

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: 1.3 del 25 de octubre 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, posVálida?, esIngreso?, ingresoSuperior?, ingresoInferior?,
vecinos, distancia, proxPosición, ingresosMásCercanos

géneros campus

igualdad observacional

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

observadores básicos

filas : campus \rightarrow nat

columnas : campus \rightarrow nat

ocupada? : posición $p \times$ campus $c \rightarrow$ bool {posVálida?(p,c)}

generadores

crearCampus : nat \times nat \rightarrow campus

agregarObstáculo : posición $p \times$ campus $c \rightarrow$ campus {posVálida?(p,c) \wedge_{L} \neg ocupada?(p,c)}

otras operaciones

posVálida? : posición \times campus \rightarrow bool

ingresoSuperior? : posición $p \times$ campus $c \rightarrow$ bool

ingresoInferior? : posición $p \times$ campus $c \rightarrow$ bool

esIngreso? : posición $p \times$ campus $c \rightarrow$ bool

vecinos : posición $p \times$ campus $c \rightarrow$ conj(posición) {posVálida?(p,c)}

vecinosComunes : posición $p \times$ posición $p2 \times$ campus $c \rightarrow$ conj(posición) {posVálida?(p,c) \wedge posVálida?(p2,c)}

vecinosVálidos : conj(posición) \times campus \rightarrow conj(posición)

distancia : posición $p \times$ posición $p2 \times$ campus $c \rightarrow$ nat

proxPosición : posición $p \times$ dirección \times campus $c \rightarrow$ posición {posVálida?(p,c)}

ingresosMásCercanos : posición $p \times$ campus $c \rightarrow$ conj(posición) {posVálida?(p,c)}

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

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))    ≡ false
ocupada?(p2, agregarObstáculo(p, c)) ≡ p2 = p ∨ ocupada?(p2,c)
posVálida?(p,c)                     ≡ (0 < p.X) ∧ (p.X ≤ filas(c)) ∧ (0 < p.Y) ∧ (p.Y ≤ columnas(c))
ingresoSuperior?(p,c)               ≡ p.Y = 1
ingresoInferior?(p,c)               ≡ p.Y = filas(c)
esIngreso?(p,c)                     ≡ ingresoSuperior?(p,c) ∨ ingresoInferior?(p,c)
vecinos(p,c)                       ≡ 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 >, ∅))), c)
vecinosComunes(p,p2,c)              ≡ vecinos(p,c) ∩ vecinos(p2,c)
vecinosVálidos(ps,c)                ≡ if ∅?(ps) then
    ∅
    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)                   ≡ |p.X - p2.X| + |p.Y - p2.Y|
proxPosición(p,dir,c)               ≡ 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)             ≡ if distancia(p, < p.X, 1 >,c) < distancia(p, < p.X, filas(c)>,c) then
    Ag(< p.X, 1 >, ∅)
    else
        if distancia(p, < p.X, 1 >,c) > distancia(p, < p.X, filas(c)>,c) then
            Ag(< p.X, filas(c)>,∅)
        else
            (Ag(< p.X, 1 >,Ag(< p.X, filas(c)>,∅)))
        fi
    fi
fi

```

Fin TAD

TAD CAMPUSSEGURO

exporta generadores, observadores, cantHippies, cantEstudiantes, másVigilante

géneros campusSeguro

observadores básicos

campus : campusSeguro \rightarrow campus
 estudiantes : campusSeguro \rightarrow conj(nombre)
 hippies : campusSeguro \rightarrow conj(nombre)
 agentes : campusSeguro \rightarrow conj(agente)
 posEstudianteYHippie: nombre $id \times$ campusSeguro $cs \rightarrow$ posición { $id \in (\text{estudiantes}(cs) \cup \text{hippies}(cs))$ }
 posAgente : agente $a \times$ campusSeguro $ps \rightarrow$ posición { $a \in \text{agentes}(cs)$ }
 cantSanciones : agente $a \times$ campusSeguro $ps \rightarrow$ nat { $a \in \text{agentes}(cs)$ }
 cantHippiesAtrapados : agente $a \times$ campusSeguro $ps \rightarrow$ nat { $a \in \text{agentes}(cs)$ }

generadores

comenzarRastrillaje: campus $c \times$ dicc(agente \times posición) $d \rightarrow$ campusSeguro
 $\left\{ (\forall a : \text{agente}) (\text{def?}(a, d) \Rightarrow_L (\text{posVálida?}(\text{obtener}(a, d)) \wedge \neg \text{ocupada?}(\text{obtener}(a, d), c))) \wedge (\forall a, a2 : \right.$
 $\left. \text{agente}) ((\text{def?}(a, d) \wedge \text{def?}(a2, d) \wedge a \neq a2) \Rightarrow_L \text{obtener}(a, d) \neq \text{obtener}(a2, d)) \right\}$
 ingresarEstudiante : nombre $e \times$ posición $p \times$ campusSeguro $cs \rightarrow$ campusSeguro
 $\{e \notin (\text{estudiantes}(cs) \cup \text{hippies}(cs)) \wedge \text{esIngreso?}(p, \text{campus}(cs)) \wedge \neg \text{estaOcupada?}(p, cs)\}$
 ingresarHippie : nombre $h \times$ posición $p \times$ campusSeguro $cs \rightarrow$ campusSeguro
 $\{h \notin (\text{estudiantes}(cs) \cup \text{hippies}(cs)) \wedge \text{esIngreso?}(p, \text{campus}(cs)) \wedge \neg \text{estaOcupada?}(p, cs)\}$
 moverEstudiante : nombre $e \times$ dirección $d \times$ campusSeguro $cs \rightarrow$ campusSeguro
 $\left\{ \begin{array}{l} e \in \text{estudiantes}(cs) \wedge (\text{seRetira}(e, \text{dir}, cs) \vee \\ (\text{posVálida?}(\text{proxPosición}(\text{posEstudianteYHippie}(e, cs), \text{dir}, \text{campus}(cs)), \text{campus}(cs)) \wedge \\ \neg \text{estaOcupada?}(\text{proxPosición}(\text{posEstudianteYHippie}(e, cs), \text{dir}, \text{campus}(cs)), cs))) \end{array} \right\}$
 moverhippie : nombre $h \times$ campusSeguro $cs \rightarrow$ campusSeguro
 $\{h \in \text{hippies}(cs) \wedge \neg \text{todasOcupadas?}(\text{vecinos}(\text{posEstudianteYHippie}(h, cs), \text{campus}(cs)), cs)\}$
 moverAgente : agente $a \times$ campusSeguro $cs \rightarrow$ campusSeguro
 $\left\{ \begin{array}{l} a \in \text{agentes}(cs) \wedge_L \text{cantSanciones}(a, cs) \leq 3 \wedge \\ \neg \text{todasOcupadas?}(\text{vecinos}(\text{posAgente}(a, cs), \text{campus}(cs)), cs) \end{array} \right\}$

otras operaciones

estaOcupada? : posición $p \times$ campusSeguro $cs \rightarrow$ bool
 $\{\text{posVálida?}(p, \text{campus}(cs))\}$
 seRetira : nombre $e \times$ dirección $dir \times$ campusSeguro $cs \rightarrow$ bool
 $\{e \in \text{estudiantes}(cs)\}$
 todasOcupadas? : conj(posición) $ps \times$ campusSeguro $cs \rightarrow$ bool
 $\{(\forall p2 : \text{posición}) (p2 \in ps \Rightarrow_L \text{posVálida?}(p2, \text{campus}(cs)))\}$
 posConHippies : conj(posición) $ps \times$ campusSeguro $cs \rightarrow$ conj(nombre)
 $\{(\forall p : \text{posición}) (p \in ps \Rightarrow_L \text{posVálida?}(p, \text{campus}(cs)))\}$
 hippiesRodeados : posición $p \times$ conj(posición) $ps \times$ campusSeguro $cs \rightarrow$ conj(nombre)
 $\{\text{posVálida?}(p, \text{campus}(cs)) \wedge (\forall p2 : \text{posición}) (p2 \in ps \Rightarrow_L \text{posVálida?}(p2, \text{campus}(cs)))\}$
 hippiesConvertidos : posición $p \times$ conj(posición) $ps \times$ campusSeguro $cs \rightarrow$ conj(nombre)
 $\{\text{posVálida?}(p, \text{campus}(cs)) \wedge (\forall p2 : \text{posición}) (p2 \in ps \Rightarrow_L \text{posVálida?}(p2, \text{campus}(cs)))\}$
 estáRodeadoPorEstudiantes conj(posición) $ps \times$ campusSeguro $cs \rightarrow$ bool
 $\{(\forall p : \text{posición}) (p \in ps \Rightarrow_L \text{posVálida?}(p, \text{campus}(cs)))\}$
 todosEstudiantes : conj(posición) $ps \times$ campusSeguro $cs \rightarrow$ bool
 $\{(\forall p : \text{posición}) (p \in ps \Rightarrow_L \text{posVálida?}(p, \text{campus}(cs)))\}$
 estudiantesHippificados : posición $p \times$ conj(posición) $ps \times$ campusSeguro $cs \rightarrow$ conj(nombre)
 $\{\text{posVálida?}(p, \text{campus}(cs)) \wedge (\forall p2 : \text{posición}) (p2 \in ps \Rightarrow_L \text{posVálida?}(p2, \text{campus}(cs)))\}$
 estudiantesRodeados : posición $p \times$ conj(posición) $ps \times$ campusSeguro $cs \rightarrow$ conj(nombre)
 $\{\text{posVálida?}(p, \text{campus}(cs)) \wedge (\forall p2 : \text{posición}) (p2 \in ps \Rightarrow_L \text{posVálida?}(p2, \text{campus}(cs)))\}$
 quedoAtrapado : conj(posición) $ps \times$ campusSeguro $cs \rightarrow$ bool
 $\{(\forall p : \text{posición}) (p \in ps \Rightarrow_L \text{posVálida?}(p, \text{campus}(cs)))\}$
 alMenos1Agente : conj(posición) $ps \times$ campusSeguro $cs \rightarrow$ bool
 $\{(\forall p : \text{posición}) (p \in ps \Rightarrow_L \text{posVálida?}(p, \text{campus}(cs)))\}$
 agentesDelEquipo : posición $p \times$ conj(nombre) $ids \times$ campusSeguro $cs \rightarrow$ conj(agente)
 $\{\text{posVálida?}(p, \text{campus}(cs)) \wedge ids \subseteq (\text{hippies}(cs) \cup \text{estudiantes}(cs))\}$
 agentesDelEquipoAux : conj(posición) $ps \times$ campusSeguro $cs \rightarrow$ conj(agente)

		$\{(\forall p : \text{posición})(p \in ps \Rightarrow_L \text{posVálida?}(p, \text{campus}(cs)))\}$
quedoAtrapadoPorA	: agente $a \times$ nombre $id \times$ campusSeguro cs	$\rightarrow \text{bool}$ $\{a \in \text{agentes}(cs) \wedge id \in \text{hippies}(cs)\}$
hippiesAtrapados	: posición $p \times$ conj(posición) $ps \times$ campusSeguro cs	$\rightarrow \text{conj}(\text{nombre})$ $\{\text{posVálida?}(p, \text{campus}(cs)) \wedge (\forall p2 : \text{posición})(p2 \in ps \Rightarrow_L \text{posVálida?}(p2, \text{campus}(cs)))\}$
proxPosiciónHippie	: nombre $h \times$ campusSeguro cs	$\rightarrow \text{posición}$ $\{h \in \text{hippies}(cs)\}$
proxPosiciónAgente	: agente $a \times$ campusSeguro cs	$\rightarrow \text{posición}$ $\{a \in \text{agentes}(cs)\}$
nuevaPosición	: posición $p \times$ conj(nombre) $ids \times$ campusSeguro cs	$\rightarrow \text{posición}$ $\{\text{posVálida?}(p, \text{campus}(cs)) \wedge ids \subseteq (\text{hippies}(cs) \cup \text{estudiantes}(cs))\}$
acercarse	: posición $p \times$ conj(posición) $ps \times$ conj(posición) $ps2 \times$ campusSeguro cs	$\rightarrow \text{posición}$ $\{\text{posVálida?}(p, \text{campus}(cs)) \wedge ps \subseteq \text{vecinos}(p, \text{campus}(cs)) \wedge (\forall p2 : \text{posición})(p2 \in ps2 \Rightarrow_L \text{posVálida?}(p2, \text{campus}(cs)))\}$
PosicionesParaAcercarse	: posición $p \times$ conj(posición) $ps \times$ conj(posición) $ps2 \times$ campusSeguro cs	$\rightarrow \text{conj}(\text{posición})$ $\{\text{posVálida?}(p, \text{campus}(cs)) \wedge ps \subseteq \text{vecinos}(p, \text{campus}(cs)) \wedge (\forall p2 : \text{posición})(p2 \in ps2 \Rightarrow_L \text{posVálida?}(p2, \text{campus}(cs)))\}$
seAcercaA	: posición $p \times$ posición $p2 \times$ conj(posición) $ps2 \times$ campusSeguro cs	$\rightarrow \text{bool}$ $\{\text{posVálida?}(p, \text{campus}(cs)) \wedge p2 \in \text{vecinos}(p, \text{campus}(cs)) \wedge (\forall p2 : \text{posición})(p2 \in ps2 \Rightarrow_L \text{posVálida?}(p2, \text{campus}(cs)))\}$
losQueEstánADistanciaN	: posición $p \times$ conj(nombre) $ids \times$ nat $n \times$ campusSeguro cs	$\rightarrow \text{conj}(\text{posición})$ $\{\text{posVálida?}(p, \text{campus}(cs)) \wedge ids \subseteq (\text{hippies}(cs) \cup \text{estudiantes}(cs))\}$
distanciaAlMásCercano	: posición $p \times$ conj(nombre) $ids \times$ campusSeguro cs	$\rightarrow \text{nat}$ $\{\text{posVálida?}(p, \text{campus}(cs)) \wedge ids \subseteq (\text{hippies}(cs) \cup \text{estudiantes}(cs))\}$
hayHippieOEstudiante	: posición $p \times$ conj(nombre) $ids \times$ campusSeguro cs	$\rightarrow \text{bool}$ $\{\text{posVálida?}(p, \text{campus}(cs)) \wedge ids \subseteq (\text{hippies}(cs) \cup \text{estudiantes}(cs))\}$
esAgente?	: posición $p \times$ conj(agente) $as \times$ campusSeguro cs	$\rightarrow \text{bool}$ $\{\text{posVálida?}(p, \text{campus}(cs)) \wedge as \subseteq \text{agentes}(cs)\}$
buscar	: posición $p \times$ conj(nombre) $ids \times$ campusSeguro cs	$\rightarrow \text{nombre}$ $\{\text{posVálida?}(p, \text{campus}(cs)) \wedge ids \subseteq (\text{hippies}(cs) \cup \text{estudiantes}(cs)) \wedge_L (\exists id : \text{nombre}) (id \in ids \wedge_L \text{posEstudianteYHippie}(id, cs) = p)\}$
buscarAgente	: posición $p \times$ conj(agente) $as \times$ campusSeguro cs	$\rightarrow \text{agente}$ $\{\text{posVálida?}(p, \text{campus}(cs)) \wedge as \subseteq \text{agentes}(cs) \wedge_L (\exists a : \text{agente}) (a \in as \wedge_L \text{posAgente}(a, cs) = p)\}$
cantHippies	: campusSeguro cs	$\rightarrow \text{nat}$
cantEstudiantes	: campusSeguro cs	$\rightarrow \text{nat}$
masVigilante	: campusSeguro cs	$\rightarrow \text{agente}$
mayorAtrapada	: conj(agente) $as \times$ campusSeguro cs	$\rightarrow \text{nat}$ $\{as \subseteq \text{agentes}(cs)\}$
losMasVigilantes	: conj(agente) $as \times$ nat $d \times$ campusSeguro cs	$\rightarrow \text{nat}$ $\{as \subseteq \text{agentes}(cs)\}$
agenteDeMenorPlaca	: conj(agente) $as \times$ campusSeguro cs	$\rightarrow \text{nat}$ $\{as \subseteq \text{agentes}(cs)\}$
conMismasSanciones	: agente $a \times$ campusSeguro cs	$\rightarrow \text{conj}(\text{agente})$ $\{a \in \text{agentes}(cs)\}$
conKsanciones	: nat $k \times$ campusSeguro	$\rightarrow \text{conj}(\text{agente})$
agentesConKsanciones	: nat $k \times$ conj(agente) $as \times$ campusSeguro cs	$\rightarrow \text{conj}(\text{agente})$ $\{as \subseteq \text{agentes}(cs)\}$

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

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

```

estudiantes(ingresarEstudiante(id, p, cs))  ≡ if #posConHippies(vecinos(p, campus(cs)), cs) ≥ 2 then
    estudiantes(cs)
    else
        Ag(id, estudiantes(cs))
    fi ∪ hippiesRodeados(p, vecinos(p, campus(cs)), cs)
estudiantes(ingresarHippie(id, p, cs))      ≡ 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))  ≡ if seRetira(id, dir, campus(cs)) then
    estudiantes(cs) - {id}
    else
        estudiantes(cs)
        fi ∪ hippiesConvertidos( proxPosición(posEstudianteYHippie(id, cs),
            dir, campus(cs)), vecinos(proxPosición(posEstudianteYHippie(id,
            cs), dir, campus(cs), campus(cs)), cs)
estudiantes(moverHippie(id, cs))           ≡ estudiantes(cs) - estudiantesHippificados(proxPosiciónHippie(id,
            cs), vecinos((proxPosiciónHippie(id, cs), campus(cs)), cs)
estudiantes(moverAgente(a, cs))           ≡ estudiantes(cs)
hippies(comenzarRastrillaje(c, d))        ≡ ∅
hippies(ingresarEstudiante(id, p, cs))    ≡ 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))        ≡ ( if estáRodeadoPorEstudiantes(vecinos(p, campus(cs)), cs) ∨
    quedoAtrapado(vecinos(p, campus(cs)), cs) then
        hippies(cs)
    else
        Ag(id, hippies(cs))
        fi - hippiesACapturar(p, vecinos(p, campus(cs)), cs) ∪
        estudiantesHippificados(p, vecinos(p, campus(cs)), cs)
hippies(moverEstudiante(id, dir, cs))    ≡ 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))             ≡ (hippies(cs) - hippiesACapturar(p, vecinos(p, campus(cs)),
            cs)) ∪ estudiantesHippificados(proxPosiciónHippie(id, cs),
            vecinos((proxPosiciónHippie(id, cs), campus(cs)), cs)
hippies(moverAgente(a, cs))              ≡ hippies(cs) - hippiesRodeados(proxPosiciónAgente(a, cs),
            vecinos((proxPosiciónAgente(a, cs), campus(cs)), cs)
agentes(comenzarRastrillaje(c, d))        ≡ claves(d)
agentes(ingresarEstudiante(id, p, cs))    ≡ agentes(cs)
agentes(ingresarHippie(id, p, cs))        ≡ agentes(cs)
agentes(moverEstudiante(id, dir, cs))    ≡ agentes(cs)
agentes(moverHippie(id, cs))              ≡ agentes(cs)
agentes(moverAgente(a, cs))              ≡ agentes(cs)
posEstudianteYHippie(id2, ingresarEstudiante(id, p, cs)) ≡ if id2 = id then
    p
    else
        posEstudianteYHippie(id2, cs)
    fi
posEstudianteYHippie(id2, ingresarHippie(id, p, cs))  ≡ if id2 = id then
    p
    else
        posEstudianteYHippie(id2, cs)
    fi
posEstudianteYHippie(id2, moverEstudiante(id, dir, cs)) ≡ 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 ∈ (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 (proxPo-
sición( 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) + # (estudiantesRodea-
dos( 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) +
if seHippifica(vecinos(p,campus(cs)), cs) ∧
quedoAtrapadoPorA(a, vecinos(p,campus(cs)), cs)
then
1
else
# (hippiesACapturar(p,vecinosComunes
(p,posAgente(a,cs),campus(cs)),cs))
fi
cantHippiesAtrapados(a,ingresarHippie(id, p,cs))    ≡ cantHippiesAtrapados(a,cs) +
if quedoAtrapadoPorA(a, vecinos(p,campus(cs)), cs)
then
1
else
# (hippiesACapturar(p,vecinosComunes
(p,posAgente(a,cs),campus(cs)),cs))
fi
cantHippiesAtrapados(a,moverEstudiante(id,dir,cs))  ≡ cantHippiesAtrapados(a,cs) + # (hippiesACapturar
(proxPosición( posEstudianteYHippie(id,cs)), vecinos-
Comunes(proxPosición( posEstudianteYHippie(id,cs)),
posAgente(a,cs), campus(cs)),cs))

```

```

cantHippiesAtrapados(a,moverHippie(id, cs))      ≡ cantHippiesAtrapados(a,cs) + #(hippiesACapturar
                                                    (proxPosición(posEstudianteyHippie(id,cs)), vecinos-
Comunes(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),campus(cs)),cs))

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) ≡ (ingresoSuperior(posEstudianteYHippie(id,cs), campus(cs)) ∧ dir = Arriba) ∨
(ingresoInferior(posEstudianteYHippie(id,cs), campus(cs)) ∧ dir = Abajo)

todasOcupadas?(ps,cs) ≡ if ∅?(ps) then
    true
else
    if estaOcupada?(dameUno(ps),cs) then
        todasOcupadas?(sinUno(ps),cs)
    else
        false
    fi
fi

posConHippies(ps,cs) ≡ if ∅?(ps) then
    ∅
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) ≡ if ∅?(ps) then
    ∅
else
    if hayHippieOEstudiante(dameUno(ps),hippies(cs), cs) ∧
todasOcupadas?((vecinos(dameUno(ps),campus(cs))-{p}), cs) ∧
(alMenos1Agente(vecinos(dameUno(ps)), cs) ∨
todosEstudiantes(vecinos(dameUno(ps)), cs)) then
        Ag(buscar(dameUno(ps),hippies(cs), cs), hippiesRodeados(p,sinUno(ps),cs))
    else
        hippiesRodeados(p,sinUno(ps),cs)
    fi
fi

hippiesConvertidos(p,ps,cs) ≡ if ∅?(ps) then
    ∅
else
    if hayHippieOEstudiante(dameUno(ps),hippies(cs), cs) ∧
todasOcupadas?((vecinos(dameUno(ps),campus(cs))-{p}), cs) ∧
todosEstudiantes(vecinos(dameUno(ps)), cs) then
        Ag(buscar(dameUno(ps),hippies(cs), cs), hippiesConvertidos(p,sinUno(ps),cs))
    else
        hippiesConvertidos(p,sinUno(ps),cs)
    fi
fi

hippiesACapturar(p,ps,cs) ≡ if ∅?(ps) then
    ∅
else
    if hayHippieOEstudiante(dameUno(ps),hippies(cs), cs) ∧
todasOcupadas?((vecinos(dameUno(ps),campus(cs))-{p}), cs) ∧
alMenos1Agente(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) ≡ todasOcupadas?(ps, cs) ∧ todosEstudiantes(ps, cs)

```

```

todosEstudiantes( $ps, cs$ )  $\equiv$  if  $\emptyset?(ps)$  then
    true
else
    if hayHippieOEstudiante(dameUno( $ps$ ),  $cs$ ) then
        todosEstudiantes(sinUno( $ps$ ),  $cs$ )
    else
        false
    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( $cs$ )),  $cs$ )  $\geq 1$  then
        Ag(buscar(dameUno( $ps$ ), estudiantes( $cs$ ),
            estudiantesHippificados( $p$ , sinUno( $ps$ ),  $cs$ ))
         $cs$ ),
        else
            estudiantesHippificados( $p$ , sinUno( $ps$ ),  $cs$ )
        fi
    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( $cs$ ),
            estudiantesRodeados( $p$ , sinUno( $ps$ ),  $cs$ ))
         $cs$ ),
        else
            estudiantesRodeados( $p$ , sinUno( $ps$ ),  $cs$ )
        fi
    fi
fi

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

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

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

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

seHippifica( $ps, cs$ )  $\equiv$  todasOcupadas?( $ps, cs$ )  $\wedge$  (#posConHippies( $ps, cs$ )  $\geq 2$ )

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( $cs$ ),  $cs$ )

nuevaPosición( $p, ids, cs$ )  $\equiv$  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( $p, ids, cs$ ),  $cs$ ))
    fi
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( $p, p2, ps2, cs$ )  $\equiv$  if  $\emptyset?(ps2)$  then
    false
else
    if distancia( $p2, dameUno(ps2), campus(cs)$ ) < distancia( $p, dameUno(ps2), campus(cs)$ )  $\wedge$   $\neg$ estaOcupada(( $p2$ ),  $cs$ ) then
        true
    else
        seAcercaA( $p, p2, sinUno(ps2), cs$ )
    fi
fi
losQueEstánADistanciaN( $p, ids, n, cs$ )  $\equiv$  if  $\emptyset?(ids) \vee n = (\text{filas}(\text{campus}(cs)) + \text{columnas}(\text{campus}(cs)) + 1)$  then
     $\emptyset$ 
else
    if distancia( $p, posEstudianteYHippie(dameUno(ids), cs), campus(cs)$ ) =  $n$  then
        Ag( $posEstudianteYHippie(dameUno(ids), cs), losQueEstánADistanciaN(p, sinUno(ids), n, cs)$ )
    else
        losQueEstánADistanciaN( $p, sinUno(ids), n, cs$ )
    fi
fi
distanciaAlMásCercano( $p, ids, cs$ )  $\equiv$  if  $\emptyset?(ids)$  then
     $\text{filas}(\text{campus}(cs)) + \text{columnas}(\text{campus}(cs)) + 1$ 
else
     $\min(\text{distancia}(p, posEstudianteYHippie(dameUno(ids), cs), \text{campus}(cs)), \text{distanciaAlMásCercano}(p, sinUno(ids), cs))$ 
fi
hayHippieOEstudiante( $p, ids, cs$ )  $\equiv$  if  $\emptyset?(ids)$  then
    false
else
    if  $posEstudianteYHippie(dameUno(ids), cs) = p$  then
        true
    else
        hayHippieOEstudiante( $p, sinUno(ids), cs$ )
    fi
fi
esAgente?( $p, as, cs$ )  $\equiv$  if  $\emptyset?(as)$  then
    false
else
    if  $posAgente(dameUno(as), cs) = p$  then
        true
    else
        esAgente?( $p, sinUno(as), cs$ )
    fi
fi
buscar( $p, ids, cs$ )  $\equiv$  if  $posEstudianteYHippie(dameUno(ids), cs) = p$  then
    dameUno( $ids$ )
else
    buscar( $p, sinUno(ids), cs$ )
fi

```

```

buscarAgente( $p, as, cs$ )       $\equiv$  if posAgente(dameUno( $as$ ), $cs$ ) =  $p$  then
                                dameUno( $as$ )
                                else
                                    buscar( $p, sinUno(as), cs$ )
                                fi
cantHippies( $cs$ )       $\equiv$  #(hippies( $cs$ ))
cantEstudiantes( $cs$ )  $\equiv$  #(estudiantes( $cs$ ))
masVigilante( $cs$ )  $\equiv$  agenteDeMenorPlaca(losMasVigilantes(agentes( $cs$ ), mayorAtrapada(agentes( $cs$ ), $cs$ ),  $cs$ ),
                                 $cs$ )
agenteDeMenorPlaca( $as, cs$ )  $\equiv$  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