DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 1 - Especificación de TADs

Criptomonedas

Algoritmos y Estructura de Datos 2

Grupo Pythonisbetter

Integrante	LU	Correo electrónico
Perez Marzo, Jordan Alexis	738/24	jordanpm30@gmail.com
Nuñez Geronimo, Sebastian Javier	986/24	sebustos2394@gmail.com
Camacho Gomez, Lucero Belen	667/23	camacholu14@gmail.com
Suarez, Ricardo Javier	127/20	ricardojaviersuarezz@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. Definición de Tipos

type *Bloque* ES $tupla\langle \mathbb{Z}, seq\langle Transaccion\rangle\rangle$
type *idBloque* ES $\mathbb{Z}$
type *Transaccion* ES $struct\langle id : \mathbb{Z}, idC : \mathbb{Z}, idV : \mathbb{Z}, monto : \mathbb{Z}\rangle$
type *Usuario, Saldo* ES $\mathbb{Z}$
type *Cotizacion* ES $\mathbb{Z}$

2. Especificación

TAD $BerretaCoin {

obs $bloques = seq\langle Bloque\rangle$
obs $dineroUsuariosTotal = dict\langle Usuario, Saldo\rangle$

2.1. Ejercicio 1

`proc $BerretaCoinVacio` () : $ Berretacoin {
`asegura` $\{res.bloque = \langle\rangle\}$
`asegura` $\{res.dineroUsuarioTotal = \{\}\}$
}

`proc agregarBloque` (inout BCoin : $ Berretacoin , in b : Bloque) : {
`requiere` $\{BCoin = BCoin_0\}$
`requiere` $\{esBloqueValido(BCoin_0.bloques, BCoin_0.dineroUsuarioTotal, b)\}$
`asegura` $\{BCoin.bloques = BCoin_0.bloques + +\langle b\rangle\}$
`asegura` $\{seActualizoDineroUsuarios(BCoin_0.dineroUsuarioTotal, Bcoin.dineroUsuarioTotal, b)\}$
}

`pred esBloqueValido` (s: $seq\langle Bloque\rangle$,d: dict$\langle Usuario, Saldo\rangle, b : Bloque$){
$0 < |b_1| \leq 50 \wedge cuentaConTransaccionEscpecial(s, d, b) \wedge nadieGastaMasDeLoQueTiene(d, b) \wedge$
$esBloqueConsecutivoConLosDemasPorId(s, b)$
}

`pred cuentaConTransaccionEspecial` (s:$seq\langle Bloque\rangle$,d: dict$\langle Usuario, Saldo\rangle, b : Bloque$){
$(|s| < 3000 \longrightarrow (bloqueValidoConTransaccionEspecial(b) \wedge vendedorDeCreacionSiempreDistinto(s, b))) \wedge$
$(|s| \geq 3000 \longrightarrow bloqueValidoSinTransaccionEspecial(b))$
}

`pred bloqueValidoConTransaccionEspecial` (b: Bloque) {
$(b_1[0].idC = 0 \wedge b_1[0].idV > 0 \wedge b_1[0].monto = 1 \wedge b_1[0].id > 0) \wedge (\forall i : Z)(1 \leq i < |b_1| \longrightarrow_L$
$idTransaccionYUsuariosYMontosValidos(b_1[i])) \wedge tieneTransaccionesConsecutivas(b)$
}
`pred vendedorDeCreacionSiempreDistinto` (s:$seq\langle Bloque\rangle$,b: Bloque) {
$(\forall i : Z)(0 \leq i < |s| \longrightarrow_L s[i]_1[0].idV \neq b_1[0].idV)$
}
`pred bloqueValidoSinTransaccionEspecial` (b: Bloque) {
$(\forall i : Z)(0 \leq i < |b_1| \longrightarrow_L idTransaccionYUsuariosYMontosValidos(b_1[i])) \wedge tieneTransaccionesConsecutivas(b)$
}
`pred idTransaccionYUsuariosYMontosValidos` (t: Transaccion) {
$t.idC > 0 \wedge_L t.idV > 0 \wedge_L t.idC \neq t.idV \wedge_L t.monto > 0 \wedge_L t.id > 0)$
}
`pred tieneTransaccionesConsecutivas` (b: Bloque) {
$(\forall j : Z)(0 \leq j < |b_1| - 1) \longrightarrow_L b_1[j].id < b_1[j + 1].id)$
}

`pred nadieGastaMasDeLoQueTiene` (d: dict$\langle Usuario, Saldo\rangle, b : Bloque$){
$(\forall i : \mathbb{Z})(0 \leq i < |b_1| \longrightarrow_L IfThenElseFi(j \notin d, saldoNeto(subseq(b_1, 0, i + 1), j), d[j] +$
$saldoNeto(subseq(b_1, 0, i + 1), j)) \geq 0)$
}

`aux saldoNeto` (t:$seq\langle Transaccion\rangle$, u: Usuario) : Z $= montoVendido(t, u) - montoComprado(t, u)$;
`aux montoComprado` (t:$seq\langle Transaccion\rangle$, u: Usuario) : Z $=$
$\sum_{i=0}^{|t|-1}$ `IfThenElse`$(t[i].idC = u, t[i].monto, 0)$;

aux montoVendido (t:$seq\langle Transaccion\rangle$, u: Usuario) : Z =
$\sum_{i=0}^{|t|-1}$ IfThenElse$(t[i].idV = u, t[i].monto, 0)$;

pred esBloqueConsecutivoConLosDemasPorId (s:$seq\langle Bloque\rangle$,b: Bloque) {
$|s| > 0 \wedge s[|s|-1]_0 = b_0 - 1$
}

pred seActualizoDineroUsuarios ($d_0 : dict\langle Usuario, Saldo\rangle, d : \langle Usuario, Saldo\rangle, b : Bloque$){
$siEsUsuarioNuevoOExistente(d_0, d, b) \wedge siEsUsuarioExistentePeroNoEstaEnElBloque(d_0, d, b) \wedge$
$siNoEsUsuarioNoEsta(d_0, d, b)$
}
pred siEsUsuarioNuevoOExistente ($d_0 : dict\langle Usuario, Saldo\rangle, d : \langle Usuario, Saldo\rangle, b : Bloque$){
$(\forall j : Z)((esUsuarioDelBloque(j, b)) \longrightarrow_L d[j] =$ IfThenElse$(j \notin d_0, saldoNeto(b_1, j), d_0[j] + saldoNeto(b_1, j))$
}
pred siEsUsuarioExistentePeroNoEstaEnElBloque ($d_0 : dict\langle Usuario, Saldo\rangle, d : \langle Usuario, Saldo\rangle, b : Bloque$){
$(\forall j : Z)((j \in d_0 \wedge \neg esUsuarioDelBloque(j, b)) \longrightarrow_L d[j] = d_0[j]))$
}
pred siNoEsUsuarioNoEsta ($d_0 : dict\langle Usuario, Saldo\rangle, d : \langle Usuario, Saldo\rangle, b : Bloque$){
$(\forall j : Z)((j \notin d_0 \wedge \neg esUsuarioDelBloque(j, b)) \longrightarrow_L j \notin d)$
}
pred esUsuarioDelBloque (u: Usuario , b: Bloque) {
$(\exists i : Z)(0 \leq i < |b_1| \wedge_L (b_{1[i]}.idC = u \vee b_{1[i]}.idV = u))$
}

2.2. Ejercicio 2

proc maximoTenedores (in BCoin : $BerretaCoin) : $seq\langle\mathbb{Z}\rangle${
requiere $\{True\}$
asegura $\{(\forall i : Usuario)\ (0 \leq i < |res| \longrightarrow_L usuarioConMasBCoin(res[i], BCoin.dineroUsuarioTotal))\}$
asegura $\{noHayRepetidos(res)\}$
asegura $\{(\forall j : Usuario)(j \in Bcoin.dineroUsuarioTotal \wedge_L$
$usuarioConMasBCoin(j, BCoin.dinerousuarioTotal)) \longrightarrow_L j \in res\}$
}
pred usuarioConMasBCoin (u : Usuario, d : $dict\ \langle$Usuario, Saldo$\rangle$){
$(\forall j : Usuario)(j \in d \wedge u \in d \wedge j \neq u) \longrightarrow_L d[j] \leq d[u]$
}
pred noHayRepetidos (s:$seq\langle T\rangle$) {
$(\forall i : Z)(0 \leq i < |s| \longrightarrow_L \neg(\exists j : Z)((0 \leq j < |s| \wedge j \neq i) \wedge_L s[i] = s[j])$
}

2.3. Ejercicio 3

proc montoMedio (in BCoin : $ BerretaCoin) : R {
asegura $\{|BCoin.bloques| = 0 \longrightarrow res = 0\}$
asegura $\{|BCoin.bloques| > 0 \longrightarrow res = \frac{montoTotalTransacciones(BCoin.bloques)}{cantidadTransacciones(BCoin.bloques)}\}$
}
aux montoTotalTransacciones (s:$seq\langle bloque\rangle$) : R =
$\sum_{i=0}^{|s|-1} montoPorBloque(s[i])$;
aux cantidadTransacciones (s:$seq\langle$bloque$\rangle$) : R =
$\sum_{i=0}^{|s|-1}$ IfThenElse$(s[i]_{1_0}.idC = 0, |s[i]_1| - 1, |s[i]_1|)$;

2.4. Ejercicio 4

proc cotizacionAPesos (in BCoin : $ BerretaCoin , in c : $seq\langle Cotizacion\rangle$) : $seq\langle Z\rangle$\{

requiere $\{|BCoin.bloques| = |c|\}$

requiere $\{(\forall i : Z)(0 \leq i < |c| \longrightarrow_L c[i] > 0)\}$

asegura $\{|res| = |c|\}$

asegura $\{(\forall i : Z)(0 \leq i < |res|) \longrightarrow_L res[i] = montoPorBloque(BCoin.bloques[i]) * c[i]\}$

}

Auxiliar global:
aux montoPorBloque (b:bloque) : R =
$\sum_{i=0}^{|b|-1}$ IfThenElse$(b_1[j].idC \neq 0, b_1[j].monto, 0)$;