros devolver aquel que divida a mas elementos de la secuencia. El elemento tiene que pertenecer a la secuencia original. Si existen mas de un elemento que cumplen esa propiedad devolver alguno de ellos.

proc divideAMas (in l: $seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{Z}\rangle$
requiere $\{\exists n \in l : n \neq 0\}$
asegura $\{\exists n \in l : esElQueMasDivide(n, l) \wedge_L res = n\}$

pred esElQueMasDivide (n: $\mathbb{Z}$, s: $seq\langle\mathbb{Z}\rangle$) {
$\forall\ m \in s, \sum_{j=0}^{|s|-1}$ if $s[j]\ mod\ m = 0$ then 1 else 0 fi $\leq \sum_{j=0}^{|s|-1}$ if $s[j]\ mod\ n = 0$ then 1 else 0 fi
}

Dada una secuencia de secuencias de enteros l, devolver una secuencia de l que contenga el maximo valor.

proc secuenciaDelMaximo (in l: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) : $seq\langle \mathbb{Z}\rangle$
requiere {true}
asegura $\{(\exists m, i : \mathbb{Z})\ (0 \leq i < |l| \wedge_L m \in l[i] \wedge_L \forall secuencia \in l, esMayorQueTodos(m, secuencia)) \wedge_L res = l[i]\}$

pred esMayorQueTodos (n: $\mathbb{Z}$, l: $seq\langle \mathbb{Z}\rangle$) {
$\forall valor \in l, n \geq valor$
}

proc partes (in l: $seq\langle T\rangle$) : $seq\langle seq\langle T\rangle\rangle$
requiere $\{res = \langle\rangle \longleftrightarrow |l| = 0\}$
asegura $\{(\forall i : \mathbb{Z})\ (0 \leq i < |l| \wedge_L (\forall k : \mathbb{Z})\ (0 \leq k < |l| - i) \wedge_L \sum_{j=0}^{i} \langle l[k+j]\rangle \in res)\}$

Especificacion con parametros inout

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

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

incorrecta pues no tiene sentido que todos los parametros sean inout

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

Esta es la correcta. El valor previo de c no es requerido asi hay que hacer $C_0 = c$

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

no tiene sentido que a y b sean inout y el requiere

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

no tiene sentido que a y b sean inout y el requiere

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

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

Incorrecto pues no asegura que el resto de la secuencia se mantiene igual.

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

Incorrecto pues l se mantiene con su primer elemento.

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

Incorrecto pues puede ser que devolvamos la head pero que a la lista le falte cualquier otro elemento.

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

Correcta pues aseguramos que l es la secuencia original sin la cabeza y retornamos la cabeza.

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

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

Parece correcta

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

Redundancia al aclarar que los indices impares son iguales.

proc duplicarPares (inout l: $seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{Z}\rangle$
requiere $\{\text{true}\}$
asegura $\{|l| = |res|$
$\dots$
$\dots\}$

No usas el parametro inout y simplemente explusas una lista que hace lo que queres, pero no modificas correctamente la secuencia original.

Ejercicio 17: Especificar los siguientes problemas de modificacion de secuencias:

proc primosHermanos(inout l: $seq\langle\mathbb{Z}\rangle$), que dada una secuencia de enteros mayores a dos, reemplaza dichos valores por el numero primo menor mas cercano.

proc primosHermanos (inout l: $seq\langle\mathbb{Z}\rangle$)
requiere $\{todosMayoresA2(l)\}$
requiere $\{|l| > 0\}$
requiere $\{l = L_0\}$
asegura $\{(\forall i : \mathbb{Z})\ (0 \leq i < |l| \land_L esElPrimoMenorMasCercano(l[i], L_0[i]))\}$

pred esElPrimoMenorMasCercano (candidato: $\mathbb{N}$, numero: $\mathbb{N}$) {
$esPrimo(candidato) \land_L \nexists\ p \in \mathbb{N} : candidato < p < numero \land esPrimo(p)$
}
pred todosMayoresA2 (l: $seq\langle\mathbb{Z}\rangle$) {
$\forall n \in l, n > 2$
}

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

proc reemplazar (inout l:$seq\langle char\rangle$, in a,b : char)
requiere $\{l = L_0\}$
asegura $\{(\forall i : \mathbb{Z})\ (0 \leq i < |l| \land_L L_0[i] = a \land_L l[i] = b)\}$

`proc limpiarDuplicados(inout l :` $seq\langle char\rangle$`, out dup :` $seq\langle char\rangle$`)`, que elimina todos los elementos duplicados de l dejando solo su primera aparicion (en el orden original). Devuelve ademas, 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` $\{\text{true}\}$
`requiere` $\{l = L_0\}$
`requiere` $\{dup = \langle\rangle\}$
`asegura` $\{(\forall i : \mathbb{Z})\ (0 \leq i < |L_0| \wedge_L L_0[i] \in l \wedge \#(L_0[i], l) = 1 \wedge_L estaOrdenada(l, L_0))\}$
`asegura` $\{(\forall i : \mathbb{Z})\ (0 \leq i < |l| \wedge_L \#(dup, l[i]) = \#(L_0, l[i]) - 1)\}$

`pred estaOrdenada` (l: $seq\langle char\rangle$, l0: $seq\langle char\rangle$) {
$(\forall i : \mathbb{Z})\ (0 \leq i < |l| \wedge_L$
$(\exists!\ k : \mathbb{Z})\ (0 \leq k < |s0| \wedge_L l[i] = l0[k] \wedge_L (\nexists\ e \in \mathbb{Z} : 0 \leq e < l0[k] \wedge_L l0[e] = l[i])))$
}