Algoritmos y Estructura de Datos 2

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

Exorcismo Extremo

TP1

Integrante	LU	Correo electrónico
Rosinov, Gaston Einan	37/18	grosinov@gmail.com
Schuster, Martin Ariel	208/18	m.a.schuster98@gmail.com
Panichelli, Manuel	72/18	panicmanu@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. TAD Juego

TAD Juego

géneros juego

exporta observadores, generadores, puntaje

usa Habitacion

igualdad observacional

$$(\forall j, j' : \text{juego}) \left(j =_{\text{obs}} j' \iff \begin{pmatrix} (n = 0? =_{\text{obs}} m = 0?) \wedge_{\text{L}} \\ (\neg(n = 0?) \Rightarrow_{\text{L}} (\text{pred}(n) =_{\text{obs}} \text{pred}(m))) \end{pmatrix} \right)$$

igualdad observacional

$$(\forall j, j' : \text{juego}) \left(j =_{\text{obs}} j' \iff \begin{pmatrix} (\text{accionesPJs(j)} =_{\text{obs}} \text{accionesPJs(j')}) \wedge \\ (\text{accionesFan(j)} =_{\text{obs}} \text{accionesFan(j')}) \wedge \\ (\text{localizarJugadores(j)} =_{\text{obs}} \text{localizarJugadores(j')}) \wedge \\ (\text{hab(j)} =_{\text{obs}} \text{hab(j')}) \wedge \\ ((\forall\ \text{p} : \text{pj})\ (\text{vivePJ?(j, p)} =_{\text{obs}} \text{vivePJ?(j', p)})) \wedge \\ ((\forall\ \text{f} : \text{fantasma})\ ((\text{viveFan?(j, p)} =_{\text{obs}} \text{viveFan?(j', p)}) \wedge \\ (\text{ubicacionInicialFan(j, f)} =_{\text{obs}} \text{ubicacionInicialFan(j', f)})) \end{pmatrix} \right)$$

observadores básicos

accionesPJs	: juego	⟶	dicc(pj, secu(accion))	
accionesFan	: juego	⟶	dicc(pj, secu(accion))	
hab	: juego	⟶	hab	
vivePJ?	: juego j × pj p	⟶	bool	{p ∈ jugadores(j)}
viveFan?	: juego j × fantasma f	⟶	bool	{f ∈ fantasmas(j)}
ubicacionInicialFan	: juego j × fantasma f	⟶	ubicacion	{f ∈ fantasmas(f)}
localizarJugadores	: juego	⟶	dicc(pj, ubicacion)	

generadores

iniciar	: conj(pj) pjs × secu(accion) as × ubicacion u × hab h	⟶	juego	{esConexa?(h) ∧ ¬ ∅?(as) ∧ ¬ ∅?(pjs) ∧ esValida?(h, pos(u))}
proxPaso	: juego j × pj p × accion a	⟶	juego	{p ∈ jugadores(j) ∧ₗ vivePJ?(j, p) ∧ ¬ termino?(j) ∧ ¬ esMirar(a)}

otras operaciones

jugadores	: juego	⟶	conj(pj)	
fantasmas	: juego	⟶	conj(fantasma)	
nombreSiguienteFan	: juego	⟶	fantasma	
puntaje	: juego	⟶	nat	
ronda	: juego	⟶	nat	
paso	: juego	⟶	nat	
cantAcciones	: juego × conj(pj)	⟶	nat	
terminaRonda	: juego j × pj p × accion	⟶	bool	{p ∈ jugadores(j)}
fantasmaEspecial	: juego	⟶	fantasma	

termino?	: juego	$\longrightarrow$	bool	
estanVivos	: juego $\times$ conj(pj) pjs	$\longrightarrow$	bool	{pjs $\subseteq$ jugadores(j)}
ubicacionInicialPJ	: juego j $\times$ pj p	$\longrightarrow$	ubicacion	{p $\in$ jugadores(j)}
ubicacionPJ	: juego j $\times$ pj p	$\longrightarrow$	ubicacion	{p $\in$ jugadores(j)}
ubicacionFan	: juego j $\times$ fantamsa f	$\longrightarrow$	ubicacion	{f $\in$ fantasmas(j)}
deducirUbicacion	: juego j $\times$ ubicacion u $\times$ acciones	$\longrightarrow$	ubicacion	{esValida?(hab(j), pos(u))}
moriraFantasma	: juego j $\times$ pj p $\times$ accion $\times$ fantasma f	$\longrightarrow$	bool	{p $\in$ jugadores(j) $\wedge$ f $\in$ fantasmas(j) $\wedge_{\text{L}}$ viveFan?(j, f) $\wedge$ vivePJ?(j, p)}
moriraPJ	: juego j $\times$ conj(fantasma) fs $\times$ pj p $\times$ accion	$\longrightarrow$	bool	{p $\in$ jugadores(j) $\wedge$ fs $\subseteq$ fantasmas(j) $\wedge_{\text{L}}$ vivePJ?(j, p)}
moriraPJPorFan	: juego j $\times$ fantasma f $\times$ pj p $\times$ accion	$\longrightarrow$	bool	{p $\in$ jugadores(j) $\wedge$ f $\in$ fantasmas(j) $\wedge_{\text{L}}$ vivePJ?(j, p) $\wedge$ viveFan?(j, f)}
accionFan	: juego j $\times$ fantasma f	$\longrightarrow$	accion	{f $\in$ fantasmas(j) $\wedge_{\text{L}}$ viveFan?(j, f)}
inicializarAcciones	: conj(pj)	$\longrightarrow$	dicc(pj, secu(accion))	
agregarFantasma	: juego j $\times$ ubicacion u $\times$ dicc(fantasma $\times$ secu(accion)) $\times$ fantasma $\times$ secu(accion)	$\longrightarrow$	dicc(fantasma, secu(accion))	{esValida?(hab(j), pos(u))}
generarAccionesFan	: juego j $\times$ ubicacion u $\times$ secu(accion)	$\longrightarrow$	secu(accion)	{esValida?(hab(j), pos(u))}

axiomas

$\forall\ p$: pj,
$\forall\ pjs$: conj(pj),
$\forall\ f$: fantasma,
$\forall\ fs$: conj(fantasma),
$\forall\ j$: juego,
$\forall\ h$: hab,
$\forall\ u, uInicialPJ$: ubicacion,
$\forall\ a$: accion,
$\forall\ as$: secu(accion)

accionesPJs(iniciar(pjs, as, u, h))	$\equiv$	inicializarAcciones(pjs)
accionesPJs(proxPaso(j, p, a))	$\equiv$	**if** $\neg$ terminaRonda(j, p, a) **then** definir(p, obtener(p, accionesPJs(j)) $\circ$ a, accionesPJs(j)) **else** inicializarAcciones(jugadores(j)) **fi**
accionesFan(iniciar(pjs, as, u, h))	$\equiv$	definir(nombreSiguienteFan(j), as, vacio)
accionesFan(proxPaso(j, p, a))	$\equiv$	**if** $\neg$ terminaRonda(j, p, a) **then** accionesFan(j) **else** agregarFantasma(j, ubicacionInicialPJ(j, p), accionesFan(j), nombreSiguienteFan(j), obtener(p, accionesPJs(j)) $\circ$ a) **fi**
hab(iniciar(pjs, as, u, h))	$\equiv$	h
hab(proxPaso(j, p, a))	$\equiv$	hab(j)
vivePJ?(iniciar(pjs, as, u, h), p')	$\equiv$	true

vivePJ?(proxPaso(j, p, a), p') ≡ terminaRonda?(j, p, a) ∨
if p = p'
then ¬ moriraPJ(j, fantasmas(j), p, a)
else vivePJ?(j, p') ∧$_L$ ¬ moriraPJ(j, fantasmas(j), p, a)
fi

viveFan?(iniciar(pjs, as, u, h), f) ≡ true

viveFan?(proxPaso(j, p, a), f) ≡ terminaRonda?(j, p, a) ∨
(viveFan?(j, f) ∧$_L$ ¬ moriraFantasma(j, p, a, f))

ubicacionInicialFan(iniciar(pjs, as, u, h)) ≡ u

ubicacionInicialFan(proxPaso(j, p, a)) ≡ **if** f ∈ fantasmas(j)
then ubicacionInicialFan(j, f)
else ubicacionInicialPJ(j, p)
fi

jugadores(j) ≡ claves(accionesPJs(j))

fantasmas(j) ≡ claves(accionesFan(j))

nombreSiguienteFan(j) ≡ #(claves(accionesFan(j))) + 1

puntaje(j) ≡ ronda(j) - 1

ronda(j) ≡ #(fantasmas(j))

paso(j) ≡ cantAcciones(j, jugadores(j))

cantAcciones(j, pjs) ≡ **if** ∅?(pjs)
then 0
else long(obtener(dameUno(pjs), accionesPJs(j))) +
cantAcciones(j, sinUno(pjs))
fi

termino?(j) ≡ ¬ estanVivos(j, jugadores(j))

estanVivos(j, pjs) ≡ **if** ∅?(pjs)
then true
else vivePJ?(j, dameUno(pjs)) ∧
estanVivos(j, sinUno(pjs))
fi

fantasmaEspecial(j) ≡ #(claves(accionesFan(j)))

ubicacionInicialPJ(j, p) ≡ obtener(p, localizarJugadores(j))

ubicacionPJ(j, p) ≡ deducirUbicacion(j, ubicacionInicialPJ(j, p),
obtener(p, accionesPJs(j)))

ubicacionFan(j, f) ≡ deducirUbicacion(j, ubicacionInicialFan(j, f),
obtener(f, accionesFan(j)))

deducirUbicacion(j, u, as) ≡ **if** vacia?(as)
then u
else deducirUbicacion(j, ubicacionLuegoDe(prim(as), hab(j), u),
fin(as))
fi

agregarFantasma(j, uInicialPJ,
accionesFantasmas, f, as) ≡ definir(f, generarAccionesFantasma(j, uInicialPJ, as),
accionesFantasmas)

generarAccionesFan(j, uInicialPJ, as) ≡ as
& (nada • nada • nada • nada • nada)
& invertir(hab(j), uInicialPJ, as)

inicializarAcciones(pjs) ≡ **if** ∅?(pjs)
then vacio
else definir(dameUno(pjs), <>,
inicializarAcciones(sinUno(pjs)))
fi

terminaRonda(j, p, a) ≡ moriraFantasma(j, p, a, fantasmaEspecial(j))

moriraFantasma(j, p, a, f) ≡ pos(ubicacionFan(j, f)) ∈ posicionesAfectadasPor(a, hab(j), ubicacionPJ(j, p))

moriraPJ(j, fs, p, a) ≡ **if** ∅?(fs) **then** false **else** (viveFan?(j, dameUno(fs)) $\wedge_{\text{L}}$ moriraPJPorFan(j, dameUno(fs), p, a)) $\vee_{\text{L}}$ moriraPJ(j, sinUno(fs), p, a) **fi**

moriraPJPorFan(j, f, p, a) ≡ ¬ moriraFantasma(j, p, a, f) ∧ (pos(ubicacionLuegoDe(a, hab(j), ubicacionPJ(j, p))) ∈ posicionesAfectadasPor(accionFan(j, f), hab(j), ubicacionFan(j, f)))

accionFan(j, f) ≡ obtener(accionesFan(j), f)[paso(j) % obtener(accionesFan(j), f)]

Fin TAD

2. TAD Habitacion

TAD Habitacion

géneros hab

exporta hab, observadores, generadores, esConexa?

usa posicion, bool, nat

igualdad observacional

$$(\forall h, h' : \text{hab}) \left(h =_{\text{obs}} h' \iff \begin{pmatrix} (\forall p : \text{posicion})(\text{esValida?}(p, h) =_{\text{obs}} \text{esValida?}(p, h') \wedge_{\text{L}} \\ (\text{esValida?}(p, h) \Rightarrow_{\text{L}} \\ (\text{estaOcupada?}(p, h) =_{\text{obs}} \text{estaOcupada?}(p, h')))) \end{pmatrix} \right)$$

observadores básicos

esValida?	: hab × posicion	⟶ bool	
estaOcupada?	: hab h × posicion p	⟶ bool	{esValida?(h, p)}

generadores

nueva	: nat n	⟶ hab	{n>1}
ocupar	: hab h × posicion p	⟶ hab	{esValida?(h, p) $\wedge_{\text{L}}$¬ estaOcupada?(h, p)}

otras operaciones

esConexa?	: hab	⟶ bool	
tamano	: hab	⟶ nat	
posiciones	: hab	⟶ conj(posicion)	
posicionesLibres	: hab h × conj(posicion) ps	⟶ conj(posicion)	{ps ⊆ posiciones(h)}
verificarAlcance	: hab h × conj(posicion) ps	⟶ bool	{ps ⊆ posiciones(h)}
verificarAlcancePos	: hab h × conj(posicion) ps × posicion p	⟶ bool	{ps ⊆ posiciones(h) ∧ p ∈ posiciones(h)}

axiomas ∀ h: hab ∀ ps: conj(posicion) ∀ p: posicion ∀ n, k, tam: nat

esValida?(nueva(n),p) ≡ $0 \leq \Pi_1(\text{p}) < \text{n} \wedge 0 \leq \Pi_2(\text{p}) < \text{n}$

esValida?(ocupar(h,p'),p)	≡	p = p' $\vee_L$ esValida?(h, p)
estaOcupada?(nueva(n),p)	≡	false
estaOcupada?(ocupar(h,p'),p)	≡	p = p' $\vee$ estaOcupada?(h, p)
tamano(nueva(n))	≡	n
tamano(ocupar(h, p))	≡	tamano(h)
esConexa?(h)	≡	verificarAlcance(h, posicionesLibres(posiciones(h)))
posicionesLibres(h, ps)	≡	**if** $\emptyset?(ps)$ **then** $\emptyset$ **else** (**if** estaOcupada?(h, dameUno(ps)) **then** $\emptyset$ **else** {dameUno(ps)} **fi**) $\cup$ posicionesLibres(h, sinUno(ps)) **fi**
verificarAlcance(h, ps)	≡	**if** $\emptyset?(ps)$ **then** true **else** verificarAlancePos(h, ps, dameUno(ps)) $\wedge$ verificarAlcance(h, p) **fi**
verificarAlcancePos(h, ps, p)	≡	**if** $\emptyset?(ps)$ **then** true **else** esAlcanzable(h, p, dameUno(ps)) $\wedge$ verificarAlcancePos(h, p, sinUno(ps)) **fi**
posiciones(h)	≡	darPosiciones(h, tamano(h) - 1, tamano(h) - 1, tamano(h) - 1)
darPosiciones(h, n, k, tam)	≡	**if** $n = 0? \wedge k = 0?$ **then** $\emptyset$ **else if** $k = 0?$ **then** Ag((n,k), darPosiciones(h, n - 1, tam, tam)) **else** Ag((n,k), darPosiciones(h, n, k - 1, tam)) **fi** **fi**

Fin TAD

3. TAD Accion

TAD Accion

géneros accion

exporta observadores, generadores, genero, otras operaciones

igualdad observacional

$$(\forall a, a' : \text{accion}) \left(a =_{\text{obs}} a' \iff \begin{pmatrix} \text{esNada(a)} =_{\text{obs}} \text{esNada(a')} \wedge \\ \text{esDisparar(a)} =_{\text{obs}} \text{esDisparar(a')} \wedge \\ \text{esMover(a)} =_{\text{obs}} \text{esMover(a')} \wedge \\ \text{esMirar(a)} =_{\text{obs}} \text{esMirar(a')} \wedge \\ ((\text{esMover(a)} \vee \text{esMirar(a)}) \Rightarrow_L \text{direccion(a)} =_{\text{obs}} \text{direccion(a')}) \end{pmatrix} \right)$$

secu(accion)

observadores básicos

esMover	: accion	$\longrightarrow$	bool
esMirar	: accion	$\longrightarrow$	bool

esDisparar	: accion	$\longrightarrow$ bool		
esNada	: accion	$\longrightarrow$ bool		
direccion	: accion a	$\longrightarrow$ direccion	{esMirar(a) $\vee$ esMover(a)}	

generadores

mover	: direccion	$\longrightarrow$ accion		
mirar	: direccion	$\longrightarrow$ accion		
disparar	:	$\longrightarrow$ accion		
nada	:	$\longrightarrow$ accion		

otras operaciones

ubicacionLuegoDe	: accion a $\times$ hab h $\times$ ubicacion u	$\longrightarrow$ conj(pos)	{esValida?(h, pos(u))}	
posicionesAfectadasPor	: accion a $\times$ hab h $\times$ ubicacion u	$\longrightarrow$ conj(pos)	{esValida?(h, pos(u))}	
$\neg \bullet$	: accion	$\longrightarrow$ accion		
invertir	: hab h $\times$ ubicacion u $\times$ secu(accion)	$\longrightarrow$ secu(accion)	{esValida?(h, pos(u))}	

axiomas $\forall\, n, m$: nat, $\forall\, u$: ubicacion, $\forall\, a$: habitacion

posicionesAfectadasPor(mover(d), h, u) $\equiv \emptyset$

posicionesAfectadasPor(mirar(d), h, u) $\equiv \emptyset$

posicionesAfectadasPor(nada, h, u) $\equiv \emptyset$

posicionesAfectadasPor(disparar, h, u) $\equiv$ **if** esValida?(h, proxPosEnDir(dir(u), pos(u)) $\wedge_{\text{L}}$ ¬estaOcupada?(h, proxPosEnDir(dir(u), pos(u))) **then** Ag(proxPosEnDir(dir(u), pos(u)),posicionesAfectadasPor(disparar,h, ⟨proxPosEnDir(dir(u), pos(u)), dir(u)⟩)) **else** $\emptyset$ **fi**

invertir(h, u, as) $\equiv$ **if** vacia?(as) **then** <> **else** invertir(h, ubicacionLuegoDe(prim(as), h, u), fin(as)) $\vee$ ¬(prim(as), h, u) **fi**

¬(mover(d), h, u) $\equiv$ **if** pos(ubicacionLuegoDe(mover(d), h, u)) = pos(u) **then** mirar(opuesta(d)) **else** mover(opuesta(d)) **fi**

¬(mirar(d), h, u) $\equiv$ mirar(opuesta(d))

¬(disparar, h, u) $\equiv$ disparar

¬(nada, h, u) $\equiv$ nada

ubicacionLuegoDe(nada, h, u) $\equiv$ u

ubicacionLuegoDe(disparar, h, u) $\equiv$ u

ubicacionLuegoDe(mirar(d), h, u) $\equiv$ ⟨pos(u), d⟩

ubicacionLuegoDe(mover(d), h, u) $\equiv$ ⟨(**if** esValida?(h, proxPosEnDir(d, pos(u))) $\wedge_{\text{L}}$ ¬estaOcupada?(h, proxPosEnDir(d, pos(u))) **then** proxPosEnDir(d, pos(u)) **else** pos(u) **fi**), d⟩

esMirar(mirar(d)) $\equiv$ true

esMirar(mover(d)) $\equiv$ false

esMirar(disparar)	$\equiv$	false
esMirar(nada)	$\equiv$	false
esMover(mirar(d))	$\equiv$	false
esMover(mover(d))	$\equiv$	true
esMover(disparar)	$\equiv$	false
esMover(nada)	$\equiv$	false
esDisparar(mirar(d))	$\equiv$	false
esDisparar(mover(d))	$\equiv$	false
esDisparar(disparar)	$\equiv$	true
esDisparar(nada)	$\equiv$	false
esNada(mirar(d))	$\equiv$	false
esNada(mover(d))	$\equiv$	false
esNada(disparar)	$\equiv$	false
esNada(nada)	$\equiv$	true
direccion(mirar(d))	$\equiv$	d
direccion(mover(d))	$\equiv$	d

Fin TAD

4. TAD Direccion

TAD Direccion

géneros direccion

exporta observadores, generadores, otras operaciones

igualdad observacional

$$(\forall d, d' : \text{direccion}) \left(d =_{\text{obs}} d' \iff \begin{pmatrix} \text{esArriba(d)} =_{\text{obs}} \text{esArriba(d')} \wedge \\ \text{esAbajo(d)} =_{\text{obs}} \text{esAbajo(d')} \wedge \\ \text{esIzquierda(d)} =_{\text{obs}} \text{esIzquierda(d')} \wedge \\ \text{esDerecha(d)} =_{\text{obs}} \text{esDerecha(d')} \end{pmatrix} \right)$$

observadores básicos

esArriba	:	direccion	$\longrightarrow$	bool
esAbajo	:	direccion	$\longrightarrow$	bool
esIzquierda	:	direccion	$\longrightarrow$	bool
esDerecha	:	direccion	$\longrightarrow$	bool

generadores

arriba	:		$\longrightarrow$	direccion
abajo	:		$\longrightarrow$	direccion
izquierda	:		$\longrightarrow$	direccion
derecha	:		$\longrightarrow$	direccion

otras operaciones

opuesta	:	direccion	$\longrightarrow$	direccion

proxPosEnDir : direccion × posicion ⟶ posicion

axiomas

opuesta(arriba)	≡ abajo
opuesta(abajo)	≡ arriba
opuesta(izquierda)	≡ derecha
opuesta(derecha)	≡ izquierda
proxPosEnDir(arriba, p)	≡ $\langle \Pi_1(p), \Pi_2(p) + 1 \rangle$
proxPosEnDir(abajo, p)	≡ $\langle \Pi_1(p), \Pi_2(p) - 1 \rangle$
proxPosEnDir(izquierda, p)	≡ $\langle \Pi_1(p) - 1, \Pi_2(p) \rangle$
proxPosEnDir(derecha, p)	≡ $\langle \Pi_1(p) + 1, \Pi_2(p) \rangle$
esArriba(arriba)	≡ true
esArriba(abajo)	≡ false
esArriba(izquierda)	≡ false
esArriba(derecha)	≡ false
esAbajo(arriba)	≡ false
esAbajo(abajo)	≡ true
esAbajo(izquierda)	≡ false
esAbajo(derecha)	≡ false
esIzquierda(arriba)	≡ false
esIzquierda(abajo)	≡ false
esIzquierda(izquierda)	≡ true
esIzquierda(derecha)	≡ false
esDerecha(arriba)	≡ false
esDerecha(abajo)	≡ false
esDerecha(izquierda)	≡ false
esDerecha(derecha)	≡ true

Fin TAD

5. Extensiones y Renombres

TAD Fantasma **ES** Nat

TAD PJ **ES** Nat

TAD Posicion **ES** Tupla(Nat, Nat)

TAD Nat **extiende** Nat

otras operaciones

• % • : nat × nat ⟶ nat

axiomas ∀ n, m: nat

n % m $\equiv$ **if** n $<$ m **then** n **else** (n - m) % m **fi**

Fin TAD

TAD Ubicacion **extiende** Tupla(Posicion, Direccion)

otras operaciones

pos : ubicacion $\longrightarrow$ posicion

dir : ubicacion $\longrightarrow$ direccion

axiomas $\forall$ u: ubicacion

pos(u) $\equiv$ Π_1(u)

dir(u) $\equiv$ Π_2(u)

Fin TAD

TAD Secuencia **extiende** Secuencia

otras operaciones

$\bullet[\bullet]$: secu(α) s $\times$ nat i $\longrightarrow$ α {i $<$ long(s)}

axiomas $\forall$ s: secu(α), $\forall$ i: nat

s[i] $\equiv$ **if** $i = 0$? **then** prim(s) **else** fin(s)[$i - 1$] **fi**

Fin TAD