Algoritmos y Estructuras de Datos II

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

Trabajo Práctico 2: "Fichin Pacalgo2"

Grupo: tomarAgua()

Integrante	LU	Correo electrónico
Reyna Maciel, Guillermo José	393/20	guille.j.reyna@gmail.com
Casado Farall, Joaquin	072/20	joakinfarall@gmail.com
Fernández Spandau, Luciana	368/20	fernandezspandau@gmail.com
Chumacero, Carlos Nehemias	492/20	chumacero2013@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

1. Fichín

TAD FICHÍN

géneros fichin

exporta fichin, generadores, observadores, mejorPuntajeDelJugador, proximoMejorPuntaje, nombreDelProximo

usa BOOL, NAT, CONJUNTO(α), PARTIDA, DICCIONARIO(NOMBRE, NAT), NOMBRE

igualdad observacional

$$(\forall f, f' : \text{fichin}) \left(f =_{\text{obs}} f' \iff \begin{pmatrix} \text{ranking}(f) =_{\text{obs}} \text{ranking}(f') \wedge \\ ((\neg\text{partidaEnCurso}(f) \wedge \neg\text{partidaEnCurso}(f')) \vee \\ ((\text{partidaEnCurso}(f) \wedge \text{partidaEnCurso}(f')) \wedge_L \\ \text{partida}(f) =_{\text{obs}} \text{partida}(f') \wedge \\ \text{nombreJugador}(f) =_{\text{obs}} \text{nombreJugador}(f')) \end{pmatrix} \right)$$

observadores básicos

nivel	: fichin	$\longrightarrow$	partida	
partidaEnCurso	: fichin	$\longrightarrow$	bool	
partida	: fichin f	$\longrightarrow$	partida	{partidaEnCurso(f)}
ranking	: fichin	$\longrightarrow$	dicc(nombre, nat)	
nombreJugador	: fichin f	$\longrightarrow$	nombre	{partidaEnCurso(f)}

generadores

iniciarFichin	: partida p	$\longrightarrow$	fichin	{partidaRecienEmpezada(p)}
iniciarPartida	: fichin f $\times$ nombre	$\longrightarrow$	fichin	{$\neg$partidaEnCurso(f)}
mover	: fichin f $\times$ dirección d	$\longrightarrow$	fichin	{partidaEnCurso(f) $\wedge_L$ esMovimientoValido(partida(f), d)}

otras operaciones

mejorPuntajeDelJugador	: fichin f	$\longrightarrow$	nat	{partidaEnCurso(f) $\wedge_L$ jugadorEnRanking(f)}
proximoMejorPuntaje	: fichin f	$\longrightarrow$	nat	$\left\{\begin{array}{l}\text{partidaEnCurso}(f) \wedge \neg\text{vacio?}(\text{claves}(\text{ranking}(f))) \wedge_L \\ \text{jugadorEnRanking}(f) \wedge_L \neg\text{jugadorPrimero}(f)\end{array}\right\}$
nombreDelProximo	: fichin f	$\longrightarrow$	nombre	$\left\{\begin{array}{l}\text{partidaEnCurso}(f) \wedge \neg\text{vacio?}(\text{claves}(\text{ranking}(f))) \wedge_L \\ \text{jugadorEnRanking}(f) \wedge_L \neg\text{jugadorPrimero}(f)\end{array}\right\}$
jugadorEnRanking	: fichin f	$\longrightarrow$	bool	{partidaEnCurso(f)}
jugadorPrimero	: fichin f	$\longrightarrow$	bool	{partidaEnCurso(f)}
siguienteClaveMayor	: dicc d $\times$ nat	$\longrightarrow$	α	{$\neg$vacio?(claves(d)}
mayorClave	: dicc d	$\longrightarrow$	α	{$\neg$vacio?(claves(d)}

axiomas $\forall f$: fichín, $\forall p$: partida, $\forall n$: nombre, $\forall m$: nat, $\forall r$: dicc(nombre, nat)

nivel(iniciarFichin(p))	$\equiv$	p
nivel(iniciarPartida(f, n))	$\equiv$	nivel(f)
nivel(mover(f, s))	$\equiv$	nivel(f)
partidaEnCurso(iniciarFichin)	$\equiv$	false
partidaEnCurso(iniciarPartida(f, n))	$\equiv$	$\neg$terminóElJuego(p)
partidaEnCurso(moverF(f, d))	$\equiv$	$\neg$terminóElJuego(mover(partida(f), d))
partida(iniciarPartida(f, n))	$\equiv$	nivel(f)
partida(moverF(f, d))	$\equiv$	mover(partida(f), d)
ranking(iniciarFichin)	$\equiv$	vacío
ranking(iniciarPartida(f, n))	$\equiv$	ranking(f)

ranking(moverF(f, d)) $\equiv$ **if** ganó?(partida(moverF(f, d))) $\wedge$ ($\neg$jugadorEnRanking(f) $\vee_L$ puntaje(partida(moverF(f, d))) $<$ mejorPuntajeDelJugador(f)) **then** definir(nombreJugador(f), puntaje(partida(moverF(f, d)))) **else** ranking(f) **fi**

nombreJugador(iniciarPartida(f, n)) $\equiv$ n

nombreJugador(moverF(f, d)) $\equiv$ nombreJugador(f)

mejorPuntajeDelJugador(f) $\equiv$ obtener(nombreJugador(f), ranking(f))

proximoMejorPuntaje(f) $\equiv$ obtener(nombreDelProximo(f), ranking(f))

nombreDelProximo(f) $\equiv$ siguienteClaveMayor(ranking(f), mejorPuntajeDelJugador(f)))

siguienteClaveMayor(r, m) $\equiv$ **if** #(claves(r)) = 1 $\vee$ ($m <$ obtener(dameUno(claves(r)), r) $\wedge$ obtener(dameUno(claves(r)), r) $<$ siguienteClaveMayor(borrar(dameUno(claves(r)), r)), m)) **then** dameUno(claves(r)) **else** siguienteClaveMayor(borrar(dameUno(claves(r)), r)), m) **fi**

jugadorEnRanking(f) $\equiv$ def?(nombreJugador(f), ranking(f))

jugadorPrimero(f) $\equiv$ mayorClave(ranking(f)) = nombreJugador(f)

mayorClave(r) $\equiv$ **if** #(claves(r)) = 1 $\vee$ (obtener(mayorClave(sinUno(claves(r))), sinUno(claves(r))) $<$ obtener(dameUno(claves(r)), r)) **then** dameUno(claves(r)) **else** mayorClave(sinUno(claves(r))) **fi**

Fin TAD

TAD DIRECCION es STRING

TAD DIMENSIÓN, COORDENADAS

extiende TUPLA(NAT, NAT)

otras operaciones

$\bullet = \bullet$: tupla(nat $\times$ nat) $\times$ tupla(nat $\times$ nat) $\longrightarrow$ bool

mover : tupla(nat $\times$ nat) t $\times$ direccion d $\longrightarrow$ tupla(nat,nat) $\{d \in \{$ "arr","abj","der","izq" $\}\}$

axiomas $\forall a, a_1, a_2$: tupla(nat, nat)

$a_1 = a_2$ $\equiv$ $\pi_1(a_1) = \pi_1(a_2) \wedge \pi_2(a_1) = \pi_2(a_2)$

mover(a, d) $\equiv$ **if** $d =$ "arr" **then** $< \pi_1(a_1), \pi_1(a_2) + 1 >$ **else** **if** $d =$ "abj" **then** $< \pi_1(a_1), \pi_1(a_2) - 1 >$ **else** **if** $d =$ "izq" **then** $< \pi_1(a_1)$ - 1, $\pi_1(a_2) >$ **else** $< \pi_1(a_1) + 1, \pi_1(a_2) >$ **fi** **fi** **fi**

Fin TAD

TAD PARTIDA

géneros partida

exporta partida, generadores, observadores, seAsustó?, ganó?, terminóElJuego?, esMovimientoValido

usa BOOL, NAT, CONJUNTO(α), MAPA, COORDENADAS, DIMENSIÓN

igualdad observacional

$$(\forall p, p' : \text{partida}) \left(p =_{\text{obs}} p' \iff \begin{pmatrix} \text{jugador}(p) =_{\text{obs}} \text{jugador}(p') \wedge \\ \text{mapa}(p) =_{\text{obs}} \text{mapa}(p') \wedge \\ \text{chocolates}(p) =_{\text{obs}} \text{chocolates}(p') \wedge \\ \text{movimientosConInmunidad}(p) \\ =_{\text{obs}} \text{movimientosConInmunidad}(p') \wedge \\ \text{puntaje}(p) =_{\text{obs}} \text{puntaje}(p') \end{pmatrix} \right)$$

observadores básicos

jugador	: partida	$\longrightarrow$	coordenadas
mapa	: partida	$\longrightarrow$	mapa
chocolates	: partida	$\longrightarrow$	conj(coordenadas)
movimientosConInmunidad	: partida	$\longrightarrow$	nat
puntaje	: partida	$\longrightarrow$	nat

generadores

iniciarPartida	: mapa m $\times$ conj(coordenadas) *chocolates*	$\longrightarrow$	partida $\left\{ \begin{matrix} (\forall \text{c:coordenadas})(\text{c} \in \text{chocolates} \Rightarrow \text{enRango?(c,dimensión(m))} \wedge_L \\ \neg(\text{c} \in \text{puntoDeSalida(m)} \vee \text{c} \in \text{paredes(m)})) \end{matrix} \right\}$
mover	: partida p $\times$ dirección d	$\longrightarrow$	partida $\{\text{esMovimientoValido}(p, d)\}$

otras operaciones

seAsustó?	: partida	$\longrightarrow$	bool
ganó?	: partida	$\longrightarrow$	bool
terminóElJuego?	: partida	$\longrightarrow$	bool
hayFantasmasCerca	: coordenadas $\times$ conj(coordenadas)	$\longrightarrow$	bool
esMovimientoValido	: partida $\times$ dirección	$\longrightarrow$	bool

axiomas $\forall p$: partida, $\forall m$: mapa, $\forall c$: coordenadas, $\forall f, c$: conj(coordenadas)

jugador(iniciarPartida(m))	$\equiv$	puntoDeSalida(m)
jugador(mover(p, d))	$\equiv$	moverT(jugador(p), d)
mapa(iniciarPartida(m))	$\equiv$	m
mapa(mover(p, d))	$\equiv$	mapa(p)
chocolates(iniciarPartida(m, c))	$\equiv$	c
chocolates(mover(p, d))	$\equiv$	chocolates(p) $-$ {jugador(mover(p, d))}
movimientosConInmunidad(iniciarPartida(m, c))	$\equiv$	0
movimientosConInmunidad(mover(p, d))	$\equiv$	**if** jugador(mover(p, d)) $\in$ chocolates(p) **then** 10 **else** max(0, movimientosConInmunidad(p)$-$1) **fi**
puntaje(iniciarPartida(m, c))	$\equiv$	0
puntaje(mover(p, d))	$\equiv$	puntaje(p)+1
seAsustó?(p)	$\equiv$	hayFantasmasCerca(jugador(p), fantasmas(mapa(p))) $\wedge$ movimientosConInmunidad(p) = 0
ganó?(p)	$\equiv$	jugador(p) = puntoDeLlegada(mapa(p))
terminóElJuego?(p)	$\equiv$	seAsustó(p) $\vee$ ganó(p)

hayFantasmasCerca(c, f) $\equiv$ **if** vacío?(f) **then**
false
else
$|\pi_1(c) - \pi_1(\text{dameUno}(f))| +$
$|\pi_2(c) - \pi_2(\text{dameUno}(f))| < 3$
$\vee$ hayFantasmasCerca(sinUno(f))
fi

esMovimientoValido(p, d) $\equiv$ $d \in \{"arr", "abj", "izq", "der"\} \wedge_L$
$\neg$terminóElJuego(p) $\wedge$
enRango?(moverT(d, jugador(p)), dimensión(mapa(p)))
$\wedge$ $\neg$(moverT(d, jugador(p)) $\in$ paredes(mapa(p)))

Fin TAD

TAD MAPA

géneros mapa

exporta mapa, generadores, observadores, enRango?, estaOcupado?

usa BOOL, NAT, CONJUNTO(α), COORDENADAS, DIMENSIÓN

igualdad observacional

$$(\forall m, m' : \text{mapa}) \left(m =_{\text{obs}} m' \iff \begin{pmatrix} \text{dimensión}(m) =_{\text{obs}} \text{dimensión}(m') \wedge \\ \text{paredes}(m) =_{\text{obs}} \text{paredes}(m') \wedge \\ \text{fantasmas}(m) =_{\text{obs}} \text{fantasmas}(m') \wedge \\ \text{puntoDeSalida}(m) =_{\text{obs}} \text{puntoDeSalida}(m') \wedge \\ \text{puntoDeLlegada}(m) =_{\text{obs}} \text{puntoDeLlegada}(m') \end{pmatrix} \right)$$

observadores básicos

dimensión	: mapa	$\longrightarrow$	dimensión	
paredes	: mapa	$\longrightarrow$	conj(coordenadas)	
fantasmas	: mapa	$\longrightarrow$	conj(coordenadas)	
puntoDeSalida	: mapa	$\longrightarrow$	coordenadas	
puntoDeLlegada	: mapa	$\longrightarrow$	coordenadas	

generadores

nuevoMapa	: dimensión d $\times$ coordenadas *inicio* $\times$ coordenadas fin	$\longrightarrow$	mapa	{π_1(d) * π_2(d) $\geq 2 \wedge \neg$ (inicio = fin) $\wedge$ enRango?(inicio, d) $\wedge$ enRango?(fin, d)}
agregarFantasma	: mapa m $\times$ coordenadas c	$\longrightarrow$	mapa	{enRango?(c, dimensión(m)) $\wedge$ $\neg$estaOcupado?(c, m)}
agregarPared	: mapa m $\times$ coordenadas c	$\longrightarrow$	mapa	{enRango?(c, dimensión(m)) $\wedge$ $\neg$estaOcupado?(c, m)}

otras operaciones

esRango?	: coordenadas $\times$ dimensión	$\longrightarrow$	bool	
estaOcupado?	: coordenadas c $\times$ mapa m	$\longrightarrow$	bool	{enRango?(c, dimension(m))}

axiomas $\forall m$: mapa, $\forall i, f, c$: coordenadas, $\forall d$: dimensión

dimension(nuevoMapa(d, i, f)) $\equiv$ d
dimension(agregarFantasma(m, c)) $\equiv$ dimension(m)
dimension(agregarPared(m, c)) $\equiv$ dimension(m)

paredes(nuevoMapa(d, i, f)) $\equiv$ $\emptyset$
paredes(agregarFantasma(m, c)) $\equiv$ paredes(m)
paredes(agregarPared(m, c)) $\equiv$ Ag(c, paredes(m))

fantasmas(nuevoMapa(d, i, f)) $\equiv$ $\emptyset$
fantasmas(agregarFantasma(m, c)) $\equiv$ Ag(c, paredes(m))
fantasmas(agregarPared(m, c)) $\equiv$ paredes(m)

puntoDeSalida(nuevoMapa(d, i, f)) $\equiv$ i
puntoDeSalida(agregarFantasma(m, c)) $\equiv$ puntoDeSalida(m)
puntoDeSalida(agregarPared(m, c)) $\equiv$ puntoDeSalida(m)

puntoDeLlegada(nuevoMapa(d, i, f))	$\equiv$	f
puntoDeLlegada(agregarFantasma(m, c))	$\equiv$	puntoDeLlegada(m)
puntoDeLlegada(agregarPared(m, c))	$\equiv$	puntoDeLlegada(m)
enRango?(c, d)	$\equiv$	$\pi_1(c) < \pi_1(d) \land \pi_2(c) < \pi_2(d)$
estaOcupado?(c, m)	$\equiv$	$\neg(c \in (\text{paredes}(m) \cup \text{fantasmas}(m) \cup \{\text{puntoDeSalida}(m), \text{puntoDeLlegada}(m)\})$

Fin TAD

2. Aclaraciones

2.1. Fichín

- En la axiomatización de partidaEnCurso, en el caso de iniciarPartida consideramos que como no está restringido que una partida se inicie con un fantasma en rango del jugador, hay casos posibles en los que partidaEnCurso de iniciarPartida no es true.

2.2. Partida - Mapa

- Estamos utilizando un sistema de coordenadas al estilo cartesiano, donde la primera coordenada representa el eje horizontal y aumenta hacia la derecha y la segunda coordenada representa el eje vertical y aumenta hacia arriba.
- Por conveniencia, interpretamos que los chocolates no pueden solaparse con el punto de salida (porque para comer un chocolate el jugador debe moverse a la coordenada en donde se encuentra el chocolate) ni con paredes. Interpretamos que sí pueden solaparse con fantasmas y con el punto de llegada.
- En la axiomatización del observador chocolates de partida, para simplificar la escritura utilizamos la propiedad de que $c - \{a\} = c$ cuando $a \notin c$, pues si la intersección de dos conjuntos es vacía, la resta no los modifica.
- Interpretamos que los movimientos con inmunidad no son acomulativos, sino que se recargan. Es decir, comer un chocolate mientras el jugador ya tiene inmunidad de otro chocolate no le suma 10 movimientos más de inmunidad, sino que le recarga los movimientos con inmunidad a un máximo de 10 (similar a la funcionalidad de la inmunidad en el Pac-Man original).
- Si bien esta especificación permite partidas inganables o imperdibles (e.g. el punto de llegada esté rodeado de paredes, o que haya un fantasma al lado del punto de salida), consideramos que restringir estas posibilidades sería sobreespecificar, puesto que el enunciado no las menciona.
- Interpretamos que el punto de salida y el punto de llegada no pueden estar en el mismo espacio y que estos no se pueden solapar con fantasmas o paredes, puesto que el enunciado dice que se asignan dos casilleros especiales para ellos.
- En la axiomatización de inmunidadAlMover, la función máx se asume como la del TAD Natural, pero el parámetro movimientosConInmunidad(p) $-$ 1 puede ser -1 cuando movimientosConInmunidad(p) $= 0$. Asumimos que la función máximo se extiende con los enteros.
- El enunciado dice que el puntaje se debe poder calcular al final de la partida. En nuestra especificación el puntaje se puede calcular en cualquier punto de la partida, y en particular cuando la partida finalizó y el jugador ganó.