



# Trabajo práctico 1

## Especificación y WP

21/4/2024

AED

### Grupo IATOGYSWWBKAFJVCRWKR

Integrante	LU	Correo electrónico
Calo, Agustín	390/23	caloagustin4@gmail.com
Seri, Rafael Nicolás	362/23	rafaelnicoseri@gmail.com
Pintos Oliveira, Sol María Marcela	428/23	solpintosoliveira@gmail.com
Páez Torrico, Santiago	713/23	santiagopaez122@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. Especificación

## 1.1. redistribucionDeLosFrutos

```

proc redistribucionDeLosFrutos (in recursos: seq⟨ℝ⟩, in cooperan: seq⟨Bool⟩) : seq⟨ℝ⟩
  requiere {|recursos| = |cooperan|}
  requiere {todosPositivos(recursos)}
  asegura {|res| = |recursos|}
  asegura {(∀i : ℤ) (0 ≤ i < |res| →L if cooperan[i] then res[i] = totalARepartir(recursos, cooperan) else res[i] =
    recursos[i] + totalARepartir(recursos, cooperan) fi)}

aux totalARepartir (recursos: seq⟨ℝ⟩, cooperan: seq⟨Bool⟩) : ℝ =
  (∑i=0|recursos|-1 if cooperan[i] then recursos[i] else 0 fi)/|recursos|;

```

## 1.2. trayectoriaDeLosFrutosIndividualesALargoPlazo

```

proc trayectoriaDeLosFrutosIndividualesALargoPlazo (inout trayectorias: seq⟨seq⟨ℝ⟩⟩, in cooperan: seq⟨Bool⟩, in apues-
tas: seq⟨seq⟨ℝ⟩⟩, in pagos: seq⟨seq⟨ℝ⟩⟩, in eventos: seq⟨seq⟨ℝ⟩⟩)
  requiere {trayectorias = old(trayectorias)}
  requiere {|cooperan| = |pagos| = |apuestas| = |eventos| = |trayectorias|}
  requiere {(∀i : ℤ) (0 ≤ i < |pagos| →L todosPositivos(pagos[i]) ∧ todosPositivos(apuestas[i]))}
  requiere {(∀i : ℤ) (0 ≤ i < |trayectoria| →L trayectoria[i][0] > 0)}
  requiere {(∀j : ℤ) (-1 < j < |apuestas| →L ∑i=0|eventos[j]|-1 apuestas[j][i] = 1)}
  asegura {|trayectorias| = |old(trayectorias)|}
  asegura {(∀i : ℤ) (0 ≤ i < |old(trayectorias)| →L |trayectorias[i]| = |old(trayectorias)[i]| + |eventos[i]|)}
  asegura {(∀i : ℤ) (0 ≤ i < |old(trayectorias)| →L trayectorias[i][0] = old(trayectorias)[i][0])}
  asegura {(∀i : ℤ) (0 ≤ i < |old(trayectorias)| →L (∀j : ℤ) (0 ≤ j < |eventos[i]| →L trayectorias[i][j + 1] =
    if cooperan[i] then distribucion(aporteIndividual(trayectorias, apuestas, pagos, eventos, cooperan, i, j)) else aporte-
    Individual(trayectorias, apuestas, pagos, eventos, i, j) + distribucion(trayectorias, apuestas, pagos, eventos, cooperan,
    j) fi))}

aux distribucion (trayectorias: seq⟨seq⟨ℝ⟩⟩, apuestas: seq⟨seq⟨ℝ⟩⟩, pagos: seq⟨seq⟨ℝ⟩⟩, eventos: seq⟨seq⟨ℝ⟩⟩, cooperan:
seq⟨Bool⟩, m: ℕ) : ℝ =
  (∑k=0|cooperan|-1 if cooperan[k] then aporteIndividual(trayectorias, apuestas, pagos, eventos, k, m) else 0 fi)/|cooperan|;

```

## 1.3. trayectoriaExtrañaEscalera

```

proc trayectoriaExtrañaEscalera (in trayectorias: seq⟨ℝ⟩) : Bool
  requiere {|trayectoria| > 0}
  asegura {res = True ⇔ |trayectoria| = 1 ∨ (trayectoria[0] > trayectoria[1] ∧ ¬maximoLocal(trayectoria) ∧
    trayectoria[|trayectoria|-1] < trayectoria[|trayectoria|-2]) ∨ (trayectoria[|trayectoria|-1] > trayectoria[|trayectoria|-
    2] ∧ ¬maximoLocal(trayectoria) ∧ trayectoria[0] < trayectoria[1] ∨ (∃i : ℤ) (0 < i < |trayectoria|-1 ∧L (trayectoria[i] >
    trayectoria[i + 1] ∧ trayectoria[i] > trayectoria[i - 1]) ∧ (∀i : ℤ) (0 < j < |trayectoria| - 1 ∧L (trayectoria[j] >
    trayectoria[j + 1] ∧ trayectoria[j] > trayectoria[j - 1]) →L j = i)))}

pred maximoLocal (s: seq⟨ℝ⟩) {
  (∃i : ℤ) (0 < i < |s| - 1 ∧L (s[i] > s[i + 1] ∧ s[i] > s[i - 1]))
}

```

## 1.4. individuoDecideSiCooperarONo

```

proc individuoDecideSiCooperarONo (in individuo: ℕ, in recursos: seq⟨ℝ⟩, inout cooperan: seq⟨Bool⟩, in apuestas: seq⟨seq⟨ℝ⟩⟩,
in pagos: seq⟨seq⟨ℝ⟩⟩, in eventos: seq⟨seq⟨ℝ⟩⟩)
  requiere {cooperan = old(cooperan)}
  requiere {|cooperan| = |recursos| = |apuestas| = |pagos| = |eventos|}
  requiere {(∀i : ℤ) (0 ≤ i < |apuestas| →L todosPositivos(recursos) ∧ todosPositivos(apuestas[i]) ∧
    todosPositivos(pagos[i]))}
  requiere {0 ≤ individuo < |cooperan|}
  asegura {|cooperan| = |old(cooperan)|}
  asegura {(∀i : ℤ) (0 ≤ individuo < |cooperan| ∧ i ≠ individuo →L cooperan[i] = old(cooperan)[i])}
  asegura {(∃s, p : seq⟨seq⟨ℝ⟩⟩) (|s| = |p| = |cooperan| ∧ (∀n : ℤ) (0 ≤ n < |s| ∧L |s[n]| = |p[n]| = (|eventos| + 1) ∧
    s[n][0] = p[n][0] = recursos[n] ∧ (∀k : ℤ) (0 < k < |s[n]| →L s[n][k] = (if old(cooperan)[k] ∨ k = individuo then 0 else
    aporteIndividual(s, apuestas, pagos, eventos, n, k) fi) + distribucionCoop(s, apuestas, pagos, eventos, old(cooperan),
    k, individuo) ∧ p[n][k] = (if ¬(old(cooperan)[k]) ∨ k = individuo then aporteIndividual(p, apuestas, pagos, eventos, n, k)

```

```

    else 0 fi) + distribucionNoCoop(p, apuestas, pagos, eventos, old(cooperan), k, individuo)))  $\wedge_L$  cooperan[individuo] =
    p[individuo][p[individuo] - 1]  $\leq$  s[individuo][s[individuo] - 1])}

aux distribucionCoop (trayectorias: seq(seq( $\mathbb{R}$ )), apuestas: seq(seq( $\mathbb{R}$ )), pagos: seq(seq( $\mathbb{R}$ )), eventos: seq(seq( $\mathbb{N}$ )), cooperan: seq(Bool), m:  $\mathbb{N}$ , individuo:  $\mathbb{N}$ ) :  $\mathbb{R}$  =
( $\sum_{k=0}^{|cooperan|-1}$  if cooperan[k]  $\vee$  k = individuo then aporteIndividual(trayectorias, apuestas, pagos, eventos, k, m)
else 0 fi)/|cooperan|;
aux distribucionNoCoop (trayectorias: seq(seq( $\mathbb{R}$ )), apuestas: seq(seq( $\mathbb{R}$ )), pagos: seq(seq( $\mathbb{R}$ )), eventos: seq(seq( $\mathbb{N}$ )), cooperan: seq(Bool), m:  $\mathbb{N}$ , individuo:  $\mathbb{N}$ ) :  $\mathbb{R}$  =
( $\sum_{k=0}^{|cooperan|-1}$  if cooperan[k]  $\wedge$  k  $\neq$  individuo then aporteIndividual(trayectorias, apuestas, pagos, eventos, k, m)
else 0 fi)/|cooperan|;

```

## 1.5. individuoActualizaApuesta

```

proc individuoActualizaApuesta (in individuo:  $\mathbb{N}$ , in recursos: seq( $\mathbb{R}$ ), in cooperan: seq(Bool), in/out apuestas: seq(seq(Bool)),
in pagos: seq(seq( $\mathbb{R}$ )), in eventos: seq(seq( $\mathbb{N}$ )))
    requiere {apuestas = old(apuestas)}
    requiere {|cooperan| = |recursos| = |apuestas| = |pagos| = |eventos|}
    requiere {0  $\leq$  individuo < |cooperan|}
    asegura {|apuestas| = |old(apuestas)|}
    asegura {( $\forall i : \mathbb{Z}$ ) (-1 < i < |apuestas|  $\longrightarrow_L$  |apuestas[i] = |old(apuestas)[i])}
    asegura {( $\exists p, s : seq(seq(\mathbb{R}))$ )( $\exists mejorApuesta : seq(\mathbb{R})$ )( $(\forall posibleApuesta : seq(\mathbb{R}))$  (ultElem(p, recursos, old(apuestas),
    pagos, cooperan, eventos, individuo, mejorApuesta)  $\geq$  ultElem(s, recursos, old(apuestas), pagos, cooperan, eventos,
    individuo, posibleApuesta)  $\longrightarrow_L$  apuestas[individuo] = mejorApuesta))}

aux ultElem (t: seq(seq( $\mathbb{R}$ )), recursos: seq( $\mathbb{R}$ ), apuestas: seq(seq( $\mathbb{R}$ )), pagos: seq(seq( $\mathbb{R}$ )), cooperan: seq(Bool), eventos:
seq( $\mathbb{N}$ ), individuo:  $\mathbb{N}$ , posibleApuesta: seq( $\mathbb{N}$ )) :  $\mathbb{R}$  =
if trayectoriaPosible(t, recursos, apuestas, pagos, cooperan, eventos, individuo, posibleApuesta) then t[individuo][t[individuo] -
1] else -1 fi;
pred trayectoriaPosible (t: seq(seq( $\mathbb{R}$ )), recursos: seq( $\mathbb{R}$ ), apuestas: seq(seq( $\mathbb{R}$ )), pagos: seq(seq( $\mathbb{R}$ )), cooperan: seq(Bool),
eventos: seq( $\mathbb{N}$ ), individuo:  $\mathbb{N}$ , posibleApuesta: seq( $\mathbb{N}$ )) {
    |posibleApuesta| = |apuestas[individuo]|  $\wedge$  sumElem(posibleApuesta) = 1  $\wedge$  todosPositivos(posibleApuesta) |t| = |cooperan|  $\wedge$ 
    ( $\forall i : \mathbb{Z}$ ) (-1 < i < |t|  $\wedge_L$  |t[i]| = (|eventos| + 1)  $\wedge$  t[i][0] = recursos[i]  $\wedge$  ( $\forall j : \mathbb{Z}$ ) (0  $\leq$  j < |t[i]|  $\longrightarrow_L$  t[i][j + 1] =
    (if cooperan[i] then 0 else aporteIndDiferido(t[i], apuestas[i], pagos[i], eventos[i], i, j, individuo, posibleApuesta) fi) +
    distribucionDiferida(t[i], apuestas[i], pagos[i], eventos[i], cooperan, i, j, individuo, posibleApuesta)))
}
aux aporteIndDiferido (trayectoria: seq( $\mathbb{R}$ ), apuestas: seq( $\mathbb{R}$ ), pagos: seq( $\mathbb{R}$ ), eventos: seq( $\mathbb{N}$ ), k:  $\mathbb{N}$ , m:  $\mathbb{N}$ , individuo:  $\mathbb{N}$ ,
apuestaInd: seq( $\mathbb{R}$ )) :  $\mathbb{R}$  =
if k = individuo then trayectorias[m]  $\cdot$  apuestaInd[eventos[m]]  $\cdot$  pagos[eventos[m]] else trayectorias[m]  $\cdot$  apuestas[eventos[m]]  $\cdot$ 
pagos[eventos[m]] fi;
aux distribucionDiferida (trayectoria: seq( $\mathbb{R}$ ), apuestas: seq( $\mathbb{R}$ ), pagos: seq( $\mathbb{R}$ ), eventos: seq( $\mathbb{N}$ ), cooperan: seq(Bool), k:
 $\mathbb{N}$ , m:  $\mathbb{N}$ , individuo:  $\mathbb{N}$ , apuestaInd: seq( $\mathbb{R}$ )) :  $\mathbb{R}$  =
( $\sum_{k=0}^{|cooperan|-1}$  if cooperan[k] then aporteIndDiferido(trayectorias, apuestas, pagos, eventos, k, m, individuo) else 0 fi)/|cooperan|;
aux sumElem (s: seq( $\mathbb{R}$ )) :  $\mathbb{R}$  =
 $\sum_{i=0}^{|s|-1} s[i]$ ;

```

## Auxiliares y predicados globales

```

pred todosPositivos (s: seq( $\mathbb{R}$ )) {
    ( $\forall i : \mathbb{Z}$ ) (0  $\leq$  i < |s|  $\longrightarrow_L$  s[i] > 0)
}
aux aporteIndividual (trayectorias: seq(seq( $\mathbb{R}$ )), apuestas: seq(seq( $\mathbb{R}$ )), pagos: seq(seq( $\mathbb{R}$ )), eventos: seq(seq( $\mathbb{N}$ )), k:  $\mathbb{N}$ ,
m:  $\mathbb{N}$ ) :  $\mathbb{R}$  = trayectorias[k][m]  $\cdot$  apuestas[k][eventos[k][m]]  $\cdot$  pagos[k][eventos[k][m]] ;

```

## 2. Demostraciones de correctitud

Demostrar que la siguiente especificación es correcta respecto de su implementación.

La función **frutoDelTrabajoPuramenteIndividual** calcula, para el ejemplo de apuestas al juego de cara o seca, cuánto se ganaría si se juega completamente solo. Se contempla que el evento True es cuando sale cara.

```

proc frutoDelTrabajoPuramenteIndividual (in recurso:  $\mathbb{R}$ , in apuesta:  $\langle s : \mathbb{R}, c : \mathbb{R} \rangle$ , in pago:  $\langle s : \mathbb{R}, c : \mathbb{R} \rangle$ , in eventos:
seq(Bool), out res:  $\mathbb{R}$ )
    requiere {apuestac + apuestas = 1  $\wedge$  pagoc > 0  $\wedge$  pagos > 0  $\wedge$  apuestac > 0  $\wedge$  apuestas > 0  $\wedge$  recurso > 0}
    asegura {res = recurso(apuestac, pagoc)#apariciones(eventos, T)(apuestas, pagos)#apariciones(eventos, F)}

```

Donde  $\#apariciones(eventos, T)$  es el auxiliar utilizado en la teórica, y  $\#(eventos, T)$  es su abreviación.

```
1  res := recursos
2  i := 0
3  while (i < |eventos|) do
4      if eventos[i] then
5          res := (res * apuesta.c) * pago.c
6      else
7          res := (res * apuesta.s) * pago.s
8      endif
9      i := i + 1
10 endwhile
```