

# 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	haris@live.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**

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

**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  
 $\left\{ \begin{array}{l} posValida(campus(a), pe) \quad \wedge_L \quad hayEst?(a, pe) \quad \wedge \quad adyacente(campus(a), pe, pd) \\ posValidaPersona(as, pd) \end{array} \right\}$

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(<math>a, p_1, p_2</math>), <math>p</math>)</code>	$\equiv$ <code>if <math>p_1 = p</math> then     <math>False</math> else     if <math>p_2 = p</math> then         <math>\neg(hippiesAlrededor(a, p_2) \geq 2)</math>     else         <math>hayEst?(a, p)</math>     fi fi</code>
<code>hayEst?(nuevoHippie(<math>a, p_1</math>), <math>p</math>)</code>	$\equiv$ <code><math>hayEst?(a, p)</math></code>
<code>hayEst?(sacarEst(<math>a, p_1</math>), <math>p</math>)</code>	$\equiv$ <code>if <math>p_1 = p</math> then <math>False</math> else <math>hayEst?(a, p)</math> fi</code>
<code>hayHippie?(nueva(<math>c, segs</math>), <math>p</math>)</code>	$\equiv$ <code><math>False</math></code>
<code>hayHippie?(nuevoHippie(<math>a, p_1</math>), <math>p</math>)</code>	$\equiv$ <code>if <math>p_1 = p</math> then <math>True</math> else <math>hayHippie?(a, p)</math> fi</code>
<code>hayHippie?(moverEst(<math>a, p_0, p_1</math>), <math>p</math>)</code>	$\equiv$ <code>if <math>hayHippie?(a, p)</math> then     if <math>\neg(hippieEncerrado?(a, p))</math> then         <math>p \in proxPossHippies(a, possHippies(a))</math>     else         <math>False</math>     fi else     if <math>hayEst?(a, p)</math> then         <math>(hippiesAlrededor(a, (a, p)) \geq 2)</math>     else         <math>p \in proxPossHippies(a, possHippies(a))</math>     fi fi</code>
<code>hayHippie?(nuevoEst(<math>a, p_1</math>), <math>p</math>)</code>	$\equiv$ <code><math>hayHippie?(a, p)</math></code>
<code>hayHippie?(sacarEst(<math>a, p_1</math>), <math>p</math>)</code>	$\equiv$ <code><math>hayHippie?(a, p)</math></code>
<code>#capturas(nueva(<math>a, segs</math>), <math>s</math>)</code>	$\equiv$ <code><math>0</math></code>
<code>#capturas(nuevoHippie(<math>a, p_1</math>), <math>s</math>)</code>	$\equiv$ <code>if (<math>adyacente(a, p_1, posSeg(a, s))</math>) then     <math>capturadoHippie(a, p_1) + \#capturas(a, s)</math> else     <math>\#capturas(a, s)</math> fi</code>
<code>#capturas(nuevoEst(<math>a, p_1</math>), <math>s</math>)</code>	$\equiv$ <code><math>\#capturas(a, s)</math></code>
<code>#capturas(sacarEst(<math>a, p_1</math>), <math>s</math>)</code>	$\equiv$ <code><math>\#capturas(a, s)</math></code>
<code>#capturas(moverEst(<math>a, p_1, p_2</math>), <math>s</math>)</code>	$\equiv$ <code><math>capturadoHippie(a, &lt; \pi_1(posSeg(a, s)) + 1, \pi_2(posSeg(a, s)) &gt;) +</math>     <math>capturadoHippie(a, &lt; \pi_1(posSeg(a, s)) - 1, \pi_2(posSeg(a, s)) &gt;) +</math>     <math>capturadoHippie(a, &lt; \pi_1(posSeg(a, s)), \pi_2(posSeg(a, s)) + 1 &gt;) +</math>     <math>capturadoHippie(a, &lt; \pi_1(posSeg(a, s)), \pi_2(posSeg(a, s)) - 1 &gt;) +</math>     <math>\#capturas(a, s)</math></code>
<code>#sanciones(nueva(<math>a, segs</math>), <math>s</math>)</code>	$\equiv$ <code><math>0</math></code>
<code>#sanciones(nuevoHippie(<math>a, p_1</math>), <math>s</math>)</code>	$\equiv$ <code><math>\#sanciones(a, s)</math></code>
<code>#sanciones(nuevoEst(<math>a, p_1</math>), <math>s</math>)</code>	$\equiv$ <code><math>\#sanciones(a, s)</math></code>
<code>#sanciones(sacarEst(<math>a, p_1</math>), <math>s</math>)</code>	$\equiv$ <code><math>\#sanciones(a, s)</math></code>
<code>#sanciones(moverEst(<math>a, p_1, p_2</math>), <math>s</math>)</code>	$\equiv$ <code><math>capturadoEst(a, &lt; \pi_1(posSeg(a, s)) + 1, \pi_2(posSeg(a, s)) &gt;) +</math>     <math>capturadoEst(a, &lt; \pi_1(posSeg(a, s)) - 1, \pi_2(posSeg(a, s)) &gt;) +</math>     <math>capturadoEst(a, &lt; \pi_1(posSeg(a, s)), \pi_2(posSeg(a, s)) + 1 &gt;) +</math>     <math>capturadoEst(a, &lt; \pi_1(posSeg(a, s)), \pi_2(posSeg(a, s)) - 1 &gt;) +</math>     <math>\#sanciones(a, s)</math></code>
<code>#hippies(<math>a</math>)</code>	$\equiv$ <code><math>\#(posHippies(a, conjPos(campus(a))))</math></code>
<code>#estudiantes(<math>a</math>)</code>	$\equiv$ <code><math>\#(posEsts(a, conjPos(campus(a))))</math></code>

masVigilante(a)	≡ dameUno(masCapturas(a, seguridad(a)))
masCapturas(a,segs)	≡ <b>if</b> $\neg(\emptyset?(segs))$ <b>then</b> <b>if</b> #capturas(a, dameUno(segs)) ≥ maxCapturas(a, segs) <b>then</b> ag(masCapturas(a, sinUno(segs)), dameUno(segs)) <b>else</b> masCapturas(a, sinUno(segs)) <b>fi</b> <b>else</b> $\emptyset$ <b>fi</b>
maxCapturas(a,segs)	≡ <b>if</b> $\emptyset?(segs)$ <b>then</b> 0 <b>else</b> <b>if</b> #capturas(a, dameUno(segs)) ≥ maxCapturas(a, sinUno(segs)) <b>then</b> #capturas(a, dameUno(segs)) <b>else</b> maxCapturas(a, sinUno(segs)) <b>fi</b> <b>fi</b>
moverTodos(a,segs)	≡ <b>if</b> $\emptyset?(segs)$ <b>then</b> $\emptyset$ <b>else</b> <b>if</b> (hayHippies?(a)) <b>then</b> Ag(moverTodos(a, sinUno(segs)), moverSeg(a, dameUno(segs), dameUno(proxPoss(hippiesMasCerca(a, dameUno(segs)))))) <b>else</b> moverIngreso(a, segs) <b>fi</b> <b>fi</b>

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)	≡	<b>if</b> ( <i>distMan</i> ( <i>campus</i> ( <i>a</i> ), $\pi_2(seg)$ , <i>nPos</i> ) $\geq 2$ $\vee \neg(posValidaAS(a, nPos))$ ) <b>then</b> <i>seg</i> <b>else</b> <b>if</b> # <i>sanciones</i> ( <i>a</i> , <i>seg</i> ) < 3 <b>then</b> < $\pi_1(seg)$ , <i>nPos</i> > <b>else</b> <i>seg</i> <b>fi</b> <b>fi</b>
haySeg?(a,p,segs)	≡	<b>if</b> $\emptyset?(segs)$ <b>then</b> <i>False</i> <b>else</b> <b>if</b> $\pi_2(dameUno(segs)) == p$ <b>then</b> <i>True</i> <b>else</b> <i>haySeg?(a, p, sinUno(segs))</i> <b>fi</b> <b>fi</b>
proxPossHippies(a,possHippies)	≡	<b>if</b> $\emptyset?(possHippies)$ <b>then</b> $\emptyset$ <b>else</b> <i>proxPoss(a, estsCerca(dameUno(possHippies), dameUno(possHippies)</i> <i><math>\cup proxPossHippies(a, sinUno(possHippies))</math></i> <b>fi</b>
hippiesMasCerca(a,seg)	≡	<i>minDistsPos(campus(a), <math>\pi_2(seg)</math>, <i>posHippies(a, conjPos(campus(a)))</i>)</i>
estsCerca(a,p)	≡	<i>minDistsPos(campus(a), p, <i>posEsts(a, conjPos(campus(a)))</i>)</i>
posHippies(a,conjpos)	≡	<b>if</b> $\emptyset?(conjpos)$ <b>then</b> $\emptyset$ <b>else</b> <b>if</b> <i>hayHippie?(a, dameUno(conjpos))</i> <b>then</b> <i>Ag(posHippies(a, sinUno(conjpos)), dameUno(conjpos))</i> <b>else</b> <i>posHippies(a, sinUno(conjpos))</i> <b>fi</b> <b>fi</b>
posEsts(a,conjpos)	≡	<b>if</b> $\emptyset?(conjpos)$ <b>then</b> $\emptyset$ <b>else</b> <b>if</b> <i>hayEst?(a, dameUno(conjpos))</i> <b>then</b> <i>Ag(posEsts(a, sinUno(conjpos)), dameUno(conjpos))</i> <b>else</b> <i>posEsts(a, sinUno(conjpos))</i> <b>fi</b> <b>fi</b>

```

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

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

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

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

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

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

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