Trabajo Practico de Especificacion

Especificacion y WP(Weakness Precondition)

9 de septiembre de 2024 — Algoritmos y Estructuras de Datos 1

pesutipolimardiano

Integrante	LU	Correo electrónico
Nievas, Martin	453/24	`tinnivas@gmail.com`
Bercovich, Maximo	002/01	`email2@dominio.com`
Apellido, Nicolas	003/01	`email3@dominio.com`
Pomsztein, Andy	624/24	`pomszteinandy@gmai.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. Enunciados

1.1. Especficiacion

1. grandesCiudades A partir de una lista de ciudades, devuelve aquellas que tienen m´as de 50.000 habitantes.
proc grandesCiudades (in ciudades: $seq\langle Ciudad\rangle$) : $seq\langle Ciudad\rangle$

2. sumaDeHabitantes: Por cuestiones de planificacion urbana, las ciudades registran sus habitantes mayores de edad por un lado y menores de edad por el otro. Dadas dos listas de ciudades del mismo largo con los mismos nombres, una con sus habitantes mayores y otra con sus habitantes menores, este procedimiento debe devolver una lista de ciudades con la cantidad total de sus habitantes.
proc sumaDeHabitantes (in menoresDeCiudades: $seq\langle Ciudad\rangle$, in mayoresDeCiudades: $seq\langle Ciudad\rangle$) : $seq\langle Ciudad\rangle$

3. hayCamino: Un mapa de ciudades esta conformada por ciudades y caminos que unen a algunas de ellas. A partir de este mapa, podemos definir las distancias entre ciudades como una matriz donde cada celda i, j representa la distancia entre la ciudad i y la ciudad j (Fig. 2). Una distancia de 0 equivale a no haber camino entre i y j. Notar que la distancia de una ciudad hacia s´ı misma es cero y la distancia entre A y B es la misma que entre B y A.

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

4. cantidadCaminosNSaltos: Dentro del contexto de redes informaticas, nos interesa contar la cantidad de "saltos" que realizan los paquetes de datos, donde un salto se define como pasar por un nodo. Ası como definimos la matriz de distancias, podemos definir la matriz de conexion entre nodos, donde cada celda i, j tiene un 1 si hay un unico camino a un salto de distancia entre el nodo i y el nodo j, y un 0 en caso contrario. En este caso, se trata de una matriz de conexion de orden 1, ya que indica cu´ales pares de nodos poseen 1 camino entre ellos a 1 salto de distancia. Dada la matriz de conexion de orden 1, este procedimineto debe obtener aquella de orden n que indica cuantos caminos de n saltos hay entre los distintos nodos. Notar que la multiplicacion de una matriz de conexion de orden 1 consigo misma nos da la matriz de conexion de orden 2, y ası sucesivamente.4
proc cantidadNSaltos s (inout conexion: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, in n: $\mathbb{Z}$)

5. caminoMınimo: caminoMınimo: Dada una matriz de distancias, una ciudad de origen y una ciudad de destino, este procedimiento debe devolver la lista de ciudades que conforman el camino mas corto entre ambas. En caso de no existir un camino, se debe devolver una lista vacıa.
proc caminoMinimo (in origen: $\mathbb{Z}$, in destino: $\mathbb{Z}$, in distancias: seq⟨seq⟨$\mathbb{Z}$⟩⟩) : seq⟨$\mathbb{Z}$⟩

1.2. WP (Weakest Precondiction)

La funcion poblacionTotal recibe una lista de ciudades donde al menos una de ellas es grande (es decir, supera los 50.000 habitantes) y devuelve la cantidad total de habitantes. Dada la siguiente especificacion:

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

`requiere` $\{(\exists i : \mathbb{Z})\ (0 \leq i < |ciudades| \wedge_L ciudades[i].habitantes > 50,000 \wedge (\forall i : \mathbb{Z})\ (0 \leq i < |ciudades| \longrightarrow_L ciudades[i].habitantes \geq 0) \wedge (\forall i, j : \mathbb{Z})\ (0 \leq i < j \longrightarrow_L ciudades[i].nombre \neq ciudades[j].nombre))\}$

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

Con la siguiente implementacion:

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

- 1. Demostrar que la implementacion es correcta con respecto a la especificacion.
- 2. Demostrar que el valor devuelto es mayor a 50.000.

2. Predicados Reutilizables

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

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

}

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

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

}

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

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

}

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

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

}

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

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

}

`pred esCamino` (distancias : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$, c : $seq\langle\mathbb{Z}\rangle$, d : $\mathbb{Z}$, h : $\mathbb{Z}$) {

$(esMatrizCuadrada(distancias) \land |c| \geq 2) \land_L (\forall e : \mathbb{Z})\ (e \in c \longrightarrow_L 0 \leq e < |distancias| \land (c[0] = d \land c[|c|-1] = h) \land (\forall i : \mathbb{Z})\ (0 \leq i < |c| - 1 \longrightarrow_L distancias[c[i]][c[i+1]] > 0))$

}

3. Resolucion de Ejercicios

3.1. Especificacion

Ejercicio 1:

`proc grandesCiudades` (in ciudades : $seq\langle Ciudad\rangle$) : $seq\langle Ciudad\rangle$

`requiere` {true}

`asegura` $\{|res| = cantidadCiudadesGrandes(ciudades) \land sonTodasCiudadesGrandes(res) \land (\forall c : Ciudad)\ (c \in res \longrightarrow_L c \in ciudades)\}$

`pred sonTodasCiudadesGrandes` (ciudades : $seq\langle Ciudad\rangle$) {

$((\forall i : \mathbb{Z})\ (0 \leq i < |ciudades| \longrightarrow_L ciudades[i].habitantes > 50000))$

}

`aux cantidadCiudadesGrandes` (ciudades :$seq\langle Ciudad\rangle$) : $\mathbb{Z} = \sum_{i=0}^{|s|-1}$ (if $s[i].habitantes > 50000$ then 1 else 0 fi) ;

Ejercicio 2:

`proc sumaDeHabitantes` (in menoresDeCiudades : $seq\langle Ciudad\rangle$, in mayoresDeCiudad : $seq\langle Ciudad\rangle$) : $seq\langle Ciudad\rangle$

`requiere` $\{|menoresDeCiudades| = |mayoresDeCiudades| \land mismasCiudades(menoresDeCiudades, mayoresDeCiudades \land ciudadesDistintas(menoresDeCiudades) \land ciudadesDistintas(mayoresDeCiudades))\}$

`asegura` $\{|res| = |menoresDeCiudades| \land esLaSuma(res, menoresDeCiudades, mayoresDeCiudades) \land mismasCiudades(res, menoresDeCiudades) \land mismasCiudades(res, mayoresDeCiudades)\}$

`pred mismasCiudades` (s: $seq\langle Ciudad\rangle$, l: $seq\langle Ciudad\rangle$) {

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

}

`pred esLaSuma` (res : $seq\langle Ciudad\rangle$, s: $seq\langle Ciudad\rangle$, l: $seq\langle Ciudad\rangle$) {

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

}

`pred ciudadesDistintas` (ciudades : $seq\langle Ciudad\rangle$) {

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

}

Ejercicio 3:

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

`requiere` $\{esMatrizCuadrada(distancias) \land diagonalEnCeros(distancias) \land esMatrizSimetrica(distancias) \land distanciasValidas(distancias) \land (0 \leq desde < |distancias| \land 0 \leq hasta < |distancias|)\}$

`asegura` $\{res = \text{true} \longleftrightarrow (\exists c : seq\langle \mathbb{Z}\rangle)\ (esCamino(distancias, c, desde, hasta))\}$

Ejercicio 4:

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

`requiere` $\{conexion = conexion_0 \land esMatrizCuadrada(conexion) \land cerosEnLaDiagonal(conexion) \land esMatrizSimetrica(conexion) \land esMatrizConCerosYUnos(conexion)\}$

`asegura` $\{|conexion| = |conexion_0| \land_L (\forall i : \mathbb{Z})\ (0 \leq i < |conexion_0| \longrightarrow_L |conexion[i]| = |conexion_0| \land esMatrizDeOrdenN(conexion, conexion_0, n))\}$

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

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

}

`pred esIdentidad` (m : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(esMatrizCuadra) \land_L (\forall i, j : \mathbb{Z})\ (0 \leq i < |m| \land 0 \leq j < |m[i]| \longrightarrow_L ((i = j \land m[i][j] = 1) \lor (i \neq j \land m[i][j] = 0)))$

}

`pred esProducto` (m : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, n : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, o : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, n : $\mathbb{Z}$) {

$(\forall i, j : \mathbb{Z})\ (0 \leq i < |m| \land 0 \leq j < |m[i]| \longrightarrow_L m[i][j] = \sum_{k=0}^{|n|-1} n[i][k] * o[k][j])$

}

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

$(\exists lista : seq\langle seq\langle seq\langle \mathbb{Z}\rangle\rangle\rangle)\ ((|lista| = n+1 \land esIdentidad(lista[0]) \land lista[1] = l \land lista[n] = s) \land (\forall i : \mathbb{Z})\ (1 \leq i \leq n \longrightarrow_L (esProducto(lista[i], lista[i-1], lista[1]))))$

}

Ejericio 5:

`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` $\{(diagonalEnCeros(distancias) \land esMatrizCuadrada(distancias) \land esMatrizSimetrica(distancias) \land distanciasValidas(distancias) \land (0 \leq origen < |distancias| \land 0 \leq destino < |distancias|))\}$

`asegura` $\{(esCamino(res) \land_L (\forall c : seq\langle \mathbb{Z}\rangle)\ (esCamino(c) \longrightarrow_L distanciaRecorrida(distancias, res) \geq distanciaRecorrida(distancias, c))) \lor (res = [\,]) \longleftrightarrow \neg(\exists c : seq\langle \mathbb{Z}\rangle)\ (esCamino(distancias, c, origen, destino))\}$

`aux distanciaRecorrida` (distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, c : $seq\langle \mathbb{Z}\rangle$) : $\mathbb{Z} = \sum_{i=0}^{|c|-2} distancias[c[i]][c[i+1]]$;

4. WP (Weakest Precondition)

Primer item:
Para demostrar la correctitud de la implementación del programa con respecto a su especificación, debemos ver los siguientes puntos:

- 1. **Precondicion, guarda y postcondicion del ciclo**
- 2. **P** $\longrightarrow P_c \Longleftrightarrow P_c \longrightarrow \text{Wp}(\text{res} := 0; \text{Wp}(i = 0, P_c))$(La precondición de la especificación implica la precondición del ciclo.)

- 3. **Correctitud parcial del ciclo**
- 4. **Terminación del ciclo**

4.1. Precondicion, guarda y postcondicion del ciclo

Precondicion del ciclo:
$P_c \equiv \{(P \wedge i = 0 \wedge res = 0)\}$ (Donde P es la precondicion de la especificacion).

Guarda del ciclo:

$\mathbf{B} \equiv \{(i < |ciudades|)\}$

Postcondicion del ciclo:

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

Una vez definido la precondicion, guarda y postcondicion ahora veamos si P (la precondicion de la especificacion) implica la P_c (la precondicion de ciclo) eso lo vamos a hacer calculando la Wp(res := 0 ; Wp(i :=, P_c))

4.2. $P \longrightarrow P_c$

Calculo de la Wp(res := 0 ; Wp(i := 0, P_c)):
llamemos ① a Wp(i := 0, P_c) y ② Wp(res := 0, ①) (utilizo el aximo 1 de asignacion y axioma 2)