DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 1: Especificación y Correctitud

Subtítulo del tp

12 de septiembre de 2024 — Algoritmos y Estructuras de Datos

Grupo ElDobleMenosUno

Integrante	LU	Correo electrónico
Deukmedjian, Iván	001/01	email1@dominio.com
Stescovich Curi, Agustín Ezequieñ	184/24	agustinstescovichcuri@gmail.com
Feito, Agustín	236/24	agustinfeito@hotmail.com
Raffo, Pedro	004/01	email4@dominio.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

1. Especificación

1.1. grandesCiudades

`proc grandesCiudades` (in $ciudades : seq\langle \mathsf{Ciudad} \rangle$)
  `requiere` $\{\mathsf{noHayNombresRepetidos}(ciudades) \wedge \mathsf{todasPoblacionesPositivas}(ciudades)\}$
  `asegura` $\{((c \in ciudades \wedge c_1 > 50000) \leftrightarrow c \in res) \wedge \mathsf{noHayNombresRepetidos}(res)\}$

`pred noHayNombresRepetidos` $(s : seq\langle \mathsf{Ciudad} \rangle)$ $\{(\forall i : \mathbb{Z})(0 \leq i < |s| \rightarrow_L \neg(\exists j : \mathbb{Z})(0 \leq j < |s| \wedge_L s \neq j \wedge s[i]_0 = s[j]_0))\}$

`pred todasPoblacionesPositivas` $(s : seq\langle \mathsf{Ciudad} \rangle)$ $\{(\forall i : \mathbb{Z})(0 \leq i < |s| \rightarrow_L s[i]_1 \geq 0)\}$

1.2. sumaDeHabitantes

Para una mejor legibilidad de la especificacion se sustituye a menoresDeCiudades por men y a mayoresDeCiudades por may, es decir:
$menoresDeCiudades = men$ y $mayoresDeCiudades = may$

`proc sumaDeHabitantes` (in men, may :$seq\langle Ciudad \rangle$) : $seq\langle Ciudad \rangle$
  `requiere` $\{noHayRepetidos(men) \wedge todasPoblacionesPositivas(men)\}$
  `requiere` $\{noHayRepetidos(may) \wedge todasPoblacionesPositivas(may)\}$
  `requiere` $\{mismasCiudades(may, men)$
  }
  `asegura` $\{mismasCiudades(res, men)\}$
  `asegura` $\{(\forall i, j : \mathbb{Z})$ $((0 \leq i < |men| \wedge 0 \leq j < |may| \wedge_L men[i]_0 = may[j]_0) \rightarrow_L$
  $((\exists k : \mathbb{Z})$ $(0 \leq |res| \wedge_L res[k]_0 = men[i]_0 \wedge_L men[i]_1 + may[j]_1 = res[k]_1)))$
  }

`pred comparteTodosLosElem` (in c_1, c_2:$seq\langle Ciudad \rangle$) {
  $(\forall i : \mathbb{Z})$ $(0 \leq i < |c_1| \rightarrow_L (\exists j : \mathbb{Z})$ $(0 \leq j < |c_2| \wedge_L c_1[i]_0 = c_2[j]_0))$
}
`pred mismasCiudades` (in c_1, c_2:$seq\langle Ciudad \rangle$) {
  $comparteTodosLosElem(c_1, c_2) \wedge_L comparteTodosLosElem(c_2, c_1)$
}

1.3. hayCamino

`proc hayCamino` (in $distancias : seq\langle seq\langle \mathbb{Z} \rangle \rangle$, in $desde, hasta : \mathbb{Z}$) : Bool
  `requiere` $\{\mathsf{esMatrizCuadrada}(distancias) \wedge \mathsf{esMatrizDiagonal}(distancias)\}$
  `requiere` $\{\mathsf{ningunElementoNegativo}(distancias) \wedge 0 \leq desde, hasta < |distancias|\}$
  `asegura` $\{res = \mathtt{true} \leftrightarrow (\exists i : seq\langle \mathbb{Z} \rangle)(\mathsf{esCaminoValido}(i, desde, hasta, distancias)\}$

`pred esMatrizCuadrada` $(m : seq\langle seq\langle \mathbb{Z} \rangle \rangle)$ {
  $(\forall i : \mathbb{Z})(0 \leq i < |m| \rightarrow_L |m| = |m[i]|)\}$

`pred esMatrizDiagonal` $(m : seq\langle seq\langle \mathbb{Z} \rangle \rangle)$ {
  $(\forall i, j : \mathbb{Z})(0 \leq i, j < |m| \rightarrow_L m[i][j] = m[j][i])\}$

`pred ningunElementoNegativo` $(s : seq\langle seq\langle \mathbb{Z} \rangle \rangle)$ {
  $(\forall i, j : \mathbb{Z})(0 \leq i, j < |distancias| \rightarrow_L distancias[i][j] \geq 0)\}$

`pred esCaminoValido` $(camino : seq\langle \mathbb{Z} \rangle, origen : \mathbb{Z}, destino : \mathbb{Z}, m : seq\langle \langle \mathbb{Z} \rangle \rangle)$ {
  $|camino| \geq 2 \wedge_L camino[0] = origen \wedge camino[|camino| - 1] = destino$
  $\wedge(\forall i : \mathbb{Z})(0 \leq i < |camino| - 1 \rightarrow_L (camino[i] \neq camino[i+1] \rightarrow m[i][i+1] > 0))\}$

1.4. cantidadCaminosNSaltos

`proc cantidadCaminosNSaltos` (inout conexion : $seq\langle seq\langle \mathbb{Z} \rangle \rangle$,in n : $\mathbb{Z}$)
  `requiere` $\{(\forall i, j : \mathbb{Z})$ $(0 \leq i, j < |conexion| \rightarrow_L 0 \leq conexion[i][j] \leq 1)\}$
  `requiere` $\{esMatrizcuadrada(conexion) \wedge conexion = C_0 \wedge n \geq 1\}$
  `asegura` $\{esMatrizN - Esima(conexion, c_0, n)$
  }

`pred esMatrizN-Esima` $(c, c_0 : seq\langle seq\langle \mathbb{Z} \rangle, n : \mathbb{Z} \rangle)$ {
  $(\exists s : seq\langle seq\langle seq\langle \mathbb{Z} \rangle \rangle \rangle)$ $(|s| = n \ \wedge s[0] = c_0 \ \wedge s[n-1] = c \ \wedge_L$
  $(\forall i : \mathbb{Z})$ $(1 \leq i < n \rightarrow_L esProducto(s[i], s[i-1], s[0])))$

}
pred esProducto (in t, s_1, s_2: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i, j : \mathbb{Z})\ (0 \leq i, j < |t| \rightarrow_L t[i][j] = \sum_{k=0}^{|t|-1} s_1[i][k] * s_2[k][j])$

}

1.5. caminoMinimo

proc caminoMinimo $(origen : \mathbb{Z}, destino : \mathbb{Z}, distancias : seq\langle\langle \mathbb{Z}\rangle\rangle) : seq\langle \mathbb{Z}\rangle$ {
requiere $\{0 \leq origen, destino < |distancias| \wedge \text{esMatrizCuadrada}(distancias) \wedge \text{esMatrizDiagonal}(distancias)\}$
asegura $\{res = \langle\rangle \leftrightarrow \neg(\exists i : seq\langle \mathbb{Z}\rangle)(\text{esCaminoValido}(i, origen, destino, distancias)\}$
asegura $\{res \neq \langle\rangle \leftrightarrow \text{esCaminoMaximo}(res, origen, destino, distancias)\}$

aux distanciaCamino $(camino : seq\langle \mathbb{Z}\rangle, m : seq\langle\langle \mathbb{Z}\rangle\rangle) = \sum_{i=0}^{|camino|-1} m[i][i+1]$

pred esCaminoMaximo $(camino : seq\langle \mathbb{Z}\rangle, origen : \mathbb{Z}, destino : \mathbb{Z}, m : seq\langle\langle \mathbb{Z}\rangle\rangle)$ {
$\text{esCaminoValido}(res, origen, destino, m) \wedge_L$
$(\forall i : seq\langle \mathbb{Z}\rangle)(\text{esCaminoValido}(i, origen, destino, m) \rightarrow_L \text{distanciaCamino}(i, m) \leq \text{distanciaCamino}(res, m))\}$

2. Correctitud

2.1. Demostración de correctitud de la implementación de **poblaciónTotal**

2.1.1. Teorema del invariante

(hay que pasar las cosas de test.tex acá)

2.1.2. Teorema de terminación

Proponemos la función variante $fv = |s| - i$ y buscamos probar que el ciclo satisface el teorema de terminación.

Primero, verifiquemos que la tripla $\{I \wedge B \wedge fv = v_0\}s\{fv < v_0\}$ sea válida.
Para ello, debemos calcular la precondición más débil del ciclo respecto de $fv < v_0$:

$$\begin{aligned} &wp(\texttt{res} := \texttt{res} + \texttt{ciudades}[\texttt{i}].\texttt{habitantes}; \texttt{i} := \texttt{i} + 1, fv < v_0) \\ \equiv &wp(\texttt{res} := \texttt{res} + \texttt{ciudades}[\texttt{i}].\texttt{habitantes}, (wp(\texttt{i} := \texttt{i} + 1, fv < v_0)) \end{aligned} \tag{1}$$

Calculemos primero la WP anidada:

$$\begin{aligned} &wp(\texttt{i} := \texttt{i} + 1, fv < v_0) \\ \equiv &wp(\texttt{i} := \texttt{i} + 1, |ciudades| - i < v_0) \\ \equiv &def(i+1) \wedge_L Q^i_{i+1} \\ \equiv &True \wedge_L |ciudades| - i - 1 < v_0 \\ \equiv &|ciudades| - i - 1 < v_0 \end{aligned} \tag{2}$$

Usando esto,

$$\begin{aligned} &wp(\texttt{res} := \texttt{res} + \texttt{ciudades}[\texttt{i}].\texttt{habitantes}, (wp(\texttt{i} := \texttt{i} + 1, fv < v_0)) \\ \equiv &wp(\texttt{res} := \texttt{res} + \texttt{ciudades}[\texttt{i}].\texttt{habitantes}, |ciudades| - i - 1 < v_0) \\ \equiv &def(res + ciudades[i].habitantes) \wedge_L Q^{res}_{res+ciudades[i].habitantes} \\ \equiv &0 \leq i < |ciudades| \wedge_L |s| - i - 1 < v_0 \end{aligned} \tag{3}$$

Dada la precondición más débil, comprobamos ahora que $I \wedge B \wedge fv = v_0$ la fuerza:

$$\begin{aligned} &I \wedge B \wedge fv = v_0 \\ \equiv &0 \leq i \leq |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge i < |ciudades| \wedge |ciudades| - i = v_0 \end{aligned} \tag{4}$$

Basta ver que $|ciudades| - i = v_0 \rightarrow |ciudades| - i - 1 < v_0$, pues $|ciudades| - i - 1 < |ciudades| - i$, o bien $-1 < 0$. Por lo tanto, la tripla de Hoare $\{I \wedge B \wedge fv = v_0\}\texttt{s}\{fv < v_0\}$ es válida.

Ahora, debemos mostrar que $(I \wedge fv \leq 0) \rightarrow \neg B$, es decir, que la guarda del ciclo deja de valer cuando la función variante es menor o igual a 0.

$$I \wedge fv \leq 0 \equiv 0 \leq i \leq |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge |ciudades| - i < 0 \tag{5}$$

Tenemos que...

$$\begin{aligned} &|ciudades| - i < 0 \wedge 0 \leq i \leq |ciudades| \\ &\rightarrow i = |ciudades| \\ &\rightarrow \neg(i < |ciudades|) \end{aligned} \tag{6}$$

...que es la implicación que buscábamos probar.

2.1.3. Finalización de la demostración

Sabemos que el ciclo del programa es correcto respecto de (P_c, Q_c), pero falta demostrar la validez de $\{P\}\texttt{res} := \texttt{0}; \texttt{i} := \texttt{0}\{P_c\}$, es decir, que se llegue siempre a la precondición del ciclo partiendo de la especificación.

Comenzamos calculando la precondición más debil de las instrucciones anteriores al ciclo respecto de P_c:

$$\begin{aligned} &wp(\texttt{res} := \texttt{0}; \texttt{i} := \texttt{0}, P_c) \\ \equiv &wp(\texttt{res} := \texttt{0}, wp(\texttt{i} := \texttt{0}, P_c)) \end{aligned} \tag{7}$$

(NOTA: para ahorrar espacio, recordamos que $P_c \equiv P \wedge res = 0 \wedge i = 0$.)

$$\begin{aligned} &wp(\texttt{i} := \texttt{0}, P_c)) \\ \equiv &True \wedge_L P \wedge res = 0 \wedge 0 = 0 \\ \equiv &P \wedge res = 0 \end{aligned} \tag{8}$$

Entonces,

$$\begin{aligned} &wp(\texttt{res} := \texttt{0}, wp(\texttt{i} := \texttt{0}, P_c)) \\ \equiv &wp(\texttt{res} := \texttt{0}, P \wedge res = 0) \\ \equiv &P \wedge 0 = 0 \\ \equiv &P \end{aligned} \tag{9}$$

Se sigue trivialmente que $P \rightarrow P$, y por lo tanto la tripla $\{P\}\texttt{res} := \texttt{0}; \texttt{i} := \texttt{0}\{P_c\}$ es correcta.
Luego, por monotonía, tenemos que $\{P\}\texttt{res} := \texttt{0}; \texttt{i} := \texttt{0}\{P_c\} \wedge \{P_c\}\texttt{s}\{Q\} \rightarrow \{P\}\texttt{res} := \texttt{0}; \texttt{i} := \texttt{0}; \texttt{s}\{Q\}$. Así, hemos demostrado que la implementación de poblaciónTotal es correcta respecto de su especificación.

2.2. ¿El valor devuelto por el programa es mayor a 50.000?

Demostramos anteriormente que la implementación s es correcta respecto de la especificación, y por tanto basta ver que la especificación cumple lo pedido.
Obtenemos de la precondición del programa que:

- Sabemos que dado un i entero tal que $0 \leq i < |ciudades|$ existe al menos un elemento de *ciudades* tal que ciudades[i].habitantes es mayor a 50,000 (por lo que hay al menos una segunda componente de un elemento en *ciudades* con un valor mayor o igual a 50.000),
- Ninguna segunda componente de un elemento en *ciudades* es negativa

y de la postcondición que:

- El resultado obtenido es la sumatoria de toda segunda componenete (habitantes) de un elemento en *ciudades*.

Por tanto, dadas estas condiciones, se garantiza que $\sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes$ sea mayor a 50,000.