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))) $\leq$ 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)) ≡ popularidad(s)
popularidad(unir(s, s')) ≡ popularidad(s) + 1
turnos(iniciar(m)) ≡ 0
turnos(avanzarTurno(s, cs)) ≡ turnos(s) + 1
turnos(unir(s, s')) ≡ **if** turnos(s) < turnos(s') **then** turnos(s') **else** turnos(s) **fi**
construcc(s) ≡ unirDicc(casas(s), comercios(s))
unirDicc(cs, cs') ≡ **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) ≡ masNivelAux(construcc(s), nivelMaximo(construcc(s)))
masNivelAux(cs, n) ≡ **if** *vacio*?(cs) **then** ∅ **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) ≡ **if** *vacio*?(cs) **then** 0 **else**
max(obtener(dameUno(claves(cs)), cs),
nivelMaximo(borrar(dameUno(claves(cs)), cs)))
sacarRepes(cs, cs') ≡ **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

2. Módulos de referencia

2.1. Módulo Mapa

Interfaz

se explica con: MAPA

géneros: `mapa`

Operaciones 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

Representación 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 $\Longleftrightarrow$ 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