Algoritmos y Estructuras de Datos II

Trabajo Práctico 1

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Pacalgo2

Los inertes

Integrante	LU	Correo electrónico
Bruno Robbio	480/09	brobbio@hotmail.com
Nicolas Andres Kinaschuk	248/15	nicolaskinaschuk@gmail.com
Pedro Joel Burgos	804/18	facultadburgospedrojoel@hotmail.com
Valentina Madelaine Saravia Ruiz	257/18	valentina.saraviaruiz@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

1. Introducción

1.1. Consideraciones TP1

- No se puede arrancar el mapa ganando o perdiendo
- El mapa puede no tener un camino ganador o perdedor
- El vértice del mapa es la esquina inferior izquierda

1.2. Consideraciones TP2

- Puede haber más de un jugador en una posición del ranking
- Si el jugador actual está en el primer puesto, el contrincante a superar puede ser él mismo
- No todos los jugadores juegan el mismo mapa

2. Desarrollo

2.1. TP 1

TAD CASILLERO

extiende Tupla(int, int)

usa Int

géneros casillero

exporta casillero, +, -, aDistanciaMenosDeN

otras operaciones

- $\bullet + \bullet$: casillero $\times$ casillero $\longrightarrow$ casillero
- $\bullet$ - $\bullet$: casillero $\times$ casillero $\longrightarrow$ casillero

aDistanciaMenosDeN : casillero $\times$ nat $\longrightarrow$ conj(casillero)

axiomas

$\pi_1(c1 + c2) \equiv \pi_1(c_1) + \pi_1(c_2)$

$\pi_2(c1 + c2) \equiv \pi_2(c_1) + \pi_2(c_2)$

$\pi_1(c1 - c2) \equiv \pi_1(c_1) - \pi_1(c_2)$

$\pi_2(c1 - c2) \equiv \pi_2(c_1) - \pi_2(c_2)$

aDistanciaMenosDeN(c, n) $\equiv$ **if** n=0? **then**
$\{c\}$
else
(aDistanciaMenosDeN(c + $\langle$1,0$\rangle$, $n-1$) $\cup$
aDistanciaMenosDeN(c - $\langle$1,0$\rangle$, $n-1$) $\cup$
aDistanciaMenosDeN(c + $\langle$0,1$\rangle$, $n-1$) $\cup$
aDistanciaMenosDeN(c - $\langle$0,1$\rangle$, $n-1$) $\cup$
aDistanciaMenosDeN(c, $n-1$))
fi

Fin TAD

TAD MAPA

usa Casillero, Conjunto, Bool

géneros mapa

exporta mapa, observadores, generadores, casillerosLibres

igualdad observacional

$$(\forall m_1, m_2 : \text{mapa}) \left(m_1 =_{\text{obs}} m_2 \iff \begin{pmatrix} \text{conjFantasmas}(m_1) =_{\text{obs}} \text{conjFantasmas}(m_2) \land \\ \text{conjParedes}(m_1) =_{\text{obs}} \text{conjParedes}(m_2) \land \\ \text{conjChocolates}(m_1) =_{\text{obs}} \text{conjChocolates}(m_2) \land \\ \text{dimensiones}(m_1) =_{\text{obs}} \text{dimensiones}(m_2) \land \\ \text{casilleroInicial}(m_1) =_{\text{obs}} \text{casilleroInicial}(m_2) \land \\ \text{vértice}(m_1) =_{\text{obs}} \text{vértice}(m_2) \land \\ \text{casilleroDeLlegada}(m_1) =_{\text{obs}} \text{casilleroDeLlegada}(m_2) \end{pmatrix} \right)$$

observadores básicos

conjFantasmas : mapa $\longrightarrow$ conj(casillero)

conjParedes : mapa $\longrightarrow$ conj(casillero)

conjChocolates : mapa $\longrightarrow$ conj(casillero)

dimensiones : mapa $\longrightarrow$ tupla(nat,nat)

vértice : mapa $\longrightarrow$ casillero

casilleroInicial : mapa $\longrightarrow$ casillero

casilleroDeLlegada : mapa $\longrightarrow$ casillero

generadores

nuevoMapa : tupla(nat, nat) *dimensión* $\times$ casillero *vértice* $\times$ casillero *inicio* $\times$ casillero fin $\times$
conj(casillero) $fantasmas$ $\times$ conj(casillero) *paredes* $\times$ conj(casillero) *chocolates* $\longrightarrow$ mapa

$$\left\{ \begin{array}{l} \emptyset?(fantasmas \cap paredes) \ \land \emptyset?(chocolates \cap paredes) \land \\ \emptyset?(fantasmas \cap chocolates) \ \land (inicio \neq fin) \ \land \\ dentroDeLasDimensiones(dimensión, vértice, inicio) \ \land \\ dentroDeLasDimensiones(dimensión, vértice, fin) \ \land \\ (\forall f \in fantasmas)(dentroDeLasDimensiones(dimensión, vértice, f) \ \land \\ \quad (f \notin aDistanciaMenosDeN(inicio, f, 3) \lor (inicio \in chocolates))) \ \land \\ (\forall c \in chocolates)(dentroDeLasDimensiones(dimensión, vértice, c)) \ \land \\ (\forall p \in paredes)(dentroDeLasDimensiones(dimensión, vértice, p)) \end{array} \right\}$$

otras operaciones

casillerosLibres : mapa $\longrightarrow$ conj(casillero)

dentroDeLasDimensiones : tupla(int $\times$ int) $\times$ casillero $\times$ casillero $\longrightarrow$ bool

casillerosADerecha : casillero c $\times$ mapa m $\longrightarrow$ conj(casillero)
{dentroDeLasDimensiones(dimensión(m), vértice(m), c)}

casillerosIzquierda : casillero c $\times$ mapa m $\longrightarrow$ conj(casillero)
{dentroDeLasDimensiones(dimensión(m), vértice(m), c)}

casillerosHorizontales : casillero c $\times$ mapa m $\longrightarrow$ conj(casillero)
{dentroDeLasDimensiones(dimensión(m), vértice(m), c)}

casillerosArriba : casillero c $\times$ mapa m $\longrightarrow$ conj(casillero)
{dentroDeLasDimensiones(dimensión(m), vértice(m), c)}

casillerosAbajo : casillero c × mapa m $\longrightarrow$ conj(casillero)
{dentroDeLasDimensiones(dimensión(m), vértice(m), c)}

casillerosVerticales : casillero c × mapa m $\longrightarrow$ conj(casillero)
{dentroDeLasDimensiones(dimensión(m), vértice(m), c)}

generarCasillerosHorizontales : mapa m × conj(casillero) C $\longrightarrow$ conj(casillero)
{($\forall$ $c \in C$)(dentroDeLasDimensiones(dimensión(m), vértice(m), c))}

axiomas

conjFantasmas(nuevoMapa(*dimensión*, *vértice*, *inicio*, *fin*, *fantasmas*, *paredes*, *chocolates*)) $\equiv$ *fantasmas*

conjParedes(nuevoMapa(*dimensión*, *vértice*, *inicio*, *fin*, *fantasmas*, *paredes*, *chocolates*)) $\equiv$ *paredes*

conjChocolates(nuevoMapa(*dimensión*, *vértice*, *inicio*, *fin*, *fantasmas*, *paredes*, *chocolates*)) $\equiv$ *chocolates*

dimensiones(nuevoMapa(*dimensión*, *vértice*, *inicio*, *fin*, *fantasmas*, *paredes*, *chocolates*)) $\equiv$ *dimensión*

casilleroInicial(nuevoMapa(*dimensión*, *vértice*, *inicio*, *fin*, *fantasmas*, *paredes*, *chocolates*)) $\equiv$ *inicio*

casilleroDeLlegada(nuevoMapa(*dimensión*, *vértice*, *inicio*, *fin*, *fantasmas*, *paredes*, *chocolates*)) $\equiv$ *fin*

vértice(nuevoMapa(*dimensión*, *vértice*, *inicio*, *fin*, *fantasmas*, *paredes*, *chocolates*)) $\equiv$ *vértice*

dentroDeLasDimensiones(*dimensión*, *vértice*, *casilla*) $\equiv$ $(0 \leq \pi_1(casilla) - \pi_1(vértice) < \pi_1(dimensión)) \wedge$
$(0 \leq \pi_2(casilla) - \pi_2(vértice) < \pi_2(dimensión))$

casillerosLibres(m) $\equiv$ generarCasillerosHorizontales(m,casillerosVerticales(vértice(m),m))−(conjFantasmas(m) ∪ conjParedes(m))

generarCasillerosHorizontales(m, casilleros) $\equiv$ **if** casilleros = $\emptyset$ **then**
$\emptyset$
else
casillerosHorizontales(dameUno(casilleros, m))∪
generarCasillerosHorizontales(m, sinUno(casilleros))
fi

casillerosHorizontales(c,m) $\equiv$ casillerosADerecha(c,m) ∪ casillerosAIzquierda(c, m)

casillerosADerecha(c,m) $\equiv$ (**if** dentroDeLasDimensiones(dimensión(m), vértice(m), $c + \langle 1, 0\rangle$) **then**
casillerosADerecha(c+$\langle 1, 0\rangle$, m)
else
$\emptyset$
fi)∪{c}

casillerosAIzquierda(c,m) $\equiv$ (**if** dentroDeLasDimensiones(dimensión(m), vértice(m), $c - \langle 1, 0\rangle$) **then**
casillerosAIzquierda(c−$\langle 1, 0\rangle$, m)
else
$\emptyset$
fi)∪{c}

casillerosVerticales(c,m) $\equiv$ casillerosArriba(c,m)∪casillerosAbajo(c, m)

casillerosArriba(c,m) $\equiv$ (**if** dentroDeLasDimensiones(dimensión(m), vértice(m), $c + \langle 0, 1\rangle$) **then**
casillerosArriba(c+$\langle 0, 1\rangle$, m)
else
$\emptyset$
fi)∪{c}

casillerosAbajo(c,m) $\equiv$ (**if** dentroDeLasDimensiones(dimensión(m), vértice(m), $c - \langle 0, 1\rangle$) **then**
casillerosAbajo(c−$\langle 0, 1\rangle$, m)
else
$\emptyset$
fi)∪{c}

Fin TAD

TAD PACALGO2

usa Mapa

géneros pacalgo2

exporta pacalgo2, observadores, generadores, direccionesPosibles, perdió?, ganó?

igualdad observacional

$$(\forall p_1, p_2 : \text{pacalgo2}) \left(p_1 =_{\text{obs}} p_2 \iff \begin{pmatrix} \text{verMapa}(p_1) =_{\text{obs}} \text{verMapa}(p_2) \wedge \\ \text{posiciónActual}(p_1) =_{\text{obs}} \text{posiciónActual}(p_2) \wedge \\ \text{nivelDeChocolate}(p_1) =_{\text{obs}} \text{nivelDeChocolate}(p_2) \wedge \\ \text{chocolatesRestantes}(p_1) =_{\text{obs}} \text{chocolatesRestantes}(p_2) \wedge \\ \text{ganó?}(p_1) =_{\text{obs}} \text{ganó?}(p_2) \wedge \\ (\text{ganó?}(p_1) \Rightarrow_L \text{puntaje}(p_1) =_{\text{obs}} \text{puntaje}(p_2)) \end{pmatrix} \right)$$

observadores básicos

verMapa	: pacalgo2	$\longrightarrow$	mapa	
posiciónActual	: pacalgo2	$\longrightarrow$	casillero	
puntaje	: pacalgo2	$\longrightarrow$	nat	{ganó?(p)}
nivelDeChocolate	: pacalgo2	$\longrightarrow$	nat	
chocolatesRestantes	: pacalgo2	$\longrightarrow$	conj(casillero)	

generadores

inicializarJuego	: mapa	$\longrightarrow$	pacalgo2
arriba	: pacalgo2 p	$\longrightarrow$	pacalgo2 $\{(\text{posiciónActual}(p) + \langle 0,1 \rangle) \in \text{direccionesPosibles}(p) \wedge \neg\text{ganó?}(p) \wedge \neg\text{perdió?}(p)\}$
abajo	: pacalgo2 p	$\longrightarrow$	pacalgo2 $\{(\text{posiciónActual}(p) - \langle 0,1 \rangle) \in \text{direccionesPosibles}(p) \wedge \neg\text{ganó?}(p) \wedge \neg\text{perdió?}(p)\}$
derecha	: pacalgo2 p	$\longrightarrow$	pacalgo2 $\{(\text{posiciónActual}(p) + \langle 1,0 \rangle) \in \text{direccionesPosibles}(p) \wedge \neg\text{ganó?}(p) \wedge \neg\text{perdió?}(p)\}$
izquierda	: pacalgo2 p	$\longrightarrow$	pacalgo2 $\{(\text{posiciónActual}(p) - \langle 1,0 \rangle) \in \text{direccionesPosibles}(p) \wedge \neg\text{ganó?}(p) \wedge \neg\text{perdió?}(p)\}$

otras operaciones

direccionesPosibles	: pacalgo2	$\longrightarrow$	conj(casillero)
perdió?	: pacalgo2	$\longrightarrow$	bool
ganó?	: pacalgo2	$\longrightarrow$	bool
pasos	: pacalgo2	$\longrightarrow$	nat

axiomas

verMapa(inicializarJuego(m))	$\equiv$	m
verMapa(arriba(p))	$\equiv$	verMapa(p)
verMapa(abajo(p))	$\equiv$	verMapa(p)
verMapa(izquierda(p))	$\equiv$	verMapa(p)
verMapa(derecha(p))	$\equiv$	verMapa(p)
posiciónActual(inicializarJuego m)	$\equiv$	casilleroInicial(m)
posiciónActual(arriba(p))	$\equiv$	posiciónActual(p) + $\langle 0,1 \rangle$
posiciónActual(abajo(p))	$\equiv$	posiciónActual(p) - $\langle 0,1 \rangle$

posiciónActual(izquierda(p)) $\equiv$ posiciónActual(p) - $\langle 1, 0 \rangle$

posiciónActual(derecha(p)) $\equiv$ posiciónActual(p) + $\langle 1, 0 \rangle$

pasos(inicializarJuego(m)) $\equiv$ 0

pasos(arriba(p)) $\equiv$ 1 + pasos(p)

pasos(abajo(p)) $\equiv$ 1 + pasos(p)

pasos(izquierda(p)) $\equiv$ 1 + pasos(p)

pasos(derecha(p)) $\equiv$ 1 + pasos(p)

puntaje(p) $\equiv$ pasos(p)

nivelDeChocolate(inicializarJuego(m)) $\equiv$ **if** casilleroInicial(m) $\in$ conjChocolates(m) **then**
10
else
0
fi

nivelDeChocolate(arriba(p)) $\equiv$ **if** (posiciónActual(p) + $\langle 0, 1 \rangle$) $\in$ chocolatesRestantes(p) **then**
10
else
if 0?(nivelDeChocolate(p)) **then**
nivelDeChocolate(p)
else
nivelDeChocolate(p) - 1
fi
fi

nivelDeChocolate(abajo(p)) $\equiv$ **if** (posiciónActual(p) - $\langle 0, 1 \rangle$) $\in$ chocolatesRestantes(p) **then**
10
else
if 0?(nivelDeChocolate(p)) **then**
nivelDeChocolate(p)
else
nivelDeChocolate(p) - 1
fi
fi

nivelDeChocolate(izquierda(p)) $\equiv$ **if** (posiciónActual(p) - $\langle 1, 0 \rangle$) $\in$ chocolatesRestantes(p) **then**
10
else
if 0?(nivelDeChocolate(p)) **then**
nivelDeChocolate(p)
else
nivelDeChocolate(p) - 1
fi
fi

nivelDeChocolate(derecha(p)) $\equiv$ **if** (posiciónActual(p) + $\langle 1, 0 \rangle$) $\in$ chocolatesRestantes(p) **then**
10
else
if 0?(nivelDeChocolate(p)) **then**
nivelDeChocolate(p)
else
nivelDeChocolate(p) - 1
fi
fi

chocolatesRestantes(inicializarJuego(m)) $\equiv$ **if** casilleroInicial(m) $\in$ conjChocolates(m) **then**
conjChocolates(m) - { casilleroInicial(m) }
else
conjChocolates(m)
fi

chocolatesRestantes(arriba(p)) $\equiv$ **if** (posiciónActual(p) + $\langle 0,1 \rangle$) $\in$ chocolatesRestantes(p) **then**
chocolatesRestantes(p) - { (posiciónActual(p) + $\langle 0,1 \rangle$) }
else
chocolatesRestantes(p)
fi

chocolatesRestantes(abajo(p)) $\equiv$ **if** (posiciónActual(p) - $\langle 0,1 \rangle$) $\in$ chocolatesRestantes(p) **then**
chocolatesRestantes(p) - { (posiciónActual(p) - $\langle 0,1 \rangle$) }
else
chocolatesRestantes(p)
fi

chocolatesRestantes(izquierda(p)) $\equiv$ **if** (posiciónActual(p) - $\langle 1,0 \rangle$) $\in$ chocolatesRestantes(p) **then**
chocolatesRestantes(p) - { (posiciónActual(p) - $\langle 1,0 \rangle$) }
else
chocolatesRestantes(p)
fi

chocolatesRestantes(derecha(p)) $\equiv$ **if** (posiciónActual(p) + $\langle 1,0 \rangle$) $\in$ chocolatesRestantes(p) **then**
chocolatesRestantes(p) - { (posiciónActual(p) + $\langle 1,0 \rangle$) }
else
chocolatesRestantes(p)
fi

perdió?(p) $\equiv$ $\neg\emptyset$?(conjFantasmas(verMapa(p)) $\cap$
aDistanciaMenosDeN(posiciónActual(p),3))$\wedge$
0?(nivelDeChocolate(p))

ganó?(p) $\equiv$ posiciónActual(p)=casilleroDeLlegada(verMapa(p))

direccionesPosibles(p) $\equiv$ **if** 0?(nivelDeChocolate(p)) **then**
(aDistanciaMenosDeN(posiciónActual(p),1)$-$posiciónActual(p))
$\cap$
casillerosLibres(verMapa(p))
else
(aDistanciaMenosDeN(posiciónActual(p),1)$-$posiciónActual(p))
$\cap$
(casillerosLibres(verMapa(p)) $\cup$ conjFantasmas(verMapa(p)))
fi

Fin TAD

2.2. TP 2

TAD persona es String

TAD RANKING

usa Nat, persona

géneros ranking

exporta observadores, generadores, contrincante

igualdad observacional

$(\forall r_1, r_2 : \text{ranking})\ (r_1 =_{\text{obs}} r_2 \iff (\text{ver}(r_1) =_{\text{obs}} \text{ver}(r_2)))$

observadores básicos

ver : ranking $\longrightarrow$ dicc(persona, nat)

generadores

iniciarRanking : $\longrightarrow$ ranking

cargarPuntaje : ranking $\times$ persona $\times$ nat $\longrightarrow$ ranking

otras operaciones

contrincante	: ranking r $\times$ persona j	$\longrightarrow$	tupla(persona, nat) {def?(j, ver(r))}
superioresInmediatos	: ranking r $\times$ persona j	$\longrightarrow$	conj(persona) {def?(j, ver(r))}
elMaximoEntreMenores[1]	: conj(nat) $conjunto$ $\times$ nat p	$\longrightarrow$	nat
todosLosPuntajes	: ranking r $\times$ conj(persona) $nombres$	$\longrightarrow$	conj(nat) {$nombres \subseteq$ claves(ver(r))}
jugadoresConPuntajeIgual	: ranking r $\times$ conj(persona) $nombres$ $\times$ nat	$\longrightarrow$	conj(persona) {$nombres \subseteq$ claves(ver(r))}
menoresQueP	: conj(nat) $\times$ nat	$\longrightarrow$	conj(nat)
máximoDeConjunto	: conj(nat) C	$\longrightarrow$	nat {$\neg\emptyset?(C)$}

axiomas

ver(iniciarRanking()) $\equiv$ vacío

ver(cargarPuntaje($ranking$, $nombre$, $puntaje$)) $\equiv$ **if** def?($nombre$, ver($ranking$)) **then**
  if $puntaje$ <obtener(ver(r), $nombre$) **then**
    definir(ver($ranking$), ($nombre$, $puntaje$))
  else
    ver($ranking$)
  fi
else
  definir(ver($ranking$), ($nombre$, $puntaje$))
fi

todosLosPuntajes(r) $\equiv$ **if** $\emptyset?$(ver(r)) **then**
  $\emptyset$
else
  Ag(obtener(ver(r), dameUno(claves(ver(r)))), todosLosPuntajes(r))
fi

[1] Si no hay un menor que p en el $conjunto$, devuelve p

jugadoresConPuntajeIgual(r, *nombres*, *puntaje*) ≡ **if** ∅?(*nombres*) ∨ ∅?(claves(ver(r))) **then**
∅
else
if obtener(ver(r), dameUno(*nombres*)) = *puntaje* **then**
Ag(dameUno(*nombres*), jugadorConPuntajeIgual(r, sinUno(*nombres*), *puntaje*))
else
jugadorConPuntajeIgual(r,sinUno(*nombres*),*puntaje*)
fi
fi

superioresInmediatos(r, j) ≡ jugadoresConPuntajeIgual(r,claves(ver(r)), elMaximoEntreMenores(todosLosPuntajes(r), obtener(ver(r),j)))

elMaximoEntreMenores(*puntajes*, p) ≡ **if** ∅?(menoresQueP(*puntajes*, p)) **then**
p
else
máximoDeConjunto(menoresQueP(*puntajes*, p))
fi

menoresQueP(*puntajes*, p) ≡ **if** ∅?(*puntajes*) **then**
∅
else
if dameUno(*puntajes*)<p **then**
Ag(dameUno(*puntajes*), menoresQueP(sinUno(*puntajes*), p))
else
menoresQueP(sinUno(*puntajes*), p)
fi
fi

máximoDeConjunto(C) ≡ **if** ∅?(sinUno(C)) **then**
dameUno(C)
else
máx(dameUno(C), máximoDeConjunto(sinUno(C)))
fi

contrincante(r,j) ≡ (dameUno(superioresInmediatos(r,j)), obtener(ver(r), dameUno(superioresInmediatos(r,j))))

Fin TAD

TAD FICHÍN

usa Ranking, Pacalgo2

géneros fichín

exporta observadores, generadores, otras operaciones

igualdad observacional

$$(\forall f_1, f_2 : \text{fichín}) \left(f_1 =_{\text{obs}} f_2 \iff \begin{pmatrix} \text{conocerRanking}(f_1) =_{\text{obs}} \text{conocerRanking}(f_2) \wedge \\ \text{estáLibre?}(f_1) =_{\text{obs}} \text{estáLibre?}(f_2) \wedge \\ (\neg\text{estáLibre?}(f_1) \Rightarrow_L \\ \text{verJuegoActual}(f_1) =_{\text{obs}} \text{verJuegoActual}(f_2) \wedge \\ \text{jugadorActual}(f_1) =_{\text{obs}} \text{jugadorActual}(f_2)) \end{pmatrix} \right)$$

observadores básicos

conocerRanking	: fichín	$\longrightarrow$	ranking	
verJuegoActual	: fichín f	$\longrightarrow$	pacalgo2	$\{\neg\text{estáLibre?}(f)\}$
jugadorActual	: fichín f	$\longrightarrow$	persona	$\{\neg\text{estáLibre?}(f)\}$

generadores

nuevoFichin	:	$\longrightarrow$	fichín	
nuevaPartida	: fichín f $\times$ pacalgo2 p $\times$ persona j	$\longrightarrow$	fichín	$\{\text{estáLibre?}(f)\}$
palancaArriba	: fichín f	$\longrightarrow$	fichín	$\left\{\begin{matrix}\neg\text{estáLibre?}(f) \wedge_L \\ (\text{posiciónActual}(\text{verJuegoActual}(f)) + \langle 0,1\rangle) \in \text{direccionesPosibles}(\text{verJuegoActual}(f))\end{matrix}\right\}$
palancaAbajo	: fichín f	$\longrightarrow$	fichín	$\left\{\begin{matrix}\neg\text{estáLibre?}(f) \wedge_L \\ (\text{posiciónActual}(\text{verJuegoActual}(f)) + \langle 0,-1\rangle) \in \text{direccionesPosibles}(\text{verJuegoActual}(f))\end{matrix}\right\}$
palancaDerecha	: fichín f	$\longrightarrow$	fichín	$\left\{\begin{matrix}\neg\text{estáLibre?}(f) \wedge_L \\ (\text{posiciónActual}(\text{verJuegoActual}(f)) + \langle 1,0\rangle) \in \text{direccionesPosibles}(\text{verJuegoActual}(f))\end{matrix}\right\}$
palancaIzquierda	: fichín f	$\longrightarrow$	fichín	$\left\{\begin{matrix}\neg\text{estáLibre?}(f) \wedge_L \\ (\text{posiciónActual}(\text{verJuegoActual}(f)) + \langle -1,0\rangle) \in \text{direccionesPosibles}(\text{verJuegoActual}(f))\end{matrix}\right\}$

otras operaciones

puntajeDeJugadorActual	: fichín f	$\longrightarrow$	nat $\{\neg\text{estálibre?}(f) \wedge_L \text{jugadorActual}(f) \in \text{claves}(\text{ver}(\text{conocerRanking}(f)))\}$
verContrincante	: fichín f	$\longrightarrow$	tupla(persona, nat) $\{\neg\text{estáLibre?}(f) \wedge_L \text{jugadorActual}(f) \in \text{claves}(\text{ver}(\text{conocerRanking}(f)))\}$
estáLibre?	: fichín	$\longrightarrow$	Bool
puntajeParaMejorar	: fichín f	$\longrightarrow$	nat $\{\neg\text{estálibre?}(f) \wedge_L \text{jugadorActual}(f) \in \text{claves}(\text{ver}(\text{conocerRanking}(f)))\}$

axiomas

estáLibre?(nuevoFichin())	$\equiv$	True
estáLibre?(nuevaPartida(f,p,j))	$\equiv$	False
estáLibre?(palancaArriba(f))	$\equiv$	ganó?(arriba(verJuegoActual(f)))$\vee$perdió?(arriba(verJuegoActual(f)))
estáLibre?(palancaAbajo(f))	$\equiv$	ganó?(abajo(verJuegoActual(f)))$\vee$perdió?(abajo(verJuegoActual(f)))
estáLibre?(palancaDerecha(f))	$\equiv$	ganó?(derecha(verJuegoActual(f)))$\vee$perdió?(derecha(verJuegoActual(f)))

estáLibre?(palancaIzquierda(f))	$\equiv$	ganó?(izquierda(verJuegoActual(f)))$\vee$perdió?(izquierda(verJuegoActual(f))
jugadorActual(nuevaPartida(f,p,j))	$\equiv$	j
jugadorActual(palancaArriba(f))	$\equiv$	jugadorActual(f)
jugadorActual(palancaAbajo(f))	$\equiv$	jugadorActual(f)
jugadorActual(palancaDerecha(f))	$\equiv$	jugadorActual(f)
jugadorActual(palancaIzquierda(f))	$\equiv$	jugadorActual(f)
conocerRanking(nuevoFichin())	$\equiv$	iniciarRanking()
conocerRanking(nuevaPartida(f,p,j))	$\equiv$	conocerRanking(f)

conocerRanking(palancaArriba(f))	$\equiv$	**if** ganó?(arriba(verJuegoActual(f))) **then**
		cargarPuntaje(conocerRanking(f),
		puntaje(verJuegoActual(palancaArriba(f))))
		else
		conocerRanking(f)
		fi

conocerRanking(palancaAbajo(f))	$\equiv$	**if** ganó?(abajo(verJuegoActual(f))) **then**
		cargarPuntaje(conocerRanking(f),
		puntaje(verJuegoActual(palancaAbajo(f))))
		else
		conocerRanking(f)
		fi

conocerRanking(palancaDerecha(f))	$\equiv$	**if** ganó?(derecha(verJuegoActual(f))) **then**
		cargarPuntaje(conocerRanking(f),
		puntaje(verJuegoActual(palancaDerecha(f))))
		else
		conocerRanking(f)
		fi

conocerRanking(palancaIzquierda(f))	$\equiv$	**if** ganó?(izquierda(verJuegoActual(f))) **then**
		cargarPuntaje(conocerRanking(f),
		puntaje(verJuegoActual(palancaIzquierda(f))))
		else
		conocerRanking(f)
		fi

verJuegoActual(nuevaPartida(f, p, j))	$\equiv$	p
verJuegoActual(palancaArriba(f))	$\equiv$	arriba(verJuegoActual(f))
verJuegoActual(palancaAbajo(f))	$\equiv$	abajo(verJuegoActual(f))
verJuegoActual(palancaDerecha(f))	$\equiv$	derecha(verJuegoActual(f))
verJuegoActual(palancaIzquierda(f))	$\equiv$	izquierda(verJuegoActual(f))
verContrincante(f)	$\equiv$	contrincante(conocerRanking(f),jugadorActual(f))
puntajeDeJugadorActual(f)	$\equiv$	obtener(ver(conocerRanking(f)),jugadorActual(f))
puntajeParaMejorar(f)	$\equiv$	pred(π_2(verContrincante(f)))

Fin TAD