ALGUNAS DE MIS SOLUCIONES

Sergio Esteban

1er Cuatrimestre 2022 - Algoritmos y Estructura de Datos 1

1 Introducción al lenguaje de especificación

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

- estaAcotada, que determina si todos los elementos de una secuencia están dentro del rango [1,100].

s[0]	s[1]	$\cdots$	s[i]	$\cdots$	s[n-2]	s[n-1]

  $\Rightarrow s[i] \in [1, 100]$

  `pred estaAcotada(` $s{:}seq\langle Z\rangle$ `){` $(\forall i : Z)(0 \leq i < n \Rightarrow_L s[i] \in [1, 100])$ `}`

- capicua, que es verdadera si una secuencia es capicúa.

s[0]	s[1]	$\cdots$	s[i]	$\cdots$	s[n-2]	s[n-1]

  $\Rightarrow s[0] = s[n-1] \wedge s[1] = s[n-2] \wedge \cdots \wedge s[i] = s[n-(i+1)]$

  `pred esCapicua(` $s{:}seq\langle Z\rangle$ `){` $(\forall i : Z)(0 \leq i < n \Rightarrow_L s[i] = s[n-(i+1)])$ `}`

- esPrefijo, que es verdadera si una secuencia es prefijo de otra.

  RPTA. Los prefijos no pueden especificarse pues varian según el idioma y palabra.

- estaOrdenada, que es verdadera si la secuencia está ordenada de menor a mayor.

s[0]	s[1]	$\cdots$	s[i]	$\cdots$	s[n-2]	s[n-1]

  $\Rightarrow s[0] \leq s[1] \leq \cdots \leq s[i] \leq s[i+1] \leq \cdots \leq s[n-1]$

  `pred estaOrdenada(` $s{:}seq\langle Z\rangle$ `){` $(\forall i : Z)(0 \leq i < n-1 \Rightarrow_L s[i] \leq s[i+1])$ `}`

- todosPrimos, que es verdadera si todos los elementos de la secuencia son números primos.

s[0]	s[1]	$\cdots$	s[i]	$\cdots$	s[n-2]	s[n-1]

  $\Rightarrow esPrimo(s[0]) \wedge esPrimo(s[1]) \wedge \cdots \wedge esPrimo(s[n-1])$

  `pred todosPrimos(` $s{:}seq\langle Z\rangle$ `){` $(\forall i : Z)(0 \leq i < n \Rightarrow_L esPrimo(s[i]))$ `}`

  `pred esPrimo( p:Z ){ /* Se definio en la teórica*\ }`

- primosEnPosicionesPares, que es verdadero si todos los elementos primos de una secuencia están en una posición par.

s[0]	s[1]	$\cdots$	s[i]	$\cdots$	s[n-2]	s[n-1]

  $\Rightarrow esPrimo(s[0]) \wedge esPrimo(s[2]) \wedge \cdots \wedge esPrimo(s[i])$

  $\iff i \equiv 0(mod 2)$

  `pred primosEnPosicionesPares(` $s{:}seq\langle Z\rangle$ `){` $(\forall i : Z)((0 \leq i < n \wedge i \bmod 2 = 0) \Rightarrow_L esPrimo(s[i]))$ `}`

- todosIguales, que es verdadera sii todos los elementos de la secuencia son iguales.

s[0]	s[1]	$\cdots$	s[i]	$\cdots$	s[n-2]	s[n-1]

$\Rightarrow s[0] = s[1] \wedge s[1] = s[2] \wedge \cdots \wedge s[i] = s[i+1] \wedge \cdots \wedge$ $s[n-2] = s[n-1]$

`pred todosIguales(` $s{:}seq\langle Z\rangle$ `){` $(\forall i : Z)(0 \leq i < n - 1 \Rightarrow_L s[i] = s[i+1])$ `}`

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

s[0]	s[1]	$\cdots$	s[i]	$\cdots$	s[n-2]	s[n-1]

$\Rightarrow$ Existe un $s[i]$ que cumple $(s[i] \bmod 2 = 0)$ tal que $s[i] \mid s[0] \wedge$ $s[i] \mid s[i] \wedge \cdots \wedge s[i] \mid s[n-1]$

`pred hayUnoParQueDivideAlResto(` $s{:}seq\langle Z\rangle$ `){` $(\exists i : Z)(0 \leq i < n \ \wedge_L s[i] \bmod 2 = 0 \ \wedge_L (\forall j : Z)(0 \leq j < n \Rightarrow_L s[j] \bmod s[i] = 0))$ `}`

- hayUnoEnPosiciónParQueDivideAlResto, que determina si hay un elemento en una posición par de la secuencia que divide a todos los otros elementos contenidos en la secuencia.

s[0]	s[1]	$\cdots$	s[i]	$\cdots$	s[n-2]	s[n-1]

$\Rightarrow$ Existe un $s[i]$ en s tal que $i \bmod 2 = 0 \Rightarrow s[i] \mid s[0] \wedge$ $s[i] \mid s[1] \wedge \cdots \wedge s[i] \mid s[n-1]$

`pred hayUnoEnPosiciónParQueDivideAlResto(` $s{:}seq\langle Z\rangle$ `){` $(\exists i : Z)((0 \leq i < n \wedge i \bmod 2 = 0) \ \wedge_L (\forall j : Z)(0 \leq j < n \Rightarrow_L s[j] \bmod s[i] = 0))$ `}`

- sinRepetidos, que determina si la secuencia no tiene repetidos.

s[0]	s[1]	$\cdots$	s[i]	$\cdots$	s[n-2]	s[n-1]

$\Rightarrow \#cantidadDeApariciones(s[0]) = 1 \wedge$ $\#cantidadDeApariciones(s[1]) = 1 \wedge \cdots \wedge \#cantidadDeApariciones(s[n-1]) = 1$

`pred sinRepetidos(` $s{:}seq\langle Z\rangle$ `){` $(\forall i : Z)(0 \leq i < n \Rightarrow_L \#cantidadDeApariciones(s[i]) = 1)$ `}`

- otroMayorADerecha, que determina si todo elemento de la secuencia, salvo el último, tiene otro mayor a su derecha.

s[0]	s[1]	$\cdots$	s[i]	$\cdots$	s[n-2]	s[n-1]

$\Rightarrow s[0] < s[1] < \cdots < s[i] < s[i+1] < \cdots < s[n-1]$

`pred otroMayorADerecha(` $s{:}seq\langle Z\rangle$ `){` $(\forall i : Z)(0 \leq i < n - 1 \Rightarrow_L s[i] < s[i+1])$ `}`

- todoEsMúltiplo, que determina si todo elemento de la secuencia es múltiplo de algún otro.

s[0]	s[1]	$\cdots$	s[i]	$\cdots$	s[n-2]	s[n-1]

$\Rightarrow$ Para todo s[i] con $0 \leq i < n$ existe un $s[j]$ en s tal que $s[j] \bmod s[i] = 0$

`pred todoEsMúltiplo(` $s{:}seq\langle Z\rangle$ `){` $(\forall i : Z)(0 \leq i < n \Rightarrow_L (\exists j : Z)(0 \leq j < n \wedge_L s[j] \bmod s[i] = 0))$ `}`

¿Es válido este predicado?¿Por qué?

`pred todoEsMúltiplo(` $s{:}seq\langle Z\rangle$ `){` $(\forall s[i] : Z)(s[i] \in s \Rightarrow (\exists s[j] : Z)(s[j] \in s \wedge s[j] \bmod s[i] = 0))$ `}`

- enTresPartes, que determina si en la secuencia aparecen (de izquierda a derecha) primero 0s, después 1s y por último 2s

s[0]	s[1]	⋯	s[i]	⋯	s[j]	⋯	s[n-2]	s[n-1]

$\Rightarrow s[0] = s[1] = \cdots = s[i-1] = 0 \wedge$

$s[i] = s[i+1] = \cdots = s[j-1] = 1 \wedge s[j] = s[j+1] = \cdots = s[n-1] = 2$

`pred enTresPartes(` `s:`$seq\langle s \rangle$ `){` $|s| = 0 \vee ($

$(\forall i, j : Z)(0 \leq i < j < n \Rightarrow_L todosIgualesDesdeHastaValen(0, i, 0, s) \wedge$

$todosIgualesDesdeHastaValen(i, j, 1, s) \wedge todosIgualesDesdeHastaValen(j, n, 2, s))) \}$

`pred todosIgualesDesdeHastaValen(` `i:Z, j:Z, valor:Z, s:`$seq\langle Z \rangle$ `){`

$(\forall k : Z)(i \leq k < j \Rightarrow_L s[k] = valor) \}$

- esPermutaciónOrdenada, que dadas dos secuencias s y t sea verdadero si s es permutación de t y está ordenada.

  RPTA.¿Ordenada de que forma?

Ejercicio 5:
Especificar las siguientes funciones y predicados auxiliares. En caso de no ser posible, explicar las razones.

- intercambiarPrimeroPorUltimo, que intercambia el último valor por el primero en una secuencia

s[0]	s[1]	⋯	s[i]	⋯	s[n-2]	s[n-1]

$\Rightarrow$

s[n-1]	s[1]	⋯	s[i]	⋯	s[n-2]	s[0]

`aux intercambiarPrimeroPorUltimo`(s:$seq\langle Z \rangle$): $seq\langle Z \rangle$ {setAt(setAt(s,0,s[n-1])), n - 1, s[0]}

- esReverso,que indica si la secuencia s es el reverso de la secuencia t.

t:

t[0]	t[1]	⋯	t[i]	⋯	t[n-2]	t[n-1]

s:

t[n-1]	t[n-2]	⋯	t[i]	⋯	t[1]	t[0]

Entonces $s[0] = t[n-1] \wedge s[1] = t[n-2] \wedge \cdots \wedge s[n-1] = t[0]$

`pred esReverso(` `s:`$seq\langle Z \rangle$`,t:`$seq\langle Z \rangle$ `){` $|s| = |t| \wedge_L (\forall i : Z)(0 \leq i < n \Rightarrow_L s[i] = t[n - (i+1)]) \}$

- reverso, que indica el reverso de una secuencia.

s[0]	s[1]	⋯	s[i]	⋯	s[n-2]	s[n-1]

$\Rightarrow$

s[n-1]	s[n-2]	⋯	s[i]	⋯	s[1]	s[0]

$setAt(s, 0, s[n-1]) \Rightarrow setAt(setAt(s, 0, s[n-1]), 1, s[n-2]) \Rightarrow$

$setAt(setAt(setAt(s, 0, s[n-1]), 2, s[n-3]) \Rightarrow \cdots$

Ya que no conocemos la longitud de la secuencia y no podemos usar recursividad en el lenguaje de espicifación, no podemos formar una función auxiliar.

`aux reverso`(s:$seq\langle Z \rangle$): $seq\langle Z \rangle$ {/* No se puede :(*\}

- agregaTresCeros, que agrega 3 ceros al final de la secuencia s.

s[0]	s[1]	⋯	s[i]	⋯	s[n-2]	s[n-1]

++

0	0	0

`aux agregaTresCeros`(s:$seq\langle Z \rangle$): $seq\langle Z \rangle$ { s ++ $< 0, 0, 0 >$ }

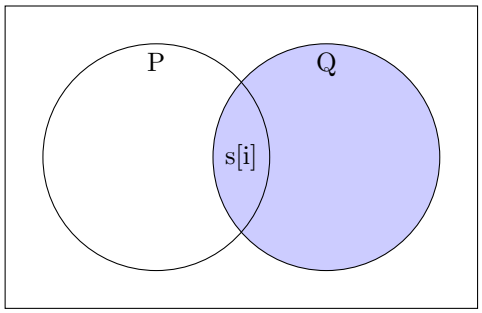
- agregarNCeros, que agrega n ceros al final de la secuencia s.

  No se puede, ya que no podemos usar recursividad en el lenguaje de especifación. Y para "crear" una secuencia de ceros, tarde o temprano caeremos en una recursividad. Al igual que f) y g), no podemos usar **un reemplazo sintáctico** con la herramientas que tenemos para resolver estás "acciones" sobre las secuencias o elementos de estas.
  Solo podemos hacer algo como en a), cuando conocemos a priori la cantidad de elementos en la lista y es un número razonable.

Ejercicio 6:
Sean $P(x : Z)$ y $Q(x : Z)$ dos predicados cualesquiera que nunca se indefinen y sea s una secuencia de enteros. Escribir el predicado asociado a cada uno de los siguientes enunciados:
`*Estos problemas los pensaremos con la noción de conjunto*`

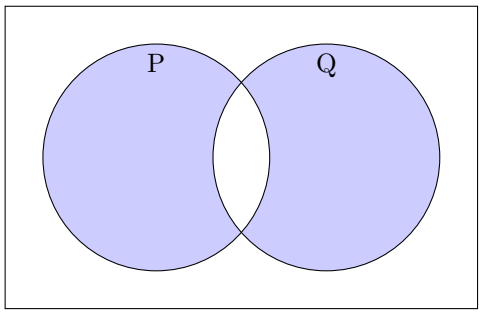
- "Si un entero en s cumple P, también cumple Q"

  Sean los conjuntos donde solo hay elementos en el area pintada, por definición si s[i] satisface las propiedades de P, entonces también satisface las propiedades de Q(**Nota:** Queria hacer un conjunto P contenido en uno Q pero no sabia dibujarlo).



  $(\forall s[i] : Z)(s[i] \in s \Rightarrow_L P) \Rightarrow (\forall s[j] : Z)(s[j] \in s \Rightarrow_L Q)$

- "Todos los enteros de s que cumplen P, no cumplen Q"



  $(\forall i : Z)(s[i] \in s \Rightarrow_L P) \Rightarrow (\exists s[j] : Z)(s[j] \in s \wedge_L \neg Q)$

Ejercicio 7:
Sea $P(x : Z)$ un predicado cualquiera y s una secuencia de enteros. Explicar cúal es el error de traducción a fórmulas de los siguientes enunciados. Dar un ejemplo en el cúal sucede el problema y luego corregirlo.

a) "Todo elemento en una posición válida de la secuencia cumple P":

  $(\forall i : Z)(0 \leq i < n \wedge_L P(s[i]))$.

  Si $0 \leq i < n \equiv Falso$, entonces $Falso \wedge_L P(s[i]) \equiv Falso$ ¡Siempre!

  Debo cuidar que ese enunciado no sea $Falso$, que pasa si probamos con el $\vee_L$