DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Especificación y WP

15 de septiembre de 2024 — Algoritmo y Estructura de datos II

Grupo Gaussianos

Integrante	LU	Correo electrónico
Nuñez Moreno, Gabriel	55/21	`gabrielnm07@gmail.com`
Nakasone, Julián	1074/22	`julunaka@gmail.com`
Pacheco Parrondo, Gerónimo Gabriel	811/23	`pachecogero16@gmail.com`
Cuestas, Martín	466/24	`martincuestas51@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. Definición de Tipos

type ciudad = Char $\times$ $\mathbb{Z}$

type ciudades = $seq\langle Ciudad\rangle$

Donde leemos a ciudad como $< nombre, habitantes >$

2. Especificación de consignas

2.1. Ejercicio 1: grandesCiudades

`proc grandesCiudades` (in ciudades : $seq\langle Ciudad\rangle$) : $seq\langle Ciudad\rangle$
  `requiere` $\{sinRepetidosCiudades(ciudades)\}$
  `requiere` $\{habitantesValidos(ciudades)\}$
  `asegura` $\{enCiudades(res, ciudades) \land grandesEn(res, ciudades) \land sinRepetidosCiudades(res)\}$

`pred habitantesValidos` (ciudades: $seq\langle Ciudad\rangle$) {
  $(\forall i : \mathbb{Z})\ (0 \leq i < |ciudades|) \longrightarrow_L (ciudad[i].habitantes \geq 0)$
}
`pred enCiudades` (res: $seq\langle Ciudad\rangle$, ciudades: $seq\langle ciudad\rangle$) {
  $(\forall i : \mathbb{Z})\ (0 \leq i < |res|) \longrightarrow_L (res[i] \in ciudades) \land (res[i].habitantes > 50.000))$
}
`pred grandesEn` (res: $seq\langle Ciudad\rangle$, ciudades: $seq\langle Ciudad\rangle$) {
  $(\forall i : \mathbb{Z})\ ((0 \leq i < |ciudades| \land_L (ciudades[i].habitantes > 50.000)) \longrightarrow_L ciudades[i] \in res)$
}
`pred sinRepetidosCiudades` (c: $seq\langle Ciudad\rangle$) {
  $(\forall i : \mathbb{Z})\ (0 \leq i < |c| \longrightarrow_L \neg(\exists j : \mathbb{Z})\ (0 \leq j < |c| \land i \neq j \land_L c[i].nombre = c[j].nombre))$
}

2.2. Ejercicio 2: sumaDeHabitantes

`proc sumaDeHabitantes` (in menoresDeCiudades: $seq\langle Ciudad\rangle$, in mayoresDeCiudades : $seq\langle Ciudad\rangle$) : $seq\langle Ciudad\rangle$
  `requiere` $\{|menoresDeCiudades| = |mayoresDeCiudades|\}$
  `requiere` $\{habitantesValidos(menoresDeCiudades) \land habitantesValidos(mayoresDeCiudades)\}$
  `requiere` $\{sinRepetidosCiudades(mayoresDeCiudades) \land sinRepetidosCiudades(menoresDeCiudades)\}$
  `requiere` $\{mismasCiudades(menoresDeCiudades, mayoresDeCiudades\}$
  `asegura` $\{|res| = |mayoresDeCiudades|\}$
  `asegura` $\{mismasCiudades(res, mayoresDeCiudades) \land mismasCiudades(mayoresDeCiudades, res)\}$
  `asegura` $\{sonSumadeAmbas(res, menoresDeCiudades, mayoresDeCiudades) \land sinRepetidosCiudades(res)\}$

`pred mismasCiudades` (r: $seq\langle Ciudad\rangle$, c: $seq\langle Ciudad\rangle$) {
  $(\forall i : \mathbb{Z})\ (0 \leq i < |r| \longrightarrow_L (\exists j : \mathbb{Z})\ (0 \leq j < |c| \land_L (r[i].nombre = c[j].nombre)))$
}
`pred sonSumadeAmbas` (r: $seq\langle Ciudad\rangle$, menores: $seq\langle Ciudad\rangle$, mayores: $seq\langle Ciudad\rangle$) {
  $(\forall i : \mathbb{Z})\ (0 \leq i < |r| \longrightarrow_L (\exists j, k : \mathbb{Z})\ (0 \leq j < |menores| \land 0 \leq k < |mayores| \land_L (r[i].nombre = menores[j].nombre = mayores[k].nombre) \land (r[i].habitantes = menores[j].habitantes + mayores[k].habitantes)))$
}

2.3. Ejercicio 3: hayCamino

`proc hayCamino` (in distancias: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$, in desde: $\mathbb{Z}$, in hasta: $\mathbb{Z}$) : bool
  `requiere` $\{matrizValida(distancias) \land_L enRango(desde, distancias) \land enRango(hasta, distancias)\}$
  `asegura` $\{res = True \iff (\exists s : seq\langle\mathbb{Z}\rangle)\ (|s| \geq 2 \land_L (s[0] = desde) \land (s[|s-1|] = hasta) \land (allConnected(s, distancias))\}$

`pred matrizValida` (M: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
  $esCuadrada(M) \land_L esSimetrica(M) \land sinNegativos(M) \land diagonalConCeros(M)$
}
`pred esCuadrada` (M: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
  $(\forall i : \mathbb{Z})\ (0 \leq i < |M| \longrightarrow_L |M[i]| = |M|)$
}

`pred esSimetrica` (M: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
$(\forall i:\mathbb{Z})\ (0\leq i<|M| \longrightarrow_L (\forall j:\mathbb{Z})\ (0\leq j<|M| \longrightarrow_L M[i][j]=M[j][i]))$
}
`pred sinNegativos` (M: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
$(\forall i:\mathbb{Z})\ (0\leq i<|M| \longrightarrow_L todosPositivos(M[i]))$
}
`pred todosPositivos` (fila: $seq\langle\mathbb{Z}\rangle$) {
$(\forall i:\mathbb{Z})\ (0\leq i<|fila| \longrightarrow_L fila[i]\geq 0)$
}
`pred allConnected` (s: $seq\langle\mathbb{Z}\rangle$,M: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
$(\forall i:\mathbb{Z})\ (0\leq i<|s|-1 \longrightarrow_L hayCaminoDirecto(s[i],s[i+1],M))$
}
`pred diagonalConCeros` (M: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
$(\forall i:\mathbb{Z})\ (0\leq i<|M| \longrightarrow_L (\forall j:\mathbb{Z})\ (0\leq j<|M[i]| \wedge (i=j) \longrightarrow_L M[i][j]=0))$
}
`pred hayCaminoDirecto` (desde: $\mathbb{Z}$, hasta: $\mathbb{Z}$, M: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
$M[desde][hasta]>0$
}
`pred enRango` (x:$\mathbb{Z}$, M:$seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
$0\leq x\leq |M|-1$
}

2.4. Ejercicio 4: cantidadCaminoNSaltos

Para este ejercicio definimos lo siguiente para hacer mas clara la lectura:
- type matriz = $seq\langle seq\langle\mathbb{Z}\rangle\rangle$

`proc cantidadCaminoNSaltos` (inout conexion: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$, in n: $\mathbb{Z}$)
`requiere` $\{esDeConexion(conexion) \wedge conexion = C_0 \wedge n\geq 1\}$
`asegura` $\{(\exists L: seq\langle matriz\rangle)\ (|L|=n \wedge_L L[0]=C_0 \wedge L[|L|-1]=conexion \wedge (\forall i:\mathbb{Z})\ (1\leq i<|L| \longrightarrow_L esProductoMatricial(L[i],L[i-1],L[0]))))\}$

`pred esDeConexion` (M: matriz) {
$esCuadrada(M) \wedge_L esSimetrica(M) \wedge diagonalConCeros(M) \wedge UnosyCeros(M)$
}
`pred UnosyCeros` (M: matriz) {
$(\forall i:\mathbb{Z})(\forall j:\mathbb{Z})\ (0\leq i<|M| \wedge 0\leq j<|M| \longrightarrow_L (M[i][j]=0) \vee (M[i][j]=1))$
}
`pred esProductoMatricial` (R: matriz, M_1: matriz, M_2: matriz) {
$$(\forall i:\mathbb{Z})(\forall j:\mathbb{Z})\ (0\leq i<|M_1| \wedge 0\leq j<|M_2| \longrightarrow_L R[i][j]=\sum_{k=0}^{|R|-1} M_1[i][k]*M_2[k][j])$$
}

2.5. Ejercicio 5: caminoMinimo

`proc caminoMinimo` (in origen: $\mathbb{Z}$, in destino: $\mathbb{Z}$, in distancias: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$
`requiere` $\{matrizValida(distancias) \wedge_L enRango(origen,distancias) \wedge enRango(destino,distancias)\}$
`asegura` $\{(|res|=0 \wedge \neg(\exists s: seq\langle\mathbb{Z}\rangle)\ (esCaminoMinimo(s,origen,destino,distancias)) \vee (esCaminoMinimo(res,origen,destino,distancias))\}$
s

`pred esCaminoMinimo` (s: $seq\langle\mathbb{Z}\rangle$, origen: $\mathbb{Z}$, destino: $\mathbb{Z}$, distancias: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
$|s|\geq 2 \wedge_L (s[0]=origen) \wedge (s[|s|-1]=destino) \wedge$
$allConnected(s,distancias) \wedge masOptimo(s,origen,destino,distancias))$
}
`pred masOptimo` (s: $seq\langle\mathbb{Z}\rangle$, origen: $\mathbb{Z}$, destino: $\mathbb{Z}$, distancias: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
$\neg(\exists l: seq\langle\mathbb{Z}\rangle)\ (|l|\geq 2 \wedge_L (l[0]=origen) \wedge (l[|l-1]=destino) \wedge allConnected(l,distancias) \wedge$
$distanciasTotal(l,distancias) < distanciaTotal(s,distancias))$
}
`aux distanciaTotal` (s: $seq\langle\mathbb{Z}\rangle$, M: $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z} = \sum_{i=0}^{|s|-2} distancia(s[i],s[i+1],M)$;

`aux distancia` (d: $\mathbb{Z}$, h: $\mathbb{Z}$, M: $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z} = M[d][h]$;

3. Demostraciones de correctitud

3.1. Ejercicio 2.1

Tenemos la siguiente especificacion:

`proc` `poblacionTotal` (in ciudades: $seq\langle Ciudad\rangle$) : $\mathbb{Z}$

`requiere` $\{(\exists i : \mathbb{Z})\ (0 \leq i < |ciudades| \wedge_L ciudades[i].habitantes > 50000) \wedge$
$(\forall i : \mathbb{Z})\ (0 \leq i < |ciudades| \longrightarrow_L ciudades[i].habitantes \geq 0) \wedge$
$(\forall i : \mathbb{Z})\ ((\forall j : \mathbb{Z})\ (0 \leq i < j < |ciudades| \longrightarrow_L ciudades[i].nombre \neq ciudades[j].nombre))\}$

`asegura` $\{res = \sum_{i=0}^{|ciudades|-1} ciudades[i].habitantes\}$

Y su implementacion:

```
res := 0;
i := 0;
while (i < ciudades.length) do
    res := res + ciudades[i].habitantes;
    i := i + 1
endwhile
```

Código 1: Implementacion de la especificacion en SmallLang

Para demostrar que la implementación es correcta con respecto a la especificación debemos demostrar los 3 puntos del teorema del invariante y los 2 puntos del teorema de terminación de ciclo, que son:

1. $P_C \Rightarrow I$
2. $\{I \wedge B\}S\{I\}$,
3. $I \wedge \neg B \Rightarrow Q_C$,
4. $\{I \wedge B \wedge v_0 = fv\}\,\mathbf{S}\,\{fv < v_0\}$, y
5. $I \wedge fv \leq 0 \Rightarrow \neg B$

Primero, definimos nuestra Precondicion del Ciclo (P_C), la guarda (B), la Postcondicion del Ciclo (Q_C), nuestra propuesta de invariante (I) y la funcion variante (f_v):

- $P_C \equiv res = 0 \wedge i = 0$
- $B \equiv i < ciudades.length$
- $Q_C \equiv res = \sum_{i=0}^{|ciudades|-1} ciudades[i].habitantes$
- $I \equiv 0 \leq i \leq |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes$
- $f_v = |ciudades| - i$ Notemos tambien que:
- $S_1 \equiv res := res + ciudades[i].habitantes$
- $S_2 \equiv i := i + 1$

Aclaraciones previas:
- Tomamos las definiciones de variables con el valor de True, para hacer más clara la lectura.
- En casos donde las definiciones tengan influencia sobre el desarrollo, estarán incluidas.

Empezamos con el 1. $P_C \longrightarrow I$

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

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

$i = 0 \wedge res = 0 \longrightarrow True \wedge (0 = 0)$

$i = 0 \wedge res = 0 \longrightarrow True \wedge True$

$\equiv True$ ✓

Seguimos con el 2. $\{I \wedge B\}\, \mathbf{S}\, \{I\}$

$WP(S, I) \equiv WP(S_1; S_2, I)$ esto vale por el axioma 3

WP($res := res + ciudades[i].habitantes, WP_1(i := i + 1, 0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes) \equiv$

Primero vemos WP_1:

Por axioma 1: $WP_1 \equiv (0 \leq i + 1 \leq |ciudades| \wedge_L res = \sum_{j=0}^{i} ciudades[i].habitantes) \equiv$

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

Ahora seguimos con la WP original:

$WP(res := res + ciudades[i].habitantes, -1 \leq i < |ciudades| \wedge_L res = \sum_{j=0}^{i} ciudades[j].habitantes)$

Por axioma 1:

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

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

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

Luego, compruebo que $\{I \wedge B\} \longrightarrow WP$

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

$\equiv True$ ✓

Continuamos con el 3. $I \wedge \neg B \longrightarrow Q_c$

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

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

Entonces como: $res = \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes \longrightarrow Q_C \equiv res = \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes$✓

Luego: $i = |ciudades| \wedge res = \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes \longrightarrow Q_C$✓

Hasta este punto ya hemos probado la correctitud parcial del ciclo, ahora probaremos que el ciclo finaliza.

4. $\{I \wedge B \wedge v_0 = fv\}\, \mathbf{S}\, \{fv < v_0\}$

WP(S,$f_v < v_0$) $\equiv WP$(S$_1$; S_2, $f_v < v_0$) por axioma 3

WP$_1$($i := i+1, (|ciudades| < i) < v_0) \equiv (|ciudades| - i - 1) < v_0$ por axioma 1

WP($res := res + ciudades[i].habitantes, (|ciudades| - i - 1) < v_0) \equiv$

$\equiv (0 \leq i < |ciudades| \wedge |ciudades| - i - 1 < v_0$ $\equiv$WP

Veamos que $\{I \wedge B \wedge v_0 = fv\} \longrightarrow$ WP

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

Al ser $res = \sum_{j=0}^{i-1} ciudades[j].habitantes$ irrelevante para la demostración, lo daremos por hecho (i.e no estará presente)

$\equiv \; 0 \leq i < |ciudades| \wedge (|ciudades| - i = v_0) \longrightarrow (0 \leq i < |ciudades|) \wedge (|ciudades| - i - 1 < v_0)$

$a.$ $0 \leq i < |ciudades| \longrightarrow 0 \leq i < |ciudades|$ ✓

$b.$ $|ciudades| - i = v_0 \longrightarrow |ciudades| - i - 1 < |ciudades| - i \equiv$

$\equiv \; |ciudades| - i = v_0 \longrightarrow True \equiv$

$\equiv \; True$ ✓

Entonces $(0 \leq i < |ciudades|) \wedge (|ciudades| - i = v_0) \longrightarrow (0 \leq i < |ciudades|) \wedge (|ciudades| - i - 1 < v_0)$ ✓

5. $I \wedge f_v \leq 0 \longrightarrow \neg B$

Al ser $res = \sum_{j=0}^{i-1} ciudades[j].habitantes$ irrelevante para la demostración, lo daremos por hecho y no estará presente.

$\equiv 0 \leq i \leq |ciudades| \wedge (|ciudades| - i \leq 0) \longrightarrow i \geq |ciudades|$

$\equiv 0 \leq i \leq |ciudades| \wedge (|ciudades| \leq i) \longrightarrow i \geq |ciudades|$

$\equiv i = |ciudades| \longrightarrow i \geq |ciudades|$

$\equiv True$ ✓

Finalmente, hemos probado la correctitud del ciclo, y la finalizacion del mismo. Por lo tanto, solo resta probar la correctitud del programa respeto a la especificación. Para ello estudiaremos la siguiente tripla de Hoare.

$$\{\text{Requiere}\}\, \mathbf{S}\, \{P_C\}$$

Recordemos el requiere, S y la P_C aqui

Buscamos la WP(S1;S2,P_C) $\equiv$WP(S1, WP$_1$($S2, Pc) \equiv WP(res := 0, WP_1(i = 0, res = 0 \wedge i = 0))$

Por axioma 1: $WP(res := 0, res = 0 \wedge \; 0 = 0) \equiv \; 0 = 0 \wedge \; 0 = 0 \equiv \; True$

Luego, como ($WP \equiv True$), tenemos que ($requiere \longrightarrow True$) es una tautología. De esta forma probamos que el programa es correcto respecto a la especificación dada.

3.2. Ejercicio 2.2

Para probar que con la finalización del programa el valor de habitantes devuelto es mayor a 50.000, utilizo un nuevo Q_C y para realizar nuevamente las demostraciones con este nuevo Q_C, utilizo un nuevo Invariante, y una nueva P_C

- $Q_C \equiv res > 50000 \wedge res = \sum_{i=0}^{|ciudades|-1} ciudades[i].habitantes$
- I $\equiv \quad 0 \leq i \leq |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes > 50000$
- $P_C \equiv i = 0 \quad \wedge res = 0 \wedge (\exists i : \mathbb{Z})\ (0 \leq i < |ciudades| \wedge_L ciudades[i].habitantes > 50000) \wedge$
  $(\forall i : \mathbb{Z})\ (0 \leq i < |ciudades| \longrightarrow_L ciudades[i].habitantes \geq 0)$

Ahora, explicaremos como cambia cada punto de la demostración de correctitud parcial con este nuevo invariante.

$P_C \longrightarrow I$:

$i = 0 \wedge res = 0 \wedge (\exists i : \mathbb{Z})\ (0 \leq i < |ciudades| \wedge_L ciudades[i].habitantes > 50000) \wedge$
$(\forall i : \mathbb{Z})\ (0 \leq i < |ciudades| \longrightarrow_L ciudades[i].habitantes \geq 0) \longrightarrow 0 \leq |ciudades| \wedge 0 = \sum_{j=0}^{-1} ciudades[i].habitantes \wedge \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes > 50000$

$i = 0 \wedge res = 0 \wedge (\exists i : \mathbb{Z})\ (0 \leq i < |ciudades| \wedge_L ciudades[i].habitantes > 50000) \wedge$
$(\forall i : \mathbb{Z})\ (0 \leq i < |ciudades| \longrightarrow_L ciudades[i].habitantes \geq 0) \longrightarrow True \wedge (0 = 0) \wedge \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes > 50000$

$i = 0 \wedge res = 0 \wedge (\exists i : \mathbb{Z})\ (0 \leq i < |ciudades| \wedge_L ciudades[i].habitantes > 50000) \wedge$
$(\forall i : \mathbb{Z})\ (0 \leq i < |ciudades| \longrightarrow_L ciudades[i].habitantes \geq 0) \longrightarrow True \wedge True \wedge \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes > 50000$

$\equiv True \checkmark$

En este caso, la demostración es equivalente a la dada anteriormente, excepto que el agregado al nuevo invariante se prueba con el requiere de la especificación, ya que este último nos asegura de que existirá al menos una ciudad en ciudades que cuente con más de 50000 habitantes (cifra que sera sumada al total), y además, no pueden existir habitantes negativos, por lo que en el transcurso de la sumatoria el total jamás disminuirá.

Veamos ahora: $\{I \wedge B\} \quad S \quad \{I\}$:

$WP(S, I) \equiv WP(S_1; S_2, I)$

Por axioma 3

WP($S_1, WP_1(S_2, 0 \leq i \leq |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes > 50000)$

Por axioma 1:

$WP_1(0 \leq i+1 \leq |ciudades| \wedge_L res = \sum_{j=0}^{i} ciudades[i].habitantes) \wedge \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes > 50000) \equiv$

$\equiv (-1 \leq i < |ciudades| \wedge_L res = \sum_{j=0}^{i} ciudades[i].habitantes) \wedge \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes > 50000)$

Ahora seguimos con la WP original:

$WP(S_1, -1 \leq i < |ciudades| \wedge_L res = \sum_{j=0}^{i} ciudades[j].habitantes) \wedge \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes > 50000)$

Por axioma 1:

$(-1 \leq i < |ciudades| \wedge_L res + ciudades[i].habitantes = \sum\limits_{j=0}^{i} ciudades[j].habitantes) \wedge \sum\limits_{j=0}^{|ciudades|-1} ciudades[j].habitantes > 50000) \equiv$

$\equiv (-1 \leq i < |ciudades| \wedge_L res = \sum\limits_{j=0}^{i-1} ciudades[j].habitantes) \wedge \sum\limits_{j=0}^{|ciudades|-1} ciudades[j].habitantes > 50000)$

Luego, como $\{I \wedge B\} \longrightarrow WP$ es equivalente a la demostracion hecha con anterioridad, sigue valiendo.

Sigamos con: $I \wedge \neg B \Rightarrow Q_C$

$i = |ciudades| \wedge_L res = \sum\limits_{j=0}^{i-1} ciudades[j].habitantes \wedge \sum\limits_{j=0}^{|ciudades|-1} ciudades[j].habitantes > 50000 \longrightarrow Q_C$

Como $Q_C \equiv res > 50000 \wedge res = \sum\limits_{i=0}^{|ciudades|-1} ciudades[i].habitantes$

Tenemos que: $i = |ciudades| \wedge_L res = \sum\limits_{j=0}^{i-1} ciudades[j].habitantes \wedge \sum\limits_{j=0}^{|ciudades|-1} ciudades[j].habitantes > 50000 \longrightarrow res = \sum\limits_{j=0}^{|ciudades|-1} ciudades[j].habitantes \wedge \sum\limits_{j=0}^{|ciudades|-1} ciudades[j].habitantes > 50000$

Es decir: $res = \sum\limits_{j=0}^{|ciudades|-1} ciudades[j].habitantes \wedge res > 50000 \equiv Q_C$

$Q_c \longrightarrow Q_c \equiv True$

Pasando a la demostración de finalización, para asegurar que el ciclo efectivamente terminara en una serie finita de pasos, bien sabemos que en los dos desarrollos dados anteriormente, $(res = \sum\limits_{j=0}^{i-1} ciudades[j].habitantes)$ es redundante, ya que no influye de ninguna forma. Con el nuevo invariante dado, se puede afirmar que el nuevo agregado:

$\sum\limits_{j=0}^{|ciudades|-1} ciudades[j].habitantes$ >50.000 tampoco tendrá ninguna influencia en los desarrollos, por lo que la demostración se dará exactamente de la misma forma, y seguirá siendo valida, por lo que el ciclo terminará.

A su vez, en cuanto a la demostración de correctitud de la implementación sobre la especificación, el único cambio se encuentra en P_C, y este al buscar la WP para probar la tripla de Hoare (

$$\{\text{Requiere}\}\ \mathbf{S}\ \{P_C\}$$

) no es modificado por S, por lo tanto quedara la misma expresión en la WP encontrada. Dicha expresion se encuentra en el requiere, y cuando realizamos la implicacion (Requiere $\longrightarrow$WP) para ver si la tripla de Hoare es correcta, sera True.