Algoritmos y Estructuras de Datos II

Trabajo Práctico 1

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

Pacalgo2

Integrante	LU	Correo electrónico
Asmad Murga, Victor Manuel	760/19	victorasmad2@gmail.com
Azzollini, Mauro Ezequiel	774/19	mauro.azzo3210@gmail.com
Battolla, Gianfranco	017/20	gianfrancobtl@gmail.com
Porres, Emanuel Martín	264/20	emanuelmporres@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Parte 1

Bool extendido.

TAD Coordenada ES tupla(nat, nat)

TAD Direccion ES String — Pueden usar "TAD Direccion es ENUM {ARRIBA, ABAJO, IZQUIERDA, DERECHA}"

TAD Mapa

géneros mapa

usa Tupla, Coordenada, Conjunto, Bool, Nat

exporta géneros, observadores básicos, generadores, coordenadasDelMapa, esCoordenadaValidaParaAvanzar, espaciosVacios

igualdad observacional

$$(\forall m, m' : \text{mapa}) \left(m =_{\text{obs}} m' \iff \left(\begin{array}{l} \text{dimension(m)} =\text{obs dimension(m')} \wedge \text{fantasmas(m)} =\text{obs} \\ \text{fantasmas(m')} \wedge \text{paredes (m)} \\ =\text{obs paredes(m')} \wedge \text{puntoInicio(m)} =\text{obs puntoInicio(m')} \wedge \\ \text{puntoLlegada(m)} =\text{obs puntoLlegada(m')} \end{array} \right) \right)$$

observadores básicos

Error. Restricción U2 sobre parametros

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

fantasmas : mapa m $\longrightarrow$ conj(Coordenada) $\{\text{paredes(m)} \cap \text{fantasmas(m)} = \emptyset\}$

paredes : mapa m $\longrightarrow$ conj(Coordenada) $\{\text{paredes(m)} \cap \text{fantasmas(m)} = \emptyset\}$

puntoInicio : mapa m $\longrightarrow$ Coordenada $\{\neg(\text{puntoInicio} \in (\text{paredes(m)} \cup \text{fantasmas(m)}))\}$

puntoLlegada : mapa m $\longrightarrow$ Coordenada $\{\neg(\text{puntoLlegada} \in (\text{paredes(m)} \cup \text{fantasmas(m)}))\}$

Estas restricciones se deberían cumplir por los axiomas.

No es necesario ponerlo acá.

generadores

crearMapa : nat $tamX$ $\times$ nat $tamY$ $\times$ Coordenada $posInicio$ $\times$ Coordenada $posFin$ $\longrightarrow$ mapa

$$\left\{ \begin{array}{l} \neg(\text{posFin}=\text{posInicio}) \wedge \text{tamX} > 0 \wedge \text{tamY} > 0 \wedge 0 \leq \pi_1(\text{posInicio}) < \text{tamX} \\ \wedge\ 0 \leq \pi_2(\text{posInicio}) < \text{tamY} \wedge 0 \leq \pi_1(\text{posFin}) < \text{tamX} \wedge 0 \leq \pi_2(\text{posFin}) \\ < \text{tamY} \end{array} \right\}$$

agregarPared : mapa m $\times$ Coordenada $posPared$ $\longrightarrow$ mapa

$$\left\{ \begin{array}{l} \text{posPared} \in \text{coordenadasDelMapa(m)} \wedge \neg(\text{posPared} \in \text{fantasmas(m)}) \\ \wedge \neg(\text{posPared}=\text{puntoInicio(m)}) \wedge \neg(\text{posPared}=\text{puntoLlegada(m)}) \end{array} \right\}$$

agregarFantasma : mapa m $\times$ Coordenada $posFantasma$ $\longrightarrow$ mapa

$$\left\{ \begin{array}{l} \text{posFantasma} \in \text{coordenadasDelMapa(m)} \wedge \neg(\text{posFantasma} \in \text{paredes(m)}) \\ \wedge \neg(\text{posFantasma}=\text{puntoInicio(m)}) \wedge \neg(\text{posFantasma}=\text{puntoLlegada(m)}) \end{array} \right\}$$

otras operaciones

coordenadasDelMapa : mapa $\longrightarrow$ conj(Coordenada)

esCoordenadaValidaParaAvanzar? : mapa $\times$ Coordenada $\longrightarrow$ Bool

espaciosVacios : mapa m $\longrightarrow$ conj(Coordenada) $\{(\text{paredes(m)} \cup \text{fantasmas(m)}) \cap \text{espaciosVacios(m)} = \emptyset\}$ Idem arriba

generarEspaciosVacios : conj(Coordenada) × conj(Coordenada) × conj(Coordenada) ⟶ conj(Coordenada

posiblesCombinaciones : nat x × nat y ⟶ conj(Coordenada) {0 ≤ x ∧ 0 ≤ y}

posiblesCombinacionesAux : nat x × nat y ⟶ conj(Coordenada) {0 ≤ x ∧ 0 ≤ y}

axiomas

dimension(crearMapa(tX, tY, pI, pL)) ≡ <tX, tY>

dimension(AgregarPared(m, pos)) ≡ dimension(m)

dimension(AgregarFantasma(m, pos)) ≡ dimension(m)

fantasmas(crearMapa(tX, tY, pI, pL)) ≡ ∅

fantasmas(AgregarPared(m, pos)) ≡ fantasmas(m)

fantasmas(AgregarFantasma(m, pos)) ≡ Ag(pos, fantasmas(m))

paredes(crearMapa(tX, tY, pI, pL)) ≡ ∅

paredes(AgregarPared(m, pos)) ≡ Ag(pos, paredes(m))

paredes(AgregarFantasma(m, pos)) ≡ paredes(m)

puntoInicio(crearMapa(tX, tY, pI, pL)) ≡ pI

puntoInicio(AgregarPared(m, pos)) ≡ puntoInicio(m)

puntoInicio(AgregarFantasma(m, pos)) ≡ puntoInicio(m)

puntoLlegada(crearMapa(tX, tY, pI, pL)) ≡ pL

puntoLlegada(AgregarPared(m, pos)) ≡ puntoLlegada(m)

puntoLlegada(AgregarFantasma(m, pos)) ≡ puntoLlegada(m)

coordenadasDelMapa(m) ≡ posiblesCombinaciones(π_1(dimension(m)), π_2(dimension(m)))

```
posiblesCombinaciones(x,y) ≡ if y = 0 then
                                posiblesCombinacionesAux(x, y)
                             else
                                posiblesCombinacionesAux(x, y)
                                ∪ posiblesCombinaciones(x, y-1)
                             fi
```

```
posiblesCombinacionesAux(x, y) ≡ if x = 0 then
                                    <x, y>
                                 else
                                    Ag(<x,y>, posiblesCombinacionesAux(x-1, y))
                                 fi
```

esCoordenadaValidaParaAvanzar?(m, Coordenada) $\equiv$

(Coordenada $\in$ coordenadasDelMapa(m)) $\wedge$ $\neg$(Coordenada $\in$ paredes(m))

espaciosVacios(m) $\equiv$ generarEspaciosVacios(coordenadasDelMapa(m), paredes(m), fantasmas(m))

generarEspaciosVacios(coordenadasDelMapa, paredes, fantasmas) $\equiv$

if $\emptyset?(paredes) \wedge \emptyset?(fantasmas)$ **then**
  coordenadasDelMapa
else
  if $\emptyset?(paredes)$ **then**
    generarEspaciosVacios(coordenadasDelMapa - dameUno(fantasmas), paredes, sinUno(fantasmas))
  else
    if $\emptyset?(fantasmas)$ **then**
      generarEspaciosVacios(coordenadasDelMapa - dameUno(paredes), sinUno(paredes), fantasmas)
    else
      generarEspaciosVacios((coordenadasDelMapa - dameUno(paredes)) - dameUno(fantasmas), sinUno(paredes), sinUno(fantasmas))
    fi
  fi
fi

Podían usar resta de conjuntos

Existe la operacion resta en conjuntos

Fin TAD

TAD Partida

géneros partida

usa Mapa, Direccion, Coordenada, Conjunto, Bool, Secuencia

exporta géneros, observadores básicos, generadores, partidaTerminada?, posActual, fantasmaMasCercano

igualdad observacional

$$(\forall p, p' : \text{partida}) \left(p =_{\text{obs}} p' \iff \begin{pmatrix} \text{mapa(p)} =\text{obs mapa(p')} \wedge \text{movi-} \\ \text{mientos(p)} =\text{obs movimientos(p')} \end{pmatrix} \right)$$

observadores básicos

movimientos : partida $\longrightarrow$ secu(Coordenada) ¿Porqué observan la secuencia entera de movimientos?

mapa : partida $\longrightarrow$ mapa

Importa? ⇒ Tema de enunciado.

generadores

nuevaPartida : mapa m $\longrightarrow$ partida

No puede ir a otra operación ⇒ Rompe Congruencia.

moverse : partida $\times$ mapa $\times$ Direccion d $\longrightarrow$ partida
{d='ARRIBA'$\vee$ d='ABAJO'$\vee$ d='DERECHA'$\vee$ d='IZQUIERDA'}

Se podría restringir el generador (está en los axiomas).

otras operaciones

perdio? : partida $\longrightarrow$ Bool

gano? : partida $\longrightarrow$ Bool

partidaTerminada? : partida $\longrightarrow$ Bool

fantasmaMasCercano : partida $\longrightarrow$ Coordenada

calcularDistanciaAFantasma : Coordenada $\times$ Coordenada $\longrightarrow$ Nat

siguienteCoordenada : partida $\longrightarrow$ secu(Coordenada)

buscarFantasmaMasCercano : Coordenada $\times$ conj(Coordenada) $\times$ Coordenada $\longrightarrow$ Coordenada

posActual : partida $\longrightarrow$ Coordenada

axiomas

movimientos(nuevaPartida(m)) $\equiv$ <puntoInicio(m)>

movimientos(moverse(p, m, d)) $\equiv$ **if** $\neg$ partidaTerminada?(p) **then**
siguienteCoordenada(p) $\bullet$ movimientos(p)
else
movimientos(p)
fi

mapa(nuevaPartida(m)) $\equiv$ m

mapa(moverse(p, m, d)) $\equiv$ mapa(p)

siguienteCoordenada(p) $\equiv$

if d = 'ARRIBA'∧
esCoordenadaValidaParaAvanzar?((π_1(ult(movimientos(p))), π_2(ult(movimientos(p))) + 1))
then
<(π_1(ult(movimientos(p))), π_2(ult(movimientos(p))) + 1)>
else
if d = 'ABAJO'∧
esCoordenadaValidaParaAvanzar?((π_1(ult(movimientos(p))), π_2(ult(movimientos(p))) - 1))
then
<(π_1(ult(movimientos(p))), π_2(ult(movimientos(p))) - 1)>
else
if d = 'DERECHA'∧
esCoordenadaValidaParaAvanzar?((π_1(ult(movimientos(p))) + 1,
π_2(ult(movimientos(p))))) **then**
<(π_1(ult(movimientos(p))) + 1, π_2(ult(movimientos(p))))>
else
if d = 'IZQUIERDA'∧
esCoordenadaValidaParaAvanzar?((π_1(ult(movimientos(m))) - 1,
π_2(ult(movimientos(m))))) **then**
<(π_1(ult(movimientos(p))) -1, π_2(ult(movimientos(p))))>
else
<>
fi
fi
fi
fi

Se podría tener una función aparte que calcule la coordenada siguiente dada la dirección y aca queda un solo if.

Mientras se puede avanzar y calculo de movimiento

posActual(p) ≡ ult(movimientos(p))

perdio?(p) ≡ calcularDistanciaAFantasma(fantasmaMasCercano(p), posActual(p)) <3

gano?(p) ≡ posActual(p) = puntoLlegada(mapa(p))

partidaTerminada?(p) ≡ perdio?(p) ∨ gano?(p)

fantasmaMasCercano(p) ≡ buscarFantasmaMasCercano(posActual(p),
sinUno(fantasmas(mapa(p))), da-
meUno(fantasmas(mapa(p))))

calcularDistanciaAFantasma(coordenada1, coordenada2) ≡ $(\max(\pi_1(coordenada1), \pi_1(coordenada2) - min(\pi_1(coordenada1), \pi_1(coordenada2))) + (max(\pi_2(coordenada1), \pi_2(coordenada2) - min(\pi_2(coordenada1), \pi_2(coordenada2))))$

buscarFantasmaMasCercano(posActual, fantasmas, masCercano) ≡

if $\emptyset?(fantasmas)$ **then**
masCercano
else

if calcularDistanciaAFantasma(posActual, dameUno(fantasmas)) < calcularDistanciaA-Fantasma(posActual, masCercano) **then**
buscarFantasmaMasCercano(posActual, sinUno(fantasmas), dameUno(fantasmas))
else
buscarFantasmaMasCercano(posActual, sinUno(fantasmas), masCercano)
fi

fi

Fin TAD

Parte 2

TAD Mapa

géneros mapa

usa Tupla, Coordenada, Conjunto, Bool, Nat

exporta géneros, observadores básicos, generadores, coordenadasDelMapa, esCoordenadaValidaParaAvanzar, espaciosVacios

igualdad observacional

$$(\forall m, m' : \text{mapa}) \left(m =_{\text{obs}} m' \iff \left(\begin{array}{l} \text{dimension(m)} =\text{obs dimension(m')} \wedge \text{fantasmas(m)} =\text{obs fantasmas(m')} \wedge \text{paredes (m)} \\ =\text{obs paredes(m')} \wedge \text{puntoInicio(m)} =\text{obs puntoInicio(m')} \wedge \\ \text{puntoLlegada(m)} =\text{obs puntoLlegada(m')} \wedge \text{barrasDeChocolate(m)} =\text{obs barrasDeChocolate(m')} \end{array} \right) \right)$$

observadores básicos

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

fantasmas : mapa m $\longrightarrow$ conj(Coordenada) $\{\text{paredes(m)} \cap \text{fantasmas(m)} = \emptyset\}$

paredes : mapa m $\longrightarrow$ conj(Coordenada) $\{\text{paredes(m)} \cap \text{fantasmas(m)} = \emptyset\}$

puntoInicio : mapa m $\longrightarrow$ Coordenada $\{\neg(\text{puntoInicio} \in (\text{paredes(m)} \cup \text{fantasmas(m)}))\}$

puntoLlegada : mapa m $\longrightarrow$ Coordenada $\{\neg(\text{puntoLlegada} \in (\text{paredes(m)} \cup \text{fantasmas(m)}))\}$

barrasDeChocolate : mapa m $\longrightarrow$ conj(Coordenada) $\{\text{barrasDeChocolate(m)} \cap (\text{paredes(m)} \cup \text{fantasmas(m)}) = \emptyset\}$

Idem parte 1

generadores

crearMapa : nat $tamX$ $\times$ nat $tamY$ $\times$ Coordenada $posInicio$ $\times$ Coordenada $posFin$ $\longrightarrow$ mapa
$\left\{ \begin{array}{l} \neg(\text{posFin=posInicio}) \wedge \text{tamX} > 0 \wedge \text{tamY} > 0 \wedge 0 \leq \pi_1(\text{posInicio}) < \text{tamX} \\ \wedge\ 0 \leq \pi_2(\text{posInicio}) < \text{tamY} \wedge 0 \leq \pi_1(\text{posFin}) < \text{tamX} \wedge 0 \leq \pi_2(\text{posFin}) \\ < \text{tamY} \end{array} \right\}$

agregarPared : mapa m $\times$ Coordenada $posPared$ $\longrightarrow$ mapa
$\left\{ \begin{array}{l} \text{posPared} \in \text{coordenadasDelMapa(m)} \wedge \neg(\text{posPared} \in \text{fantasmas(m)}) \\ \wedge \neg(\text{posPared=puntoInicio(m)}) \wedge \neg(\text{posPared=puntoLlegada(m)}) \end{array} \right\}$

agregarFantasma : mapa m $\times$ Coordenada $posFantasma$ $\longrightarrow$ mapa
$\left\{ \begin{array}{l} \text{posFantasma} \in \text{coordenadasDelMapa(m)} \wedge \neg(\text{posFantasma} \in \text{paredes(m)}) \\ \wedge \neg(\text{posFantasma=puntoInicio(m)}) \wedge \neg(\text{posFantasma=puntoLlegada(m)}) \end{array} \right\}$

agregarBarraDeChocolate : mapa m $\times$ Coordenada $posBarraChocolate$ $\longrightarrow$ mapa
$\left\{ \begin{array}{l} \text{posBarraChocolate} \in \text{coordenadasDelMapa(m)} \wedge \neg(\text{posBarraChocolate} \in \\ \text{paredes(m)}) \wedge \neg(\text{posBarraChocolate} \in \text{fantasmas(m)}) \end{array} \right\}$

Me permite agregar paredes y fantasmas sobre barras de chocolate

Validar

otras operaciones

coordenadasDelMapa : mapa $\longrightarrow$ conj(Coordenada)

esCoordenadaValidaParaAvanzar? : mapa $\times$ Coordenada $\longrightarrow$ Bool

espaciosVacios : mapa m $\longrightarrow$ conj(Coordenada)
$\{(\text{paredes(m)} \cup \text{fantasmas(m)}) \cap \text{espaciosVacios(m)} = \emptyset\}$

generarEspaciosVacios : conj(Coordenada) $\times$ conj(Coordenada) $\times$ conj(Coordenada) $\longrightarrow$
{conj(Coordenada)}

posiblesCombinaciones : nat x $\times$ nat y $\longrightarrow$ conj(Coordenada) $\{0 \leq \text{x} \wedge 0 \leq \text{y}\}$

posiblesCombinacionesAux : nat x $\times$ nat y $\longrightarrow$ conj(Coordenada)
$\{0 \leq \text{x} \wedge 0 \leq \text{y}\}$

axiomas

dimension (crearMapa(tX, tY, pI, pL)) $\equiv$ $\langle tX, tY \rangle$

dimension (agregarPared(m, pos)) $\equiv$ dimension(m)

dimension (agregarFantasma(m, pos)) $\equiv$ dimension(m)

dimension (agregarBarraDeChocolate(m, pos)) $\equiv$ dimension(m)

barrasDeChocolate(crearMapa(tX, tY, pI, pL)) $\equiv$ $\emptyset$

barrasDeChocolate(AgregarPared(m, pos)) $\equiv$ barrasDeChocolate(m)

barrasDeChocolate(AgregarFantasma(m, pos)) $\equiv$ barrasDeChocolate(m)

barrasDeChocolate(AgregarBarraDeChocolate(m, pos)) $\equiv$ Ag(pos, barrasDeChocolate(m))

fantasmas(crearMapa(tX, tY, pI, pL)) $\equiv$ $\emptyset$

fantasmas(AgregarPared(m, pos)) $\equiv$ fantasmas(m)

fantasmas(AgregarFantasma(m, pos)) $\equiv$ Ag(pos, fantasmas(m))

fantasmas(AgregarBarraDeChocolate(m, pos)) $\equiv$ fantasmas(m)

paredes(crearMapa(tX, tY, pI, pL)) $\equiv$ $\emptyset$

paredes(AgregarPared(m, pos)) $\equiv$ Ag(pos, paredes(m))

paredes(AgregarFantasma(m, pos)) $\equiv$ paredes(m)

paredes(AgregarBarraDeChocolate(m, pos)) $\equiv$ paredes(m)

puntoInicio(crearMapa(tX, tY, pI, pL)) $\equiv$ pI

puntoInicio(AgregarPared(m, pos)) $\equiv$ puntoInicio(m)

puntoInicio(AgregarFantasma(m, pos)) $\equiv$ puntoInicio(m)

puntoInicio(AgregarBarraDeChocolate(m, pos)) $\equiv$ puntoInicio(m)

puntoLlegada(crearMapa(tX, tY, pI, pL)) $\equiv$ pL

puntoLlegada(AgregarPared(m, pos)) $\equiv$ puntoLlegada(m)

puntoLlegada(AgregarFantasma(m, pos)) $\equiv$ puntoLlegada(m)

puntoLlegada(AgregarBarraDeChocolate(m, pos)) $\equiv$ puntoLlegada(m)

coordenadasDelMapa(m) $\equiv$ posiblesCombinaciones($\pi_1(casilleros(m)), \pi_2(casilleros(m))$)

posiblesCombinaciones(x,y) $\equiv$ **if** $y = 0$ **then**
<0,0>
else
posiblesCombinacionesAux(x, y)
$\cup$ posiblesCombinaciones(x, y-1)
fi

posiblesCombinacionesAux(x, y) $\equiv$ **if** $x = 0$ **then**
<0, y>
else
Ag(<x,y>, posiblesCombinacionesAux(x-1, y))
fi

esCoordenadaValidaParaAvanzar?(m, coordenada) $\equiv$ (coordenada $\in$ $coordenadasDelMapa(m)) \wedge$ $\neg(coordenada \in paredes(m))$

espaciosVacios(m) $\equiv$ generarEspaciosVacios(coordenadasDelMapa(m), paredes(m), fantasmas(m))

generarEspaciosVacios(coordenadasDelMapa, paredes, fantasmas) $\equiv$

if $\emptyset?(paredes) \wedge \emptyset?(fantasmas)$ **then**
coordenadasDelMapa
else
if $\emptyset?(paredes)$ **then**
generarEspaciosVacios(coordenadasDelMapa - dameUno(fantasmas),
paredes, sinUno(fantasmas))
else
if $\emptyset?(fantasmas)$ **then**
generarEspaciosVacios(coordenadasDelMapa - dameUno(paredes),
sinUno(paredes), fantasmas)
else
generarEspaciosVacios((coordenadasDelMapa - dameUno(paredes))
- dameUno(fantasmas), sinUno(paredes), sinUno(fantasmas))
fi
fi
fi

Fin TAD

TAD Partida

géneros partida

usa Nat, Coordenada, Bool, Direccion, Secuencia, Conjunto, Mapa

exporta géneros, observadores básicos, generadores, partidaTerminada?, posActual, fantasmaMasCercano, puntajeDeLaPartida, movimientosDeInmunidadDisponibles,

igualdad observacional

$$(\forall p, p' : \text{partida}) \left(p =_{\text{obs}} p' \iff \begin{pmatrix} \text{mapa(p)} =\text{obs mapa(p')} \wedge \text{movi-} \\ \text{mientos(p)} =\text{obs movimientos(p')} \\ \wedge \quad \text{barrasDeChocolateComidas(p)} \\ =\text{obs barrasDeChocolateComi-} \\ \text{das(p')} \end{pmatrix} \right)$$

observadores básicos

movimientos : partida $\longrightarrow$ secu(Coordenada)

mapa : partida $\longrightarrow$ mapa

barrasDeChocolateComidas : partida $\longrightarrow$ conj(Coordenada)

generadores

nuevaPartida : mapa m $\longrightarrow$ partida

moverse : partida $\times$ mapa $\times$ Direccion d $\longrightarrow$ partida
{d='ARRIBA'$\vee$ d='ABAJO'$\vee$ d='DERECHA'$\vee$ d='IZQUIERDA'}

otras operaciones

perdio? : partida $\longrightarrow$ Bool

gano? : partida $\longrightarrow$ Bool

puedeMoverse? : partida $\longrightarrow$ Bool

esSiguienteCoordenadaChocolate? : partida $\longrightarrow$ Bool

partidaTerminada? : partida $\longrightarrow$ Bool

fantasmaMasCercano : partida $\longrightarrow$ Coordenada

agregarTodosLosFantasmas : Coordenada $\times$ conj(Coordenada) $\longrightarrow$ mapa

agregarTodasLasParedes : Coordenada $\times$ conj(Coordenada) $\longrightarrow$ mapa

agregarTodosLosChocolates : Coordenada $\times$ conj(Coordenada) $\times$ conj(Coordenada) $\longrightarrow$ mapa

calcularDistanciaAFantasma : Coordenada $\times$ Coordenada $\longrightarrow$ nat

siguienteCoordenada : partida $\longrightarrow$ secu(Coordenada)

buscarFantasmaMasCercano : Coordenada $\times$ conj(Coordenada) $\times$ Coordenada $\longrightarrow$ Coordenada

posActual : partida $\longrightarrow$ Coordenada

puntajeDeLaPartida : partida $\longrightarrow$ nat

movimientosDeInmunidadDisponibles : partida p $\longrightarrow$ nat
{$\neg$(partidaTerminada?(p))}

Movimientos ⇒ Rehacer todo donde se mueve este obs.

⇒ No se reentrega a priori

axiomas

movimientos(nuevaPartida(m)) ≡ <puntoInicio(m)>

movimientos(moverse(p, m, d)) ≡ **if** ¬ partidaTerminada?(p) **then**
siguienteCoordenada(p) • movimientos(p)
else
movimientos(p)
fi

mapa(nuevaPartida(m)) ≡ m

mapa(moverse(p, m, d)) ≡

Barras Iniciales
- comidas
= finales.

if puedeMoverse?(p) ∧ esSiguienteCoordenadaChocolate?(p) **then**
agregarTodosLosFantasmas(agregarTodasLasParedes(
agregarTodosLosChocolates(crearMapa(
π_1(dimension(mapa(p))), π_2(dimension(mapa(p))), puntoInicio(mapa(p)),
puntoLlegada(mapa(p)), barrasDeChocolate(mapa(p)), barrasDeChocolateComidas(p)),
paredes(mapa(p))), fantasmas(mapa(p))))
else
mapa(p)
fi

Esta axiomatización no es necesaria con el observador de barras de chocolate comidas

Queda Bien ✓

barrasDeChocolateComidas(nuevaPartida(m)) ≡

if puntoInicio ∈ barrasDeChocolate(m) **then** Ag(puntoInicio, ∅) **else** ∅ **fi**

barrasDeChocolateComidas(moverse(p,m,d)) ≡

if puedeMoverse?(p) ∧ esSiguienteCoordenadaChocolate?(p) **then**
Ag(prim(siguienteCoordenada(p)), barrasDeChocolateComidas(p))
else
barrasDeChocolateComidas(p)
fi

• Obs ⇒ Que la observación está disponible siempre. ✓

⇒ Es razonable distinguir los mapas ✓

⇒ mapa ⇒ Barras de Chocolate - Chocolates Comidos ✓

⇒ Usar mapa actual siempre

siguienteCoordenada(p, d) ≡

if d = 'ARRIBA'∧
esCoordenadaValidaParaAvanzar?((π1(ult(movimientos(p))), π2(ult(movimientos(p))) + 1))
then
<(π1(ult(movimientos(p))), π2(ult(movimientos(p))) + 1)>
else
if d = 'ABAJO'∧
esCoordenadaValidaParaAvanzar?((π1(ult(movimientos(p))), π2(ult(movimientos(p))) - 1))
then
<(π1(ult(movimientos(p))), π2(ult(movimientos(p))) - 1)>
else
if d = 'DERECHA'∧
esCoordenadaValidaParaAvanzar?((π1(ult(movimientos(p))) + 1,
π2(ult(movimientos(p))))) **then**
<(π1(ult(movimientos(p))) + 1, π2(ult(movimientos(p))))>
else
if d = 'IZQUIERDA'∧
esCoordenadaValidaParaAvanzar?((π1(ult(movimientos(p))) - 1,
π2(ult(movimientos(p))))) **then**
<(π1(ult(movimientos(p))) -1, π2(ult(movimientos(p))))>
else
<>
fi
fi
fi
fi

puedeMoverse?(p) ≡ $\neg partidaTerminada?(p)$

esSiguienteCoordenadaChocolate?(p) ≡ $\neg vacia?(siguienteCoordenada(p)) \wedge_L$
$(prim(siguienteCoordenada(p)) \in$
$barrasDeChocolate(mapa(p)))$

movimientosDeInmunidadDisponibles(nuevaPartida(m)) ≡
if puntoInicio(m) ∈ barrasDeChocolate(m) **then** 10 **else** 0 **fi**

Rompe Congruencia?
No, porque tenés
los movimientos

movimientosDeInmunidadDisponibles(moverse(p,m,d)) ≡
if ¬puedeMoverse?(p) **then**
movimientosDeInmunidadDisponibles(p)
else

(*) La re-entrega tiene que tener los errores corregidos.

```
    if vacia?(siguienteCoordenada(p)) then
        movimientosDeInmunidadDisponibles(p)
    else
        if prim(siguienteCoordenada(p)) ∈ barrasDeChocolate(mapa(p)) then
            10
        else
            if movimientosDeInmunidadDisponibles(p) >0 then
                movimientosDeInmunidadDisponibles(p) - 1
            else
                0
            fi
        fi
    fi
fi
```

```
    agregarTodosLosFantasmas(mapa, fantasmas) ≡
if ∅?(fantasmas) then
    mapa
else
    agregarTodosLosFantasmas(AgregarFantasma(mapa, dameUno(fantasmas)), sinUno(fantasmas)
fi
```

```
    agregarTodasLasParedes(mapa, paredes ≡
if ∅?(paredes) then
    mapa
else
    agregarTodasLasParedes(AgregarPared(mapa, dameUno(paredes)), sinUno(paredes))
fi
```

```
    agregarTodosLosChocolates(mapa, barrasDeChocolate, barrasDeChocolateComidas) ≡
if dameUno(barrasDeChocolate) ∈ barrasDeChocolateComidas then
    agregarTodosLosChocolates(mapa, sinUno(barrasDeChocolate), barrasDeChocolateComidas)
else
    agregarTodosLosChocolates(AgregarBarraDeChocolate(mapa, dameUno(barrasDeChocolate)), sinUno(barrasDeChocolate), barrasDeChocolateComidas)
fi
```

```
    posActual(p) ≡ ult(movimientos(p))
```

```
    perdio?(p) ≡ movimientosDeInmunidadDisponibles(p) = 0
                 ∧ calcularDistanciaAFantasma(fantasmaMasCercano(p), posActual(p))
                 < 3
```

```
    gano?(p) ≡ posActual(p) = puntoLlegada(mapa(p))
```

partidaTerminada?(p) $\equiv$ perdio?(p) $\vee$ gano?(p)

fantasmaMasCercano(p) $\equiv$ buscarFantasmaMasCercano(posActual(p),
sinUno(fantasmas(mapa(p))),
dameUno(fantasmas(mapa(p))))

calcularDistanciaAFantasma(coordenada1, coordenada2) $\equiv$ (max(π_1(coordenada1),
π_1(coordenada2)) -
min(π_1(coordenada1),
π_1(coordenada2))) +
(max(π_2(coordenada1),
π_2(coordenada2)) -
min(π_2(coordenada1),
π_1(coordenada2)))

buscarFantasmaMasCercano(posActual, fantasmas, masCercano) $\equiv$

if $\emptyset?(fantasmas)$ **then**
masCercano
else
if calcularDistanciaAFantasma(posActual, dameUno(fantasmas)) $<$ calcularDistanciaAFantasma(posActual, masCercano) **then**
buscarFantasmaMasCercano(posActual, sinUno(fantasmas), dameUno(fantasmas))
else
buscarFantasmaMasCercano(posActual, sinUno(fantasmas), masCercano)
fi

fi

puntajeDeLaPartida(p) $\equiv$ long(movimientos(p)) - 1

Fin TAD