DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

TP de Especificación

Especificación - Parte I

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

Grupo 7

Integrante	LU	Correo electrónico
Esteban Muñoz, Sergio	125/21	semunoz@dc.uba.ar
Abrahan Cerimeli, Facundo	11/18	facundoroot@gmail.com
De Bonis, Matias	442/21	madebonis@dc.uba.ar
Romano, Iván	325/17	ivanromano.a@gmail.com.ar

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 = $\mathbb{R}$ (Tiempo transcurrido desde media noche 01/01/1970)

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. Constantes

aux MIN : $\mathbb{Z}$ = 1;

aux T : $\mathbb{Z}$ = 20;

3. Problemas

3.1. Ejercicio 1

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

Pre {true}

Post $\{result = true \iff esViajeValido(v)\}$

}

pred esViajeValido (v: Viaje) {

$estaEnRangoTiempo(v) \land estaEnRangoGPS(v)$

}

pred estaEnRangoTiempo (v: Viaje) {

/*Estan en progresion aritmetica de razon 20*\
$(\forall i : \mathbb{Z})(0 \leq i < |v| \longrightarrow_L 0 \leq (v[i])_0 \land ((\exists j : \mathbb{Z})(0 \leq j < |v| \land_L (v[i])_0 + 20 = (v[j])_0) \lor esMaximo(v([i])_0))$
$\land$
$NoSonIgualesLosTiempos(v)$

}

pred NoSonIgualesLosTiempos (v :Viaje) {

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

}

pred esMaximo (tiempo:Tiempo) {

$(\forall i : Z)(0 \leq i < |v| \longrightarrow_L 0 \leq (v[i])_0 \leq tiempo)$

}

pred estaEnRangoGPS (v: Viaje) {

$(\forall i : \mathbb{Z})(0 <= i < |v| \longrightarrow_L (-90 \leq ((v[i])_1)_0 \leq 90) \land (-180 \leq ((v[i])_1)_1 \leq 180)$

}

3.2. Ejercicio 2:

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

Pre $\{|\ v\ | \neq 0\}$

Post $\{result = true \iff estaEnRangoGPSRecorrido(v) \wedge losPuntosEstanA20DeDistancia\}$

}

pred estaEnRangoGPSRecorrido (r: Recorrido) {

$(\forall i : \mathbb{Z})(0 \leq i < |r| \longrightarrow_L (-90 \leq (r[i])_0 \leq 90) \wedge (-180 \leq (r[i])_1 \leq 180))$

}

pred losPuntosEstanA20DeDistancia (r: Recorrido) {

$(\forall i : \mathbb{Z})(0 \leq i < |r| - 1 \longrightarrow_L dist(r[i], r[i+1]) = 20)$

}

3.3. Ejercicio 3

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

Pre $\{esViajeValido(v) \wedge r > 0\}$

Post $\{result = true \iff ExistePuntoQueSuRadioContieneATodos(v, r)\}$

}

pred ExistePuntoQueSuRadioContieneATodos (v:Viaje, r :Dist) {

$(\exists punto : GPS)(estaEnRangoGPS(punto) \wedge (\forall j : Z)(0 \leq j < |v| \wedge \longrightarrow_L dist(punto, (v[j])_1) \leq r * 1000))$

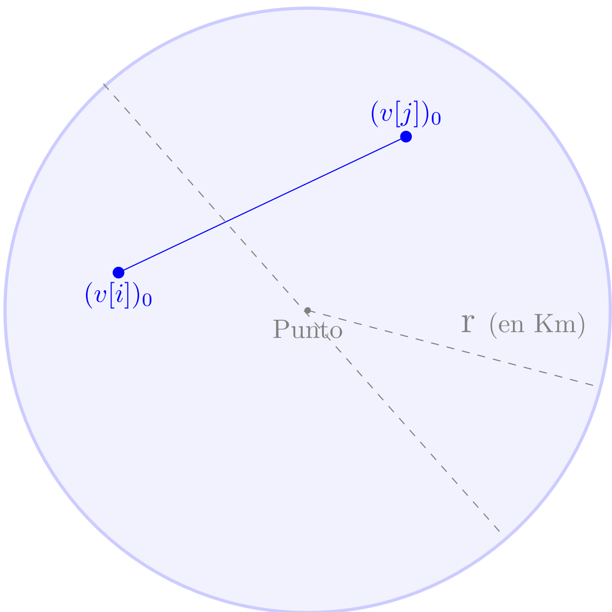
}

pred estaEnRangoGPS (gps: GPS) {

$(-90 \leq (gps)_0 \leq 90) \wedge (-180 \leq (gps)_1 \leq 180)$

}

3.3.1. Intuición geométrica



3.4. Ejercicio 4:

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

Pre $\{esViajeValido(v)\}$

Post $\{esLaRestaDelMinimoConElMaximo(t, v)\}$

}

`pred` `esLaRestaDelMinimoConElMaximo` (t: Tiempo, v:Viaje) {

$(\forall x, y : Tiempo)(esMaximo(x,v) \land esMinino(y,v) \rightarrow t = x - y)$

}

`pred` `esMaximo` (t: Tiempo, v :Viaje) {

$(\forall i : \mathbb{Z})(0 \leq i < |v| \longrightarrow_L (\exists max : Tiempo \times GPS)(\ max \in v \land (max)_0 \geq (v[i])_0 \land_L t = (max)_0)$

}

`pred` `esMinimo` (t: Tiempo, v :Viaje) {

$(\forall i : \mathbb{Z})(0 \leq i < |v| \longrightarrow_L (\exists min : Tiempo \times GPS)(min \in v \land min \leq (v[i])_0 \land_L t = (min)_0)$

}

3.5. Ejercicio 5:

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

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

`Post` $\{(\exists vt : Viaje)(\ esPermutacion(vt,v) \land esViajeOrenadoPorTiempo(vt) \land_L d = \sum_{i=1}^{|vt|-1} dist((v[i])_1, (v[i-1])_1\ )\}$

}

`pred` `esViajeOrdenadoPorTiempo` (v: Viaje) {

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

}

`pred` `esPermutacion` (vt :Viaje, v :Viaje) {

$|vt| = |v| \land (\forall i : \mathbb{Z})(0 \leq i < |vt| \longrightarrow_L \#cantidadDeAparaciones(vt[i], vt) = \#cantidadDeAparaciones(v[i]), v)$

}

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

3.6. Ejercicio 6:

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

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

`Post` $\{res = true \iff excedeLosOchentaKilometrosPorHora(v)\}$

}

`pred` `excedeLosOchentaKilometrosPorHora` (v: Viaje) {

$(\exists i : \mathbb{Z})(0 \leq i < |v| \land_L (\exists j : \mathbb{Z})(0 \leq j < |v| \land_L ((v[i])_0 - (v[j])_0) = 20 \land \frac{dist((v[i])_1, (v[j])_1}{20} \times 3{,}6 \geq 80)$

}

3.7. Ejercicio 7:

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

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

`Post` $\{(\exists sv : seq\langle Viaje\rangle)(((\forall i : \mathbb{Z})(0 \leq i < |sv| \longrightarrow_L sv[i] \in v \land$
$viajeEnRangoDeTiempos(sv[i], t_0, t_f) \land$
$sonTodosLosQueCumplen(v, sv, t_0, t_f)))) \land res = |sv|)\}$

}

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

$(\exists j : \mathbb{Z})(0 \leq j < |v| \wedge_L t_0 \leq (v[j])_0 \leq t_f)$

}

`pred sonTodosLosQueCumplen` (v : $seq\langle Viaje\rangle$, sv:$seq\langle Viaje\rangle$, t_0: Tiempo, t_f: Tiempo) {

$\neg(\exists viaje : Viaje)(viaje \in v \wedge viajeEnRangoDeTiempos(v, t_0, t_f) \wedge \neg(viaje \in sv))$

}

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

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

}

3.8. Ejercicio 8:

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

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

`Post` $\{(\forall i : \mathbb{Z})(0 \leq i < |res| \longrightarrow_L res[i] \in r \wedge \neg(PuntoCubierto(res[i], u, v))) \wedge$
$\neg(\exists reco : GPS)(reco \in r \wedge_L \neg(PuntoCubierto(reco, u, v)) \wedge \neg(reco \in res)\ ) \wedge NoHayRepetidos(res)\}$

}

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

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

}

`pred NoHayRepetidos` (r: $seq\langle GPS\rangle$) {

$(\forall i : \mathbb{Z})(0 \leq i < |r| \longrightarrow_L \#cantidadDeApariciones(r[i], r) = 1)$

}

3.9. Ejercicio 9:

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

`Pre` $\{(esq1)_0 > (esq2)_0 \wedge (esq1)_1 < (esq2)_1\}$

`Post` $\{|g| = n{*}m \wedge existePrimeraCasilla(esq1, g) \wedge (\forall i : \mathbb{Z})(1 \leq i \leq n{*}m \longrightarrow_L longitudCorrecta(g[i], esq1, esq2, m) \wedge$
$latitudCorrecta(g[i], esq1, esq2, n) \wedge$
$existeUnaCasillaALaDerechaOEsLaUltima(g[i], g, m, longitud(esq1, esq2))) \wedge$
$existeUnaCasillaAbajoOEsLaUltima(g[i], g, m, latitud(esq1, esq2)))\}$

}

`pred longitudCorrecta` (casilla: $GPS \times GPS \times nombre$, esq1: GPS, esq2: GPS, m: $\mathbb{Z}$) {

$longitud((casilla)_0, (casilla)_1)) = longitud(esq1, esq2)/m)$

}

`pred latitudCorrecta` (casilla: $GPS \times GPS \times nombre$, esq1: GPS, esq2: GPS, n: $\mathbb{Z}$) {

$latitud((casilla)_0, (casilla)_1)) = latitud(esq1, esq2)/n)$

}

`pred existePrimeraCasilla` (esq1: GPS, g: Grilla) {

$(\exists c : GPS \times GPS \times nombre)(c \in g \wedge_L (c)_2 = (1, 1) \wedge c_0 = esq1)$

}

`pred existeUnaCasillaALaDerechaOEsLaUltima` (casilla: $GPS \times GPS \times nombre$, g: Grilla, m: $\mathbb{Z}$, longitudTotal: $\mathbb{Z}$) {

$(\exists c : GPS \times GPS \times nombre)(c \in g \land_L ((c)_2 = (((casilla)_2)_0, (casilla_2)_1 + 1) \lor$
$((c)_2)_1 = m) \land dist((c)_0, (casilla)_0) = longitudTotal)$

}

`pred existeUnaCasillaAbajoOEsLaUltima` (casilla: $GPS \times GPS \times nombre$, g: Grilla, m: $\mathbb{Z}$, latitudTotal: $\mathbb{Z}$) {

$(\exists c : GPS \times GPS \times nombre)(c \in g \land_L ((c)_2 = (((casilla)_2)_0 + 1, (casilla_2)_1) \lor$
$((c)_2)_1 = n) \land dist((c)_0, (casilla)_0) = latitudTotal)$

}

`aux longitud` (esq1: GPS, esq2: GPS) : $\mathbb{Z}$ $= dist(esq1, ((esq2)_0 \times (esq1)_1))$;

`aux latitud` (esq1: GPS, esq2: GPS) : $\mathbb{Z}$ $= dist(esq1, ((esq1)_0 \times (esq2)_1))$;

3.10. Ejercicio 10:

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

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

`Post` $\{|r| = |res| \land EstaOrdenadaPorRegionesVisitadasPorElColectivo(res, r, g)\}$

}

`pred EstaOrdenadaPorRegionesVisitadasPorElColectivo` (res: $seq\langle Nombre\rangle$, r: recorrido, g: Grilla) {

$(\forall i : \mathbb{Z})(0 \leq i < |res| \longrightarrow_L (\exists j : \mathbb{Z})(0 \leq j < |g| \land_L res[i] = (g[j])_2) \land PuntoDentroDeRegion(r[i], res[i]))$

}

`pred PuntoDentroDeRegion` (p: GPS, region: $GPS \times GPS \times nombre$) {

$(((region)_0)_0 < (p)_0 < ((region)_1)_0 \land$
$(((region)_0)_1 < (p)_1 < ((region)_1)_1$

}

`pred GrillaValida` (g: Grilla) {

$(\exists esq1, esq2 : GPS)((esq1)_0 > (esq2)_0 \land (esq1)_1 < (esq2)_1 \land_L (\exists n, m : \mathbb{Z})(|g| = n * m \land_L existePrimeraCasilla(esq1, g) \land$
$(\forall i : \mathbb{Z})(1 \leq i \leq n * m \longrightarrow_L longitudCorrecta(g[i], esq1, esq2, m) \land$
$latitudCorrecta(g[i], esq1, esq2, n) \land$
$existeUnaCasillaALaDerechaOEsLaUltima(g[i], g, m, longitud(esq1, esq2))) \land$
$existeUnaCasillaAbajoOEsLaUltima(g[i], g, m, latitud(esq1, esq2))))))$

}

3.11. Ejercicio 11:

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

`Pre` $\{esViajeValido(v) \land GrillaValida(g)\}$

`Post` $\{(\exists vt : Viaje)(esPermutacion(vt, v) \land$
$esViajeOrdenadoPorTiempo(vt) \land seanTodosSaltos(v, vt) \land res = |vt|)\}$

}

`pred seanTodosSaltos` (vi: Viaje, vt: Viaje, g: Grilla) {

$(\forall i : \mathbb{Z})(0 \leq i < |vi| - 1 \longrightarrow_L vi[i] \in vt \land$
$estanAAlmenosDosCasillas(vi[i + 1], vi[i]) \land$
$\neg(\exists j : \mathbb{Z})(0 \leq i < |vt| \land_L estanAAlmenosDosCasillas \land vt[i]\neg \in vi)$

}

`pred puntosDentroDeCasillaYSonSaltos` (p1: GPS, p2: GPS, g: Grilla) {

$(\exists casilla1, casilla2 : GPS \times GPS \times Nombre)(casilla1 \in g \land casilla2 \in g \land esteDentro(p1, casilla1) \land esteDentro(p2, casilla2) \land_L$
$estanAAlMenosDosCasillas(p1, p2, g)$

}

`aux abs` (x: $\mathbb{R}$) : $\mathbb{R}$ = if $x \geq 0$ then x else $-x$ fi ;

`pred estanAAlMenosDosCasillas` (c1: $GPS \times GPS \times Nombre$, c2: $GPS \times GPS \times Nombre$) {

$abs(((c1)_2)_0 - ((c2)_2)_0) \geq 2 \vee abs(((c1)_2)_1 - ((c2)_2)_1) \geq 2$

}

`pred esteDentro` (p: GPS, casilla: $GPS \times GPS \times Nombre$) {

$(((casilla)_0)_0 \geq (p)_0 \geq ((casilla)_1)_0) \wedge (((casilla)_0)_1 \geq (p)_1 \geq ((casilla)_1)_1)$

}

3.12. Ejercicio 12:

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

`Pre` $\{v = v_0 \wedge esViajeValido(v) \wedge |v| > 5 \wedge (0 < |errores| \leq |v| * 0{,}1)\}$

`Post` $\{(\forall i : \mathbb{Z})((0 \leq i < |v_0| \wedge_L debeSerCorregido(v_0[i], errores)) \longrightarrow_L v_0[i] \notin v \wedge puntoCorregido(v_0[i], v_0, v, errores)) \wedge$

$((\forall i : \mathbb{Z})(0 \leq i < |v_0| \wedge_L \neg debeSerCorregido(v_0[i], errores) \longrightarrow_L v_0[i] = v[i]))\}$

}

`pred debeSerCorregido` (punto: $Tiempo \times$GPS, errores: $seq\langle Tiempo\rangle$) {

$(punto)_0 \in errores$

}

`pred puntoCorregido` (p: $Tiempo \times GPS$, v_0: Viaje, v: Viaje, errores: $seq\langle Tiempo\rangle$) {

$(\exists p1, p2 : Tiempo \times GPS)(p1 \in v_0 \wedge p2 \in v_0 \wedge_L esElAnteriorQueNoDebeCorregirse(p, p1, v_0, errores) \wedge$
$esElSiguienteQueNoDebeCorregirse(p2, v_0, errores) \wedge_L esCorregido(p, p1, p2, v))$

}

`pred esElAnteriorQueNoDebeCorregirse` (p: $Tiempo \times GPS$, p1: $Tiempo \times GPS$ v_0: Viaje, errores : $seq\langle Tiempo\rangle$) {

$\neg debeSerCorregido(p1) \wedge \neg(\exists b : Tiempo \times GPS)(b \in v_0 \wedge_L \neg debeSerCorregido(b, errores) \wedge (p1)_0 < (b)_0 < (p)_0))$

}

`pred esElSiguienteQueNoDebeCorregirse` (p: $Tiempo \times GPS$, p2: $Tiempo \times GPS$ v_0: Viaje, errores : $seq\langle Tiempo\rangle$) {

$\neg debeSerCorregido(p2) \wedge \neg(\exists b : Tiempo \times GPS)(b \in v_0 \wedge_L \neg debeSerCorregido(b, errores) \wedge (p)_0 < (b)_0 < (p2)_0))$

}

`pred esCorregido` (p: $Tiempo \times GPS$, p1: $Tiempo \times GPS$, p2: $Tiempo \times GPS$, v: Viaje) {

$(\exists pCorregido : Tiempo \times GPS)(pCorregido \in v \wedge_L$
$(pCorregido)_0 = (p)_0 \wedge estaADistanciasValidasDeLosPuntos(pCorregido, p1, p2))$

}

`pred estaADistanciasValidasDeLosPuntos` (p: $Tiempo \times GPS$, p1: $Tiempo \times GPS$, p2: $Tiempo \times GPS$) {

$(dist((p)_1, (p1)_1) = velocidadMedia(p1, p2) * 20) \wedge (dist((p)_1, (p2)_1) = dist(p1, p2) - dist((p)_1, (p1)_1))$

}

`aux velocidadMedia` (p1: $Tiempo \times GPS$, p2: $Tiempo \times GPS$) : $\mathbb{R}$ = $\dfrac{dist(p1, p2)}{abs((p1)_0 - (p2)_0)}$;

3.13. 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` $\{bins > 0 \wedge listaDeViajesValidos(xs)\}$

`Post` $\{|limites| = |cuentas| + 1 \wedge |cuentas| = Bins \wedge_L$
$(\exists velocidadesMaximas : seq\langle\Re\rangle)(|velocidadesMaximas| = |xs| \wedge_L$
$todoslosElementosSonVelocidadesMaximas(velocidadesMaximas, xs) \wedge$

$limitesEsValido(limites, velocidadesMaximas, bins) \wedge$
$cuentasEsValido(limites, velocidadesMaximas)\}$

}

`pred todoslosElementosSonVelocidadesMaximas` (velocidadesMaximas: $seq\langle\Re\rangle$, xs: $seq\langle Viajes\rangle$) {

$(\forall i : \mathbb{Z})(0 \leq i < |velocidadesMaximas| \longrightarrow_L esVelocidadMaximaDelViaje(velocidadesMaximas[i], xs[i])$

}

`pred esVelocidadMaximaDelViaje(velocidadesMaximas` (velocidad: $\Re$, viaje: Viaje) {

$(\forall j : \mathbb{Z})(0 \leq j < |viaje| \longrightarrow_L velocidad >= velocidadMedia(v[j]_1, v[j+1]_1))$

}

`pred esVelocidadMaximaDelViaje` (velocidad: $\mathbb{R}$, v: Viajes) {

$(\forall x, y : Tiempo \times GPS)(x \in v \wedge y \in v \longrightarrow_L x_0 < y_0 \wedge_L$
$velocidad = \dfrac{dist(y_1, x_1)}{y_0 - x_0} \wedge esMaxima(velocidad, x, y, v))$

}

`aux velocidadMedia` (p1: GPS, p2: GPS) : $\Re = \dfrac{dist(p1, p2)}{20}$;

`pred limitesEsValido` (limites: $seq\langle\mathbb{R}, velocidadesMaximas : seq\langle\Re\rangle, bins : \mathbb{Z}\rangle$) {

$(\exists velMax, velMin : \Re)(velMax \neq velMin \wedge_L esVelMaxima(velMax, veolcidadesMaximas) \wedge$
$esVelMinima(velMin, velocidadesMaximas) \wedge$
$(\forall i : \mathbb{Z})(1 \leq i < |limites| - 1 \longrightarrow_L limites[i] = limites[i-1] + velMax/bins \wedge$
$limites[0] = velMin \wedge limites[|limites| - 1] = velMax))$

}

`pred esVelMaxima` (velMax: $\Re$, velocidadesMaximas: $seq\langle\Re\rangle$) {

$\neg(\exists i : \mathbb{Z})(0 \leq i < |velocidadesMaximas| \wedge_L velocidadesMaximas[i] > velMax \wedge$
$(\exists j : \mathbb{Z})(0 \leq i < |velocidadesMaximas| \wedge_L velocidadesMaximas[j] = velMax))$

}

`pred esVelMinima` (velMin: $\Re$, velocidadesMaximas: $seq\langle\Re\rangle$) {

$\neg(\exists i : \mathbb{Z})(0 \leq i < |velocidadesMaximas| \wedge_L velocidadesMaximas[i] < velMin \wedge$
$(\exists j : \mathbb{Z})(0 \leq i < |velocidadesMaximas| \wedge_L velocidadesMaximas[j] = velMin))$

}

`pred cuentasEsValido` (limites: $seq\langle\mathbb{R}\rangle$, velocidadesMaximas: $seq\langle\mathbb{R}\rangle$, cuentas: $seq\langle\mathbb{Z}\rangle$) {

$(\forall i : \mathbb{Z})(0 \leq i < |limites| - 1 \longrightarrow_L (\exists velocidadesFiltradas : seq\langle\mathbb{R}\rangle)((\forall elements : \mathbb{R})(elements \in velocidadesMaximas \wedge$
$elements \in velocidadesFiltradas \wedge_L$
$esteEntreLimitesAC(elements, limites[i], limites[i+1], limites)) \wedge cuentas[i] = |velocidadesFiltradas|)$

}

`pred esteEntreLimites` (valor:$\mathbb{R}$, cotaInferior:$\mathbb{R}$, cotaSuperior: $\mathbb{R}$, limites: $seq\langle\mathbb{R}\rangle$) {

$cotaInferior \leq valor < cotaSuperior \vee (cotaSuperior = limites[|limites| - 1] \wedge cotaInferior \leq valor \leq cotaSuperior)$

}