July 23, 2025

TAD $BerretaCoin {

mirojoRGB200,20,20 azulRGB100, 160, 255 verdeRGB120, 200, 120 amarilloRGB204, 160, 0

transaccion = struct $< idTransaccion : \text{Z}, idComprador : \text{Z}, idVendedor : \text{Z}, monto : \text{Z} >$

bloque = $tuple < (\text{Z}, seq < transaccion >) >$

Obs cad : $seq < (\text{Z}, seq < transaccion >) >$

1 Procesos

Proc nuevo$Berretacoin() : $BerretaCoin {
Requiere { True }
Asegura { res.cad = $\langle\rangle$ }
}

Proc agregarBloque (in b : bloque ; inout bc : $BerretaCoin) {
Requiere { bc = bc_0}
Requiere {cadenaValida(bc, b) $\wedge bloqueValido(b)$}
Requiere { ($\forall i, j : \text{Z}) verde(0 \leq i < |bc.cad| \Rightarrow_L 0 \leq j < |bc.cad[i]_1| \Rightarrow_L mirojo(Gastobloque(bc, b, bc.cad[i]_1[j].idVendedor) \wedge Gastobloque(bc, b, bc.cad[i]_1[j].idComprado$
Asegura { bc.cad = concat($bc_0.cad$, b) }
}

Proc maximosTenedores (in bc : $BerretaCoin) : Seq< Z > {
Requiere { $|bc.cad| > 0$ }
Asegura { $(\forall i, j : \mathrm{Z})((0 \leq j < |res| \land 0 \leq i < |res| \land i \neq j) \Rightarrow_L res[i] \neq res[j])$}
Asegura { $(\forall i, j : \mathrm{Z}) verde(0 \leq i < |bc.cad| \Rightarrow_L 0 \leq j < |bc.cad[i]_1| \Rightarrow_L amarillo(bc.cad[i]_1[j].idVendedor \in res \leftrightarrow$
azul(($\forall n, m : Z) mirojo(0 \leq n < |bc.cad| \land 0 \leq m < |bc.cad[i]_1| \Rightarrow_L$ montoFinal($bc.cad[i]_1[j].idVendedor, bc$)
$\geq$ montoFinal($bc.cad[n]_1[m].idVendedor, bc) mirojo) azul) amarillo) verde)$}
}

2 Predicados y Auxiliares

Aux compraTotal(id : Z, bc : $BerretaCoin): Z
$=\{ \sum_{j=0}^{(|bc.cad|-1)} \sum_{i=0}^{(|bc.cad[j]_1|)}$ ifThenElse(bc.cad$[j]_1[i].idVendedor =$ id, bc.cad$[j]_1[i].monto, 0)\}$

Aux ventaTotal(id : Z, bc : $BerretaCoin): Z
$= \{ \sum_{j=0}^{(|bc.cad|-1)} \sum_{i=0}^{(|bc.cad[j]_1|)}$ ifThenElse(bc.cad$[j]_1[i].idComprador =$ id, bc.cad$[j]_1[i].monto, 0)\}$

Aux montoFinal(id: Z, bc : $Berretacoin) : Z
$=$ {compraTotal(id, bc) - ventaTotal(id, bc) }

Aux montoTotal(bc : $BerretaCoin): Z
$=\sum_{i=0}^{|bc.cad|-1} \sum_{j=0}^{(|bc.cad[j]_1|-1)}$ ifThenElse($bc.cad[i]_1[j].idComprador \neq 0, bc.cad[i]_1[j].mont$

Aux transaccionesTotal(bc : $BerretaCoin): Z
$=\sum_{i=0}^{|bc.cad|-1} \sum_{j=0}^{(|bc.cad[j]_1|-1)}$ ifThenElse($bc.cad[i]_1[j].idComprador \neq 0, 1, 0$)

Pred transaccionValida(t : transaccion) {
t.idTransaccion $\geq 0 \land t.idComprador \geq 0 \land t.idVendedor \geq 1 \land t.monto \geq 1 \land t.idComprador$
$\neq t.idVendedor$
}

Pred Gastobloque(bc : $BerretaCoin ; b: bloque, id: Z){

$(\forall i : Z)(0 \leq i < |b|_1 \Rightarrow_L montoFinal(bc, id) +$
$\sum_{j=0}^{i} \text{ifThenElse}(\text{b}_1[j].idVendedor, \text{b}_1[j].monto, 0) \geq$
$\sum_{j=0}^{i} \text{ifThenElse}(\text{b}_1[j].idComprador, \text{b}_1[j].monto, 0)$
}

Pred bloqueValido(b : bloque) {
$\wedge(\forall m : Z)(0 < m < |b_1|) \Rightarrow_L \text{b}_1[m-1].idTransaccion < \text{b}_1[m].idTransaccion$
$\wedge(\forall n : Z)(0 \leq n < |b_1| \Rightarrow_L \text{transaccionValida}(b_1[n]))$
$\wedge((\forall j : Z)(0 < j < |b_1| \Rightarrow_L b_1[j].idComprador \geq 1))$
$\wedge(b_0 \geq 0)$
}

Pred cadenaValida(bc : $BerretaCoin, - b: bloque) {
$(\ 0 < |b_1| \leq 50) \wedge_L$
$(|bc.cad| > 3000 \Rightarrow b_1[0].idComprador \geq 1) \wedge$
$(|bc.cad| \leq 3000 \Rightarrow azul(b_1[0].idComprador = 0 \wedge b_1[0].monto = 1azul) \wedge$
$(\forall i : Z)verde(mirojo(0 \leq i < |bc.cad| \Rightarrow_L b_1[0].idVendedor \neq bc.cad[i]_1[0].idVendedormirojo) \wedge$
$(\forall j : Z)amarillo(0 \leq j < |bc.cad| \Rightarrow_L (i \neq j \Rightarrow bc.cad[i]_1[0].idVendedor \neq bc.cad[j]_1[0].idVendedor)amarillo)verde) \wedge$
$(0 < |bc.cad| \Rightarrow_L bc.cad[|bc.cad| - 1]_0 < b_0)$
}

3 Comentarios

3.1 Sobre los procesos

Proc agregarBloque

- Tercer requiere: Para cada id, verificar su input en gasto bloque
- Asegura: Uso el comando concat para expresar que el bloque b, se agrega a la cadena inicial

Proc maximoTenedores

- Requiere: La lista no tiene que ser vacia pues no existe un maximo en un dato vacio
- Primer asegura: La id no se repite
- Segundo asegura: Una id esta en la lista sii, para cada id que podamos elegir, la primera id tiene mayor o igual cantidad de monedas

Proc montoMedio

- Primer requiere: Una lista vacia no puede tener un promedio
- Segundo requiere: Verifica que haya una transaccion que no sea de creacion, pues estas no se tienen en cuenta

3.2 Sobre los predicados y auxiliares

Pred GastoBloque

- Este predicado busca que ningun usuario gaste mas monedas de las que tiene
- Agrego el cuantificador para no solo verificar al final de la lista, sino para verificarlo en cada posicion
- Primero con monto final sumo las monedas que se tenian de bloques anteriores, y con la primera sumatoria, obtengo todas las monedas ganadas en este bloque

- Luego esto es comparado con la sumatoria de todo lo que se gasto en el bloque. Y debe ser igual o menor, pues solo se puede gastar lo que se tiene o menos

Pred bloqueValido

- Primer condicional: El bloque no es vacio y la cantidad de transacciones es menor a 50
- Segundo condicional: Todas las id de transaccion son distintas
- Tercer condicional: Una id es menor a la que le sigue
- Cuarto condicional: Para cada transaccion verifico transaccion valida
- Quinto condicional: Verifico que todo elemento que no sea el primero, sea mayor a 0, pues no hay id de creacion
- Sexto condicional: La id no puede ser negativa

Pred cadenaValido

- Primer condicional: El bloque no es vacio y la cantidad de transacciones es menor a 50
- Segundo condicional: Si la longitud de la cadena es menor o igual a 3000, entonces verifico que la primera id sea de creacion
- Tercer condicional: Si la longitud es mayor a 3000, entonces verifico que la primera id no sea de creacion
- Cuarto condicional: Verifico que las id del bloque esten ordenadas de menor a mayor