



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	natanvek11@gmail.com
Arienti, Federico	316/21	fa.arianti@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

1.1. Mapa

TAD MAPA

igualdad observacional

$$(\forall m, m' : \text{Mapa}) \left(m =_{\text{obs}} m' \iff \left(\text{horizontales}(m) =_{\text{obs}} \text{horizontales}(m') \wedge_L \text{verticales}(m) =_{\text{obs}} \text{verticales}(m') \right) \right)$$

géneros Mapa

exporta Mapa, observadores, generadores, $\bullet + \bullet$, esRio

usa Nat, conj(a), Pos, Bool

observadores básicos

horizontales : Mapa \rightarrow conj(Nat)

verticales : Mapa \rightarrow conj(Nat)

Mapa

generadores

crear : conj(Nat) \times conj(Nat) \rightarrow Mapa

otras operaciones

Mapa

$\bullet + \bullet$: Mapa \times Mapa \rightarrow Mapa

esRio : Mapa \times Pos \rightarrow Bool

axiomas $\forall hs, vs : \text{conj}(\text{Nat}), \forall m1, m2 : \text{Mapa}, \forall p : \text{Pos}$

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

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

$m1 + m2 \equiv \text{crear}(\text{horizontales}(m1) \cup \text{horizontales}(m2), \text{verticales}(m1) \cup \text{verticales}(m2))$

$\text{esRio}(m1, p) \equiv p.x \in \text{verticales}(m1) \vee p.y \in \text{horizontales}(m1)$

Fin TAD

1.2. SimCity

TAD SIMCITY

igualdad observacional

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

géneros SimCity

exporta SimCity, observadores, generadores, turnos

usa Mapa, Nat, Pos, Construcccion, $\text{dicc}(\alpha, \beta)$, Nivel

observadores básicos

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

generadores

iniciar : Mapa \longrightarrow SimCity
avanzarTurno : SimCity $s \times \text{dicc}(\text{Pos} \times \text{Construcccion})$ $cs \longrightarrow$ SimCity $\{*\text{avanzarTurnoValido}(s, cs)\}$
unir : SimCity $a \times \text{SimCity } b \longrightarrow$ SimCity $\{*\text{unirValido}(a, b)\}$

otras operaciones

turnos : SimCity \longrightarrow Nat
construcciones : SimCity \longrightarrow $\text{dicc}(\text{Pos}, \text{Nivel})$
• \cup_{dicc} • : $\text{dicc}(\alpha \times \beta) \times \text{dicc}(\alpha \times \beta) \longrightarrow$ $\text{dicc}(\alpha, \beta)$
agCasas : $\text{dicc}(\text{Pos} \times \text{Nivel}) \times \text{dicc}(\text{Pos} \times \text{Construcccion}) \longrightarrow$ $\text{dicc}(\text{Pos}, \text{Nivel})$
agComercios : $\text{SimCity} \times \text{dicc}(\text{Pos} \times \text{Nivel}) \times \text{dicc}(\text{Pos} \times \text{Construcccion}) \longrightarrow$ $\text{dicc}(\text{Pos}, \text{Nivel})$
nivelCom : $\text{Pos} \times \text{dicc}(\text{Pos} \times \text{Nivel}) \longrightarrow$ Nat
distManhatt : $\text{Pos} \times \text{Pos} \longrightarrow$ Nat
sacarRepes : $\text{dicc}(\text{Pos} \times \text{Construcccion}) \times \text{dicc}(\text{Pos} \times \text{Construcccion}) \longrightarrow$ $\text{dicc}(\text{Pos}, \text{Construcccion})$

axiomas $\forall s, s' : \text{simcity}, \forall cs, cs' : \text{dicc}(\text{Pos}, \text{Construcccion}), \forall cn, cn' : \text{dicc}(\text{Pos}, \text{Nivel}), \forall d, d' : \text{dicc}(\alpha, \beta)$

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

```

comercios(iniciar(m))           ≡ vacío
comercios(avanzarTurno(s, cs))  ≡ agComercios(s, comercios(s), cs)
comercios(unir(s, s'))          ≡ agComercios(s,
                                comercios(s),
                                sacarRepes(construcciones(s), construcciones(s')))
agComercios(s, cn, cs) ≡ if vacío?(claves(cs)) then
  cn
else
  if obtener(dameUno(claves(cs)), cs) =obs "comercio" then
    agComercios(definir(dameUno(claves(cs)),
                        nivelCom(dameUno(claves(cs)), casas(s)), cn),
                borrar(dameUno(claves(cs)), cs))
  else
    agComercios(cn, borrar(dameUno(claves(cs)), cs))
  fi
fi
nivelCom(p, cn) ≡ if vacío?(claves(cn)) then
  1
else
  if distManhatt(p, dameUno(claves(cn))) ≤ 3 then
    max(obtener(dameUno(claves(cn)), cn),
        nivelCom(p, borrar(dameUno(claves(cn)), cn)))
  else
    nivelCom(p, borrar(dameUno(claves(cn)), cn))
  fi
fi
distManhatt(p, q) ≡ if π0(p) < π0(q) then q - p else p - q fi
+
if π1(p) < π1(q) then q - p else p - q fi
popularidad(iniciar(m)) ≡ 0
popularidad(avanzarTurno(s, cs)) ≡ popularidad(s)
popularidad(unir(s, s')) ≡ popularidad(s) + 1 + popularidad s'
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
construcciones(s) ≡ casas(s) ∪dicc comercios(s)
d ∪dicc d' ≡ if vacío?(claves(d')) then
  d
else
  definir(dameUno(claves(d')),
          obtener(dameUno(claves(d')), d'),
          d ∪dicc borrar(dameUno(claves(d')), d'))
  fi
sacarRepes(cs, cs') ≡ if vacío?(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

*donde:

avanzarTurnoValido : SimCity $s \times \text{dicc}(\text{Pos} \times \text{Construccion}) \text{ cs} \longrightarrow \text{boolean}$

$$\begin{aligned} \text{avanzarTurnoValido}(s, \text{cs}) \equiv & \neg \text{vacio?}(\text{claves}(\text{cs})) \wedge \\ & (\forall p : \text{Pos})(\text{def?}(p, \text{cs}) \Rightarrow_{\text{L}} \\ & \quad (\neg p \in \text{claves}(\text{construcciones}(s)) \wedge \\ & \quad \neg \pi_0(p) \in \text{horizontales}(\text{mapa}(s)) \wedge \neg \pi_1(p) \in \text{verticales}(\text{mapa}(s)) \wedge \\ & \quad (\text{obtener}(p, \text{cs}) =_{\text{obs}} \text{"casa"} \vee \text{obtener}(p, \text{cs}) =_{\text{obs}} \text{"comercio"})) \\ &) \end{aligned}$$

unirValido : Simcity $a \times \text{SimCity } b \longrightarrow \text{boolean}$

$$\begin{aligned} \text{unirValido}(a, b) \equiv & (\forall p : \text{Pos})(\text{def?}(p, \text{construcciones}(a)) \Rightarrow_{\text{L}} \\ & \quad (\neg \pi_0(p) \in \text{horizontales}(\text{mapa}(b)) \wedge \neg \pi_1(p) \in \text{verticales}(\text{mapa}(b)) \wedge \\ & \quad (\neg (\exists \text{otra} : \text{Pos})(\text{def?}(\text{otra}, \text{construcciones}(a)) \wedge_{\text{L}} \\ & \quad \quad \text{obtener}(\text{otra}, \text{construcciones}(a)) > \text{obtener}(p, \text{construcciones}(a)) \Rightarrow_{\text{L}} \\ & \quad \quad \neg \text{def?}(p, \text{construcciones}(b)))))) \\ &) \wedge \\ & (\forall p : \text{Pos})(\text{def?}(p, \text{construcciones}(b)) \Rightarrow_{\text{L}} \\ & \quad (\neg \pi_0(p) \in \text{horizontales}(\text{mapa}(a)) \wedge \neg \pi_1(p) \in \text{verticales}(\text{mapa}(a)) \wedge \\ & \quad (\neg (\exists \text{otra} : \text{Pos})(\text{def?}(\text{otra}, \text{construcciones}(b)) \wedge_{\text{L}} \\ & \quad \quad \text{obtener}(\text{otra}, \text{construcciones}(b)) > \text{obtener}(p, \text{construcciones}(b)) \Rightarrow_{\text{L}} \\ & \quad \quad \neg \text{def?}(p, \text{construcciones}(a)))))) \end{aligned}$$

1.3. Servidor

TAD SERVIDOR

géneros server

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

usa SimCity, Mapa, Nombre, Pos, Construcccion, Nivel, Nat, bool, $\text{dicc}(\alpha, \beta)$, $\text{conj}(\alpha)$

igualdad observacional

$$(\forall s, s' : \text{server}) \left(s =_{\text{obs}} s' \iff \left(\begin{array}{l} \text{partidas}(s) =_{\text{obs}} \text{partidas}(s') \wedge_L \\ \text{congeladas}(s) =_{\text{obs}} \text{congeladas}(s') \wedge \\ (\forall p : \text{Nombre})(\text{def?}(p, \text{partidas}(s)) \Rightarrow_L \\ \text{pendientes}(s, p) =_{\text{obs}} \text{pendientes}(s', p)) \end{array} \right) \right)$$

observadores básicos

partidas : server $\longrightarrow \text{dicc}(\text{Nombre}, \text{SimCity})$

congeladas : server $\longrightarrow \text{conj}(\text{Nombre})$

pendientes : server $s \times \text{Nombre } p \longrightarrow \text{dicc}(\text{Pos}, \text{Construcccion}) \quad \{ \text{def?}(p, \text{partidas}(s)) \}$

generadores

nuevoServer : $\longrightarrow \text{server}$

nuevaPartida : server $s \times \text{Nombre } p \times \text{Mapa} \longrightarrow \text{server} \quad \{ \neg \text{def?}(p, \text{partidas}(s)) \}$

unirPartidas : server $s \times \text{Nombre } p1 \times \text{Nombre } p2 \longrightarrow \text{server} \quad \{ * \text{unionValida}(s, p1, p2, cs) \}$

avanzarTurnoPartida : server $s \times \text{Nombre } p \longrightarrow \text{server} \quad \{ \text{def?}(p, s) \wedge_L * \text{avanzarTurnoValido}(s, p, \text{pendientes}(s, p)) \}$

agregarCasa : server $s \times \text{Nombre } p \times \text{Pos } pos \longrightarrow \text{server} \quad \{ * \text{agregarValido}(s, p, pos) \}$

agregarComercio : server $s \times \text{Nombre } p \times \text{Pos } pos \longrightarrow \text{server} \quad \{ * \text{agregarValido}(s, p, pos) \}$

otras operaciones

verMapa : server $s \times \text{Nombre } p \longrightarrow \text{Mapa} \quad \{ \text{def?}(p, \text{partidas}(s)) \}$

verCasas : server $s \times \text{Nombre } p \longrightarrow \text{dicc}(\text{Pos}, \text{Nivel}) \quad \{ \text{def?}(p, \text{partidas}(s)) \}$

verComercios : server $s \times \text{Nombre } p \longrightarrow \text{dicc}(\text{Pos}, \text{Nivel}) \quad \{ \text{def?}(p, \text{partidas}(s)) \}$

verPopularidad : server $s \times \text{Nombre } p \longrightarrow \text{Nat} \quad \{ \text{def?}(p, \text{partidas}(s)) \}$

verTurno : server $s \times \text{Nombre } p \longrightarrow \text{Nat} \quad \{ \text{def?}(p, \text{partidas}(s)) \}$

axiomas $\forall s: \text{server}, \forall p, p1, p2: \text{Nombre}, \forall m: \text{Mapa}, \forall cs: \text{conj}(\text{Pos})$

partidas(nuevoServer) $\equiv \text{vacío}$

partidas(nuevaPartida(s, p, m)) $\equiv \text{definir}(p, \text{iniciar}(m), \text{partidas}(s))$

partidas(unirPartidas(s, p1, p2)) $\equiv \text{definir}(p1, \text{unir}(\text{obtener}(p1, \text{partidas}(s)), \text{obtener}(p2, \text{partidas}(s))), \text{partidas}(s))$

partidas(avanzarTurnoPartida(s, p)) $\equiv \text{definir}(p, \text{avanzarTurno}(\text{obtener}(p, \text{partidas}(s)), \text{pendientes}(s, p)), \text{partidas}(s))$

partidas(agregarCasa(s, p, pos)) $\equiv \text{partidas}(s)$

partidas(agregarComercio(s, p, pos)) $\equiv \text{partidas}(s)$

congeladas(nuevoServer) $\equiv \emptyset$

congeladas(nuevaPartida(s, p, m)) $\equiv \text{congeladas}(s)$

congeladas(avanzarTurnoPartida(s, p)) $\equiv \text{congeladas}(s)$

congeladas(unirPartidas(s, p1, p2)) $\equiv \text{Ag}(p2, \text{congeladas}(s))$

congeladas(agregarCasa(s, p, pos)) $\equiv \text{congeladas}(s)$

congeladas(agregarComercio(s, p, pos)) $\equiv \text{congeladas}(s)$

pendientes(nuevaPartida(s, p1, m), p) $\equiv \text{if } p =_{\text{obs}} p1 \text{ then vacío else pendientes}(s, p) \text{ fi}$

pendientes(unirPartidas(s, p1, p2), p) $\equiv \text{pendientes}(s, p)$

```

pendientes(agregarCasa(s, p1, pos), p)    ≡ if  $p =_{\text{obs}} p1$  then
                                         definir(pos, "casa", pendientes(s, p))
                                         else
                                         pendientes(s, p)
                                         fi
pendientes(agregarComercio(s, p1, pos), p) ≡ if  $p =_{\text{obs}} p1$  then
                                         definir(pos, "comercio", pendientes(s, p))
                                         else
                                         pendientes(s, p)
                                         fi
pendientes(avanzarTurnoPartida(s, p1), p) ≡ if  $p =_{\text{obs}} p1$  then vacio else pendientes(s, p) fi
// oo
verMapa(s, p)                             ≡ mapa(obtener(p, partidas(s)))
verCasas(s, p)                             ≡ casas(obtener(p, partidas(s)))
verComercios(s, p)                         ≡ comercios(obtener(p, partidas(s)))
verPopularidad(s, p)                       ≡ popularidad(obtener(p, partidas(s)))
verTurno(s, p)                             ≡ turnos(obtener(p, partidas(s)))

```

Fin TAD

*donde:

unionValida : server s × Nombre p1 × Nombre p2 → boolean

$$\begin{aligned} \text{unionValida}(s, p1, p2) \equiv & \text{def?}(p1, \text{partidas}(s)) \wedge \text{def?}(p2, \text{partidas}(s)) \wedge \\ & p1 \notin \text{congeladas}(s) \wedge_L \\ & \text{vacio?}(\text{claves}(\text{pendientes}(s, p1))) \wedge \text{vacio?}(\text{claves}(\text{pendientes}(s, p2))) \wedge \\ & (\forall \text{pos} : \text{Pos})(\text{pos} \in \text{claves}(\text{constr1}) \Rightarrow_L \\ & \quad \neg \text{sobreRio}(\text{pos}, \text{sim2}) \wedge \\ & \quad ((\nexists \text{otra} : \text{Pos})(\text{otra} \in \text{constr1} \wedge_L \\ & \quad \quad \text{obtener}(\text{pos}, \text{constr1}).\text{nivel} < \text{obtener}(\text{otra}, \text{constr1}).\text{nivel} \\ & \quad) \Rightarrow_L \neg \text{def?}(\text{pos}, \text{constr2})) \\ &) \wedge \\ & (\forall \text{pos} : \text{Pos})(\text{pos} \in \text{claves}(\text{constr2}) \Rightarrow_L \\ & \quad \neg \text{sobreRio}(\text{pos}, \text{sim1}) \wedge \\ & \quad ((\nexists \text{otra} : \text{Pos})(\text{otra} \in \text{constr2} \wedge_L \\ & \quad \quad \text{obtener}(\text{pos}, \text{constr2}).\text{nivel} < \text{obtener}(\text{otra}, \text{constr2}).\text{nivel} \\ & \quad) \Rightarrow_L \neg \text{def?}(\text{pos}, \text{constr1})) \\ &) \end{aligned}$$

donde $\text{sim1} \equiv \text{obtener}(p1, \text{partidas}(s))$,
 $\text{sim2} \equiv \text{obtener}(p2, \text{partidas}(s))$,
 $\text{constr1} \equiv \text{casas}(\text{sim1}) \cup \text{comercios}(\text{sim1})$,
 $\text{constr2} \equiv \text{casas}(\text{sim2}) \cup \text{comercios}(\text{sim2})$

avanzarTurnoValido : server s × Nombre p × dicc(Pos × Construcccion) cs → boolean

$$\begin{aligned} \text{avanzarTurnoValido}(s, p, cs) \equiv & \text{def?}(p, \text{partidas}(s)) \wedge \\ & p \notin \text{congeladas}(s) \wedge \\ & \neg \text{vacio?}(\text{claves}(cs)) \wedge_L \end{aligned}$$

donde $\text{sim} \equiv \text{obtener}(p, \text{partidas}(s))$

agregarValido : server s × Nombre p × Pos pos → boolean

$$\begin{aligned} \text{agregarValido}(s, p, pos) \equiv & \text{def?}(p, \text{partidas}(s)) \wedge_L \neg p \in \text{congeladas}(s) \wedge \\ & \neg \text{def?}(pos, \text{verCasas}(s, p)) \wedge \neg \text{def?}(pos, \text{verComercios}(s, p)) \wedge \\ & \neg \text{esRio}(pos, \text{verMapa}(s, p)) \end{aligned}$$

• ∪ • : dicc($\alpha \times \beta$) × dicc($\alpha \times \beta$) → dicc(α, β)

$a \cup b \equiv _ \text{definir}(a, b, \text{claves}(b))$

$_ \text{union} : \text{dicc}(\alpha \times \beta) \times \text{dicc}(\alpha \times \beta) \times b \times \text{conj}(\alpha) \text{ cs} \rightarrow \text{dicc}(\alpha, \beta) \quad \{\text{cs} \subseteq \text{claves}(b)\}$

$_ \text{union}(a, b, cs) \equiv \text{if } \text{vacio?}(cs) \text{ then}$

a

else

$_ \text{union}(\text{definir}(\text{dameUno}(cs), \text{obtener}(\text{dameUno}(cs), b)), b, \text{sinUno}(cs))$

fi

2. Módulos de referencia

2.1. Módulo Mapa

Interfaz

se explica con: MAPA

géneros: mapa

TP de Especificación y Diseño Operaciones básicas de mapa

CREAR(**in** $hs : \text{conj}(\text{Nat})$, **in** $vs : \text{conj}(\text{Nat})$) $\rightarrow res : \text{mapa}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{mapa}(hs, vs)\}$

Complejidad: $O(\text{copy}(hs), \text{copy}(vs))$

Descripción: crea un mapa

ESRIO(**in** $m1 : \text{Mapa}$, **in** $p : \text{Pos}$) $\rightarrow res : \text{Bool}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{esRio}(m1, p)\}$

Complejidad: $O(1)$

Descripción: verifica si en determinada pos hay rio

SUMA(**in** $m1 : \text{Mapa}$, **in** $m2 : \text{Mapa}$) $\rightarrow res : \text{Mapa}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} m1 + m2\}$

Complejidad: $O(\text{crear}(m1) + \text{crear}(m2))$

Descripción: une 2 mapas

Representación

TP de Especificación y Diseño 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 $\text{tupla}(\text{horizontales: conj}(\text{Nat}), \text{verticales: conj}(\text{Nat}))$

$\text{Rep} : \text{estr} \rightarrow \text{boolean}$

$\text{Rep}(e) \equiv \text{true} \iff \text{true}$

$\text{Abs} : \text{estr } m \rightarrow \text{mapa}$

$\{\text{Rep}(m)\}$

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

Algoritmos

crear(*in* $hs : \text{conj}(\text{Nat})$, *in* $vs : \text{conj}(\text{Nat})$) $\rightarrow res : \text{estr}$

1: $estr.horizontalas \leftarrow hs$
 2: $estr.verticales \leftarrow vs$ **return** $estr$

Complejidad: $O(\text{copy}(hs) + \text{copy}(vs))$

esRio(*in* $m1 : \text{Mapa}$, *in* $p : \text{Pos}$) $\rightarrow res : \text{Bool}$

1: $bool\ res \leftarrow false$
 2: *for*($Nat\ y : estr.horizontalas$)
 3: *if*($y =_{obs}\ p.y$) *then*
 4: $res \leftarrow true$
 5: *else*
 6: *skip*
 7: *for*($Nat\ x : estr.verticales$)
 8: *if*($x =_{obs}\ p.x$) *then*
 9: $res \leftarrow true$
 10: *else*
 11: *skip* **return** res

Complejidad: $O(\#horizontalas(m1) + \#verticales(m1))$

Suma(*in* $hs : \text{conj}(\text{Nat})$, *in* $vs : \text{conj}(\text{Nat})$) $\rightarrow res : \text{estr}$

1: *for*($Nat\ n : m1.horizontalas$)
 2: $Ag(n, estr.horizontalas)$
 3: *for*($Nat\ n : m2.horizontalas$)
 4: $Ag(n, estr.horizontalas)$
 5: *for*($Nat\ n : m1.verticales$)
 6: $Ag(n, estr.verticales)$
 7: *for*($Nat\ n : m2.verticales$)
 8: $Ag(n, estr.verticales)$ **return** $estr$

Complejidad: $O(\#horizontalas(m1) + \#verticales(m1)\#horizontalas(m2) + \#verticales(m2))$

2.2. Módulo SimCity

Interfaz

usa: completar

exporta: completar

se explica con: SIMCITY

géneros: SimCity

Operaciones básicas de SimCity

VERMAPA(**in** $s : \text{SimCity}$) $\rightarrow res : \text{Mapa}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{mapa}(\hat{s})\}$

Complejidad: $\Theta(\sum_{s_i \in U} \text{copy}(\text{mapa}(s_i)))$ donde U es el conjunto de SimCities en union con S, y S.

Descripción: Crea una copia del mapa sobre el que se desarrolla el juego actual. *podria ser por ref?*

Representación

SimCity se compone por la *ubicacion* y *nivel* de una serie de *construcciones*, de tipo *casa* o *comercio*, sobre un *Mapa*, y de una *popularidad* respecto a la cantidad de uniones que lo modificaron.

La ubicación de las casas se representan sobre un diccionario lineal con clave $Pos \equiv \text{tupla} < Nat, Nat >$ y significado $Nivel \equiv Nat$. La ubicación de los comercios se representan similarmente, pero su significado responde a un $NivelBase \equiv Nat$ a partir del cual se calcula propiamente su *nivel*. El mapa es de tipo *Mapa* y las uniones se representan a través de una *lista* que contiene punteros a los *SimCitys* unidos e información relevante para calcular el nivel de sus construcciones. Ya que, una vez unido a otro, un *SimCity* debe permanecer sin modificación.

SimCity se representa con *estr*

donde *estr* es *tupla*(*turno* : Nat,
 popularidad : Nat,
 mapa : Mapa,
 casas : diccLineal(*pos*, Nivel) ,
 comercios : diccLineal(*pos*, NivelBase) ,
 uniones : lista(*hijo*))

donde *hijo* es *tupla*(*sc* : puntero(*estr*) ,
 turnosDesdeUnion : Nat)

donde *pos* es *tupla*(*x* : Nat , *y* : Nat)

$\text{Rep} : \text{estr}^1 \rightarrow \text{boolean}$

$\text{Rep}(e) \equiv \text{true} \iff ($
 $(\&e \notin \text{Unidos})^2 \wedge_L$
 $(e.\text{popularidad} = \#(\text{Unidos}))^3 \wedge$
 $(\forall p : \text{puntero}(\text{estr})) (p \in \text{Unidos} \Rightarrow_L$
 $e.\text{turno} \geq (*p).\text{turno}$
 $)^4 \wedge$
 $(\forall p : \text{pos}) (p \in \text{claves}(\text{Casas}) \Rightarrow_L$
 $\neg \text{def}(p, e.\text{comercios})^5 \wedge \neg \text{esRio}(p, \text{Mapas})^6 \wedge (\text{obtener}(p, \text{Casas}) < e.\text{turno})^7$
 $) \wedge$
 $(\forall p : \text{pos}) (p \in \text{claves}(\text{Comercios}) \Rightarrow_L$
 $\neg \text{def}(p, e.\text{casas})^8 \wedge \neg \text{esRio}(p, \text{Mapas})^9 \wedge (\text{obtener}(p, \text{Comercios}) < e.\text{turno})^{10}$
 $) \wedge$
 $(\forall h : \text{hijo}) (\text{esta?}(h, e.\text{uniones}) \Rightarrow_L$
 $(h.\text{sc} \neq \text{null} \wedge_L h.\text{sc} \notin \text{unirPunteros}(\text{remove}(p, e.\text{uniones})))^{11} \wedge_L$
 $\text{rep}(*h.\text{sc})^{12} \wedge_L$
 $(e.\text{turno} \geq h.\text{turnosDesdeUnion})^{13} \wedge$
 $(\forall h_2 : \text{hijo}) (\text{esta?}(h_2, e.\text{uniones}) \wedge_L \text{pos}(h_2, e.\text{uniones}) > \text{pos}(h, e.\text{uniones}) \Rightarrow_L$
 $h_2.\text{turnosDesdeUnion} \leq h.\text{turnosDesdeUnion}$
 $)^{14}$
 $) \wedge$
 $\text{unionesValidas}(e, e.\text{uniones})^{15}$
 $)$

donde

$\text{Unidos} \equiv \text{unirPunteros}(e.\text{uniones})$
 $\text{Casas} \equiv \text{unirCasas}(\text{Ag}(\&e, \text{Unidos}))$
 $\text{Comercios} \equiv \text{unirComercios}(\text{Ag}(\&e, \text{Unidos}))$
 $\text{Mapas} \equiv \text{unirMapas}(\text{Ag}(\&e, \text{Unidos}))$

$\text{Abs} : \text{estr } e \rightarrow \text{SimCity}$

$\{\text{Rep}(e)\}$

$$\begin{aligned}
\text{Abs}(e) \equiv & \text{sc} : \text{SimCity} \mid \\
& \text{mapa}(\text{sc}) =_{\text{obs}} \text{Mapas} \wedge \\
& \text{casas}(\text{sc}) =_{\text{obs}} \text{nivelar}(\text{Casas}) \wedge \\
& \text{comercios}(\text{sc}) =_{\text{obs}} \text{nivelar}(\text{Comercios}) \wedge \\
& \text{popularidad}(\text{sc}) =_{\text{obs}} e.\text{popularidad}
\end{aligned}$$

donde

$$\begin{aligned}
\text{Unidos} & \equiv \text{unirPunteros}(e.\text{uniones}) \\
\text{Casas} & \equiv \text{unirCasas}(\text{Ag}(\&e, \text{Unidos})) \\
\text{Comercios} & \equiv \text{unirComercios}(\text{Ag}(\&e, \text{Unidos})) \\
\text{Mapas} & \equiv \text{unirMapas}(\text{Ag}(\&e, \text{Unidos}))
\end{aligned}$$

1. Se asume el traspaso de toda estructura de representación a su equivalente abstracto (se aplica el sombrerito).
2. la estructura no loopea consigo misma.
3. el turno actual es mayor o igual al turno de cualquier simCity hijo.
4. la popularidad es igual a la cantidad de uniones.
5. ninguna casa en la unión está en la posición de uno de los comercios de este simCity particular.
6. ninguna casa en la unión está sobre un río perteneciente a cualquier mapa en la unión.
7. el turno es mas grande que el nivel de cualquier casa en la unión.
8. ningún comercio en la unión está en la posición de una de las casas de este simCity particular.
9. ningún comercio en la unión está sobre un río perteneciente a cualquier mapa en la unión.
10. el turno es mas grande que el nivel base de cualquier comercio en la unión.
11. Cada hijo apunta a un SimCity y su puntero no aparece en ningún otro SimCity de la unión.
12. Cada hijo apunta a un Simcity válido.
13. El turno es mayor o igual a la cantidad de turnos que pasaron desde la unión.
14. Las uniones están ordenadas de más antiguas a más recientes.
15. No se solapan posiciones máximas entre esta estructura hasta el hijo 'x', descontando construcciones agregadas después de la unión, y ese hijo, para todo hijo.

auxiliares para la representación

$\text{unirPunteros} : \text{secu}(\text{hijo}) \rightarrow \text{conj}(\text{puntero}(\text{estr}))$

$\text{unirPunteros}(s) \equiv _ \text{unirPunteros}(s, \emptyset)$

$_ \text{unirPunteros} : \text{secu}(\text{hijo}) \times \text{conj}(\text{puntero}(\text{estr})) \rightarrow \text{conj}(\text{puntero}(\text{estr}))$

$_ \text{unirPunteros}(s, p) \equiv \text{if } \text{vacía?}(s) \text{ then}$
 $\quad p$
 else if $\text{prim}(s).sc \in p$ **then** // por si hay loops
 $_ \text{unirPunteros}(\text{fin}(s), p)$
 else
 $_ \text{unirPunteros}((*(\text{prim}(s).sc)).\text{uniones}, \text{Ag}(\text{prim}(s).sc, p)) \cup$
 $_ \text{unirPunteros}(\text{fin}(s), \text{Ag}(\text{prim}(s).sc, p))$
 fi

$\text{unirMapas} : \text{conj}(\text{puntero}(\text{estr})) \rightarrow \text{Mapa}$

$\text{unirMapas}(ps) \equiv \text{if } \text{vacío?}(ps) \text{ then}$
 $\text{crear}(\emptyset, \emptyset)$
 else
 $(*(\text{dameUno}(ps))).\text{mapa} + \text{UnirMapas}(\text{sinUno}(ps))$
 fi

$\text{unirCasas} : \text{conj}(\text{puntero}(\text{estr})) \rightarrow \text{dicc}(\text{Pos}, \text{Nivel})$

$\text{unirCasas}(ps) \equiv \text{if } \text{vacío?}(ps) \text{ then}$
 vacío
 else
 $(*p).\text{casas} \cup_{\text{dicc}} \text{unirCasas}(\text{sinUno}(ps))$
 fi

$\text{unirComercios} : \text{conj}(\text{puntero}(\text{estr})) \rightarrow \text{dicc}(\text{Pos}, \text{Nat})$

$\text{unirComercios}(ps) \equiv \text{if } \text{vacío?}(ps) \text{ then}$
 vacío
 else
 $(*p).\text{comercios} \cup_{\text{dicc}} \text{unirComercios}(\text{sinUno}(ps))$
 fi

$\text{remove} : \text{secu}(\alpha) \times \alpha \rightarrow \text{secu}(\alpha)$ // remueve la primer aparición

$\text{remove}(s, a) \equiv \text{if } \text{vacía?}(s) \text{ then}$
 $\langle \rangle$
 else if $a = \text{prim}(s)$ **then**
 $\text{fin}(s)$
 else
 $\text{prim}(s) \bullet \text{remove}(\text{fin}(s), a)$
 fi

$\text{unionesValidas} : \text{estr} \times \text{secu}(\text{hijo}) \rightarrow \text{bool}$

$\text{unionesValidas}(e, s) \equiv \text{vacío?}(s) \vee_{\text{L}} (\text{maxcons}(e, \text{izq}) \cap \text{maxcons}(e, \text{der}) = \emptyset \wedge \text{unionesValidas}(e, \text{com}(s)))$

donde

$\text{com} \equiv \text{unirPunteros}(\text{com}(s))$
 $\text{ult} \equiv \text{ult}(s) \bullet \langle \rangle$
 $\text{casascom} \equiv \text{unirCasas}(\text{com}) \cup_{\text{dicc}} \text{filtrar}(e.\text{casas}, \text{ult}(s).\text{turnosDesdeUnion})^1$
 $\text{comercom} \equiv \text{unirComercios}(\text{com}) \cup_{\text{dicc}} \text{filtrar}(e.\text{comercios}, \text{ult}(s).\text{turnosDesdeUnion})^1$
 $\text{casasult} \equiv \text{unirCasas}(\text{ult})$
 $\text{comerult} \equiv \text{unirComercios}(\text{ult})$
 $\text{izq} \equiv \text{claves}(\text{casascom}) \cup \text{claves}(\text{comercom})$
 $\text{der} \equiv \text{claves}(\text{casasult}) \cup \text{claves}(\text{comerult})$

1. las casas o comercios de éste simCity particular con nivel o nivel base \leq turnosDesdeUnion son aquellas que se agregaron después de la unión.

$\text{filtrar} : \text{dicc}(\text{Pos} \times \text{Nat}) \times \text{Nat} \longrightarrow \text{dicc}(\text{Pos}, \text{Nat})$

$\text{filtrar}(d, n) \equiv \text{if } \text{vacio?}(d) \text{ then}$
 vacio
 else if $\text{sig} \leq n$ **then**
 $\text{filtrar}(\text{borrar}(\text{clave}, d), n)$
 else
 $\text{definir}(\text{clave}, \text{sig}, \text{filtrar}(\text{borrar}(\text{clave}, d), n))$
 fi

donde

$\text{clave} \equiv \text{dameUno}(\text{claves}(d))$
 $\text{sig} \equiv \text{obtener}(\text{clave}, d)$

$\text{maxcons} : \text{estr} \times \text{conj}(\text{Pos}) \longrightarrow \text{conj}(\text{Pos})$

$\text{maxcons}(e, c) \equiv _ \text{maxcons}(e, c, \emptyset, 0)$

$_ \text{maxcons} : \text{estr} \times \text{conj}(\text{Pos}) \times \text{conj}(\text{Pos}) \times \text{Nat} \longrightarrow \text{conj}(\text{Pos})$

$_ \text{maxcons}(e, c, \text{max}, n) \equiv \text{if } \text{vacio?}(c) \text{ then}$
 max
 else if $\text{nivel}_i > n$ **then**
 $_ \text{maxcons}(e, \text{sinUno}(c), \text{Ag}(\text{pos}_i, \emptyset), \text{nivel}_i)$
 else if $\text{nivel}_i = n$ **then**
 $_ \text{maxcons}(e, \text{sinUno}(c), \text{Ag}(\text{pos}_i, \text{max}), n)$
 else
 $_ \text{maxcons}(e, \text{sinUno}(c), \text{max}, n)$
 fi

donde

$\text{pos}_i \equiv \text{dameUno}(c)$
 $\text{nivel}_i \equiv \text{nivel}(e, \text{pos}_i)$

$\text{nivel} : \text{estr} \times \text{pos} \longrightarrow \text{Nat}$

$\text{nivel}(e, \text{pos}) \equiv \text{if } \text{def?}(\text{pos}, \text{Casas}) \text{ then}$
 $\text{obtener}(\text{pos}, \text{Casas}) + \text{nivelesPorUnion}(e, \text{pos})$
 else
 $\text{maximo}(\text{obtener}(\text{pos}, \text{Comercios}) \bullet n\text{Manhattan}) + \text{nivelesPorUnion}(e, \text{pos})$
 fi

donde:

$\text{Unidos} \equiv \text{unirPunteros}(e.\text{uniones})$
 $\text{Casas} \equiv \text{unirCasas}(\text{Ag}(\&e, \text{Unidos}))$
 $\text{Comercios} \equiv \text{unirComercios}(\text{Ag}(\&e, \text{Unidos}))$
 $n\text{Manhattan} \equiv \text{significados}(\text{manhattan}(\text{pos}, 3), \text{Casas})$

$\text{nivelesPorUnion} : \text{estr} \times \text{pos} \longrightarrow \text{Nat}$

$\text{nivelesPorUnion}(e, \text{pos}) \equiv \text{if } \text{def?}(\text{pos}, e.\text{casas}) \vee \text{def?}(\text{pos}, e.\text{comercios}) \text{ then}$
 0
 else
 $\text{hijoCorrecto.turnosDesdeUnion} +$
 $\text{nivelesPorUnion}(\text{hijoCorrecto.sc}, \text{pos})$
 fi

donde:

$\text{hijoCorrecto} \equiv \text{llegar}(e.\text{uniones}, \text{pos})$

$\text{llegar} : \text{secu}(\text{hijo}) \times \text{pos} \longrightarrow \text{hijo}$

$\text{llegar}(s, p) \equiv \text{if } \text{def?}(\text{pos}, \text{unirCasas}(\text{hijo}) \vee \text{def?}(\text{pos}, \text{unirComercios}(\text{hijo})) \text{ then}$
 $\text{prim}(s)$
 else
 $\text{llegar}(\text{fin}(s))$
 fi

donde

hijo $\equiv Ag(prim(s).sc, \emptyset)$

manhattan : Pos \times Nat \longrightarrow Conj(Pos)

```
manhattan(p, dist)  $\equiv$  if dist = 0 then
    Ag(p,  $\emptyset$ )
else
    diagonal({p.x, p.y + dist}, {p.x + dist, p.y})  $\cup$ 
    if p.x - dist  $\geq$  0 then
        diagonal({p.x, p.y + dist}, {p.xdist, p.y})
    else
         $\emptyset$ 
    fi  $\cup$ 
    if p.y - dist  $\geq$  0 then
        diagonal({p.x, p.ydist}, {p.x + dist, p.y})
    else
         $\emptyset$ 
    fi  $\cup$ 
    if p.x - dist  $\geq$  0  $\wedge$  p.y - dist  $\geq$  0 then
        diagonal({p.x, p.ydist}, {p.xdist, p.y})
    else
         $\emptyset$ 
    fi  $\cup$ 
    manhattan(p, dist - 1)
fi
```

diagonal : pos \times pos \times Nat n \longrightarrow Conj(pos)

```
diagonal(d, h, y)  $\equiv$  if y = |h.y - d.y| then
    Ag(h,  $\emptyset$ )
else
    Ag(caminar(d, h, y), diagonal(d, h, y + 1))
fi
```

caminar : pos \times pos \times Nat \longrightarrow pos

```
caminar(d, h, y)  $\equiv$  if d.y  $\leq$  h.y then
    if d.x  $\leq$  h.x then
        {d.x + y, d.y + y}
    else
        {d.x - y, d.y + y}
    fi
else
    if d.x  $\leq$  h.x then
        {d.x + y, d.y - y}
    else
        {d.x - y, d.y - y}
    fi
fi
```

significados : conj(α) \times dicc($\alpha \times \beta$) \longrightarrow secu(α)

```
significados(c, d)  $\equiv$  if vacio?(c) then
    <>
else if def?(dameUno(c), d) then
    obtener(c, d) • significados(sinUno(c), d)
else
    significados(sinUno(c), d)
fi
```

maximo : secu(Nat) a \longrightarrow Nat

{*long(a)* > 0}

```
maximo(s)  $\equiv$  if long(s) = 1 then prim(s) else max(prim(s), maximo(fin(s))) fi
```

nivelar : estr \times dicc(Pos \times Nat) \longrightarrow dicc(Pos, Nat)

```
nivelar(d)  $\equiv$  if vacio?(d) then vacio else definir(clave, nivel(e, clave), nivelar(e, borrar(clave, d))) fi  
donde  
    clave  $\equiv$  dameUno(claves(d))
```

2.3. Módulo Servidor

Interfaz

se explica con: SERVIDOR

géneros: server

TP de Especificación y Diseño Operaciones básicas de server

NUEVOSEVER() $\rightarrow res : server$

Pre $\equiv \{true\}$

Post $\equiv \{res =_{obs} nuevoServer\}$

Complejidad: $O()$

Descripción: Crea un servidor

Aliasing: No tiene

PARTIDAS(in s: server) $\rightarrow res : dicc(string, SimCity)$

Pre $\equiv \{true\}$

Post $\equiv \{res =_{obs} partidas(s)\}$

Complejidad: $O()$

Descripción: Devuelve un diccionario con todas las partidas del servidor

Aliasing: Devuelve una copia (Esto habria que verlo, ya que no tenemos este dicc(nombre, simcity) por asi decirlo. Maybe hacemos uno de cero? Y tambien habria que ver si lo devolvemos con los SimCity en las hojas o son punteros?)

CONGELADAS(in s: server) $\rightarrow res : conj(string)$

Pre $\equiv \{true\}$

Post $\equiv \{res =_{obs} congeladas(s)\}$

Complejidad: $O()$

Descripción: Devuelve el conjunto con los nombres de las partidas no modificables

Aliasing: Devuelve una copia

NUEVAPARTIDA(in/out s: server, in p: string, in m: mapa)

Pre $\equiv \{s =_{obs} s0 \wedge \neg def?(p, partidas(s))\}$

Post $\equiv \{s =_{obs} nuevaPartida(s0, p, m)\}$

Complejidad: $O()$

Descripción: Agrega una partida nueva al servidor

Aliasing: No tiene

UNIRPARTIDAS(in/out s: server, in p1: string, in p2: string)

Pre $\equiv \{*unionValida(s, p1, p2)\}$

Post $\equiv \{s =_{obs} unirPartidas(s0, p1, p2)\}$

Complejidad: $O()$

Descripción: Une dos partidas de simcity en una, p2 pasa a ser no modificable

Aliasing: No tiene

AVANZARTURNOPARTIDA(in/out s: server, in p: string)

Pre $\equiv \{def?(p, s) \wedge_L *avanzarTurnoValido(s, p, pendientes(p, s))\}$

Post $\equiv \{s =_{obs} avanzarTurnoPartida(s0, p)\}$

Complejidad: $O()$

Descripción: Avanza el turno de una partida y agrega las construcciones definidas en el diccionario de pendientes

Aliasing: No tiene

AGREGARCASA(in/out s: server, in p: string, in pos: Pos)

Pre $\equiv \{s =_{obs} s0 \wedge *agregarValido(s, p, pos)\}$

Post $\equiv \{s =_{obs} agregarCasa(s0, p, pos)\}$

Complejidad: $O()$

Descripción: Agrega una nueva casa al diccionario de pendientes de la partida

Aliasing: No tiene

AGREGARCOMERCIO(**in/out** s : server, **in** $p1$: string, **in** $p2$: string)

Pre $\equiv \{s =_{\text{obs}} s0 \wedge *agregarValido(s, p, pos)\}$

Post $\equiv \{s =_{\text{obs}} agregarComercio(s0, p, pos)\}$

Complejidad: $O()$

Descripción: Agrega un nuevo comercio al diccionario de pendientes de la partida

Aliasing: No tiene

POPULARIDAD(**in** s : server, **in** p : string) $\rightarrow res$: Nat

Pre $\equiv \{def?(p, partidas(s))\}$

Post $\equiv \{res =_{\text{obs}} verPopularidad(s, p)\}$

Complejidad: $O()$

Descripción: Devuelve la popularidad de la partida

Aliasing: Devuelve una referencia no modificable

ANTIGUEDAD(**in** s : server, **in** p : string) $\rightarrow res$: Nat

Pre $\equiv \{def?(p, partidas(s))\}$

Post $\equiv \{res =_{\text{obs}} verTurno(s, p)\}$

Complejidad: $O()$

Descripción: Devuelve la antigüedad de la partida

Aliasing: Devuelve una referencia no modificable

MAPA(**in** s : server, **in** p : string) $\rightarrow res$: mapa

Pre $\equiv \{def?(p, partidas(s))\}$

Post $\equiv \{res =_{\text{obs}} verMapa(s, p)\}$

Complejidad: $O()$

Descripción: Devuelve el mapa de la partida

Aliasing: Devuelve una copia? (habria que ver como funciona mapa en la implementacion del simcity)

VERCASAS(**in** s : server, **in** p : string) $\rightarrow res$: dicc(Pos, Nat)

Pre $\equiv \{def?(p, partidas(s))\}$

Post $\equiv \{res =_{\text{obs}} verCasas(s, p)\}$

Complejidad: $O()$

Descripción: Devuelve un diccionario con las posiciones y niveles de las casas de la partida

Aliasing: Devuelve una copia? (habria que ver como funciona casas en la implementacion del simcity)

VERCOMERCIOS(**in** s : server, **in** p : string) $\rightarrow res$: dicc(Pos, Nat)

Pre $\equiv \{def?(p, partidas(s))\}$

Post $\equiv \{res =_{\text{obs}} verComercios(s, p)\}$

Complejidad: $O()$

Descripción: Devuelve un diccionario con las posiciones y niveles de los comercios de la partida

Aliasing: Devuelve una copia? (habria que ver como funciona comercios en la implementacion del simcity)

*donde:

$\text{unionValida} : \text{server } s \times \text{Nombre } p1 \times \text{Nombre } p2 \longrightarrow \text{boolean}$

$$\begin{aligned} \text{unionValida}(s, p1, p2) \equiv & \text{def?}(p1, \text{partidas}(s)) \wedge \text{def?}(p2, \text{partidas}(s)) \wedge \\ & p1 \notin \text{congeladas}(s) \wedge_L \\ & \text{vacio?}(\text{claves}(\text{pendientes}(s, p1))) \wedge \text{vacio?}(\text{claves}(\text{pendientes}(s, p2))) \wedge \\ & (\forall \text{pos} : \text{Pos})(\text{pos} \in \text{claves}(\text{constr1}) \Rightarrow_L \\ & \quad \neg \text{sobreRio}(\text{pos}, \text{sim2}) \wedge \\ & \quad ((\nexists \text{otra} : \text{Pos})(\text{otra} \in \text{constr1} \wedge_L \\ & \quad \quad \text{obtener}(\text{pos}, \text{constr1}).\text{nivel} < \text{obtener}(\text{otra}, \text{constr1}).\text{nivel} \\ & \quad) \Rightarrow_L \neg \text{def?}(\text{pos}, \text{constr2})) \\ &) \wedge \\ & (\forall \text{pos} : \text{Pos})(\text{pos} \in \text{claves}(\text{constr2}) \Rightarrow_L \\ & \quad \neg \text{sobreRio}(\text{pos}, \text{sim1}) \wedge \\ & \quad ((\nexists \text{otra} : \text{Pos})(\text{otra} \in \text{constr2} \wedge_L \\ & \quad \quad \text{obtener}(\text{pos}, \text{constr2}).\text{nivel} < \text{obtener}(\text{otra}, \text{constr2}).\text{nivel} \\ & \quad) \Rightarrow_L \neg \text{def?}(\text{pos}, \text{constr1})) \\ &) \end{aligned}$$

donde $\text{sim1} \equiv \text{obtener}(p1, \text{partidas}(s))$,
 $\text{sim2} \equiv \text{obtener}(p2, \text{partidas}(s))$,
 $\text{constr1} \equiv \text{casas}(\text{sim1}) \cup_{\text{dicc}} \text{comercios}(\text{sim1})$,
 $\text{constr2} \equiv \text{casas}(\text{sim2}) \cup_{\text{dicc}} \text{comercios}(\text{sim2})$

$\text{avanzarTurnoValido} : \text{server } s \times \text{Nombre } p \times \text{dicc}(\text{Pos} \times \text{Construccion}) \text{ cs} \longrightarrow \text{boolean}$

$$\begin{aligned} \text{avanzarTurnoValido}(s, p, \text{cs}) \equiv & \text{def?}(p, \text{partidas}(s)) \wedge \\ & p \notin \text{congeladas}(s) \wedge \\ & \neg \text{vacio?}(\text{claves}(\text{cs})) \end{aligned}$$

$\text{agregarValido} : \text{server } s \times \text{Nombre } p \times \text{Pos } \text{pos} \longrightarrow \text{boolean}$

$$\begin{aligned} \text{agregarValido}(s, p, \text{pos}) \equiv & \text{def?}(p, \text{partidas}(s)) \wedge_L \neg p \in \text{congeladas}(s) \wedge \\ & \neg \text{def?}(\text{pos}, \text{verCasas}(s, p)) \wedge \neg \text{def?}(\text{pos}, \text{verComercios}(s, p)) \wedge \\ & \neg \text{esRio}(\text{pos}, \text{verMapa}(s, p)) \end{aligned}$$

$\bullet \cup_{\text{dicc}} \bullet : \text{dicc}(\alpha \times \beta) \times \text{dicc}(\alpha \times \beta) \longrightarrow \text{dicc}(\alpha, \beta)$

$d \cup_{\text{dicc}} d' \equiv \text{if } \text{vacio?}(\text{claves}(d')) \text{ then}$
 $\quad d$
 else
 $\quad \text{definir}(\text{dameUno}(\text{claves}(d')),$
 $\quad \quad \text{obtener}(\text{dameUno}(\text{claves}(d')), d'),$
 $\quad \quad d \cup_{\text{dicc}} \text{borrar}(\text{dameUno}(\text{claves}(d')), d'))$
 fi

Representación

TP de Especificación y Diseño Representación de servidor

Un servidor almacena y actualiza los diferentes SimCity. Se representa como un diccionario implementado en un trie, donde las claves son los nombres de las partidas y los significados un puntero al SimCity y su estado (si es modificable o no).

servidor **se representa con** `diccTrie(Nombre, partida)`

donde `partida` es `tupla(modificable : bool ,
sim : puntero(SimCity) ,
pendientes : dicc(Pos, Construcccion))`

donde `pos` es `tupla(x : Nat , y : Nat)`

donde `Nombre` es `string`

donde `Construcccion` es `string`

`Rep : estr \rightarrow boolean`

`Rep(e) \equiv true \iff`
 $(\forall \text{partida}_1, \text{partida}_2 : \text{Nombre})$
 $((\text{def?}(\text{partida}_1, e) \wedge \text{def?}(\text{partida}_2, e) \wedge \text{partida}_1 \neq \text{partida}_2) \Rightarrow_L$
 $\text{obtener}(\text{partida}_1, e).\text{sim} \neq \text{obtener}(\text{partida}_2, e).\text{sim}) \wedge$
 $(\forall \text{partida} : \text{Nombre})(\text{def?}(\text{partida}, e) \Rightarrow_L$
 $p.\text{sim} \neq \text{NULL} \wedge$
 $(p.\text{modificable} =_{\text{obs}} \text{false} \Rightarrow_L \text{vacio?}(\text{claves}(p.\text{pendientes}))) \wedge$
 $(\forall \text{pos} : \text{Pos})(\text{def?}(\text{pos}, p.\text{pendientes}) \Rightarrow_L$
 $(\text{obtener}(\text{pos}, p.\text{pendientes}) \in \{ \text{"casa"}, \text{"comercio"} \} \wedge$
 $\neg \text{def?}(\text{pos}, \text{construcciones}(*p.\text{sim})))$
 $)$
 $)$

donde p es `obtener(partida, e)`

`Abs : estr $e \rightarrow$ servidor`

`{Rep(e)}`

`Abs(e) \equiv s: servidor |`
 $(\forall \text{nombre} : \text{Nombre})$
 $(\text{nombre} \in \text{congelados}(s) \iff$
 $(\text{def?}(\text{nombre}, e) \wedge_L \text{obtener}(\text{nombre}, e).\text{modificable} =_{\text{obs}} \text{false}))$
 \wedge
 $(\forall \text{nombre} : \text{Nombre})$
 $(\text{def?}(\text{nombre}, \text{partidas}(s)) \iff \text{def?}(\text{nombre}, e))$
 \wedge_L
 $(\forall \text{nombre} : \text{Nombre})$
 $(\text{def?}(\text{nombre}, \text{partidas}(s)) \Rightarrow_L$
 $(\text{obtener}(\text{nombre}, \text{partidas}(s)) =_{\text{obs}} *(\text{obtener}(\text{nombre}, e).\text{sim}) \wedge$
 $\text{pendientes}(s, \text{nombre}) =_{\text{obs}} \text{obtener}(\text{nombre}, e).\text{pendientes}))$