



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 03

Integrante	LU	Correo electrónico
Fresone, Juan Francisco	749/21	fresone.juan@gmail.com
Iannantuono, Ignacio	1897/21	ignacio.iannantuono@gmail.com
Yudcovsky, Dafne Sol	1888/21	dafneyudcovsky@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. Problemas

### 2.1. Ejercicio 1

```
proc viajeValido (in v: Viaje, out res: Bool) {
  Pre {True}
  Post {res = true  $\leftrightarrow$  esUnViajeValido(v)}
  pred esTiempoValido (v: Viaje) {
     $(\forall i : \mathbb{Z})(0 \leq i < |v| \rightarrow_L ((v[i]_0 \geq 0 \wedge_L (\forall j : \mathbb{Z})(0 \leq j < |v|) \rightarrow_L (v[j]_0 \neq v[i]_0))))$ 
  }
  pred esGPSValido (v: Viaje) {
     $(\forall i : \mathbb{Z})(0 \leq i < |v| \rightarrow_L ((-90 \leq v[i]_{1_0} \leq 90) \wedge (-180 \leq v[i]_{1_1} \leq 180)))$ 
  }
  pred esUnViajeValido (v: Viaje) {
    esTiempoValido(v)  $\wedge$  esGPSValido(v)
  }
}
```

### 2.2. Ejercicio 2

```
proc recorridoValido (in v: Recorrido, out res: Bool) {
  Pre {True}
  Post {res = true  $\leftrightarrow$  (recorridoCorrecto(v))}
  pred recorridoCorrecto (v: Recorrido) {
     $(\forall i : \mathbb{Z})(0 \leq i < |v| \rightarrow_L ((-90 \leq v[i]_0 \leq 90) \wedge (-180 \leq v[i]_1 \leq 180)))$ 
  }
}
```

### 2.3. Ejercicio 3


```
proc enTerritorio (in v: Viaje, in r: Dist, out res: Bool) {
  Pre {esUnViajeValido(v)  $\wedge$  (r > 0)}
  Post {res = true  $\leftrightarrow$   $((\forall i, j : \mathbb{Z})(0 \leq i, j < |v|) \rightarrow_L (dist((v[i]_1), (v[j]_1)) \leq (2 \times r \times 1000))))$ 
}
```

## 2.4. Ejercicio 4

```

proc tiempoTotal (in v: Viaje, out t: Tiempo) {
  Pre {esUnViajeValido(v)}
  Post {esTiempoTotal(v, t)}
  pred esTiempoTotal (v: Viaje, t: Tiempo) {
     $(\exists n, m : \mathbb{Z})(0 \leq n, m < |v|) \wedge_L (\forall i : \mathbb{Z})(0 \leq i < |v| \wedge (v[n]_0 \geq v[i]_0) \wedge (v[m]_0 \leq v[i]_0)) \longrightarrow_L (t = v[n]_0 - v[m]_0)$ 
  }
}

```


Este paréntesis no va.

## 2.5. Ejercicio 5

```

proc distanciaTotal (in v: Viaje, out d: Dist) {
  Pre {esUnViajeValido(v)}
  Post { $(\forall i : \mathbb{Z})(0 \leq i < |v|) \longrightarrow_L (\exists v' : Viaje)((v'[i] \in v \wedge |v'| = |v| \wedge estaOrdenado(v')) \wedge_L d = sumaDeTramos(v'))$ }
  pred estaOrdenado (v: Viaje) {
     $(\forall i : \mathbb{Z})(0 \leq i < |v|) \longrightarrow_L ((v[i]_0 < (v[i + 1]_0))$ 
  }
  aux sumaDeTramos (v: Viaje) : Dist =  $\sum_{i=0}^{|v|-2} (dist((v[i]_1), (v[i + 1]_1)) \times 1000)$ ;
}

```

El orden de los cuantificadores está al revés.  
 Están diciendo  
 "para todo índice de v, hay una secuencia ordenada cuyo elemento en esa  
 posición está en v y ...",  
 cuando lo correcto sería,  
 "existe una secuencia ordenada que es permutación de v y ...".

## 2.6. Ejercicio 6

```

proc excesoDeVelocidad (in v: Viaje, out res: Bool) {
  Pre {esUnViajeValido(v)}
  Post { $(\forall i : \mathbb{Z})(0 \leq i < |v|) \longrightarrow_L (\exists v' : Viaje)((v'[i] \in v \wedge |v'| = |v| \wedge estaOrdenado(v')) \wedge_L res = true \leftrightarrow (excesoV(v'))$ }
  }
  pred excesoV (v: Viaje) {
     $(\exists i : \mathbb{Z})(0 \leq i < |v| - 1 \wedge_L (dist((v[i]_1), (v[i + 1]_1)) > 4000/9))$ 
  }
}

```

## 2.7. Ejercicio 7

```

proc flota (in v: seq< Viaje>, in t0 : Tiempo, in tf : Tiempo, out res :  $\mathbb{Z}$ ) {
  Pre {todosViajesValidos(v)  $\wedge$  t0  $\geq$  0  $\wedge$  tf  $\geq$  t0}
  Post { $(\forall i : \mathbb{Z})(0 \leq i < |v|) \longrightarrow_L (\exists v' : seq< Viaje>)((v'[i] \in v \wedge |v'| = |v| \wedge (viajeOrdenado(v[i]')) \wedge_L$   

  res = contadorViaje(v', t0, tf))}
  pred todosViajesValidos (v: seq< Viaje>) {
     $(\forall i : \mathbb{Z})(0 \leq i < |v| \longrightarrow_L esUnViajeValido(v[i]))$ 
  }
  pred enRutaDuranteT0yTF (v: Viaje, t0 : Tiempo, tf : Tiempo) {
     $((v[0]_0 \leq t_0 \leq v[|v| - 1]_0) \vee (v[0]_0 \leq t_f \leq v[|v| - 1]_0) \vee (t_0 \leq v[0]_0 \wedge v[|v| - 1] \leq t_f))$ 
  }
}

```

```

aux contadorViaje (v:seq⟨Viaje⟩, t0 : Tiempo, tf : Tiempo) :  $\mathbb{Z}$  =  $\sum_{i=0}^{|v|-1} (\text{if } \text{enRutaDuranteT0yTF}(v[i], t_0, t_f) \text{ then } 1 \text{ else } 0 \text{ fi});$ 
}

```

## 2.8. Ejercicio 8

```

proc recorridoNoCubierto (in v:Viaje, in r:Recorrido, in u:Dist, out res:seq⟨GPS⟩) {
  Pre {u > 0 ∧ esUnViajeValido(v) ∧ recorridoCorrecto(r)}
  Post {(∀p : GPS)(p ∈ res ⟶ ¬estaCubierto(v, p, u) ∧ p ∈ r) ∧
        (∀p : GPS)(p ∈ r ∧ ¬estaCubierto(v, p, u) ⟶ p ∈ res)}
  pred estaCubierto (v:Viaje, p:GPS, u:Dist) {
    (∃j :  $\mathbb{Z}$ )(0 ≤ j < |v|) ∧L (u × 1000 ≥ (dist((p), (v[j]1))))
  }
}

```

## 2.9. Ejercicio 9

```

proc construirGrilla (in esq1:GPS, in esq2:GPS, in n: $\mathbb{Z}$ , In m: $\mathbb{Z}$ , out g:Grilla) {
  Pre {grillaValida(esq1, esq2, n, m)}
  Post {mapeoGrilla(g, esq1, esq2, n, m)}
  pred posicionValida (esq1:GPS, esq2:GPS) {
    ((-90 ≤ (esq1)0 ≤ 90 ∧ -180 ≤ (esq1)1 ≤ 180 ∧ -90 ≤ (esq2)0 ≤ 90 ∧ -180 ≤ (esq2)1 ≤ 180) ∧L (((esq1)0 >
    (esq2)0) ∧ ((esq1)1 < (esq2)1)))
  }
  pred grillaValida (esq1:GPS, esq2:GPS, n: $\mathbb{Z}$ , m: $\mathbb{Z}$ ) {
    (n > 0 ∧ m > 0 ∧ posicionValida(esq1, esq2))
  }
  pred dimensionValidaGrilla (g: Grilla, n  $\mathbb{Z}$ , m  $\mathbb{Z}$ ) {
    n * m = |g|
  }
  pred esquinasExternas (g: Grilla, esq1:GPS, esq2:GPS, n: $\mathbb{Z}$ , m: $\mathbb{Z}$ ) {
    (∀a :  $\mathbb{Z}$ )(0 ≤ a ≤ |g|) ⟶L ((∃b :  $\mathbb{Z}$ )(0 ≤ b ≤ |g| ∧L
    (g[b]0_0 ≥ g[a]0_0 ∧ g[b]0_1 ≤ g[a]0_1) ∧L
    ((g[b]2) = (1, 1) ∧ (g[b]0 = esq1))))
    (∀i :  $\mathbb{Z}$ )(0 ≤ i ≤ |g|) ⟶L ((∃j :  $\mathbb{Z}$ )(0 ≤ j ≤ |g| ∧L
    (g[j]1_0 ≤ g[i]1_0 ∧ g[j]1_1 ≥ g[i]1_1) ∧L
    ((g[j]2) = (n, m) ∧ (g[j]1 = esq2))))
  }
  pred nombresNoRepetidos (g:Grilla, n: $\mathbb{Z}$ , m: $\mathbb{Z}$ ) {
    (∀i, j :  $\mathbb{Z}$ )(0 ≤ i, j < |g| ∧ i ≠ j) ⟶L
    ((1 ≤ g[i]2_0 ≤ n ∧ 1 ≤ g[j]2_0 ≤ n) ∧L
    (1 ≤ g[i]2_1 ≤ m ∧ 1 ≤ g[j]2_1 ≤ m) ∧L
    g[i]2 ≠ g[j]2)
  }
  pred mapeoGrilla (g:Grilla, esq1:GPS, esq2:GPS, n: $\mathbb{Z}$ , m: $\mathbb{Z}$ ) {
    dimensionValidaGrilla(g, n, m) ∧
    esquinasExternas(g, esq1, esq2) ∧

```

```

    nombresNoRepetidos(g) ∧
    celdasIntermediasValidas(g, esq1, esq2, n, m)
}

pred celdasIntermediasValidas (g:Grilla, esq1:GPS, esq2:GPS, n:ℤ, m:ℤ) {
    (∀i : ℤ)(0 ≤ i < |g|) →L
    (esqSupIzq(g[i]) = esq1 + ((nombre(g[i])0 - 1) * ancho(esq1, esq2, m), (nombre(g[i])1 - 1) * alto(esq1, esq2, n)))
}

aux ancho (esq1:GPS, esq2:GPS, m:ℤ) : ℝ =
     $\frac{esq2_0 - esq1_0}{m}$  ;

aux alto (esq1:GPS, esq2:GPS, n:ℤ) : ℝ =
     $\frac{esq1_1 - esq2_1}{n}$  ;
}

```

## 2.10. Ejercicio 10

```

proc regiones (in r: Recorrido, in g: Grilla, out res: seq(Nombre)) {
    Pre {recorridoCorrecto(r) ∧ grillaCorrecta(g)}
    Post {|res| = |r| ∧L
    (∀i : ℤ)(0 ≤ i < |res|) →L (∃celda : Nombre)(celdaDePuntoDelRecorrido(celda, r[i], g) ∧ res[i] = celda)}
    pred grillaCorrecta (g:Grilla,) {
        (∀i : ℤ)((-90 ≤ g[i]00 ≤ 90 ∧ -180 ≤ g[i]01 ≤ 180 ∧ -90 ≤ g[i]10 ≤ 90 ∧ -180 ≤ g[i]11 ≤ 180) ∧L (g[i]00 >
        g[i]10 ∧ (g[i]11 > g[i]01)) ∧L (g[i]20 > 0 ∧ g[i]21 > 0))
    }
    pred celdaDePuntoDelRecorrido (celda:Nombre, x:GPS, g:Grilla) {
        (∃i : ℤ)(0 ≤ i < |g| ∧L (x ∈ celda ∧ celda = g[i]2))
    }
}

```

## 2.11. Ejercicio 11

```

proc cantidadDeSaltos (in g: Grilla, in v: Viaje, out res: ℤ) {
    Pre {grillaCorrecta(g) ∧ esUnViajeValido(v)}
    Post {(∀i : ℤ)(0 ≤ i < |v|) →L ((∃v' : Viaje)(v'[i] ∈ v ∧ |v'| = |v| ∧ estaOrdenado(v')) ∧L res = contadorSaltos(g, v'))}
    pred haySaltos (g:Grilla, x:(Tiempo × GPS), y : (Tiempo × GPS)) {
        ((∃n, m : ℤ)(0 ≤ n, m < |g|) ∧L ((x ∈ g[n] ∧ y ∈ g[m]) ∧
        ((|g[n]20 - g[m]20| ≥ 2) ∨ (|g[n]21 - g[m]21| ≥ 2)))
    }
}

aux contadorSaltos (g:Grilla, v:Viaje) : ℤ =  $\sum_{i=0}^{|v|-2}$  ( if haySaltos(g, v[i], v[i+1]) then 1 else 0 fi) ;
}

```

Los cuantificadores están al revés.

Esta expresión no tipa.  
Falta especificar que g[n] es la celda que contiene x correctamente; lo mismo para g[m] e y.

## 2.12. Ejercicio 12

```

proc corregirViaje (inout v: Viaje, in errores: seq(Tiempo)) {
    Pre {esUnViajeValido(v) ∧ viajesMinimos(v) ∧ cantidadErrores(v, errores) ∧ v = v0}
    Post {|v| = |v0| ∧ (erroresCorregidos(v, v0, errores) ∧ restoIguales(v, v0, errores))}
}

```

```

pred viajesMinimos (v: Viaje) {
  (|v| ≥ 5)
}

pred cantidadErrores (v: Viaje, errores: seq⟨Tiempo⟩) {
  (|errores| ≤ |v| × 0,1) ∧L
  (∀i: ℤ)(0 ≤ i < |errores| →L (errores[i] ∈ v))
}

pred erroresCorregidos (v: Viaje, v0: Viaje, errores: seq⟨Tiempo⟩){

  (∃v': Viaje)(esPermutacionOrdenada(v', v0) ∧
  (∀i: ℤ)(0 ≤ i < |v0|) →L (∀p: Tiempo × GPS)(p0 ∈ errores ∧ p ∈ v0) →
  (∃p': Tiempo × GPS)((puntoCorregido(v', errores, p, p')) ∧
  (v0[i] == p → v[i] == p'))))
}

pred restoIguales (v: Viaje, v0: Viaje, errores: seq⟨Tiempo⟩){

  (∀i: ℤ)(0 ≤ i < |v|) →L (∀p: Tiempo × GPS)(p0 ∉ errores → v[i] == p)
}

pred esPermutacionOrdenada (viajeOrdenado, v: Viaje) {

  (∀x: Tiempo × GPS)(apariciones(viajeOrdenado, x) = apariciones(v, x)) ∧
  estaOrdenado(viajeOrdenado)
}

aux apariciones (v: Viaje, x: Tiempo × GPS) : ℤ = ∑i=0|v|-1 ( if v[i] = x then 1 else 0 fi) ;
pred puntoCorregido (v0: Viaje, errores: seq⟨Tiempo⟩, v1: Viaje, p, p': Tiempo × GPS){

  (∀i, j: ℤ)(0 < i < |v0| ∧ 0 ≤ j < |errores|) →L
  ((p0 = errores[j] ∧ v0[i] = errores[j]) →
  ((p'10 == (v0[i-1]10 × ((1 - porcentajeViajado(v0[i-1], v0[i+1])) + v0[i+1]11 × porcentajeViajado(v0[i-1], v0[i+1])))) ∧
  (p'11 == (v0[i-1]11 × ((1 - porcentajeViajado(v0[i-1], v0[i+1])) + v0[i+1]11 × porcentajeViajado(v0[i-1], v0[i+1]))))))
}

aux distancia (v0: Viaje, v1: Viaje) : Dist = √((v110 - v010)2 + (v111 - v011)2);
aux velocidadMedia (v0: Viaje, v1: Viaje) : ℤ =  $\frac{dist((v_{01}), (v_{11}))}{40}$ ;
aux tiempoViaje (v0: Viaje, v1: Viaje) : Tiempo =  $\frac{distancia(v_0, v_1)}{velocidadMedia(v_0, v_1)}$ ;
aux porcentajeViajado (v0: Viaje, v1: Viaje) : ℝ =  $\frac{40}{tiempoViaje(v_0, v_1)}$ ;
}

```

Esta expresión no tipa: errores[i] es un Tiempo, mientras que v es un Viaje, cuyos elementos no son de tipo Tiempo.

Este para todo dice: "para todo índice de v, v[i] tiene que ser igual a todos los puntos que posibles cuyo tiempo no está en errores".

Esto siempre será falso, porque v[i] no va a poder valer más de una cosa a la vez. Lo correcto sería decir: "para todo i: si está en rango y el tiempo de v<sub>0</sub>[i] no está en errores, entonces v[i] debe estar igual".

## 2.13. Ejercicio 13

```

proc histograma (in xs: seq⟨Viaje⟩, in bins: ℤ, out cuentas: seq⟨ℤ⟩, out limites: seq⟨ℝ⟩) {
  Pre {bins > 0 ∧ todosViajesValidos(xs)}
  Post {|cuentas| = |limites| - 1 ∧ elementosLimites(xs, bins, limites) ∧ elementosCuentas(xs, bins, limites, cuentas)}
  pred elementosLimites (xs: seq⟨Viaje⟩, bins: ℤ, limites: seq⟨ℝ⟩) {
    (∀i: ℤ)(0 ≤ i ≤ bins) →L
    ((∃menorVMax: ℝ)(esMenorVMax(menorVMax, xs)) ∧
    (∃mayorVMax: ℝ)(esMayorVMax(mayorVMax, xs)) ∧
    (limites[i] = (menorVMax + (i × ( $\frac{mayorVMax - menorVMax}{bins}$ ))))))
  }
  pred esMenorVMax (menorVMax: ℝ, xs: seq⟨Viaje⟩) {

```

```

( $\exists v : \text{Viaje}$ )( $v \in xs \wedge \text{esVelocidadMax}(\text{menorVMax}, v) \wedge$ 
( $\forall v : \text{Viaje}$ )( $v \in xs \longrightarrow (\exists vMax : \mathbb{R})(\text{esVelocidadMax}(vMax, v) \wedge \text{menorVMax} \leq vMax)$ ))
}
pred esMayorVMax ( $\text{mayorVMax} : \mathbb{R}, xs : \text{seq}\langle \text{Viaje} \rangle$ ) {

( $\exists v : \text{Viaje}$ )( $v \in xs \wedge \text{esVelocidadMax}(\text{mayorVMax}, v) \wedge$ 
( $\forall v : \text{Viaje}$ )( $v \in xs \longrightarrow (\exists vMax : \mathbb{R})(\text{esVelocidadMax}(vMax, v) \wedge \text{mayorVMax} \geq vMax)$ ))
}
pred esVelocidadMax ( $vMax : \mathbb{R}, v : \text{Viaje}$ ) {

( $\exists \text{viajeOrdenado} : \text{Viaje}$ )( $\text{esPermutacionOrdenada}(\text{viajeOrdenado}, v) \wedge$ 
( $\exists i : \mathbb{Z})(0 \leq i < |\text{viajeOrdenado}| - 1 \wedge_L \text{velocidad}(\text{viajeOrdenado}[i], \text{viajeOrdenado}[i + 1]) = vMax) \wedge$ 
( $\forall j : \mathbb{Z})(0 \leq j < |\text{viajeOrdenado}| - 1 \longrightarrow_L \text{velocidad}(\text{viajeOrdenado}[j], \text{viajeOrdenado}[j + 1]) \leq vMax)$ ))
}
aux velocidad ( $x, y : \text{Tiempo} \times \text{GPS}$ ) :  $\mathbb{R} =$ 
 $\frac{\text{dist}((x_1), (y_1))}{20}$ 
;
pred elementosCuentas ( $\text{bins} : \mathbb{Z}, xs : \text{seq}\langle \text{Viaje} \rangle, \text{cuentas} : \text{seq}\langle \mathbb{Z} \rangle, \text{limites} : \text{seq}\langle \mathbb{R} \rangle$ ) {

( $\forall i : \mathbb{Z})(0 \leq i \leq \text{bins}) \longrightarrow_L (\exists \text{velocidadesMax} : \text{seq}\langle \mathbb{R} \rangle)(\text{secuenciaVelocidadesMax}(\text{velocidadesMax}, xs) \wedge (\text{cuentas}[i] =$ 
( $\text{sumadorCuentas}(\text{bins}, \text{velocidadesMax}, \text{limites}[i], \text{limites}[i + 1])))$ ))
}
pred secuenciaVelocidadesMax ( $\text{velocidadesMax} : \text{seq}\langle \mathbb{R} \rangle, xs : \text{seq}\langle \text{Viaje} \rangle$ ) {

 $|\text{velocidadesMax}| = |xs| \wedge_L$ 
( $\forall i : \mathbb{Z})(0 \leq i < |\text{velocidadesMax}| \longrightarrow_L (\exists vMax : \mathbb{R})(\text{esVelocidadMax}(vMax, xs[i]) \wedge \text{velocidadesMax}[i] = vMax)$ ))
}
aux sumadorCuentas ( $\text{bins} : \mathbb{Z}, \text{velocidadesMax} : \text{seq}\langle \mathbb{R} \rangle, x : \mathbb{R}, y : \mathbb{R}$ ) :  $\mathbb{Z} =$ 
 $\sum_{i=0}^{|\text{velocidadesMax}|-1} (\text{ if } x \leq \text{velocidadesMax}[i] \leq y \text{ then } 1 \text{ else } 0 \text{ fi})$  ;
}

```