DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

TP de Especificación

Esperando el Bondi

30 de Marzo de 2022 — Algoritmos y Estructuras de Datos I

Grupo 4

Integrante	LU	Correo electrónico
Quintiero, Manuel	71/20	manu_quintiero@hotmail.com
Gangui, Matias	155/20	ganguimatias@gmail.com
Poggi, Octavio	810/21	octaviootih@gmail.com
Vega, Angela	585/21	luciavsm@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. Parte 1

1.1. Ejercicio 1

proc viajeValido (in v: Viaje, out res: Bool) {

Pre $\{True\}$

Post $\{esViajeValido(v) \leftrightarrow res = true\}$

}

pred esViajeValido (v: Viaje) {

$(\forall i : \mathbb{Z})(0 \leq i < |v| \longrightarrow_L (esValidoTiempo(v[i][0]) \wedge esValidoGPS(v[i][1])))$

}

pred esValidoTiempo (n:$\mathbb{R}$) {

$n \geq 0$

}

pred esValidoGPS (gps: GPS) {

$(-90{,}0 \leq gps[0] \leq 90{,}0) \wedge (-180{,}0 \leq gps[1] \leq 180{,}0)$

}

1.2. Ejercicio 2

proc recorridoValido (in v: Recorrido, out result: Bool) {

Pre $\{True\}$

Post $\{result = true \leftrightarrow esValidoRecorrido(v)\}$

}

pred esValidoRecorrido (v: Recorrido) {

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

}

1.3. Ejercicio 3

Dado un viaje valido, suponemos un centro de radio r tal que abarque dentro suyo todos los viajes.

proc enTerritorio (in v: Viaje, in r: Dist,out res: Bool) {

Pre $\{esViajeValido(v)\}$

Post $\{res = true \leftrightarrow (\exists\ centro : GPS)(esValidoGPS(centro) \wedge_L$
$(\forall i : \mathbb{Z})((0 \leq i < |v|) \longrightarrow_L dist(centro, v[i][1]) < r * 1000))\}$

}

1.4. Ejercicio 4

Para encontrar el tiempo total, se necesita buscar el tiempo máximo y el mínimo del viaje ingresado. Para esto, usamos los auxiliares tiempoMax y tiempoMin, que busca en el viaje un valor de tiempo que sea mayor o menor a todos los demás. Al tener esos valores, hacemos la resta y nos da el tiempo transcurrido entre ambos. Asumimos que no pueden haber dos tiempos iguales.

proc tiempoTotal (in v: Viaje, out t: Tiempo) {

Pre $\{esViajeValido(v)\}$

Post $\{(\exists\ ti : Tiempo, tf : Tiempo)((esMinTiempo(v, ti)\ \wedge\ esMaxTiempo(v, tf))\ \wedge\ t = tf - ti)\}$

}

pred esTiempoDelViaje (a: Tiempo, v: Viaje) {

$(\exists i : \mathbb{Z})(0 \leq i < |v| \wedge_L v[i][0] = a)$

}

pred esMinTiempo (v: Viaje, a: $\mathbb{R}$) {

$esTiempoDelViaje(a, v) \wedge (\forall i : \mathbb{Z})(0 \leq i < |s| \longrightarrow_L a \leq v[i][0])$

}

pred esMaxTiempo (v: Viaje, a: $\mathbb{R}$) {

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

}

1.5. Ejercicio 5

Ordenamos el viaje de menor a mayor con respecto al tiempo, y sumamos la distancia entre todos los puntos consecutivos.

proc distanciaTotal (in v: Viaje, out d: Dist) {

Pre { $esViajeValido(v) \wedge |v| > 1$}

Post {$(\exists v0 : Viaje)(esViajeOrdenado(v, v0) \wedge_L d = \sum_{i=0}^{|v|-2} distancia(v0[i][1], v0[i+1][1]))$}

}

pred esViajeOrdenado (v: Viaje, v0: Viaje) {

$|v| = |v0| \wedge estaOrdenadoPorTiempo(v0)$
$\wedge ((\forall e : TiempoxGps)(cantidadApariciones(e, v) = cantidadApariciones(e, v0)))$

}

pred estaOrdenadoPorTiempo (v: Viaje) {

$(\forall i : \mathbb{Z})(0 \leq i < |v| - 1 \longrightarrow_L v[i][0] < v[i+1][0])$

}

1.6. Ejercicio 6

Buscamos si existen dos puntos del viaje entre los cuales se supere la velocidad de 80km/h.

proc excesoDeVelocidad (in v: Viaje, out res: Bool) {

Pre {$esViajeValido(v) \wedge |v| > 1$}

Post {$res = true \leftrightarrow velocidadSupera80km/h(v)$}

}

pred velocidadSupera80km/h (in v: Viaje) {

$(\exists i, j : \mathbb{Z})((0 \leq i, j < |v| \wedge i \neq j) \longrightarrow_L (distancia(v[i][1], v[j][1])/(v[i][0] - v[j][0])) * 3,6 \geq 80)$

}

2. Parte 2

2.1. Ejercicio 7

El enRuta se fija si hay 1 tiempo registrado en el viaje que está en el intervalo [t0,tf], o que hay un tiempo menor que t0 y uno mayor a tf, lo cual implica que [t0,tf] está incluido dentro del intervalo de tiempo total del viaje.

`proc flota` (in v: $seq\langle Viaje\rangle$, in t0 : Tiempo, in tf : Tiempo, out res : $\mathbb{Z}$) {

`Pre` $\{(\forall i : \mathbb{Z})(0 \leq i \leq |v| - 1 \longrightarrow_L esViajeValido(v[i]))\}$

`Post` $\{res = \sum_{i=0}^{|v|-1} \text{if } enRuta(v[i], t0, tf) \text{ then } 1 \text{ else } 0 \text{ fi}\}$

}

`pred enRuta` (l: Viaje, t0: Tiempo, tf: Tiempo) {

$(\exists i : \mathbb{Z})(0 \leq i < |l| \wedge_L (t0 \leq l[i][0] \leq tf))$
$\vee\ (\exists j, k : \mathbb{Z})(0 \leq j, k < |l| \longrightarrow_L (l[j][0] \leq t0 \vee l[k][0] \geq tf))$

}

2.2. Ejercicio 8

`proc recorridoNoCubierto` (in v: Viaje, in r: Recorrido, in u: Dist. out res: $seq\langle GPS\rangle$) {

`Pre` $\{esViajeValido(v) \wedge recorridoValido(r)\}$

`Post` $\{(\forall i : \mathbb{Z})(0 \leq i < |r| \wedge_L (noCubierto(v, u, r[i]) \leftrightarrow r[i] \in res)) \wedge noHayRepetidos(res)\}$

}

`pred noHayRepetidos` (res: $seq\langle GPS\rangle$) {

$(\forall i, j : \mathbb{Z})((0 \leq i, j < |res| \wedge i \neq j) \longrightarrow_L res[i] \neq res[j])$

}

`pred noCubierto` (v: Viaje, u: Dist, n: GPS) {

$(\forall i : \mathbb{Z})(0 \leq i < |l| \longrightarrow_L dist(v[i][1], n) > u * 1000)$

}

Como u viene en kms, hay que convertirlo a metros multiplicandolo por 1000.

2.3. Ejercicio 9

Para el pred nombreCelda: Sabiendo la longitud y latitud esperadas de las celdas, buscamos la diferencia entre el final de la celda y el comienzo de la grilla y la dividimos por ese valor para encontrar los números del nombre de las celdas.

`proc construirGrilla` (in esq1: GPS, in esq2: GPS, in n: $\mathbb{Z}$, in m: $\mathbb{Z}$, out g: Grilla) {

`Pre` $\{n > 0 \wedge m > 0 \wedge ((esq1[1] < esq2[1]) \wedge (esq1[0] > esq2[0]) \wedge esValidoGPS(esq1) \wedge esValidoGPS(esq2))\}$

`Post` $\{noHayCeldasRepetidas(g)\ \wedge\ estanEnLaGrilla(g, esq1, esq2)\ \wedge\ |g| = n\ *\ m$
$\wedge\ celdasDelMismoTamaño(esq1, esq2, n, m, g)\ \wedge nombreCelda(esq1, esq2, n, m, g)\}$

}

`pred estanEnLaGrilla` (g: Grilla, esq1: GPS, esq2: GPS) {

$(\forall i : \mathbb{Z})(0 \leq i < |g| \longrightarrow_L (esq2[0] \leq g[i][1][0], g[i][0][0] \leq esq1[0] \wedge esq1[1] \leq g[i][1][1], g[i][0][1] \leq esq2[1]))$

}

`pred noHayCeldasRepetidas` (g : Grilla) {

$(\forall i, j : \mathbb{Z})((0 \leq i, j < |g| \wedge i \neq j)\ \longrightarrow_L g[i][2]\ \neq\ g[j][2])$

}

`pred celdasDelMismoTamaño` (esq1: GPS, esq2: GPS, n: $\mathbb{Z}$, m: $\mathbb{Z}$, g: Grilla) {

$(\forall i : \mathbb{Z})(0 \leq i \leq |g| - 1 \longrightarrow_L (largoCelda(esq1, esq2, m) = ||g[i][1][1] - g[i][0][1]|| \wedge$
$alturaCelda(esq1, esq2, n) = ||g[i][1][0] - g[i][0][0]||))$

}

`pred nombreCelda` (esq1: GPS, esq2: GPS, n: $\mathbb{Z}$, m: $\mathbb{Z}$, g: Grilla) {

$(\forall i : \mathbb{Z})(0 \leq i < |g| \longrightarrow_L (g[i][2][0] = ||(esq1[0] - g[i][1][0])||/alturaCelda(esq1, esq2, n) \wedge$
$(g[i][2][1] = ||(g[i][1][1] - esq1[1])||/largoCelda(esq1, esq2, m))))$

}

`aux alturaCelda` (esq1: GPS, esq2: GPS, n:$\mathbb{Z}$) : $\mathbb{R} = (esq1[0] - esq2[0])/n$;

`aux largoCelda` (esq1: GPS, esq2: GPS, m:$\mathbb{Z}$) : $\mathbb{R} = (esq2[1] - esq1[1])/m$;

2.4. Ejercicio 10

Para cada elemento de res, verificamos que las coordenadas de ese elemento de recorrido esté efectivamente en la celda que nos devuelve. Asumimos que la grilla ingresada es válida.

`proc regiones` (in r: Recorrido, in g: Grilla, out res: $seq\langle Nombre\rangle$) {

`Pre` $\{|r| \geq 1\}$

`Post` $\{|r| = |res| \wedge (\forall i : \mathbb{Z})(0 \leq i < |r| \longrightarrow_L seCorresponde(r[i], g, res[i]))\}$

}

`pred seCorresponde` (a: GPS, g: Grilla, n: Nombre) {

$estaEntreLasLatitudes(a, g, n) \wedge estaEntreLasLongitudes(a, g, n)$

}

`pred estaEntreLasLatitudes` (a: GPS, g: Grilla, n: Nombre) {

$(\exists i : \mathbb{Z})(0 \leq i < |g| \wedge_L g[i][2] = n \wedge_L (g[i][1][0] < a[0] \leq g[i][0][0]))$

}

`pred estaEntreLasLongitudes` (a: GPS, g: Grilla, n: Nombre) {

$(\exists i : \mathbb{Z})(0 \leq i < |g| \wedge_L g[i][2] = n \wedge_L (g[i][0][1] < a[0] \leq g[i][1][1]))$

}

2.5. Ejercicio 11

Asumimos que ningun viaje pasa perfectamente por la diagonal entre 2 celdas (es decir, consideramos que entre 2 celdas diagonales hay 2 celdas de distancia).

`proc cantidadDeSaltos` (in g: Grilla, in v: Viaje, out res : $\mathbb{Z}$) {

`Pre` $\{esGrillaValida(g) \wedge viajeEnGrilla(v, g) \wedge esViajeValido(v) \wedge |v| \geq 2\}$

`Post` $\{(\exists\, v0 : Viaje)(esViajeOrdenado(v, v0) \wedge_L res = \sum_{i=0}^{|v0|-2}(\text{if } seMueveDosOMasCeldas(v0, g, i) \text{ then } 1 \text{ else } 0 \text{ fi})\}$

}

`pred seMueveDosOMasCeldas` (v0: Viaje, g: Grilla, i: $\mathbb{Z}$) {

$(\exists\, c1, c2 : Nombre)((esCeldaDeLaGrilla(c1, g) \wedge esCeldaDeLaGrilla(c2, g)) \wedge_L$
$(seCorresponde(v0[i], g, c1) \wedge seCorresponde(v0[i+1], g, c2)) \wedge_L$
$(cuantasCeldasSeMovio(c1, c2) \geq 2))$

}

`pred viajeEnGrilla` (v: Viaje, g: Grilla) {

$(\exists\, g0 : Grilla)(mismaGrillaOrdenada(g, g0) \wedge (\forall i : \mathbb{Z})(0 \leq i < |v| \longrightarrow_L (g0[0][0][1] \leq v[i][1][1] \leq g0[|g0| - 1][1][1] \wedge$
$g0[|g0| - 1][1][0] \leq v[i][1][0] \leq g0[0][0][0])))$

}

Para todos los puntos GPS de viaje nos fijamos que estén entre la latitud y longitud de los bordes de las grillas, esto lo hicimos sabiendo que la esquina de arriba a la izquierda de la primera celda es "esq1" y la esquina de abajo a la derecha de la última celda es "esq2".

`pred esCeldaDeLaGrilla` (c: Nombre, g: Grilla) {

$(\exists\, i : \mathbb{Z})(0 \leq i < |g| \wedge_L g[i][2] = c)$

}

`pred esGrillaValida` (g: Grilla) {

$|g| > 0 \wedge_L (\exists\, g0 : Grilla)((mismaGrillaOrdenada(g, g0) \ \wedge \ (columnas(g0) > 0 \ \wedge \ filas(g0) > 0) \ \wedge$
$(|g0| = filas(g0) \ * \ columnas(g0))) \ \wedge_L ((\forall i : \mathbb{Z})(0 \leq i < |g0| \longrightarrow_L (esValidoGPS(g[i][0]) \ \wedge$
$esValidoGPS(g[i][1]))) \ \wedge \ (esquina1(g0)[1] < esquina2(g0)[1]) \ \wedge \ (esquina1(g0)[0] > esquina2(g0)[0]) \ \wedge$
$noHayCeldasRepetidas(g0) \ \wedge \ estanEnLaGrilla(g0, esquina1(g0), esquina2(g0)) \ \wedge$
$celdasDelMismoTamaño(esquina1(g0), esquina2(g0), filas(g0), columnas(g0), g0) \ \wedge$
$nombreCelda(esquina1(g0), esquina2(g0), filas(g0), columnas(g0), g0)))$

}

`pred esGrillaOrdenada` (g: Grilla) {

$(\forall i : \mathbb{Z})(0 \leq i < |g| - 1 \longrightarrow_L (g[i][2][0] = g[i+1][2][0] \ \wedge \ g[i][2][1] = g[i+1][2][1] - 1) \vee$
$(g[i+1][2][0] = g[i][2][0] + 1 \ \wedge \ g[i+1][2][1] = 1)$

}

`pred mismaGrillaOrdenada` (g: Grilla, g0: Grilla) {

$|g| = |g0| \ \wedge \ (\forall\, e : < GPS * GPS * Nombre >)(cantidadApariciones(g, e) = cantidadApariciones(g0, e)) \ \wedge$
$esGrillaOrdenada(g0)$

}

`aux esquina1` (g: Grilla) : GPS $= g[0][0]$;

`aux esquina2` (g: Grilla) : GPS $= g[|g| - 1][1]$;

`aux columnas` (g: Grilla) : $\mathbb{Z} = g[|g| - 1][2][1]$;

`aux filas` (g: Grilla) : $\mathbb{Z} = g[|g| - 1][2][0]$;

`aux cuantasCeldasSeMovio` (celdaInicio: Nombre, celdaFinal: Nombre) : $\mathbb{Z} =$
$||celdaInicio[0] - celdaFinal[0]|| + ||celdaInicio[1] - celdaFinal[1]||$;

Pedimos chequear los mismos requisitos de construirGrilla a base de una grilla ordenada (g0) que tiene los mismo elementos de g. 1) Ordenamos la grilla para sacar esquina1, esquina2, columnas y filas. 2) Con eso, pedimos las mismas propiedades que en el ej 9.

2.6. Ejercicio 12

Planteamos un existe para encontrar los índices de los dos puntos simultaneamente correctos y mas cercanos (temporalmente) al punto errado. Una vez conseeguidos los índices, exigimos las propiedades del GPS corregido y se lo pedimos a los errores en el viaje de la post condición.

`proc corregirViaje` (inout v: Viaje, in errores: $seq\langle Tiempo\rangle$) {

`Pre` $\{v = v0 \wedge esViajeValido(v) \wedge |v| > 5 \wedge |errores| \leq 0{,}1 * |v| \wedge$
$(\forall i : \mathbb{Z})(0 \leq i < |errores| \longrightarrow_L esTiempoDelViaje(errores[i], v))\}$

`Post` $\{|v| = |v0| \wedge (\forall i : \mathbb{Z})(0 \leq i < |v| \longrightarrow_L (v[i][0] = v0[i][0] \wedge esViajeCorregido(v, v0, errores, i)))\}$

}

`pred esViajeCorregido` (v, v1: Viaje, errores: $seq\langle Tiempo\rangle$, i: $\mathbb{Z}$) {

$((esUnError(v, errores, i) \wedge esPuntoCorregido(v[i], v, errores)) \vee ((\neg esUnError(v, errores, i) \wedge v[i][1] = v0[i][1])))$

}

pred esPuntoCorregido (a: Tiempo x GPS, v: Viaje, errores: $seq\langle Tiempo\rangle$) {

$(\exists i, j : \mathbb{Z})(0 \leq i, j < |v| \wedge j \neq i \wedge_L (\neg(esUnError(v, errores, i)) \wedge \neg(esUnError(v, errores, j))$
$\wedge\, tiempoEntrePuntos(a[0], v[i][0]) \leq tiempoEntrePuntos(a[0], v[j][0])$
$\wedge \neg(\exists k : Z)(0 \leq k < |v| \wedge k \neq i \wedge k \neq j \wedge \neg(esUnError(v, errores, k))$
$\wedge_L\, tiempoEntrePuntos(a[0], v[j][0]) > tiempoEntrePuntos(a[0], v[k][0]))$
$\wedge_L\, esNuevoGPS(v, i, j, a)))$

}

pred esNuevoGPS (v: Viaje, i: $\mathbb{Z}$, j: $\mathbb{Z}$, puntoCorregido: Tiempo x GPS)) {

$esValidoGPS(puntoCorregido[1]) \wedge_L (dist(puntoCorregido[1], v[i][1]) =$
$velocidadMedia(v[i][1], v[j][1]) * tiempoEntrePuntos(puntoCorregido[0], v[i][0])$
$\wedge\, dist(puntoCorregido[1], v[j][1]) = velocidadMedia(v[i][1], v[j][1]) * tiempoEntrePuntos(puntoCorregido[0], v[j][0]))$

}

pred esUnError (v1: Viaje, errores: $seq\langle Tiempo\rangle$, j: $\mathbb{Z}$) {

$(\exists i : \mathbb{Z})(0 \leq i < |errores| \wedge_L v[j][0] = errores[i])$

}

aux velocidadMedia (a: Tiempo x GPS, b: Tiempo x GPS) : $\mathbb{R} = ||dist(a[1], b[1])/(a[0] - b[0])||$;

aux tiempoEntrePuntos (a: Tiempo x GPS, b: Tiempo x GPS) : $\mathbb{R} = ||a[0] - b[0]||$;

2.7. Ejercicio 13

proc histograma (in xs: $seq\langle Viaje\rangle$, in bins: $\mathbb{Z}$, out cuentas: $seq\langle\mathbb{Z}\rangle$, limites: $seq\langle\mathbb{R}\rangle$) {

Pre $\{bins > 1 \wedge |xs| > 1 \wedge (\forall i : \mathbb{Z})(i \in xs \longrightarrow_L esViajeValido(xs[i]))\}$

Post $\{|limites| = bins + 1 \wedge |cuentas| = bins \wedge ((\exists vMaximas : seq\langle\mathbb{R}\rangle)((|vMaximas| = |xs|) \wedge$
$(\forall i : \mathbb{Z})(0 \leq i \leq |vMaximas| \longrightarrow_L esVelocidadMaxima(vMaximas[i], xs[i])$
$\wedge\, esLimite(vMaximas, limites, bins) \wedge esCuentas(vMaximas, limites, cuentas)))$
}

}

pred esCuentas (vMaximas: $seq\langle\mathbb{R}\rangle$, limites: $seq\langle\mathbb{R}\rangle$, cuentas: $seq\langle\mathbb{Z}\rangle$) {

$(\forall i : \mathbb{Z})((0 \leq i < |cuentas| - 1 \longrightarrow_L cuentas[i] = estanEnElIntervalo(vMaximas, limites[i], limites[i + 1]))) \wedge$
$cuentas[|cuentas| - 1] = estanEnElIntervaloCerrado(vMaximas, limites[|cuentas| - 1], limites[|cuentas|])$

}

pred esLimites (vMaximas: $seq\langle\mathbb{R}\rangle$, limites: $seq\langle\mathbb{R}\rangle$, bins: $\mathbb{Z}$) {

$(\exists\, a, b, intervalo : \mathbb{R})((esMenor(vMaximas, a) \wedge esMayor(vMaximas, b) \wedge intervalo = (b - a)/bins) \wedge_L (\forall i : \mathbb{Z})((0 \leq$
$i < |limites| \longrightarrow_L (i = 0 \wedge esMenor(vMaximas, limites[i])) \vee$
$(i \neq 0 \wedge limites[i] = limites[0] + intervalo * i))))$

}

pred esVelocidadMaxima (velocidad0: $\mathbb{R}$, v: Viaje) {

$\neg(\exists i, j : \mathbb{Z})((0 \leq i, j < |v| \wedge i \neq j) \wedge_L velocidad(v[i][1], v[j][1], v[i][0], v[j][0]) > velocidad0)$

}

pred esMayor (s: $seq\langle\mathbb{R}\rangle$, a: $\mathbb{R}$) {

$a \in s \wedge (\forall i : Z)(0 \leq i < |s| \longrightarrow_L a \geq s[i])$

}

pred esMenor (s: $seq\langle\mathbb{R}\rangle$, a: $\mathbb{R}$) {

$a \in s \wedge (\forall i : Z)(0 \leq i < |s| \longrightarrow_L a \leq s[i])$

}

aux velocidad (gps1: GPS, gps2: GPS, t0: Tiempo, tf: Tiempo) : $\mathbb{R} = distancia(gps1, gps2)/(tf - t0)$;

`aux estanEnElIntervalo` (vMaximas: $seq\langle\mathbb{R}\rangle$, limite1: $\mathbb{R}$, limite2: $\mathbb{R}$) : $\mathbb{R}$ = $\sum_{i=0}^{|vMaximas|-1}$ if $limite1 \leq vMaximas[i] < limite2$ then 1 else 0 fi ;

`aux estanEnElIntervaloCerrado` (vMaximas: $seq\langle\mathbb{R}\rangle$, limite1: $\mathbb{R}$, limite2: $\mathbb{R}$) : $\mathbb{R}$ = $\sum_{i=0}^{|vMaximas|-1}$ if $limite1 \leq vMaximas[i] \leq limite2$ then 1 else 0 fi ;