Trabajo práctico 2: Diseño

Normativa

Límite de entrega: domingo 18 de octubre *hasta las 22:00 hs.* Enviar a algo2.dc@gmail.com
Normas de entrega: Ver "Información sobre la cursada" en el sitio Web de la materia.
(`http://www.dc.uba.ar/materias/aed2/2015/2c/informacion`)

Versión: 1.0 del 29 de septiembre de 2015 (ver TP2_Changelog.txt)

TAD POSICIÓN **es** TUPLA(X:NAT, Y:NAT)
TAD DIRECCIÓN **es** ENUM {IZQ,DER,ARRIBA,ABAJO}
TAD AGENTE **es** NAT
TAD NOMBRE **es** STRING

Especificación TP2

TAD CAMPUS

exporta generadores, observadores, posValida?, esIngreso?, ingresoSuperior?, ingresoInferior?, vecinos, distancia, proxPosición, ingresosMásCercanos

géneros campus

igualdad observacional

$$(\forall c1, c2 : \text{campus}) \left(\begin{array}{l} c1 =_{\text{obs}} c2 \iff \\ \qquad \text{filas}(c1) =_{\text{obs}} \text{filas}(c2) \land \text{columnas}(c1) =_{\text{obs}} \text{columnas}(c2) \\ \qquad \land_{\text{L}} \quad (\forall p : \text{posición})\ (\text{posVálida?}(p,c1) \Rightarrow_{\text{L}} (\text{ocupada?}(p,c1) =_{\text{obs}} \text{ocupada?}(p,c2))) \end{array} \right)$$

observadores básicos

filas	: campus	$\longrightarrow$	nat		
columnas	: campus	$\longrightarrow$	nat		
ocupada?	: posición p × campus c	$\longrightarrow$	nat		{posVálida?(p,c)}

generadores

crearCampus	: nat × nat	$\longrightarrow$	campus	
agregarObstáculo	: posición p × campus c	$\longrightarrow$	campus	{posVálida?(p,c) $\land_{\text{L}}$ ¬ocupada?(p,c)}

otras operaciones

posVálida?	: posición × campus	$\longrightarrow$	bool	
ingresoSuperior?	: posición p × campus c	$\longrightarrow$	bool	
ingresoInferior?	: posición p × campus c	$\longrightarrow$	bool	
esIngreso?	: posición p × campus c	$\longrightarrow$	bool	
vecinos	: posición p × campus c	$\longrightarrow$	conj(posición)	{posVálida?(p,c)}
vecinosComunes	: posición p × posición $p2$ × campus c	$\longrightarrow$	conj(posición)	{posVálida?(p,c) ∧ posVálida?($p2$,c)}
vecinosVálidos	: conj(posición) × campus	$\longrightarrow$	conj(posición)	
distancia	: posición p × posición $p2$ × campus c	$\longrightarrow$	nat	
proxPosición	: posición p × dirección × campus c	$\longrightarrow$	posición	{posVálida?(p,c)}
ingresosMásCercanos	: posición p × campus c	$\longrightarrow$	conj(posición)	{posVálida?(p,c)}

axiomas $(\forall c : \text{campus}, \forall n1, n2 : \text{nat}, \forall p, p2 : \text{posición}, \forall ps : \text{conj(posición)}, \forall dir : \text{dirección})$

- filas(crearCampus($n1$, $n2$)) $\equiv$ $n2$
- filas(agregarObstáculo(p, c)) $\equiv$ filas(c)
- columnas(crearCampus($n1$, $n2$)) $\equiv$ $n1$
- columnas(agregarObstáculo(p, c)) $\equiv$ columnas(c)

ocupada?(p, crearCampus($n1$, $n2$)) $\equiv$ false
ocupada?($p2$, agregarObstáculo(p, c)) $\equiv$ $p2 = p \vee$ ocupada?($p2$,c)
posVálida(p,c) $\equiv$ $(0 < p.X) \wedge (p.X \leq$ filas(c)$) \wedge (0 < p.Y) \wedge (p.Y \leq$ colunas(c))
ingresoSuperior?(p,c) $\equiv$ $p.Y = 1$
ingresoInferior?(p,c) $\equiv$ $p.Y =$ filas(c)
esIngreso?(p,c) $\equiv$ ingresoSuperior?(p,c) $\vee$ ingresoInferior?(p,c)
vecinos(p,c) $\equiv$ vecinosVálidos(Ag($< p.X - 1, p.Y >$, Ag($< p.X + 1, p.Y >$, Ag($< p.X, p.Y - 1 >$, Ag($< p.X, p.Y + 1 >$), $\emptyset$))), c)
vecinosComunes(p,$p2$,c) $\equiv$ vecinos(p,c) $\cap$ vecinos($p2$,c)
vecinosVálidos(ps,c) $\equiv$ **if** $\emptyset$?(ps) **then**
$\emptyset$
else
if posVálida?(dameUno(ps), c) **then**
Ag(dameUno(ps),vecinosVálidos(sinUno(ps),c))
else
vecinosVálidos(sinUno(ps),c)
fi
fi
distancia(p,$p2$,c) $\equiv$ $|p.X - p2.X| + |p.Y - p2.Y|$
proxPosición(p,dir,c) $\equiv$ **if** dir = izq **then**
$< p.X - 1, p.Y >$
else
if dir = der **then**
$< p.X + 1, p.Y >$
else
if dir =arriba **then** $< p.X, p.Y - 1 >$ **else** $< p.X, p.Y + 1 >$ **fi**
fi
fi
ingresosMáCercanos(p,c) $\equiv$ **if** distancia(p, $< p.X, 1 >$,c) $<$ distancia(p, $< p.X$, filas(c)$>$,c) **then**
Ag($< p.X, 1 >$, $\emptyset$)
else
if distancia(p, $< p.X, 1 >$,c) $>$ distancia(p, $< p.X$, filas(c)$>$,c) **then**
Ag($< p.X$, filas(c)$>$,$\emptyset$)
else
($<$Ag($< p.X, 1 >$,Ag($< p.X$, filas(c)$>$,$\emptyset$)))
fi
fi

Fin TAD

TAD CampusSeguro

exporta generadores, observadores, cantHippies, cantEstudiantes, másVigilante

géneros campusSeguro

observadores básicos

campus : campusSeguro $\longrightarrow$ campus

estudiantes : campusSeguro $\longrightarrow$ conj(nombre)

hippies : campusSeguro $\longrightarrow$ conj(nombre)

agentes : campusSeguro $\longrightarrow$ conj(agente)

posEstudianteYHippie : nombre id × campusSeguro cs $\longrightarrow$ posición { id $\in$ (estudiantes(cs) $\cup$ hippies(cs))}

posAgente : agente a × campusSeguro ps $\longrightarrow$ posición {a $\in$ agentes(cs)}

cantSanciones : agente a × campusSeguro ps $\longrightarrow$ nat {a $\in$ agentes(cs)}

cantHippiesAtrapados : agente a × campusSeguro ps $\longrightarrow$ nat {a $\in$ agentes(cs)}

generadores

comenzarRastrillaje : campus c × dicc(agente × posición) d $\longrightarrow$ campusSeguro
$\left\{\begin{array}{l}(\forall a : \text{agente})\ (\text{def?}(a,d) \Rightarrow_{\text{L}} (\text{posVálida}(\text{obtener}(a,d)) \wedge \neg\text{ocupada?}(\text{obtener}(a,d),c))) \wedge (\forall a, a2 : \\ \text{agente})\ ((\text{def?}(a,d) \wedge \text{def?}(a2,d) \wedge a \neq a2) \Rightarrow_{\text{L}} \text{obtener}(a,d) \neq \text{obtener}(a2,d))\end{array}\right\}$

ingresarEstudiante : nombre e × posición p × campusSeguro cs $\longrightarrow$ campusSeguro
{$e \notin$ (estudiantes(cs) $\cup$ hippies(cs)) $\wedge$ esIngreso?(p,campus(cs)) $\wedge$ $\neg$estaOcupada?(p,cs)}

ingresarHippie : nombre h × posición p × campusSeguro cs $\longrightarrow$ campusSeguro
{$h \notin$ (estudiantes(cs) $\cup$ hippies(cs)) $\wedge$ esIngreso?(p,campus(cs)) $\wedge$ $\neg$estaOcupada?(p,cs)}

moverEstudiante : nombre e × dirección d × campusSeguro cs $\longrightarrow$ campusSeguro
$\left\{\begin{array}{l}e \in \text{estudiantes}(cs) \wedge (\text{seRetira}(e,dir,cs) \vee \\ (\text{posVálida}(\text{proxPosición}(\text{posEstudianteYHippie}(e,cs),dir,\text{campus}(cs)), \quad \text{campus}(cs)) \quad \wedge \\ \neg\text{estaOcupada?}(\text{proxPosición}(\text{posEstudianteYHippie}(e,cs),dir,\text{campus}(cs)),\ cs)))\end{array}\right\}$

moverhippie : nombre h × campusSeguro cs $\longrightarrow$ campusSeguro
{$h \in$ hippies(cs) $\wedge$ $\neg$todasOcupadas?(vecinos(posEstudianteYHippie(h,cs), campus(cs)), cs)}

moverAgente : agente a × campusSeguro cs $\longrightarrow$ campusSeguro
$\left\{\begin{array}{l}a \in \text{agentes}(cs) \wedge_{\text{L}} \text{cantSanciones}(a,cs) \leq 3 \wedge \\ \neg\text{todasOcupadas?}(\text{vecinos}(\text{posAgente}(a,cs),\ \text{campus}(cs)),\ cs)\end{array}\right\}$

otras operaciones

estaOcupada? : posición p × campusSeguro cs $\longrightarrow$ bool
{posVálida?(p, campus(cs))}

seRetira : nombre e × dirección dir × campusSeguro cs $\longrightarrow$ bool
{$e \in$ estudiantes(cs)}

todasOcupadas? : conj(posición) ps × campusSeguro cs $\longrightarrow$ bool
{$(\forall p2 : \text{posición})(p2 \in ps \Rightarrow_{\text{L}} \text{posVálida?}(p2,\text{campus}(cs))$}

posConHippies : conj(posición) ps × campusSeguro cs $\longrightarrow$ conj(nombre)
{$(\forall p : \text{posición})(p \in ps \Rightarrow_{\text{L}} \text{posVálida?}(p,\text{campus}(cs))$}

hippiesRodeados : posición p × conj(posición) ps × campusSeguro cs $\longrightarrow$ conj(nombre)
{posVálida?(p, campus(cs)) $\wedge$ $(\forall p2 : \text{posición})(p2 \in ps \Rightarrow_{\text{L}} \text{posVálida?}(p2,\text{campus}(cs))$}

estáRodeadoPorEstudiantes conj(posición) ps × campusSeguro cs $\longrightarrow$ bool
{$(\forall p : \text{posición})(p \in ps \Rightarrow_{\text{L}} \text{posVálida?}(p,\text{campus}(cs))$}

alMenos1Estudiantes : conj(posición) ps × campusSeguro cs $\longrightarrow$ bool
{$(\forall p : \text{posición})(p \in ps \Rightarrow_{\text{L}} \text{posVálida?}(p,\text{campus}(cs))$}

estudiantesHippificados : posición p × conj(posición) ps × campusSeguro cs $\longrightarrow$ conj(nombre)
{posVálida?(p, campus(cs)) $\wedge$ $(\forall p2 : \text{posición})(p2 \in ps \Rightarrow_{\text{L}} \text{posVálida?}(p2,\text{campus}(cs))$}

estudiantesRodeados : posición p × conj(posición) ps × campusSeguro cs $\longrightarrow$ conj(nombre)
{posVálida?(p, campus(cs)) $\wedge$ $(\forall p2 : \text{posición})(p2 \in ps \Rightarrow_{\text{L}} \text{posVálida?}(p2,\text{campus}(cs))$}

quedoAtrapado : conj(posición) ps × campusSeguro cs $\longrightarrow$ bool
{$(\forall p : \text{posición})(p \in ps \Rightarrow_{\text{L}} \text{posVálida?}(p,\text{campus}(cs))$}

alMenos1Agente : conj(posición) ps × campusSeguro cs $\longrightarrow$ bool
{$(\forall p : \text{posición})(p \in ps \Rightarrow_{\text{L}} \text{posVálida?}(p,\text{campus}(cs))$}

agentesDelEquipo : posición p × conj(nombre) ids × campusSeguro cs) $\longrightarrow$ conj(agente)
{posVálida?(p, campus(cs)) $\wedge$ $ids \subseteq$ (hippies(cs) $\cup$ estudiantes(cs))}

agentesDelEquipoAux : conj(posición) ps × campusSeguro cs $\longrightarrow$ conj(agente)
{$(\forall p : \text{posición})(p \in ps \Rightarrow_{\text{L}} \text{posVálida?}(p,\text{campus}(cs))$}

quedoAtrapadoPorA : agente a × nombre id × campusSeguro cs $\longrightarrow$ bool

$\{a \in \text{agentes}(cs) \land id \in \text{hippies}(cs)\}$

hippiesAtrapados : posición p × conj(posición) ps × campusSeguro cs ⟶ conj(nombre)
$\{\text{posVálida?}(p, \text{campus}(cs)) \land (\forall p2 : \text{posición})(p2 \in ps \Rightarrow_{\text{L}} \text{posVálida?}(p2,\text{campus}(cs))\}$

proxPosiciónHippie : nombre h × campusSeguro cs ⟶ posición
$\{h \in \text{hippies}(cs)\}$

proxPosiciónAgente : agente a × campusSeguro cs ⟶ posición
$\{a \in \text{agentes}(cs)\}$

nuevaPosición : posición p × conj(nombre) ids × campusSeguro cs ⟶ posición
$\{\text{posVálida?}(p, \text{campus}(cs)) \land ids \subseteq (\text{hippies}(cs) \cup \text{estudiantes}(cs))\}$

acercarse : posición p × conj(posición) ps × conj(posición) $ps2$ × campusSeguro cs ⟶ posición
$\left\{\begin{array}{l}\text{posVálida?}(p, \text{campus}(cs)) \land ps \subseteq \text{vecinos}(p, \text{campus}(cs)) \land (\forall p2 : \text{posición})(p2 \in ps2 \Rightarrow_{\text{L}} \\ \text{posVálida?}(p2,\text{campus}(cs))\end{array}\right\}$

PosicionesParaAcercarse : posición p × conj(posición) ps × conj(posición) $ps2$ × campusSeguro cs ⟶ conj(posición)
$\left\{\begin{array}{l}\text{posVálida?}(p, \text{campus}(cs)) \land ps \subseteq \text{vecinos}(p, \text{campus}(cs)) \land (\forall p2 : \text{posición})(p2 \in ps2 \Rightarrow_{\text{L}} \\ \text{posVálida?}(p2,\text{campus}(cs))\end{array}\right\}$

seAcercaA : posición p × posición $p2$ × conj(posición) $ps2$ × campusSeguro cs ⟶ bool
$\left\{\begin{array}{l}\text{posVálida?}(p, \text{campus}(cs)) \land p2 \in \text{vecinos}(p, \text{campus}(cs)) \land (\forall p2 : \text{posición})(p2 \in ps2 \Rightarrow_{\text{L}} \\ \text{posVálida?}(p2,\text{campus}(cs))\end{array}\right\}$

losQueEstánADistanciaN : posición p × conj(nombre) ids × nat n × campusSeguro cs ⟶ conj(posición)
$\{\text{posVálida?}(p, \text{campus}(cs)) \land ids \subseteq (\text{hippies}(cs) \cup \text{estudiantes}(cs))\}$

distanciaAlMásCercano : posición p × conj(nombre) ids × campusSeguro cs ⟶ nat
$\{\text{posVálida?}(p, \text{campus}(cs)) \land ids \subseteq (\text{hippies}(cs) \cup \text{estudiantes}(cs))\}$

hayHippieOEstudiante : posición p × conj(DNI) ids × campusSeguro cs ⟶ bool
$\{\text{posVálida?}(p, \text{campus}(cs)) \land ids \subseteq (\text{hippies}(cs) \cup \text{estudiantes}(cs))\}$

esAgente? : posición p × conj(agente) as × campusSeguro cs ⟶ bool
$\{\text{posVálida?}(p, \text{campus}(cs)) \land as \subseteq \text{agentes}(cs)\}$

buscar : posición p × conj(nombre) ids × campusSeguro cs ⟶ nombre
$\left\{\begin{array}{l}\text{posVálida?}(p, \text{campus}(cs)) \land ids \subseteq (\text{hippies}(cs) \cup \text{estudiantes}(cs)) \land_{\text{L}} (\exists id : \text{nombre})\ (id \in ids \land_{\text{L}} \\ \text{posEstudiantesYHippies}(id,cs) = p)\end{array}\right\}$

buscarAgente : posición p × conj(agente) as × campusSeguro cs ⟶ agente
$\{\text{posVálida?}(p, \text{campus}(cs)) \land as \subseteq \text{agentes}(cs) \land_{\text{L}} (\exists a : \text{agente})\ (a \in as \land_{\text{L}} \text{posAgente}(a,cs) = p)\}$

cantHippies : campusSeguro cs ⟶ nat

cantEstudiantes : campusSeguro cs ⟶ nat

masVigilante : campusSeguro cs ⟶ agente

mayorAtrapada : conj(agente) as × campusSeguro cs ⟶ nat
$\{as \subseteq \text{agentes}(cs)\}$

losMasVigilantes : conj(agente) as × nat d × campusSeguro cs ⟶ nat
$\{as \subseteq \text{agentes}(cs)\}$

agenteDeMenorPlaca : conj(agente) as × campusSeguro cs ⟶ nat
$\{as \subseteq \text{agentes}(cs)\}$

conMismasSanciones : agente a × campusSeguro cs ⟶ conj(agente)
$\{a \in \text{agentes}(cs)\}$

conKsanciones : nat × campusSeguro ⟶ conj(agente)

agentesConKsanciones : nat k × conj(agente) as × campusSeguro cs ⟶ conj(agente)
$\{as \subseteq \text{agentes}(cs)\}$

axiomas $(\forall cs : \text{campusSeguro}, \forall c : \text{campus}, \forall d : \text{dicc(agente,posición)}, \forall id, id2 : \text{nombre}, \forall n, k : \text{nat}, \forall dir : \text{dirección}, \forall a, a2 : \text{agente}, \forall p, p2 : \text{posición}, \forall ps,$

$\text{campus}(\text{comenzarRastrillaje}(c, d)) \equiv c$
$\text{campus}(\text{ingresarEstudiante}(id, p, cs)) \equiv \text{campus}(cs)$
$\text{campus}(\text{ingresarHippie}(id, p, cs)) \equiv \text{campus}(cs)$
$\text{campus}(\text{moverEstudiante}(id, dir, cs)) \equiv \text{campus}(cs)$
$\text{campus}(\text{moverHippie}(id, cs)) \equiv \text{campus}(cs)$
$\text{campus}(\text{moverAgente}(a, cs)) \equiv \text{campus}(cs)$
$\text{estudiantes}(\text{comenzarRastrillaje}(c, d)) \equiv \emptyset$

estudiantes(ingresarEstudiante(id, p, cs)) $\equiv$ **if** #posConHippies(vecinos(p, campus(cs)), cs) ≥ 2 **then**
estudiantes(cs)
else
Ag(id, estudiantes(cs))
fi $\cup$ hippiesRodeados(p, vecinos(p, campus(cs)), cs)

estudiantes(ingresarHippie(id, p, cs)) $\equiv$ **if** estáRodeadoPorEstudiantes(vecinos(p, campus(cs)), cs) **then**
Ag(id, estudiantes(cs))
else
estudiantes(cs)
fi - estudiantesHippificados(p, vecinos(p, campus(cs)), cs)

estudiantes(moverEstudiante(id,dir,cs)) $\equiv$ **if** seRetira(id, dir, campus(cs)) **then**
estudiantes(cs) - {id}
else
estudiantes(cs)
fi $\cup$ hippiesRodeados(proxPosición(posEstudianteYHippie(id,cs), dir, campus(cs)), vecinos(proxPosición(posEstudianteYHippie(id, cs), dir, campus(cs), campus(cs)), cs)

estudiantes(moverHippie(id, cs)) $\equiv$ estudiantes(cs) - estudiantesHippificados(proxPosiciónHippie(id, cs), vecinos((proxPosiciónHippie(id, cs), campus(cs)), cs)

estudiantes(moverAgente(a, cs)) $\equiv$ estudantes(cs)

hippies(comenzarRastrillaje(c,d)) $\equiv \emptyset$

hippies(ingresarEstudiante(id, p,cs)) $\equiv$ **if** #vecinosHippies(vecinos(p, campus(cs), cs) ≥ 2) **then**
Ag(id, hippies(cs))
else
hippies(cs)
fi - hippiesRodeados(p, vecinos(p, campus(cs)), cs)

hippies(ingresarHippie(id, p,cs)) $\equiv$ (**if** estáRodeadoPorEstudiantes(vecinos(p, campus(cs)), cs) $\vee$ quedoAtrapado(vecinos(p, campus(cs)),cs) **then**
hippies(cs)
else
Ag(id, hippies(cs))
fi - hippiesACapturar(p, vecinos(p, campus(cs)), cs)) $\cup$ estudiantesHippificados(p, vecinos(p, campus(cs)), cs)

hippies(moverEstudiante(id, dir,cs)) $\equiv$ hippies(cs) - hippiesRodeados(proxPosición(posEstudianteYHippie(id, cs), dir, campus(cs)), vecinos(proxPosición(posEstudianteYHippie(id, cs), dir, campus(cs)), campus(cs)), cs)

hippies(moverHippie(id, cs)) $\equiv$ (hippies(cs) - hippiesACapturar(p, vecinos(p, campus(cs)), cs))) $\cup$ estudiantesHippificados(proxPosiciónHippie(id,cs), vecinos((proxPosiciónHippie(id,cs), campus(cs)), cs)

hippies(moverAgente(a, cs)) $\equiv$ hippies(cs) - hippiesRodeados(proxPosiciónAgente(a,cs), vecinos((proxPosiciónAgente(a,cs), campus(cs)), cs)

agentes(comenzarRastrillaje(c,d)) $\equiv$ claves(d)

agentes(ingresarEstudiante(id, p,cs)) $\equiv$ agentes(cs)

agentes(ingresarHippie(id, p,cs)) $\equiv$ agentes(cs)

agentes(moverEstudiante(id, dir,cs)) $\equiv$ agentes(cs)

agentes(moverHippie(id, cs)) $\equiv$ agentes(cs)

agentes(moverAgente(a, cs)) $\equiv$ agentes(cs)

posEstudianteYHippie($id2$,ingresarEstudiante(id, p,cs)) $\equiv$ **if** $id2 = id$ **then**
p
else
posEstudianteYHippie($id2$, cs)
fi

posEstudianteYHippie($id2$, ingresarHippie(id, p,cs)) $\equiv$ **if** $id2 = id$ **then**
p
else
posEstudianteYHippie($id2$, cs)
fi

posEstudianteYHippie($id2$,moverEstudiante(id,dir,cs)) $\equiv$ **if** $id2 = id$ **then**
proxPosición(posEstudianteYHippie(id, cs), dir, campus(cs))
else
posEstudianteYHippie($id2$, cs)
fi

posEstudianteYHippie($id2$, moverHippie(id, cs)) ≡ **if** $id2 = id$ **then**
proxPosiciónHippie(id, cs)
else
posEstudianteYHippie($id2$, cs)
fi

posEstudianteYHippie(id, moverAgente(a, cs)) ≡ posEstudianteYHippie(id,cs)

posAgentes(a,comenzarRastrillaje(c,d)) ≡ obtener(a,d)

posAgentes(a,ingresarEstudiante(id, p,cs)) ≡ posAgentes(a,cs)

posAgentes(a,ingresarHippie(id, p,cs)) ≡ posAgentes(a,cs)

posAgentes(a,moverEstudiante(id, dir,cs)) ≡ posAgentes(a,cs)

posAgentes(a,moverHippie(id, cs)) ≡ posAgentes(a,cs)

posAgentes($a2$,moverAgente(a, cs)) ≡ **if** $a2 = a$ **then**
proxPosiciónAgente(a,cs)
else
posEstudianteYHippie($a2$, cs)
fi

cantSanciones(a,comenzarRastrillaje(c,d)) ≡ 0

cantSanciones(a,ingresarEstudiante(id, p, cs)) ≡ cantSanciones(a,cs) + **if** $id \in$(estudiantesRodeados(posAgente(a, cs), vecinos(posAgente(a, cs), campus(cs)), ingresarEstudiante(id, p,cs))) **then**
1
else
0
fi + #(estudiantesRodeados(p, vecinosComunes(posAgente(a, cs), p, campus(cs)), cs))

cantSanciones(a,ingresarHippie(id, p,cs)) ≡ cantSanciones(a,cs) + #(estudiantesRodeados(p, vecinosComunes(posAgente(a, cs), p, campus(cs)), ingresarHippie(id, p, cs)))

cantSanciones(a,moverEstudiante(id,dir,cs)) ≡ cantSanciones(a,cs) + #(estudiantesRodeados (proxPosición(posEstudianteYHippie(id,cs), dir, campus(cs)), vecinosComunes(posAgente(a,cs), proxPosición(posEstudianteYHippie(id,cs), dir, campus(cs)), campus(cs)), moverEstudiante(id,dir,cs)))

cantSanciones(a,moverHippie(id, cs)) ≡ cantSanciones(a,cs) + #(estudiantesRodeados(proxPosiciónHippie(id, cs), dir, campus(cs)), vecinos-Comunes(posAgente(a,cs), proxPosiciónHippie(id, cs), campus(cs)), moverHippie(id, cs))

cantSanciones($a2$,moverAgente(a, cs)) ≡ cantSanciones($a2$, cs) + #(estudiantesRodeados(proxPosiciónAgente(a, cs), vecinosComunes(proxPosiciónAgente(a,cs), posAgente($a2$, cs), campus(cs)), moverAgente(a, cs)))

cantHippiesAtrapados(a,comenzarRastrillaje(c,d)) ≡ 0

cantHippiesAtrapados(a,ingresarEstudiante(id, p,cs)) ≡ cantHippiesAtrapados(a,cs)

cantHippiesAtrapados(a,ingresarHippie(id, p,cs)) ≡ cantHippiesAtrapados(a,cs) + **if** quedoAtrapado(vecinos(p,campus(cs)), cs) **then**
1
else
0
fi + #(hippiesACapturar(p,vecinosComunes(p,posAgente(a,cs),ca

cantHippiesAtrapados(a,moverEstudiante(id,dir,cs)) ≡ cantHippiesAtrapados(a,cs) + #(hippiesACaptu-rar(proxPosición(posEstudianteyHippie(id,cs)), vecinosComunes(proxPosición(posEstudianteyHippie(id,cs)), posAgente(a,cs), campus(cs)),cs))

cantHippiesAtrapados(a,moverHippie(id, cs)) ≡ cantHippiesAtrapados(a,cs) + #(hippiesACapturar(proxPosición(posEstudianteyHippie(id,cs)), vecinosComunes(proxPosición(posEstudianteyHippie(id,cs)), posAgente(a,cs), campus(cs)),cs))

cantHippiesAtrapados($a2$,moverAgente(a, cs)) ≡ cantHippiesAtrapados($a2$,cs) + #(hippiesACapturar(proxPosiciónAgente(a,cs), vecinosComunes(proxPosiciónAgente(a,cs),posAgente($a2$,cs),cam

estaOcupada?(p,cs) ≡ ocupada?(p, campus(cs)) ∨ hayHippieOEstudiante(p,hippies(cs),cs) ∨ hayHippieOEstudiante(p,estudiantes(cs),cs) ∨ esAgente?(p,agentes(cs),cs)

seRetira(id,dir,cs) $\equiv$ (ingresoSuperior(posEstudianteYHippie(id,cs), campus(cs)) $\wedge$ dir = Arriba) $\vee$ (ingresoInferior(posEstudianteYHippie(id,cs), campus(cs)) $\wedge$ dir = Abajo)

todasOcupadas?(ps,cs) $\equiv$ **if** $\emptyset$?(ps) **then**
true
else
if estaOcupada?(dameUno(ps),cs) **then**
todasOcupadas?(sinUno(ps),cs)
else
false
fi
fi

posConHippies(ps,cs) $\equiv$ **if** $\emptyset$?(ps) **then**
$\emptyset$
else
if hayHippieOEstudiante(dameUno(ps),hippies(cs), cs) **then**
Ag(dameUno(ps),posConHippies(sinUno(ps), cs))
else
posConHippies(sinUno(ps), cs)
fi
fi

hippiesRodeados(p,ps,cs) $\equiv$ **if** $\emptyset$?(ps) **then**
$\emptyset$
else
if hayHippieOEstudiante(dameUno(ps),hippies(cs), cs) $\wedge$ todasOcupadas?((vecinos(dameUno(ps),campus(cs))-{p}), cs) **then**
Ag(buscar(dameUno(ps),hippies(cs), cs), hippiesRodeados(p,sinUno(ps),cs))
else
hippiesRodeados(p,sinUno(ps),cs
fi
fi

hippiesACapturar(p,ps,cs) $\equiv$ **if** $\emptyset$?(ps) **then**
$\emptyset$
else
if hayHippieOEstudiante(dameUno(ps),hippies(cs), cs) $\wedge$ todasOcupadas?((vecinos(dameUno(ps),campus(cs))-{p}), cs) $\wedge$ alMenos1Agente(vecinos(vecinos(dameUno(ps),campus(cs))-{p}, cs) **then**
Ag(buscar(dameUno(ps),hippies(cs), cs), hippiesACapturar(p,sinUno(ps),cs))
else
hippiesACapturar(p,sinUno(ps),cs
fi
fi

estáRodeadoPorEstudiantes(ps,cs) $\equiv$ todasOcupadas?(ps, cs) $\wedge$ alMenos1Estudiante(ps, cs)

alMenos1Estudiante(ps, cs) $\equiv$ **if** $\emptyset$?(ps) **then**
false
else
if hayHippieOEstudiante(dameUno(ps), cs) **then**
true
else
alMenos1Estudiante(sinUno(ps), cs)
fi
fi

estudiantesHippificados(p, ps,cs) $\equiv$ **if** $\emptyset$?(ps) **then**
$\emptyset$
else
if hayEstudiante(dameUno(ps),estudiantes(cs), cs) $\wedge$ #posConHippies(vecinos(p,campus(cs)), cs) ≥ 1) **then**
Ag(buscar(dameUno(ps),estudiantes(cs), cs), estudiantesHippificados(p,sinUno(ps),cs))
else
estudiantesHippificados(p,sinUno(ps),cs)
fi
fi

estudiantesRodeados(p,ps,cs) $\equiv$ **if** $\emptyset?(ps)$ **then**
$\emptyset$
else
if hayEstudiante(dameUno(ps),estudiantes(cs), cs) $\wedge$ todasOcupadas?((vecinos(dameUno(ps),campus(cs))-{p}), cs) **then**
Ag(buscar(dameUno(ps),estudiantes(cs), cs), estudiantesRodeados(p,sinUno(ps),cs))
else
estudiantesRodeados(p,sinUno(ps),cs)
fi
fi

agentesDelEquipo(p,ids,cs) $\equiv$ **if** $\emptyset?(ids)$ **then**
$\emptyset$
else
agentesDelEquipoAux(vecinos(posEstudiantesYHippies(dameUno(ids), cs), campus(cs))-p, cs) $\cup$ agentesDelEquipo(p,sinUno(ids),cs)
fi

agentesDelEquipoAux(ps,cs) $\equiv$ **if** $\emptyset?(ps)$ **then**
$\emptyset$
else
if esAgente?(dameUno(ps),agentes(cs), cs) **then**
Ag(buscarAgente(dameUno(ps),agentes(cs), cs), agentesDelEquipoAux(p,sinUno(ps),cs))
else
agentesDelEquipoAux(p,sinUno(ps),cs)
fi
fi

quedoAtrapadoPorA(a,id,cs) $\equiv$ posAgente(a,cs) $\in$ vecinos(posEstudiantesyHippies(id,cs), campus(cs)) $\wedge$ todasOcupadas?(vecinos(posEstudiantesyHippies(id,cs), campus(cs), cs)

quedoAtrapado(ps,cs) $\equiv$ todasOcupadas?(ps, cs) $\wedge$ alMenos1Agente(ps, cs)

alMenos1Agente(ps, cs) $\equiv$ **if** $\emptyset?(ps)$ **then**
false
else
if esAgente?(dameUno(ps), cs) **then**
true
else
alMenos1Agente(sinUno(ps), cs)
fi
fi

proxPosiciónHippie(id,cs) $\equiv$ nuevaPosición(posEstudiantes(id,cs), estudiantes(cs), cs)

proxPosiciónAgente(a,cs) $\equiv$ nuevaPosición(posAgente(a,cs), hippies(cs), cs)

nuevaPosició(p,ids, cs) $\equiv$ **if** $\emptyset?(ids)$ **then**
acercarse(p, vecinos(p, campus(cs)),ingresosMásCercanos(p,campus(cs)), cs))
else
acercarse(p, vecinos(p,campus(cs)), losQueEstánADistanciaN(p, ids, distanciaAlMásCercano(p,ids,cs), cs),cs)
fi

acercarse(p,ps, $ps2$,cs) $\equiv$ **if** $\emptyset?$(posicionesParaAcercarse(p,ps, $ps2$,cs)) **then**
p
else
dameUno((posicionesParaAcercarse(p,ps, $ps2$,cs)))
fi

posicionesParaAcercarse(p,ps, $ps2$,cs) $\equiv$ **if** $\emptyset?(ps)$ **then**
$\emptyset$
else
if seAcercaA(p, dameUno(ps), $ps2$,cs) **then**
Ag(ps, posicionesParaAcercarse(p,sinUno(ps), $ps2$,cs))
else
posicionesParaAcercarse(p,sinUno(ps), $ps2$,cs)
fi
fi

```
seAcercaA(p,p2, ps2,cs)  ≡ if ∅?(ps2) then
                              false
                           else
                              if distancia(p2, dameUno(ps2), campus(cs)) <distancia(p,dameUno(ps2),
                              campus(cs)) ∧ ¬estaOcupada((p2), cs) then
                                 true
                              else
                                 seAcerca(p,p2, sinUno(ps2),cs)
                              fi
                           fi
losQueEstánADistanciaN(p,ids, n,cs)  ≡ if ∅?(ids) ∨ n = (filas(campus(cs)) + columnas(campus(cs)) + 1)
                                       then
                                          ∅
                                       else
                                          if distancia(p, posEstudiantesYHippies(dameUno(ids),cs),
                                          campus(cs)) = n then
                                             Ag(posEstudiantesYHippies(dameUno(ids),cs),
                                             losQueEstánADistanciaN(p,sinUno(ids), n,cs))
                                          else
                                             losQueEstánADistanciaN(p,sinUno(ids), n,cs)
                                          fi
                                       fi
distanciaAlMásCercano(p,ids, cs)  ≡ if ∅?(ids) then
                                       filas(campus(cs)) + columnas(campus(cs)) + 1
                                    else
                                       min(distancia(p,posEstudiantesYHippies(dameUno(ids),
                                       cs),campus(cs)) distanciaAlMásCercano(p,sinUno(ids), cs))
                                    fi
hayHippie(p,ids, cs)  ≡ if ∅?(ids) then
                           false
                        else
                           if posEstudiantesYHippies(dameUno(ids),cs) = p then
                              true
                           else
                              hayHippieOEstudiante(p,sinUno(ids), cs)
                           fi
                        fi
esAgente?(p,as, cs)  ≡ if ∅?(as) then
                           false
                        else
                           if posAgente(dameUno(as),cs) = p then
                              true
                           else
                              esAgente?(p,sinUno(as), cs)
                           fi
                        fi
buscar(p,ids, cs)  ≡ if posEstudiantesYHippies(dameUno(ids),cs) = p then
                        dameUno(ids)
                     else
                        buscar(p,sinUno(ids), cs)
                     fi
buscarAgente(p,as, cs)  ≡ if posAgente(dameUno(as),cs) = p then
                             dameUno(as)
                          else
                             buscar(p,sinUno(as), cs)
                          fi
cantHippies(cs)  ≡ #(hippies(cs))
cantEstudiantes(cs)  ≡ #(estudiantes(cs))
masVigilante(cs)  ≡ agenteDeMenorPlaca(losMasVigilantes(agentes(cs), mayorAtrapada(agentes(cs),cs), cs),
                    cs)
agenteDeMenorPlaca(as, cs)  ≡ if #(as) = 1 then
                                 dameUno(as)
                              else
                                 min(dameUno(as),agenteDeMenorPlaca(sinUno(as), cs))
                              fi
```

mayorAtrapada(as, cs) $\equiv$ **if** $\emptyset?(as)$ **then**
0
else
max(cantHippiesAtrapados(dameUno(as),cs),mayorAtrapada(sinUno(as), cs))
fi

losMasVigilantes(as,n, cs) $\equiv$ **if** $\emptyset?(as)$ **then**
$\emptyset$
else
if cantHippiesAtrapados(dameUno(as),cs) $= n$ **then**
Ag(dameUno(as), losMasVigilantes(sinUno(as), d,cs))
else
losMasVigilantes(sinUno(as), n,cs)
fi
fi

conMismasSanciones(a, cs) $\equiv$ agentesConKsanciones(cantSanciones(a,cs), agentes(cs), cs)

conKsanciones(k,cs) $\equiv$ agentesConKsanciones(k, agentes(cs) , cs)

agentesConKsanciones(k,as, cs) $\equiv$ **if** $\emptyset?(as)$ **then**
$\emptyset$
else
if cantSanciones(dameUno(as),cs) $= k$ **then**
Ag(dameUno(as), agentesConKsanciones(k,sinUno(as), cs))
else
agentesConKsanciones(k,sinUno(as), cs)
fi
fi

Fin TAD