DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Practico Nº1: Especificación y WP

20 de mayo de 2024 — Algoritcmos y Estructuras de Datos II

Grupo (AguanteTaaàllere)

Integrante	LU	Correo electrónico
Olmos, Francisco José	1101/23	francisco.olmos.99@gmail.com
Andina Silva, Augusto	1344/23	augustoandinasilva@gmail.com
López Porto, Gregorio	1376/23	gregoriolopezporto@gmail.com
Quintana , Joaquín Ezequiel	1356/23	joaquin32flores@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. Consigna

1.1. redistribucionDeLosFrutos

Calcula los recursos que obtiene cada uno de los individuos luego de que se redistribuyen los recursos del fondo monetario común en partes iguales. El fondo monetario común se compone de la suma de recursos iniciales aportados por todas las personas que cooperan. La salida es la lista de recursos que tendrá cada jugador.

`proc redistribucionDeLosFrutos` (in recursos: $seq\langle\mathbb{R}\rangle$, in cooperan : $seq\langle\text{Bool}\rangle$) : $seq\langle\mathbb{R}\rangle$

`requiere` $\{listaNoVacia(recursos) \ \wedge_L \ igualLargo(recursos, cooperan)\}$

`asegura` $\{igualLargo(res, recursos) \wedge (\forall i : \mathbb{Z})(0 \leq i < |cooperan|) \rightarrow_L$
$((\text{cooperan[i]} \wedge res[i] = \frac{TotalDeRecursos(recursos,cooperan)}{|cooperan|}) \vee$
$(\neg cooperan[i] \wedge res[i] = recursos[i] + \frac{TotalDeRecursos(recursos,cooperan)}{|cooperan|}))\}$

`aux TotalDeRecursos(`$cooperan : seq\langle\text{Bool}\rangle, recusos : seq\langle\mathbb{R}\rangle$`)` : $\mathbb{R} = \sum_{i=0}^{|recursos|-1}$ if $cooperan[i]$ then $recursos[i]$ else 0 fi ;

1.2. trayectoriaDeLosFrutosIndividualesALargoPlazo

Actualiza (In/Out) la lista de trayectorias de los los recursos de cada uno de los individuos. Inicialmente, cada una de las trayectorias (listas de recursos) contiene un único elemento que representa los recursos iniciales del individuo. El procedimiento agrega a las trayectorias los recursos que los individuos van obteniendo a medida que se van produciendo los resultados de los eventos en función de la lista de pagos que le ofrece la naturaleza (o casa de apuestas) a cada uno de los individuos, las apuestas (o inversiones) que realizan los individuos en cada paso temporal, y la lista de individuos que cooperan aportando al fondo monetario común.

`proc trayectoriaDeLosFrutosIndividualesALargoPlazo` (inout trayectorias: $seq\langle seq\langle\mathbb{R}\rangle\rangle$, in cooperan: $seq\langle\text{Bool}\rangle$, in apuestas: $seq\langle seq\langle\mathbb{R}\rangle\rangle$, in pagos: $seq\langle seq\langle\mathbb{R}\rangle\rangle$, in eventos: $seq\langle\mathbb{N}\rangle$) :

`requiere` $\{|trayectoria| = |cooperan| = |apuestas| = |pagos| = |eventos|\}$

`requiere` $\{pagosValidos(pagos)\}$

`requiere` $\{apuestasValidas(apuestas)\}$

`requiere` $\{|apuestas[0]| == |pagos[0]| \wedge concibeTodosLosEventos(eventos, apuestas)$

Este requiere pide que todos los parametros de entrada que tenga el codigo cumplan las condiciones minimas y necesarias con la que los datos fueron definidos, de manera que pide que todos los datos recibidos sean del tipo listas o matrices, y estas tengan la misma cantidad de individuos respectivamente. Que los pagos correspondan a valores positivos, igual que con las apuestas a la vez que verifica que todas estas esten repartidas de forma çorrecta"siendo que representan un porcentaje del total de los recursos actuales, el total de apuestas sumadas deberia ser igual a 1. Por ultimo pide que para cada evento en la matriz apuestas haya un evento correspondido en la matriz pagos y que cada valor de la matriz eventos represente un elemento que exista en la matriz apuestas(unicamente con esta matriz ya que con todo lo que cumplen las matrices entre si que cumpla ese predicado con una sola implicaria que lo cumpla con todas}

La idea es que la nueva trayectoria tenga el largo de los eventos que ocurrieron (el total de tiempos) + 1 que seria el tiempo 0 (los recursos iniciales)

`asegura` $\{trayectoriasValidas(trayectorias) \wedge |trayectorias| = |old(trayectorias)| \wedge |trayectorias[0]| = |eventos[0] + 1|\}$

`asegura` $\{(\forall i : Z)(0 \leq i < |trayectorias| \rightarrow_L trayectorias[i][0] = old(trayectorias[i][0]))\}$

`asegura` $\{(\forall i : \mathbb{Z})(0 \leq i < |trayectorias|) \rightarrow_L (\forall t : \mathbb{Z})(0 \leq t < |trayectorias[i]|) \rightarrow_L$
(trayectorias[i][t+1] = redistribucionDeFrutosParaTrayectoria(nuevosRecursos(trayectorias, pagos, apuestas, eventos, t), cooperan)[i])}

`aux RedistribucionDeFrutosParaTrayectoria` (recursos: $seq\langle\mathbb{R}\rangle$, cooperan: $seq\langle\text{Bool}\rangle$) : $seq\langle\mathbb{R}\rangle = |res| = |recursos|) \wedge$
$(\forall i : \mathbb{Z})(0 \leq i < |cooperan|) \rightarrow_L ((cooperan[i] \wedge res[i] = \frac{TotalDeRecursos(recursos,cooperan)}{|cooperan|}) \vee (\neg cooperan[i] \wedge res[i] = recursos[i] + \frac{TotalDeRecursos(recursos,cooperan)}{|cooperan|}))$;

`aux nuevosRecursos` (trayectoria: $seq\langle seq\langle\mathbb{R}\rangle\rangle$, pagos: $seq\langle seq\langle\mathbb{R}\rangle\rangle$, apuestas: $seq\langle seq\langle\mathbb{R}\rangle\rangle$, eventos: $seq\langle seq\langle\mathbb{N}\rangle\rangle$, t: $\mathbb{Z}$)
: $seq\langle\mathbb{R}\rangle = |res| = |trayectorias| \wedge_L (\forall i : \mathbb{Z})(0 \leq i < |trayectorias| \rightarrow_L res[i] = trayectoria[i][t] * apuestas[i][eventos[i][t]] * pagos[i][eventos[i][t]]$;

1.3. trayectoriaExtrañaEscalera

Esta función devuelve True si, y solo si, en la trayectoria de un individuo existe un único punto mayor a sus vecinos (llamado máximo local). Un elemento es máximo local si es mayor estricto que sus vecinos inmediatos.

`proc trayectoriaExtrañaEscalera` (in trayectoria : $seq\langle\mathbb{R}\rangle$) : Bool

`requiere` $\{listaNoVacia(trayectoria) \land todosPositivos(trayectoria)\}$
`requiere` $\{trayectoriaValida(trayectoria)\}$

`asegura` $\{(|trayectoria| = 1)$
$\veebar(trayectoria[0] > trayectoria[1])$
$\veebar(trayectoria[|trayectoria| - 1] > trayectoria[|trayectoria| - 2])$
$\veebar(\exists! i : \mathbb{Z})(1 \leq i < |trayectoria| - 1) \land_L (trayectoria[i] > trayectoria[i-1] \land trayectoria[i] > trayectoria[i+1])\}$

1.4. individuoDecideSiCooperaONo

Un individuo actualiza su comportamiento cooperativo / no-cooperativo (cooperan[individuo]) en función de los recursos iniciales, de quienes cooperan, de los pagos que se le ofrecen a cada individuo, de las inversiones o apuestas de cada individuo, y del resultado los eventos que recibe cada individuo, eligiendo el comportamiento que maximiza sus recursos individuales a largo plazo.

`proc individuoDecideSiCooperaONo` (inout cooperan : $seq\langle \mathsf{Bool} \rangle$, in individuo: $\mathbb{N}$, in recursos: $seq\langle \mathbb{R} \rangle$, in apuestas: $seq\langle seq\langle \mathbb{R} \rangle \rangle$, in pagos: $seq\langle seq\langle \mathbb{R} \rangle \rangle$, in eventos: $seq\langle seq\langle \mathbb{N} \rangle \rangle$)
`requiere` $\{|recursos| == |cooperan| == |apuestas| == |pagos| == |eventos| \land individuo < |cooperan|\}$
`requiere` $\{listaNoVacia(recursos) \land todosPositivos(recursos)\}$
`requiere` $\{pagosValidos(pagos)\}$
`requiere` $\{apuestasValidas(apuestas)\}$
`requiere` $\{eventosValidos(eventos)\}$
`requiere` $\{|apuestas[0]| == |pagos[0]| \land concibeTodosLosEventos(eventos, apuestas)\}$

`asegura` {if $calculoTrayectoria(recursos, individuo, SetAt(cooperan, individuo, True), apuestas, pagos, eventos)$
[individuo][| $eventos[0]$ |]
$\geq$
calculoTrayectoria(recursos, individuo, SetAt(cooperan, individuo, False), apuestas,pagos, eventos)
[individuo][| $eventos[0]$ |]
then cooperan = setAt(cooperan, individuo, True) else setAt(cooperan, individuo, False fi}

`aux calculoDeTrayectoria` (recursos: $seq\langle seq\langle \mathbb{R} \rangle \rangle$, individuo: $\mathbb{N}$, cooperan: $seq\langle \mathsf{Bool} \rangle$, apuestas: $seq\langle seq\langle \mathbb{R} \rangle \rangle$, pagos : $seq\langle seq\langle \mathbb{R} \rangle \rangle$, eventos: $seq\langle seq\langle \mathbb{N} \rangle \rangle$) : $seq\langle seq\langle \mathbb{R} \rangle \rangle = |res| = |recursos| \land (\forall i : \mathbb{Z})(0 \leq i < |recursos| \rightarrow_L$
res[i][0] = recursos[i] $\land |res[i][0]| = |eventos[0]| + 1) \land esMatriz(res)$
$\land$
$(\forall i : \mathbb{Z})(0 \leq i < |eventos| \rightarrow_L (\forall t : \mathbb{Z})(0 \leq t < |eventos[i]| \rightarrow_L$
res[i][t+1] = redistribucionDeFrutosParaTrayectoria(nuevoRecurso(res, eventos,apuestas, pagos, t),cooperan)[i] ;

Funciona como el asegura del punto 1.2 pero partiendo de una lista y llegando a una matriz con el auxiliar calculoDeTrayectoria y luego se evalúan los 2 posibles casos en el que un individuo coopera o no y se determina si la lista cooperan se le asigna un True o un False al individuo

1.5. individuoActualizaApuesta

Un individuo actualiza su apuesta (apuestas[individuo]) en función de los recursos iniciales, de la lista de individuos que cooperan, de los pagos que se le ofrecen a cada individuo, de las inversiones o apuestas de cada individuo y del resultado los eventos que recibe cada individuo, eligiendo la apuesta que maximiza sus recursos individuales a largo plazo

`proc individuoActualizaApuesta` (inout apuestas : $seq\langle seq\langle \mathbb{R} \rangle \rangle$, in individuo: $\mathbb{N}$, in recursos: $seq\langle \mathbb{R} \rangle$, in cooperan: $seq\langle \mathsf{Bool} \rangle$, in pagos: $seq\langle seq\langle \mathbb{R} \rangle \rangle$, in eventos: $seq\langle seq\langle \mathbb{N} \rangle \rangle$)
`requiere` $\{individuo < |recursos|\}$
`requiere` $\{|recursos| == |cooperan| == |apuestas| == |pagos| == |eventos|\}$
`requiere` $\{listaNoVacia(recursos)\}$
`requiere` $\{pagosValidos(pagos)\}$
`requiere` $\{apuestasValidas(apuestas)\}$
`requiere` $\{todosPositivos(recursos)\}$
`requiere` $\{concibeTodosLosEventos(eventos, apuestas)\}$
`requiere` $\{|apuestas[0]| == |pagos[0]|\}$

`asegura` $\{(\exists mejorApuesta : seq\langle \mathbb{R} \rangle)(listaApuestaValida(mejorApuesta, apuestas)) \land_L$
$(\forall apuestaPosible : seq\langle \mathbb{R} \rangle)(listaApuestaValida(apuestaPosible, apuestas) \rightarrow_L$
calculoDeTrayectoria(recursos, individuo, cooperan, SetAt(apuestas, individuo,mejorApuesta), pagos, eventos)
[individuo][| $eventos[0]$ |]
$\geq$
calculoDeTrayectoria(recursos, individuo, cooperan, SetAt(apuestas, individuo, apuestaPosible), pagos, eventos)
[individuo][| $eventos[0]$ |]}

Es la misma lógica que el ejercicio anterior (1.4) pero en vez de tomar 2 posibles valores toma todos los posibles valores y toma uno que sea el mas adecuado para lo pedido

2. Demostracion de correctitud

Demostrar que la siguiente especificacion es correcta respecto de su implementacion. La funcion frutoDelTrabajoPuramenteIndividual calcula, para el ejemplo de apuestas al juego de cara o sello, cuanto se ganaria si se juega completamente solo. se contempla el evento True es cuando sale cara.

`proc frutoDelTrabajoPuramenteIndividual` (in recurso: $\mathbb{R}$, in apuesta: $seq\langle s : \mathbb{R}, c : \mathbb{R}\rangle$, in pago: $seq\langle s : \mathbb{R}, c : \mathbb{R}\rangle$, in eventos $seq\langle \mathsf{Bool}\rangle$, out res: $\mathbb{R}$)

`requiere` $\{apuesta_c + apuesta_s = 1 \land pago_c > 0 \land pago_s > 0 \land apuesta_c > 0 \land apuesta_s > 0 \land recurso > 0\}$
`asegura` $\{res = recurso(apuesta_c pago_c)^{\#apariciones(eventos,T)}(apuesta_s pago_s)^{\#apariciones(eventos,F)}\}$

Donde # aparciones(eventos,T) es el auxiliar en la teorica, y # (eventos, T) su abreviacion

```
1  res := 0;
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
```

Demostracion de programas con ciclos

Que tenemos que hacer para probar que {Pre} S1; while...;S3 {Post} es valida?
1. Pre $\rightarrow_L wp(S1; P_C)$
2. $P_C \rightarrow_L wp(while...; Q_C)$
3. $Q_C \rightarrow_L wp(S3; Post)$

Axiomas de wp

$Axioma1 : wp(x := E, Q) \equiv def(E) \land_L Q^x_E$
$Axioma2 : wp(skip, Q) \equiv Q$
$Axioma3 : wp(S1; S2, Q) \equiv wp(S1, wp(S2, Q))$
$Axioma4 : wp(\text{if } B \text{ then } S1 \text{ else } S2 \text{ fi}, Q) \equiv def(B) \land_L ((B \land wp(S1, Q)) \lor (\neg B \land wp(S2, Q))$
$Axioma5 : wp(\text{while } B \text{ do } S \text{ endwhile}, Q) \equiv (\exists_{i \geq 0})(H_i(Q))$

Teorema del Invariante

Si def(B) y existe un predicado I tal que

1.$P_C \rightarrow I$
2.$\{I \land B\}\ S\ \{I\}$
3.$I \land \neg B \rightarrow Q_C$

.. y el ciclo termina, entonces la siguiente tripla de Hoare es valida:

$$\{P_C\}\ \textbf{while B do S endwhile}\ \{Q_C\}$$

Teorema de terminacion de un ciclo

Sea V el producto cartesiano de los dominios de las variables del programa y sea I un invariante del ciclo **while B do S endwhile** . Si existe una funcion fv : V $\rightarrow \mathcal{Z} talque$

1.$\{I \land B \land v_0 = f_V\}\ S\ \{f_V < v_0\}$
2.$I \land f_V \leq 0 \rightarrow \neg B$

Entonces, por (1)-(5) , se cumplen las hipotesis de ambos teoremas (teorema del invariante + teorema de terminacion). Por lo tanto, la tripla de Hoare es valida (i.e., dada P_C , el ciclo siempre termina y vale Q_C)

Proponemos un invariante

I $\equiv \{0 \leq i \leq |eventos| \wedge$

res = recurso * $\prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c * pago.c) * \prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.s * pago.s)\}$

$P_C \equiv \{i = 0 \wedge res = recursos\}$

1) $P_C \rightarrow I$

{i=0 $\wedge res = recursos\} \rightarrow \{0 \leq i \leq |eventos| \wedge$

res = recurso * $\prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c * pago.c) * \prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.s * pago.s)\}$

asumimos como hipotesis que P_C es verdadera entonces tomamos $i = 0 \wedge res = recursos\}$ y tratamos de llegar al I

{i=0 $\wedge res = recursos\} \rightarrow \{0 \leq 0 \leq |eventos| \wedge$

res = recurso * $\prod_{j=1}^{\#(subseq(eventos,0,0),T)}(apuesta.c * pago.c) * \prod_{j=1}^{\#(subseq(eventos,0,0),F)}(apuesta.s * pago.s)\}$

(subseq(eventos, 0, 0), T) $\equiv \#(<>, T) = 0$

(subseq(eventos, 0, 0), F) $\equiv \#(<>, F) = 0$

res = recurso * $\prod_{j=1}^{0}(apuesta.c * pago.c) * \prod_{j=1}^{0}(apuesta.s * pago.s)\} = recurso * 1 * 1 = recursos \rightarrow res = recursos$

dado que $from \leq to$ $(1 \leq 0)$ no se cumple, la productoria se indefine, lo que es igual a 1

---

2) $\{I \wedge B_C\}\ S\ \{I\}$

Declaramos los predicados siguientes para poder referenciarlos a lo largo de la demostracion:

B_C es la guarda del ciclo y B' es la guarda del if, para poder distinguirlos

$B_C \equiv \{i < |eventos|\}$

$S \equiv \{\text{if } B' \text{ then } S1;\ \text{else } S2;\ \text{fi}\}$

$B' \equiv \{eventos[i]\}$

$S_1 \equiv \{res := res * apuesta.c * pagos.c\}$

$S_2 \equiv \{res := res * apuesta.s * pagos.s\}$

$S \equiv \{\text{if } eventos[i] \text{ then } res := res * apuesa.c * pago.c;\ \text{else } res := res * apuesa.s * pago.s;\ \text{fi}\}$

$\{I \wedge B_C\} \rightarrow wp(S, I)$

$\{I \wedge B_C\}$
$\equiv \{i < |eventos| \wedge 0 \leq i \leq |eventos| \wedge$

res = recurso * $\prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c * pago.c) * \prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.s * pago.s)\}$
$\equiv \{0 \leq i < |eventos| \wedge$

res = recurso * $\prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c * pago.c) * \prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.s * pago.s)\}$

$wp(S, I) \equiv def(B') \wedge_L ((B' \wedge wp(S_1, I)) \vee (\neg B' \wedge wp(S_2, I)))$

def(B') $\equiv def(eventos[i]) \equiv 0 \leq i < |eventos|$

$wp(S_1, I) \equiv def(res * apuesta.c * pagos.c) \wedge_L I^{res}_{res*apuesta.c*pagos.c}$

def(res * apuesta.c * pagos.c) $\equiv True$

$wp(S_2, I) \equiv def(res * apuesta.s * pagos.s) \wedge_L I^{res}_{res*apuesta.s*pagos.s}$

$I^{res}_{res*apuesta.c*pagos.c}$
$\equiv \{res*apuesta.c*pagos.c = recurso*\prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c*pago.c)*\prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.$
$pago.s)\}$
$\equiv \{res = recurso*\prod_{j=1}^{\#(subseq(eventos,0,i-1),T)}(apuesta.c*pago.c)*\prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.s*pago.s)\}$

def(res * apuesta.s * pagos.s) $\equiv True$

$I^{res}_{res*apuesta.s*pagos.s}$
$\equiv \{res*apuesta.s*pagos.s = recurso*\prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c*pago.c)*\prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta$
$pago.s)\}$
$\equiv \{res = recurso*\prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c*pago.c)*\prod_{j=1}^{\#(subseq(eventos,0,i-1),F)}(apuesta.s*pago.s)\}$

$0 \leq i < |eventos| \wedge_L$

{ eventos[i] $\wedge res = recurso*\prod_{j=1}^{\#(subseq(eventos,0,i-1),T)}(apuesta.c*pago.c)*\prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.s*$
$pago.s)\}$

{ $\neg eventos[i] \wedge res = recurso*\prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c*pago.c)*\prod_{j=1}^{\#(subseq(eventos,0,i-1),F)}(apuesta.s*$
$pago.s)\}$

{ $0 \leq i < |eventos| \wedge_L$
res = recurso * $\prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c * pago.c) * \prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.s * pago.s)\}$
$\rightarrow$
{eventos[i] $\wedge res = recurso*\prod_{j=1}^{\#(subseq(eventos,0,i-1),T)}(apuesta.c*pago.c)*\prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.s*$
$pago.s)\}$
$\vee$
{$\neg eventos[i] \wedge res = recurso*\prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c*pago.c)*\prod_{j=1}^{\#(subseq(eventos,0,i-1),F)}(apuesta.s*$
$pago.s)\}$
{eventos[i] $\wedge res = recurso*\prod_{j=1}^{\#(subseq(eventos,0,i-1),T)}(apuesta.c*pago.c)*\prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.s*$
$pago.s)\}$
$\equiv \{res = recurso * \prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c * pago.c) * \prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.s * pago.s)\}$
{$\neg eventos[i] \wedge res = recurso*\prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c*pago.c)*\prod_{j=1}^{\#(subseq(eventos,0,i-1),F)}(apuesta.s*$
$pago.s)\}$
$\equiv \{res = recurso * \prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c * pago.c) * \prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.s * pago.s)\}$
{ { res = recurso * $\prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c*pago.c)*\prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.s*pago.s)\}\vee$
$\{res = recurso * \prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c * pago.c) * \prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.s * pago.s)\}\}$
$\equiv$
{ res = recurso * $\prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c * pago.c) * \prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.s * pago.s)\}$

---

3) $I \wedge \neg B_C \rightarrow Q_C$

$\neg B_C \equiv \{i \geq |eventos|\}$

$Q_C \equiv \{res = recurso\ (apuesta_c pago_c)^{\#apariciones(eventos,T)}\ (apuesta_s pago_s)^{\#apariciones(eventos,F)}\}$

$I \wedge \neg B_C \equiv$

{ $0 \leq i \leq |eventos| \wedge i \geq |eventos| \wedge$

res = recurso * $\prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c * pago.c) * \prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.s * pago.s)\}$

$\{0 \leq i \leq |eventos| \land i \geq |eventos|\} \equiv \{i = |eventos|\}$

$\{i = |eventos| \land res = recurso * \prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c * pago.c) * \prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.s * pago.s)\} \equiv \{res = recurso * \prod_{j=1}^{\#(subseq(eventos,0,|eventos|),T)}(apuesta.c * pago.c) * \prod_{j=1}^{\#(subseq(eventos,0,|eventos|),F)}(apuesta.s * pago.s)\}$

$subseq(eventos, 0, |eventos|) =$ eventos

$\{$ res = recurso $* \prod_{j=1}^{\#(subseq(eventos,0,|eventos|),T)}(apuesta.c * pago.c) * \prod_{j=1}^{\#(subseq(eventos,0,|eventos|),F)}(apuesta.s * pago.s)\} \equiv Q_C$

Teorema de terminacion de un ciclo

1.$\{I \land B \land v_0 = f_V\}\ S\ \{f_V < v_0\}$
2.$I \land f_V \leq 0 \rightarrow \neg B$
Proponemos una funcion variante
$f_V = |eventos| - i$

1) queremos demostrar que $\{I \land B \land v_0 = f_V\}\ S\ \{f_V < v_0\}$ nos alcanza con probar:

$$\{I \land B \land v_0 = f_V\} \rightarrow wp(S, \{f_V < v_0\})$$

$\{I \land B \land v_0 = f_V\} \equiv$
$\{\ 0 \leq i \leq |eventos|$
$\land res = recurso * \prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c * pago.c) * \prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.s * pago.s)\}$
$\land i < |eventos|$
$\land f_V = |eventos|\}$
$\equiv \{0 \leq i < |eventos| \land f_V = |eventos| \land I\}$

2) $I \land f_V \leq 0 \rightarrow \neg B$

$\neg B_C \equiv \{i \geq |eventos|\}$

$\{\ 0 \leq i \leq |eventos|$
$\land res = recurso * \prod_{j=1}^{\#(subseq(eventos,0,i),T)}(apuesta.c * pago.c) * \prod_{j=1}^{\#(subseq(eventos,0,i),F)}(apuesta.s * pago.s)\ \land |eventos| - i \leq 0\}$

$\{0 \leq i \leq |eventos| \land |eventos| - i \leq 0\} \equiv \{0 \leq i \leq |eventos| \land |eventos| \leq i\} \equiv \{|eventos| = i\}$

$\equiv \{I \land i = |eventos|\} \rightarrow \{i \geq |eventos|\}$

Por ultimo, para demostrar que la tripla {Pre} S1; while...;S3 {Post} es valida desglosamos en casos:

Para el caso $requiere \rightarrow P_C$ como P_C no depende de las variables del requiere, este siempre es verdadero por lo que nos queda $requiere \rightarrow True$ lo que siempre es verdadero por lo que se demuestra que requiere implica Pc.

El segundo caso donde $P_C \rightarrow wp(while.., Q_C)$ quedo probado en la demostracion del teorema del invariante.

Y para el caso de que $Q_C \rightarrow asegura$ planetamos que Q_C es exactamente igual que el predicado del asegura

Por eso determinamos que la tripla de Horae {Pre} S1; while...;S3 {Post} dado el programa es una tripla valida.

3. Anexo

Predicados y funciones auxiliares que utilizaremos a lo largo del Trabajo practico.

`pred listaNoVacia` (in s: $seq\langle T\rangle$) {
$|s| > 0$
}

`pred todosPositivos` (in s: $seq\langle T\rangle$) {
$(\forall i : Z)(0 \leq i < |s|) \rightarrow_L (s[i] > 0)$
}

`pred mismoLargo` (in s: $seq\langle T\rangle$, in l: $seq\langle T\rangle$) {
$|s| = |l|$
}

`pred esMatriz` (in s:$seq\langle seq\langle T\rangle\rangle$) {
$(\forall i : \mathbb{Z})(0 \leq i < |m| \rightarrow_L |m[i]| > 0 \wedge (\forall j : \mathbb{Z})(0 \leq j < |m| \rightarrow_L |m[i]| = |m[j]|))$
}

`pred apuestaValida` (in s:$seq\langle seq\langle \mathbb{R}\rangle\rangle$) {
$esMatriz(s) \wedge (\forall i : \mathbb{Z})(0 \leq i < |s|) \rightarrow_L todosPositivos(s[i]) \wedge (\forall j : \mathbb{Z})(0 \leq j < |s[i]|) \rightarrow_L (\sum s[i][j] = 1)$
}

`pred listaApuestaValida` (in listaApuesta: $seq\langle \mathbb{R}\rangle$, in apuestas: $seq\langle seq\langle \mathbb{R}\rangle\rangle$) {
$|listaApuesta| == |apuestas[0]| \ \wedge_L \ (\sum_{i=0}^{|listaApuesta|-1} listaApuesta[i] = 1) \ \wedge$
todosPositivosEstrictos(listaApuesta)
}

`pred pagosValidos` (in s:$seq\langle seq\langle \mathbb{R}\rangle\rangle$) {
$esMatriz(s) \wedge (\forall i : \mathbb{Z})(0 \leq i < |s| \rightarrow_L todosPositivos(s[i]))$
}

`pred trayectoriasValidas` (in t: $seq\langle seq\langle \mathbb{R}\rangle\rangle$) {
$esMatriz(t) \wedge (\forall i : \mathbb{Z})(0 \leq i < |t| \rightarrow_L (|t[i]| == 1 \wedge_L t[i][0] > 0))$
}

`pred concibeTodosLosEventos` (in eventos: $seq\langle seq\langle \mathbb{N}\rangle\rangle$, in matriz: $seq\langle seq\langle T\rangle\rangle$) {
$esMatriz(eventos) \wedge (\forall i : \mathbb{Z})(0 \leq i < |eventos| \rightarrow_L (\forall j : \mathbb{Z})(0 \leq j < |eventos[0]| \rightarrow_L eventos[i][j] < |matriz[0]|))$
}

`pred esTrayectoriasIniciales` ((trayectoriasIniciales: $seq\langle seq\langle \mathbb{R}\rangle\rangle$, recursos:$seq\langle \mathbb{R}\rangle$)) {
|trayectoriasIniciales| = $|recursos| \ \wedge (\forall i : \mathbb{Z})$ ($(0 \leq i <$ |trayectoriasIniciales| $\rightarrow_L$
(|trayectoriasIniciales[i]| = 1 $\wedge$ trayectoriasIniciales[i][0] = recursos[i]))
}
`pred sonIndCooperaYNoCoopera` ((cooperan: $seq\langle \mathsf{Bool}\rangle$, indCoopera: $seq\langle \mathsf{Bool}\rangle$, indNoCoopera: $seq\langle \mathsf{Bool}\rangle$, individuo: N)) {
|indCoopera| = |indNoCoopera| = |cooperan| $\wedge$
$(\forall i : \mathbb{Z})((0 \leq i <$ |cooperan| $\wedge\ i \neq$ individuo) $\rightarrow_L$
indCoopera[i] = indNoCoopera[i] = cooperan[i]) $\wedge$
indCoopera[individuo] = true $\wedge$ indNoCoopera[individuo] = false
}

`aux TotalDeRecursos` (cooperan: $seq\langle \mathsf{Bool}\rangle$, recursos: $seq\langle \mathbb{R}\rangle$) : $\mathbb{R}$ =
$\sum_{i=0}^{|recursos|-1}$ (if $cooperan[i]$ then $recursos[i]$ else 0 fi) ;

`aux ultimosRecursos` (trayectoria: $seq\langle seq\langle \mathbb{R}\rangle\rangle$) : $seq\langle \mathbb{R}\rangle = |res| = |trayectoria| \wedge (\forall i : \mathbb{Z})(0 \leq i < |trayectoria| \rightarrow res[i] = trayectoria[i][|trayectoria| - 1])$;

`aux redistribucionDeRecursos` (trayectorias: $seq\langle seq\langle \mathbb{R}\rangle\rangle$, cooperan: $seq\langle \mathsf{Bool}\rangle$) : $seq\langle \mathbb{R}\rangle =$
mismoLargo(res, ultimosRecursos(trayectorias)) $\wedge$
$(\forall i : \mathbb{Z})(0 \leq i < |cooperan|) \rightarrow_L ((cooperan[i] \wedge res[i] = \frac{totalDeRecursos(cooperan, ultimosRecursos(trayectorias))}{|cooperan|}) \vee$
$(\neg cooperan[i] \wedge res[i] = ultimosRecursos(trayectorias)[i] + \frac{totalDeRecursos(cooperan, ultimosRecursos(trayectorias))}{|cooperan|}))$;