DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Especificación SimCity

Algoritmos y Estructuras de Datos II

Integrante	LU	Correo electrónico
Sosa, Patricio	218/16	patriciososa91@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

1. Especificación SimCity

Usaremos los siguientes tipos auxiliares:

El TAD pos representa un casillero valido en la grilla, representado como <nat,nat>.

EL TAD dir representa una dirección, horizontal = 1, vertical = 0

TAD SimCity

usa mapa

igualdad observacional

$$(\forall\ s1, s2 : \text{SimCity})\ \left(s1 =_{\text{obs}} s2 \iff \begin{pmatrix} ((\forall\ \text{p: pos})(\text{ocupada?(p,s1)} =_{\text{obs}} \text{ocupada?(p,s2)})) \wedge \\ (casas(s1) =_{\text{obs}} casas(s2)) \wedge \\ (comercios(s1) =_{\text{obs}} comercios(s2)) \wedge \\ (nivel(s1) =_{\text{obs}} nivel(s2)) \wedge \\ sigTurnoHabilitado?(s1) =_{\text{obs}} \\ sigTurnoHabilitado?(s2) \end{pmatrix} \right)$$

generadores

nuevaPartida	: mapa	⟶	SimCity	
agregarCasa	: pos p × SimCity s	⟶	SimCity	{¬ ocupada?(p,s)}
agregarComercio	: pos p × SimCity s	⟶	SimCity	{¬ ocupada?(p,s)}
pasarTurno	: SimCity s	⟶	SimCity	{sigTurnoHabilitado(s)}

observadores básicos

ocupada?	: pos × SimCity	⟶	bool
casas	: SimCity	⟶	conj(pos)
comercios	: SimCity	⟶	conj(pos)
nivel	: SimCity	⟶	dicc(pos,nat)
sigTurnoHabilitado?	: SimCity	⟶	bool

otras operaciones

esConstruccion?	: pos × SimCity	⟶	bool	
maxNiv	: conj(pos) c × SimCity s	⟶	nat	{c ⊆ casas(s)}
busquedamanhattan	: pos p × conj(pos) c	⟶	conj(pos)	{esConstruccion?(p,s)}
esmanhattan?	: pos p × SimCity s	⟶	bool	{p ∈ comercios(s)}
redefinirNiveles	: dicc(pos,nat) d × conj(pos) c × Simcity s	⟶	dicc(pos,nat)	{c ⊆ casas(s) ∪ comercios(s)}
manhattan	: pos p1 × pos p2	⟶	nat	
nivelParticular	: SimCity s × pos p	⟶	nat	{esConstruccion?(p,s)}
esCasa?	: pos p × Simcity s	⟶	bool	{esConstruccion?(p,s)}
esComercio?	: pos p × SimCity s	⟶	bool	{esConstruccion?(p,s)}

axiomas $\forall m$: mapa, $\forall p, p1, p2$: pos, $\forall s$: SimCity, $\forall c$: conjunto

ocupada?(p,nuevaPartida(m))	≡	casillaOcupada?(m,p)
ocupada?(p1,agregarCasa(p2,s))	≡	**if** p1=p2 **then** True **else** ocupada?(p1,s) **fi**
ocupada?(p1,agregarComercio(p2,s))	≡	**if** p1=p2 **then** True **else** ocupada?(p1,s) **fi**
ocupada?(p,pasarTurno(s))	≡	ocupada?(p,s)
casas(nuevaPartida(m))	≡	∅
casas(agregarCasa(p,s))	≡	ag(p,casas(s))
casas(agregarComercio(p,s))	≡	casas(s)
casas(pasarTurno(s))	≡	casas(s)
comercios(nuevaPartida(m))	≡	∅
comercios(agregarCasa(p,s))	≡	comercios(s)
comercios(agregarComercio(p,s))	≡	ag(p,comercios(s))
comercios(pasarTurno(s))	≡	comercios(s)

nivel(nuevaPartida(m)) ≡ vacio
nivel(agregarCasa(p,s)) ≡ definir(p,1,nivel(s))
nivel(agregarComercio(p,s))) ≡ **if** esmanhattan?(p,s) **then**
definir(p,maxNiv(bosquedaManhattan(p,casas(s))),nivel(s))
else
definir(p,1,nivel(s))
fi
nivel(pasarTurno(s)) ≡ redefinirNiveles(nivel(s),claves(nivel(s))
sigTurnoHabilitado?(nuevaPartida(m)) ≡ false
sigTurnoHabilitado?(agregarCasa(p,s)) ≡ true
sigTurnoHabilitado?(agregarComercio(p,s)) ≡ true
sigTurnoHabilitado?(pasarTurno(s)) ≡ false
esConstruccion?(p,s) ≡ p ∈ casas(s) ∨ p ∈ comercios(s)
maxNiv(c,s) ≡ **if** ∅?(c) **then**
0
else
if obtener(dameUno(c),nivel(s)) > maxNiv(sinUno(c),s)
then
obtener(dameUno(c),nivel(s))
else
maxNiv(sinUno(c),s)
fi
fi.
busquedaManhattan(p,c) ≡ **if** ∅?(*c*) **then**
∅
else
if manhattan(p,dameUno(c)) ≤ 3 **then**
Ag(dameUno(c),busquedaManhattan(p,sinUno(c)))
else
busquedaManhattan(p,sinUno(c))
fi
fi
esManhattan?(p,c) ≡ ¬(∅?(*busquedaManhattan*(*p*, *c*)))
manhattan(p1,p2) ≡ $|\pi_1(p2) - \pi_1(p1)| + |\pi_2(p2) - \pi_2(p1)|$
redefinirNiveles(d,c,s) ≡ **if** ∅?(*c*) **then**
∅
else
definir(dameUno(c),obtener(dameUno(c),d)+1
,redefinirNiveles(d,sinUno(c),s))
fi
nivelParticular(s,p) ≡ obtener(p,nivel(s))
esCasa?(p,s) ≡ p ∈ casas(s)
esComercio?(p,s) ≡ p ∈ comercios(s)

Fin TAD

TAD MAPA

usa secuencia,conjunto,nat

igualdad observacional

$$(\forall\ m1, m2 : \text{mapa})\ \left(m1 =_{\text{obs}} m2 \iff \left(\begin{array}{l}((\forall\ \text{p: pos}) \\ (\text{casillaOcupada?(p,s1)} =_{\text{obs}} \text{casillaOcupada?(p,s2)}))\end{array}\right)\right)$$

generadores

nuevoMapa : conj(<nat,dir>) $\longrightarrow$ mapa

observadores básicos

casillaOcupada? : mapa m $\times$ pos p $\longrightarrow$ bool

otras operaciones

axiomas

casillaOcupada?(nuevoMapa(c),p) $\equiv$ **if** $\emptyset$?(c) **then**
false
else
if $\pi1$(p) = $\pi1$(dameUno(c)) $\wedge\pi2$(dameUno(c)) = 1 **then**
true
else
if $\pi2$(p) = $\pi1$(dameUno(c)) $\wedge\pi2$(dameUno(c)) = 0 **then**
true
else
casillaOcupada?(nuevoMapa(sinUno(c),p))
fi.
fi.
fi.

Fin TAD

2. Decisiones Tomadas

2.1. Tad Mapa

EL TAD MAPA, modela la grilla del juego.Decidimos que la grilla se representa con un tupla$< nat, nat >$ infinita, es decir cualquiera combinación de naturales es valida, sin tener en cuenta si esta ocupada o no.

Generadores

NuevaMapa: Crea una nueva instancia de un mapa, que toma como parámetro un conjunto de tuplas que indica las posiciones iniciales desde donde se extienden los ríos

Observadores

casillaOcupada: Verifica si cada posición p pasada como parámetro es un río o no.

TAD SimCity Extended

igualdad observacional

$$(\forall\ s1, s2 : \text{SimCity})\ \left(s1 =_{\text{obs}} s2 \iff \begin{pmatrix} \text{popularidad(s1)} =_{\text{obs}} \text{popularidad(s2)} \wedge \\ constrMaxNivel(s1) =_{\text{obs}} \\ constrMaxNivel(s2) \end{pmatrix} \right)$$

usa SimCity

géneros simCity

generadores

agregarSimCity : simCity s1 × SimCity s2 ⟶ SimCity

$$\left\{ \begin{array}{l} \text{estanDesocupadas?(casas(s1) } \cup \text{ comercios(s1) } \cup \text{ casas(s2) } \cup \text{ comercios(s2))} \wedge \\ \text{constrMaxNivel(s1)} \cap (\text{casas(s2)} \cup \text{comercios(s2)}) = \emptyset \wedge \\ \text{constrMaxNivel(s2)} \cap (\text{casas(s1)} \cup \text{comercios(s1)}) = \emptyset \end{array} \right\}$$

observadores básicos

popularidad : SimCity ⟶ nat

constrMaxNivel : SimCity ⟶ conj(pos)

otras operaciones

estanDesocupadas? : conj(pos) × SimCity ⟶ bool

reordenar : dicc(pos,nat) × dicc(pos,nat) ⟶ dicc(pos,nat)

antiguedad : SimCity ⟶ nat

filtrarComercios : conj(pos) × simCity × simCity ⟶ conj(pos)

filtrarCasas : conj(pos) × simCity × simCity ⟶ conj(pos)

axiomas

ocupada?(pos, agregarSimCity(s1,s2)) ≡ ocupada?(pos,s1) v ocupada?(pos,s2)

casas(agregarSimCity(s1,s2)) ≡ **if** casas(s1) ∩ comercios(s2) = ∅ ∧ casas(s2) ∩ comercios(s1) = ∅ **then**
casas(s1) ∪ casas(s2)
else
casas(s1) ∪ casas(s2) -
filtrarComercios(casas(s1)∪casas(s2)∩comercios(s1)∪comercios(s2),s1,s2)
fi.

comercios(agregarSimCity(s1,s2)) ≡ **if** casas(s1) ∩ comercios(s2) = ∅ ∧ casas(s2) ∩ comercios(s1) = ∅ **then**
comercios(s1) ∪ comercios(s2)
else
comercios(s1) ∪ comercios(s2) -
filtrarCasas(casas(s1)∪casas(s2)∩comercios(s1)∪comercios(s2),s1,s2)
fi.

nivel(agregarSimCity(s1,s2)) ≡ reordenar(nivel(s1),nivel(s2))

sigTurnoHabilitado?(agregarSimCity(s1,s2)) ≡ false

popularidad(nuevaPartida(m)) ≡ 0

popularidad(agregarCasa(p,s)) ≡ popularidad(s)

popularidad(agregarComercio(p,s)) ≡ popularidad(s))

popularidad(pasarTurno(s)) ≡ popularidad(s))

popularidad(agregarSimCity(s1,s2)) ≡ popularidad(s)+1

constrMaxNivel(nuevaPartida(m)) ≡ ∅

constrMaxNivel(agregarCasa(p,s)) ≡ **if** antiguedad(s)=0 **then**
ag(p,constrMaxNivel(s))
else
contrMaxNivel(s)
fi.

constrMaxNivel(agregarComercio(p,s)) ≡ **if** antiguedad(s)=0 **then**
ag(p,constrMaxNivel(s))
else
contrMaxNivel(s)
fi.

constrMaxNivel(pasarTurno(p,s)) ≡ contrMaxNivel(s)

```
constrMaxNivel(agregarSimCity(s1,s2)) ≡ if antiguedad(s1)=antiguedad(s2) then
                                            constrMaxNivel(s1) ∪ constrMaxNivel(s2)
                                        else
                                            if antiguedad(s1)<antiguedad(s2) then
                                                constrMaxNivel(s2)
                                            else
                                                constrMaxNivel(s1)
                                            fi.
                                        fi.
estanDesocupadas?(c,s) ≡ if ∅?(c) then
                             true
                             else if ocupada?(dameUno(c),s) then
                                 False
                             else
                                 estanDesocupadas?(sinUno(c),s) fi.
                             fi.
reordenar(d1,d2) ≡ if ∅?(claves(d2)) then
                       d1

                   else
                       if def?(dameUno(claves(d2)),d1) ∧ def?(dameUno(claves(d2)),d2) then
                           if obtener(dameUno(claves(d2)),d1)≤obtener(dameUno(claves(d2)),d2) then
                               reordenar(definir(dameUno(claves(d2)),obtener(dameUno(claves(d2)),d2),d1),
                               borrar(dameUno(claves(d2)),d2))

                           else
                               reordenar(d1,borrar(dameUno(claves(d2)),d2))
                           fi.
                       else
                           reordenar(definir(dameUno(claves(d2)),obtener(dameUno(claves(d2)),d2),d1),
                           borrar(dameUno(claves(d2)),d2))

                       fi.
                   fi.
antiguedad(s) ≡ maxNiv(claves(nivel(s)),s)
filtrarComercio(c,s1,s2) ≡ if vacio?(c) then
                               ∅
                           else
                               if dameUno?(c) ∈ comercios(s1) then
                                   if nivelParticular(dameUno(c),s2) > nivelParticular(dameUno(c),s1) then
                                       filtrarComercios(sinUno(c),s1,s2)
                                   else
                                       ag(dameUno(c),filtrarComercios(sinUno(c),s1,s2)
                                   fi.
                               else
                                   if nivelParticular(dameUno(c),s1) > nivelParticular(dameUno(c),s2) then
                                       filtrarComercios(sinUno(c),s1,s2)
                                   else
                                       ag(dameUno(c),filtrarComercios(sinUno(c),s1,s2)
                                   fi.
                               fi.
                           fi.
```

```
filtrarCasas(c,s1,s2) ≡ if vacio?(c) then
                            ∅
                        else
                            if dameUno?(c) ∈ casas(s1) then
                                if nivelParticular(dameUno(c),s2) ≥ nivelParticular(dameUno(c),s1) then
                                    filtrarCasas(sinUno(c),s1,s2)
                                else
                                    ag(dameUno(c),filtrarCasas(sinUno(c),s1,s2)
                                fi.
                            else
                                if nivelParticular(dameUno(c),s1) ≥ nivelParticular(dameUno(c),s2) then
                                    filtrarCasas(sinUno(c),s1,s2)
                                else
                                    ag(dameUno(c),filtrarCasas(sinUno(c),s1,s2)
                                fi.
                            fi.
                        fi.
```

Fin TAD

3. Decisiones Tomadas

Extensión del tad SimCity

Al tad SimCity original se agregan todas extensiones provistas en en el tad SIMCITY EXTENDED

Unión De SimCitys: Si al crear una nueva partida colaborativa,la construcción de máximo nivel de un simcity se solapa con otra , no se puede crear dicha partida

Casa vs Casa,Comercio vs Comercio,Casa vs Comercio: Si coinciden dos construcciones en una misma posición se reemplaza por la de mayor nivel, con excepción de que si la construcción a reemplazar es la de mayor nivel de alguno de los SimCitys