Tipos abstractos de datos básicos

Algoritmos y Estructuras de Datos II, DC, UBA.

1. TAD Tablero

TAD Tablero

géneros tablero

exporta bool, generadores, $\neg$, $\vee$, $\wedge$, $\Rightarrow$, $\vee_L$, $\wedge_L$, $\Rightarrow_L$

igualdad observacional

$$(\forall t, t' : \text{tablero}) \left(t =_{\text{obs}} t' \iff \begin{pmatrix} (\#casilleros(t) =_{\text{obs}} \#casilleros(t')) \wedge_L \\ (\forall (c, c' : nat)) c, c' \leq \#casilleros(t) \Rightarrow_L \\ (cont(c,t) =_{\text{obs}} cont(c,t') \wedge \\ movsDesdeHasta(c,c',t) =_{\text{obs}} movsDesdeHasta(c,c',t'))) \end{pmatrix} \right)$$

generadores

crearTablero : continente $\times$ continente $\times$ mov $\times$ mov $\longrightarrow$ tablero

agregarCasillero : casillero c $\times$ continente k $\times$ mov m $\times$ mov m' $\times$ tablero t $\longrightarrow$ tablero

$$\left\{\begin{array}{l} c \leq \#\text{casilleros}(t) \wedge_L (k =_{\text{obs}} cont(c,t) \vee ((\forall c' : nat) c' \leq \#\text{casilleros}(t) \Rightarrow_L cont(c',t) \neq k)) \wedge_L \\ m' \notin \text{todosLosMovs}(c,t) \end{array}\right\}$$

conectar : casillero c $\times$ casillero c' $\times$ mov m $\times$ mov m' $\times$ tablero t $\longrightarrow$ tablero

$$\{c, c' \leq \#\text{casilleros}(t) \wedge c \neq c' \wedge_L m \notin \text{todosLosMovs}(c,t) \wedge_L m' \notin \text{todosLosMovs}(c',t)\}$$

agregarFlecha : casillero c $\times$ casillero c' $\times$ mov m $\times$ tablero t $\longrightarrow$ tablero

$$\{c, c' \leq \#\text{casilleros}(t) \wedge_L \text{conectados?}(c,c',t) \wedge m \notin \text{todosLosMovs}(c,t)\}$$

observadores básicos

#casilleros : tablero $\longrightarrow$ nat

cont : casillero c $\times$ tab t $\longrightarrow$ continente $\{c \leq \#\text{casilleros(t)}\}$

movsDesdeHasta : casillero c $\times$ casillero c' $\times$ tab t $\longrightarrow$ conj(mov) $\{c, c' \leq \#\text{casilleros(t)}\}$

otras operaciones

todosLosMovs : casillero c $\times$ tab t $\longrightarrow$ conj(mov) $\{c \leq \#\text{casilleros(t)}\}$

conectados? : casillero c $\times$ casillero c' $\times$ tab t $\longrightarrow$ bool $\{c, c' \leq \#\text{casilleros(t)}\}$

casillConMovMHastaC : casillero c $\times$ movimiento m $\times$ tab t $\longrightarrow$ conj(casillero) $\{c \leq \#\text{casilleros(t)}\}$

casillConMovMHastaCRecursión : casillero c $\times$ movimiento m $\times$ nat n $\times$ tab t $\longrightarrow$ conj(casillero) $\{c \leq \#\text{casilleros(t)}\}$

axiomas $\forall\ t, t'$: tablero

$\#$casilleros(crearTablero(k,k',m,m')) $\equiv$ 2

$\#$casilleros(agregarCasillero(c,k,m,m',t)) $\equiv$ suc($\#$casilleros(t))

$\#$casilleros(conectar(c,c',m,m',t)) $\equiv$ $\#$casilleros(t)

$\#$casilleros(agregarFlecha(c,c',m,t)) $\equiv$ $\#$casilleros(t)

cont(c,crearTablero(k,k',m,m')) $\equiv$ **if** $c = 1$ **then** k **else** k' **fi**

cont(c,agregarCasillero$(\tilde{c},k,m,m',t)$) $\equiv$ **if** $c =$ suc($\#$casilleros(t)) **then** k **else** cont(c,t) **fi**

cont(c,conectar$(\tilde{c},\tilde{c}',m,m',t)$) $\equiv$ cont(c,t)

cont(c,agregarFlecha$(\tilde{c},\tilde{c}',m,t)$) $\equiv$ cont(c,t)

movsDesdeHasta(c,c',crearTablero(k,k',m,m')) $\equiv$ **if** $c = 1 \wedge c' = 2$ **then**
  $\{m\}$
else
  if $c = 2 \wedge c' = 1$ **then** $\{m'\}$ **else** $\emptyset$ **fi**
fi

$$
\begin{array}{l}
\text{movsDesdeHasta}(c, c', \text{agregarCasillero}(\tilde{c}, k, m, m', t)) \equiv \textbf{if } c = \text{suc}(\#\text{casilleros}(t)) \textbf{ then} \\
\quad \textbf{if } c' = \tilde{c} \textbf{ then } \{m\} \textbf{ else } \emptyset \textbf{ fi} \\
\textbf{else} \\
\quad \textbf{if } c = \tilde{c} \wedge c' = \text{suc}(\#\text{casilleros}(t)) \textbf{ then} \\
\quad\quad \{m'\} \\
\quad \textbf{else} \\
\quad\quad \text{movsDesdeHasta}(c, c', t) \\
\quad \textbf{fi} \\
\textbf{fi}
\end{array}
$$

$$
\begin{array}{l}
\text{movsDesdeHasta}(c, c', \text{conectar}(\tilde{c}, \tilde{c}', m, m', t)) \equiv \textbf{if } c = \tilde{c} \wedge c' = \tilde{c}' \textbf{ then} \\
\quad \{m\} \cup \text{movsDesdeHasta}(c, c', t) \\
\textbf{else} \\
\quad \textbf{if } c = \tilde{c}' \wedge c' = \tilde{c} \textbf{ then} \\
\quad\quad \{m'\} \cup \text{movsDesdeHasta}(c, c', t) \\
\quad \textbf{else} \\
\quad\quad \text{movsDesdeHasta}(c, c', t) \\
\quad \textbf{fi} \\
\textbf{fi}
\end{array}
$$

$$
\begin{array}{l}
\text{movsDesdeHasta}(c, c', \text{agregarFlecha}(\tilde{c}, \tilde{c}', m, t)) \equiv \textbf{if } c = \tilde{c} \wedge c' = \tilde{c}' \textbf{ then} \\
\quad \{m\} \cup \text{movsDesdeHasta}(c, c', t) \\
\textbf{else} \\
\quad \text{movsDesdeHasta}(c, c', t) \\
\textbf{fi}
\end{array}
$$

$$\text{todosLosMovs}(c,\text{crearTablero}(k, k', m, m')) \equiv \textbf{if } c = 1 \textbf{ then } \{m\} \textbf{ else } \{m'\} \textbf{ fi}$$

$$
\begin{array}{l}
\text{todosLosMovs}(c,\text{agregarCasillero}(\tilde{c}, k, m, m', t)) \equiv \textbf{if } c = \text{suc}(\#\text{casilleros}(t)) \textbf{ then} \\
\quad \{m\} \\
\textbf{else} \\
\quad \textbf{if } c = \tilde{c} \textbf{ then} \\
\quad\quad \{m'\} \cup \text{todosLosMovs}(c, t) \\
\quad \textbf{else} \\
\quad\quad \text{todosLosMovs}(c, t) \\
\quad \textbf{fi} \\
\textbf{fi}
\end{array}
$$

$$
\begin{array}{l}
\text{todosLosMovs}(c,\text{conectar}(\tilde{c}, \tilde{c}', m, m', t)) \equiv \textbf{if } c = \tilde{c} \textbf{ then} \\
\quad \{m\} \cup \text{todosLosMovs}(c, t) \\
\textbf{else} \\
\quad \textbf{if } c = \tilde{c}' \textbf{ then} \\
\quad\quad \{m'\} \cup \text{todosLosMovs}(c, t) \\
\quad \textbf{else} \\
\quad\quad \text{todosLosMovs}(c, t) \\
\quad \textbf{fi} \\
\textbf{fi}
\end{array}
$$

$$
\begin{array}{l}
\text{todosLosMovs}(c,\text{agregarFlecha}(\tilde{c}, \tilde{c}', m, t)) \equiv \textbf{if } c = \tilde{c} \textbf{ then} \\
\quad \{m\} \cup \text{todosLosMovs}(c, t) \\
\textbf{else} \\
\quad \text{todosLosMovs}(c, t) \\
\textbf{fi}
\end{array}
$$

$$\text{conectados?}(c, c', t) \equiv \neg\emptyset?(\text{movsDesdeHasta}(c, c', t))$$

$$\text{casillConMovMHastaC}(m, c, t) \equiv \text{casillConMovMHastaCRecursión}(m, c,\#\text{casilleros}(t),t)$$

$$
\begin{array}{l}
\text{casillConMovMHastaCRecursión}(m, c, n, t) \equiv \textbf{if } m \in \text{movsDesdeHasta}(n, c, t) \textbf{ then} \\
\quad \{n\} \cup \text{movsDesdeHasta}(n - 1, c, t) \\
\textbf{else} \\
\quad \text{movsDesdeHasta}(n - 1, c, t) \\
\textbf{fi}
\end{array}
$$

Fin TAD

2. TAD Partida

TAD Partida

géneros partida

exporta nat, generadores, observadores, $+$, $-$, $\times$, $<$, $\leq$, mín, máx

usa Bool

igualdad observacional

$$(\forall p, p' : \text{partida}) \left(p =_{\text{obs}} p' \iff \begin{pmatrix} (\text{tablero}(p) =_{\text{obs}} \text{tablero}(p') \wedge \\ \#\text{jugadores}(p) =_{\text{obs}} \#\text{jugadores}(p') \wedge \\ ((\forall c : nat) c \leq \#\text{casilleros}(\text{tablero}(p)) \Rightarrow_{\text{L}} \\ \text{fichasEnCasillero}(c, p) =_{\text{obs}} \text{fichasEnCasillero}(c, p')) \wedge \\ ((\forall j : nat) j \leq \#\text{jugadores}(p) \Rightarrow_{\text{L}} \\ (\text{misión}(j, p) =_{\text{obs}} \text{misión}(j, p)) \wedge \\ \text{fichasPuestas}(j, p) =_{\text{obs}} \text{fichasPuestas}(j, p')) \end{pmatrix} \right)$$

observadores básicos

tablero : partida $\longrightarrow$ tablero

#jugadores : partida $\longrightarrow$ nat

fichasEnCasillero : casillero c $\times$ partida p $\longrightarrow$ multiconj(jugador) $\{c \leq \#\text{casilleros}(\text{tablero}(p))\}$

misión : jugador j $\times$ partida p $\longrightarrow$ continente $\{j \leq \#\text{jugadores}(p)\}$

fichasPuestas : jugador j $\times$ partida p $\longrightarrow$ nat $\{j \leq \#\text{jugadores}(p)\}$

generadores

crearPartida : tablero t $\times$ nat js $\times$ secu(casillero) cs $\times$ secu(continente) ks $\longrightarrow$ partida

$$\left\{ \begin{array}{l} 2 \leq js \wedge \text{long}(cs) = \text{long}(ks) = js \wedge \text{sinRepetidos}(cs) \wedge ((\forall k : ks) k \in \text{dameConts}(t)) \wedge \\ ((\forall c : cs) c \leq \#\text{casilleros}(t)) \end{array} \right\}$$

agregarFicha : jugador j $\times$ casillero c $\times$ partida p $\longrightarrow$ partida

$$\left\{ \begin{array}{l} j \leq \#\text{jugadores}(p) \wedge_{\text{L}} \text{estáActivo?}(j, p) \wedge \neg\text{terminada?}(p) \wedge c \leq \#\text{casilleros}(\text{tablero}(p)) \wedge_{\text{L}} \\ ((\forall j' : nat)(1 \leq j' \leq \#\text{jugadores}(p) \wedge j' \neq j) \Rightarrow_{\text{L}} \text{fichasEnCasillero}(j, c, p) = 0) \end{array} \right\}$$

mover : jugador j $\times$ movimiento m $\times$ nat n $\times$ partida p $\longrightarrow$ partida $\{j \leq \#\text{jugadores}(p) \wedge \text{estáActivo?}(j, p) \wedge \neg\text{terminada?}(p)\}$

otras operaciones

fichasVecinasDeJ : casillero c $\times$ jugador j $\times$ movimiento m $\times$ partida p $\longrightarrow$ multiconj(jugador) $\{c \leq \#\text{casilleros}(\text{tablero}(p))\}$

estáActivo? : jugador j $\times$ partida p $\longrightarrow$ bool $\{j \leq \#\text{jugadores}(p)\}$

jugadoresActivos : partida $\longrightarrow$ conj(jugador)

jugadoresEliminados : partida $\longrightarrow$ conj(jugador)

tieneFichasEnAlgúnCasillero? : jugador j $\times$ nat n $\times$ partida p $\longrightarrow$ bool $\{j \leq \#\text{jugadores}(p)\}$

jugadoresActivos : partida $\times$ nat $\longrightarrow$ conj(jugador)

terminada? : partida $\longrightarrow$ bool

algunoCompletóLaMisión? : nat $\times$ partida $\longrightarrow$ bool

completóLaMisión? : jugador j $\times$ partida p $\longrightarrow$ bool $\{j \leq \#\text{jugadores}(p)\}$

cuántosLeFaltan : jugador j $\times$ partida p $\longrightarrow$ nat $\{j \leq \#\text{jugadores}(p)\}$

contarNoDominadosHasta : jugador j $\times$ nat $\times$ partida p $\longrightarrow$ nat $\{j \leq \#\text{jugadores}(p)\}$

estáDominado? : casillero c $\times$ partida p $\longrightarrow$ bool $\{c \leq \#\text{casilleros}(\text{tablero}(p))\}$

jugadoresEnCasillero : casillero c $\times$ partida p $\longrightarrow$ conj(jugador) $\{c \leq \#\text{casilleros}(\text{tablero}(p))\}$

dominadoPor : jugador j × casillero c × partida p ⟶ bool {$j \leq$ #jugadores(p) $\wedge$ $c \leq$ #casilleros(tablero(p))}

ganador : partida ⟶ jugador {terminada?(p)}

maxiFourcade : nat × partida ⟶ jugador {terminada?(p)}

buscarElGanador : nat × partida ⟶ jugador {terminada?(p)}

axiomas ∀ p: partida

tablero(crearPartida(t, js, cs, ks)) ≡ t

tablero(agregarFicha(j, c, p)) ≡ tablero(p)

tablero(mover(j, m, n, p)) ≡ tablero(p)

#jugadores(crearPartida(t, js, cs, ks)) ≡ js

#jugadores(agregarFicha(j, c, p)) ≡ #jugadores(p)

#jugadores(mover(j, m, n, p)) ≡ #jugadores(p)

fichasEnCasillero(c, crearPartida(t, js, cs, ks)) ≡ **if** está?(c, cs) **then** {suc(posición(c, cs))} **else** ∅ **fi**

fichasEnCasillero(c, agregarFicha($j, \tilde{c}, p$)) ≡ fichasEnCasillero(c, p) ∪ (**if** $c = \tilde{c}$ **then** {j} **else** ∅ **fi**)

fichasEnCasillero(c, mover(j, m, n, p)) ≡ **if** dominado?(c, p) **then**
  if j ∈ fichasEnCasillero(c,p) ∧ m ∈ todosLosMovs(c,tablero(p))
  then
    (fichasEnCasillero(c, p) - agNVeces(j, n, ∅)) ∪
    fichasVecinasDeJ(c, j, m, p)
  else
    fichasEnCasillero(c, p) ∪ fichasVecinasDeJ(c, j, m, p)
  fi
else
  if ¬∅?(fichasEnCasillero(c, p)) **then**
    sinUno(fichasEnCasillero(c, p)) ∪ fichasVecinasDeJ(c, j, m, p)
  else
    fichasEnCasillero(c, p) ∪ fichasVecinasDeJ(c, j, m, p)
  fi
fi

fichasVecinasDeJ(c, j, m, p) ≡ **if** **then** **else** **fi**

misión(j,crearPartida(t, js, cs, ks)) ≡ $ks[j - 1]$

misión(j,agregarFicha(j, c, p)) ≡ misión(j, p)

misión(j,mover(j, m, n, p)) ≡ misión(j, p)

fichasPuestas(j,crearPartida(t, js, cs, ks)) ≡ 1

fichasPuestas(j,agregarFicha($\tilde{j}, c, p$)) ≡ fichasPuestas(j, p) + (**if** $j = \tilde{j}$ **then** 1 **else** 0 **fi**)

fichasPuestas(j,mover(t, js, cs, ks)) ≡ fichasPuestas(j, p)

estáActivo?(j) ≡ tieneFichasEnAlgúnCasillero(j,#casilleros(tablero(p)),p)

tieneFichasEnAlgúnCasillero(j, n, p) ≡ **if** $n = 0$ **then**
  false
else
  $0 <$ fichasEnCasillero(j, n, p)∨
  tieneFichasEnAlgúnCasillero($j, n - 1, p$)
fi

jugadoresActivos(p) ≡ losActivos(p,#jugadores(p))

losActivos(p, n) ≡ **if** $n = 0$ **then**
  ∅
else
  if estáActivo?(n, p) **then** Ag(n,losActivos($p, n - 1$)) **else** losActivos($p, n - 1$) **fi**
fi

jugadoresEliminados(p) $\equiv$ losEliminados(p,#jugadores(p))

losEliminados(p, n) $\equiv$ **if** $n = 0$ **then**
$\emptyset$
else
if $\neg$ estáActivo?(n, p) **then**
Ag(n,losEliminados($p, n-1$))
else
losEliminados($p, n-1$)
fi
fi

terminada?(p) $\equiv$ #(jugadoresActivos(p))= 1 $\vee$ algunoCompletóLaMisión?(#jugadores(p),p)

algunoCompletóLaMisión?(n, p) $\equiv$ **if** $n = 0$ **then**
false
else
completóLaMisión?(n, p) $\vee$ algunoCompletóLaMisión?($n-1, p$)
fi

completóLaMisión?(j, p) $\equiv$ cuántosLeFaltan(j, p) = 0

cuántosLeFaltan(j, p) $\equiv$ contarNoDominadosHasta(j,#casilleros(tablero(p)),p)

contarNoDominadosHasta(j, n, p) $\equiv$ **if** $\neg$ dominadoPor(j, n, p) $\wedge$ cont(n,tablero(p)) = misión(j, p) **then**
suc(contarNoDominadosHasta($j, n-1, p$))
else
contarNoDominadosHasta($j, n-1, p$)
fi

dominadoPor(j, n, p) $\equiv$ estáDominado?(c, p) $\wedge j \in$ jugadoresEnCasillero(c, p)

estáDominado?(c, p) $\equiv$ #(jugadoresEnCasillero(c, p)) = 1

jugadoresEnCasillero(c, p) $\equiv$ aConj(fichasEnCasillero(c, p))

ganador(p) $\equiv$ maxiFourcade(#jugadores(p),p)

//La función maxiFourcade te devuelve al más ganador de todos. Si no lo conocés, googlealo.

maxiFourcade(n, p) $\equiv$ **if** $n = 1$ *or* completóLaMisión?(n, p) **then** n **else** maxiFourcade($n-1, p$) **fi**

Fin TAD

3. TAD Tupla(α_1, ..., α_n)

TAD Tupla(α_1, ..., α_n)

igualdad observacional
$(\forall t, t' : \text{tupla}(\alpha_1, \ldots, \alpha_n))\ (t =_{\text{obs}} t' \iff (\Pi_1(t) =_{\text{obs}} \Pi_1(t') \wedge \ldots \wedge \Pi_n(t) =_{\text{obs}} \Pi_n(t')))$

parámetros formales
géneros α_1, ..., α_n

géneros tupla($\alpha_1, \ldots, \alpha_n$)

exporta tupla, generadores, observadores

observadores básicos

Π_1 : tupla($\alpha_1, \ldots, \alpha_n$) $\longrightarrow \alpha_1$

$\vdots$

Π_n : tupla$(\alpha_1, \ldots, \alpha_n)$ $\longrightarrow$ α_n

generadores

$\langle \bullet, \ldots, \bullet \rangle$: $\alpha_1 \times \ldots \times \alpha_n$ $\longrightarrow$ tupla$(\alpha_1, \ldots, \alpha_n)$

axiomas $\forall\ a_1$: $\alpha_1 \ldots \forall\ a_n$: α_n

$\Pi_1(\langle a_1, \ldots, a_n \rangle) \equiv a_1$

$\vdots \equiv \vdots$

$\Pi_n(\langle a_1, \ldots, a_n \rangle) \equiv a_n$

Fin TAD

4. TAD Secuencia(α)

TAD Secuencia(α)

igualdad observacional

$$(\forall s, s' : \text{secu}(\alpha)) \left(s =_{\text{obs}} s' \Longleftrightarrow \begin{pmatrix} \text{vacía?}(s) =_{\text{obs}} \text{vacía?}(s') \wedge_{\text{L}} \\ (\neg\ \text{vacía?}(s) \Rightarrow_{\text{L}} (\text{prim}(s) =_{\text{obs}} \text{prim}(s') \wedge \text{fin}(s) =_{\text{obs}} \\ \text{fin}(s')) \end{pmatrix} \right)$$

parámetros formales

géneros α

géneros secu(α)

exporta secu(α), generadores, observadores, &, $\circ$, ult, com, long, está?

usa Bool, Nat

observadores básicos

vacía? : secu(α) $\longrightarrow$ bool

prim : secu(α) s $\longrightarrow$ α $\{\neg$ vacía?(s)$\}$

fin : secu(α) s $\longrightarrow$ secu(α) $\{\neg$ vacía?(s)$\}$

generadores

<> : $\longrightarrow$ secu(α)

$\bullet \bullet \bullet$: $\alpha \times$ secu(α) $\longrightarrow$ secu(α)

otras operaciones

$\bullet \circ \bullet$: secu(α) $\times\ \alpha$ $\longrightarrow$ secu(α)

$\bullet\ \&\ \bullet$: secu(α) $\times$ secu(α) $\longrightarrow$ secu(α)

ult : secu(α) s $\longrightarrow$ α $\{\neg$ vacía?(s)$\}$

com : secu(α) s $\longrightarrow$ secu(α) $\{\neg$ vacía?(s)$\}$

long : secu(α) $\longrightarrow$ nat

está? : $\alpha \times$ secu(α) $\longrightarrow$ bool

axiomas $\forall\ s, t$: secu(α), $\forall\ e$: α

vacía?(<>) $\equiv$ true

vacía?(e $\bullet$ s) $\equiv$

false

prim(e $\bullet$ s)	$\equiv$	e
fin(e $\bullet$ s)	$\equiv$	s
$s \circ e$	$\equiv$	**if** vacía?(s) **then** e $\bullet$ <> **else** prim(s) $\bullet$ (fin(s) $\circ$ e) **fi**
s & t	$\equiv$	**if** vacía?(s) **then** t **else** prim(s) $\bullet$ (fin(s) & t) **fi**
ult(s)	$\equiv$	**if** vacía?(fin(s)) **then** prim(s) **else** ult(fin(s)) **fi**
com(s)	$\equiv$	**if** vacía?(fin(s)) **then** <> **else** prim(s) $\bullet$ com(fin(s)) **fi**
long(s)	$\equiv$	**if** vacía?(s) **then** 0 **else** 1 + long(fin(s)) **fi**
está?(e, s)	$\equiv$	$\neg$ vacía?(s) $\wedge_{\text{L}}$ (e = prim(s) $\vee$ está?(e, fin(s)))

Fin TAD

5. TAD Conjunto(α)

TAD Conjunto(α)

igualdad observacional

$(\forall c, c' : \text{conj}(\alpha))\ \ (c =_{\text{obs}} c' \iff ((\forall a : \alpha)(a \in c =_{\text{obs}} a \in c')))$

parámetros formales

géneros α

géneros conj(α)

exporta conj(α), generadores, observadores, $\emptyset$?, $\cup$, $\cap$, #, $\bullet - \{\bullet\}$, dameUno, sinUno, $\subseteq$, $\bullet - \bullet$

usa Bool, Nat

observadores básicos

$\bullet \in \bullet$	: $\alpha \times \text{conj}(\alpha)$	$\longrightarrow$ bool	

generadores

$\emptyset$	:	$\longrightarrow$ conj(α)	
Ag	: $\alpha \times \text{conj}(\alpha)$	$\longrightarrow$ conj(α)	

otras operaciones

$\emptyset$?	: conj(α)	$\longrightarrow$ bool	
#	: conj(α)	$\longrightarrow$ nat	
$\bullet - \{\bullet\}$	: conj(α) $\times$ α	$\longrightarrow$ conj(α)	
$\bullet \cup \bullet$	: conj(α) $\times$ conj(α)	$\longrightarrow$ conj(α)	
$\bullet \cap \bullet$	: conj(α) $\times$ conj(α)	$\longrightarrow$ conj(α)	
dameUno	: conj(α) c	$\longrightarrow$ α	$\{\neg\emptyset?(c)\}$
sinUno	: conj(α) c	$\longrightarrow$ conj(α)	$\{\neg\emptyset?(c)\}$
$\bullet \subseteq \bullet$	: conj(α) $\times$ conj(α)	$\longrightarrow$ bool	
$\bullet - \bullet$	: conj(α) $\times$ conj(α)	$\longrightarrow$ conj(α)	

axiomas $\forall$ c, d: conj(α), $\forall$ a, b: α

$a \in \emptyset$	$\equiv$	false
$a \in$ Ag(b, c)	$\equiv$	$(a = b) \vee (a \in c)$
$\emptyset?(\emptyset)$	$\equiv$	true

$$
\begin{array}{lll}
\emptyset?(\mathrm{Ag}(b,\ c)) & \equiv & \text{false} \\
\#(\emptyset) & \equiv & 0 \\
\#(\mathrm{Ag}(a,\ c)) & \equiv & 1 + \#(c \text{ - } \{\ a\ \}\ ) \\
c - \{a\} & \equiv & c - \mathrm{Ag}(a,\ \emptyset) \\
\emptyset \cup c & \equiv & c \\
\mathrm{Ag}(a,\ c) \cup d & \equiv & \mathrm{Ag}(a,\ c \cup d) \\
\emptyset \cap c & \equiv & \emptyset \\
\mathrm{Ag}(a,\ c) \cap d & \equiv & \textbf{if } a \in d \textbf{ then } \mathrm{Ag}(a,\ c \cap d) \textbf{ else } c \cap d \textbf{ fi} \\
\mathrm{dameUno}(c) \in c & \equiv & \text{true} \\
\mathrm{sinUno}(c) & \equiv & c - \{\mathrm{dameUno}(c)\} \\
c \subseteq d & \equiv & c \cap d = c \\
\emptyset - c & \equiv & \emptyset \\
\mathrm{Ag}(a,\ c) - d & \equiv & \textbf{if } a \in d \textbf{ then } c - d \textbf{ else } \mathrm{Ag}(a,\ c - d) \textbf{ fi}
\end{array}
$$

Fin TAD

6. TAD Multiconjunto(α)

TAD Multiconjunto(α)

igualdad observacional

$(\forall c, c' : \mathrm{multiconj}(\alpha))\ \ (c =_{\mathrm{obs}} c' \iff ((\forall a : \alpha)(\#(a, c) =_{\mathrm{obs}} \#(a, c'))))$

parámetros formales

géneros α

géneros multiconj(α)

exporta multiconj(α), generadores, observadores, $\in$, $\emptyset?$, $\#$, $\cup$, $\cap$, $\in$, $\bullet - \{\ \bullet\ \}$, dameUno, sinUno

usa Bool, Nat

observadores básicos

$$
\begin{array}{lcll}
\# & : & \alpha \times \mathrm{multiconj}(\alpha) & \longrightarrow \mathrm{nat}
\end{array}
$$

generadores

$$
\begin{array}{lcll}
\emptyset & : & & \longrightarrow \mathrm{multiconj}(\alpha) \\
\mathrm{Ag} & : & \alpha \times \mathrm{multiconj}(\alpha) & \longrightarrow \mathrm{multiconj}(\alpha)
\end{array}
$$

otras operaciones

$$
\begin{array}{lclll}
\bullet \in \bullet & : & \alpha \times \mathrm{multiconj}(\alpha) & \longrightarrow \mathrm{bool} & \\
\emptyset? & : & \mathrm{multiconj}(\alpha) & \longrightarrow \mathrm{bool} & \\
\# & : & \mathrm{multiconj}(\alpha) & \longrightarrow \mathrm{nat} & \\
\bullet - \{\bullet\} & : & \mathrm{multiconj}(\alpha) \times \alpha & \longrightarrow \mathrm{multiconj}(\alpha) & \\
\bullet \cup \bullet & : & \mathrm{multiconj}(\alpha) \times \mathrm{multiconj}(\alpha) & \longrightarrow \mathrm{multiconj}(\alpha) & \\
\bullet \cap \bullet & : & \mathrm{multiconj}(\alpha) \times \mathrm{multiconj}(\alpha) & \longrightarrow \mathrm{multiconj}(\alpha) & \\
\mathrm{dameUno} & : & \mathrm{multiconj}(\alpha)\ c & \longrightarrow \alpha & \{\neg\emptyset?(c)\}
\end{array}
$$

sinUno : multiconj(α) c $\longrightarrow$ multiconj(α) $\{\neg\emptyset?(c)\}$

axiomas $\forall\ c, d$: multiconj(α), $\forall\ a, b$: α

$\#(a,\ \emptyset) \equiv 0$

$\#(a,\ \text{Ag}(b,\ c)) \equiv$ **if** $a = b$ **then** 1 **else** 0 **fi** $+\ \#(a,\ c)$

$a \in c \equiv \#(a,\ c) > 0$

$\emptyset?(\emptyset) \equiv \text{true}$

$\emptyset?(\text{Ag}(a,\ c)) \equiv \text{false}$

$\#(\emptyset) \equiv 0$

$\#(\text{Ag}(a,\ c)) \equiv 1 + \#(c)$

$\emptyset - \{a\} \equiv \emptyset$

$\text{Ag}(a,\ c) - \{b\} \equiv$ **if** $a = b$ **then** c **else** $\text{Ag}(a,\ c - \{b\})$ **fi**

$\emptyset \cup c \equiv c$

$\text{Ag}(a,\ c) \cup d \equiv \text{Ag}(a,\ c \cup d)$

$\emptyset \cap c \equiv \emptyset$

$\text{Ag}(a,\ c) \cap d \equiv$ **if** $a \in d$ **then** $\text{Ag}(a,\ c \cap (d - \{a\}))$ **else** $c \cap d$ **fi**

$\text{dameUno}(c) \in c \equiv \text{true}$

$\text{sinUno}(c) \equiv c - \{\text{dameUno}(c)\}$

Fin TAD

7. TAD Arreglo dimensionable(α)

TAD Arreglo dimensionable(α)

igualdad observacional

$$(\forall a, a' : \text{ad}(\alpha))\ \left(a =_{\text{obs}} a' \iff \begin{pmatrix} \text{tam}(a) =_{\text{obs}} \text{tam}(a') \land \\ (\forall n : \text{nat})(\text{definido?}(a, n) =_{\text{obs}} \text{definido?}(a', n) \land \\ (\text{definido?}(a, n) \ \Rightarrow\ a[n] =_{\text{obs}} a'[n])) \end{pmatrix}\right)$$

parámetros formales

géneros α

géneros ad(α)

exporta ad(α), generadores, observadores

usa Bool, Nat

observadores básicos

tam : ad(α) $\longrightarrow$ nat

definido? : ad(α) $\times$ nat $\longrightarrow$ bool

$\bullet\,[\,\bullet\,]$: ad(α) a $\times$ nat n $\longrightarrow$ α $\{\text{definido?}(a,\ n)\}$

generadores

crearArreglo : nat $\longrightarrow$ ad(α)

$\bullet\,[\,\bullet\,] \leftarrow \bullet$: ad(α) a $\times$ nat n $\times$ α $\longrightarrow$ ad(α) $\{n < \text{tam}(a)\}$

axiomas $\forall\ a$: ad(α), $\forall\ e$: α, $\forall\ n, m$: nat

$\text{tam}(\text{crearArreglo}(n)) \equiv n$

$\text{tam}(a\ [\ n\ ] \leftarrow e)$	$\equiv$	$\text{tam}(a)$
$\text{definido}(\text{crearArreglo}(n),\ m))$	$\equiv$	false
$\text{definido}(a\ [\ n\ ] \leftarrow e,\ m)$	$\equiv$	$n = m \lor \text{definido?}(a,\ m)$
$(a\ [\ n\ ] \leftarrow e)\ [\ m\ ]$	$\equiv$	**if** $n = m$ **then** e **else** $a\ [\ m\ ]$ **fi**

Fin TAD

8. TAD Pila(α)

TAD Pila(α)

igualdad observacional

$$(\forall p, p' : \text{pila}(\alpha)) \left(p =_{\text{obs}} p' \iff \begin{pmatrix} \text{vacía?}(p) =_{\text{obs}} \text{vacía?}(p')) \land_{\text{L}} (\neg\ \text{vacía?}(p) \Rightarrow_{\text{L}} \\ (\text{tope}(p) =_{\text{obs}} \text{tope}(p') \land \text{desapilar}(p) =_{\text{obs}} \text{desapilar}(p')) \end{pmatrix} \right)$$

parámetros formales

géneros α

géneros pila(α)

exporta pila(α), generadores, observadores, tamaño

usa Bool, Nat

observadores básicos

vacía?	:	pila(α)	$\longrightarrow$ bool	
tope	:	pila(α) p	$\longrightarrow$ α	$\{\neg\ \text{vacía?}(p)\}$
desapilar	:	pila(α) p	$\longrightarrow$ pila(α)	$\{\neg\ \text{vacía?}(p)\}$

generadores

vacía	:		$\longrightarrow$ pila(α)
apilar	:	$\alpha \times$ pila(α)	$\longrightarrow$ pila(α)

otras operaciones

tamaño	:	pila(α)	$\longrightarrow$ nat

axiomas $\forall\ p$: pila(α), $\forall\ e$: α

vacía?(vacía)	$\equiv$	true
vacía?(apilar(e,p))	$\equiv$	false
tope(apilar(e,p))	$\equiv$	e
desapilar(apilar(e,p))	$\equiv$	p
tamaño(p)	$\equiv$	**if** vacía?(p) **then** 0 **else** 1 + tamaño(desapilar(p)) **fi**

Fin TAD

9. TAD Cola(α)

TAD Cola(α)

igualdad observacional

$$(\forall c, c' : \text{cola}(\alpha)) \left(c =_{\text{obs}} c' \iff \begin{pmatrix} \text{vacía?}(c) =_{\text{obs}} \text{vacía?}(c') \land_{\text{L}} \\ (\neg\text{vacía?}(c) \Rightarrow_{\text{L}} (\text{próximo}(c) =_{\text{obs}} \text{próximo}(c') \land \\ \text{desencolar}(c) =_{\text{obs}} \text{desencolar}(c')) \end{pmatrix} \right)$$

parámetros formales

  géneros α

géneros cola(α)

exporta cola(α), generadores, observadores, tamaño

usa Bool, Nat

observadores básicos

vacía?	:	cola(α)	$\longrightarrow$	bool	
próximo	:	cola(α) c	$\longrightarrow$	α	$\{\neg$ vacía?$(c)\}$
desencolar	:	cola(α) c	$\longrightarrow$	cola(α)	$\{\neg$ vacía?$(c)\}$

generadores

vacía	:		$\longrightarrow$	cola(α)
encolar	:	$\alpha \times$ cola(α)	$\longrightarrow$	cola(α)

otras operaciones

tamaño	:	cola(α)	$\longrightarrow$	nat

axiomas $\forall$ c: cola(α), $\forall$ e: α

vacía?(vacía)	$\equiv$	true
vacía?(encolar(e,c))	$\equiv$	false
próximo(encolar(e,c))	$\equiv$	**if** vacia?(c) **then** e **else** próximo(c) **fi**
desencolar(encolar(e,c))	$\equiv$	**if** vacía?(c) **then** vacía **else** encolar(e, desencolar(c)) **fi**
tamaño(c)	$\equiv$	**if** vacía?(c) **then** 0 **else** 1 + tamaño(desencolar(c)) **fi**

Fin TAD

10. TAD Árbol binario(α)

TAD Árbol binario(α)

igualdad observacional

$$(\forall a, a' : \text{ab}(\alpha)) \left(a =_{\text{obs}} a' \iff \begin{pmatrix} \text{nil?}(a) =_{\text{obs}} \text{nil?}(a') \land_{\text{L}} (\neg\ \text{nil?}(a) \Rightarrow_{\text{L}} (\text{raiz}(a) =_{\text{obs}} \text{raiz}(a') \\ \land\ \text{izq}(a) =_{\text{obs}} \text{izq}(a') \land \text{der}(a) =_{\text{obs}} \text{der}(a')) \end{pmatrix} \right)$$

parámetros formales

  géneros α

géneros ab(α)

exporta ab(α), generadores, observadores, tamaño, altura, tamaño, inorder, preorder, postorder

usa Bool, Nat, Secuencia(α)

observadores básicos

nil?	:	ab(α)	$\longrightarrow$	bool	
raiz	:	ab(α) a	$\longrightarrow$	α	$\{\neg$ nil?$(a)\}$

izq	: ab(α) a	$\longrightarrow$	ab(α)	$\{\neg$ nil?(a)$\}$
der	: ab(α) a	$\longrightarrow$	ab(α)	$\{\neg$ nil?(a)$\}$

generadores

nil	:	$\longrightarrow$	ab(α)
bin	: ab(α) $\times$ α $\times$ ab(α)	$\longrightarrow$	ab(α)

otras operaciones

altura	: ab(α)	$\longrightarrow$	nat
tamaño	: ab(α)	$\longrightarrow$	nat
inorder	: ab(α)	$\longrightarrow$	secu(α)
preorder	: ab(α)	$\longrightarrow$	secu(α)
postorder	: ab(α)	$\longrightarrow$	secu(α)

axiomas $\forall\ a, b$: ab(α), $\forall\ e$: α

nil?(nil)	$\equiv$ true
nil?(bin(a,e,b))	$\equiv$ false
raiz(bin(a,e,b))	$\equiv$ e
izq(bin(a,e,b))	$\equiv$ a
der(bin(a,e,b))	$\equiv$ b
altura(a)	$\equiv$ **if** nil?(a) **then** 0 **else** 1 + máx(altura(izq(a)), altura(der(a))) **fi**
tamaño(a)	$\equiv$ **if** nil?(a) **then** 0 **else** 1 + tamaño(izq(a)) + tamaño(der(a)) **fi**
inorder(a)	$\equiv$ **if** nil?(a) **then** <> **else** inorder(izq(a)) & (raiz(a) $\bullet$ inorder(der(a))) **fi**
preorder(a)	$\equiv$ **if** nil?(a) **then** <> **else** (raiz(a) $\bullet$ preorder(izq(a))) & preorder(der(a)) **fi**
postorder(a)	$\equiv$ **if** nil?(a) **then** <> **else** postorder(izq(a)) & (postorder(der(a)) $\circ$ raiz(a)) **fi**

Fin TAD

11. TAD Diccionario(clave, significado)

TAD Diccionario(clave, significado)

igualdad observacional

$$(\forall d, d' : \mathrm{dicc}(\kappa, \sigma)) \left(d =_{\mathrm{obs}} d' \iff \begin{pmatrix} (\forall c : \kappa)(\mathrm{def?}(c, d) =_{\mathrm{obs}} \mathrm{def?}(c, d') \wedge_{\mathrm{L}} \\ (\mathrm{def?}(c, d) \Rightarrow_{\mathrm{L}} \mathrm{obtener}(c, d) =_{\mathrm{obs}} \mathrm{obtener}(c, d'))) \end{pmatrix} \right)$$

parámetros formales

géneros clave, significado

géneros dicc(clave, significado)

exporta dicc(clave, significado), generadores, observadores, borrar, claves

usa Bool, Nat, Conjunto(clave)

observadores básicos

def?	: clave $\times$ dicc(clave, significado)	$\longrightarrow$	bool	
obtener	: clave c $\times$ dicc(clave, significado) d	$\longrightarrow$	significado	$\{$def?(c, d)$\}$

generadores

vacío : $\longrightarrow$ dicc(clave, significado)

definir : clave × significado × dicc(clave, significado) $\longrightarrow$ dicc(clave, significado)

otras operaciones

borrar : clave c × dicc(clave, significado) d $\longrightarrow$ dicc(clave, significado) {def?(c,d)}

claves : dicc(clave, significado) $\longrightarrow$ conj(clave)

axiomas $\forall$ d: dicc(clave, significado), $\forall$ c, k: clave, $\forall$ s: significado

def?(c,vacío) $\equiv$ false

def?(c, definir(k, s, d) $\equiv$ $c = k \lor$ def?(c, d)

obtener(c, definir(k, s, d)) $\equiv$ **if** $c = k$ **then** s **else** obtener(c, d) **fi**

borrar(c, definir(k, s, d)) $\equiv$ **if** $c = k$ **then**
  if def?(c,d) **then** borrar(c,d) **else** d **fi**
else
  definir(k, s, borrar(c, d))
fi

claves(vacío) $\equiv$ $\emptyset$

claves(definir(c,s,d)) $\equiv$ Ag(c, claves(d))

Fin TAD

12. TAD Cola de prioridad(α)

TAD Cola de prioridad(α)

igualdad observacional

$$(\forall c, c' : \text{colaPrior}(\alpha)) \left(c =_{\text{obs}} c' \Longleftrightarrow \left(\begin{array}{l} \text{vacía?}(c) =_{\text{obs}} \text{vacía?}(c') \land_{\text{L}} \\ (\neg\text{vacía?}(c) \Rightarrow_{\text{L}} (\text{próximo}(c) =_{\text{obs}} \text{próximo}(c') \land \\ \qquad \text{desencolar}(c) =_{\text{obs}} \text{desencolar}(c')) \end{array} \right) \right)$$

parámetros formales

géneros α

operaciones $\bullet < \bullet$: $\alpha \times \alpha$ $\longrightarrow$ bool — Relación de orden total estricto[1]

géneros colaPrior(α)

exporta colaPrior(α), generadores, observadores

usa Bool

observadores básicos

vacía? : colaPrior(α) $\longrightarrow$ bool

próximo : colaPrior(α) c $\longrightarrow$ α {$\neg$ vacía?(c)}

desencolar : colaPrior(α) c $\longrightarrow$ colaPrior(α) {$\neg$ vacía?(c)}

generadores

[1] Una relación es un orden total estricto cuando se cumple:

Antirreflexividad: $\neg\, a < a$ para todo $a : \alpha$

Antisimetría: $(a < b \Rightarrow \neg\, b < a)$ para todo $a, b : \alpha$, $a \neq b$

Transitividad: $((a < b \land b < c) \Rightarrow a < c)$ para todo $a, b, c : \alpha$

Totalidad: $(a < b \lor b < a)$ para todo $a, b : \alpha$

vacía	:		$\longrightarrow$ colaPrior(α)
encolar	:	α × colaPrior(α)	$\longrightarrow$ colaPrior(α)

axiomas $\forall$ c: colaPrior(α), $\forall$ e: α

vacía?(vacía) $\equiv$ true

vacía?(encolar(e, c)) $\equiv$ false

próximo(encolar(e, c)) $\equiv$ **if** vacía?(c) $\vee_{\text{L}}$ proximo(c) $< e$ **then** e **else** próximo(c) **fi**

desencolar(encolar(e, c)) $\equiv$ **if** vacía?(c) $\vee_{\text{L}}$ proximo(c) $< e$ **then** c **else** encolar(e, desencolar(c)) **fi**

Fin TAD