Trabajo Practico I, Alta Seguridad nos cuida

Algoritmos y Estructuras de Datos II, DC, UBA.

Índice

TAD ID ES NAT
TAD PLACA ES NAT
TAD EST ES SECU(DIR)
TAD DIR ES Enum{ n, s, e, o }

1. TAD UNIVERSIDAD

TAD UNIVERSIDAD

géneros uni

exporta uni, Generadores, Observadores Basicos, masVigilante, cuantosHippies, cuantosEstudiantes

usa NAT, CONJ(α), BOOL, TUPLA($\alpha_1, \ldots, \alpha_n$), MULTICONJ($\alpha$), ID, PLACA, EST, CAMPUS, DIR, DICCIONARIO(CLAVE, SIGNIFICADO)

igualdad observacional

$$(\forall u, u' : \text{uni}) \left(u =_{\text{obs}} u' \iff \begin{pmatrix} \text{campus?}(u) =_{\text{obs}} \text{campus?}(u') \wedge \\ \text{estudiantes?}(u) =_{\text{obs}} \text{estudiantes?}(u') \wedge \\ \text{hippies?}(u) =_{\text{obs}} \text{hippies?}(u') \wedge \\ \text{agentes?}(u) =_{\text{obs}} \text{agentes?}(u') \wedge_{\text{L}} \\ (\forall p : \text{placa})\ \text{def?}(p, \text{agentes?}(u)) \Rightarrow_{\text{L}} \\ (\text{sanciones?}(u, p) =_{\text{obs}} \text{sanciones?}(u', p) \wedge \\ \text{capturas?}(u, p) =_{\text{obs}} \text{capturas?}(u', p)) \end{pmatrix} \right)$$

Los Estudiantes son representados como una secuencia de direcciones en las cuales moverse. Si en algun momento su camino esta bloqueado, no se mueven, ya que consideramos que lo interesante se encuentra en el camino que debe recorrer.

observadores básicos

campus? : uni $\longrightarrow$ campus

agentes? : uni $\longrightarrow$ dicc(placa, pos)

hippies? : uni $\longrightarrow$ dicc(id, pos)

estudiantes? : uni $\longrightarrow$ dicc(id, $\langle$ est, pos $\rangle$)

sanciones? : placa *pl* $\times$ uni u $\longrightarrow$ nat {def?(*pl*, agentes?(u))}

capturas? : placa *pl* $\times$ uni u $\longrightarrow$ nat {def?(*pl*, agentes?(u))}

generadores

nuevaUni : campus c $\times$ dicc(placa, pos) as $\longrightarrow$ uni {posicionesValidas(c, as)}

agregarE : uni u $\times$ id i $\times$ est e $\times$ pos p $\longrightarrow$ uni {$\neg$ (def?(i, hippies?(u))) $\wedge$ $\neg$ (def?(i, estudiantes?(u))) $\wedge$ posValida?(p, campus?(u)) $\wedge$ entrada?(p, campus?(u)) $\wedge_{\text{L}}$ posicionNoOcupada(p, u)}

agregarH : uni u $\times$ id i $\times$ pos p $\longrightarrow$ uni {$\neg$ (def?(i, hippies?(u))) $\wedge$ $\neg$ (def?(i, estudiantes?(u))) $\wedge$ posValida?(p, campus?(u)) $\wedge$ entrada?(p, campus?(u)) $\wedge_{\text{L}}$ posicionNoOcupada(p, u)}

moverAS : uni u $\times$ placa p $\longrightarrow$ uni { (def?(p, agentes?(u))) $\wedge_{\text{L}}$ (sanciones?(p, u) < 3)}

moverH : uni u $\times$ id i $\longrightarrow$ uni {def?(*i*, hippies?(*u*))}

moverE : uni u $\times$ id i $\longrightarrow$ uni {def?(*i*, estudiantes?(*u*))}

otras operaciones

moverAgente : placa pl $\times$ uni u $\longrightarrow$ dicc(placa, pos) {def?(pl, agentes?(u))}

hayDosHippies? : pos $\times$ dicc(id,pos) $\longrightarrow$ bool

algunHippieEncerradoPorEstudiantes : dicc(id,pos) $\times$ dicc(id,$\langle$est,pos$\rangle$) $\longrightarrow$ bool

hippieEncerradoPorEstudiantes : dicc(id,pos) dH × dicc(id,⟨est,pos⟩) dE ⟶ id {algunHippieEncerradoPorEstudiantes(dH, dE)}

movRestringido? : pos p × campus c × dicc(placa,pos) dAs × dicc(id,pos) dH × dicc(id,⟨est,pos⟩) dE ⟶ bool {posValida?(p, c)}

hayUnAgenteAlrededor? : pos × dicc(placa,pos) ⟶ bool

murioAlgunHippie? : campus × dicc(placa,pos) × dicc(id,pos) × dicc(id,⟨est,pos⟩) ⟶ bool

hippieMuerto : campus c × dicc(placa,pos) dAs × dicc(id,pos) dH × dicc(id,⟨est,pos⟩) dE ⟶ id {murioAlgunHippie?(c, dAs, dH, dE)}

moverEstudiante : id i × uni u ⟶ dicc(id , ⟨ est , pos ⟩) {def?(i, estudiantes?(u))}

borrarTodosLosHippiesMuertos : campus c × dicc(placa,pos) dAs × dicc(id,pos) dH × dicc(id,⟨est,pos⟩) dE ⟶ dicc(id , pos)

borrarTodosLosHippiesEncerradosPorEstudiantes : dicc(id,pos) × dicc(id,⟨est,pos⟩) ⟶ dicc(id , pos)

conviertoAlgunEstudiante? : dicc(id,pos) × dicc(id,⟨est,pos⟩) ⟶ bool

moverHippie : id i × uni u ⟶ dicc(id , pos) {def?(i, hippies?(u))}

agregoEstudiantesConvertidos : dicc(id,pos) dH × dicc(id,⟨est,pos⟩) dE ⟶ dicc(id , pos) {conviertoAlgunEstudiante?(dH, dE)}

estudianteConvertido : dicc(id,pos) dH × dicc(id,⟨est,pos⟩) dE ⟶ id {conviertoAlgunEstudiante?(dH, dE)}

borrarTodosEstudiantesConvertidos : dicc(id,pos) × dicc(id,⟨est,pos⟩) ⟶ dicc(id , ⟨ est , pos ⟩)

convertirTodosHippies : dicc(id,pos) × dicc(id,⟨est,pos⟩) ⟶ dicc(id , pos)

Para la tupla, asumimos que este hippie convertido toma la secuencia de direcciones de algun estudiante del campus.

agentesAlrededor : pos × dicc(placa,pos) ⟶ conj(placa)

algunEstudianteEncerradoPorAgente? : campus × dicc(placa,pos) × dicc(id,pos) × dicc(id,⟨est,pos⟩) ⟶ bool

agentesQueAtrapanEstudiantes : campus c × dicc(placa,pos) dAs × dicc(id,pos) dH × dicc(id,⟨est,pos⟩) dE ⟶ conj(placa) {algunEstudianteEncerradoPorAgente?(c, dAs, dH, dE)}

agentesQueAtrapanHippies : campus × dicc(placa,pos) dAs × dicc(id,pos) × dicc(id,⟨est,pos⟩) ⟶ conj(placa) {¬∅?(claves(dAs))}

En funciones como la de arriba, tuvimos que sacar algunos nombres de parÃ¡metros para que nos entrara en el ancho. De todos modos, para mejorar la claridad, mantuvimos siempre la misma nomenclatura

posibleMovAs : placa × uni ⟶ conj(dir)

dirLibres : pos p × uni u ⟶ conj(dir) {posValida?(p, campus?(u))}

posNoOcupadas : conj(pos) cP × uni u ⟶ conj(pos) {(∀ p: pos) p ∈ cP ⇒ posValida?(p, u)}

dirValidas : pos p × campus c ⟶ conj(pos) {posValida?(p, c)}

posicionesHippies : dicc(id,pos) ⟶ conj(pos)

posicionesAgentes : dicc(placa,pos) ⟶ conj(pos)

posicionesEstudiantes : dicc(id,⟨est,pos⟩) ⟶ conj(pos)

posicionNoOcupada : pos p × uni u ⟶ bool {posValida?(p, campus?(u))}

posVecinasValidas : pos p × campus c ⟶ conj(pos) {posValida(p, c)}

posicionesValidas : conj(pos) cp × campus c ⟶ conj(pos) {(∀ p: pos) p ∈ cP ⇒ posValida?(p, u)}

murioAlgunHippieAux : dicc(id,pos) × campus × dicc(placa,pos) × dicc(id,pos) × dicc(id,⟨est,pos⟩) ⟶ bool

hippieMuertoAux : dicc(id,pos) × campus × dicc(placa,pos) × dicc(id,pos) × dicc(id,⟨est,pos⟩) ⟶ id {murioAlgunHippie?(c, dAs, dH, dE)}

posibleMovHippie : id i × uni u ⟶ conj(dir) {def?(i, hippies?(u))}

aEEPAaux : dicc(id,pos) × campus c × dicc(placa,pos) dAs × dicc(id,pos) dH × dicc(id,⟨est,pos⟩) dE ⟶ bool

aQAEaux : dicc(id,pos) × campus × dicc(placa,pos) × dicc(id,pos) × dicc(id,⟨est,pos⟩) ⟶ conj(placa)

aQAHaux : dicc(id,pos) × campus × dicc(placa,pos) × dicc(id,pos) × dicc(id,⟨est,pos⟩) ⟶ conj(placa)

cuantosHippies? : uni ⟶ nat

cuantosEstudiantes? : uni ⟶ nat

masVigilante? : uni u ⟶ placa {¬∅?(claves(agentes?(u)))}

losMasVigilantes : conj(placa) × uni × nat ⟶ conj(placa)

maxCapturas : conj(placa) × uni ⟶ nat

menorPlaca : conj(placa) cPl ⟶ nat {¬∅?(cPl)}

saleDeCampus : dir d × pos p × campus c ⟶ bool {posValida?(p, c)}

entradaSuperior : pos p ⟶ bool

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

axiomas ∀ c: campus ∀ dAs: dicc(placa,pos) ∀ dH: dicc(id, pos) ∀ dE: dicc(id, <est, pos>) ∀ $cDirs$: conj(dir) ∀ mc: nat ∀ $cPos, cP$: conj(pos) ∀ u: uni ∀ i: id ∀ pl: placa ∀ p: pos

Observadores Basicos

campus? (nuevaUni(c, dAs)) ≡
c

campus? (agregarE(u, i, e, p)) ≡
campus?(u)

campus? (agregarH(u, i, p)) ≡
campus?(u)

campus? (moverAs(u, p)) ≡
campus?(u)

campus? (moverH(u, i)) ≡
campus?(u)

campus? (moverE(u, i)) ≡
campus?(u)

agentes? (nuevaUni(c, dAs)) ≡
dAs

agentes? (agregarE(u, i, e, p)) ≡
agentes?(u)

agentes? (agregarH(u, i, pos)) ≡
agentes?(u)

agentes? (moverAs(u, p)) ≡
moverAgente(p, u)

agentes? (moverE(u, i)) ≡
agentes?(u)

agentes? (moverH(u, i)) ≡
agentes?(u)

hippies? (nuevaUni(c, dAs)) $\equiv$
vacio

hippies? (agregarE(u, i, e, p)) $\equiv$
if hayDosHippies?(p, hippies?(u)) **then**
  definir(i, p, hippies?(u))
else
  if algunHippieEncerradoPorEstudiantes(hippies?(u), definir(i, $\langle e, p \rangle$, estudiantes?(u))) **then**
    borrar(hippieEncerradoPorEstudiantes(hippies?(u), definir(i, $\langle e, p \rangle$, estudiantes?(u))), hippies?(u))
  else
    hippies?(u)
  fi
fi

hippies? (agregarH(u, i, p)) $\equiv$
if murioAlgunHippie(campus?(u), agentes?(u), definir(i, p, hippies?(u)), estudiantes?(u)) **then**
  borrarTodosHippiesMuertos(campus?(u), agentes?(u), definir(i, p, hippies?(u)), estudiantes?(u))
else
  definir(i, p, hippies?(u))
fi

hippies? (moverAs(u, i)) $\equiv$
if murioAlgunHippie(campus?(u), moverAgente(i, u), hippies?(u), estudiantes?(u)) **then**
  borrarTodosHippiesMuertos(campus?(u), moverAgente(i, u), hippies?(u), estudiantes?(u))
else
  hippies?(u)
fi

hippies? (moverE(u, i)) $\equiv$
if murioAlgunHippie(campus?(u), agentes?(u), hippies?(u), moverEstudiante(i, u)) **then**
  if hayDosHippies?(obtener(i, moverEstudiante(i, u)), borrarTodosHippiesMuertos(campus?(u), agentes?(u), hippies?(u), moverEstudiante(i, u))) **then**
    definir(i, obtener(i, moverEstudiante(i, u)), borrarTodosHippiesMuertos(campus?(u), agentes?(u), hippies?(u), moverEstudiante(i, u)))
  else
    if algunHippieEncerradoPorEstudiantes(borrarTodosHippiesMuertos(campus?(u), agentes?(u), hippies?(u), moverEstudiante(i, u)), moverEstudiante(i, u)) **then**
      borrarTodosHippiesEncerradoPorEstudiantes(borrarTodosHippiesMuertos(campus?(u), agentes?(u), hippies?(u), moverEstudiante(i, u)), moverEstudiante(i, u))
    else
      borrarTodosHippiesMuertos(campus?(u), agentes?(u), hippies?(u), moverEstudiante(i, u))
    fi
  fi
else
  if hayDosHippies?(obtener(i, moverEstudiante(i, u)), hippies?(u)) **then**
    definir(i, obtener(i, moverEstudiante(i, u)), hippies?(u))
  else
    if algunHippieEncerradoPorEstudiantes(hippies?(u), moverEstudiante(i, u)) **then**
      borrarTodosHippiesEncerradoPorEstudiantes(hippies?(u), moverEstudiante(i, u))
    else
      hippies?(u)
    fi
  fi
fi

hippies? (moverH(u, i)) $\equiv$

if murioAlgunHippie(campus?(u), agentes?(u), moverHippie(i, u), estudiantes?(u)) **then**
  if conviertoAlgunEstudiante?(borrarTodosHippiesMuertos(campus?(u), agentes?(u), moverHippie(i, u), estudiantes?(u)), estudiantes?(u)) **then**
    borrarTodosHippiesMuertos(campus?(u), agentes?(u), agregoEstudiantesConvertidos(borrarTodosHippiesMuertos(campus?(u), agentes?(u), moverHippie(i, u), estudiantes?(u)), estudiantes?(u))
  else
    borrarTodosHippiesMuertos(campus?(u), agentes?(u), moverHippie(i, u), estudiantes?(u))
  fi
else
  if conviertoAlgunEstudiante?(moverHippie(i, u), estudiantes?(u)) **then**
    borrarTodosHippiesMuertos(campus?(u), agentes?(u), agregoEstudiantesConvertidos(moverHippie(i, u), estudiantes?(u), estudiantes?(u))
  else
    hippies?(u)
  fi
fi

estudiantes? (nuevaUni(c, dAs)) $\equiv$
vacio

estudiantes? (agregarE(u, i, e, p)) $\equiv$
if hayDosHippies?(p, hippies?(u)) **then** estudiantes?(u) **else** definir(i, $\langle$ e, p $\rangle$, estudiantes?(u)) **fi**

estudiantes? (agregarH(u, i, p)) $\equiv$
if conviertoAlgunEstudiante?(definir(i, p, hippies?(u)), estudiantes?(u)) **then**
  borrarTodosEstudiantesConvertidos(definir(i, p, hippies?(u)), estudiantes?(u))
else
  estudiantes?(u)
fi

estudiantes? (moverAs(u, i)) $\equiv$
estudiantes?(u)

estudiantes? (moverE(u, i)) $\equiv$
if murioAlgunHippie(campus?(u), agentes?(u), hippies?(u), moverEstudiante(i, u)) **then**
  if hayDosHippies?(obtener(i, moverEstudiante(i, u)), borrarTodosHippiesMuertos(campus?(u), agentes?(u), hippies?(u), moverEstudiante(i, u))) **then**
    borrar(i, estudiantes?(u))
  else
    if algunHippieEncerradoPorEstudiantes(borrarTodosHippiesMuertos(campus?(u), agentes?(u), hippies?(u), moverEstudiante(i, u)), moverEstudiante(i, u)) **then**
      convertirTodosHippies(borrarTodosHippiesMuertos(campus?(u), agentes?(u), hippies?(u), moverEstudiante(i, u)), moverEstudiante(i, u))
    else
      estudiantes?(u)
    fi
  fi
else
  if hayDosHippies?(obtener(i, moverEstudiante(i, u)), hippies?(u)) **then**
    borrar(i, estudiantes?(u))
  else
    if algunHippieEncerradoPorEstudiantes(hippies?(u), moverEstudiante(i, u)) **then**
      convertirTodosHippies(hippies?(u), moverEstudiante(i, u))
    else
      estudiantes?(u)
    fi
  fi
fi

estudiantes? (moverH(u, i)) $\equiv$
if conviertoAlgunEstudiante?(moverHippie(i, u), estudiantes?(u)) **then**
  borrarTodosEstudiantesConvertidos(moverHippie(i, u), estudiantes?(u))
else
  estudiantes?(u)
fi

sanciones? (pl, nuevaUni(c, dAs)) $\equiv$
0

sanciones? (pl, agregarE(u, i, e, p)) $\equiv$
if hayUnAgenteAlrededor?(p, agentes?(u)) $\wedge$
movRestringido?(p, campus?(u), agentes?(u), hippies?(u), definir(i, $\langle e, p \rangle$, estudiantes?(u))) **then**
  if $pl \in$ agentesAlrededor(p, agentes?(u)) **then** 1+ sanciones?(pl, u) **else** sanciones?(pl, u) **fi**
else
  sanciones?(pl, u)
fi

sanciones? (pl, agregarH(u, i, p)) $\equiv$
if algunEstudianteEncerradoPorAgentes?(campus?(u), agentes?(u), definir(i, p, hippies?(u)), estudiantes?(u))
then
  if $pl \in$ agentesQueAtrapanEstudiantes(campus?(u), agentes?(u), definir(i, p, hippies?(u)), estudiantes?(u))
  then
    1+ sanciones?(pl, u)
  else
    sanciones?(pl, u)
  fi
else
  sanciones?(pl, u)
fi

sanciones? (pl, moverAs(u, p)) $\equiv$
if algunEstudianteEncerradoPorAgentes?(campus?(u), moverAgente(pl, u), hippies?(u), estudiantes?(u)) **then**
  if $pl \in$ agentesQueAtrapanEstudiantes(campus?(u), moverAgente(pl, u), hippies?(u), estudiantes?(u)) **then**
    1+ sanciones?(pl, u)
  else
    sanciones?(pl, u)
  fi
else
  sanciones?(pl, u)
fi

sanciones? (pl, moverH(u, i)) $\equiv$
if algunEstudianteEncerradoPorAgentes?(campus?(u), agentes?(u), moverHippie(i, u), estudiantes?(u)) **then**
  if $pl \in$ agentesQueAtrapanEstudiantes(campus?(u), agentes?(u), moverHippie(i, u), estudiantes?(u)) **then**
    1+ sanciones?(pl, u)
  else
    sanciones?(pl, u)
  fi
else
  sanciones?(pl, u)
fi

sanciones? (pl, moverE(u, i)) $\equiv$

if algunEstudianteEncerradoPorAgentes?(campus?(u), agentes?(u), hippies?(u), moverEstudiante(i,u)) **then**
  if $pl \in$ agentesQueAtrapanEstudiantes(campus?(u), agentes?(u), hippies?(u), moverEstudiante(i,u)) **then**
    1+ sanciones?(pl,u)
  else
    sanciones?(pl,u)
  fi
else
  sanciones?(pl,u)
fi

capturas? $(pl,$ nuevaUni$(c,dAs))$ $\equiv$
0

capturas? $(pl,$ agregarE$(u,i,e,p))$ $\equiv$
if murioAlgunHippie?(campus?(u), agentes?(u), hippies?(u), definir$(i,\ \langle e,p\rangle,$ estudiantes?$(u))$) **then**
  if $pl \in$ agentesQueAtrapanHippies(campus?(u), agentes?(u), hippies?(u), definir$(i,\ \langle e,p\rangle,$ estudiantes?$(u))$) **then**
    1+ capturas?(pl,u)
  else
    capturas?(pl,u)
  fi
else
  capturas?(pl,u)
fi

capturas? $(pl,$ agregarH$(u,i,p))$ $\equiv$
if murioAlgunHippie?(campus?(u), agentes?(u), definir$(i,p,$ hippies?$(u))$, estudiantes?(u)) **then**
  if $pl \in$ agentesQueAtrapanHippies(campus?(u), agentes?(u), definir$(i,p,$ hippies?$(u))$, estudiantes?(u)) **then**
    1+ capturas?(pl,u)
  else
    capturas?(pl,u)
  fi
else
  capturas?(pl,u)
fi

capturas? $(pl,$ moverAs$(u,p))$ $\equiv$
if murioAlgunHippie?(campus?(u), moverAgente(pl,u), hippies?(u), estudiantes?(u)) **then**
  if $pl \in$ agentesQueAtrapanHippies(campus?(u), moverAgente(pl,u), hippies?(u), estudiantes?(u)) **then**
    1+ capturas?(pl,u)
  else
    capturas?(pl,u)
  fi
else
  capturas?(pl,u)
fi

capturas? $(pl,$ moverH$(u,i))$ $\equiv$
if murioAlgunHippie?(campus?(u), agentes?(u), moverHippie(i,u), estudiantes?(u)) **then**
  if $pl \in$ agentesQueAtrapanHippies(campus?(u), agentes?(u), moverHippie(i,u), estudiantes?(u)) **then**
    1+ capturas?(pl,u)
  else
    capturas?(pl,u)
  fi
else
  capturas?(pl,u)
fi

capturas? $(pl,$ moverE$(u,i))$ $\equiv$

if murioAlgunHippie?(campus?(u), agentes?(u), hippies?(u), moverEstudiante(i, u)) **then**
  if $pl \in$ agentesQueAtrapanHippies(campus?(u), agentes?(u), hippies?(u), moverEstudiante(i, u)) **then**
    1+ capturas?(pl, u)
  else
    capturas?(pl, u)
  fi
else
  capturas?(pl, u)
fi

moverAgente(pl, u) $\equiv$
if ($\emptyset$?(claves(hippies?(u))) $\wedge$ entrada?(obtener(pl, agentes?(u)), campus?(u)) $\vee$ sanciones?(pl, u) ≥ 3 **then**
  agentes?(u)
else
  if $\neg$ $\emptyset$?(posibleMovAs(pl, u)) **then**
    definir(pl, mover(obtener(pl, agentes?(u)), dameUno(posibleMovAs(pl, u))), agentes?(u))
  else
    agentes?(u)
  fi
fi

moverHippie(i, u) $\equiv$
if $\emptyset$?(claves(estudiantes?(u))) **then**
  hippies?(u)
else
  if $\neg$ $\emptyset$?(posibleMovHippie(i, u)) **then**
    definir(i, mover(obtener(i, hippies?(u)), dameUno(posibleMovHippie(i, u))), hippies?(u))
  else
    hippies?(u)
  fi
fi

moverEstudiante(i, u) $\equiv$
if saleDeCampus(prim(Π_1(obtener(i, estudiantes?(u)))), Π_2(obtener(i, estudiantes?(u))), campus?(u)) **then**
  borrar(i, estudiantes?(u))
else
  if prim(Π_1(obtener(i, estudiantes?(u)))) $\in$ dirLibres(Π_2(obtener(i, estudiantes?(u))) , u) **then**
    definir(i, $\langle$ fin(Π_1(obtener(i, estudiantes?(u)))), mover(Π_2(obtener(i, estudiantes?(u))), prim(Π_1(obtener(i, estudiantes?(u))))) $\rangle$, estudiantes?(u))
  else
    estudiantes?(u)
  fi
fi

posibleMovAs (pl, u) $\equiv$

if (sanciones?$(pl, u) \geq 3$) **then**
  $\emptyset$
else
  if ($\emptyset$?(claves(hippies?(u)))) **then**
    if $\emptyset$?(dirLibres(obtener(pl, u), u) $\cap$ direccionesOptimas(dameUno(cPosMasCercanaShort(obtener(pl, u), posNoOcupadas(entradas(campus?(u)), u))))) **then**
      dirLibres(obtener(pl, u), u)
    else
      dirLibres(obtener(pl, u), u) $\cap$ direccionesOptimas(dameUno(cPosMasCercanaShort(obtener(pl, u), posNoOcupadas(entradas(campus?(u)), u))))
    fi
  else
    if $\emptyset$?(dirLibres(obtener(pl, u), u) $\cap$ direccionesOptimas(dameUno(cPosMasCercanaShort(obtener(pl, u), posicionesHippies(hippies?(u)))))) **then**
      dirLibres(obtener(pl, u), u)
    else
      dirLibres(obtener(pl, u), u) $\cap$ direccionesOptimas(dameUno(cPosMasCercanaShort(obtener(pl, u), posicionesHippies(hippies?(u)))))
    fi
  fi
fi

posibleMovHippie (i, u) $\equiv$
if vacio?(claves(estudiantes?(u))) **then**
  $\emptyset$
else
  if $\emptyset$? (dirLibres(obtener(i, hippies?(u)), u) $\cap$ direccionesOptimas(dameUno(cPosMasCercanaShort(obtener(i, hippies?(u)), posicionesEstudiantes(estudiantes?(u)))))) **then**
    dirLibres(obtener(i, hippies?(u)), u)
  else
    dirLibres(obtener(i, hippies?(u)), u) $\cap$ direccionesOptimas(dameUno(cPosMasCercanaShort(obtener(i, hippies?(u)), posicionesEstudiantes(estudiantes?(u)))))
  fi
fi

dirLibres(p, u) $\equiv$
dirNoOcupadas(dirValidas(p, campus?(u)), p, u)

dirValidas(p, c) $\equiv$
{n,s,e,o} - (**if** col?(p) = 0 **then** {o} **else** $\emptyset$ **fi**) - (**if** col?(p) = ancho?(c)-1 **then** {e} **else** $\emptyset$ **fi**) - (**if** fila?(p) = 0 **then** {n} **else** $\emptyset$ **fi**) - (**if** fila?(p) = alto?(c)-1 **then** {s} **else** $\emptyset$ **fi**)

dirNoOcupadas($cDirs, p, u$) $\equiv$
if ($\emptyset$?(cDirs)) **then**
  $\emptyset$
else
  if posicionNoOcupada(mover(p, dameUno($cDirs$)), u) **then**
    Ag(dameUno($cDirs$), dirNoOcupadas(sinUno($cDirs$), p, u))
  else
    dirNoOcupadas(sinUno($cDirs$), p, u)
  fi
fi

posNoOcupadas($cPos, u$) $\equiv$

if (vacio?($cPos$)) **then**
  $\emptyset$
else
  if posicionNoOcupada(dameUno($cPos$), u) **then**
    Ag(dameUno($cPos$), posNoOcupadas(sinUno($cPos$), u))
  else
    posNoOcupadas(sinUno($cPos$), u)
  fi
fi

posicionesHippies(dH) $\equiv$
if $\emptyset$?(claves(dH)) **then**
  $\emptyset$
else
  Ag (obtener(dameUno(claves(dH)), dH), posicionesHippies(borrar(dameUno(claves(dH)), dH)))
fi

posicionesAgentes(dAs) $\equiv$
if $\emptyset$?(claves(dAs)) **then**
  $\emptyset$
else
  Ag (obtener(dameUno(claves(dAs)), dAs), posicionesAgentes(borrar(dameUno(claves(dAs)), dH)))
fi

posicionesEstudiantes(dEs) $\equiv$
if $\emptyset$?(claves(dEs)) **then**
  $\emptyset$
else
  Ag (Π_2(obtener(dameUno(claves(dEs)), dEs)), posicionesEstudiantes(borrar(dameUno(claves(dEs)), dH)))
fi

posicionNoOcupada(p, u) $\equiv$
$p \in$ (posicionesAgentes(agentes?(u)) $\cup$ posicionesEstudiantes(estudiantes?(u)) $\cup$ posicionesHippies(hippies?(u)) $\cup$ obstaculos?(campus?(u)))

hayDosHippies?(p, dH) $\equiv$
$\#$(posiciones4Vecinas(p) $\cap$ posicionesHippies(dH)) ≥ 2

algunHippieEncerradoPorEstudiantes(dH, dE) $\equiv$
if $\emptyset$?(claves(dH)) **then**
  false
else
  ($\#$(posiciones4Vecinas(obtener(dameUno(claves(dH)), dH)) $\cap$ posicionesEstudiantes(dE)) $=$ 4) $\vee$ algunHippieEncerradoPorEstudiantes(borrar(dameUno(claves(dH)), dH), dE)
fi

hippieEncerradoPorEstudiante(dH, dE) $\equiv$

if #(posiciones4Vecinas(obtener(dameUno(claves(dH)), dH)) ∩ posicionesEstudiantes(dE)) = 4 **then**
  dameUno(claves(dH))
else
  hippieEncerradoPorEstudiante(borrar(dameUno(claves(dH)), dH), dE)
fi

movRestringido?(p, c, dAs, dH, dE) ≡
#((((posVecinasValidas(p, c) - obstaculos?(c)) - posicionesAgentes(dAs)) - posicionesHippies(dH)) - posicionesEstudiantes(dE)) = 0

posVecinasValidas(p, c) ≡
posicionesValidas(posiciones4Vecinas(p), c)

posicionesValidas(cP, c) ≡
if ∅?(cP) **then**
  ∅
else
  if posValida?(dameUno(cP), c) **then**
    Ag(dameUno(cP), posicionesValidas(sinUno(cP), c))
  else
    posicionesValidas(sinUno(cP), c)
  fi
fi

hayUnAgenteAlrededor?(p, dAs) ≡
#(posiciones4Vecinas(p) ∩ posicionesAgentes(dAs)) ≥ 1

murioAlgunHippie?(c, dAs, dH, dE) ≡
murioAlgunHippieAux(dH, c, dAs, dH, dE)

murioAlgunHippieAux(rec, c, dAs, dH, dE) ≡
if ∅?(claves(rec)) **then**
  false
else
  if movRestringido?(obtener(dameUno(claves(rec)), rec), c, dAs, dH, dE) ∧ hayUnAgenteAlrededor?(obtener(dameUno(claves(rec)), rec), dAs) **then**
    true
  else
    murioAlgunHippieAux(borrar(dameUno(claves(rec)), rec) , c, dAs, dH, dE)
  fi
fi

hippieMuerto(c, dAs, dH, dE) ≡
hippieMuertoAux(dH, c, dAs, dH, dE)

hippieMuertoAux(rec, c, dAs, dH, dE) ≡

if movRestringido?(obtener(dameUno(claves(rec)), rec), c, dAs, dH, dE) $\wedge$
hayUnAgenteAlrededor?(obtener(dameUno(claves(rec)), rec), dAs) **then**
  dameUno(claves(rec))
else
  hippieMuertoAux(borrar(dAmeUno(claves(rec)), rec) , c, dAs, dH, dE)
fi

borrarTodosHippiesMuertos(c, dAs, dH, dE) $\equiv$
if murioAlgunHippie?(c, dAs, dH, dE) **then**
  borrarTodosHippiesMuertos(c, dAs, borrar(hippieMuerto(c, dAs, dH, dE), dH), dE)
else
  dH
fi

borrarTodosHippiesEncerradoPorEstudiantes(c, dAs, dH, dE) $\equiv$
if algunHippieEncerradoPorEstudiantes(dH, dE) **then**
  borrarTodosHippiesMuertos (c, dAs, borrar(hippieEncerradoPorEstudiante(dH, dE), dH), dE)
else
  dH
fi

conviertoAlgunEstudiante?(dH, dE) $\equiv$
if $\emptyset$?(claves(dE)) **then**
  false
else
  if hayDosHippies(Π_2(obtener(dameUno(claves(dE)), dE)) ,dH) **then**
    true
  else
    conviertoAlgunEstudiante?(dH, borrar(dameUno(claves(dE)), dE))
  fi
fi

agregoEstudiantesConvertidos(dH, dE) $\equiv$
if $\emptyset$?(claves(dE)) **then**
  dH
else
  if hayDosHippies(Π_2(obtener(dameUno(claves(dE)), dE)) ,dH) **then**
    definir(dameUno(claves(dE)), Π_2(obtener(dameUno(claves(dE)), dE)),
    agregoEstudiantesConvertidos(dH, borrar(dameUno(claves(dE)), dE)))
  else
    agregoEstudiantesConvertidos(dH, borrar(dameUno(claves(dE)), dE))
  fi
fi

estudianteConvertido(dH, dE) $\equiv$
if hayDosHippies(Π_2(obtener(dameUno(claves(dE)), dE)) ,dH) **then**
  dameUno(claves(dE))
else
  estudianteConvertido(dH, borrar(dameUno(claves(dE)), dE))
fi

borrarTodosEstudiantesConvertidos(dH, dE) $\equiv$

if conviertoAlgunEstudiante?(dH, dE) **then**
  borrarTodosEstudiantesConvertidos(dH, borrar(estudianteConvertido(dH, dE), dE))
else
  dE
fi

convertirTodosHippies(dH, dE) $\equiv$
if algunHippieEncerradoPorEstudiantes(dH, dE) **then**
  definir(hippieEncerradoPorEstudiantes(dH, dE), $\langle$ Π_1(obtener(dameUno(claves(dE)), dE)),
  obtener(hippieEncerradoPorEstudiantes(dH, dE), dH) $\rangle$,
  convertirTodosHippies(borrar(hippieEncerradoPorEstudiantes(dH, dE), dH), dE))
else
  dE
fi

agentesAlrededor(p, dAs) $\equiv$
if $\emptyset$?(claves(dAs)) **then**
  $\emptyset$
else
  if obtener(dameUno(claves(dAs)), dAs) $\in$ posiciones4Vecinas(p) **then**
    Ag(dameUno(claves(dAs)), agentesAlrededor(p, borrar(dameUno(claves(dAs)), dAs)))
  else
    agentesAlrededor(p, borrar(dameUno(claves(dAs)), dAs))
  fi
fi

algunEstudianteEncerradoPorAgentes?(c, dAs, dH, dE) $\equiv$
aEEPAaux(dE, c, dAs, dH, dE)

aEEPAaux(rec, c, dAs, dH, dE) $\equiv$
if $\emptyset$?(claves(rec)) **then**
  false
else
  if movRestringido(Π_2(obtener(dameUno(claves(rec)), rec)), c, dAs, dH, dE) $\wedge$
  hayUnAgenteAlrededor?(Π_2(obtener(dameUno(claves(rec)), rec)), dAs) **then**
    true
  else
    aEEPAaux(borrar(dameUno(claves(rec)), rec), c, dAs, dH, dE)
  fi
fi

agentesQueAtrapanEstudiantes(c, dAs, dH, dE) $\equiv$
aQAEaux(dE, c, dAs, dH, dE)

aQAEaux(rec, c, dAs, dH, dE) $\equiv$

if $\emptyset?(\text{claves}(rec))$ **then**
  $\emptyset$
else
  if movRestringido(Π_2(obtener(dameUno(claves(rec)), rec)), c, dAs, dH, dE) $\wedge$ hayUnAgenteAlrededor?(Π_2(obtener(dameUno(claves(rec)), rec)), dAs) **then**
    agentesAlrededor(Π_2(obtener(dameUno(claves(rec)), rec)), dAs) $\cup$ aQAEaux(borrar(dameUno(claves(rec)), rec), c, dAs, dH, dE)
  else
    aQAEaux(borrar(dameUno(claves(rec)), rec), c, dAs, dH, dE)
  fi
fi

agentesQueAtrapanHippies(c, dAs, dH, dE) $\equiv$
aQAHaux(dH, c, dAs, dH, dE)

aQAHaux(rec, c, dAs, dH, dE) $\equiv$
if $\emptyset?(\text{claves}(rec))$ **then**
  $\emptyset$
else
  if movRestringido(obtener(dameUno(claves(rec)), rec), c, dAs, dH, dE) $\wedge$ hayUnAgenteAlrededor?(obtener(dameUno(claves(rec)), rec), dAs) **then**
    agentesAlrededor(obtener(dameUno(claves(rec)), rec), dAs) $\cup$ aQAHaux(borrar(dameUno(claves(rec)), rec), c, dAs, dH, dE)
  else
    aQAHaux(borrar(dameUno(claves(rec)), rec), c, dAs, dH, dE)
  fi
fi

cuantosHippies?(u) $\equiv$
#(claves(hippies?(u)))

cuantosEstudiantes?(u) $\equiv$
#(claves(estudiantes?(u)))

masVigilante?(u) $\equiv$
menorPlaca(losMasVigilantes(claves(agentes?(u)), u, maxCapturas(claves(agentes?(u)), u)))

losMasVigilantes(cPl, u, mC) $\equiv$
if $\emptyset?(cPl)$ **then**
  $\emptyset$
else
  if capturas?(dameUno(cPl), u) $= mC$ **then**
    Ag(cPl, losMasVigilantes(sinUno(cPl), u))
  else
    losMasVigilantes(sinUno(cPl), u)
  fi
fi

maxCapturas(cPl, u) $\equiv$
if $\emptyset?(cPl)$ **then** 0 **else** max{ capturas?(dameUno(cPl), u), maxCapturas(sinUno(cPl), u) } **fi**

menorPlaca(cPl) $\equiv$
if $\emptyset$?(sinUno(cPl)) **then** dameUno(cPl) **else** min { dameUno(cPl), menorPlaca(sinUno(cPl)) } **fi**

saleDeCampus(d, p, c) $\equiv$
((d = n) $\wedge$ (entradaSuperior(p))) $\vee$ ((d = s) $\wedge$ (entradaInferior(p,c)))

entradaSuperior(p) $\equiv$
fila?(p) = 0

entradaInferior(p,c) $\equiv$
entrada?(p,c) $\wedge$ $\neg$(entradaSuperior(p))

Fin TAD

2. TAD Posicion

TAD Posicion

géneros pos

exporta pos, Generadores, Observadores Basicos,Otras Operaciones

usa Nat ,Bool, Conjunto(α), Dir

igualdad observacional
$(\forall p, p' : \text{pos})$ $(p =_{\text{obs}} p' \iff$ (fila?(p) $=_{\text{obs}}$ fila(p') $\wedge$ col?(p) $=_{\text{obs}}$ col(p')))

observadores básicos

fila? : pos $\longrightarrow$ nat

col? : pos $\longrightarrow$ nat

generadores

nuevaPos : nat f $\times$ nat c $\longrightarrow$ pos {(f $\geq$ 0) $\wedge$(c $\geq$ 0)}

otras operaciones

direccionesOptimas : pos $\times$ pos $\longrightarrow$ conj(dir)

cPosMasCercana : pos $\times$ conj(pos) $\times$ nat $\longrightarrow$ conj(pos))

cPosMasCercanaShort : pos $\times$ conj(pos) $\longrightarrow$ conj(pos))

menorDistancia : pos $\times$ conj(pos) $cPos$ $\longrightarrow$ nat {$\neg$ Vacia?(cPos)}

dist : pos $\times$ pos $\longrightarrow$ nat

posiciones4Vecinas : pos $\longrightarrow$ conj(pos)

mover : pos p $\times$ dir $\longrightarrow$ pos {$\neg$(col?(p) = 0 $\wedge$ (o) = dir) $\wedge$ $\neg$(fila?(p) = 0 $\wedge$ (n) = dir)}

axiomas $\forall$ p, p': pos $\wedge$ $\forall$ $f, c, menorDist$: nat $\forall$ d: dir

fila?(nuevaPos(f,c)) $\equiv$
f

col?(nuevaPos(f,c)) $\equiv$
c

direccionesOptimas(p, p') $\equiv$

{n,s,e,o} − (**if** fila?(p) <= fila?(p') **then** {n} **else** ∅ **fi**)
− (**if** fila?(p) >= fila?(p') **then** {s} **else** ∅ **fi**)
− (**if** col?(p) <= col?(p') **then** {o} **else** ∅ **fi**)
− (**if** col?(p) >= col?(p') **then** {e} **else** ∅ **fi**)

PosMasCercanaShort(p, cPos) ≡
cPosMasCercana(p, cPos, menorDistancia(p,cPos))

cPosMasCercana(p, cPos, menorDist) ≡
if vacio?(cPos) **then**
  ∅
else
  if dist(p, dameUno(cPos)) = menorDist **then**
    Ag(p, cPosMasCercana(p, sinUno(cPos), menorDist))
  else
    cPosMasCercana(p, sinUno(cPos), menorDist)
  fi
fi

menorDistancia(p, cPos) ≡
if ¬ vacia?(sinUno(cpos)) **then**
  min(dist(p, dameUno(cPos)), menorDistancia(p, sinUno(cpos)))
else
  dist(p, dameUno(cPos))
fi

dist(p,p') ≡
(max(fila?(p), fila?(p')) - min(fila?(p), fila?(p')) + (max(col?(p), col?(p')) - min(col?(p), col?(p'))

mover(p, d) ≡
if (d = n) **then**
  nuevaPos(fila?(p)-1, col?(p))
else
  if (d=s) **then**
    nuevaPos(fila?(p)+1, col?(p))
  else
    if (dir = o) **then** nuevaPos(fila?(p), col?(p)-1) **else** nuevaPos(fila?(p), col?(p)+1) **fi**
  fi
fi

posiciones4Vecinas(p) ≡

if (fila?(p) = 0) ∧ (col?(p) = 0) **then**
Ag(nuevaPos(1,0), Ag(nuevaPos(0,1)),∅))
else
if fila?(p) = 0 **then**
Ag(nuevaPos(0, col?(p)-1),Ag(nuevaPos(0, col?(p)+1), Ag(nuevaPos(1, col?(p),∅))))
else
if col?(p) = 0 **then**
Ag(nuevaPos(fila?(p)-1, 0),Ag(nuevaPos(fila?(p)+1, 0),Ag(nuevaPos(fila?(p), 1),∅)))
else
Ag(nuevaPos(fila?(p), col?(p)-1), Ag(nuevaPos(fila?(p), col?(p)+1),
Ag(nuevaPos(fila?(p)+1, col?(p)), Ag(nuevaPos(fila?(p)-1, col?(p)),∅))))
fi
fi
fi

Fin TAD

3. TAD Campus

TAD Campus

géneros campus

exporta campus, Generadores, Observadores Basicos,posValida?

usa Nat ,Bool, Conjunto(α),Posicion

igualdad observacional

$$(\forall ca, ca' : \text{campus}) \left(ca =_{\text{obs}} ca' \iff \begin{pmatrix} \text{ancho?(ca)} = \text{ancho?(ca')} \wedge \text{alto?(ca)} = \text{alto?(ca')} \wedge \\ \text{obstaculos?(ca)} = \text{obstaculos?(ca')} \end{pmatrix} \right)$$

observadores básicos

alto? : campus $\longrightarrow$ nat

ancho? : campus $\longrightarrow$ nat

obstaculos? : campus $\longrightarrow$ conj(*pos*)

generadores

nuevoCampus : nat al x nat al $\longrightarrow$ campus

agregarObs : pos p x campus c $\longrightarrow$ campus {enRango(p,c) ∧ ¬(p ∈ obstaculos?(c)) }

otras operaciones

enRango : pos p x campus c $\longrightarrow$ bool

posValida? : pos p x campus c $\longrightarrow$ bool

axiomas ∀ *c*: campus ∧ ∀ *al*, *an*: nat ∀ *p*: pos

ancho?(nuevoCampus(al,an)) ≡
an

ancho?(agregarObs(p,c)) ≡
ancho?(c)

alto?(nuevoCampus(al,an)) ≡

al

alto?(agregarObs(p,c)) $\equiv$
alto?(c)

obstaculos?(nuevoCampus(al,an)) $\equiv$
$\emptyset$

obstaculos?(agregarObs(p,c)) $\equiv$
Ag(p,obstaculos?(c))

enRango(p,c) $\equiv$
fila?(p) $<$ alto?(c) $\wedge$ col?(p) $<$ ancho?(c)

posValida?(p,c) $\equiv$
fila?(p) $<$ alto?(c) $\wedge$ col?(p) $<$ ancho?(c) $\wedge \neg$(p $\in$ obstaculos?(c)

Fin TAD