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

Especificaciï¿½n

Integrante	LU	Correo electrónico
BENITEZ, Nelson	945/13	nelson.benitez92@gmail.com
ROIZMAN, Violeta	273/11	violeroizman@gmail.com
Vï¿½ZQUEZ, Jï¿½sica	318/13	jesis_93@hotmail.com
ZAVALLA, Agustï¿½n	670/13	nkm747@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. TAD AS

TAD AS

géneros as

igualdad observacional

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

usa CAMPUS,BOOL,NAT,TUPLA,SEG

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

observadores básicos

campus : as ⟶ campus

seguridad : as ⟶ conj(seguridad)

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

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

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

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

generadores

nueva : campus × conj(seguridad) ⟶ as
{(∀ *segs*:e) posValida(c,pos(e)) ∧ (∀ *segs*:s,s1) id(s)!=id(s1) ⇒ pos(s)!=pos(s1)}

moverEst : as a × pos pe × pos pd ⟶ as
$\left\{ \begin{array}{l} posValida(campus(a), pe) \quad \wedge_{\text{L}} \quad hayEst?(a, pe) \quad \wedge \quad adyacente(campus(a), pe, pd) \quad \wedge \\ posValidaPersona(as, pd) \end{array} \right\}$

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

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

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

otras operaciones

#hippies : as a ⟶ nat

#estudiantes : as a ⟶ nat

#masVigilante : as a ⟶ nat

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

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

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

moverTodos : as a × conj(seguridad) *segs* ⟶ conj(seguridad)

moverSeg : as a × seguridad *seg* × pos *posSig* ⟶ seguridad

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

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

estsCerca : as a × pos p $\longrightarrow$ conj(pos)

hippieEncerrado? : as a × pos p $\longrightarrow$ bool

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

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

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

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

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

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

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

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

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

capturadaHippie : as a × pos p $\longrightarrow$ nat

capturadaEst : as a × pos p $\longrightarrow$ nat

validas : as a × conj(pos) p $\longrightarrow$ conj(pos)

posValidaAS : as a × pos p $\longrightarrow$ bool

estsCerca : as a × pos p $\longrightarrow$ conj(pos)

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

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

axiomas

campus(nueva($c, segs$)) $\equiv c$

campus(moverEst(a, p_1, p_2)) $\equiv campus(a)$

campus(nuevoEst(a, p_1)) $\equiv campus(a)$

campus(nuevoHippie(a, p_1)) $\equiv campus(a)$

campus(sacarEst(a, p_1)) $\equiv campus(a)$

seguridad(nueva($c, segs$)) $\equiv segs$

seguridad(moverEst(a, p_1, p_2)) $\equiv moverTodos(a, seguridad(a))$

seguridad(nuevoEst(a, p_1)) $\equiv moverTodos(a, seguridad(a))$

seguridad(nuevoHippie(a, p_1)) $\equiv seguridad(a)$

seguridad(sacarEst(a, p_1)) $\equiv seguridad(a)$

hayEst?(nueva($c, segs$),p) $\equiv False$

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

hayEst?(moverEst(a, p_1, p_2),p) $\equiv$

$$\begin{array}{l} \textbf{if } p_1 = p \textbf{ then} \\ \quad False \\ \textbf{else} \\ \quad \textbf{if } p_2 = p \textbf{ then} \\ \quad\quad \neg(hippiesVecinos(a,p_2) \geq 2) \\ \quad \textbf{else} \\ \quad\quad hayEst?(a,p) \\ \quad \textbf{fi} \\ \textbf{fi} \end{array}$$

hayEst?(nuevoHippie(a, p_1),p) $\equiv hayEst?(a,p)$

hayEst?(sacarEst(a, p_1),p) $\equiv$ **if** $p_1 = p$ **then** $False$ **else** $hayEst?(a,p)$ **fi**

```
hayHippie?(nueva($c, segs$),$p$)              ≡ $False$
hayHippie?((nuevoHippie($a, p_1$),$p$)         ≡ if $p_1 = p$ then $True$ else $hayHippie?(a, p)$ fi
hayHippie?((moverEst($a, p_0, p_1$),$p$)       ≡ 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
                                                         $(hippiesVecinos(a, p) \geq 2)$
                                                     else
                                                         $p \in proxPossHippies(a, possHippies(a))$
                                                     fi
                                                 fi
hayHippie?(nuevoEst($a, p_1$),$p$)             ≡ $hayHippie?(a, p)$
hayHippie?(sacarEst($a, p_1$),$p$)             ≡ $hayHippie?(a, p)$
#capturas(nueva($a, segs$),$s$)                ≡ 0
#capturas(moverEst($a, p_1, p_2$),$s$)         ≡ $\#capturas(a, s)$
#capturas(nuevoHippie($a, p_1$),$s$)           ≡ if $(adyacente(a, p_1, posSeg(a, s)) \land encerrado(a, p_1))$ then
                                                     $1 + \#capturas(a, s)$
                                                 else
                                                     $\#capturas(a, s)$
                                                 fi
#capturas(nuevoEst($a, p_1$),$s$)              ≡ $\#capturas(a, s)$
#capturas(sacarEst($a, p_1$),$s$)              ≡ $\#capturas(a, s)$
#capturas(moverEst($a, p_1, p_2$),$s$)         ≡ $capturadoHippie(a, < \pi_1(posSeg) + 1, \pi_2(posSeg) >) + capturadoHippie(a, < \pi_1(posSeg) - 1, \pi_2(posSeg) >) + capturadoHippie(a, < \pi_1(posSeg), \pi_2(posSeg) + 1 >) + capturadoHippie(a, < \pi_1(posSeg), \pi_2(posSeg) - 1 >) + \#capturas(a, s)$
#sanciones(nueva($a, segs$),$s$)               ≡ 0
#sanciones(moverEst($a, p_1, p_2$),$s$)        ≡ $\#sanciones(a, s)$
#sanciones(nuevoHippie($a, p_1$),$s$)          ≡ if $(cercanos?(a, p_1, posSeg(a, s)) \land_{L} (hayEst?(casilleroEnComun(a, p_1, posSeg(a, s))) \land encerrado(casilleroEnComun(a, p_1, posSeg(a, s)))))$ then
                                                     $1 + \#sanciones(a, s)$
                                                 else
                                                     $\#sanciones(a, s)$
                                                 fi
#sanciones(nuevoEst($a, p_1$),$s$)             ≡ $\#sanciones(a, s)$
#sanciones(sacarEst($a, p_1$),$s$)             ≡ $\#sanciones(a, s)$
#sanciones(moverEst($a, p_1, p_2$),$s$)        ≡ $capturadoEst(a, < \pi_1(posSeg) + 1, \pi_2(posSeg) >) + capturadoEst(a, < \pi_1(posSeg) - 1, \pi_2(posSeg) >) + capturadoEst(a, < \pi_1(posSeg), \pi_2(posSeg) + 1 >) + capturadoEst(a, < \pi_1(posSeg), \pi_2(posSeg) - 1 >) + \#sanciones(a, s)$
```

```
moverTodos(a,segs)    ≡ 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
moverIngreso(a,segs)  ≡ 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
```

```
proxPoss(entCerca, p)    ≡ 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
                           fi
```

```
posValidaAS(a,p)                 ≡ $posValida(dameUno(poss))\land$
                                   $\neg(hayHippie?(a, dameUno(poss)))\land$
                                   $\neg(hayEst?(a, dameUno(poss)))\land$
                                   $\neg(haySeg?(a, dameUno(poss), seguridad(a))$
validas(a,poss)                  ≡ 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)            ≡ $hipEncerradoEst?(a, p, adyacentes(campus(a), p))$          $\land$
                                   $hipEncerradoSeg?(a, p, adyacentes(campus(a), p))$
hipEncerradoEst?(a,p,adys)       ≡ if $\emptyset?(adys)$ then
                                       $True$
                                   else
                                       if $posValida?(campus(a), dameUno(adys))$ then
                                           $hayEst?(a, p) \land hipEncerradoEst?(a, p, sinUno(adys))$
                                       else
                                           $False$
                                       fi
                                   fi
hipEncerradoSeg?(a,p,adys)       ≡ if $\emptyset?(adys)$ then
                                       $True$
                                   else
                                       if $posValida?(campus(a), dameUno(adys))$ then
                                           $haySeg?(a, p, seguridad(a))$                        $\land$
                                           $hipEncerradoSeg?(a, p, sinUno(adys))$
                                       else
                                           $False$
                                       fi
                                   fi
moverSeg(a,seg,nPos)             ≡ if $(distMan(campus(a), \pi_2(seg), nPos) \geq 2$
                                   $\lor\neg(posValidaAS(a, nPos)))$ then
                                       $seg$
                                   else
                                       if $\#sanciones(a, seg) < 3$ then
                                           $< \pi_1(seg), nPos >$
                                       else
                                           $seg$
                                       fi
                                   fi
haySeg?(a,p,segs)                ≡ 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)   ≡ if $\emptyset?(possHippies)$ then
                                       $\emptyset$
                                   else
                                       $proxPoss(a, estsCerca(dameUno(possHippies), dameUno(possHippies$
                                       $proxPossHippies(a, sinUno(possHippies))$
                                   fi
```

```
hippiesMasCerca(a,seg)   ≡ $minDistsPos(campus(a), \pi_2(seg), posHippies(a, conjPos(campus(a))))$
estsCerca(a,p)           ≡ $minDistsPos(campus(a), p, posEsts(a, conjPos(campus(a))))$
posHippies(a,conjpos)    ≡ 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)       ≡ 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
#hippies(a)              ≡ $\#(posHippies(a, conjPos(campus(a))))$
#estudiantes(a)          ≡ $\#(posEsts(a, conjPos(campus(a))))$
masVigilante(a)          ≡ $dameUno(masCapturas(a, seguridad(a)))$
masCapturas(a,segs)      ≡ 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)      ≡ 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
```

```
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
```

```
hippieNatural($a, p$)      ≡ 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($a, p$)   ≡ if $\neg(hayObstaculo?(campus(a), p))$ then
                                 ($\pi_2(p) = 0 \lor \pi_2$=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 ⟶ nat

ancho : campus ⟶ nat

obstaculos : campus ⟶ conj(pos)

generadores

nuevo : nat *ancho* × nat *alto* × conj(pos) *obst* ⟶ campus
$\{1 \leq ancho \land 1 \leq alto \land (\forall\ p\text{:pos})\ p \in obst \Rightarrow_{\text{L}} posValida(c, p)\}$

otras operaciones

adyacente : campus c × pos pe × pos pd ⟶ bool $\{posValida(c, pe) \land posValida(c, pd)\}$

posValida : campus c × pos p ⟶ bool

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

minDistsPos : campus c × pos p × conj(pos) *posiciones* ⟶ conj(pos)
$\{posValida(c, p) \land \neg(\emptyset?(posiciones))\}$

minDist : campus c × pos p × conj(posiciones) *posiciones* ⟶ nat
$\{posValida(c, p) \land \neg(\emptyset?(posiciones))\}$

distMan : campus c × pos $p1$ × pos $p2$ ⟶ nat $\{posValida(c, p1) \land posValida(c, p2)\}$

restaAbs : nat × nat ⟶ nat

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

adyacentes : campus × pos $\longrightarrow$ conj(pos)

hayObstaculo? : campus c × pos p $\longrightarrow$ 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(nuevo(*ancho*,*alto*,*obst*),p_1) ≡ $\pi_1(p_1) < ancho \wedge \pi_2(p_1) < alto$

adyacente(nuevo(*ancho*,*alto*,*obst*),p_1,p_2) ≡ $(\pi_1(p_1) = \pi_1(p_2) - 1 \vee \pi_1(p_1) = \pi_1(p_2) + 1) \wedge (\pi_2(p_1) = \pi_2(p_2) - 1 \vee \pi_2(p_1) = \pi_2(p_2) + 1)$

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(campus,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