Algoritmos y Estructuras de Datos

Guía Práctica 2
Especificación de problemas
Primer Cuatrimestre 2025

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

2.1. Funciones auxiliares

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

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

> ***Solución***
>
> b)
> - En lenguaje natural, se debe cumplir que:
>   - y es primo
>   - y divide a x
>   - Cualquier otro primo p que divide a x es menor o igual a y
> - Formalmente `pred mayorPrimoQueDivide` (x, y: $\mathbb{Z}$) {
> $esPrimo(y) \land_L x \mod y = 0 \land (\forall p : \mathbb{Z})((esPrimo(p) \land_L x \mod p = 0) \rightarrow_L p \leq y)$
>   }

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

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

d) *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*)?

> ***Solución***
>
> c)
> - En lenguaje natural tenemos que decir que hay *algún elemento de la secuencia* que
>   - Es par
>   - Divide a todos los elementos de la secuencia
> - Para escribirlo formalmente tenemos dos formas de expresar *algún elemento de la secuencia*
>   - Predicando sobre los índices de la secuencia
>   `pred hayUnoParQueDivideAlResto` (s: $seq\langle\mathbb{Z}\rangle$) {
> $\Big(\exists i : \mathbb{Z}\Big)\Big(\big(0 \leq i < |s| \land_L esPar(s[i])\big) \land_L \big(\forall j : \mathbb{Z}\big)\big(0 \leq j < |s|\big) \rightarrow_L divideA(s[i], s[j])\big)\Big)$
>   }
>   - Predicando sobre los elementos que *pertenecen* a la secuencia
>   `pred hayUnoParQueDivideAlResto` (s: $seq\langle\mathbb{Z}\rangle$) {
> $\Big(\exists n : \mathbb{Z}\Big)\Big(\big(n \in s \land esPar(n)\big) \land \big(\forall m : \mathbb{Z}\big)\big(m \in s \rightarrow divideA(n, m)\big)\Big)$

}

Nota: Para este problema las dos opciones son equivalentes, pero no siempre se puede usar la opción de $\in$.

- Falta definir los predicados auxiliares

  `pred esPar` (n: $\mathbb{Z}$) {

  $n \mod 2 = 0$

  }

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

  $n \neq 0 \wedge_L m \mod n = 0$

  }

d) `pred enTresPartes` (s: $seq\langle\mathbb{Z}\rangle$) {

$estáOrdenada(s) \wedge sóloCeroUnoYDos(s)$

}

donde

`pred sóloCeroUnoYDos` (s: $seq\langle\mathbb{Z}\rangle$) {

$(\forall i : \mathbb{Z})(0 \leq i < |s|) \rightarrow_L (s[i] = 0 \vee s[i] = 1 \vee s[i] = 2)$

}

Ejercicio 5. Sea s una secuencia de elementos de tipo $\mathbb{Z}$. Escribir una expresión (utilizando sumatoria y productoria) tal que:

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

Solución

b) $sumaPosImpares = \sum_{i=0}^{|s|-1} \mathsf{IfThenElse}(i \mod 2 \neq 0, s[i], 0)$

2.2. Análisis de especificación

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

$$f(a,b) = \begin{cases} 2b & \text{si } a < 0 \\ b-1 & \text{en otro caso} \end{cases}$$

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` $\{(a < 0 \wedge res = 2 * b) \wedge (a \geq 0 \wedge res = b - 1)\}$

}

b) `proc f` (in a, b: $\mathbb{R}$) : $\mathbb{R}$ {

`requiere` $\{True\}$

`asegura` $\{(a < 0 \wedge res = 2 * b) \vee (a \geq 0 \wedge res = b - 1)\}$

}

c) `proc f` (in a, b: $\mathbb{R}$) : $\mathbb{R}$ {

  `requiere` $\{True\}$

  `asegura` $\{(a < 0 \rightarrow res = 2 * b) \vee (a \geq 0 \rightarrow res = b - 1)\}$

}

d) `proc f` (in a, b: $\mathbb{R}$) : $\mathbb{R}$ {

  `requiere` $\{True\}$

  `asegura` $\{res = \mathsf{IfThenElse}(a < 0, 2 * b, b - 1)\}$

}

Solución

a) Es **incorrecta**. No se cumple nunca. Se puede ver que es equivalente a $(P(a) \wedge Q(b)) \wedge (\neg P(a) \wedge R(b)) \equiv (P(a) \wedge \neg P(a)) \wedge Q(b) \wedge R(b)$ que es una contradicción.

b) Es **correcta**. Es la forma de desglosar un condicional (if).

c) Es **incorrecta**. Es siempre verdadera. Cuando el antecedente de uno es falso la implicación es verdadera y es trivial notar que a es $a < 0$ o $a \geq 0$.

d) Es **correcta**. Evalúa una condición que depende de a. La rama verdadera $(a < 0)$ retorna $2 * b$, mientras que la rama falsa $(a \geq 0)$ retorna $b - 1$. En ambos casos se asigna un término a la variable res de salida que coincide con lo mostrado para la función partida f.

2.3. Relación de fuerza

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` $\{x \neq 0\}$
  `asegura` $\{x^n - 1 < res \leq x^n\}$
}

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

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`?

Solución

a) El consecuente de la precondición de `p2` es la pre de `p1`, por lo tanto, sin importar el valor de n, si la pre de `p1` es verdadera entonces también lo es la de `p2`

   También podemos pensarlo como dos casos. $n \leq 0$ y $n > 0$. La precondición de `p2` requiere que en el primer caso $x \neq 0$. Basados en la precondición de `p1`, vemos que esto se cumple para ese caso como para cuando $n > 0$.

b) Sí, $x^n - 1 < \lfloor x^n \rfloor \leq x^n$ por la definición de $\lfloor x \rfloor$.

c) Sí, ya que cumple con las condiciones planteadas en el ítem **8.d**.

2.4. Especificación de problemas

Ejercicio 12. Especificar los siguientes problemas:

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

Solución

c) `proc` `divisoresPositivos` (in n: $\mathbb{Z}$) : $seq\langle\mathbb{Z}\rangle$ {

   `requiere` $\{True\}$

   `asegura` {

   $sinRepetidos(res) \wedge$

   $(\forall d : \mathbb{Z})((d > 0 \wedge_L n \mod d = 0) \rightarrow_L d \in res) \wedge$

   $(\forall r : \mathbb{Z})(r \in res \rightarrow_L (r > 0 \wedge_L n \mod r = 0))$

   }

}

Ejercicio 13. Especificar los siguientes problemas sobre secuencias:

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

Solución

b) `proc interseccion` (in s,t: $seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{Z}\rangle$ {

  `requiere` $\{True\}$

  `asegura` {
    $(\forall r : \mathbb{Z})(r \in res \leftrightarrow (r \in s \land r \in t)) \land$
    $(\forall e : \mathbb{Z})(e \in res \rightarrow_L (\#apariciones(e, res) = min(\#apariciones(e, s), \#apariciones(e, t))))$
  }

}

c) `proc elQueMasDivide` (in s: $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z}$ {

  `requiere` $\{True\}$

  `asegura` $\{res \in s \land (\forall e : \mathbb{Z})(e \in s \rightarrow (divisiblesPor(e, s) \leq divisiblesPor(res, s)))\}$

  `aux divisiblesPor` (n: $\mathbb{Z}$, s: $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z}$ {

    $\sum_0^{|s|-1} \mathsf{IfThenElse}(divideA(n, s[i]), 1, 0)$

  }

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

    $n \neq 0 \land_L m \mod n = 0$

  }

}

2.5. Especificación de problemas usando **inout**

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`$(l)$, $l = \langle 5, 3, 7, 13\rangle$

Solución

a) `proc primosHermanos` (inout l: $seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{Z}\rangle$ {

  `requiere` $\{l = L_0 \land (\forall i : \mathbb{Z})((0 \leq i < |l|) \rightarrow (l[i] > 2))\}$

  `asegura` {
    $|l| = |L_0| \land_L$
    $(\forall i : \mathbb{Z})((0 \leq i < |l|) \rightarrow esPrimoMasCercano(l[i], L_0[i]))$
  }

  `pred esPrimoMasCercano` (p, n) {

    $esPrimo(p) \land p < n \land (\forall q : \mathbb{Z})((esPrimo(q) \land q < n) \rightarrow (q \leq p))$

  }

}

2.6. Ejercicios de parciales anteriores

Ejercicio 18. Especificar los siguientes problemas. En todos los casos es recomendable ayudarse escribiendo predicados y funciones auxiliares.

d) Se desea especificar el problema *positivosAumentados* que dada una secuencia s de enteros devuelve la secuencia pero con los valores positivos reemplazados por su valor multiplicado por la posición en que se encuentra.

- $positivosAumentados([0, 1, 2, 3, 4, 5]) = [0, 1, 4, 9, 16, 25]$
- $positivosAumentados([-2, -1, 5, 3, 0, -4, 7]) = [-2, -1, 10, 9, 0, -4, 42]$

e) Se desea especificar el problema `procesarPrefijos` que dada una secuencia s de palabras y una palabra p, remueve todas las palabras de s que no tengan como prefijo a p y además retorna la longitud de la palabra más larga que tiene de prefijo a p. Por ejemplo, dados: `s = ["casa", "calamar", "banco", "recuperatorio", "aprobar", "cansado"]` y `p = "ca"` un posible valor para la secuencia s luego de aplicar `procesarPrefijos(s, p)` puede ser `["casa", "calamar", "cansado"]` y el valor devuelto será `7`.

Solución

d) `proc positivosAumentados` (in s: $seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{Z}\rangle$ {

`requiere` $\{true\}$

`asegura` {

$|res| = |s| \land_L ($

$positivosCambiados(s, res) \land$

$negativosIguales(s, res)$

$)$

}

`pred positivosCambiados` (s: $seq\langle\mathbb{Z}\rangle$, res: $seq\langle\mathbb{Z}\rangle$) {

$(\forall i : \mathbb{Z})((0 \leq i < |s| \land_L s[i] > 0) \to_L res[i] = s[i] * i)$

}

`pred negativosIguales` (s: $seq\langle\mathbb{Z}\rangle$, res: $seq\langle\mathbb{Z}\rangle$) {

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

}

}

e) `proc procesarPrefijos` (inout s: $seq\langle string\rangle$, in p: string) : $\mathbb{Z}$ {

`requiere` $\{s = S_0\}$

`asegura` {

$(\forall r : string)(r \in s \leftrightarrow (r \in S_0 \land \neg esPrefijo(r, p))) \land$

$maxTamañoConPrefijo(S_0, p, res)$

}

`pred maxTamañoConPrefijo` (s: $seq\langle string\rangle$, p: string, m: $\mathbb{Z}$) {

$(\exists r : string)(r \in s \land esPrefijo(r, p) \land |r| = m) \land$

$(\forall r : string)(r \in s \land esPrefijo(r, p) \to |r| \leq m)$

}

}