DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

TP de Especificación y Diseño

Modelado de SimCity

1 de Junio de 2022 — Algoritmos y Estructuras de Datos II

Grupo 01 - hasTADlaVista, turno mañana

Integrante	LU	Correo electrónico
Lakowsky, Manuel	511/21	mlakowsky@gmail.com
Vekselman, Natán	338/21	COMPLETAR@gmail.com
Arienti, Federico	316/21	fa.arienti@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

1. Especificación

TAD MAPA

igualdad observacional

$$(\forall m, m' : \text{Mapa}) \left(m =_{\text{obs}} m' \iff \begin{pmatrix} \text{horizontales}(m) =_{\text{obs}} \text{horizontales}(m') \wedge_{\text{L}} \\ \text{verticales}(m) =_{\text{obs}} \text{verticales}(m') \end{pmatrix} \right)$$

géneros Mapa

exporta completar

usa completar

observadores básicos

horizontales : Mapa $\longrightarrow$ conj(Nat)
verticales : Mapa $\longrightarrow$ conj(Nat)

Mapa

generadores

crear : conj(Nat) $\times$ conj(Nat) $\longrightarrow$ Mapa

axiomas $\forall hs, vs$: conj(Nat)

horizontales(crear(hs, vs)) $\equiv$ hs
verticales(crear(hs, vs)) $\equiv$ vs

Fin TAD

TAD SimCity

igualdad observacional

$$(\forall s, s' : \text{SimCity}) \left(s =_{\text{obs}} s' \iff \begin{pmatrix} \text{mapa}(s) =_{\text{obs}} \text{mapa}(s') \wedge_{\text{L}} \\ \text{casas}(s) =_{\text{obs}} \text{casas}(s') \wedge \\ \text{comercios}(s) =_{\text{obs}} \text{comercios}(s') \wedge \\ \text{popularidad}(s) =_{\text{obs}} \text{popularidad}(s') \end{pmatrix} \right)$$

géneros SimCity

exporta completar

usa completar

observadores básicos

mapa	: SimCity	$\longrightarrow$	Mapa
casas	: SimCity	$\longrightarrow$	dicc(Pos, Nivel)
comercios	: SimCity	$\longrightarrow$	dicc(Pos, Nivel)
popularidad	: SimCity	$\longrightarrow$	Nat

generadores

iniciar	: Mapa	$\longrightarrow$	SimCity
avanzarTurno	: SimCity s $\times$ dicc(Pos $\times$ Construccion) cs	$\longrightarrow$	SimCity $\left\{ \begin{array}{l} (\forall p : \text{Pos})\ (\text{def?}(p, cs) \Rightarrow_{\text{L}} (\neg\ p \in \text{claves}(\text{construcc}(s)) \wedge \\ \neg\pi_0(p) \in \text{horizontales}(\text{mapa}(s)) \wedge \neg\pi_1(p) \in \text{verticales}(\text{mapa}(s)) \wedge \\ (\text{obtener}(p, cs) =_{\text{obs}} 1 \vee \text{obtener}(p, cs) =_{\text{obs}} 2))) \end{array} \right\}$
unir	: SimCity a $\times$ SimCity b	$\longrightarrow$	SimCity $\left\{ \begin{array}{l} (\forall p : \text{Pos})\ (\text{def?}(p, \text{construcc}(a)) \Rightarrow_{\text{L}} \\ (\neg\pi_0(p) \in \text{horizontales}(\text{mapa}(b)) \wedge \neg\pi_1(p) \in \text{verticales}(\text{mapa}(b)) \wedge \\ (p \in \text{masNivel}(a)\ \Rightarrow\ \neg p \in \text{construcc}(b)))) \wedge \\ (\forall p : \text{Pos})\ (\text{def?}(p, \text{construcc}(b)) \Rightarrow_{\text{L}} \\ (\neg\pi_0(p) \in \text{horizontales}(\text{mapa}(a)) \wedge \neg\pi_1(p) \in \text{verticales}(\text{mapa}(a)) \wedge \\ (p \in \text{masNivel}(b)\ \Rightarrow\ \neg p \in \text{construcc}(a)))) \end{array} \right\}$

otras operaciones

turnos	: SimCity	$\longrightarrow$	Nat
construcc	: SimCity	$\longrightarrow$	dicc(Pos, Nivel)
unirDicc	: dicc(Pos $\times$ Nivel) $\times$ dicc(Pos $\times$ Nivel)	$\longrightarrow$	dicc(Pos, Nivel)
masNivel	: SimCity	$\longrightarrow$	conj(Pos)
masNivelAux	: dicc(Pos $\times$ Nivel) $\times$ Nat	$\longrightarrow$	conj(Pos)
nivelMaximo	: dicc(Pos $\times$ Nivel)	$\longrightarrow$	Nat
agCasas	: dicc(Pos $\times$ Nivel) $\times$ dicc(Pos $\times$ Construccion)	$\longrightarrow$	dicc(Pos,Nivel)
agComercios	: dicc(Pos $\times$ Nivel) $\times$ dicc(Pos $\times$ Construccion)	$\longrightarrow$	dicc(Pos,Nivel)
nivelCom	: Pos $\times$ SimCity	$\longrightarrow$	Nat
conjManhatt	: Pos $\times$ dicc(Pos $\times$ Nivel)	$\longrightarrow$	dicc(Pos, Nivel)
distManhatt	: Pos $\times$ Pos	$\longrightarrow$	Nat
sacarRepes	: dicc(Pos $\times$ Construccion) $\times$ dicc(Pos $\times$ Construccion)	$\longrightarrow$	dicc(Pos, Construccion)

axiomas $\forall s, s'$: simcity, $\forall cs, cs'$: dicc(Pos, Construccion)

mapa(iniciar(m))	$\equiv$	m
mapa(avanzarTurno(s, cs))	$\equiv$	mapa(s)
mapa(unir(s, s'))	$\equiv$	crear(horizontales(s) $\cup$ horizontales(s'), verticales(s) $\cup$ verticales(s'))
casas(iniciar(m))	$\equiv$	vacio
casas(avanzarTurno(s, cs))	$\equiv$	agCasas(casas(s), cs)
casas(unir(s, s'))	$\equiv$	agCasas(casas(s), sacarRepes(construcc(s), construcc(s')))

agCasas(cs, cs') $\equiv$ **if** *vacio*?(claves(cs')) **then** cs **else**
if obtener(dameUno(claves(cs')), cs') $=_{\text{obs}}$ 1 **then**
agCasas(definir(dameUno(claves(cs')), 1, cs),
borrar(dameUno(claves(cs')), cs'))
else
agCasas(cs, borrar(dameUno(claves(cs')), cs'))
fi fi

comercios(iniciar(m)) $\equiv$ vacio

comercios(avanzarTurno(s, cs)) $\equiv$ agComercios(comercios(s), cs)

comercios(unir(s, s')) $\equiv$ unirConstrucc(s, casas(s, casas(s')))

agComercios(cs, cs') $\equiv$ **if** *vacio*?(claves(cs')) **then** cs **else**
if obtener(dameUno(claves(cs')), cs') $=_{\text{obs}}$ 2 **then**
agComercios(definir(dameUno(claves(cs')),
nivelCom(dameUno(claves(cs')), s), cs),
borrar(dameUno(claves(cs')), cs'))
else
agComercios(cs, borrar(dameUno(claves(cs')), cs'))
fi fi

nivelCom(p, s) $\equiv$ **if** $\neg$*vacio*?(manhattan(p, casas(s))) **then**
nivelMaximo(manhattan(p), casas(s))
else 1 **fi**

conjManhatt(p, cs) $\equiv$ **if***vacio*?(cs) **then** $\emptyset$ **else**
if distManhatt(p, dameUno(claves(cs))) ≤ 3 **then**
definir(dameUno(claves(cs)), obtener(dameUno(claves(cs)), cs),
conjManhatt(p, borrar(dameUno(claves(cs)), cs)))
else
conjManhatt(p, borrar(dameUno(claves(cs)), cs))
fi

distManhatt(p, q) $\equiv$ **if** $\pi_0(p) < \pi_0(q)$ **then** $q - p$ **else** $p - q$ **fi**
$+$
if $\pi_1(p) < \pi_1(q)$ **then** $q - p$ **else** $p - q$ **fi**

popularidad(iniciar(m)) $\equiv$ 0

popularidad(avanzarTurno(s, cs)) $\equiv$ popularidad(s)

popularidad(unir(s, s')) $\equiv$ popularidad(s) + 1

turnos(iniciar(m)) $\equiv$ 0

turnos(avanzarTurno(s, cs)) $\equiv$ turnos(s) + 1

turnos(unir(s, s')) $\equiv$ **if** turnos(s) $<$ turnos(s') **then** turnos(s') **else** turnos(s) **fi**

construcc(s) $\equiv$ unirDicc(casas(s), comercios(s))

unirDicc(cs, cs') $\equiv$ **if** *vacio*?(claves(cs')) **then** cs **else**
definir(dameUno(claves(cs')),
obtener(dameUno(claves(cs')), cs'),
unirDicc(cs, borrar(dameUno(claves(cs')), cs'))) **fi**

masNivel(s) $\equiv$ masNivelAux(construcc(s), nivelMaximo(construcc(s)))

masNivelAux(cs, n) $\equiv$ **if** *vacio*?(cs) **then** $\emptyset$ **else**
if obtener(dameUno(claves(cs)), cs) $=_{\text{obs}}$ n **then**
ag(dameUno(claves(cs)),
masNivelAux(borrar(dameUno(claves(cs)), cs), n))
else
masNivelAux(borrar(dameUno(claves(cs)), cs), n)
fi fi

nivelMaximo(cs) $\equiv$ **if** *vacio*?(cs) **then** 0 **else**
max(obtener(dameUno(claves(cs)), cs),
nivelMaximo(borrar(dameUno(claves(cs)), cs)))

sacarRepes(cs, cs') $\equiv$ **if** *vacio*?(claves(cs)) **then** cs' **else**
if def?(dameUno(claves(cs)), cs') **then**
sacarRepes(borrar(dameUno(claves(cs)), cs),
borrar(dameUno(claves(cs)), cs'))
else
sacarRepes(borrar(dameUno(claves(cs)), cs), cs')
fi fi

Fin TAD

TAD SERVIDOR

géneros server

exporta observadores, generadores, verMapa, verCasas, verComercios, verPopularidad y verTurno

usa SimCity, Mapa, Nombre, Pos, Construccion, Nivel, Nat, bool, dicc(α, β), conj(α)

igualdad observacional

$$(\forall s, s' : \text{server}) \left(s =_{\text{obs}} s' \iff \begin{pmatrix} \text{partidas}(s) =_{\text{obs}} \text{partidas}(s') \wedge \\ \text{congeladas}(s) =_{\text{obs}} \text{congeladas}(s') \end{pmatrix} \right)$$

observadores básicos

partidas : server $\longrightarrow$ dicc(Nombre, SimCity)
congeladas : server $\longrightarrow$ conj(Nombre)

generadores

nuevoServer : $\longrightarrow$ server
nuevaPartida : server s × Nombre p × Mapa $\longrightarrow$ server $\{\neg def?(p, partidas(s))\}$
unirPartidas : server s × Nombre p1 × Nombre p2 $\longrightarrow$ server $\{*unionValida(s, p1, p2, cs)\}$
avanzarTurnoPartida : server s × Nombre p × dicc(Pos × Construccion) cs $\longrightarrow$ server $\{*avanzarTurnoValido(s, p, cs)\}$

otras operaciones

verMapa : server s × Nombre p $\longrightarrow$ Mapa $\{def?(p, partidas(s))\}$
verCasas : server s × Nombre p $\longrightarrow$ dicc(Pos, Nivel) $\{def?(p, partidas(s))\}$
verComercios : server s × Nombre p $\longrightarrow$ dicc(Pos, Nivel) $\{def?(p, partidas(s))\}$
verPopularidad : server s × Nombre p $\longrightarrow$ Nat $\{def?(p, partidas(s))\}$
verTurno : server s × Nombre p $\longrightarrow$ Nat $\{def?(p, partidas(s))\}$

axiomas $\forall s$: server, $\forall p$, $p1$, $p2$: Nombre, $\forall m$: Mapa, $\forall cs$: conj(Pos), $\forall a$, b: dicc(Pos, Nivel)

partidas(nuevoServer) $\equiv$ $vacio$
partidas(nuevaPartida(s, p, m)) $\equiv$ $definir(p, iniciar(m), partidas(s))$
partidas(unirPartidas(s, p1, p2)) $\equiv$ $definir(p1, unir(obtener(p1, partidas(s)), obtener(p2, partidas(s))), partidas(s))$
partidas(avanzarTurnoPartida(s, p, cs)) $\equiv$ $definir(p, avanzarTurno(obtener(p, partidas(s)), cs), partidas(s))$
congeladas(nuevaPartida) $\equiv$ $\emptyset$
congeladas(nuevaPartida(s, p, m)) $\equiv$ $congeladas(s)$
congeladas(avanzarTurnoPartida(s, p, cs)) $\equiv$ $congeladas(s)$
congeladas(unirPartidas(s, p1, p2)) $\equiv$ $Ag(p2, congeladas(s))$
// *oo*
verMapa(s, p) $\equiv$ $mapa(obtener(p, partidas(s)))$
verCasas(s, p) $\equiv$ $casas(obtener(p, partidas(s)))$
verComercios(s, p) $\equiv$ $comercios(obtener(p, partidas(s)))$
verPopularidad(s, p) $\equiv$ $popularidad(obtener(p, partidas(s)))$
verTurno(s, p) $\equiv$ $turnos(obtener(p, partidas(s)))$

Fin TAD

*donde:

unionValida : server s $\times$ Nombre p1 $\times$ Nombre p2 $\longrightarrow$ boolean
unionValida(s, p1, p2) $\equiv$ $def?(p,\ partidas(s)) \wedge$
$p \notin congeladas(s) \wedge$
$\neg vacia?(claves(cs)) \wedge_{\text{L}}$
$(\forall pos : Pos)(pos \in claves(cs) \Rightarrow_{\text{L}}$
$\neg sobreRio(pos,\ mapa(sim)) \wedge$
$\neg def?(pos,\ casas(sim)) \wedge$
$\neg def?(pos,\ comercios(sim))$
)
donde $sim \equiv obtener(p,\ partidas(s))$

avanzarTurnoValido : server s $\times$ Nombre p $\times$ dicc(Pos $\times$ Construccion) cs $\longrightarrow$ boolean
avanzarTurnoValido(s, p, cs) $\equiv$ $def?(p1,\ partidas(s)) \wedge def?(p2,\ partidas(s)) \wedge$
$p1 \notin congeladas(s) \wedge p2 \notin congeladas(s) \wedge_{\text{L}}$
$(\forall\ pos : Pos)(pos \in claves(constr1) \Rightarrow_{\text{L}}$
$\neg sobreRio(pos,\ sim2) \wedge$
$((\nexists\ otra : Pos)(otra \in constr1 \wedge_{\text{L}}$
$obtener(pos,\ constr1).nivel < obtener(otra,\ constr1).nivel$
$) \Rightarrow_{\text{L}} \neg def?(pos,\ constr2))$
$) \wedge$
$(\forall\ pos : Pos)(pos \in claves(constr2) \Rightarrow_{\text{L}}$
$\neg sobreRio(pos,\ sim1) \wedge$
$((\nexists otra : Pos)(otra \in constr2 \wedge_{\text{L}}$
$obtener(pos,\ constr2).nivel < obtener(otra,\ constr2).nivel$
$) \Rightarrow_{\text{L}} \neg def?(pos,\ constr1))$
)
donde $sim1 \equiv obtener(p1,\ partidas(s))$,
$sim2 \equiv obtener(p2,\ partidas(s))$,
$constr1 \equiv casas(sim1) \cup comercios(sim1)$,
$constr2 \equiv casas(sim2) \cup comercios(sim2)$

$\bullet \cup \bullet$: dicc($\alpha \times \beta$) $\times$ dicc($\alpha \times \beta$) $\longrightarrow$ dicc(α, β)
$\bullet \cup \bullet \equiv \_definir(a,\ b,\ claves(b))$

_definir : dicc($\alpha \times \beta$) $\times$ dicc($\alpha \times \beta$) b $\times$ conj(α) cs $\longrightarrow$ dicc(α, β) $\{cs \subseteq claves(b)\}$
_definir(a, b, cs) $\equiv$ **if** $vacio?(cs)$ **then**
a
else
$\_definir(definir(dameUno(cs), obtener(dameUno(cs), b)), b, sinUno(cs))$
fi

2. Módulos de referencia

2.1. Módulo Mapa

Interfaz

se explica con: MAPA

géneros: `mapa`

TP de Especificación y DiseñoOperaciones básicas de mapa

CREAR(**in** hs: `conj(Nat)`,**in** vs: `conj(Nat)`) $\rightarrow$ res : `mapa`
Pre $\equiv$ {true}
Post $\equiv$ $\{res =_{\text{obs}} mapa(hs, vs)\}$
Complejidad: $O(copy(hs), copy(vs))$
Descripción: crea un mapa

completar

Representación

TP de Especificación y DiseñoRepresentación de mapa

Un mapa contiene rios infinitos horizontales y verticales. Los rios se representan como conjuntos lineales de naturales que indican la posición en los ejes de los ríos.

`mapa` **se representa con** `estr`

donde `estr` es `tupla`(*horizontales*: `conj(Nat)`, *verticales*: `conj(Nat)`)

Rep : estr $\longrightarrow$ bool
Rep(e) $\equiv$ true $\iff$ true

Abs : estr m $\longrightarrow$ mapa $\{\text{Rep}(m)\}$
Abs(m) $\equiv$ horizontales(m) $=$ $estr$.horizontales $\wedge$ verticales(m) $=$ $estr$.verticales

Algoritmos

---

crear(**in** hs: `conj(Nat)`, **in** vs: `conj(Nat)`) $\rightarrow$ res : estr
1: $estr.horizontales \leftarrow hs$
2: $estr.verticales \leftarrow vs$ **return** estr

Complejidad: $O(copy(hs) + copy(vs))$

---

completar