DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 1

Especificación y WP

October 9, 2024 — Algoritmos y Estructuras de Datos

Grupo ElDobleMenosUno

Integrante	LU	Correo electrónico
Deukmedjian, Iván	521/24	`deukivan@gmail.com`
Stescovich Curi, Agustín Ezequiel	184/24	`agustinstescovichcuri@gmail.com`
Feito, Agustín	236/24	`agustinfeito@hotmail.com`
Raffo, Pedro	168/24	`pedroraffo25@gmail.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 : $seq\langle \textsf{Ciudad}\rangle$) : $seq\langle \textsf{Ciudad}\rangle$

  `requiere` $\{\textsf{noHayNombresRepetidos}(ciudades) \land \textsf{todasPoblacionesPositivas}(ciudades)\}$

  `asegura` $\{\textsf{noHayNombresRepetidos}(res) \land (\forall c : \textsf{Ciudad})\ ((c \in ciudades \land c.habitantes > 50000) \leftrightarrow c \in res)\}$

`pred` `noHayNombresRepetidos` (s : $seq\langle \textsf{Ciudad}\rangle$) {

  $(\forall i : \mathbb{Z})\ (0 \leq i < |s| \longrightarrow_L \neg(\exists j : \mathbb{Z})\ (0 \leq j < |s| \land_L i \neq j \land s[i].nombre = s[j].nombre))$

}

`pred` `todasPoblacionesPositivas` (s : $seq\langle \textsf{Ciudad}\rangle$) {

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

}

1.2 sumaDeHabitantes

(NOTA 1: Para una mejor legibilidad de la especificación, se sustituye a $menoresDeCiudades$ por men y a $mayoresDeCiudades$ por may, es decir: $menoresDeCiudades = men$ y $mayoresDeCiudades = may$.)
(NOTA 2: Reusamos aquí los predicados noHayNombresRepetidos y todasPoblacionesPositivas.)

`proc` `sumaDeHabitantes` (in men, may : $seq\langle \textsf{Ciudad}\rangle$) : $seq\langle \textsf{Ciudad}\rangle$

  `requiere` $\{\textsf{noHayNombresRepetidos}(men) \land \textsf{noHayNombresRepetidos}(men) \land \textsf{mismasCiudades}(may,\ men)\}$

  `requiere` $\{\textsf{todasPoblacionesPositivas}(may) \land \textsf{todasPoblacionesPositivas}(may)\}$

  `asegura` $\{\textsf{noHayNombresRepetidos}(res) \land \textsf{mismasCiudades}(res, men)\}$

  `asegura` $\{(\forall i, j : \mathbb{Z})\ ((0 \leq i < |men| \land 0 \leq j < |may| \land_L men[i].nombre = may[j].nombre) \longrightarrow_L$
   $(\exists k : \mathbb{Z})\ (0 \leq k < |res| \land_L res[k].nombre = men[i].nombre$
   $\land_L men[i].habitantes + may[j].habitantes = res[k].habitantes))\}$

`pred` `mismasCiudades` (c1, c2 : $seq\langle \textsf{Ciudad}\rangle$) {

  $|c1| = |c2| \land_L (\forall i : \mathbb{Z})\ (0 \leq i < |c1| \longrightarrow_L (\exists j : \mathbb{Z})\ (0 \leq j < |c2| \land_L c1[i].nombre = c2[j].nombre))$

}

1.3 hayCamino

`proc` `hayCamino` (in distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, in desde, hasta : $\mathbb{Z}$) : Bool

  `requiere` $\{\textsf{esMatrizCuadrada}(distancias) \land \textsf{esMatrizSimetrica}(distancias)\}$

  `requiere` $\{\textsf{ningunElementoNegativo}(distancias) \land 0 \leq desde, hasta < |distancias|\}$

  `asegura` $\{res = \text{true} \leftrightarrow (\exists i : seq\langle \mathbb{Z}\rangle)\ (\textsf{esCaminoValido}(i, desde, hasta, distancias))\}$

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

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

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

`pred` `esCaminoValido` (camino : $seq\langle \mathbb{Z}\rangle$, origen, destino : $\mathbb{Z}$, m : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

  $|camino| \geq 2 \land_L (camino[0] = origen \land camino[|camino| - 1] = destino) \land_L$
  $(\forall i : \mathbb{Z})\ (0 \leq i < |camino| - 1 \longrightarrow_L m[camino[i]][camino[i+1]] > 0)$

}

1.4 cantidadCaminosNSaltos

(NOTA: Reusamos aquí los predicados esMatrizCuadrada y esMatrizSimetrica.)

proc cantidadCaminosNSaltos (inout conexion : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, in n : $\mathbb{Z}$)

requiere $\{conexion = C_0 \land \text{esMatrizCuadrada}(conexion) \land \text{esMatrizSimetrica}(conexion) \land n \geq 1\}$

requiere $\{(\forall i, j : \mathbb{Z})\ (0 \leq i, j < |conexion| \longrightarrow_L 0 \leq conexion[i][j] \leq 1)\}$

asegura $\{\text{esMatrizCuadrada}(conexion) \land \text{esMatrizSimetrica}(conexion) \land \text{esNEsimaPotencia}(conexion,\ C_0,\ n)\}$

pred esNEsimaPotencia (matriz, base : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, exp : $\mathbb{Z}$) {

$(\exists s : seq\langle seq\langle seq\langle \mathbb{Z}\rangle\rangle\rangle)\ (|s| = exp \land_L s[0] = base \land s[exp-1] = matriz \land_L$
$(\forall i : \mathbb{Z})\ (0 \leq i < exp \longrightarrow_L |s[i]| = |s[0]| \land \text{esMatrizCuadrada}(s[i]) \land_L$
$(i \neq 0 \longrightarrow_L \text{esProducto}(s[i],\ s[i-1],\ s[0]))))$

}

pred esProducto (matriz, factor1, factor2 : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i, j : \mathbb{Z})\ (0 \leq i, j < |matriz| \longrightarrow_L matriz[i][j] = \sum_{k=0}^{|matriz|-1} factor1[i][k] * factor2[k][j])$

}

1.5 caminoMinimo

(NOTA: Reusamos aquí los predicados esMatrizCuadrada, esMatrizSimetrica y ningunElementoNegativo.)

proc caminoMinimo (origen, destino : $\mathbb{Z}$, distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) : $seq\langle \mathbb{Z}\rangle$

requiere $\{\text{esMatrizCuadrada}(distancias) \land \text{esMatrizSimetrica}(distancias)\}$

requiere $\{\text{ningunElementoNegativo}(distancias) \land 0 \leq origen,\ destino < |distancias|\}$

asegura $\{res = \langle\rangle \leftrightarrow \neg(\exists i : seq\langle \mathbb{Z}\rangle)\ (\text{esCaminoValido}(i,\ origen,\ destino,\ distancias))\}$

asegura $\{res \neq \langle\rangle \leftrightarrow \text{esCaminoMinimo}(res,\ origen,\ destino,\ distancias)\}$

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

pred esCaminoMinimo (camino : $seq\langle \mathbb{Z}\rangle$, origen, destino : $\mathbb{Z}$, m : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$\text{esCaminoValido}(res,\ origen,\ destino,\ m) \land_L (\forall i : seq\langle \mathbb{Z}\rangle)$ (
$\text{esCaminoValido}(i,\ origen,\ destino,\ m) \longrightarrow_L \text{distanciaCamino}(i,\ m) \geq \text{distanciaCamino}(res,\ m))$

}

2 Correctitud

2.1 Demostración de correctitud de la implementación de poblaciónTotal

Esta es la tripla de Hoare a probar:

$$\begin{aligned}&\mathbf{P} \equiv \{(\exists i : \mathbb{Z})(0 \leq i < |ciudades| \wedge_L ciudades[i].habitantes > 50{,}000) \wedge \\ &(\forall i : \mathbb{Z})(0 \leq i < |ciudades| \longrightarrow_L ciudades[i].habitantes \geq 0) \wedge \\ &(\forall i : \mathbb{Z})(\forall j \mathbb{Z})(0 \leq i < j < |ciudades| \longrightarrow_L ciudades[i].nombre \neq ciudades[j].nombre)\}\end{aligned}$$

```
res := 0;
i := 0;
while (i < ciudades.length) do
    res = res + ciudades[i].habitantes; Nuestro S1 en el ciclo
    i := i + 1; Nuestro S2 en el ciclo
endwhile
```

$$\mathbf{Q} \equiv \{res = \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes\}$$

2.1.1 Teorema del invariante

Proponemos el siguiente invariante y demostramos la correctitud parcial del ciclo:

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

- **Condición 1: $\mathbf{P_c \Rightarrow I}$**

$$\begin{aligned}P_c \equiv\ &(\exists i : \mathbb{Z})(0 \leq i < |ciudades| \wedge_L ciudades[i].habitantes > 50{,}000)\ \wedge \\ &(\forall i : \mathbb{Z})(0 \leq i < |ciudades| \longrightarrow_L ciudades[i].habitantes \geq 0)\ \wedge \\ &(\forall i : \mathbb{Z})(\forall j : \mathbb{Z})(0 \leq i < j < |ciudades| \longrightarrow_L ciudades[i].nombre \neq ciudades[j].nombre)\ \wedge \\ &res = 0 \wedge i = 0 \Longrightarrow (0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes)\end{aligned}$$

Utilizamos que $P_c \Rightarrow res = 0 \wedge i = 0$ y reemplazamos en el invariante:

$$\begin{aligned}&0 \leq 0 \leq |ciudades| \wedge res = \sum_{j=0}^{-1} Ciudades[j].habitantes \\ &\equiv True \wedge res = 0 \\ &\equiv res = 0\end{aligned}$$

Se evidencia que $P_c \Rightarrow res = 0 \wedge i = 0 \Rightarrow res = 0$, y $\therefore$ se cumple que el antecedente implica al consecuente.

- **Condición 2: validez de la tripla $\{\mathbf{I \wedge B}\}$ S $\{\mathbf{I}\}$**

Comenzamos resolviendo la WP:

$$\mathbf{wp(\mathtt{S}, I)} \equiv wp(\mathtt{S1;\ S2},\ I) \tag{1}$$

$$\equiv wp(\mathtt{S1},\ wp(\mathtt{S2},\ I)) \tag{2}$$

$$\equiv wp(\mathtt{S1},\ wp(\mathtt{i := i + 1},\ (0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes)) \tag{3}$$

$$\equiv wp(\mathtt{S1},\ 0 \leq i+1 \leq |ciudades| \wedge res = \sum_{j=0}^{i} ciudades[j].habitantes) \tag{4}$$

$$\equiv (0 \leq i+1 \leq |ciudades| \wedge 0 \leq i < |ciudades| \wedge \tag{5}$$

$$res + ciudades[i].habitantes = (\sum_{j=0}^{i-1} ciudades[j].habitantes) + ciudades[i].habitantes \tag{6}$$

$$\equiv 0 \leq i < |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes \tag{7}$$

Ahora vemos la implicación $(\mathbf{I} \wedge \mathbf{B}) \Rightarrow \mathbf{wp}(\mathbf{S}, \mathbf{I})$:

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

$$\equiv 0 \leq i < |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes \tag{2}$$

$$\Rightarrow 0 \leq i < |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes$$

Esto se cumple dado que es una tautología del tipo $p \Rightarrow p$.

- **Condición 3: $\mathbf{I} \wedge \neg\mathbf{B} \Rightarrow \mathbf{Q_c}$**

$$I \wedge \neg B$$

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

$$\equiv i = |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \tag{2}$$

$$\equiv i = |ciudades| \wedge_L res = \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes \tag{3}$$

$$\Rightarrow res = \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes$$

$$\equiv True$$

... pues $p \wedge q \Rightarrow q$ es tautología.

2.1.2 Teorema de terminación

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

- **Condición 4: validez de la tripla $\{\mathbf{I} \wedge \mathbf{B} \wedge \mathbf{fv} = \mathbf{v_0}\}$ s $\{\mathbf{fv} < \mathbf{v_0}\}$**

Primero, debemos calcular la precondición más débil del ciclo respecto de $fv < v_0$:

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

Calculemos primero la WP anidada:

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

Usando esto,

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

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

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

Basta ver que $|ciudades| - i = v_0 \Rightarrow |ciudades| - i - 1 < v_0$, pues

$$
\begin{aligned}
& |ciudades| - i - 1 < |ciudades| - i \\
\equiv\ & -1 < 0 \\
\equiv\ & True
\end{aligned}
$$

Por lo tanto, la tripla de Hoare $\{I \wedge B \wedge fv = v_0\}$ `s` $\{fv < v_0\}$ es válida.

- **Condición 5:** $(\mathbf{I} \wedge \mathbf{fv} \leq \mathbf{0}) \Rightarrow \neg\mathbf{B}$

Ahora, debemos mostrar que la guarda del ciclo deja de valer cuando la función variante es menor o igual a 0.

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

Tenemos que...

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

...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 $\{P\}$ `res := 0; i := 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:

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

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

$$
\begin{aligned}
& wp(\texttt{i := 0},\ P_c)) \\
\equiv\ & def(0) \wedge_L Q_0^i \\
\equiv\ & True \wedge_L (P \wedge res = 0 \wedge 0 = 0) \\
\equiv\ & P \wedge res = 0
\end{aligned}
$$

Entonces,

$$
\begin{aligned}
& wp(\texttt{res := 0},\ wp(\texttt{i := 0},\ P_c)) \\
\equiv\ & wp(\texttt{res := 0},\ P \wedge res = 0) \\
\equiv\ & def(0) \wedge_L Q_0^{res} \\
\equiv\ & P \wedge 0 = 0 \\
\equiv\ & P
\end{aligned}
$$

Se sigue trivialmente que $P \Rightarrow P$, y por lo tanto la tripla $\{P\}$ `res := 0; i := 0` $\{P_c\}$ es válida.
Luego, por monotonía, como $\{P\}$ `res := 0; i := 0` $\{P_c\}$ y $\{P_c\}$ `S` $\{Q\}$ son válidas, también lo es $\{P\}$ `res := 0; i := 0; 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.

Sabemos que *ciudades* es un parámetro de tipo `in`, por lo que no se modifica en ningún punto del programa.
Además, obtenemos de la precondición del programa que...

- $(\exists i : \mathbb{Z})\ (0 \leq i < |ciudades| \wedge_L ciudades[i].habitantes > 50,000)$
  (existe al menos un elemento de *ciudades* cuya segunda componente es estrictamente mayor a $50,000$),
- $(\forall i : \mathbb{Z})\ (0 \leq i < |ciudades \longrightarrow_L ciudades[i].habitantes \geq 0)$
  (ninguna segunda componente de un elemento en *ciudades* es negativa),

  ... y de la postcondición que...
- $res = \sum\limits_{i=0}^{|ciudades|-1} ciudades[i].habitantes$
  (el resultado obtenido es la sumatoria de la segunda componente de todo elemento en *ciudades*).

Por tanto, dadas estas condiciones, se garantiza que $\sum\limits_{j=0}^{|ciudades|-1} ciudades[j].habitantes$ sea mayor a 50.000.