TP 1

Especificación y WP

15 de septiembre de 2024 — Algoritmos y Estructuras de Datos

Grupo Siu Guaranin't

Integrante	LU	Correo electrónico
Aleman, Diego Eitan	867/24	diegoaaleman17@gmail.com
Provvisionato, Sofía	152/24	sofiaprovvisionato7@gmail.com
Hoyo Correa, Bruno	473/24	hcbruno59@gmail.com
Skidelsky, Agustín	1109/23	agustinskidelski@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

$\texttt{proc grandesCiudades}$ (in Ciudades : $seq\langle Ciudad\rangle$) : $seq\langle Ciudad\rangle$

$\texttt{requiere}$ $\{(sinRepetidos(ciudades)) \land (\forall i \in \mathbb{Z})(0 \leq i < |ciudades| \longrightarrow_L ciudades[i].habitantes \geq 0)\}$

$\texttt{asegura}$ $\{|res| \leq |ciudades| \land (\forall i \in \mathbb{Z})(0 \leq i < |ciudades| \longrightarrow_L$

$(ciudades[i] \in res \leftrightarrow ciudades[i].habitantes > 50000))\}$

$\texttt{pred sinRepetidos}$ (in ciudades: $seq\langle Ciudad\rangle$)) {

$(\forall i, j \in \mathbb{Z})(0 \leq i, j < |ciudades| \land i \neq j \longrightarrow_L (ciudades[i].nombre \neq ciudades[j].nombre))$

}

era necesario que no hubiera repetidos para que funcione?

1.2. sumaHabitantes

$\texttt{proc sumaHabitantes}$ (in menoresDeCiudades : $seq\langle Ciudad\rangle$,in mayoresDeCiudades : $seq\langle Ciudad\rangle$) : $seq\langle \mathbb{Z}\rangle$

$\texttt{requiere}$ $\{(|menoresCiudades| = |mayoresCiudades|) \land$

$(sinRepetidos(menoresDeCiudades) \land sinRepetidos(mayoresDeCiudades)) \land$

$mismosElementos(menoresDeCiudades, mayoresDeCiudades) \land$

$(\forall i \in \mathbb{Z})(0 \leq i < |menoresDeCiudades| \longrightarrow_L$

$(menoresDeCiudades[i].habitantes \geq 0 \land mayoresDeCiudades[i].habitantes \geq 0))$

}

$\texttt{asegura}$ $\{(|res| = |menoresCiudades|) \land (\forall i \in \mathbb{Z})(0 \leq i < |menoresCiudades| \longrightarrow_L$

$(\ (\exists j \in \mathbb{Z})(menoresCiudades[i].nombre = mayoresCiudades[j].nombre) \land$

$((menoresCiudades[i].nombre,\ menoresCiudades[i].habitantes + mayoresCiudades[j].habitantes) \in res)))\}$

la sintaxis para crear tuplas es <elem1, elem2>

$\texttt{pred sinRepetidos}$ (in ciudades: $seq\langle Ciudad\rangle$)) {

$(\forall i, j \in \mathbb{Z})(0 \leq i, j < |ciudades| \ \ \land \ \ i \neq j \longrightarrow_L ciudades[i].nombre \neq ciudades[j].nombre)$

}

pueden declarar predicados globales con las cosas que van a querer usar mas de una vez

$\texttt{pred mismosElementos}$ (in menoresDeCiudades : $seq\langle Ciudad\rangle$,in mayoresDeCiudades : $seq\langle Ciudad\rangle$) {

$(\forall i \in \mathbb{Z})((0 \leq i \leq |menoresCiudades|) \longrightarrow_L (\exists j \in \mathbb{Z})(menoresCiudades[i].nombre = mayoresCiudades[j].nombre))\land$

$(\forall j \in \mathbb{Z})((0 \leq j \leq |mayoresCiudades|) \longrightarrow_L (\exists i \in \mathbb{Z})(mayoresCiudades[j].nombre = menoresCiudades[i].nombre))$

}

1.3. hayCamino

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

requiere $\{matrizCuadrada(distancias) \land$
$distanciaNoNegativa(distancias) \land$
$distanciaCorrecta(distancias) \land$
$distanciaASiMisma(distancias) \land$
$ciudadesEnRango(distancias, desde, hasta)$
$\}$

asegura $\{recorridoCorrecto(distancias, desde, hasta)$
$\}$

no dicen nada sobre el valor de res, basta con decir res = recorridoCorrecto

pred matrizCuadrada (distancias : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
$(\forall i \in \mathbb{Z})(0 \leq i < |distancias| \longrightarrow_L (|distancias[i]| = |distancias|))$
}

pred distanciaNoNegativa (distancias : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
$(\forall i, j \in \mathbb{Z})(0 \leq i, j < |distancias| \longrightarrow_L (distancias[i][j] \geq 0))$
}

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

pred distanciaASiMisma (distancias : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
$(\forall i \in \mathbb{Z})(0 \leq i < |distancias| \longrightarrow_L (distancias[i][i] = 0))$
}

pred ciudadesEnRango (distancias : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$,in desde: $\mathbb{Z}$, in hasta: $\mathbb{Z}$) {
$(0 \leq desde < |distancias|) \land (0 \leq hasta < |distancias|)$
}

pred recorridoCorrecto (distancias : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$,in desde: $\mathbb{Z}$, in hasta: $\mathbb{Z}$) {
$(\exists recorrido : seq\langle\mathbb{Z}\rangle)(caminoCorrecto(desde, hasta, distancias, recorrido))$
}

pred caminoCorrecto (distancias : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$,in desde: $\mathbb{Z}$, in hasta: $\mathbb{Z}$, camino:$seq\langle\mathbb{Z}\rangle$) {
$(\forall j : \mathbb{Z})(0 \leq j < |camino| \longrightarrow_L ((0 \leq camino[j] < |distancias|) \land (2 \leq |camino| \leq |distancias|))) \land$
$(\forall i : \mathbb{Z})(0 \leq i < (|camino| - 2) \longrightarrow_L (((distancias[camino[i]][camino[i+1]]) \geq 0) \land$
$((camino[0] = desde) \land (camino[|camino| - 1] = hasta))))$
}

si la distancia es 0 no hay un camino

si i vale a lo sumo |camino| - 3, nunca aseguran que el elemento en |camino| -2 y el hasta cumplan que su distancia es 0.

1.4. cantidadCaminosNSaltos

que valores puede tomar la matriz conexion?
que forma tiene que tener?

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

requiere $\{matrizCuadrada(conexion) \wedge matrizValida(conexion) \wedge n \geq 1\}$

incompleto

asegura $\{(\exists lista : seq\langle seq\langle seq\langle \mathbb{Z}\rangle\rangle\rangle)(lista[0] = conexion_0) \wedge$

$(\forall i \in \mathbb{Z})(i \geq 1 \longrightarrow_L (multiplicar(lista[i-1], lista[0], lista[i]) \wedge (conexion = lista[n-1])))$

$\}$

si quieren hablar de conexion0, tienen que declararlo en el asegura. no existe hasta que ustedes la creen

si no le ponen limite superior al rango esto vale para todos los numeros enteros osea i puede valer infinito

pred matrizCuadrada (conexion : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i \in \mathbb{Z})(0 \leq i < |conexion| \longrightarrow_L (|conexion[i]| = |conexion|))$

}

pred matrizValida (conexion : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

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

}

pred Multiplicar (m1 : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$,m2: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$,m3 : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall f, c \in \mathbb{Z})((0 \leq f, c < |m1|) \longrightarrow_L (m3[f][c] = \sum_{i=0}^{|m1|-1}(m1[f][i] \cdot m2[i][c])))$

}

1.5. caminoMinimo

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

`requiere` $\{matrizCuadrada(distancias) \wedge$

$distanciaNoNegativa(distancias) \wedge$

$distanciaCorrecta(distancias) \wedge$

$distanciaASiMisma(distancias) \wedge$

$ciudadesEnRango(distancias, origen, destino)$

$\}$

`asegura` $\{(\neg(hayCamino(distancias, origen, destino)) \longrightarrow |res| = 0) \vee$

$(hayCamino(distancias, origen, destino) \longrightarrow (recorridoMinimo(origen, destino, distancias, res)))$

$\}$

`pred matrizCuadrada` (distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i \in \mathbb{Z})(0 \leq i < |distancias| \longrightarrow_L (|distancias[i]| = |distancias|))$

}

`pred distanciaNoNegativa` (distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i \in \mathbb{Z})(\forall j \in \mathbb{Z})(0 \leq j, i < |distancias| \longrightarrow_L (|distancias[i][j]| \geq 0))$

}

`pred distanciaASiMisma` (distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i \in \mathbb{Z})(0 \leq i < |distancias| \longrightarrow_L (distancias[i][i] = 0))$

}

`pred distanciaCorrecta` (distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i \in \mathbb{Z})(\forall j \in \mathbb{Z})(0 \leq j, i < |distancias| \longrightarrow_L (distancias[i][j] = distancias[j][i]))$

}

`pred ciudadesEnRango` (distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, origen : $\mathbb{Z}$, destino : $\mathbb{Z}$) {

$(0 \leq origen < |distancias|) \wedge (0 \leq destino < |distancias|)$

}

`pred recorridoMinimo` (distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, origen : $\mathbb{Z}$, destino : $\mathbb{Z}$, res : $seq\langle \mathbb{Z}\rangle$) {

$caminoCorrecto(origen, destino, distancias, res) \wedge$

$(\forall otroRecorrido : seq\langle \mathbb{Z}\rangle)(\ caminoCorrecto(origen, destino, distancias, otroRecorrido) \longrightarrow$

$(\ \sum_{i=0}^{|res|-2}(distancias[res[i]][res[i+1]]) \quad \leq \quad \sum_{i=0}^{|otroRecorrido|-2}(distancias[otroRecorrido[i]][otroRecorrido[i+1]])$

$)\ \ )\}$

esto tiene el mismo error que arriba, no suman la distancia del anteúltimo con el último elemento

podrian armarse un pred auxiliar que sume dos listas

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

$(\forall i \in \mathbb{Z})(0 \leq i < |camino| \longrightarrow_L ((0 \leq camino[i] < |distancias|) \wedge (2 \leq |camino| \leq |distancias|))) \wedge$

$(\forall j \in \mathbb{Z})(\ 0 \leq i < (|camino| - 1) \longrightarrow_L (\ distancias[camino[i]][camino[i+1]] \geq 0\ \wedge$

$((camino[0] = origen) \wedge (camino[|camino| - 1] = destino))\ ))$

}

2. Correctitud

incompleto

2.1. Respecto a la especificación

Para demostrar la correctitud parcial del ciclo, tengo que probar que:

1. (Pc $\longrightarrow$I)
2. (I $\wedge B) S(I)$
3. (I $\wedge \neg B) \longrightarrow Qc$

Y luego, para demostrar que el ciclo finaliza:

4. $(I \wedge B \wedge v_0 = fv)\, S\, (fv < v_0)$
5. $(I \wedge fv \leq 0 \longrightarrow \neg B)$

Para eso considero las siguientes variables lógicas, que reemplazare en las fórmulas correspondientes:

$$B = i < |\text{ciudades}|$$

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

$$Pc = (\text{res} = 0) \wedge (i = 0) \wedge (i < |\text{ciudades}|)$$

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

$$fv = |\text{ciudades}| - i$$

Demostración:

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

$\equiv$ (reemplazando las identidades)$(0 \leq |\text{ciudades}|) \longrightarrow (0 \leq 0 \leq |\text{ciudades}| \wedge (0 = \sum_{j=0}^{-1} \text{ciudades}[j].\text{habitantes}))$

esta buena la idea de poner texto en medio de la demo, pero haganlo con tal de que no se mezcle. pueden poner una linea de demo, una linea de explicacion, otra linea de demo, y asi.

Por lo tanto, se cumple la primera condición.

el operador lógico que tienen que usar es el de congruencia, dice que el valor de verdad de arriba es igual al valor de verdad de lo de abajo

2. $\{I \wedge B\} \longrightarrow wp(S, I)$

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

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

Reemplazo el valor de i en las variables libres y me queda:

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

esta i debería ser i +1

Reemplazo res en las variables libres:

res + ciudades[i].habitantes = $\sum_{j=0}^{i}$ ciudades[j].habitantes

res = $\sum_{j=0}^{i-1}$ ciudades[j].habitantes y el $i + 1 \leq |ciudades|$?

$\{I \wedge B\} \longrightarrow wp(S, I)$

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

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

Sí, porque el $\wedge$ requiere para ser verdadero que ambos sean verdaderos, por lo tanto si a y b son verdaderos, entonces b es verdadero y se cumple Qc. está poco clara la explicacion. creo que quisieron decir lo siguiente

$$a \wedge b \rightarrow b \equiv True$$

si esto es lo que quieren afirmar, tienen que demostrarlo.

¿que es 3?

3. $((0 \leq i \leq$ |ciudades|) $\wedge$ (res = $\sum_{j=0}^{i-1}$ ciudades[j].habitantes) $\wedge$ ($i \geq$ |ciudades|)) $\longrightarrow$
(res = $\sum_{i=0}^{|\text{ciudades}|-1}$ ciudades[i].habitantes)

((i = |ciudades|) (porque i debe ser menor o igual y a la vez mayor o igual) $\wedge$ (res = $\sum_{j=0}^{i-1}$ ciudades[j].habitantes)) $\longrightarrow_L$

misma correccion que arriba

(res = $\sum_{i=0}^{|\text{ciudades}|-1}$ ciudades[i].habitantes)

Reemplazo i por |ciudades|

(res = $\sum_{i=0}^{|\text{ciudades}|-1}$ ciudades[i].habitantes) = (res = $\sum_{i=0}^{|\text{ciudades}|-1}$ ciudades[i].habitantes)

Ambos predicados son iguales por lo tanto se cumple la implicación. Es decir que hasta aquí se prueba la validez parcial del loop. Sigue la prueba de su terminación:

4. $\{\text{I} \wedge B \wedge v_0 = fv\} \longrightarrow WP(S, fv < v_0)$

$\text{WP(S, fv} < \text{v}_0) = WP(\text{res} = \text{res} + \text{ciudades}[i].\text{habitantes}; i = i+1, |\text{ciudades}| - i < v_0)$

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

$$= \text{def}(\text{ciudades}[i]) \wedge |\text{ciudades}| - i - 1 < v_0$$

$0 \leq i < |\text{ciudades}| \wedge |\text{ciudades}| - i - 1 < v_0$

$\{\text{I} \wedge B \wedge v_0 = fv\}$

$(i \leq |\text{ciudades}| \ \wedge \ \text{res} = \sum_{j=0}^{i-1} \text{ciudades}[j].\text{habitantes}) \wedge (i < |\text{ciudades}|) \wedge (|\text{ciudades}| - 1 = v_0)$

Hay 3 condiciones que tienen que ser verdaderas, $A \wedge B \wedge C$, me alcanza con probar que una sola de ellas implica que $fv < v_0$ para demostrar la implicación, por lo que ignoro el resto y considero:

escriban la implicacion entera,
no se entiende a que hacen referencia.
con cual de los terminos alcanza para
la demostracion?

$$|\text{ciudades}| - i - 1 < |\text{ciudades}| - 1 = v_0$$

incompleto

Se cumple, puesto que $i \geq 0$.

como cumplen que $0 \leq i < |ciudades|$?

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

como $\text{i} \leq |\text{ciudades}|$ y $|\text{ciudades}| \leq i$, entonces $i = |\text{ciudades}|$

Entonces, ignorando lo que tiene que ver con *res* pues no es relevante:

$i = |\text{ciudades}| \longrightarrow i \geq |\text{ciudades}|$

La implicación es válida, por lo que se cumple esta condición y comprobamos que **el ciclo es válido.**

y la demostracion de que el programa entero es correcto?

no es que no es relevante, es que al estar demostrando una implicacion,
falso implica cualquier cosa es verdadero. entonces el valor de verdad de todo
lo que menciona a res, no cambia lo que ustedes quieren demostrar. eso no significa
que se pueden ahorrar escribirlo. o si no lo quieren escribir, tienen que justificar por qué.
decir que lo sacan porque no es 'relevante' es incorrecto

2.2. Resultado mayor a 50.000

Para demostrar que res es mayor a 50.000, tengo que probar que el loop es correcto y que siempre resulta en una suma mayor a 50000. Para esto debo llevar a cabo el mismo razonamiento que antes, pero agregando ciertos predicados a la precondición, el invariante y la postcondición. Lo que debo probar es lo siguiente:

a) (Pc $\longrightarrow$I)

b) (I $\wedge B)S(I)$

c) (I $\wedge\neg B) \longrightarrow Qc$

por qué no hay que redemostrar la terminación?

Para eso, utilizo las siguientes variables lógicas:

$B = i < |\text{ciudades}|$

$$Qc = \text{res} = \sum_{i=0}^{|\text{ciudades}|-1} \text{ciudades}[i].\text{habitantes} \wedge \text{res} > 50000$$

$Pc = (\text{res} = 0) \wedge (i = 0) \wedge (i < |\text{ciudades}|) \wedge$

$(\exists\, i \in \mathbb{Z})(\mathit{ciudades}[i].\text{habitantes} > 50000) \wedge (\forall\, i \in \mathbb{Z})(0 < i \leq |\text{ciudades}| \longrightarrow_L (\mathit{ciudades}[i].\text{habitantes} \geq 0))$

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

$(\exists\, i \in \mathbb{Z})(\mathit{ciudades}[i].\text{habitantes} > 50000) \wedge (\forall\, i \in \mathbb{Z})(0 < i \leq |\text{ciudades}| \longrightarrow_L (\mathit{ciudades}[i].\text{habitantes} \geq 0))$

$fv = |\text{ciudades}| - i$

usen una letra distinta para cada rango, sino se confunde cual es cual

Demostración:

a)

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

Como no cambian las relaciones de fuerza, puesto que le agrego a ambos términos los mismos predicados, siendo ambos verdaderos, entonces se mantiene la demostración del ejercicio 2.1.

b) $\{\text{I} \wedge B\} \longrightarrow wp(S, I)$

$\text{WP(S, I)} = \text{WP}(\;\text{res} = \text{res} + \text{ciudades[i].habitantes};\;\; \text{i} = \text{i} + 1,\;\; \text{i} \leq |\text{ciudades}| \;\wedge\; \text{res} = \sum_{j=0}^{i-1} \text{ciudades}[j].\text{habitantes} \;\;\wedge$
$(\exists\, i \in \mathbb{Z})(\mathit{ciudades}[i].\text{habitantes} > 50000) \;\;\wedge\;\; (\forall\, i \in \mathbb{Z})(0 < i \leq |\text{ciudades}| \longrightarrow_L (\mathit{ciudades}[i].\text{habitantes} \geq 0))\;)$

$= \text{WP}\;(\text{res} = \text{res} + \text{ciudades[i].habitantes},\; \text{WP}(\;\text{i} = \text{i} + 1,\; \text{i} \leq |\text{ciudades}| \wedge \text{res} = \sum_{j=0}^{i-1} \text{ciudades}[j].\text{habitantes} \;\wedge$
$(\exists\, i \in \mathbb{Z})(\mathit{ciudades}[i].\text{habitantes} > 50000) \wedge (\forall\, i \in \mathbb{Z})(0 < i \leq |\text{ciudades}| \longrightarrow_L (\mathit{ciudades}[i].\text{habitantes} \geq 0)))\;)$

Reemplazo i en las variables libres y me queda: mismo error que arriba

WP(res = res + ciudades[i].habitantes, i $\leq$ |ciudades| $\wedge$ res $= \sum_{j=0}^{i}$ ciudades$[j]$.habitantes $\wedge$
$(\exists\, i \in \mathbb{Z})(ciudades[i]$.habitantes $> 50000) \wedge (\forall\, i \in \mathbb{Z})(0 < i \leq |$ciudades$| \longrightarrow_L (ciudades[i]$.habitantes $\geq 0))$)

Reemplazo res en las variables e ignoro todos los términos que no son relevantes:

res + ciudades[i].habitantes $= \sum_{j=0}^{i}$ ciudades$[j]$.habitantes

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

$\{$I $\wedge B\} \longrightarrow wp(S, I)$

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

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

que es c?

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

$$i = |\text{ciudades}| \wedge \text{res} = \sum_{j=0}^{i-1} \text{ciudades}[j].\text{habitantes} \wedge (\exists\, i \in \mathbb{Z})(ciudades[i].\text{habitantes} > 50000) \wedge$$

$(\forall\, i \in \mathbb{Z})(0 < i \leq |$ciudades$| \longrightarrow_L (ciudades[i]$.habitantes $\geq 0 \longrightarrow_L$ (res $= \sum_{i=0}^{|\text{ciudades}|-1}$ ciudades$[i]$.habitantes $\wedge$ res $>$ 50000)))

Reemplazo i por |ciudades|:

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

incompleto

La expresión a la que indica que existe un número natural en ciudades[i]habitantes que es mayor a 50000 y que todas las demás posiciones de la lista ciudades.habitantes son positivas. Teniendo en cuenta que

res $= (\sum_{i=0}^{|\text{ciudades}|-1}$ ciudades$[i]$.habitantes)

los predicados anteriores resultan en que la suma siempre es necesariamente mayor a 50000, puesto que sumar números positivos a al menos un elemento mayor a 50000, el resultado de dicha suma siempre será mayor a igual a 50000.

esta explicación no está TAN mal, pero no es suficiente. deberían poder probarlo
de manera lógica. tip: si le cambian el índice del para todo a j,
piensen que pasa en los casos i =j y !i=j