DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 1

Especificación y WP

21 de abril de 2024 — Algoritmos y Estructuras de Datos I

Grupo "gliptodonte24"

Integrante	LU	Correo electrónico
Maydana, Daniel	205/22	`danimaydana9@gmail.com`
Lozada, Jack	1142/22	`nothingbutjack2200@gmail.com`
Cian, Andrés Bautista	937/21	`andycia802@gmail.com`
Perez Lanzillotta, Santiago	586/16	`santi.perezl@hotmail.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

Índice

1. Especificación **2**

1.1. redistribucionDeLosFrutos . . . 2

1.1.1. Decisiones tomadas para resolverlo: . . . 2

1.1.2. Procedimiento: . . . 2

1.2. trayectoriaDeLosFrutosIndividualesALargoPlazo . . . 3

1.2.1. Decisiones tomadas para resolverlo: . . . 3

1.2.2. Procedimiento: . . . 5

1.3. trayectoriaExtrañaEscalera . . . 6

1.3.1. Decisiones tomadas para resolverlo: . . . 6

1.3.2. Procedimiento: . . . 6

1.4. individuoDecideSiCooperarONo . . . 7

1.4.1. Decisiones tomadas para resolverlo: . . . 7

1.4.2. Procedimiento: . . . 8

1.5. individuoActualizaApuesta . . . 9

1.5.1. Decisiones tomadas para resolverlo: . . . 9

1.5.2. Procedimiento: . . . 10

1.6. Auxiliares . . . 11

1.6.1. Predicados Recurrentes . . . 11

pagosValidos . . . 11

eventosValidos . . . 11

apuestasValidas . . . 11

recursosValidos . . . 11

trayectoriasConRecursosValidos . . . 11

sonTrayectoriasValidas . . . 12

tieneRecursosPorCadaEvento . . . 12

tieneRecursosIniciales . . . 12

1.6.2. Funciones . . . 13

Ejercicio 1: . . . 13

Ejercicio 2: . . . 13

Ejercicio 3: . . . 13

Ejercicios 4 y 5: . . . 13

2. Demostraciones de correctitud **14**

2.1. Objetivo . . . 14

2.2. Procedimiento: . . . 14

2.3. Sentencia **0** . . . 15

2.4. Sentencia **I** . . . 15

2.5. Sentencia **II** . . . 15

2.6. Sentencia **III** . . . 16

2.7. Sentencia **IV** . . . 18

2.8. Sentencia **V** . . . 20

1. Especificación

Observación: Las funciones auxiliares se encuentran al final del documento.

1.1. redistribucionDeLosFrutos

1.1.1. Decisiones tomadas para resolverlo:

La resolución de este ejercicio requirió definir en la sección del **Requiere** de la especificación que todo recurso recibido en la secuencia sea mayor o igual a 0, ya que no tiene sentido que un individuo participe estando endeudado.

Además, se necesita que el largo de la secuencia *recursos* sea igual al largo de la secuencia *cooperan*, pues cada índice de *recursos* representa un individuo, y para ese individuo debe estar vinculado a la secuencia *cooperan* con el mismo índice (según el enunciado) para identificar si el individuo cooperará o no durante el juego.

1.1.2. Procedimiento:

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

requiere $\{(\forall$ recurso $\in$ recursos $\longrightarrow_L recurso \geq 0) \land |recursos| = |cooperan|\}$

asegura $\{$

$(|res| = |cooperan|) \land$

$(|res| = |recursos|) \land$

$((\forall i : \mathbb{Z})\ ((0 \leq i < |res|) \longrightarrow_L$

$res[i] =$

$(\text{if } cooperan[i] = True \text{ then } \frac{fondoDeRecursoInicial(recursos, cooperan)}{|cooperan|} \text{ else } recursos[i] + \frac{fondoDeRecursoInicial(recursos, cooperan)}{|cooperan|}$

$\}$

1.2. trayectoriaDeLosFrutosIndividualesALargoPlazo

1.2.1. Decisiones tomadas para resolverlo:

Para la sección de **Requiere**:

- Se define la variable antigua y nueva de *trayectorias*, puesto que es un parámetro *inout*.
- Los tamaños de *trayectorias*, *apuestas*, *eventos*, *cooperan* y *pagos* son equivalentes:
  - Cada elemento de *trayectorias* representa la trayectoria para un individuo, y la cooperación de cada individuo se define en la secuencia *cooperan*. Entonces, deben tener el mismo tamaño.
  - Cada elemento de *apuestas* y *eventos* representa cuánto apostará el individuo para el próximo evento, por lo que si no hay evento, no hay apuesta. Por este motivo, cada elemento de estas dos secuencias se asocia a un individuo: deben, por lo tanto, tener el mismo tamaño.
  - Cada elemento de *pagos* también está asociado a un individuo, por lo que la matriz de pagos tendrá también el mismo tamaño que las mencionadas. Como todas estas secuencias de secuencias, o matrices, se asocian a individuos, todas tienen el mismo tamaño entre sí.
  - Todo individuo en *trayectoria* tiene el mismo largo que *trayectoria*.
  - Todo natural dentro de cada evento debe ser menor al tamaño de cada evento por individuo, ya que usamos este valor para indexar cuanto apostó y cuanto se pagó dado un evento en particular.

    Para todo elemento de las secuencias de secuencias (es decir, para todas las secuencias) el largo de estos son equivalentes entre sí, incluidos en la misma secuencia de secuencias.

Esto es porque todo se relaciona con la çantidad de veces que se juega": no puedo tener mas apuestas que eventos; ni más pagos que apuestas; ni más eventos que pagos o apuestas. Además, aplica lo mismo para todos los individuos (jugadores), entonces se deben corresponder entre sí los tamaños. Por ejemplo, sean las siguientes secuencias:

```
Apuesta = [0.1, 0.9]
Pago = [5, 3]
Recurso = [1]
Evento = [1, 0]
```

Dado que el primer evento salió 1, entonces $\omega_1 : 1 * Pago[1] * Apuesta[1]$

<u>Utilizamos varios predicados auxiliares:</u>

** 1.6.1 trayectoriasConRecursosValidos(trayectorias):

Dada una secuencia de secuencias de reales, el largo de cada secuencia es mayor a cero, y luego cada elemento de cada secuencia es mayor o igual a 0: dado este juego, no tiene sentido que el recurso de un jugador sea negativo.

** 1.6.1 eventosValidos(eventos):

Dada una secuencia de secuencias de naturales, cada elemento de la secuencia de eventos del individuo no puede ser mayor al rango de la secuencia: usamos el índice del evento para indexar qué pago y qué apuesta le corresponde a un evento dado. Aplica el mismo ejemplo dado anteriormente.

** 1.6.1 pagosValidos(pagos):

Dada una secuencia de secuencia de pagos, para cada secuencia de pago, cada elemento es mayor o igual a 0.

** 1.6.1 apuestasValidas(apuestas):

Dada una secuencia de secuencias de apuestas, para cada secuencia de apuesta, cada elemento es mayor o igual a 0; y la suma de todas las apuestas de un individuo es 1: pues hablamos de proporciones, y asumimos que un jugador apuesta todo en cada ronda.

Para la sección de **Asegura**:

- El largo de trayectorias es el mismo que el largo de trayectorias de entrada: esto debemos asegurarlo para que la cantidad de individuos no cambie durante el transcurso del juego, solo el tamaño de cada trayectoria dentro de trayectorias.
- El tamaño de cada trayectoria de cada individuo de trayectorias será el largo de eventos mas uno, esto se debe a que se definen todos los nuevos recursos para todas las trayectorias a partir de todos los eventos dados, y el último evento genera un nuevo recurso final obtenido, Ya que el primer recurso no se obtuvo por evento, entonces el índice está movido uno a la izquierda.
- Para cada recurso a partir de ω_1, estos se calculan a partir de eventos, pagos y apuestas, y se asignan a los índices agregados de cada trayectoria.
- Como ω_0 es el recurso inicial, este no cambia para ninguna trayectoria. Entonces, se especifica que no debe cambiar al realizarse el *out*.

Utilizamos varios predicados auxiliares:

** tieneRecursosPorCadaEvento(trayectorias, apuestas, pagos, cooperan, eventos) : Para todo individuo de trayectorias, entonces luego el largo de individuo será el largo de eventos mas uno y luego para todo recurso del individuo del índice uno al final será el calculo del nuevo recurso a partir del evento, pago, apuesta y cooperación del individuo, básicamente el nuevo recurso será calcular a partir del evento, y recurso del índice anterior el nuevo.

1.2.2. Procedimiento:

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

requiere $\{$
$(trayectorias_0 = trayectorias) \wedge$

$((\forall individuo_1, individuo_2 : \mathbb{N})\ (0 \leq individuo_1, individuo_2 < |trayectorias| \wedge individuo_1 \neq individuo_2 \longrightarrow_L$

$|trayectorias[individuo_1]| = |trayectorias[individuo_2]|)) \wedge$

$trayectoriasConRecursosValidos(trayectorias) \wedge eventosValidos(eventos) \wedge pagosValidos(pagos) \wedge$
$apuestasValidas(apuestas) \wedge$

$((\forall i, j : \mathbb{N})\ (0 \leq i, j < |apuestas| \wedge i \neq j \longrightarrow_L$
$|apuestas[i]| = |apuestas[j]| = |eventos[i]| = |eventos[j]| = |pagos[i]| = |pagos[j]|))\}$

asegura $\{$
$(|trayectorias_0| = |trayectorias|) \wedge$
$tieneRecursosPorCadaEvento(trayectorias, apuestas, pagos, cooperan, eventos) \wedge$

$((\forall individuo : \mathbb{N})\ (0 \leq individuo < |trayectorias| \longrightarrow_L$
trayectorias[individuo][0] = $\mathtt{trayectorias}_0[individuo][0]$
))
$\}$

1.3. trayectoriaExtrañaEscalera

1.3.1. Decisiones tomadas para resolverlo:

Para que exista un máximo necesitamos que por lo menos haya un elemento en la secuencia, por este motivo decidimos especificar en la sección de Requiere que el tamaño de la secuencia recibida deba ser mayor a 0.

Para la sección de Asegura, se decidió dividir en tres casos:

- Primer caso: $|trayectoria| = 1$. Esto implica que el único elemento es un máximo local, entonces si la secuencia es largo uno, devuelve True (hay máximo local).

- Segundo caso: $|trayectoria| = 2$. Si los dos elementos de la trayectoria son diferentes, entonces uno de ellos es el máximo. Si ese es el caso, se devuelve *True* (hay máximo local).

- Tercer caso: $|trayectoria| \geq 3$. Acá hay que tener en cuenta el medio y los extremos, ya que para ser máximo local el extremo solo tiene que fijarse en el elemento a su lado.

Como no es igual a uno, res devuelve *False*.Usamos contadores para el medio y para los extremos: si la suma de los 3 contadores es igual a uno, entonces hay un máximo local. De lo contrario, el predicado devuelve *False*.
Como no es igual a uno, res devuelve *False*.

1.3.2. Procedimiento:

proc trayectoriaExtrañaEscalera (in trayectoria : $seq\langle\mathbb{R}\rangle$) : Bool
requiere $\{|trayectoria| > 0\}$
asegura {
$res = True \Leftrightarrow$
$(|trayectoria| = 1) \vee$
$(|trayectoria| = 2 \wedge_L (trayectoria[0] \neq trayectoria[1])) \vee$
$(|trayectoria| >= 3 \wedge_L$
$((contadorMayoresExtremoMedio(trayectoria) +$
$contadorMayoresExtremoIzquierdo(trayectoria) +$
$contadorMayoresExtremoDerecho(trayectoria)) = 1))$
}

1.4. individuoDecideSiCooperarONo

1.4.1. Decisiones tomadas para resolverlo:

Para la sección de **Requiere**, especificamos lo necesario para que los parámetros de entrada, entrada/salida cumplan con lo necesario para poder realizar el procedimiento:

- Especificamos el nombre de entrada y de salida para la secuencia *cooperan*.

- Especificamos que el número natural *individuo* recibido como parámetro debe estar en el rango de la secuencia *cooperan*.

- Especificamos que los tamaños de las secuencias y de las secuencias de secuencias (matrices) deben ser iguales: cada elemento de estas corresponde a un individuo. Entonces, al igual que en ejercicios previos, cada individuo está representado por un elemento dentro de los arreglos: el tamaño termina siendo el mismo.

- Especificamos que los tamaños de los elementos de apuestas, pagos y eventos deben ser equivalentes: por cada evento hay una apuesta y pago por individuo, entonces sus tamaños debe ser iguales.

- Utilizamos los mismos predicados que el ejercicio 1.2: esto me asegura que las secuencias recibidas estarán bien formadas y serán válidas.

Para la sección de **Asegura**:

- Especificamos que, para todo elemento de *cooperan*, todos los individuos deban mantener su mismo valor de entrada en *cooperan*, menos el del individuo recibido como parámetro: es decir, solo deberá actualizarse el de éste.

- Luego, el valor que tendrá el individuo en *cooperan* dependerá de si le conviene cooperar o no: según si, a partir de los eventos recibidos, tendrá más recursos al final de estos cooperando o no cooperando.

Para esto, decimos que la cooperación del individuo será *True* si y solo si el último recurso del individuo es mayor al recurso que obtendría si no cooperara (de lo contrario, la implicación devolvería *False* y el individuo no cooperaría luego de la actualización).

Para calcular este recurso final, utilizamos el predicado 1.6.1 $sonTrayectoriasValidas(trayectoria, recursos, apuestas, pagos, cooperan, eventos)$.

Dados estos parámetros para el predicado, se devolverá *True* cuando el primer elemento de las trayectorias de los individuos sea el recurso asociado al mismo individuo de la secuencia recursos, según el predicado 1.6.1 $tieneRecursosIniciales(trayectorias, recursos)$.

Además, se utiliza el predicado $tieneRecursoPorCadaEvento(trayectorias, apuestas, pagos, cooperan, eventos)$, que devuelve *True* cuando la trayectoria corresponde con las apuestas, pagos y eventos de las secuencias para cada individuo, y toda *trayectoria* en *trayectorias* tiene el mismo largo.

Luego, si existe una trayectoria A que cumple esto y donde el valor de *cooperan* del individuo es *True*; y otra trayectoria B que cumple esto y donde el valor de *cooperan* del individuo sea *False*; luego comparamos qué recurso final es el mayor. Si el recurso final de la trayectoria A es mayor a la de B, el valor de *cooperan* del individuo en la secuencia de salida será *True*. De lo contrario, será *False*.

Observación: *SetAt* nos permite setear en la secuencia el valor deseado para ver qué recurso final será el mayor.

1.4.2. Procedimiento:

`proc individuoDecideSiCooperarONo` (in individuo : $\mathbb{N}$, in recursos : $seq\langle \mathbb{R} \rangle$, inout cooperan : $seq\langle \mathsf{Bool} \rangle$, in apuestas : $seq\langle seq\langle \mathbb{R} \rangle\rangle$, in pagos : $seq\langle seq\langle \mathbb{R} \rangle\rangle$, in eventos : $seq\langle seq\langle \mathbb{N} \rangle\rangle$) : $seq\langle \mathsf{Bool} \rangle$

`requiere` {
$(cooperan_0 = cooperan) \land (0 \leq individuo < |cooperan_0|) \land_L$

$(|apuestas| = |eventos| = |recursos| = |pagos| = |cooperan_0|) \land$

$eventosValidos(eventos) \land pagosValidos(pagos) \land apuestasValidas(apuestas) \land recursosValidos(recursos) \land$

$((\forall i, j : \mathbb{N})\ (0 \leq i < |apuestas| \land i \neq j \longrightarrow_L$
$|apuestas[i]| = |apuestas[j]| = |eventos[i]| = |eventos[j]| = |pagos[i]| = |pagos[j]|))$
}

`asegura` {
$|cooperan_0| = |cooperan| \land$

$((\forall i : \mathbb{N})\ (0 \leq i < |cooperan| \land i \neq individuo \longrightarrow_L$
$cooperan[i] = cooperan_0[i])) \land$

$cooperan[individuo] = True \Leftrightarrow$

$(\exists trayectoriasA : seq\langle seq\langle \mathbb{R} \rangle\rangle)$
$(sonTrayectoriasValidas(trayectoriasA, recursos, apuestas, pagos, SetAt(individuo, True, cooperan_0), eventos)) \land_L$

$(\exists trayectoriasB : seq\langle seq\langle \mathbb{R} \rangle\rangle)$
$sonTrayectoriasValidas(trayectoriasB, recursos, apuestas, pagos, SetAt(individuo, False, cooperan_0), eventos) \land_L$

$ultimaGanancia(trayectoriasA[individuo]) \geq$
$ultimaGanancia(trayectoriasB[individuo])$
}

1.5. individuoActualizaApuesta

1.5.1. Decisiones tomadas para resolverlo:

Para la sección de **Requiere**, especificamos lo necesario para que los parámetros de entrada y de entrada/salida cumplan lo necesario para poder realizar el procedimiento:

- Especificamos el nombre de la instancia de entrada y de salida para la secuencia *cooperan*.

- Especificamos que el parámetro *individuo* esté en el rango de la matriz *apuestas*.

- Al igual que en ejercicios previos, requerimos que los largos de los arreglos se correspondan entre sí: al estar relacionados con un individuo, no puede suceder que tengan más o menos apuestas, pagos, cooperación, eventos, o recursos para una cierta cantidad de individuos.

- Nos aseguramos de que los tamaños de las matrices *apuestas*, *pagos* y *eventos* todos tengan el mismo largo, y lo mismo con sus elementos (secuencias de números).

- Todo elemento de **pagoIndividual** debe ser mayor o igual a cero.

- Al igual que para cada elemento de elemento de *pagos*, toda apuesta de cada individuo debe ser positiva (no tendría sentido que un individuo apueste una cantidad negativa).

- Por último, todo elemento de la secuencia de recursos es mayor o igual a cero (los individuos no podrán endeudarse).

Para la sección de **Asegura**:

- El enunciado nos dice que se actualiza la secuencia de *apuestas* dado un individuo, según los eventos, apuestas, pagos y cooperaciones tales que sus recursos aumenten en cada ronda.

Para poder resolver esto primero tenemos que especificar que toda secuencia de *apuestas* que no correspondan al individuo se mantendrá como se recibió (la matriz *apuestas* es *inout*).

En otras palabras, nos aseguramos de que todos los elementos de *apuestas* de otros individuos sean iguales, y que el largo de cada secuencia de apuestas de otros individuos no cambie.

- Para resolver la actualización de cada apuesta según el mayor recurso a obtener, lo que decimos hacer es definir que para toda apuesta realizada por el individuo exista un valor a entre 0 y 1, tal que para cualquier otro valor b entre 0 y 1, el valor del nuevo recurso obtenido a partir del evento dado sea el mayor.

Una vez que se cumple la existencia de este a que genera el mayor recurso, se setea ese valor en el elemento de la secuencia de apuestas del individuo. Dado que este procedimiento es llevado a cabo para todo elemento de la secuencia de apuestas del individuo, cada vez que se calcule el nuevo recurso se va a tomar el recurso anteriormente obtenido.

Entonces, al final de todas las trayectorias, la secuencia de apuestas del individuo tendrá los valores que más recursos le generen.

1.5.2. Procedimiento:

proc individuoActualizaApuesta (in individuo : $\mathbb{N}$, in recursos : $seq\langle\mathbb{R}\rangle$, in cooperan : $seq\langle\mathsf{Bool}\rangle$, inout apuestas : $seq\langle seq\langle\mathbb{R}\rangle\rangle$, in pagos : $seq\langle seq\langle\mathbb{R}\rangle\rangle$, in eventos : $seq\langle seq\langle\mathbb{N}\rangle\rangle$) : $seq\langle seq\langle\mathbb{R}\rangle\rangle$

requiere {

$(apuestas_0 = apuestas) \quad \wedge$

$(0 \leq individuo < |apuestas_0|) \ \wedge_L$

$(|apuestas| = |eventos| = |recursos| = |pagos| = |cooperan_0|) \quad \wedge_L$

$((\forall i, j : \mathbb{N}) \ \ (0 \leq i < |apuestas| \wedge i \neq j \quad \longrightarrow_L$
$|apuestas[i]| = |apuestas[j]| = |eventos[i]| = |eventos[j]| = |pagos[i]| = |pagos[j]|)) \quad \wedge$

$eventosValidos(eventos) \wedge pagosValidos(pagos) \wedge apuestasValidas(apuestas) \wedge recursosValidos(recursos)$
}

asegura {

$(|apuestas[individuo]| = |apuestas_0[individuo]|) \ \wedge_L$

$((\exists secuenciaApuestasA : seq\langle\mathbb{R}\rangle) \ \ (|secuenciaApuestasA| = |apuestas_0[individuo]|) \wedge$
$apuestasValidas(secuenciaApuestasA) \ \wedge_L$

$((\forall secuenciaApuestasB : seq\langle\mathbb{R}\rangle) \ \ (secuenciaApuestasA \neq secuenciaApuestasB \wedge$
$apuestasValidas(secuenciaApuestasB) \ \wedge$
$|secuenciaApuestasB| = |apuestas_0[individuo]|)) \quad \longrightarrow_L$

$(\exists trayectoriasA : seq\langle seq\langle\mathbb{R}\rangle\rangle)$
$(sonTrayectoriasValidas(trayectoriasA, secuenciaApuestasA, pagos, cooperan, eventos) \ \wedge_L$

$(\exists trayectoriasB : seq\langle seq\langle\mathbb{R}\rangle\rangle)$
$sonTrayectoriasValidas(trayectoriasB, secuenciaApuestasB, pagos, cooperan, eventos) \ \wedge_L$

$ultimoRecurso(trayectoriasA[individuo]) \geq$
$ultimoRecurso(trayectoriasB[individuo]) \ \wedge_L$

$apuestas[individuo] = secuenciaApuestasA) \ \wedge_L$

$((\forall i : \mathbb{N}) \ \ (0 \leq i < |apuestas| \wedge i \neq individuo \quad \longrightarrow_L$
$apuestas[i] = apuestas_0[i]))$
}

1.6. Auxiliares

1.6.1. Predicados Recurrentes

pagosValidos

`pred pagosValidos (in pagos :` $seq\langle seq\langle \mathbb{R}\rangle\rangle$ `) {`

$((\forall pagosIndividuo : \mathbb{N})\ (0 \leq pagosIndividuo < |pagos| \longrightarrow_L$

$((\forall pago : \mathbb{N})\ (0 \leq pago < |pagosIndividuo| \longrightarrow_L 0 \leq pagos[pagosIndividuo][pago]))))$

`}`

eventosValidos

`pred eventosValidos ( in eventos :` $seq\langle seq\langle \mathbb{N}\rangle\rangle$`) {`

$((\forall eventosIndividuo : \mathbb{N})\ (0 \leq eventosIndividuo < |eventos| \longrightarrow_L$

$(\forall evento : \mathbb{N})\ (eventos[eventosIndividuo][evento] < |eventos[eventosIndividuo]|)))$

`}`

apuestasValidas

`pred apuestasValidas ( in apuestas :` $seq\langle seq\langle \mathbb{R}\rangle\rangle$`) {`

$((\forall apuestaIndividuo : \mathbb{N})\ (0 \leq apuestaIndividuo < |apuestas| \longrightarrow_L$

$((\forall apuesta : \mathbb{N})\ (0 \leq apuesta < |apuestaIndividuo| \longrightarrow_L 0 \leq apuestas[apuestaIndividuo][apuesta]))))\wedge_L$

$((\forall apuestaIndividuo : \mathbb{N})\ (0 \leq apuestaIndividuo < |apuestas| \longrightarrow_L$

$sumarSecuencia(apuestaIndividuo) = 1))$

`}`

recursosValidos

`pred recursosValidos ( in recursos :` $seq\langle \mathbb{R}\rangle$`) {`

$(\forall$ `recurso` $\in$ `recursos` $\longrightarrow_L recurso \geq 0))$

`}`

trayectoriasConRecursosValidos

`pred trayectoriasConRecursosValidos ( in trayectorias :` $seq\langle seq\langle \mathbb{R}\rangle\rangle$`) {`

$((\forall individuo : \mathbb{N})\ (0 \leq individuo < |trayectorias| \longrightarrow_L$

$|individuo| \geq 0 \wedge_L ((\forall w : \mathbb{N})\ (0 \leq w < individuo \longrightarrow_L 0 \leq trayectorias[individuo][w]))))$

`}`

sonTrayectoriasValidas

pred sonTrayectoriasValidas (in trayectorias : $seq\langle seq\langle\mathbb{R}\rangle\rangle$, in recursos : $seq\langle\mathbb{Z}\rangle$, in apuestas : $seq\langle seq\langle\mathbb{R}\rangle\rangle$, in pagos : $seq\langle seq\langle\mathbb{R}\rangle\rangle$, in cooperan : $seq\langle\mathsf{Bool}\rangle$, in eventos : $seq\langle\mathbb{N}\rangle$) {
$(tieneRecursosIniciales(trayectorias, recursos) \wedge |eventos| = 0) \vee$
$(tieneRecursosIniciales(trayectorias, recursos) \wedge |eventos| > 0 \wedge$
$tieneRecursoPorCadaEvento(trayectorias, apuestas, pagos, cooperan, eventos) \wedge_L$

$((\forall i, j : \mathbb{N})\ (0 \leq i, j < |trayectorias| \wedge i \neq j \longrightarrow_L$
$|trayectorias[i]| = |trayectorias[j]|))$
}

tieneRecursosPorCadaEvento

pred tieneRecursosPorCadaEvento (in trayectorias : $seq\langle seq\langle\mathbb{R}\rangle\rangle$, in apuestas : $seq\langle seq\langle\mathbb{R}\rangle\rangle$, in pagos : $seq\langle seq\langle\mathbb{R}\rangle\rangle$, in cooperan : $seq\langle\mathsf{Bool}\rangle$, in eventos : $seq\langle\mathbb{N}\rangle$) {
$((\forall individuo : \mathbb{N})\ (0 \leq individuo < |trayectorias| \longrightarrow_L$
$(|trayectorias[individuo]| = |eventos| + 1 \wedge_L$
$(\forall \omega : \mathbb{N})\ (1 \leq \omega < |individuos| \longrightarrow_L$
$trayectorias[individuo][\omega] = RecursoIndividuoEnTrayectoria(\omega - 1, individuo, trayectorias,$
$cooperan, eventos, pagos, apuestas))))$
}

tieneRecursosIniciales pred tieneRecursosIniciales (in trayectorias : $seq\langle seq\langle\mathbb{R}\rangle\rangle$, in recursos : $seq\langle\mathbb{R}\rangle$) {
$((\forall individuo : \mathbb{N})\ (0 \leq individuo < |trayectorias| \longrightarrow_L$
$trayectorias[individuo][0] = recursos[individuo])$
}

1.6.2. Funciones

Ejercicio 1:

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

Ejercicio 2:

aux fondo (in $\omega : \mathbb{N}$, in $trayectorias : seq\langle seq\langle\mathbb{R}\rangle\rangle$, in $cooperan : seq\langle\mathsf{Bool}\rangle$, in $eventos : seq\langle seq\langle\mathsf{Bool}\rangle\rangle$, in $pagos$: $seq\langle seq\langle\mathbb{R}\rangle\rangle$, in $apuestas : seq\langle seq\langle\mathbb{R}\rangle\rangle$) : $\mathbb{Z}$ =
$\sum_{i=0}^{|trayectorias|-1}$ (if $cooperan[i] = True$ then $recursoIndividuo(\omega, i, trayectorias, pagos, apuestas)$ else 0 fi) ;

aux recursoIndividuo (in $\omega : \mathbb{N}$, in $\imath : \mathbb{N}$, in $trayectorias : seq\langle seq\langle\mathbb{R}\rangle\rangle$in $eventos : seq\langle seq\langle\mathbb{N}\rangle\rangle$, in $pagos$: $seq\langle seq\langle\mathbb{R}\rangle\rangle$,
in $apuestas : seq\langle seq\langle\mathbb{R}\rangle\rangle$) : $\mathbb{Z}$ =
$(trayectorias[i][\omega] * apuestas[i][eventos[\omega]] * pagos[i][eventos[\omega])$;

aux divisionFondos (in $\omega : \mathbb{N}$, in $trayectorias : seq\langle seq\langle\mathbb{R}\rangle\rangle$, in $cooperan : seq\langle\mathsf{Bool}\rangle$, in $eventos : seq\langle seq\langle\mathbb{N}\rangle\rangle$, in $pagos$: $seq\langle seq\langle\mathbb{R}\rangle\rangle$, in $apuestas : seq\langle seq\langle\mathbb{R}\rangle\rangle$) : $\mathbb{Z}$ =

$\frac{fondo(\omega, trayectorias, cooperan, eventos, pagos, apuestas)}{|cooperan|}$;

aux recursoIndividuoEnTrayectoria (in $\omega : \mathbb{N}$, in $individuo : \mathbb{N}$, $trayectorias : seq\langle seq\langle\mathbb{R}\rangle\rangle$, in $cooperan$: $seq\langle\mathsf{Bool}\rangle$, in $eventos : seq\langle seq\langle\mathbb{N}\rangle\rangle$, in $pagos : seq\langle seq\langle\mathbb{R}\rangle\rangle$, in $apuestas : seq\langle seq\langle\mathbb{R}\rangle\rangle$) : $\mathbb{Z}$ =
if $cooperan[individuo]$ then
$divisionFondos(\omega$, trayectorias, cooperan, eventos, pagos, apuestas) else
recursoIndividuo(ω,individuo, trayectorias, eventos, pagos, apuestas) +
divisionFondos(ω, trayectorias, cooperan, eventos, pagos, apuestas) fi ;

aux sumarSecuencia (in secuencia : $seq\langle\mathbb{R}\rangle$) : $\mathbb{Z}$ =
$\sum_{i=1}^{|secuencia|-1} s[i]$;

Ejercicio 3:

aux contadorMayoresExtremoMedio (in trayectoria : $seq\langle\mathbb{R}\rangle$) : $\mathbb{Z}$ =
$\sum_{i=1}^{|trayectoria|-2}$ (if $(trayectoria[i-1]) < (trayectoria[i]) \wedge (trayectoria[i]) > (trayectoria[i+1]) = True$ then 1 else 0 fi) ;

aux contadorMayoresExtremoIzquierdo (in trayectoria : $seq\langle\mathbb{R}\rangle$) : $\mathbb{Z}$ =
if $(trayectoria[0]) > (trayectoria[1]) = True$ then 1 else 0 fi) ;

aux contadorMayoresExtremoDerecho (in trayectoria : $seq\langle\mathbb{R}\rangle$) : $\mathbb{Z}$ =
if $(trayectoria[|trayectoria|-1]) > (trayectoria[|trayectoria|-2]) = True$ then 1 else 0 fi) ;

Ejercicios 4 y 5:

aux ultimoRecurso (in secuencia : $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z}$ = $ultimoElemento(secuencia)$;
aux ultimoElemento (in secuencia : $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z}$ = $secuencia[|secuencia|-1]$;

2. Demostraciones de correctitud

2.1. Objetivo

Utilizando **Métodos Formales** y, en concreto, el **Teorema de corrección de un ciclo** visto en el apartado teórico de la materia, demostrar que el código en SmallLang dado en el enunciado es correcto.

2.2. Procedimiento:

Sean:

- $P_c \equiv apuesta_c + apuesta_s = 1 \land pago_c > 0 \land pago_s > 0 \land apuesta_c > 0 \land apuesta_s > 0 \land recurso > 0 \land res = recurso \land i = 0$
- $I \equiv 0 \leq i \leq |eventos| \land_L$
  $res = recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),T)} * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),F)}$
- $Q_c \equiv res = recurso * (apuesta_c, pago_c)^{\#(eventos,T)} * (apuesta_c, pago_c)^{\#(eventos,F)}$
- $fv \equiv |eventos| - i$

Para demostrar que la especificación dada es correcta respecto a la implementación del enunciado, debemos probar la *tripla de Hoare*:

$$\{P_c\} \texttt{ while B do S endwhile } \{Q_c\}$$

Para esto, el **Teorema de corrección de un ciclo** nos indica que basta con demostrar que las siguientes 5 sentencias son válidas:

1. $P_c \Rightarrow I$
2. $I \land \neg B \Rightarrow Q_c$
3. $\{I \land B\}\ S\ \{I\}$
4. $\{I \land B \land v_0\}\ S\ \{fv < v_0\}$
5. $I \land fv \leq 0 \Rightarrow \neg B$

2.3. Sentencia 0

¿$P \Rightarrow P_c$?

Debemos ver que el *requiere* implica la precondición del ciclo:

$apuesta_c + apuesta_s = 1 \land pago_c{>}0 \land pago_s{>}0 \land apuesta_c{>}0 \land apuesta_s{>}0 \land recurso{>}0$

$\Rightarrow$

$apuesta_c + apuesta_s = 1 \land pago_c{>}0 \land pago_s{>}0 \land apuesta_c{>}0 \land apuesta_s{>}0 \land recurso{>}0 \land res = recurso \land i = 0$

$\equiv$

$True$

□

2.4. Sentencia I

¿$P_c \Rightarrow I$?

Asumimos que vale P_c. En la implicación, reemplazamos *res* por *recurso* e *i* por 0.
Luego, en la implicación nos queda 0 $\leq$ 0, que es tautológico; y la subSeq(0, 0, eventos) devuelve una secuencia vacía, ya que su segundo parámetro es excluyente, y la cantidad de apariciones de T o F en una secuencia vacía es 0.

En conclusión, se cumple que $P_c \Rightarrow I$.

P_c

$\equiv$

$apuesta_c + apuesta_s = 1 \land pago_c{>}0 \land pago_s{>}0 \land apuesta_c{>}0 \land apuesta_s{>}0 \land recurso{>}0 \land res = recurso \land i = 0$

$\Rightarrow$ <--- Asumimos P_c y reemplazamos *res* e *i* en I

$0 \leq 0 \leq |eventos| \land_L$
$recurso = recurso * (apuesta_c, pago_c)^{\#(subSeq(0,0,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,0,eventos),F)}$

$\equiv$

$0 \leq 0 \leq |eventos| \land_L recurso = recurso * (apuesta_c, pago_c)^0 * (apuesta_s, pago_s)^0$

$\equiv$

0 $\leq 0 \leq |eventos| \land_L recurso = recurso * 1$

$\equiv$

0 $\leq |eventos| \land_L True$

$\Rightarrow$

$True$

□

2.5. Sentencia II

¿$I \land \neg B \Rightarrow Q_c$?

Probamos que el invariante y la negación de la guarda cumplen la post-condición del ciclo: esto significa que si sale del ciclo y el invariante se cumple, entonces también se cumple la post-condición.

Asumimos que vale el antecedente. En Q_c reemplazamos *i* por */eventos/*: ya que por el invariante deducimos que $i \leq |eventos|$; y, por la negación de la guarda, sabemos que $i \geq |eventos|$. entonces i solo puede ser igual a */eventos/*. Al reemplazar, llegamos a que las potencias de cada pago y apuesta nos devuelven la cantidad de apariciones de la subsecuencia completa de eventos de *T* o *F*.

En conclusión, subSeq(0,eventos,eventos) = eventos, y buscamos la cantidad de apariciones dentro de la secuencia completa de eventos, que es equivalente a Q_c. Luego, se cumple la implicación.

$I \wedge \neg B$
$\equiv$
$0 \leq i \leq |eventos| \ \wedge_L$
$res = recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i,eventos),F)} \wedge$
$i \geq |eventos|$
$\Rightarrow$
$i = |eventos| \wedge_L$
$res = recurso * (apuesta_c, pago_c)^{\#(subSeq(0,eventos,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,eventos,eventos),F)}$
$\Rightarrow$
Q_c

$\square$

2.6. Sentencia III

¿$\{I \wedge B\} \ S \ \{I\}$?

$I \wedge \ B$
$\equiv$
$0 \leq i \leq |eventos| \ \wedge_L$
$res = recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i,eventos),F)} \ \wedge$
$i < |eventos|$
$\equiv$
$0 \ \leq i < |eventos| \wedge_L$
$res = recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i,eventos),F)}$

Queremos probar que I $\wedge B \Rightarrow wp($if B then S_1 else S_2, i = i + 1, I).
Por el axioma 3 (secuencia de programas), esta expresión es equivalente a wp(if B then S_1 else S_2, wp(i = i + 1, I)).

Sea K = wp(i = i + 1, I). Aplicando el axioma 1 (asignación):

K
$=$
$def(i+1) \ \wedge_L$ 0 $\leq i+1 \leq |eventos| \ \wedge_L$
$res = recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i+1,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i+1,eventos),F)}$
$\equiv$
$True \wedge_L 0 \leq i+1 \leq |eventos| \wedge_L$
$res = recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i+1,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i+1,eventos),F)}$

Continuamos buscando wp(if B then S_1 else S_2, K). Usando el axioma 4 de la wp:

wp(if B then S_1 else S_2, K) = $def(B) \ \wedge_L \ ((B \wedge wp(S_1, K) \vee (\neg B \wedge wp(S_2, K)))$
$\equiv$
$def(i < \ eventos[i]) \wedge_L \ \ ((eventos[i] \ \wedge \ wp(res = res * apuesta_c * pago_c, K)) \vee$
$(\neg eventos[i] \wedge wp(res = res * apuesta_s * pago_s, K))$

$\equiv$ <--- Aplicando el axioma 1 en ambas wp en K:

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

$((eventos[i] \wedge def(res * apuesta_c * pago_c) \ \wedge_L$
$0 \leq i+1 \leq |eventos| \wedge_L$
$res * apuesta_c * pago_c =$
$recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i+1,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i+1,eventos),F)}) \vee$

$(\neg eventos[i] \wedge def(res * apuesta_s * pago_s) \ \wedge_L$
$0 \leq i+1 \leq |eventos| \wedge_L$

$res * apuesta_s * pago_s =$
$recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i+1,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i+1,eventos),F)}))$

Ajustando el rango de i según la conjunción:
$0 \leq i < |eventos| \wedge_L 0 \leq i+1 \leq |eventos| \equiv 0 \leq i < |eventos|$

Simplificando proposiciones:

$(0 \leq i < |eventos| \wedge_L$
$(eventos[i] \wedge_L$
$res * apuesta_c * pago_c =$
$recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i+1,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i+1,eventos),F)}) \vee$

$(\neg eventos[i] \wedge_L$
$res * apuesta_s * pago_s =$
$recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i+1,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i+1,eventos),F)}))$

Queremos ver que $I \wedge B$ implica esta fórmula:
Si analizamos por partes, el rango de *i* en el antecedente es el mismo que en el consecuente: por lo tanto, es verdadero. Ahora nos queda por analizar la unión de las dos partes del *if*, y ver si la implicación se cumple.

Analicemos el primer caso (si *eventos[i]* es *True*):

Si asumimos que *eventos[i]* es *True*, entonces podemos restarle uno al índice de la cantidad de apariciones en *eventos* de *False* (si *eventos[i]* es *True*, entonces no puede ser *False*). Por lo tanto, llegamos a que:

$eventos[i] \wedge_L$
$res * apuesta_c * pago_c =$
$recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i+1,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i,eventos),F)}$

Ahora bien, a *res* lo estamos multiplicando por $(apuesta_c, pago_c)$, entonces puedo pasarlo dividiendo para el otro lado de la igualdad: por este motivo, dejamos el segundo parámetro de la subsecuencia como i + 1 - 1 = i. Entonces, se sucede que:
$eventos[i] \wedge_L$
$res =$
$recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i+1-1,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i,eventos),F)}$

$\equiv$

$eventos[i] \wedge_L$
$res =$
$recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i,eventos),F)}$

Exactamente la misma lógica se aplica para la otra proposición con $\neg$*eventos[i]*, pero en este caso es a la inversa: sabemos que *eventos[i] = False*. Por lo tanto, podemos restarle uno al segundo parámetro de la subsecuencia de cantidad de apariciones donde *eventos* es *True*.

$\neg eventos[i] \wedge_L$
$res * apuesta_s * pago_s =$
$recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i+1,eventos),F)}$

Pasamos diviendo $apuesta_s * pago_s$, y esto nos permite mover el índice de cantidad de apariciones de *False*: se estaría ''sacando'' uno al contador al dividirlo.

Juntando todo de nuevo, se tiene que:

$((eventos[i] \wedge_L$
$res = recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i,eventos),F)})$
$\vee$
$(\neg eventos[i] \wedge_L$
$res * apuesta_s * pago_s =$

$recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i,eventos),F)}))$

Para sacar *eventos[i]* y ¬*eventos[i]*, agrego la fórmula ∧ (*eventos[i]* ∨ ¬*eventos[i]*)

$((eventos[i] \ \wedge_L$
$res =$
$recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i,eventos),F)})$
$\vee$
$(\neg eventos[i] \ \wedge_L$
$res * apuesta_s * pago_s =$
$recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i,eventos),F)})) \wedge$
$(eventos[i] \vee \neg eventos[i])$

Distribuimos y queda:

$(eventos[i] \ \wedge_L$
$res = recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i,eventos),F)} \wedge$
$(eventos[i] \vee \neg eventos[i]))$

$\vee$

$(\neg eventos[i] \ \wedge_L res * apuesta_s * pago_s$
$=$
$\neg eventos[i] \wedge_L recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i,eventos),F)} \ \wedge$
$(eventos[i] \vee \neg eventos[i]))$

Ahora agrupamos:
$(eventos[i] \wedge (eventos[i] \vee \neg eventos[i])) \ \equiv \ (\neg eventos[i] \wedge (eventos[i] \vee \neg eventos[i])) \ \equiv \ True$

Llegamos a lo siguiente para ambas proposiciones:

$True \wedge res = recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i,eventos),F)})$

Y como ya habíamos visto que el rango de *i* se implicaba, se sucede que esta proposición es igual al invariante: concluimos, entonces, que este punto de correctitud se cumple.

Uniendo todo de nuevo, podemos ver que la implicación es verdadera:

$(0 \leq i < |eventos| \ \wedge_L$
$res = recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i,eventos),F)})$

$\Rightarrow$

$(0 \leq i < |eventos| \ \wedge_L$
$res = recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i,eventos),F)})$

$\equiv$

True □

2.7. Sentencia IV

¿$\{I \wedge B \wedge fv = v_0\} \ S \ \{fv < v_0\}$?

Usamos que fv = |eventos| - i: se entra al cuerpo del ciclo |eventos| veces, ya que estoy recorriendo todo el arreglo de eventos e incrementando el valor del iterador en uno luego de cada ciclo. Cuando i sea |eventos| + 1, la función variante elegida será negativa: es válida, entonces, para realizar la prueba de terminación de ciclo.

Este es el primer ítem para verificar la terminación del programa.

Queremos ver que $I \wedge$ B $\wedge$ fv < $v_0 \Rightarrow$ wp(if B then S1 else S2, i = i + 1, fv < v_0)

Sea K = wp(i = i + 1, |eventos| - i).
Obtenemos la *weakest precondition* de K utilizando el axioma 1:

K $\equiv def(i+1) \wedge_L |eventos| - (i+1) \equiv True \wedge_L |eventos| - i - 1 < \mathrm{v}_0$

Ahora pasamos a la siguiente etapa: wp(if B then S1 else S2, K)

$def(B_{if}) \ \wedge_L ((B_{if} \wedge \ wp(S1, K)) \vee (\neg B_{if} \wedge \ wp(S2, K))$

$\equiv$

$(0 \leq i < |eventos| \ \wedge_L ((eventos[i] \wedge |eventos| - i - 1 < v_0) \vee (\neg eventos[i] \wedge |eventos| - i - 1 < v_0)))$

Agregamos una formula extra que no afecta la lógica (¡mismas tablas de verdad!) para reducir la proposición: $\wedge (eventos[i] \vee \neg eventos[i])$

$((0 \leq i < |eventos| \ \wedge_L ((eventos[i] \wedge |eventos| - i - 1 < v_0) \vee (\neg eventos[i] \wedge |eventos| - i - 1 < v_0))) \wedge (eventos[i] \vee \neg eventos[i]))$

$\equiv$

$(0 \leq i < |eventos| \ \wedge_L (eventos[i] \wedge |eventos| - i - 1 < v_0) \wedge (eventos[i] \vee \neg eventos[i]))$

Ahora agrupamos:

$(eventos[i] \wedge (eventos[i] \vee \neg eventos[i])) \quad \equiv \quad (\neg eventos[i] \wedge (eventos[i] \vee \neg eventos[i])) \quad \equiv \quad True$

Luego, al aplicar la agrupación en ambos lados, terminamos con lo siguiente:

$\equiv$ (0 $\leq i < |eventos| \ \wedge_L \ |eventos| - i - 1 < v_0)$

Sea $I \wedge B_{ciclo} \wedge fv = v_0$:

0 $\leq i \leq |eventos| \ \wedge_L \ i < |eventos| \wedge$
$res = recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i,eventos),F)} \wedge$
$|eventos| - i = v_0$

Ajustamos el rango de *i* a partir de los *Y*:

$0 \leq i < |eventos| \ \wedge_L$
$res = recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),T)} * (apuesta_s, pago_s)^{\#(subSeq(0,i,eventos),F)} \wedge$
$|eventos| - i = v_0$

Queremos ver que la línea anterior implica $(0 \leq i < |eventos| \ \wedge_L \ |eventos| - i - 1 < v_0)$.
Separando por partes, y asumiendo como verdadero el antecedente:

$0 \leq i < |eventos| \ \Rightarrow \ 0 \leq i < |eventos|$
$|eventos| - i - 1 < |eventos| - i \ \equiv \ -1 < 0 \ \equiv$ *True*

$\square$

2.8. Sentencia V

¿$I \wedge fv \leq 0 \;\;\Rightarrow\;\; \neg B$?

Usamos que fv = |eventos| - i, al igual que en el punto **IV**. Queremos ver que si la función variante es negativa y el invariante es verdadero, entonces no se cumple la guarda.

Este es el segundo ítem para verificar la terminación del programa. Dado que no se hace referencia a res en el consecuente, se sucede que el antecedente implica *True* para esa parte; queda ver qué sucede con i :

$I \;\wedge\; fv \leq 0$

$\equiv$

$0 \leq \; i \; \leq \; |eventos| \;\wedge_L$
$res = recurso * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),T)} * (apuesta_c, pago_c)^{\#(subSeq(0,i,eventos),F)}$
$\wedge \; |eventos| - i \leq 0 \;\Rightarrow\; i \geq |eventos|$

$\equiv$

$0 \;\leq\; i \;\leq\; |eventos| \;\wedge_L\; |eventos| \leq i \;\equiv\; i = |eventos| \;\Rightarrow\; i \geq |eventos|$

$\equiv$

$|eventos| \geq |eventos| \;\Leftrightarrow\; \frac{|eventos|}{|eventos|} \geq 1 \;\Leftrightarrow\; 1 \geq 1 \;\equiv\; True$

$\Rightarrow$

$\neg B$

Probamos que el programa es correcto respecto a su especificación: es decir que dada una pre-condición, entonces el programa termina y cumple la post condición.

$\square$