



# Trabajo Practico

19 de mayo de 2024

Algoritmos y Estructuras de Datos

## Grupo AlgoTango

Integrante	LU	Correo electrónico
Orsi, Lautaro Manuel	689/23	Lautaorsi@gmail.com
Zerbetto De Palma, Gerardo Gabriel	900/22	g.zerbetto@gmail.com
Simoza Sanchez, Valeria Andreina	1027/22	vsimoza.vs@gmail.com
Prieto, Matias	382/23	matiasprieto2003@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. Redistribucion De Los Frutos

```

proc redistribucionDeLosFrutos (in recursos : seq⟨ℤ⟩, in cooperan: seq⟨Bool⟩) : seq⟨ℤ⟩
  requiere {|recursos| = |cooperan| ∧ recursosValidos(recursos)}
  asegura {|res| = |recursos| ∧L (∀i : ℤ)(0 ≤ i < |recursos| →L ( (cooperan[i] = true) →L res[i] =
    
$$\frac{\text{fondoMonetario}(\text{recursos}, \text{cooperan})}{|\text{recursos}|}) \wedge ((\text{cooperan}[i] = \text{false}) \rightarrow_L \text{res}[i] = \text{recursos}[i] +$$


$$\frac{\text{fondoMonetario}(\text{recursos}, \text{cooperan})}{|\text{recursos}|})) \}}
  aux fondoMonetario (recursos:seq⟨ℝ⟩,cooperan:seq⟨Bool⟩) : ℝ = 
$$\sum_{j=0}^{|\text{recursos}|-1} \left( \text{if } (\text{cooperan}[j] = \text{true}) \text{ then } (\text{recursos}[j]) \text{ else } \right.$$

    (0) fi);
  pred recursosValidos (recursos:seq⟨ℝ⟩) {
    (∀i : ℤ)(0 ≤ i < |recursos| →L recursos[i] > 0)
  }$$

```

### Desarrollo 1.1

Para este ejercicio implementamos un aux *fondoMonetario* que calcula —en base a las listas *cooperan* y *recursos*— una sumatoria de la totalidad de recursos que seran redistribuidos al finalizar el paso temporal, sumando al fondo los recursos de aquellos que cooperen y sin sumar los que no.

Empleamos este aux para luego calcular el recurso de cada individuo, si este decidia cooperar su recurso sera la division equitativa y si no cooperaba su recurso sera la plata obtenida mas la division equitativa.

## 1.2. Trayectoria De Los Frutos Individuales a Largo Plazo

```

proc trayectoriaDeLosFrutosIndividualesALargoPlazo (inout trayectorias: seq⟨seq⟨ℝ⟩⟩, in cooperan: seq⟨Bool⟩, in apuestas: seq⟨seq⟨ℝ⟩⟩, in pagos: seq⟨seq⟨ℝ⟩⟩, in eventos: seq⟨seq⟨ℕ⟩⟩)
  requiere {trayectoriasValidas(trayectorias)
    ∧ mismaLongitud(trayectorias, pagos, apuestas, eventos, cooperan)
    ∧ pagosPositivos(pagos)
    ∧ apuestasValidas(apuestas)
    ∧ longitudApuestasPagos(apuestas, pagos)
    ∧ longitudSublistas(pagos)
    ∧ longitudSublistas(apuestas)
    ∧ longitudSublistas(eventos)
    ∧ longitudEventosApuestas(eventos, apuestas)}
  asegura {|trayectorias| = |old(trayectorias)| ∧ (∀i : ℤ)(0 ≤ i < |old(trayectorias)| →L (trayectorias[i][0] =
    (old(trayectorias)[i][0]) ∧L (∀j : ℤ)(0 ≤ j < |eventos[i]| →L trayectorias[i][j+1] = calculoDeRecursosSegunCooperacion(
    trayectorias, pagos, apuestas, eventos, cooperan, i, j)))}
  aux calculoDeRecursosSegunCooperacion (trayectorias,pagos,apuestas:seq⟨seq⟨ℝ⟩⟩,eventos:seq⟨seq⟨ℕ⟩⟩,cooperan:
    seq⟨Bool⟩,individuo:ℤ,ronda:ℤ) : ℝ = if cooperan[individuo] = true then fondoMonetarioRepartido(trayectorias, pagos,
    apuestas, eventos, cooperan, ronda) else calculoRecursos(trayectorias[individuo][ronda],
    pagos[individuo][eventos[individuo][ronda]], apuestas[individuo][eventos[individuo][ronda]]) + fondoMonetarioRepartido(
    trayectorias, pagos, apuestas, eventos, cooperan, ronda) fi;
  aux calculoRecursos (recurso,pago,apuesta:ℝ) : ℝ = recurso * pago * apuesta;
  aux fondoMonetario (trayectorias,pagos,apuestas:seq⟨seq⟨ℝ⟩⟩,eventos : seq⟨seq⟨ℕ⟩⟩,cooperan : seq⟨Bool⟩,ronda :
    ℤ)) : ℝ = 
$$\sum_{h=0}^{|\text{cooperan}|-1} \text{if } \text{cooperan}[h] = \text{true} \text{ then } \text{calculoRecursos}(\text{trayectorias}[h][\text{ronda}], \text{pagos}[h][\text{eventos}[h][\text{ronda}]],$$

    apuestas[h][eventos[h][ronda]]) else 0 fi;
  aux fondoMonetarioRepartido (trayectorias,pagos,apuestas:seq⟨seq⟨ℝ⟩⟩,eventos:seq⟨seq⟨ℕ⟩⟩,cooperan:seq⟨Bool⟩,ronda:ℤ)
    : ℝ = 
$$\frac{\text{fondoMonetario}(\text{trayectorias}, \text{pagos}, \text{apuestas}, \text{eventos}, \text{cooperan}, \text{ronda})}{|\text{cooperan}|};$$

  aux sumaApuestas (apuestas:seq⟨seq⟨ℝ⟩⟩, individuo: ℤ) : ℝ = 
$$\sum_{h=0}^{|\text{apuestas}[\text{individuo}|-1]} \text{apuestas}[\text{individuo}][h];$$

  pred mismaLongitud (lista: seq⟨T⟩,pagos,apuestas:seq⟨seq⟨ℝ⟩⟩,eventos:seq⟨seq⟨ℕ⟩⟩,cooperan:seq⟨Bool⟩) {
    |lista| = |cooperan| = |apuestas| = |pagos| = |eventos|
  }
  pred trayectoriasValidas (trayectorias:seq⟨seq⟨ℝ⟩⟩) {

```

```

|trayectorias| > 0 ∧ (∀i : ℤ)(0 ≤ i < |trayectorias| →L |trayectorias[i]| = 1 ∧L (∀x : ℝ)(x ∈ trayectorias[i] →L x > 0))
}
pred pagosPositivos (pagos:seq⟨seq⟨ℝ⟩⟩) {
  (∀i, j : ℤ)(0 ≤ i < |pagos| ∧L (0 ≤ j < |pagos[i]| →L pagos[i][j] > 0))
}
pred apuestasValidas (apuestas:seq⟨seq⟨ℝ⟩⟩) {
  (∀i, j : ℤ)(0 ≤ i < |apuestas| →L sumaApuestas(apuestas, i) = 1 ∧L (0 ≤ j < |apuestas[i]| →L 0 < apuestas[i][j] < 1))
}
pred longitudApuestasPagos (apuestas,pagos:seq⟨seq⟨ℝ⟩⟩) {
  (∀i : ℤ)(0 ≤ i < |apuestas| →L |apuestas[i]| = |pagos[i]|)
}
pred longitudSublistas (lista:seq⟨seq⟨T⟩⟩) {
  (∀i : ℤ)(0 ≤ i < |lista| →L (|lista[0]| > 0 ∧L |lista[0]| = |lista[i]|))
}
pred longitudEventosApuestas (eventos:seq⟨seq⟨ℕ⟩⟩, apuestas:seq⟨seq⟨ℝ⟩⟩) {
  (∀i, j : ℤ)(0 ≤ i < |eventos| ∧L 0 ≤ j < |eventos[i]| →L eventos[i][j] < apuestas[i])
}

```

### Desarrollo 1.2

En este ejercicio actualizamos la trayectoria de cada individuo basandonos en sus recursos y evento correspondiente por ronda (con su respectivo pago y apuesta). Usamos el aux *calculoDeRecursosSegunCooperacion* en el que calculamos los recursos que le quedarán al individuo teniendo en cuenta primeramente si coopera o no. En el primer caso recibe recursos únicamente del fondo monetario distribuido entre el total de los individuos (aux *fondoMonetarioRepartido*) y en el otro caso se calcula su ganancia individual (aux *calculoRecursos*) y se suma el fondo monetario distribuido.

### Aclaración

En el pred apuestasValidas definimos por interpretación que las apuestas individuales son mayores a 0 y menores a 1, ya que no encontramos referencias en el tp al rango exacto que deben tener.

## 1.3. Trayectoria Extraña Escalera

```

proc trayectoriaExtrañaEscalera (in trayectoria: seq⟨ℝ⟩) : Bool
  requiere {|trayectoria| > 0}
  asegura {res = true ⇔ ((|trayectoria| = 1) ∨ (|trayectoria| = 2 ∧ ¬(todosIguales(trayectoria)) ∨L (|trayectoria| > 2 ∧ hayUnicoMax(trayectoria))))}
  pred hayUnicoMax (trayectoria: seq⟨ℝ⟩) {
    (CantidadMaximos(trayectoria) = 1 ∧ ¬(UnMaxEnLimite(trayectoria))) ∨ (CantidadMaximos(trayectoria) = 0 ∧ UnMaxEnLimite(trayectoria))
  }
  pred todosIguales (lista:seq⟨ℝ⟩) {
    (∀i, j : ℤ)(0 ≤ i < |lista| ∧ 0 ≤ j < |lista| ∧ i ≠ j →L lista[i] = lista[j])
  }
  pred UnMaxEnLimite (Trayectoria:seq⟨ℝ⟩) {
    (lista[0] > lista[1] ∧ lista[|lista| - 1] ≤ lista[|lista| - 2]) ∨ (lista[0] ≤ lista[1] ∧ lista[|lista| - 1] > lista[|lista| - 2])
  }
  aux CantidadMaximos (lista: seq⟨ℝ⟩) : ℤ =  $\sum_{i=1}^{|lista|-2}$  if  $\left( (lista[i-1] < lista[i]) \wedge (lista[i] > lista[i+1]) \right)$  then (1) else (0) fi ;

```

### Desarrollo 1.3:

En este ejercicio, utilizamos una separacion en 3 distintos casos, 2 de ellos unicos y uno generalizado, es importante notar que la trayectoria que se recibe es una lista que representa los recursos a medida que avanzan las rondas (o pasos temporales). El primero, siendo que  $|trayectoria|$  es 1 (se juega una ronda) sabemos que sera maximo local pues no tiene vecinos. El segundo, siendo que  $|trayectoria|$  es 2 y que, si son distintos, trivialmente alguno es mayor que el otro siendo entonces el maximo local. El ultimo y mas general, dada una secuencia de mas de 2 elementos se busca si efectivamente hay algun numero mayor que sus numeros vecinos y que ademas sea el **unico** con esa propiedad en la secuencia.

## 1.4. Individuo Decide Si Cooperar O No

```

proc individuoDecideSiCooperarONo (in individuo: ℕ, in recursos: seq⟨ℝ⟩, inout cooperan: seq⟨Bool⟩, in apuestas: seq⟨seq⟨ℝ⟩⟩, in pagos: seq⟨ℝ⟩, in eventos: seq⟨seq⟨ℕ⟩⟩)

```

```

requiere {mismaLongitud(recursos, pagos, apuestas, eventos, cooperan)
 $\wedge$  individuo < |eventos|
 $\wedge$  pagosPositivos(pagos)
 $\wedge$  apuestasValidas(apuestas)
 $\wedge$  longitudApuestasPagos(apuestas, pagos)
 $\wedge$  longitudSublistas(apuestas)
 $\wedge$  longitudSublistas(eventos)
 $\wedge$  longitudSublistas(pagos)
 $\wedge$  longitudEventosApuestas(eventos, apuestas)}
asegura {( $\exists$  Trayectorias, TrayectoriasNegadas : seq(seq( $\mathbb{Z}$ ))) ( $\exists$  CooperanNegada seq( $\mathbb{Bool}$ ))
( $\left( \left( ValidarTrayectoria(Trayectorias, cooperan, apuestas, pagos, eventos, recursos) \right) \right.$ 
 $\wedge$  ( $\neg$ (old(cooperan)[individuo]) = cooperanNegada[Individuo])
 $\wedge$  ( $\forall i : \mathbb{Z}$ ) ( $0 \leq i < |cooperan| \wedge i \neq Individuo \wedge cooperan[i] = cooperanNegada[i]$ )
 $\wedge$  ( $ValidarTrayectorias(TrayectoriasNegada, cooperanNegada, apuestas, pagos, eventos, recursos)$ ))
 $\rightarrow_L$  ( $\left( \left( (Trayectorias[Individuo][|eventos[0]| - 1] \geq TrayectoriasNegada[Individuo][|eventos[0]| - 1]) \right. \right.$ 
 $\wedge$  cooperan[Individuo] = old(cooperan[Individuo])
 $\vee$  ( $(Trayectorias[Individuo][|eventos[0]| - 1] < TrayectoriasNegada[Individuo][|eventos[0]| - 1])$ 
 $\wedge$  cooperan[Individuo] = cooperanNegada[Individuo])
 $\left. \right)$ )}
pred ValidarTrayectoria (trayectorias: seq(seq( $\mathbb{R}$ )), cooperan: seq( $\mathbb{Bool}$ ), apuestas: seq(seq( $\mathbb{R}$ )), pagos: seq( $\mathbb{R}$ ), eventos:
seq(seq( $\mathbb{N}$ )), recursos: seq( $\mathbb{R}$ )) {
  ( $\forall i : \mathbb{Z}$ ) ( $0 \leq i < |recursos| \rightarrow_L (trayectorias[i][0] = recursos[i]) \wedge_L$ 
  ( $\forall j : \mathbb{Z}$ ) ( $0 \leq j < |eventos[i]| \rightarrow_L$ 
  trayectorias[i][j+1] = calculoDeRecursosSegunCooperacion(trayectorias, pagos, apuestas, eventos, cooperan, i, j)))
}

```

\*preds pagosPositivos, apuestasValidas, longitudApuestasPagos, longitudSublistas, mismaLongitud, longitudEventosApuestas y calculoDeRecursosSegunCooperacion (con sus dependencias) declarados en ejercicio 1.2

#### Desarrollo 1.4:

La dificultad de este ejercicio aparece en no tener la secuencia de trayectorias individuales, viendo que calcularlas llevaria mucha complejidad planteamos un cuantificador y predicamos sobre el, analizando las posibles secuencias *trayectorias* y *trayectoriasNegada* (siendo esta ultima la que corresponderia al caso de la negacion del booleano de cooperacion del individuo) podemos, utilizando los predicados y auxiliares del ejercicio 1.2 y la lista de recursos verificar que efectivamente estas listas son las que deberiamos obtener si calcularamos las trayectorias al largo plazo.

Para validar esta trayectoria comparamos que las primeras posiciones de la trayectoria correspondan a los recursos (basicamente, que el punto de partida de la trayectoria sea correcta) y luego, verificamos que cada posicion  $N$  de la lista, sea correspondiente a calcular los recursos  $N$  rondas para cada individuo, partiendo de la base de la ronda 0 (recursos). Es importante notar, que para el caso *trayectoriasNegada* utilizaremos la lista *cooperanNegada*, pues el calculo dependera de si el individuo coopera o no.

Al final, podemos observar que utilizando la comparacion de la ultima posicion de *trayectoria* del individuo (basandonos en la cantidad de eventos) con la ultima posicion de la *trayectoriaNegada* asignamos, segun corresponda el valor original (en el primer caso) y el valor negado en el segundo.-

## 1.5. Individuo Actualiza Apuesta

```

proc individuoActualizaApuesta (in individuo:  $\mathbb{N}$ , in recursos: seq( $\mathbb{R}$ ), in cooperan: seq( $\mathbb{Bool}$ ), inout apuestas: seq(seq( $\mathbb{R}$ )),
in pagos: seq(seq( $\mathbb{R}$ )), in eventos: seq(seq( $\mathbb{N}$ )))
requiere {individuo < |recursos|  $\wedge$  pagosPositivos(pagos)  $\wedge$  mismaLongitud(recursos, cooperan, apuestas, pagos, eventos)
 $\wedge$  apuestasValidas(apuestas)  $\wedge$  longitudApuestasPagos(apuestas, pagos)
 $\wedge$  longitudEventosApuestas(eventos, apuestas)  $\wedge$  longitudSublistas(pagos)  $\wedge$  longitudSublistas(apuestas)  $\wedge$ 
longitudSublistas(eventos)}
asegura {( $\forall i : \mathbb{R}$ )
( $0 \leq i < |apuestas| \wedge_L$ 
esMaximaGanancia(old(apuestas)[i], individuo, old(apuestas), pagos, eventos)  $\rightarrow_L$ 
apuestas[individuo] = old(apuestas)[i])}

```

```

)
pred esMaximaGanancia (apuesta: seq⟨ℝ⟩, individuo:ℕ, apuestas: seq⟨seq⟨ℝ⟩⟩, recursos: seq⟨ℝ⟩, cooperan: seq⟨Bool⟩,
pagos: seq⟨seq⟨ℝ⟩⟩, eventos: seq⟨seq⟨ℕ⟩⟩) {
  (
    cooperan[individuo] = true ∧ (∀j : ℝ)(0 ≤ j < |apuestas| →L
      recursosCoopFinal(setAt(apuestas, individuo, apuestas[j]), pagos, eventos, cooperan, recursos) ≤
      recursosCoopFinal(setAt(apuestas, individuo, apuesta), pagos, eventos, cooperan, recursos))
  ) ∨
  (
    cooperan[individuo] = false ∧ (∀j : ℝ)(0 ≤ j < |apuestas| →L
      recursosNoCoopFinal(apuestas[j], pagos[individuo], eventos[individuo], recursos[individuo]) ≤
      recursosNoCoopFinal(apuesta, pagos[individuo], eventos[individuo], recursos[individuo]))
  )
}
aux recursosNoCoopFinal (in apuesta: seq⟨ℝ⟩, in pago: seq⟨ℝ⟩, in evento: seq⟨ℕ⟩, in recurso: ℝ) : ℝ =
  (recurso. ∏i=0|evento|-1 (pago[evento[i]].apuesta[evento[i]])) ;
aux recursosCoopFinal (in apuestas: seq⟨seq⟨ℝ⟩⟩, in pagos: seq⟨seq⟨ℝ⟩⟩, in eventos: seq⟨seq⟨ℕ⟩⟩, in cooperan: seq⟨Bool⟩, in
recursos: seq⟨ℝ⟩) : ℝ =
  (recursoPrimerEvento(apuestas, pagos, head(eventos), recursos, cooperan) *
  crecimientoCooperativo(apuestas, pagos, eventos, recursos, cooperan)) ;
aux recursosPrimerEvento (in apuestas: seq⟨seq⟨ℝ⟩⟩, in pagos: seq⟨seq⟨ℝ⟩⟩, in evento: seq⟨ℕ⟩, in cooperan: seq⟨Bool⟩, in
recursos: seq⟨ℝ⟩) : ℝ =
  ∑i=0|cooperan|-1 if cooperan[i] = true then recursos[i] * apuestas[i][evento[i]] * pagos[i][evento[i]] else 0 fi *
  (∑i=0|cooperan|-1 if cooperan[i] = true then 1 else 0 fi)-1 ;
aux crecimientoCooperativo (in apuestas: seq⟨seq⟨ℝ⟩⟩, in pagos: seq⟨seq⟨ℝ⟩⟩, in eventos: seq⟨seq⟨ℕ⟩⟩, in cooperan:
seq⟨Bool⟩, in recursos: seq⟨ℝ⟩) : ℝ =
  (∏i=1|eventos[0]|-1 ∑h=0|cooperan|-1 if cooperan[h] = true then apuestas[h][eventos[h][i]] * pagos[h][evento[h][i]] else 0 fi) *
  (∑i=0|cooperan|-1 if cooperan[i] = true then 1 else 0 fi)-(|eventos[0]|-1) ;
pred apuestaValida (in apuesta: seq⟨ℝ⟩) {
  (∑i=0|apuesta|-1 apuesta[i]) = 1
}

```

\*pagosPositivos, mismaLongitud, apuestasValidas, longitudEventosApuestas, longitudSublistas y longitudApuestasPagos definidos en el ejercicio 1.2

### Desarrollo 1.5:

En estas especificación se plantea un cuantificador que permite describir la nueva apuesta elegida por el individuo en términos de si genera una máxima ganancia, a su vez elegida de entre las apuestas disponibles. Para establecer eso, con el predicado esMaximaGanancia se evalúa si la apuesta que se pretende cuantificar cumple con la condición de generar la máxima ganancia según si el individuo coopera o no. Para ello, compara el recurso final obtenido con la apuesta evaluada contra el recurso final obtenido con todas las otras apuestas disponibles. Luego las funciones auxiliares recursosCoopFinal y recursosNoCoopFinal computan los recursos finales según el caso.

## 2. Demostracion de correctitud

```

proc frutoDelTrabajoPuramenteIndividual (in recurso : ℝ, in apuesta : ⟨ s : ℝ, c : ℝ ⟩, in pago : ⟨ s : ℝ, c : ℝ ⟩, in eventos
: seq⟨Bool⟩, out res : ℝ)

```

```

  requiere {apuestac + apuestas = 1 ∧ pagoc > 0 ∧ pagos > 0 ∧ apuestac > 0 ∧ apuestas > 0 ∧ recurso > 0}
  asegura {res = recurso(apuestacpagoc)#apariciones(eventos,True)(apuestaspagos)#apariciones(eventos,False)}

```

Donde #apariciones(eventos, True) es el auxiliar utilizado en la teorica, y #(eventos, True) es su abreviacion

```

1 | res := recurso;
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

```

Para demostrar que la especificacion es correcta respecto a la implementacion, hay que demostrar la tripla de Hoare del requiere, la implementacion y el asegura.

Primero demostramos la correctitud del ciclo. Para esto planteamos:

$$\begin{aligned} P_c &\equiv i = 0 \wedge res = recurso \wedge apuesta_c + apuesta_s = 1 \wedge pago_c > 0 \wedge pago_s > 0 \wedge apuesta_c > 0 \wedge apuesta_s > 0 \wedge recurso > 0 \\ Q_c &\equiv res = recurso(apuesta_c, pago_c)^{\#(eventos, True)}(apuesta_s, pago_s)^{\#(eventos, False)} \\ I &\equiv 0 \leq i \leq |eventos| \wedge_L res = recurso(apuesta_c, pago_c)^{\#(subseq(eventos, 0, i), True)}(apuesta_s, pago_s)^{\#(subseq(eventos, 0, i), False)} \\ B &\equiv i < |eventos| \\ Fv &\equiv |eventos| - i \end{aligned}$$

Queremos ver que se cumplan:

- 1)  $P_c \longrightarrow I$
- 2)  $\{I \wedge B\} while....endwhile \{I\}$
- 3)  $I \wedge \neg B \longrightarrow Q_c$
- 4)  $\{I \wedge B \wedge v_0 = Fv\} while....endwhile \{Fv < v_0\}$
- 5)  $I \wedge Fv \leq 0 \longrightarrow \neg B$

1)  $P_c \longrightarrow I$ :

$$(i = 0 \wedge res = recurso \wedge apuesta_c + apuesta_s = 1 \wedge pago_c > 0 \wedge pago_s > 0 \wedge apuesta_c > 0 \wedge apuesta_s > 0 \wedge recurso > 0) \longrightarrow (0 \leq i \leq |eventos| \wedge_L res = recurso(apuesta_c, pago_c)^{\#(subseq(eventos, 0, i), True)}(apuesta_s, pago_s)^{\#(subseq(eventos, 0, i), False)})$$

Aumo que el antecedente es verdadero. Como  $i = 0$ , reemplazo  $i$  por  $0$

$$(res = recurso \wedge apuesta_c + apuesta_s = 1 \wedge pago_c > 0 \wedge pago_s > 0 \wedge apuesta_c > 0 \wedge apuesta_s > 0 \wedge recurso > 0) \longrightarrow (0 \leq 0 \leq |eventos| \wedge_L res = recurso(apuesta_c, pago_c)^{\#(subseq(eventos, 0, 0), True)}(apuesta_s, pago_s)^{\#(subseq(eventos, 0, 0), False)})$$

$$\equiv (res = recurso \wedge apuesta_c + apuesta_s = 1 \wedge pago_c > 0 \wedge pago_s > 0 \wedge apuesta_c > 0 \wedge apuesta_s > 0 \wedge recurso > 0) \longrightarrow res = recurso(apuesta_c, pago_c)^{\#(<>, True)}(apuesta_s, pago_s)^{\#(<>, False)}$$

$$\equiv (res = recurso \wedge apuesta_c + apuesta_s = 1 \wedge pago_c > 0 \wedge pago_s > 0 \wedge apuesta_c > 0 \wedge apuesta_s > 0 \wedge recurso > 0) \longrightarrow res = recurso(apuesta_c, pago_c)^0(apuesta_s, pago_s)^0$$

$$\equiv (res = recurso \wedge apuesta_c + apuesta_s = 1 \wedge pago_c > 0 \wedge pago_s > 0 \wedge apuesta_c > 0 \wedge apuesta_s > 0 \wedge recurso > 0) \longrightarrow res = recurso$$

Como siempre es verdadero, se cumple  $P_c \longrightarrow I$

2)  $\{I \wedge B\} while....endwhile \{I\}$ : Para demostrar esto hay que demostrar  $I \wedge B \longrightarrow wp(while....endwhile, I)$

$$wp(while....endwhile, I) \equiv wp(if...endif, wp(i := i+1, I) \equiv wp(if...endif, def(i) \wedge_L (0 \leq i+1 \leq |eventos| \wedge_L res = recurso(apuesta_c, pago_c)^{\#(subseq(eventos, 0, i+1), True)}(apuesta_s, pago_s)^{\#(subseq(eventos, 0, i+1), False)})))$$

$$\equiv (def(eventos[i]) \wedge_L (eventos[i] = true \wedge wp(res := res * apuesta_c * pago_c, 0 \leq i+1 \leq |eventos| \wedge_L res = recurso(apuesta_c, pago_c)^{\#(subseq(eventos, 0, i+1), True)}(apuesta_s, pago_s)^{\#(subseq(eventos, 0, i+1), False)})) \vee (eventos[i] = false \wedge wp(res := res * apuesta_s * pago_s, 0 \leq i+1 \leq |eventos| \wedge_L res = recurso(apuesta_c, pago_c)^{\#(subseq(eventos, 0, i+1), True)}(apuesta_s, pago_s)^{\#(subseq(eventos, 0, i+1), False)})))$$

$$\equiv 0 \leq i < |eventos| \wedge_L (eventos[i] = true \wedge 0 \leq i+1 \leq |eventos| \wedge_L res = recurso(apuesta_c, pago_c)^{\#(subseq(eventos, 0, i+1), True)-1}(apuesta_s, pago_s)^{\#(subseq(eventos, 0, i+1), False)}) \vee (eventos[i] = false \wedge 0 \leq i+1 \leq |eventos| \wedge_L res = recurso(apuesta_c, pago_c)^{\#(subseq(eventos, 0, i+1), True)}(apuesta_s, pago_s)^{\#(subseq(eventos, 0, i+1), False)-1})$$

Utilizando la propiedad  $P \longrightarrow (Q \vee R) \leftrightarrow (P \longrightarrow Q) \vee (P \longrightarrow R)$  vemos los casos de la implicacion original por separado  
Caso  $eventos[i]=true$ :

$$(0 \leq i \leq |eventos| \wedge_L res = recurso(apuesta_c, pago_c)^{\#(subseq(eventos, 0, i), True)}(apuesta_s, pago_s)^{\#(subseq(eventos, 0, i), False)} \wedge i < |eventos|) \longrightarrow (0 \leq i < |eventos| \wedge_L (eventos[i] = true \wedge 0 \leq i+1 \leq |eventos| \wedge_L res = recurso(apuesta_c, pago_c)^{\#(subseq(eventos, 0, i+1), True)-1}(apuesta_s, pago_s)^{\#(subseq(eventos, 0, i+1), False)}))$$

$$\equiv (0 \leq i < |eventos| \wedge_L res = recurso(apuesta_c, pago_c)^{\#(subseq(eventos, 0, i), True)}(apuesta_s, pago_s)^{\#(subseq(eventos, 0, i), False)} \longrightarrow 0 \leq i < |eventos| \wedge_L (eventos[i] = true \wedge 0 \leq i+1 \leq |eventos| \wedge_L res = recurso(apuesta_c, pago_c)^{\#(subseq(eventos, 0, i+1), True)-1}(apuesta_s, pago_s)^{\#(subseq(eventos, 0, i+1), False)}))$$

$$\begin{aligned} &\equiv (0 \leq i < |\text{eventos}| \wedge_L \text{res} = \text{recurso}(\text{apuesta}_c \text{pago}_c)^{\#(\text{subseq}(\text{eventos}, 0, i), \text{True})}(\text{apuesta}_s \text{pago}_s)^{\#(\text{subseq}(\text{eventos}, 0, i), \text{False})} \longrightarrow \\ &(\text{eventos}[i] = \text{true} \wedge 0 \leq i + 1 \leq |\text{eventos}| \wedge_L \\ &\text{res} = \text{recurso}(\text{apuesta}_c \text{pago}_c)^{\#(\text{subseq}(\text{eventos}, 0, i+1), \text{True})-1}(\text{apuesta}_s \text{pago}_s)^{\#(\text{subseq}(\text{eventos}, 0, i+1), \text{False})}) \end{aligned}$$

Asumo verdadero el antecedente y reemplazo res

$$\begin{aligned} &\equiv 0 \leq i < |\text{eventos}| \longrightarrow (\text{eventos}[i] = \text{true} \wedge 0 \leq i + 1 \leq |\text{eventos}| \wedge_L \\ &\text{recurso}(\text{apuesta}_c \text{pago}_c)^{\#(\text{subseq}(\text{eventos}, 0, i), \text{True})}(\text{apuesta}_s \text{pago}_s)^{\#(\text{subseq}(\text{eventos}, 0, i), \text{False})} = \\ &\text{recurso}(\text{apuesta}_c \text{pago}_c)^{\#(\text{subseq}(\text{eventos}, 0, i+1), \text{True})-1}(\text{apuesta}_s \text{pago}_s)^{\#(\text{subseq}(\text{eventos}, 0, i+1), \text{False})}) \end{aligned}$$

Este predicado significa que si el elemento actual es true, contar las apariciones de true en eventos hasta el elemento anterior va a dar uno menos, lo cual es verdadero

Caso eventos[i]=false:

$$\begin{aligned} &(0 \leq i \leq |\text{eventos}| \wedge_L \text{res} = \text{recurso}(\text{apuesta}_c \text{pago}_c)^{\#(\text{subseq}(\text{eventos}, 0, i), \text{True})}(\text{apuesta}_s \text{pago}_s)^{\#(\text{subseq}(\text{eventos}, 0, i), \text{False})} \wedge \\ &i < |\text{eventos}|) \longrightarrow (0 \leq i < |\text{eventos}| \wedge_L (\text{eventos}[i] = \text{false} \wedge 0 \leq i + 1 \leq |\text{eventos}| \wedge_L \\ &\text{res} = \text{recurso}(\text{apuesta}_c \text{pago}_c)^{\#(\text{subseq}(\text{eventos}, 0, i+1), \text{True})}(\text{apuesta}_s \text{pago}_s)^{\#(\text{subseq}(\text{eventos}, 0, i+1), \text{False})-1}) \end{aligned}$$

Haciendo lo mismo que en el caso anterior:

$$\begin{aligned} &\equiv 0 \leq i < |\text{eventos}| \longrightarrow (\text{eventos}[i] = \text{false} \wedge 0 \leq i + 1 \leq |\text{eventos}| \wedge_L \\ &\text{recurso}(\text{apuesta}_c \text{pago}_c)^{\#(\text{subseq}(\text{eventos}, 0, i), \text{True})}(\text{apuesta}_s \text{pago}_s)^{\#(\text{subseq}(\text{eventos}, 0, i), \text{False})} = \\ &\text{recurso}(\text{apuesta}_c \text{pago}_c)^{\#(\text{subseq}(\text{eventos}, 0, i+1), \text{True})}(\text{apuesta}_s \text{pago}_s)^{\#(\text{subseq}(\text{eventos}, 0, i+1), \text{False})-1}) \end{aligned}$$

Este predicado significa que si el elemento actual es false, contar las apariciones de false en eventos hasta el elemento anterior va a dar uno menos, lo cual es verdadero

Como ambos casos de la implicacion son verdaderos, la implicacion es verdadera.

3)  $I \wedge \neg B \longrightarrow Q_c$

$$\begin{aligned} &0 \leq i \leq |\text{eventos}| \wedge_L \text{res} = \text{recurso}(\text{apuesta}_c \text{pago}_c)^{\#(\text{subseq}(\text{eventos}, 0, i), \text{True})}(\text{apuesta}_s \text{pago}_s)^{\#(\text{subseq}(\text{eventos}, 0, i), \text{False})} \wedge \\ &i \geq |\text{eventos}| \longrightarrow \text{res} = \text{recurso}(\text{apuesta}_c \text{pago}_c)^{\#(\text{eventos}, \text{True})}(\text{apuesta}_s \text{pago}_s)^{\#(\text{eventos}, \text{False})} \end{aligned}$$

$$\begin{aligned} &\equiv i = |\text{eventos}| \wedge_L \text{res} = \text{recurso}(\text{apuesta}_c \text{pago}_c)^{\#(\text{subseq}(\text{eventos}, 0, i), \text{True})}(\text{apuesta}_s \text{pago}_s)^{\#(\text{subseq}(\text{eventos}, 0, i), \text{False})} \longrightarrow \\ &\text{res} = \text{recurso}(\text{apuesta}_c \text{pago}_c)^{\#(\text{eventos}, \text{True})}(\text{apuesta}_s \text{pago}_s)^{\#(\text{eventos}, \text{False})} \end{aligned}$$

Asumo verdadero el antecedente y reemplazo i por  $|\text{eventos}|$

$$\begin{aligned} &\equiv \text{res} = \text{recurso}(\text{apuesta}_c \text{pago}_c)^{\#(\text{subseq}(\text{eventos}, 0, |\text{eventos}|), \text{True})}(\text{apuesta}_s \text{pago}_s)^{\#(\text{subseq}(\text{eventos}, 0, |\text{eventos}|), \text{False})} \longrightarrow \\ &\text{res} = \text{recurso}(\text{apuesta}_c \text{pago}_c)^{\#(\text{eventos}, \text{True})}(\text{apuesta}_s \text{pago}_s)^{\#(\text{eventos}, \text{False})} \end{aligned}$$

Como  $\text{subseq}(\text{eventos}, 0, |\text{eventos}|) = \text{eventos}$ :

$$\begin{aligned} &\equiv \text{res} = \text{recurso}(\text{apuesta}_c \text{pago}_c)^{\#(\text{eventos}, \text{True})}(\text{apuesta}_s \text{pago}_s)^{\#(\text{eventos}, \text{False})} \longrightarrow \\ &\text{res} = \text{recurso}(\text{apuesta}_c \text{pago}_c)^{\#(\text{eventos}, \text{True})}(\text{apuesta}_s \text{pago}_s)^{\#(\text{eventos}, \text{False})} \end{aligned}$$

Ambos lados de la implicacion son iguales, entoces la implicacion es verdadera

4)  $\{I \wedge B \wedge v_0 = Fv\} \text{while} \dots \text{endwhile} \{Fv < v_0\}$ :

Para demostrar esto hay que demostrar:  $(I \wedge B \wedge v_0 = Fv) \longrightarrow wp(\text{while} \dots \text{endwhile}, Fv < v_0)$

$$\begin{aligned} &wp(\text{while} \dots \text{endwhile}, |\text{eventos}| - i < v_0) \equiv wp(\text{if} \dots \text{endif}, wp(i := i + 1, |\text{eventos}| - i < v_0)) \\ &\equiv wp(\text{if} \dots \text{endif}, |\text{eventos}| - (i + 1) < v_0) \end{aligned}$$

$$\begin{aligned} &\equiv \text{def}(\text{eventos}[i]) \wedge_L ((\text{eventos}[i] = \text{true} \wedge wp(\text{res} := \text{res} * \text{apuesta}_c * \text{pago}_c, |\text{eventos}| - (i + 1) < v_0)) \vee \\ &(\text{eventos}[i] = \text{false} \wedge wp(\text{res} := \text{res} * \text{apuesta}_s * \text{pago}_s, |\text{eventos}| - (i + 1) < v_0))) \\ &\equiv 0 \leq i < |\text{eventos}| \wedge_L (\text{eventos}[i] = \text{true} \wedge |\text{eventos}| - (i + 1) < v_0) \vee (\text{eventos}[i] = \text{false} \wedge |\text{eventos}| - (i + 1) < v_0) \\ &\equiv 0 \leq i < |\text{eventos}| \wedge_L (\text{eventos}[i] = \text{true} \vee \text{eventos}[i] = \text{false}) \wedge |\text{eventos}| - (i + 1) < v_0 \equiv |\text{eventos}| - (i + 1) < v_0 \end{aligned}$$

Volviendo a la equivalencia original:

$$\begin{aligned} &(I \wedge B \wedge v_0 = Fv) \longrightarrow wp(\text{while} \dots \text{endwhile}, Fv < v_0) \equiv (I \wedge B \wedge v_0 = |\text{eventos}| - i) \longrightarrow \\ &0 \leq i < |\text{eventos}| \wedge_L |\text{eventos}| - (i + 1) < v_0 \end{aligned}$$

Asumo verdadero el antecedente y reemplazo  $v_0$  por  $|eventos| - i$   
 $\equiv (I \wedge B) \longrightarrow 0 \leq i < |eventos| \wedge_L |eventos| - (i + 1) < |eventos| - i \equiv (I \wedge B) \longrightarrow 0 \leq i < |eventos| \wedge_L |eventos| - i - 1 < |eventos| - i$

Como el consecuente es siempre verdadero, la implicacion es verdadera

5)  $I \wedge Fv \leq 0 \longrightarrow -B$

$I \wedge |eventos| - i \leq 0 \longrightarrow i \geq |eventos| \equiv I \wedge |eventos| \leq i \longrightarrow i \geq |eventos|$

La implicacion es siempre verdadera

Con esto queda demostrado, por Teorema del Invariante y Teorema de Terminacion de Ciclo, que vale la siguiente tripla de Hoare:

$\{i = 0, res = recurso\} while....endwhile \{res = recurso (apuesta_c pago_c)^{\#(eventos, True)} (apuesta_s pago_s)^{\#(eventos, False)}\}$

Solo queda demostrar que  $P_c$  cumple:

$(apuesta_c + apuesta_s = 1 \wedge pago_c > 0 \wedge pago_s > 0 \wedge apuesta_c > 0 \wedge apuesta_s > 0 \wedge recurso > 0) \longrightarrow wp(res := recurso; i := 0, P_c)$

$\equiv (apuesta_c + apuesta_s = 1 \wedge pago_c > 0 \wedge pago_s > 0 \wedge apuesta_c > 0 \wedge apuesta_s > 0 \wedge recurso > 0) \longrightarrow wp(res := recurso; wp(i := 0, res = recurso \wedge i = 0 \wedge apuesta_c + apuesta_s = 1 \wedge pago_c > 0 \wedge pago_s > 0 \wedge apuesta_c > 0 \wedge apuesta_s > 0 \wedge recurso > 0))$

$\equiv (apuesta_c + apuesta_s = 1 \wedge pago_c > 0 \wedge pago_s > 0 \wedge apuesta_c > 0 \wedge apuesta_s > 0 \wedge recurso > 0) \longrightarrow wp(res := recurso; res = recurso \wedge 0 = 0 \wedge apuesta_c + apuesta_s = 1 \wedge pago_c > 0 \wedge pago_s > 0 \wedge apuesta_c > 0 \wedge apuesta_s > 0 \wedge recurso > 0)$

$\equiv (apuesta_c + apuesta_s = 1 \wedge pago_c > 0 \wedge pago_s > 0 \wedge apuesta_c > 0 \wedge apuesta_s > 0 \wedge recurso > 0) \longrightarrow recurso = recurso \wedge 0 = 0 \wedge apuesta_c + apuesta_s = 1 \wedge pago_c > 0 \wedge pago_s > 0 \wedge apuesta_c > 0 \wedge apuesta_s > 0 \wedge recurso > 0$

Como el consecuente es siempre verdadero, la implicacion es verdadera

Al demostrar esto queda demostrado que la especificacion es correcta respecto de la implementacion ya que la postcondicion del ciclo es equivalente a la postcondicion