DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

TP de Especificación

Esperando el Bondi

27 de mayo de 2022 — Algoritmos y Estructuras de Datos I

Grupo 2

Integrante	LU	Correo electrónico
Carrasco, Andrés Nicolas	1905/21	ncarrasco@dc.uba.ar
Braginski, Leonel	385/21	leobraginski@gmail.com
Moraut, Tobias	1507/21	tmoraut@dc.uba.ar
Harburguer, Gabriel	1461/21	gabrielharburguer@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 $Tiempo = \mathbb{R}$

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

2. Predicados

`pred gpsEnRango` (g: *GPS*) {

$(-90 \leq g_0 \leq 90) \wedge (-180 \leq g_1 \leq 180)$

}

//tomo como tiempo valido cualquiera que sea despues de la fecha de inicio (llamo 0 a la fecha 1/1/1970)

`pred tiempoValido` (t: *Tiempo*) {

$t \geq 0$

}

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

$(\forall x : \mathbb{Z})(0 \leq i < |v| \longrightarrow_L (tiempoValido((v[i])_0) \wedge gpsEnRango((v[i])_1))) \wedge_L |v| \geq 2$

}

`pred sonViajesValidos` (v: $seq\langle Viaje \rangle$) {

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

}

`pred esRecorridoValido` (r: *Recorrido*) {

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

}

`pred viajeMismosElementos` (s,t: *Viaje*) {

$|s| = |t| \wedge (\forall e : (Tiempo \times GPS))((e \in s \leftrightarrow e \in t) \wedge aparicionesEnViaje(s, e) = aparicionesEnViaje(t, e))$

}

`pred grillaMismosElementos` (g, t: *Grilla*) {

$|g| = |t| \wedge (\forall e : (GPS \times GPS \times Nombre))((e \in g \leftrightarrow e \in t) \wedge aparicionesEnGrilla(g, e) = aparicionesEnGrilla(t, e))$

}

`pred viajeOrdPorTiempo` (s: *Viaje*) {

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

}

`pred mismoViajeOrdPorTiempo` (v, f: *Viaje*) {

$viajeOrdPorTiempo(f) \wedge viajeMismosElementos(v, f)$

}

`pred puntoEnGrilla` (p: *GPS*, g: *Grilla*) {

$(\exists i : \mathbb{Z})(0 \leq i < |g| \wedge_L ((g[i]_0)_0 \geq p_0 \geq (g[i]_1)_0 \wedge (g[i]_0)_1 \leq p_1 \leq (g[i]_1)_1))$

}

`pred mismaGrillaOrdPorEsquina` (g, f: *Grilla*) {

$grillaMismosElementos(g, f) \land grillaConEsquinasOrdenadas(f)$

}

`pred grillaConEsquinasOrdenadas` (g: *Grilla*) {

$esEsq1(g[0]_0) \land esEsq2(g[|g| - 1]_1)$

}

`pred esEsq1` (p: *GPS*, g: *Grilla*) {

$(\forall i : \mathbb{Z})(0 \leq i < |g| \longrightarrow_L p_0 \geq (g[i]_0)_0 \land p_1 \leq (g[i]_0)_1)$

}

`pred esEsq2` (p: *GPS*, g: *Grilla*) {

$(\forall i : \mathbb{Z})(0 \leq i < |g| \longrightarrow_L p_0 \leq (g[i]_1)_0 \land p_1 \geq (g[i]_1)_1)$

}

`pred grillaValida` (g: *Grilla*) {

$esquinasCorrectas(g) \land mismaLongYLatEnCeldas(g) \land nombresValidos(g)$

}

`pred esquinasCorrectas` (g: *Grilla*) {

$(\exists f : Grilla)(mismaGrillaOrdPorEsquina(g, f) \land_L esq1(g)_0 > esq2(g)_0 \land esq1(g)_1 < esq2(g)_1)$

}

`pred mismaLongYLatEnCeldas` (g: *Grilla*) {

$(\forall i : \mathbb{Z})(0 \leq i < |g| - 1 \longrightarrow_L ((g[i]_0)_0 - (g[i]_1)_0 > 0 \land (g[i]_0)_0 - (g[i]_1)_0 = (g[i+1]_0)_0 - (g[i+1]_1)_0) \land ((g[i]_1)_1 - (g[i]_0)_1 > 0 \land (g[i]_1)_1 - (g[i]_0)_1 = (g[i+1]_1)_1 - (g[i+1]_0)_1))$

}

`pred nombresValidos` (g: *Grilla*) {

$(\forall i : \mathbb{Z})(0 \leq i < |g| \longrightarrow_L nombreCorrespondeACelda(g[i]_0, g[i]_2, g))$

}

`pred pEntreFilai` (p, esq1: *GPS*, latxCelda: $\mathbb{R}$, i: $\mathbb{Z}$) {

$esq1_0 - (latxCelda * (i - 1)) \geq p_0 \geq esq1_0 - (latxCelda * i)$

}

`pred pEntreColumnai` (p, esq1: *GPS*, longxCelda: $\mathbb{R}$, i: $\mathbb{Z}$) {

$esq1_1 + (longxCelda * (i - 1)) \leq p_1 \leq esq1_1 + (longxCelda * i)$

}

Alcanza con trabajar con la grilla ordenada por esquinas ya que al tenerla ordenada de esta forma sabemos que el ultimo elemento de la grilla nos indica la cantidad de columnas y filas y podemos calcular la latitud y longitud de cada celda (igual en todas)

`pred nombreCorrespondeACelda` (p: *GPS*, nombre: *Nombre*, g: *Grilla*) {

$(\exists f : Grilla)(mismaGrillaOrdPorEsquina(g, f) \land_L nombre_0 = filaDePEnGrilla(p, f) \land nombre_1 = colDePEnGrilla(p, f))$

}

3. Auxiliares

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

`aux aparicionesEnGrilla` (s: *Grilla*, e: (*GPS*$\times$*GPS*$\times$ *Nombre*)) : $\mathbb{Z} = \sum_{i=0}^{|s|-1}$(if $s[i] = e$ then 1 else 0 fi);

la velocidad se calcula dividiendo la distancia recorrida (metros) por el tiempo transcurrido (segundos), como esta

velocidad esta medida esta en m/s, se pasa a km/h multiplicándola por 3600 seg y dividiendo por 1000 metros, de esta forma se multiplica a la velocidad por 3,6 ($\frac{3600}{1000}$)

`aux velocidad` (g1, g2: *Tiempo* × *GPS*) : $\mathbb{R} = \frac{dist((g1)_1,(g2)_1)}{(g2)_0-(g1)_0} * 3{,}6$;

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

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

`aux cantFilas` (g: *Grilla*) : $\mathbb{Z} = (g[|g|-1]_2)_0$;

`aux cantCols` (g: *Grilla*) : $\mathbb{Z} = (g[|g|-1]_2)_1$;

`aux esq1` (g: *Grilla*) : $GPS = g[0]_0$;

`aux esq2` (g: *Grilla*) : $GPS = g[|g|-1]_1$;

`aux filaDePEnGrilla` (p: *GPS*, g: *Grilla*) : $\mathbb{Z} = \sum_{i=1}^{cantFilas(g)}$
(if $pEntreFilai(p, esq1(g), latxCelda(esq1(g), esq2(g)), cantfilas(g), i))$ then i else 0 fi)) ;

`aux colDePEnGrilla` (p: *GPS*, g: *Grilla*) : $\mathbb{Z} = \sum_{i=1}^{cantCols(g)}$
(if $pEntreColumnai(p, esq1(g), longxCelda(esq1(g), esq2(g)), cantCols(g), i))$ then i else 0 fi)) ;

4. Problemas

4.1. Ejercicio 1

```
proc viajeValido (in v: Viaje, inout res: Bool) {
    Pre {true}
    Post {res = true ↔ esViajeValido(v)}
}
```

4.2. Ejercicio 2

```
proc recorridoValido (in v: Recorrido, out res: Bool) {
    Pre {true}
    Post {res = true ↔ esRecorridoValido(v)}
}
```

4.3. Ejercicio 3

```
proc enTerritorio (in v: Viaje, in r: Dist, out res: Bool) {
    Pre {esViajeValido(v) ∧ r > 0}
    Post {res = True ↔ (∃g : GPS)(gpsEnRango(g) ∧_L (∀j : ℤ)(0 ≤ j < |v| ⟶_L dist(g, (v[j])_1) ≤ (r * 1000)))}
}
```

4.4. Ejercicio 4

```
proc tiempoTotal (in v: Viaje, out t: Tiempo) {
    Pre {esViajeValido(v)}
    Post {tiempoEsCorrectoParaViaje(v, t) ∧ tiempoEsTotal(v, t)}
    pred tiempoEsCorrectoParaViaje (v: Viaje, t: Tiempo) {
        (∃i, j : ℤ)((0 ≤ i < |v| ∧ 0 ≤ j < |v|) ∧_L t = (v[i])_0 − (v[j])_0)
    }
```

pred tiempoEsTotal (v: *Viaje*, t: *Tiempo*) {

$(\forall i, j : \mathbb{Z})((0 \leq i < |v| \land 0 \leq j < |v|) \longrightarrow_L t \geq (v[i])_0 - (v[j])_0)$

}

}

4.5. Ejercicio 5

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

Pre $\{esViajeValido(v)\}$

Post $\{(\exists f : Viaje)(mismoViajeOrdPorTiempo(v, f) \land d = viajeDistanciaTotal(f))\}$

aux viajeDistanciaTotal (f: *Viaje*) : $\mathbb{R} = \sum_{i=0}^{|f|-2} dist((f[i])_1, (f[i+1])_1)$;

}

4.6. Ejercicio 6

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

Pre $\{esViajeValido(v)\}$

Post $\{res = true \leftrightarrow (\exists f : Viaje)(mismoViajeOrdPorTiempo(v, f) \land viajeSupera80kmh(f))\}$

pred viajeSupera80kmh (f: *Viaje*) {

$(\exists i : \mathbb{Z})(0 \leq i < |f| - 1 \land_L velocidad(f[i], f[i+1]) > 80)$

}

}

4.7. Ejercicio 7

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

Pre $\{sonViajesValidos(v) \land (t_0 > 0 \land t_f > 0)\}$

Post $\{res = cantViajesEnFranja(v, t_0, t_f)\}$

aux cantViajesEnFranja ($v : seq\langle Viaje\rangle, t_0, t_f : Tiempo$) : $\mathbb{Z} = \sum_{i=0}^{|v|-1}$ if $viajeEnFranja(v[i], t_0, t_f)$ then 1 else 0 fi ;

pred viajeEnFranja ($v : Viaje, t_0, t_f : Tiempo$) {

$algunPuntoIncluido(v[i], t_0, t_f) \lor hayBordes(v[i], t_0, t_f)$

}

pred algunPuntoIncluido ($v : Viaje, t_0, t_f : Tiempo$) {

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

}

como el viaje registra puntos nuevos cada 20 segs, busco bordes tal que al correrlos 20 segs estan en la franja

pred hayBordes ($v : Viaje, t_0, t_f : Tiempo$) {

$(\exists i : \mathbb{Z})(0 \leq i < |v| \land_L t_0 \leq (v[i])_0 + 20) \land (\exists i : \mathbb{Z})(0 \leq i < |v| \land_L t_f \geq (v[i])_0 - 20)$

}

pred hayTiemposFueraDeFranja ($v : Viaje, t_0, t_f : Tiempo$) {

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

}

}

4.8. Ejercicio 8

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

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

Post $\{(\forall i : \mathbb{Z})(0 \leq i < |res| \longrightarrow_L (puntoEnRecorrido(r, res[i]) \land_L \neg fueCubierto(\text{v, u, res[i]})))\}$

pred puntoEnRecorrido (r: *Recorrido*, p: *GPS*) {

$(\exists i : \mathbb{Z})(0 \leq i < |r| \land_L r[i] = p)$

}

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

$(\exists i : \mathbb{Z})(0 \leq i < |v| \land_L dist((v[i])_1, p) \leq u * 1000)$

}

}

4.9. Ejercicio 9

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

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

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

pred esquinasCorrectasRespecto (esq1: *GPS*, esq2: *GPS*) {

$(gpsEnRango(esq1) \land gpsEnRango(esq2)) \land_L ((esq1)_0 > (esq2)_0 \land (esq1)_1 < (esq2)_1)$

}

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

$|g| = n * m \land celdasIguales(g, n, m, esq1, esq2) \land nombresCorrectos(g, n, m, esq1, esq2)$

}

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

$(\forall i : \mathbb{Z})(0 \leq i < |g| \longrightarrow_L (g[i]_0)_0 - (g[i]_1)_0 = latxCelda(esq1, esq2, n) \land (g[i]_1)_1 - (g[i]_0)_1 = longxCelda(esq1, esq2, m))$

}

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

$(\forall i : \mathbb{Z})(0 \leq i < |g| \longrightarrow_L nombreCorrespondeConPuntos(g[i]_2, g[i]_1, esq1, esq2, n, m))$

}

pred nombreCorrespondeConPuntos (nombre: *Nombre*, p2, esq1, esq2: *GPS*, n, m: $\mathbb{Z}$) {

$nombre_0 = \frac{esq1_0 - p2_0}{latxCelda(esq1, esq2, n)} \land nombre_1 = \frac{p2_1 - esq1_1}{longxCelda(esq1, esq2, m)}$

}

}

4.10. Ejercicio 10

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

Pre $\{grillaValida(g) \land esRecorridoValido(r) \land_L recorridoEnGrilla(r, g)\}$

Post $\{regionesCorrectas(r, g, res)\}$

pred recorridoEnGrilla (r: *Recorrido*, g: *Grilla*) {

$(\forall i : \mathbb{Z})(0 \leq i < |r| \longrightarrow_L puntoEnGrilla(r[i], g))$

}

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

$(\forall i : \mathbb{Z})(0 \leq i < |res| \longrightarrow_L nombreCorrespondeACelda(r[i], res[i], g))$

}

}

4.11. Ejercicio 11

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

Pre $\{grillaValida(g) \wedge esViajeValido(v) \wedge_L viajeEnGrilla(v, g)\}$

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

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

$(\forall i : \mathbb{Z})(0 \leq i < |v| \longrightarrow_L puntoEnGrilla(v[i]_1, g))$

}

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

$(\exists f : Viaje)(mismoViajeOrdPorTiempo(v, f) \wedge_L res = cantidadSaltos(g, f))$

}

aux cantidadSaltos (g: *Grilla*, v: *Viaje*) : $\mathbb{Z} = \sum_{i=0}^{|v|-2}$ if $saltoEnfila(v, g, i) \vee saltoEnColumna(v, g, i)$ then 1 else 0 fi) ;

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

$||filaDePEnGrilla(v[i]_1, g) - filaDePEnGrilla(v[i+1]_1, g)|| \geq 2$

}

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

$||colDePEnGrilla(v[i]_1, g) - colDePEnGrilla(v[i+1]_1, g)|| \geq 2$

}

}

4.12. Ejercicio 12

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

Pre $\{viajeACorregirValido(v, errores) \wedge listaErroresCorrecta(v, errores) \wedge v = V_0\}$

Post $\{esViajeCorregidoDe(v, V_0, errores)\}$

pred viajeACorregirValido (v: *Viaje*, errores:$seq\langle Tiempo\rangle$) {

$esViajeValido(v) \wedge |v| \geq 5 \wedge primerYUltValidos(v, errores) \wedge tieneCostadosValidos(v, errores)$

}

pred listaErroresCorrecta (v: *Viaje*, errores: $seq\langle Tiempo\rangle$) {

$\frac{|errores|}{|v|} \leq 0,1$

}

pred primerYUltValidos (v: *Viaje*, e: $seq\langle Tiempo\rangle$) {

$(\exists f : Viaje)(mismoViajeOrdPorTiempo(v, f) \wedge_L (\neg(hayError(f[0]_0, e) \wedge hayError(f[|f| - 1], e))))$

}

pred esViajeCorregidoDe (vf, vi: *Viaje*, errores: $seq\langle Tiempo\rangle$) {

$(\exists f : Viaje)((viajeMismosElementos(vi, f) \wedge esViajeOrdenadoCorregido(f, errores)) \wedge_L vf = f)$

}

pred hayError (t: *Tiempo*, errores: $seq\langle Tiempo\rangle$) {

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

}

`pred esViajeOrdenadoCorregido` (v: *Viaje*, errores: $seq\langle Tiempo\rangle$) {

$(\exists f : Viaje)(mismoViajeOrdPorTiempo(f, v) \wedge_L esViajeCorregido(f, errores))$

}

`pred esViajeCorregido` (v: *Viaje*, e: $seq\langle Tiempo\rangle$) {

$(\forall i : \mathbb{Z})((0 \leq i < |v| \wedge_L hayError(v[i]_0, e)) \longrightarrow_L puntoCorregido(v, i))$

}

necesitamos que los costados de un punto con error sean validos para poder hacer correctamente la interpolacion con esos puntos

`pred tieneCostadosValidos` (v: *Viaje*, e: $seq\langle Tiempo\rangle$) {

$(\forall i : \mathbb{Z})((1 \leq i < |v| - 1 \wedge_L hayError(v[i]_0, e)) \longrightarrow_L costadosValidos(v, e, i))$

}

`pred costadosValidos` (v: *Viaje*, e: $seq\langle Tiempo\rangle$, i: $\mathbb{Z}$) {

$\neg(hayError(v[i-1]_0, e) \wedge hayError(v[i+1]_0, e))$

}

`pred puntoCorregido` (v: *Viaje*, i: $\mathbb{Z}$) {

$dist(v[i-1]_1, v[i]_1) = velMedia(v[i-1], v[i+1]) * (v[i]_0 - v[i-1]_0)$

}

`aux velMedia` (pAnt, pSig: $Tiempo \times GPS$) : $\mathbb{R} = \frac{dist(pAnt_1, pSig_1)}{pSig_0 - pAnt_0}$;

}

4.13. Ejercicio 13

`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` $\{|xs| \geq 2 \wedge bins > 0 \wedge sonViajesValidos(xs)\}$

`Post` $\{(limitesCorrectos(xs, bins, limites) \wedge_L cuentasCorrectas(xs, limites, cuentas)\}$

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

$(\exists vMax, vMin : \mathbb{R})((esMaxVelocidadDeXs(xs, vMax) \wedge esMinVelocidadDeXs(xs, vMin))$
$\wedge (\forall i : \mathbb{Z})(0 \leq i < |limites| \longrightarrow_L (\exists k : \mathbb{Z})(0 \leq k \leq bins \wedge_L limites[i] = vMin + k * \frac{vMax - vMin}{bins})))$

}

es Max y Min velocidad de xs se refiere a que c es una velocidad máxima de algún viaje en xs y el max y min corresponde con la mas grande o la mas chica de estas

`pred esMaxVelocidadDeXs` (xs: $seq\langle Viaje\rangle$, c: $\mathbb{R}$) {

$esVelocidadMaxEnXs(xs, c) \wedge \neg(\exists d : \mathbb{R})(esVelocidadMaxEnXs(xs, d) \wedge c < d)$

}

`pred esMinVelocidadDeXs` (xs: $seq\langle Viaje\rangle$, c: $\mathbb{R}$) {

$esVelocidadMaxEnXs(xs, c) \wedge \neg(\exists d : \mathbb{R})(esVelocidadMaxEnXs(xs, d) \wedge c > d)$

}

`pred esVelocidadMaxEnXs` (xs: $seq\langle Viaje\rangle$, c: $\mathbb{R}$) {

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

}

`pred esVelocidadMaximaEnViaje` (v: *Viaje*, c: $\mathbb{R}$) {

$(\exists f : Viaje)(mismoViajeOrdPorTiempo(v, f) \wedge_L$
$(\exists i : \mathbb{Z})(0 \leq i < |f| - 1 \wedge_L c = velocidad(f[i], f[i+1])) \wedge_L (\forall i : \mathbb{Z})(0 \leq i < |f| - 1 \longrightarrow_L c \geq velocidad(f[i], f[i+1])))$

}

`pred cuentasCorrectas` (xs: $seq\langle Viaje\rangle$, lims: $seq\langle\mathbb{R}\rangle$, cuentas: $seq\langle\mathbb{Z}\rangle$) {

$(\forall i : \mathbb{Z})(0 \leq i < |cuentas| - 1 \longrightarrow_L cuentas[i] = \sum_{i=0}^{|xs|-1} \text{if } maximoEntreLimites(xs[i], lims[i], lims[i+1]) \text{ then } 1 \text{ else } 0 \text{ fi})$
$\wedge cuentas[|cuentas|-1] = \sum_{i=0}^{|xs|-1} \text{if } maxEntreLimitesCerrado(xs[i], lims[|lims|-2], lims[|lims|-1]) \text{ then } 1 \text{ else } 0 \text{ fi}$

}

`pred maximoEntreLimites` (v: *Viaje*, limAnt, limSig: $\mathbb{R}$) {

$(\exists velViaje : \mathbb{R})(esVelocidadMaximaEnViaje(v, velViaje) \wedge_L limAnt \leq velViaje < limSig)$

}

`pred maxEntreLimitesCerrado` (v: *Viaje*, limAnt, limSig: $\mathbb{R}$) {

$(\exists velViaje : \mathbb{R})(esVelocidadMaximaEnViaje(v, velViaje) \wedge_L limAnt \leq velViaje \leq limSig)$

}

}