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 1

Integrante	LU	Correo electrónico
Polonuer, Joaquin	1612/21	jtpolonuer@gmail.com
González, Facundo	1440/21	facundo2gonzalez2@gmail.com
Jaime, Brian David	411/18	brian.d.jaime97@gmail.com
Guberman, Diego	469/17	diego98g@hotmail.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 $Tiempo = \mathbb{R}$

type $Dist = \mathbb{R}$

type $GPS = \mathbb{R} \times \mathbb{R}$

type $Recorrido = seq\langle GPS \rangle$

type $Viaje = seq\langle Tiempo \times GPS \rangle$

type $Nombre = \mathbb{Z} \times \mathbb{Z}$

type $Grilla = seq\langle GPS \times GPS \times Nombre \rangle$

type $Celda = GPS \times GPS \times Nombre$

2. Constantes

3. Problemas

3.1. Ejercicio 1

Devolver verdadero si los puntos GPS del viaje y los tiempos están en rango.

```
proc viajeValido (in v: Viaje, out res: Bool) {
    Pre {True}
    Post {res = true ↔ esViajeValido(v)}
    pred esViajeValido (v: Viaje) {
        (∀i : ℤ)(0 ≤ i < |v| ⟶_L (esTiempoValido(v[i]_0) ∧ sonCoordenadasValidas(v[i]_1)))
    }
    pred esTiempoValido (t: Tiempo) {
        t ≥ 0
    }
    pred sonCoordenadasValidas (c: GPS) {
        −90,0 ≤ c_0 ≤ 90,0 ∧ −180,0 ≤ c_1 ≤ 180,0
    }
}
```

3.2. Ejercicio 2

Devolver verdadero si los puntos GPS del recorrido están en rango.

```
proc recorridoValido (in v: Recorrido, out res: Bool) {
    Pre {True}
    Post {res = true ↔ esRecorridoValido(v)}
    pred esRecorridoValido (v: Recorrido) {
        (∀i : ℤ)(0 ≤ i < |v| ⟶_L sonCoordenadasValidas(v[i]))
    }
}
```

3.3. Ejercicio 3

Chequear que todos los puntos registrados en un viaje válido se encuentren dentro de un círculo de radio r kilómetros.

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

`Pre` $\{esViajeValido(v)\}$

`Post` $\{res = \text{true} \leftrightarrow estaEnTerritorio(v, r)\}$

`pred estaEnTerritorio` (v: *Viaje*, r: *Dist*) {

$(\exists c : GPS)(sonCoordenadasValidas(c) \wedge_L (\forall i : \mathbb{Z})(0 \leq i < |v| \longrightarrow_L dist(c, v[i]_1) \leq 1000 \cdot r))$
/* Multiplico r por 1000 dado que r está dado en kilómetros y la función auxiliar $dist(p1, p2)$ devuelve su resultado en metros */

}

}

3.4. Ejercicio 4

Dado un viaje válido, determinar el tiempo total que tardó el colectivo. Este valor debe ser calculado como el tiempo transcurrido desde el primer punto registrado y hasta el último.

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

`Pre` $\{esViajeValido(v)\}$

`Post` $\{esMaximaDiferenciaTiempo(v, t)\}$

`pred esMaximaDiferenciaTiempo` (v: *Viaje*, t: *Tiempo*) {

$/* \ tesladiferenciaentredostiemposdelv \ */$
$(\exists i, j : \mathbb{Z})(0 \leq i, j < |v| \wedge_L v[i]_0 - v[j]_0 = t) \wedge$
$/* \ teslamayordiferenciaposibleentredostiemposdelviaje \ */$
$\neg(\exists n, m : \mathbb{Z})(0 \leq n, m < |v| \wedge_L v[n]_0 - v[m]_0 > t)$

}

}

3.5. Ejercicio 5

Dado un viaje válido, determinar la distancia recorrida en kilómetros aproximada utilizando toda la información registrada en el viaje, es decir, utilizando la información registrada de todos los tramos.

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

`Pre` $\{esViajeValido(v)\}$

`Post` $\{distanciaViajeOrdenado(v, d)\}$

`pred distanciaViajeOrdenado` (v: *Viaje*, d: *Dist*) {

$(\exists v' : Viaje)(esElViajeOrdenado(v, v') \land d = sumaDistanciasSucesivas(v'))$

}

`pred esElViajeOrdenado` (v,v': *Viaje*) {

$estaOrdenadoTemporalmente(v') \land esPermutacion(v, v')$

}

`pred estaOrdenadoTemporalmente` (v: *Viaje*) {

$(\forall i : \mathbb{Z})(0 \leq i < |v| - 1 \longrightarrow_L v[i]_0 < v[i+1]_0)$

}

`pred esPermutacion` (v1,v2: *Viaje*) {

$(\forall e : Tiempo \times GPS)(apariciones(v1, e) = apariciones(v2, e))$

}

`aux apariciones` (v: *Viaje*, e: $Tiempo \times GPS$) : $\mathbb{Z} = \sum_{i=0}^{|v|-1}$ if $v[i] = e$ then 1 else 0 fi ;

`aux sumaDistanciasSucesivas` (v: *Viaje*) : *Dist* $= \frac{1}{1000} \cdot \sum_{i=0}^{|v|-2} dist(v[i]_1, v[i+1]_1)$

/* Divido la sumatoria por 1000 dado que se pide el resultado en kilómetros y la función auxiliar $dist(p1, p2)$ devuelve su resultado en metros */ ;

}

3.6. Ejercicio 6

Dado un viaje válido devolver verdadero si el colectivo superó los 80 km/h en algún momento del viaje.

```
proc excesoDeVelocidad (in v: Viaje, out res: Bool) {
    Pre {esViajeValido(v)}
    Post {res = true ↔ superaVelocidad(v)}
    pred superaVelocidad (v: Viaje) {
        (∃i, j : ℤ)(0 ≤ i, j < |v| ∧_L esTramo(v, v[i], v[j]) ∧ velocidadTramo(v[i], v[j]) > 80)
    }
    pred esTramo (v: Viaje, e1,e2: Tiempo × GPS) {
        e1_0 < e2_0 ∧ ¬(∃e : Tiempo × GPS)(e ∈ v ∧ e1_0 < e_0 < e2_0)
    }
    aux velocidadTramo (e1,e2 : Tiempo × GPS) : ℝ = dist(e1_1,e2_1) / (e2_0 − e1_0) · 3,6
    /* Multiplico por 3,6 dado que se pide el resultado en kilómetros por hora y la función auxiliar
    dist(p1, p2) devuelve su resultado en metros mientras que los tiempos están en segundos */ ;
}
```

```
proc excesoDeVelocidad (in v: Viaje, out res: Bool) {
    Pre {esViajeValido(v)}
    Post {res = true ↔ superaVelocidad(v)}
    pred superaVelocidad (v: Viaje) {
        (∃v' : Viaje)(esElViajeOrdenado(v, v') ∧ viajeOrdenadoSuperaVelocidad(v'))
    }
    pred viajeOrdenadoSuperaVelocidad (v: Viaje) {
        (∃i : ℤ)(0 ≤ i < |v| − 1 ∧_L velocidadTramo(v[i], v[i + 1]) > 80)
    }
    aux velocidadTramo (e1,e2 : Tiempo × GPS) : ℝ = dist(e1_1,e2_1) / (e2_0 − e1_0) · 3,6
    /* Multiplico por 3,6 dado que se pide el resultado en kilómetros por hora y la función auxiliar
    dist(p1, p2) devuelve su resultado en metros mientras que los tiempos están en segundos */ ;
}
```

3.7. Ejercicio 7

Dada una lista de viajes válidos, calcular la cantidad de viajes que se encontraban en ruta en cualquier momento entre t_0 y t_f inclusives. Por ejemplo, si un viaje comenzó a las 13:30 y terminó a las 14:30 y la franja es de 14:00 a 15:00, el viaje debería estar considerado. Lo mismo ocurre si el viaje comenzó a las 14:10 y terminó a las 14:15 o si comenzó a las 13:30 y terminó a las 16:00.

`proc flota` (in vs: $seq\langle Viaje\rangle$, in t_0: *Tiempo*, in t_f: *Tiempo*, out res: $\mathbb{Z}$) {

`Pre` $\{sonTodosViajesValidos(vs) \land t_0 \leq t_f \land esTiempoValido(t_0) \land esTiempoValido(t_f)\}$

`Post` $\{esCantidadEnRuta(vs, t_0, t_f, res)\}$

`pred sonTodosViajesValidos` (vs: $seq\langle Viaje\rangle$) {

$(\forall v : Viaje)(v \in vs \longrightarrow_L esViajeValido(v))$

}

`pred esCantidadEnRuta` (vs: $seq\langle Viaje\rangle$, t_0,t_f: *Tiempo*, res: $\mathbb{Z}$) {

$(\exists vs' : seq\langle Viaje\rangle)$ (

$(\forall v : Viaje)$ (

$(v \in vs \land estaEnRuta(v, t_0, t_f)) \longrightarrow_L \#apariciones(v, vs') = \#apariciones(v, vs)$

) $\land |vs'| = res$

)

}

`pred estaEnRuta` (v: *Viaje*, t_0,t_f: *Tiempo*) {

$(\exists i, j : \mathbb{Z})$ (

$0 \leq i, j < |v| \land_L v[i]_0 \leq t_0 < t_f \leq v[j]_0$

) $\lor$ $(\exists i : \mathbb{Z})$ (

$0 \leq i < |v| \land_L t_0 \leq v[i]_0 \leq t_f$

)

}

`pred estaEnRuta` (v: *Viaje*, t_0,t_f: *Tiempo*) {

$(\exists i, j : \mathbb{Z})(0 \leq i \leq j < |v| \land_L (v[i]_0 \leq t_f \land v[j]_0 \geq t_0))$

}

}

3.8. Ejercicio 8

Dado un viaje v válido, un recorrido r válido y un umbral u (en kilómetros), devolver todos los puntos del recorrido que no fueron cubiertos por ningún punto del viaje. Se considera que un punto p del recorrido está cubierto si al menos un punto del viaje está a menos de u kilómetros del punto p.

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

`Pre` $\{esViajeValido(v) \land u > 0 \land esRecorridoValido(r)\}$

`Post` $\{sonTodosLosPuntosNoCubiertos(res, v, r, u)\}$

`pred sonTodosLosPuntosNoCubiertos` (res: $seq\langle GPS \rangle$, v: *Viaje*, r: *Recorrido*, u: *Dist*) {

/*Todos los puntos que están en res son puntos no cubiertos del recorrido*/
/*Todos los puntos no cubiertos del recorrido están en res*/
$(\forall p : GPS)$ (

$p \in res \leftrightarrow (p \in r \land \neg estaCubierto(p, v, u))$

)

}

`pred estaCubierto` (p: *GPS*, v: *Viaje*, u: *Dist*) {

$(\exists m : Tiempo \times GPS)$ (

$m \in v \land_L dist(m_1, p) < u \cdot 1000$

)

}

}

3.9. Ejercicio 9

Que dados dos puntos GPS, construye una grilla de n × m.

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

`Pre` $\{sonEsquinasValidas(esq1, esq2) \land n > 0 \land m > 0\}$

`Post` $\{esGrillaCorrecta(esq1, esq2, n, m, g)\}$

`pred sonEsquinasValidas` (esq1,esq2: *GPS*) {

$sonCoordenadasValidas(esq1) \land sonCoordenadasValidas(esq2) \land esq1_0 > esq2_0 \land esq1_1 < esq2_1$

}

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

$|g| = m \cdot n \land esquinasSonCombLineales(esq1, esq2, n, m, g)$

}

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

$(\forall a, b : \mathbb{Z})$ (

$(1 \leq a \leq n \land 1 \leq b \leq m) \longrightarrow_L (\exists i : \mathbb{Z})$ (

$0 \leq i < |g| \land_L$
/* Esquina superior izquierda*/
$g[i]_0 = esqSupIzq(a, b, n, m, esq1, esq2) \land$
/* Esquina inferior derecha */
$g[i]_1 = esqInfDer(a, b, n, m, esq1, esq2) \land$
/* Nombre */
$g[i]_2 = nombre(a, b)$

)

)

}

`aux esqSupIzq` (a,b,n,m: $\mathbb{Z}$, esq1,esq2: *GPS*) : *GPS* $= (esq1_0 - (a-1) \cdot (tamanoCelda(esq1, esq2, n, m))_0,$ $esq1_1 + (b-1) \cdot tamanoCelda(esq1, esq2, n, m)_1)$;

`aux esqInfDer` (a,b,n,m: $\mathbb{Z}$, esq1,esq2: *GPS*) : *GPS* $= (esq1_0 - a \cdot (tamanoCelda(esq1, esq2, n, m))_0,$ $esq1_1 + b \cdot tamanoCelda(esq1, esq2, n, m)_1)$;

`aux nombre` (a,b: $\mathbb{Z}$) : *Nombre* $= (a, b)$;

`aux tamanoCelda` (esq1,esq2: *GPS*, n,m:$\mathbb{Z}$) : $\mathbb{R} \times \mathbb{R} = (\frac{esq1_0 - esq2_0}{n}, \frac{esq2_1 - esq1_1}{m})$;

}

3.10. Ejercicio 10

Que dado un recorrido, devuelve la secuencia ordenada de regiones visitadas por el colectivo.

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

`Pre` $\{esRecorridoValido(r) \land esGrillaDelRecorrido(g, r)\}$

`Post` $\{esSecuenciaDelRecorrido(res, r, g)\}$

`pred esSecuenciaDelRecorrido` (res: $seq\langle Nombre\rangle$, r: *Recorrido*, g:*Grilla*) {

$$|res| = |r| \land (\forall i : \mathbb{Z})\ (\ 0 \leq i < |res| \longrightarrow_L (\exists c : Celda)\ (\ c \in g \land_L (nombre((c_2)_0, (c_2)_1) = res[i] \land estaEnCelda(r[i], c))\ )\ )$$

}

`pred estaEnCelda` (gps: *GPS*, c: *Celda*) {

$$(((c)_1)_0 \leq (gps)_0 \leq ((c)_0)_0) \land (((c)_0)_1 \leq (gps)_1 \leq ((c)_1)_1)$$

}

`pred esGrillaDelRecorrido` (g: *Grilla*, r: *Recorrido*) {

$$(\forall i : \mathbb{Z})\ (\ 0 \leq i < |r| \longrightarrow_L (\exists c : Celda)\ (\ c \in g \land_L estaEnCelda(r[i], c)\ )\ )$$

}

}

3.11. Ejercicio 11

Que dado un viaje valido y una grilla, determine cuantos saltos hay en el viaje

proc cantidadDeSaltos (in g: *Grilla*, in v: *Viaje*, out res: $seq\langle\mathbb{Z}\rangle$) {

Pre $\{esViajeValido(v) \land esGrillaDelViaje(g, v)\}$

Post $\{esCantidadDeSaltos(g, v, res)\}$

pred esCantidadDeSaltos (g: *Grilla*, v: *Viaje*, res: $\mathbb{Z}$) {

$(\exists v' : Viaje)$ (

$esElViajeOrdenado(v', v) \land (\exists R : seq\langle Nombre\rangle)$ (

$esSecuenciaDelViaje(R, v', g) \land cantidadDeSaltos(R) = res$

)

)

}

aux cantidadDeSaltos (R: $seq\langle Nombre\rangle$) : $\mathbb{Z}$ =
$\sum_{i=0}^{|v|-1}$ if $esCeldaContigua(R[i], R[i+1])$ then 0 else 1 fi ;

aux esCeldaContigua (n1: *Nombre*, n2: *Nombre*) : Bool =
if $(|n1_0 - n2_0| \leq 1 \land |n1_1 - n2_1| \leq 1)$ then $true$ else $false$ fi ;

pred esGrillaDelViaje (g: *Grilla*, v: *Viaje*) {

$(\forall i : \mathbb{Z})$ (

$0 \leq i < |v| \longrightarrow_L (\exists c : Celda)$ (

$c \in g \land_L estaEnCelda((v[i])_1, c)$

)

)

}

}

3.12. Ejercicio 12

Se cuenta con un viaje valido de mas de 5 puntos, y la lista errores que indica cada momento para el cual el valor registrado por el GPS fue erroneo y que debe ser corregido automaticamente

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

Pre $\{|v| > 5 \land esViajeValido(v) \land sonTiemposValidos(errores)$
$\land\, 10 * |errores| < |v| \land primeroYUltimoSinErrores(v, errores) \land v = v_0\}$

Post $\{esViajeCorregido(v, v_0, errores)\}$

pred sonTiemposValidos (e: $seq\langle Tiempo\rangle$) {

$(\forall i : \mathbb{Z})$ (

$0 \leq i < |e| \longrightarrow_L esTiempoValido(i)$

)

}

pred esViajeCorregido{

}

}

3.13. Ejercicio 13

Que dada una lista de viajes válidos, calcule el histograma de velocidades máximas registradas entre todos los viajes.

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

Pre $\{sonViajesValidos(xs) \land bins > 0\}$

Post $\{sonCuentasCorrectas(xs, bins, limites) \land sonLimitesCorrectos(limites, xs, bins)\}$

pred sonCuentasCorrectas (xs: $seq\langle Viaje\rangle$, bins: $\mathbb{Z}$, limites: $seq\langle\mathbb{R}\rangle$) {

$|cuentas| = bins \land$
$(\exists vels : seq\langle\mathbb{R}\rangle)$ (

$sonVelocidadesMaximas(vels, xs) \land$
$((\forall i : \mathbb{Z})$ (

$(0 \leq i < |cuentas| - 1) \longrightarrow_L cuentas[i] = cantEnIntervalo(vels, limites[i], limites[i+1])$

) $\land$
$(cuentas[|cuentas| - 1] = cantEnIntervaloCerrado(vels, limites[|cuentas| - 1], limites[|cuentas|]))$

)

}

pred sonLimitesCorrectos (limites: $seq\langle\mathbb{R}\rangle$, xs: $seq\langle Viaje\rangle$, bins: $\mathbb{Z}$) {

$|limites| = bins + 1 \land estaOrdenado(limites) \land$
$(\exists vels : seq\langle\mathbb{R}\rangle)$ (

$sonVelocidadesMaximas(vels, xs) \land$
$(\forall i : \mathbb{Z})$ (

$(0 \leq i < |limites|) \longrightarrow_L (\exists min, max : \mathbb{R})$ (

$(esMinimo(vels, min) \land esMaximo(vels, max)) \land limites[i] = min + (i \times \frac{max - min}{bins})$

)

)

)

}

pred sonVelocidadesMaximas (vels: $seq\langle\mathbb{R}\rangle$, xs: $seq\langle Viaje\rangle$) {

}

aux cantEnIntervalo (vels: $seq\langle\mathbb{R}\rangle$, lim1: $\mathbb{R}$, lim2: $\mathbb{R}$) : $\mathbb{R}$ =

;

aux cantEnIntervaloCerrado (vels: $seq\langle\mathbb{R}\rangle$, lim1: $\mathbb{R}$, lim2: $\mathbb{R}$) : $\mathbb{R}$ =

;

pred esMinimo (vels: $seq\langle\mathbb{R}\rangle$, min: $\mathbb{R}$) {

$min \in vels \land (\forall i : \mathbb{Z})$ (

$0 \leq i < |vels| \longrightarrow_L min \leq vels[i]$

)

}

pred esMaximo (vels: $seq\langle\mathbb{R}\rangle$, max: $\mathbb{R}$) {

$max \in vels \land (\forall i : \mathbb{Z})$ (

$0 \leq i < |vels| \longrightarrow_L vels[i] \leq max$

)

}

}