Algoritmos y Estructuras de Datos II

Primer Cuatrimestre de 2015

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Trabajo Prï¿½ctico 1

Especificacii¿½n

Integrante	LU	Correo electrónico
BENITEZ, Nelson	945/13	nelson.benitez92@gmail.com
BENZO, Mariano	198/14	marianobenzo@gmail.com
FARIAS, Mauro	821/13	farias.mauro@hotmail.com
GUTTMAN, Martin	686/14	haris@live.com.ar

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. TAD AS

TAD AS

géneros as

igualdad observacional

$$(\forall facu, facu' : \text{as}) \left(facu =_{\text{obs}} facu' \iff \begin{pmatrix} \text{campus(facu)=campus(facu')} \\ \wedge \text{ seguridad(facu)=seguridad(facu')} \\ \wedge\ (\forall\ pos\text{:p})(\text{posValida(campus(facu),p))} \\ \text{hayEst?(facu,p)} \iff \text{hayEst?(facu',p)} \\ \wedge\ (\forall\ pos\text{:p})(\text{posValida(campus(facu),p))} \\ \text{hayHippie?(facu,p)} \iff \text{hayHippie?(facu',p)} \\ \wedge\ (\forall\ seg\text{:s})(\text{s}\in\text{seguridad(a))} \\ \text{(\#capturas(facu,s)=\#capturas(facu',s)} \\ \wedge\ \text{\#sanciones(facu,s)=\#sanciones(facu',s))} \end{pmatrix} \right)$$

usa CAMPUS,BOOL,NAT,TUPLA,SEG

exporta AS,generadores, observadores,#hippies,#estudiantes,#masVigilante

observadores básicos

campus : as $\longrightarrow$ campus

seguridad : as $\longrightarrow$ conj(seguridad)

hayEst? : as a × pos p $\longrightarrow$ bool $\{posValida(campus(a), p)\}$

hayHippie? : as a × pos p $\longrightarrow$ bool $\{posValida(campus(a), p)\}$

#capturas : as a × seg s $\longrightarrow$ nat $\{s \in seguridad(a)\}$

#sanciones : as a × seg s $\longrightarrow$ nat $\{s \in seguridad(a)\}$

generadores

nueva : campus × conj(seguridad) $\longrightarrow$ as
{($\forall$ *segs*:e) posValida(c,pos(e)) $\wedge$ ($\forall$ *segs*:s,s1) id(s)!=id(s1) $\Rightarrow$ pos(s)!=pos(s1)}

moverEst : as a × pos pe × pos pd $\longrightarrow$ as
$\{posValida(campus(a), pe) \wedge_{\text{L}} hayEst?(a, pe) \wedge adyacente(campus(a), pe, pd) \wedge posValidaPersona(as, pd)\}$

nuevoHippie : as a × pos p $\longrightarrow$ as $\{posIngreso(campus(a), p) \wedge posValidaPersona(a, p)\}$

nuevoEst : as a × pos p $\longrightarrow$ as $\{posIngreso(campus(a), p) \wedge posValidaPersona(a, p)\}$

sacarEst : as a × pos p $\longrightarrow$ as $\{posValida(campus(a), p) \wedge_{\text{L}} hayEst?(a, p) \wedge posIngreso(a, p)\}$

otras operaciones

#hippies : as a $\longrightarrow$ nat

#estudiantes : as a $\longrightarrow$ nat

#masVigilante : as a $\longrightarrow$ nat

haySeg? : as a × pos p × conj(seguridad) *segs* $\longrightarrow$ bool
{posValida(campus(as),p) $\wedge$ $segs \subseteq seguridad(a)$}

posValidaPersona : as a × pos p $\longrightarrow$ bool {posValida(campus(as),p)}

posIngreso : as a × pos p $\longrightarrow$ bool {posValida(campus(as),p)}

moverTodos : as a × conj(seguridad) *segs* $\longrightarrow$ conj(seguridad)

moverSeg : as a × seguridad *seg* × pos *posSig* $\longrightarrow$ seguridad

proxPoss : as a × conj(pos) *minPos* × pos *posAct* $\longrightarrow$ conj(pos)
$\{\neg(emptyset?(minPos)) \wedge_{\text{L}} posValida(campus(a), posAct) \wedge posicionesValidas(campus(a), minPos)\}$

proxPossHippies : as a × conj(pos) $poss$ ⟶ conj(pos) $\{(\forall\ poss\text{:p})\ posValida(a,p) \wedge_L hayHippie?(a,p)\}$

estsCerca : as a × pos p ⟶ conj(pos)

hippieEncerrado? : as a × pos p ⟶ bool

hippieEncerradoEst? : as a × pos p × conj(pos) $poss$ ⟶ bool

hippieEncerradoSeg? : as a × pos p × conj(pos) $poss$ ⟶ bool

hippiesMasCerca : as a × seguridad seg ⟶ conj(pos) $\{seg \in seguridad(a) \wedge hayHippies(a)\}$

encerrado : as a × pos p ⟶ bool $\{posValida(campus(as),p) \wedge hayEst?(p)\}$

#masCapturas : as a × conj(seg) $segs$ ⟶ conj(seg) $\{(\forall\ segs\text{:s})\ \text{s} \in seguridad(a)\}$

#maxCapturas : as a × conj(seg) $segs$ ⟶ nat $\{(\forall\ segs\text{:s}) \in seguridad(a)\}$

captura? : as a × pos p ⟶ bool {posValida(campus(as),p)}

hippiesVecinos : as a × pos p ⟶ nat {posValida(campus(as),p)}

hippiesAlrededor : as a × pos p ⟶ nat {posValida(campus(as),p)}

capturadaHippie : as a × pos p ⟶ nat

capturadaEst : as a × pos p ⟶ nat

validas : as a × conj(pos) p ⟶ conj(pos)

posValidaAS : as a × pos p ⟶ bool

estsCerca : as a × pos p ⟶ conj(pos)

posHippies : as a × conj(pos) $poss$ ⟶ conj(pos) $\{(\forall\ poss\text{:p})\ posValida(p)\}$

posEsts : as a × conj(pos) $poss$ ⟶ conj(pos) $\{(\forall\ poss\text{:p})\ posValida(p)\}$

axiomas

campus(nueva($c, segs$)) ≡ c

campus(moverEst(a, p_1, p_2)) ≡ $campus(a)$

campus(nuevoEst(a, p_1)) ≡ $campus(a)$

campus(nuevoHippie(a, p_1)) ≡ $campus(a)$

campus(sacarEst(a, p_1)) ≡ $campus(a)$

seguridad(nueva($c, segs$)) ≡ $segs$

seguridad(moverEst(a, p_1, p_2)) ≡ $moverTodos(a, seguridad(a))$

seguridad(nuevoEst(a, p_1)) ≡ $moverTodos(a, seguridad(a))$

seguridad(nuevoHippie(a, p_1)) ≡ $seguridad(a)$

seguridad(sacarEst(a, p_1)) ≡ $seguridad(a)$

hayEst?(nueva($c, segs$),p) ≡ $False$

hayEst?(nuevoEst(a, p_1),p) ≡ **if** $p_1 = p$ **then** $True$ **else** $hayEst?(a,p)$ **fi**

hayEst?(moverEst(a, p_1, p_2),p) ≡ **if** $p_1 = p$ **then**
    $False$
  else
    if $p_2 = p$ **then**
      $\neg(hippiesAlrededor(a, p_2) \geq 2)$
    else
      $hayEst?(a,p)$
    fi
  fi

hayEst?(nuevoHippie(a, p_1),p) ≡ $hayEst?(a,p)$

hayEst?(sacarEst(a, p_1),p) ≡ **if** $p_1 = p$ **then** $False$ **else** $hayEst?(a,p)$ **fi** hayHippie?(nueva($c, segs$),p) $\equiv$ $False$

hayHippie?((nuevoHippie(a, p_1),p) $\equiv$ **if** $p_1 = p$ **then** $True$ **else** $hayHippie?(a, p)$ **fi**

hayHippie?((moverEst(a, p_0, p_1),p) $\equiv$ **if** $hayHippie?(a, p)$ **then**
  if $\neg(hippieEncerrado?(a, p))$ **then**
    $p \in proxPossHippies(a, possHippies(a))$
  else
    $False$
  fi
else
  if $hayEst?(a, p)$ **then**
    $(hippiesAlrededor(a, (a, p) \geq 2)$
  else
    $p \in proxPossHippies(a, possHippies(a))$
  fi
fi

hayHippie?(nuevoEst(a, p_1),p) $\equiv$ $hayHippie?(a, p)$

hayHippie?(sacarEst(a, p_1),p) $\equiv$ $hayHippie?(a, p)$

#capturas(nueva($a, segs$),s) $\equiv$ 0

#capturas(nuevoHippie(a, p_1),s) $\equiv$ **if** $(adyacente(a, p_1, posSeg(a, s)))$ **then**
  $capturadoHippie(a, p_1) + \#capturas(a, s)$
else
  $\#capturas(a, s)$
fi

#capturas(nuevoEst(a, p_1),s) $\equiv$ $\#capturas(a, s)$

#capturas(sacarEst(a, p_1),s) $\equiv$ $\#capturas(a, s)$

#capturas(moverEst(a, p_1, p_2),s) $\equiv$ $capturadoHippie(a, < \pi_1(posSeg) + 1, \pi_2(posSeg) >)+$
$capturadoHippie(a, < \pi_1(posSeg) - 1, \pi_2(posSeg) >)+$
$capturadoHippie(a, < \pi_1(posSeg), \pi_2(posSeg) + 1 >)+$
$capturadoHippie(a, < \pi_1(posSeg), \pi_2(posSeg) - 1 >)+$
$\#capturas(a, s)$

#sanciones(nueva($a, segs$),s) $\equiv$ 0

#sanciones(nuevoHippie(a, p_1),s) $\equiv$ $\#sanciones(a, s)$

#sanciones(nuevoEst(a, p_1),s) $\equiv$ $\#sanciones(a, s)$

#sanciones(sacarEst(a, p_1),s) $\equiv$ $\#sanciones(a, s)$

#sanciones(moverEst(a, p_1, p_2),s) $\equiv$ $capturadoEst(a, < \pi_1(posSeg) + 1, \pi_2(posSeg) >)+$
$capturadoEst(a, < \pi_1(posSeg) - 1, \pi_2(posSeg) >)+$
$capturadoEst(a, < \pi_1(posSeg), \pi_2(posSeg) + 1 >)+$
$capturadoEst(a, < \pi_1(posSeg), \pi_2(posSeg) - 1 >)+$
$\#sanciones(a, s)$

#hippies(a) $\equiv$ $\#(posHippies(a, conjPos(campus(a))))$

#estudiantes(a) $\equiv$ $\#(posEsts(a, conjPos(campus(a))))$

masVigilante(a) $\equiv$ $dameUno(masCapturas(a, seguridad(a)))$

masCapturas(a,segs) $\equiv$ **if** $\neg(\emptyset?(segs))$ **then**
  if $\#capturas(a, dameUno(segs)) \geq maxCapturas(a, segs)$
  then
    $ag(masCapturas(a, sinUno(segs)), dameUno(segs))$
  else
    $masCapturas(a, sinUno(segs))$
  fi
else
  $\emptyset$
fi

maxCapturas(a,segs) $\equiv$ **if** $\emptyset?(segs)$ **then**
  0
else
  if $\#capturas(a, dameUno(segs)) \geq maxCapturas(a, sinUno(segs))$
  then
    $\#capturas(a, dameUno(segs))$
  else
    $maxCapturas(a, sinUno(segs))$
  fi
fi

moverTodos(a,segs) $\equiv$ **if** $(\emptyset?(segs))$ **then**
  $\emptyset$
else
  if $(hayHippies?(a))$ **then**
    $Ag(moverTodos(a, sinUno(segs)),$
    $moverSeg(a, dameUno(segs),$
    $dameUno(proxPoss(hippiesMasCerca(a, dameUno(segs))))))$
  else
    $moverIngreso(a, segs)$
  fi
fi

proxPoss(entCerca, p) $\equiv$ **if** $\emptyset?(entCerca)$ **then**

$\emptyset$

else

if $\pi_1(dameUno(entCerca)) > \pi_1(p)$ **then**

if $\pi_2(dameUno(entCerca)) > \pi_2(pos)$ **then**

if $\emptyset?(validas(a, \{<\pi_1(pos)+1, \pi_2(p)><\pi_1(p), \pi_2(p)+1>\}))$

then

$proxPoss(sinUno(entCerca), p)$

else

$Ag(proxPoss(sinUno(entCerca), p), dameUno(validas$

$(a, \{<\pi_1(p)+1, \pi_2(p)>, <\pi_1(p), \pi_2(p)+1>\})))$

fi

else

if $\pi_2(dameUno(entCerca)) < \pi_2(p)$ **then**

if $\emptyset?(validas(a, \{<\pi_1(p)+1, \pi_2(p)><\pi_1(p), \pi_2(p)-1>\}))$

then

$proxPoss(sinUno(entCerca), p)$

else

$Ag(proxPoss(sinUno(entCerca), p), dameUno(validas$

$(a, \{<\pi_1(p)+1, \pi_2(p)><\pi_1(p), \pi_2(p)-1>\})))$

fi

else

if $\emptyset?(validas(a, \{<\pi_1(p)+1, \pi_2(p)>\}))$ **then**

$proxPoss(sinUno(entCerca), p)$

else

$Ag(proxPoss(sinUno(entCerca), p), dameUno(validas$

$(a, \{<\pi_1(p)+1, \pi_2(p)>\})))$

fi

fi

fi

else

if $\pi_1(dameUno(hscerca)) < \pi_1(p)$ **then**

if $\pi_2(dameUno(hscerca)) > \pi_2(p)$ **then**

if $\emptyset?(validas(a,\{<\pi_1(p)-1, \pi_2(p)><\pi_1(p), \pi_2(p)+1>\}))$

then

$proxPoss(sinUno(entCerca), p)$

else

$Ag(proxPoss(sinUno(entCerca), p), dameUno(validas$

$(a,\{<\pi_1(p)-1, \pi_2(p)>, <\pi_1(p), \pi_2(p)+1>\})))$

fi

else

if $\emptyset?(validas(a, \{<\pi_1(p)-1, \pi_2(p)><\pi_1(p), \pi_2(p)-1>\}))$

then

$proxPoss(sinUno(entCerca), p)$

else

$Ag(proxPoss(sinUno(entCerca), p), dameUno(validas$

$(a,\{<\pi_1(p)-1, \pi_2(p)><\pi_1(p), \pi_2(p)-1>\})))$

fi

fi

else

if $\pi_2(dameUno(hscerca)) > \pi_2(p)$ **then**

if $\emptyset?(validas(a, \{<\pi_1(p), \pi_2(p)+1>\}))$ **then**

$proxPoss(sinUno(entCerca), p)$

else

$Ag(proxPoss(sinUno(entCerca), p),$

$dameUno(validas(a,\{<\pi_1(p), \pi_2(p)+1>\})))$

fi

else

if $\emptyset?(validas(a, \{<\pi_1(p), \pi_2(p)-1>\}))$ **then**

$proxPoss(sinUno(entCerca), p)$

else

$Ag(proxPoss(sinUno(entCerca), p),$

$dameUno(validas(a,\{<\pi_1(p), \pi_2(p)-1>\})))$

fi

fi

fi

fi

moverIngreso(a,segs) $\equiv$ **if** $\emptyset?(segs)$ **then**
$\emptyset$
else
if $(alto(campus(a)){-}1) - \pi_2(dameUno(segs)) > \pi_2(dameUno(segs))$
then
$ag(moverIngreso(a, sinUno(segs)), mover(dameUno(segs), <$
$(\pi_1(dameUno(segs)), \pi_2(segs) - 1) >))$
else
if $(alto(campus(a)) - 1) - \pi_2(dameUno(segs)) < \pi_2(dameUno(segs)$
then
$ag(moverIngreso(a, sinUno(segs)), mover(dameUno(segs), <$
$(\pi_1(dameUno(segs)), \pi_2(segs) + 1) >))$
else
$ag(moverIngreso(a, sinUno(segs)), mover(dameUno(segs),$
$dameUno(\{< (\pi_1(dameUno(segs)), \pi_2(segs) - 1) >, <$
$(\pi_1(dameUno(segs)), \pi_2(segs) + 1) >\}))$
fi
fi
fi

posValidaAS(a,p) $\equiv$ $posValida(dameUno(poss)) \wedge$
$\neg(hayHippie?(a, dameUno(poss))) \wedge$
$\neg(hayEst?(a, dameUno(poss))) \wedge$
$\neg(haySeg?(a, dameUno(poss), seguridad(a))$

validas(a,poss) $\equiv$ **if** $\emptyset?(poss)$ **then**
$\emptyset$
else
if $posValidaAS(a, dameUno(poss))$ **then**
$Ag(validas(a, sinUno(poss)), dameUno(poss))$
else
$validas(a, sinUno(poss))$
fi
fi

hippieEncerrado?(a,p) $\equiv$ $hipEncerradoEst?(a, p, adyacentes(campus(a), p)) \wedge$
$hipEncerradoSeg?(a, p, adyacentes(campus(a), p))$

hipEncerradoEst?(a,p,adys) $\equiv$ **if** $\emptyset?(adys)$ **then**
$True$
else
if $posValida?(campus(a), dameUno(adys))$ **then**
$hayEst?(a, p) \wedge hipEncerradoEst?(a, p, sinUno(adys))$
else
$False$
fi
fi

hipEncerradoSeg?(a,p,adys) $\equiv$ **if** $\emptyset?(adys)$ **then**
$True$
else
if $posValida?(campus(a), dameUno(adys))$ **then**
$haySeg?(a, p, seguridad(a)) \wedge$
$hipEncerradoSeg?(a, p, sinUno(adys))$
else
$False$
fi
fi

moverSeg(a,seg,nPos) $\equiv$ **if** $(distMan(campus(a), \pi_2(seg), nPos) \geq 2$
$\vee \neg(posValidaAS(a, nPos)))$ **then**
  seg
else
  if $\#sanciones(a, seg) < 3$ **then**
    $< \pi_1(seg), nPos >$
  else
    seg
  fi
fi

haySeg?(a,p,segs) $\equiv$ **if** $\emptyset?(segs)$ **then**
  $False$
else
  if $\pi_2(dameUno(segs)) == p$ **then**
    $True$
  else
    $haySeg?(a, p, sinUno(segs))$
  fi
fi

proxPossHippies(a,possHippies) $\equiv$ **if** $\emptyset?(possHippies)$ **then**
  $\emptyset$
else
  $proxPoss(a, estsCerca(dameUno(possHippies), dameUno(possHippie$
  $\cup proxPossHippies(a, sinUno(possHippies))$
fi

hippiesMasCerca(a,seg) $\equiv$ $minDistsPos(campus(a), \pi_2(seg), posHippies(a, conjPos(campus(a))))$

estsCerca(a,p) $\equiv$ $minDistsPos(campus(a), p, posEsts(a, conjPos(campus(a))))$

posHippies(a,conjpos) $\equiv$ **if** $\emptyset?(conjpos)$ **then**
  $\emptyset$
else
  if $hayHippie?(a, dameUno(conjpos))$ **then**
    $Ag(posHippies(a, sinUno(conjpos)), dameUno(conjpos))$
  else
    $posHippies(a, sinUno(conjpos))$
  fi
fi

posEsts(a,conjpos) $\equiv$ **if** $\emptyset?(conjpos)$ **then**
  $\emptyset$
else
  if $hayEst?(a, dameUno(conjpos))$ **then**
    $Ag(posEsts(a, sinUno(conjpos)), dameUno(conjpos))$
  else
    $posEsts(a, sinUno(conjpos))$
  fi
fi

captura?(a, p) $\equiv$ **if** $(posValida(campus(a), <\pi_1(p)+1, \pi_2(p)>)$ **then**
$(hayObstaculo?(campus(a), <\pi_1(p)+1, \pi_2(p)>) \vee haySeg?(a, <\pi_1(p)+1, \pi_2(p)>, seguridad(a)))$
else
$\neg(hayEst?(a, <\pi_1(p), \pi_2(p)>))$
fi
$\wedge$
if $(posValida(campus(a), <\pi_1(p)-1, \pi_2(p)>)$ **then**
$(hayObstaculo?(campus(a), <\pi_1(p)-1, \pi_2(p)>) \vee haySeg?(a, <\pi_1(p)-1, \pi_2(p)>, seguridad(a)))$
else
$\neg(hayEst?(a, <\pi_1(p), \pi_2(p)>))$
fi
$\wedge$
if $(posValida(campus(a), <\pi_1(p), \pi_2(p)+1>)$ **then**
$(hayObstaculo?(campus(a), <\pi_1(p), \pi_2(p)+1>) \vee haySeg?(a, <\pi_1(p), \pi_2(p)+1>, seguridad(a)))$
else
$True$
fi
$\wedge$
if $(posValida(campus(a), <\pi_1(p), \pi_2(p)-1>)$ **then**
$(hayObstaculo?(campus(a), <\pi_1(p), \pi_2(p)-1>) \vee haySeg?(a, <\pi_1(p), \pi_2(p)-1>, seguridad(a)))$
else
$True$
fi

capturadoHippie$(a, p))$ $\equiv$ **if** $(PosValida(campus(a), p))$ **then**
if $(hayHippie(a, p))$ **then**
if $(captura?(a, p))$ **then** 1 **else** 0 **fi**
else
0
fi
else
0
fi

capturadoEst$(a, p))$ $\equiv$ **if** $(PosValida(campus(a), p))$ **then**
if $(hayEst(a, p))$ **then**
if $(captura?(a, p))$ **then** 1 **else** 0 **fi**
else
0
fi
else
0
fi

hippiesVecinos(a, p) $\equiv$ **if** hayHippie?(a,p) **then** 1 **else** 0 **fi**

hippiesAlrededor(a, p) $\equiv$ **if** posValida (campus(a),$<\pi_1(p)+1, \pi_2(p)>$) **then**
hippiesVecinos(a,$<\pi_1(p)+1, \pi_2(p)>$)
else
0
fi+**if** posValida (campus(a),$<\pi_1(p)-1, \pi_2(p)>$) **then**
hippiesVecinos(a,$<\pi_1(p)-1, \pi_2(p)>$)
else
0
fi+**if** posValida (campus(a),$<\pi_1(p), \pi_2(p)+1>$) **then**
hippiesVecinos(a,$<\pi_1(p), \pi_2(p)+1>$)
else
0
fi+**if** posValida (campus(a),$<\pi_1(p), \pi_2(p)-1>$) **then**
hippiesVecinos(a,$<\pi_1(p), \pi_2(p)-1>$)
else
0
fi

posValidaPersona(a, p) $\equiv$ **if** $\neg(\mathit{hayObstaculo?}(\mathit{campus}(a), p))$ **then**
($\pi_2(p) = 0 \vee \pi_2$=alto(campus(a)))
else
$False$
fi

Fin TAD

2. TAD CAMPUS

TAD CAMPUS

géneros campus

usa BOOL,NAT,TUPLA

exporta CAMPUS, observadores, generadores, posValida, posIngreso,minDistPos,adyacente,

observadores básicos

alto : campus $\longrightarrow$ nat

ancho : campus $\longrightarrow$ nat

obstaculos : campus $\longrightarrow$ conj(pos)

generadores

nuevo : nat *ancho* $\times$ nat *alto* $\times$ conj(pos) *obst* $\longrightarrow$ campus
$\{1 \leq \mathit{ancho} \wedge 1 \leq \mathit{alto} \wedge (\forall\ p\text{:pos})\ p \in \mathit{obst} \Rightarrow_{\text{L}} posValida(c, p)\}$

otras operaciones

adyacente : campus c $\times$ pos pe $\times$ pos pd $\longrightarrow$ bool $\{posValida(c, pe) \wedge posValida(c, pd)\}$

posValida : campus c $\times$ pos p $\longrightarrow$ bool

posIngreso : campus c $\times$ pos p $\longrightarrow$ bool {posValida(c,p)}

minDistsPos : campus c $\times$ pos p $\times$ conj(pos) *posiciones* $\longrightarrow$ conj(pos)
$\{posValida(c, p) \wedge \neg(\emptyset?(posiciones))\}$

minDist : campus c $\times$ pos p $\times$ conj(posiciones) *posiciones* $\longrightarrow$ nat
$\{posValida(c, p) \wedge \neg(\emptyset?(posiciones))\}$

distMan : campus c $\times$ pos $p1$ $\times$ pos $p2$ $\longrightarrow$ nat $\{posValida(c, p1) \wedge posValida(c, p2)\}$

restaAbs : nat $\times$ nat $\longrightarrow$ nat conjPos : campus × nat × nat ⟶ conj(pos)

adyacentes : campus × pos ⟶ conj(pos)

hayObstaculo? : campus c × pos p ⟶ bool {posValida(c,p)}

axiomas ∀ *alto*:nat, ∀ *ancho*:nat, ∀ *obst*:conj (pos)
∀ p_1:pos ∀ p_2:pos

alto(nuevo(*ancho*,*alto*,*obst*)) ≡ *alto*

ancho(nuevo(*ancho*,*alto*,*obst*)) ≡ *ancho*

obstaculos(nuevo(*ancho*,*alto*,*obst*)) ≡ *obst*

posValida(c,p) ≡ $\pi_1(p) < ancho \land \pi_2(p) < alto \land \neg(hayObstaculo?(a,p))$

adyacente(c,p_1,p_2) ≡ $(\pi_1(p_1) = \pi_1(p_2) - 1 \lor \pi_1(p_1) = \pi_1(p_2) + 1) \lor (\pi_2(p_1) = \pi_2(p_2) - 1 \lor \pi_2(p_1) = \pi_2(p_2) + 1)$

minDistsPos(c,p,posiciones) ≡ **if** $\emptyset?(sinUno(posiciones))$ **then**
  $dameUno(posiciones)$
else
  if $distMan(c,p,dameUno(posiciones)) \leq minDist(c,p,posiciones)$ **then**
    $Ag(minDistsPos(c,sinUno(posiciones)), dameUno(posiciones))$
  else
    $minDistsPos(c,seg,sinUno(posiciones))$
  fi
fi

minDist(c,p,posiciones) ≡ **if** $\emptyset?(sinUno(posiciones))$ **then**
  $distMan(c,p,dameUno(posiciones))$
else
  if $distMan(c,p,dameUno(posiciones)) \leq minDist(c,pos/p,sinUno(posiciones))$
  then
    $distMan(c,p,dameUno(posiciones))$
  else
    $minDist(c,p,sinUno(posiciones))$
  fi
fi

distMan(c,p_1,p_2) ≡ $restaAbs(\pi_2(p_1),\pi_2(p_2)) + restaAbs(\pi_1(p_1),\pi_1(p_2))$

restaAbs(n1,n2) ≡ **if** $n2 > n1$ **then** $n2 - n1$ **else** $n1 - n2$ **fi**

conjPos(c,x,y) ≡ **if** $x \geq ancho(c)$ **then**
  $\emptyset$
else
  if $y \geq alto(c)$ **then**
    $conjPos(c,x+1,0)$
  else
    $ag(conjPos(c,x,y+1), <x,y>)$
  fi
fi

adyacentes(c,p) ≡ $\{<\pi_1(p)+1,\pi_2(p)+1><\pi_1(p)-1,\pi_2(p)-1><\pi_1(p)+1,\pi_2(p)><\pi_1(p),\pi_2(p)+1>\}$

hayObstaculo?(c,p) ≡ $p \in obstaculos(c)$

Fin TAD

3. TAD Pos es Tupla(Nat,Nat)

4. TAD Seguridad es Tupla(Id, Pos)

5. TAD Id es Nat

6. Consideraciones

1. Como no se indica que los hippies y estudiantes deben estar identificados, consideramos que dos instancias con un hippie|estudiante en la misma posición, son iguales
2. El movimiento de los estudiantes activa el comportamiento automático de los hippies y seguridad
3. El movimiento de los hippies y seguridad siempre se realiza hacia su destino final, en caso de quedar atascados, no vuelven a buscar otro camino, se mantienen en el mismo lugar hasta que, eventualmente, su objetivo se sitúe en una posición alcanzable
4. Para evitar el conflicto que se produce al capturar un estudiante que pertenece a los estudiantes que capturan a un hippie y casos similares, las capturas se realizan al intentar mover a la entidad capturada