

Algoritmos y Estructuras de Datos II

Segundo Cuatrimestre de 2015

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

Trabajo Práctico 1

Grupo 9

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	mdg_92@yahoo.com.ar

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. TAD AS	3
2. TAD CAMPUS	11
3. TAD POS ES TUPLA(NAT,NAT)	13
4. TAD SEGURIDAD ES TUPLA(ID, POS)	13
5. TAD ID ES NAT	13
6. Consideraciones	14

1. TAD AS

TAD AS

géneros as

igualdad observacional

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

usa CAMPUS, BOOL, NAT, POS, SEGURIDAD

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

observadores básicos

campus : as \rightarrow campus

seguridad : as \rightarrow conj(seguridad)

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

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

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

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

generadores

nueva : campus \times conj(seguridad) \rightarrow as
 $\{(\forall segs:e) posValida(c, pos(e)) \wedge (\forall segs:s, s1) id(s) \neq id(s1) \Rightarrow pos(s) \neq pos(s1)\}$

moverEst : as $a \times$ pos $pe \times$ pos $pd \rightarrow$ as
 $\left\{ \begin{array}{l} posValida(campus(a), pe) \wedge_L hayEst?(a, pe) \wedge adyacente(campus(a), pe, pd) \wedge \\ posValidaPersona(as, pd) \end{array} \right\}$

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

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

sacarEst : as $a \times$ pos $p \rightarrow$ as $\{posValida(campus(a), p) \wedge_L hayEst?(a, p) \wedge posIngreso(a, p)\}$

otras operaciones

#hippies : as $a \rightarrow$ nat

#estudiantes : as $a \rightarrow$ nat

#masVigilante : as $a \rightarrow$ nat

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

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

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

moverTodos : as $a \times$ conj(seguridad) $segs \rightarrow$ conj(seguridad)

$\text{moverSeg} : \text{as } a \times \text{seguridad } seg \times \text{pos } posSig \rightarrow \text{seguridad}$	
$\text{proxPoss} : \text{as } a \times \text{conj}(\text{pos}) \text{ minPos} \times \text{pos } posAct \rightarrow \text{conj}(\text{pos})$	$\{\neg(\text{emptyset?}(\text{minPos})) \wedge_L \text{posValida}(\text{campus}(a), \text{posAct}) \wedge \text{posicionesValidas}(\text{campus}(a), \text{minPos})\}$
$\text{proxPossHippies} : \text{as } a \times \text{conj}(\text{pos}) \text{ poss} \rightarrow \text{conj}(\text{pos})$	$\{(\forall \text{ poss:p}) \text{ posValida}(a, p) \wedge_L \text{hayHippie?}(a, p)\}$
$\text{estsCerca} : \text{as } a \times \text{pos } p \rightarrow \text{conj}(\text{pos})$	
$\text{hippieEncerrado?} : \text{as } a \times \text{pos } p \rightarrow \text{bool}$	
$\text{hippieEncerradoEst?} : \text{as } a \times \text{pos } p \times \text{conj}(\text{pos}) \text{ poss} \rightarrow \text{bool}$	
$\text{hippieEncerradoSeg?} : \text{as } a \times \text{pos } p \times \text{conj}(\text{pos}) \text{ poss} \rightarrow \text{bool}$	
$\text{hippiesMasCerca} : \text{as } a \times \text{seguridad } seg \rightarrow \text{conj}(\text{pos})$	$\{seg \in \text{seguridad}(a) \wedge \text{hayHippies}(a)\}$
$\text{encerrado} : \text{as } a \times \text{pos } p \rightarrow \text{bool}$	$\{\text{posValida}(\text{campus}(as), p) \wedge \text{hayEst?}(p)\}$
$\#masCapturas : \text{as } a \times \text{conj}(\text{seg}) \text{ segs} \rightarrow \text{conj}(\text{seg})$	$\{(\forall \text{ segs:s}) s \in \text{seguridad}(a)\}$
$\#maxCapturas : \text{as } a \times \text{conj}(\text{seg}) \text{ segs} \rightarrow \text{nat}$	$\{(\forall \text{ segs:s}) s \in \text{seguridad}(a)\}$
$\text{captura?} : \text{as } a \times \text{pos } p \rightarrow \text{bool}$	$\{\text{posValida}(\text{campus}(as), p)\}$
$\text{hippiesVecinos} : \text{as } a \times \text{pos } p \rightarrow \text{nat}$	$\{\text{posValida}(\text{campus}(as), p)\}$
$\text{hippiesAlrededor} : \text{as } a \times \text{pos } p \rightarrow \text{nat}$	$\{\text{posValida}(\text{campus}(as), p)\}$
$\text{capturadaHippie} : \text{as } a \times \text{pos } p \rightarrow \text{nat}$	
$\text{capturadaEst} : \text{as } a \times \text{pos } p \rightarrow \text{nat}$	
$\text{validas} : \text{as } a \times \text{conj}(\text{pos}) p \rightarrow \text{conj}(\text{pos})$	
$\text{posValidaAS} : \text{as } a \times \text{pos } p \rightarrow \text{bool}$	
$\text{estsCerca} : \text{as } a \times \text{pos } p \rightarrow \text{conj}(\text{pos})$	
$\text{posHippies} : \text{as } a \times \text{conj}(\text{pos}) \text{ poss} \rightarrow \text{conj}(\text{pos})$	$\{(\forall \text{ poss:p}) \text{ posValida}(p)\}$
$\text{posEsts} : \text{as } a \times \text{conj}(\text{pos}) \text{ poss} \rightarrow \text{conj}(\text{pos})$	$\{(\forall \text{ poss:p}) \text{ posValida}(p)\}$

axiomas

$\text{campus}(\text{nueva}(c, \text{segs}))$	$\equiv c$
$\text{campus}(\text{moverEst}(a, p_1, p_2))$	$\equiv \text{campus}(a)$
$\text{campus}(\text{nuevoEst}(a, p_1))$	$\equiv \text{campus}(a)$
$\text{campus}(\text{nuevoHippie}(a, p_1))$	$\equiv \text{campus}(a)$
$\text{campus}(\text{sacarEst}(a, p_1))$	$\equiv \text{campus}(a)$
$\text{seguridad}(\text{nueva}(c, \text{segs}))$	$\equiv \text{segs}$
$\text{seguridad}(\text{moverEst}(a, p_1, p_2))$	$\equiv \text{moverTodos}(a, \text{seguridad}(a))$
$\text{seguridad}(\text{nuevoEst}(a, p_1))$	$\equiv \text{moverTodos}(a, \text{seguridad}(a))$
$\text{seguridad}(\text{nuevoHippie}(a, p_1))$	$\equiv \text{seguridad}(a)$
$\text{seguridad}(\text{sacarEst}(a, p_1))$	$\equiv \text{seguridad}(a)$
$\text{hayEst?}(\text{nueva}(c, \text{segs}), p)$	$\equiv \text{False}$
$\text{hayEst?}(\text{nuevoEst}(a, p_1), p)$	$\equiv \text{if } p_1 = p \text{ then True else hayEst?}(a, p) \text{ fi}$

<code>hayEst?(moverEst(a, p_1, p_2), p)</code>	\equiv <code>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</code>
<code>hayEst?(nuevoHippie(a, p_1), p)</code>	\equiv <code>$hayEst?(a, p)$</code>
<code>hayEst?(sacarEst(a, p_1), p)</code>	\equiv <code>if $p_1 = p$ then $False$ else $hayEst?(a, p)$ fi</code>
<code>hayHippie?(nueva($c, segs$), p)</code>	\equiv <code>$False$</code>
<code>hayHippie?(nuevoHippie(a, p_1), p)</code>	\equiv <code>if $p_1 = p$ then $True$ else $hayHippie?(a, p)$ fi</code>
<code>hayHippie?(moverEst(a, p_0, p_1), p)</code>	\equiv <code>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</code>
<code>hayHippie?(nuevoEst(a, p_1), p)</code>	\equiv <code>$hayHippie?(a, p)$</code>
<code>hayHippie?(sacarEst(a, p_1), p)</code>	\equiv <code>$hayHippie?(a, p)$</code>
<code>#capturas(nueva($a, segs$), s)</code>	\equiv <code>0</code>
<code>#capturas(nuevoHippie(a, p_1), s)</code>	\equiv <code>if ($adyacente(a, p_1, posSeg(a, s))$) then $capturadoHippie(a, p_1) + \#capturas(a, s)$ else $\#capturas(a, s)$ fi</code>
<code>#capturas(nuevoEst(a, p_1), s)</code>	\equiv <code>$\#capturas(a, s)$</code>
<code>#capturas(sacarEst(a, p_1), s)</code>	\equiv <code>$\#capturas(a, s)$</code>
<code>#capturas(moverEst(a, p_1, p_2), s)</code>	\equiv <code>$capturadoHippie(a, < \pi_1(posSeg(a, s)) + 1, \pi_2(posSeg(a, s)) >) +$ $capturadoHippie(a, < \pi_1(posSeg(a, s)) - 1, \pi_2(posSeg(a, s)) >) +$ $capturadoHippie(a, < \pi_1(posSeg(a, s)), \pi_2(posSeg(a, s)) + 1 >) +$ $capturadoHippie(a, < \pi_1(posSeg(a, s)), \pi_2(posSeg(a, s)) - 1 >) +$ $\#capturas(a, s)$</code>
<code>#sanciones(nueva($a, segs$), s)</code>	\equiv <code>0</code>
<code>#sanciones(nuevoHippie(a, p_1), s)</code>	\equiv <code>$\#sanciones(a, s)$</code>
<code>#sanciones(nuevoEst(a, p_1), s)</code>	\equiv <code>$\#sanciones(a, s)$</code>
<code>#sanciones(sacarEst(a, p_1), s)</code>	\equiv <code>$\#sanciones(a, s)$</code>
<code>#sanciones(moverEst(a, p_1, p_2), s)</code>	\equiv <code>$capturadoEst(a, < \pi_1(posSeg(a, s)) + 1, \pi_2(posSeg(a, s)) >) +$ $capturadoEst(a, < \pi_1(posSeg(a, s)) - 1, \pi_2(posSeg(a, s)) >) +$ $capturadoEst(a, < \pi_1(posSeg(a, s)), \pi_2(posSeg(a, s)) + 1 >) +$ $capturadoEst(a, < \pi_1(posSeg(a, s)), \pi_2(posSeg(a, s)) - 1 >) +$ $\#sanciones(a, s)$</code>
<code>#hippies(a)</code>	\equiv <code>$\#(posHippies(a, conjPos(campus(a))))$</code>
<code>#estudiantes(a)</code>	\equiv <code>$\#(posEsts(a, conjPos(campus(a))))$</code>

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)

```

≡ 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
            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
                    fi
                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
        fi
    fi
fi

```

```

moverIngreso(a,segs)      ≡ if ∅?(segs) then
                             ∅
                             else
                               if (alto(campus(a))-1) - π2(dameUno(segs)) > π2(dameUno(segs))
                               then
                                 ag(moverIngreso(a, sinUno(segs)), mover(dameUno(segs), <
                                   (π1(dameUno(segs)), π2(segs) - 1) >))
                               else
                                 if (alto(campus(a)) - 1) - π2(dameUno(segs)) < π2(dameUno(segs))
                                 then
                                   ag(moverIngreso(a, sinUno(segs)), mover(dameUno(segs), <
                                     (π1(dameUno(segs)), π2(segs) + 1) >))
                                   else
                                     ag(moverIngreso(a, sinUno(segs)), mover(dameUno(segs),
                                       dameUno({ < (π1(dameUno(segs)), π2(segs) - 1) >, <
                                         (π1(dameUno(segs)), π2(segs) + 1) > })))
                                 fi
                               fi
                             fi
                             fi

posValidaAS(a,p)          ≡ posValida(dameUno(poss)) ∧
                             ¬(hayHippie?(a, dameUno(poss))) ∧
                             ¬(hayEst?(a, dameUno(poss))) ∧
                             ¬(haySeg?(a, dameUno(poss), seguridad(a)))

validas(a,poss)           ≡ if ∅?(poss) then
                             ∅
                             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))          ∧
                             hipEncerradoSeg?(a, p, adyacentes(campus(a), p))

hipEncerradoEst?(a,p,adys) ≡ if ∅?(adys) then
                             True
                             else
                               if posValida?(campus(a), dameUno(adys)) then
                                 hayEst?(a, p) ∧ hipEncerradoEst?(a, p, sinUno(adys))
                               else
                                 False
                               fi
                             fi

hipEncerradoSeg?(a,p,adys) ≡ if ∅?(adys) then
                             True
                             else
                               if posValida?(campus(a), dameUno(adys)) then
                                 haySeg?(a, p, seguridad(a))
                                 hipEncerradoSeg?(a, p, sinUno(adys))          ∧
                               else
                                 False
                               fi
                             fi

```


moverSeg(a,seg,nPos)	≡	if (<i>distMan</i> (<i>campus</i> (<i>a</i>), π_2 (<i>seg</i>), <i>nPos</i>) ≥ 2 $\vee \neg$ (<i>posValidaAS</i> (<i>a</i> , <i>nPos</i>))) then <i>seg</i> else if # <i>sanciones</i> (<i>a</i> , <i>seg</i>) < 3 then < π_1 (<i>seg</i>), <i>nPos</i> > else <i>seg</i> fi fi
haySeg?(a,p,segs)	≡	if $\emptyset?$ (<i>segs</i>) then <i>False</i> else if π_2 (<i>dameUno</i> (<i>segs</i>)) == <i>p</i> then <i>True</i> else <i>haySeg?</i> (<i>a</i> , <i>p</i> , <i>sinUno</i> (<i>segs</i>)) fi fi
proxPossHippies(a,possHippies)	≡	if $\emptyset?$ (<i>possHippies</i>) then \emptyset else <i>proxPoss</i> (<i>a</i> , <i>estsCerca</i> (<i>dameUno</i> (<i>possHippies</i>), <i>dameUno</i> (<i>possHippies</i>)) \cup <i>proxPossHippies</i> (<i>a</i> , <i>sinUno</i> (<i>possHippies</i>))) fi fi
hippiesMasCerca(a,seg)	≡	<i>minDistsPos</i> (<i>campus</i> (<i>a</i>), π_2 (<i>seg</i>), <i>posHippies</i> (<i>a</i> , <i>conjPos</i> (<i>campus</i> (<i>a</i>))))
estsCerca(a,p)	≡	<i>minDistsPos</i> (<i>campus</i> (<i>a</i>), <i>p</i> , <i>posEsts</i> (<i>a</i> , <i>conjPos</i> (<i>campus</i> (<i>a</i>))))
posHippies(a,conjpos)	≡	if $\emptyset?$ (<i>conjpos</i>) then \emptyset else if <i>hayHippie?</i> (<i>a</i> , <i>dameUno</i> (<i>conjpos</i>)) then <i>Ag</i> (<i>posHippies</i> (<i>a</i> , <i>sinUno</i> (<i>conjpos</i>)), <i>dameUno</i> (<i>conjpos</i>)) else <i>posHippies</i> (<i>a</i> , <i>sinUno</i> (<i>conjpos</i>)) fi fi
posEsts(a,conjpos)	≡	if $\emptyset?$ (<i>conjpos</i>) then \emptyset else if <i>hayEst?</i> (<i>a</i> , <i>dameUno</i> (<i>conjpos</i>)) then <i>Ag</i> (<i>posEsts</i> (<i>a</i> , <i>sinUno</i> (<i>conjpos</i>)), <i>dameUno</i> (<i>conjpos</i>)) else <i>posEsts</i> (<i>a</i> , <i>sinUno</i> (<i>conjpos</i>)) fi fi

```

captura?(a, p)
≡ if (posValida(campus(a), < π1(p) + 1, π2(p) >) then
    (hayObstaculo?(campus(a), < π1(p) + 1, π2(p) >)
    ) ∨ haySeg?(a, < π1(p) + 1, π2(p) >, seguridad(a)))
else
    ¬(hayEst?(a, < π1(p), π2(p) >))
fi
^
if (posValida(campus(a), < π1(p) - 1, π2(p) >) then
    (hayObstaculo?(campus(a), < π1(p) - 1, π2(p) >)
    ) ∨ haySeg?(a, < π1(p) - 1, π2(p) >, seguridad(a)))
else
    ¬(hayEst?(a, < π1(p), π2(p) >))
fi
^
if (posValida(campus(a), < π1(p), π2(p) + 1 >) then
    (hayObstaculo?(campus(a), < π1(p), π2(p) + 1 >)
    ) ∨ haySeg?(a, < π1(p), π2(p) + 1 >, seguridad(a)))
else
    True
fi
^
if (posValida(campus(a), < π1(p), π2(p) - 1 >) then
    (hayObstaculo?(campus(a), < π1(p), π2(p) - 1 >)
    ) ∨ haySeg?(a, < π1(p), π2(p) - 1 >, seguridad(a)))
else
    True
fi
fi

capturadoHippie(a, p)
≡ 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)
≡ 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)
≡ if hayHippie?(a, p) then 1 else 0 fi

```

```

hippiesAlrededor(a, p)
≡ if posValida (campus(a), <π1(p) + 1, π2(p)>) then
    hippiesVecinos(a, <π1(p) + 1, π2(p)>)
  else
    0
  fi + if posValida (campus(a), <π1(p) - 1, π2(p)>) then
    hippiesVecinos(a, <π1(p) - 1, π2(p)>)
  else
    0
  fi + if posValida (campus(a), <π1(p), π2(p) + 1>) then
    hippiesVecinos(a, <π1(p), π2(p) + 1>)
  else
    0
  fi + if posValida (campus(a), <π1(p), π2(p) - 1>) then
    hippiesVecinos(a, <π1(p), π2(p) - 1>)
  else
    0
  fi
posValidaPersona(a, p)
≡ if ¬(hayObstaculo?(campus(a), p)) then
    (π2(p) = 0 ∨ π2 = alto(campus(a)))
  else
    False
  fi

```

Fin TAD

2. TAD CAMPUS

TAD CAMPUS

géneros campus

igualdad observacional

$$(\forall c, c' : \text{Campus}) \left(c =_{\text{obs}} c' \iff \left(\begin{array}{l} \text{alto}(c) =_{\text{obs}} \text{alto}(c') \wedge \\ \text{ancho}(c) =_{\text{obs}} \text{ancho}(c') \wedge \\ \text{obstaculos}(c) =_{\text{obs}} \text{obstaculos}(c') \end{array} \right) \right)$$

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

$$\{1 \leq \text{ancho} \wedge 1 \leq \text{alto} \wedge (\forall p : \text{pos}) p \in \text{obst} \Rightarrow_{\text{L}} \text{posValida}(c, p)\}$$

otras operaciones

adyacente : campus *c* × pos *pe* × pos *pd* → bool

$$\{\text{posValida}(c, pe) \wedge \text{posValida}(c, pd)\}$$

posValida : campus *c* × pos *p* → bool

posIngreso : campus *c* × pos *p* → bool

$$\{\text{posValida}(c, p)\}$$

minDistsPos : campus *c* × pos *p* × conj(pos) posiciones → conj(pos)

$$\{\text{posValida}(c, p) \wedge \neg(\emptyset?(posiciones))\}$$

$\text{minDist} : \text{campus } c \times \text{pos } p \times \text{conj}(\text{posiciones}) \text{ posiciones} \longrightarrow \text{nat}$	$\{ \text{posValida}(c, p) \wedge \neg(\emptyset?(\text{posiciones})) \}$
$\text{distMan} : \text{campus } c \times \text{pos } p_1 \times \text{pos } p_2 \longrightarrow \text{nat}$	$\{ \text{posValida}(c, p_1) \wedge \text{posValida}(c, p_2) \}$
$\text{restaAbs} : \text{nat} \times \text{nat} \longrightarrow \text{nat}$	
$\text{conjPos} : \text{campus} \times \text{nat} \times \text{nat} \longrightarrow \text{conj}(\text{pos})$	
$\text{adyacentes} : \text{campus} \times \text{pos} \longrightarrow \text{conj}(\text{pos})$	
$\text{hayObstaculo?} : \text{campus } c \times \text{pos } p \longrightarrow \text{bool}$	$\{ \text{posValida}(c, p) \}$
axiomas $\forall \text{alto}:\text{nat}, \forall \text{ancho}:\text{nat}, \forall \text{obst}:\text{conj}(\text{pos})$ $\forall p_1:\text{pos} \forall p_2:\text{pos}$	
$\text{alto}(\text{nuevo}(\text{ancho}, \text{alto}, \text{obst}))$	$\equiv \text{alto}$
$\text{ancho}(\text{nuevo}(\text{ancho}, \text{alto}, \text{obst}))$	$\equiv \text{ancho}$
$\text{obstaculos}(\text{nuevo}(\text{ancho}, \text{alto}, \text{obst}))$	$\equiv \text{obst}$
$\text{posValida}(c, p)$	$\equiv \pi_1(p) < \text{ancho} \wedge \pi_2(p) < \text{alto} \wedge \neg(\text{hayObstaculo?}(a, p))$
$\text{adyacente}(c, p_1, p_2)$	$\equiv (\pi_1(p_1) = \pi_1(p_2) - 1 \vee \pi_1(p_1) = \pi_1(p_2) + 1) \vee (\pi_2(p_1) = \pi_2(p_2) - 1 \vee \pi_2(p_1) = \pi_2(p_2) + 1)$
$\text{minDistsPos}(c, p, \text{posiciones})$	$\equiv \text{if } \emptyset?(\text{sinUno}(\text{posiciones})) \text{ then } \text{dameUno}(\text{posiciones})$ else $\text{if } \text{distMan}(c, p, \text{dameUno}(\text{posiciones})) \leq \text{minDist}(c, p, \text{posiciones}) \text{ then}$ $\text{Ag}(\text{minDistsPos}(c, \text{sinUno}(\text{posiciones})), \text{dameUno}(\text{posiciones}))$ else $\text{minDistsPos}(c, \text{seg}, \text{sinUno}(\text{posiciones}))$ fi fi
$\text{minDist}(c, p, \text{posiciones})$	$\equiv \text{if } \emptyset?(\text{sinUno}(\text{posiciones})) \text{ then } \text{distMan}(c, p, \text{dameUno}(\text{posiciones}))$ else $\text{if } \text{distMan}(c, p, \text{dameUno}(\text{posiciones})) \leq \text{minDist}(c, \text{pos}/p, \text{sinUno}(\text{posiciones})) \text{ then}$ $\text{distMan}(c, p, \text{dameUno}(\text{posiciones}))$ else $\text{minDist}(c, p, \text{sinUno}(\text{posiciones}))$ fi fi
$\text{distMan}(c, p_1, p_2)$	$\equiv \text{restaAbs}(\pi_2(p_1), \pi_2(p_2)) + \text{restaAbs}(\pi_1(p_1), \pi_1(p_2))$
$\text{restaAbs}(n_1, n_2)$	$\equiv \text{if } n_2 > n_1 \text{ then } n_2 - n_1 \text{ else } n_1 - n_2 \text{ fi}$
$\text{conjPos}(c, x, y)$	$\equiv \text{if } x \geq \text{ancho}(c) \text{ then } \emptyset$ else $\text{if } y \geq \text{alto}(c) \text{ then } \text{conjPos}(c, x + 1, 0)$ else $\text{ag}(\text{conjPos}(c, x, y + 1), < x, y >)$ fi fi
$\text{adyacentes}(c, p)$	$\equiv \{ < \pi_1(p) + 1, \pi_2(p) + 1 > < \pi_1(p) - 1, \pi_2(p) - 1 > < \pi_1(p) + 1, \pi_2(p) > < \pi_1(p), \pi_2(p) + 1 > \}$

hayObstaculo?(c,p)

$\equiv p \in obstaculos(c)$

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