Tipos abstractos de datos básicos

Algoritmos y Estructuras de Datos II, DC, UBA.

1. TAD Tablero

Casillero es Nat

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 $\{c, c' \leq \#\text{casilleros(t)}\}$

continentes : tablero $\longrightarrow$ conj(continente)

dameContinentes : nat $\times$ tablero $\longrightarrow$ conj(continente)

axiomas $\forall$ 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

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

$\text{movsDesdeHasta}(c, c', \text{conectar}(\tilde{c}, \tilde{c}', m, m', t)) \equiv$ **if** $c = \tilde{c} \wedge c' = \tilde{c}'$ **then**
  $\{m\} \cup \text{movsDesdeHasta}(c, c', t)$
else
  if $c = \tilde{c}' \wedge c' = \tilde{c}$ **then**
    $\{m'\} \cup \text{movsDesdeHasta}(c, c', t)$
  else
    $\text{movsDesdeHasta}(c, c', t)$
  fi
fi

$\text{movsDesdeHasta}(c, c', \text{agregarFlecha}(\tilde{c}, \tilde{c}', m, t)) \equiv$ **if** $c = \tilde{c} \wedge c' = \tilde{c}'$ **then**
  $\{m\} \cup \text{movsDesdeHasta}(c, c', t)$
else
  $\text{movsDesdeHasta}(c, c', t)$
fi

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

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

$\text{todosLosMovs}(c, \text{conectar}(\tilde{c}, \tilde{c}', m, m', t)) \equiv$ **if** $c = \tilde{c}$ **then**
  $\{m\} \cup \text{todosLosMovs}(c, t)$
else
  if $c = \tilde{c}'$ **then**
    $\{m'\} \cup \text{todosLosMovs}(c, t)$
  else
    $\text{todosLosMovs}(c, t)$
  fi
fi

$\text{todosLosMovs}(c, \text{agregarFlecha}(\tilde{c}, \tilde{c}', m, t)) \equiv$ **if** $c = \tilde{c}$ **then**
  $\{m\} \cup \text{todosLosMovs}(c, t)$
else
  $\text{todosLosMovs}(c, t)$
fi

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

$\text{continentes}(t) \equiv \text{dameContinentes}(\#\text{casilleros}(t), t)$

$\text{dameContinentes}(n, t) \equiv$ **if** $n = 0$ **then** $\emptyset$ **else** $\{\text{cont}(n, t)\} \cup \text{dameContinentes}(n-1, t)$ **fi**

Fin TAD

2. TAD Partida

TAD Partida

géneros partida

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

usa Bool

igualdad observacional

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

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

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

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

Funciones requeridas por la empresa

jugadoresActivos : partida $\longrightarrow$ conj(jugador)

jugadoresEliminados : partida $\longrightarrow$ conj(jugador)

terminada? : partida $\longrightarrow$ bool

ganador : partida $\longrightarrow$ jugador $\{\text{terminada?}(p)\}$

casillerosDominados : partida $\longrightarrow$ conj(tupla(casillero,conj(jugador)))

casillerosDisputados : partida $\longrightarrow$ conj(tupla(casillero,conj(jugador)))

casillerosVacíos : partida $\longrightarrow$ conj(casillero)

Funciones auxiliares

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 $\{j \leq \#\text{jugadores}(p)\}$

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

dameActivos : partida $\times$ nat $\longrightarrow$ conj(jugador) dameEliminados : partida × nat ⟶ conj(jugador)

algunoCompletóLaMisión? : nat × partida ⟶ bool

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

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

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

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

estáDisputado? : casillero c × partida p ⟶ bool $\{c \leq \#\text{casilleros}(\text{tablero}(p))\}$

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

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

dameDominados : nat × partida ⟶ conj(casillero)

dameDisputados : nat × partida ⟶ conj(casillero)

dameVacíos : nat × partida ⟶ conj(casillero)

maxiFourcade : nat × partida ⟶ jugador

axiomas $\forall$ p: partida

Observadores

tablero(crearPartida(t, js, cs, ks)) $\equiv$ t

tablero(agregarFicha(j, c, p)) $\equiv$ tablero(p)

tablero(mover(j, m, n, p)) $\equiv$ tablero(p)

#jugadores(crearPartida(t, js, cs, ks)) $\equiv$ js

#jugadores(agregarFicha(j, c, p)) $\equiv$ #jugadores(p)

#jugadores(mover(j, m, n, p)) $\equiv$ #jugadores(p)

fichasEnCasillero(c, crearPartida(t, js, cs, ks)) $\equiv$ **if** está?(c, cs) **then** {suc(posición(c, cs))} **else** $\emptyset$ **fi**

fichasEnCasillero(c, agregarFicha($j, \tilde{c}, p$)) $\equiv$ fichasEnCasillero(c, p) $\cup$ (**if** $c = \tilde{c}$ **then** $\{j\}$ **else** $\emptyset$ **fi**)

fichasEnCasillero(c, mover(j, m, n, p)) $\equiv$ **if** estáDominado?(c, p) **then**
if j $\in$ fichasEnCasillero(c,p) $\wedge$ m $\in$ todosLosMovs(c,tablero(p))
then
(fichasEnCasillero(c, p) - agNVeces($j, n, \emptyset$)) $\cup$
fichasVecinasDeJ(c, j, m, p)
else
fichasEnCasillero(c, p) $\cup$ fichasVecinasDeJ(c, j, m, p)
fi
else
if estáDisputado?(fichasEnCasillero(c, p)) **then**
sinUno(fichasEnCasillero(c, p)) $\cup$ fichasVecinasDeJ(c, j, m, p)
else
fichasEnCasillero(c, p) $\cup$ fichasVecinasDeJ(c, j, m, p)
fi
fi

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

misión(j,agregarFicha(j, c, p)) $\equiv$ misión(j, p)

misión(j,mover(j, m, n, p)) $\equiv$ misión(j, p)

fichasPuestas(j,crearPartida(t, js, cs, ks)) $\equiv$ 1

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

fichasPuestas(j,mover(t, js, cs, ks)) $\equiv$ fichasPuestas(j, p)

Funciones requeridas por la empresa

jugadoresActivos(p) $\equiv$ dameActivos(p,#jugadores(p))

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

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

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

casillerosDominados(p) $\equiv$ dameDominados(#casilleros(tablero(p)),p)

casillerosDisputados(p) $\equiv$ dameDisputados(#casilleros(tablero(p)),p)

casillerosVacíos(p) $\equiv$ dameVacíos(#casilleros(tablero(p)),p)

Funciones auxiliares

fichasVecinasDeJ(j, cn, m, c, n, p) $\equiv$ **if** $0 < cn$ **then**
if dominadoPor(j, cn, p) $\wedge$ m $\in$ movsDesdeHasta(cn, c,tablero(p)) **then**
agNVeces(j,minimo(#(j,fichasEnCasillero(cn,tablero(p))), n), $\emptyset$) $\cup$ fichasVecinasDeJ($j, cn - 1, m, c, n, p$)
else
fichasVecinasDeJ($j, cn - 1, m, c, n, p$)
fi
else
$\emptyset$
fi

estáActivo?(j) $\equiv$ tieneFichasEnAlgúnCasillero(j,#casilleros(tablero(p)),p)

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

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

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

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

estáDisputado?(c, p) $\equiv$ #(jugadoresEnCasillero(c, p)) > 1

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

dameDominados(n, p) $\equiv$ **if** $n = 0$ **then**
$\emptyset$
else
if estáDominado?(n,p) **then**
$\{\langle n,$ jugadoresEnCasillero$(n, p)\rangle\}\cup$ dameDominados$(n-1, p)$
else
dameDominados$(n-1, p)$
fi
fi

dameDisputados(n, p) $\equiv$ **if** $n = 0$ **then**
$\emptyset$
else
if estáDisputado?(n,p) **then**
$\{\langle n,$ jugadoresEnCasillero$(n, p)\rangle\}\cup$ dameDisputados$(n-1, p)$
else
dameDisputados$(n-1, p)$
fi
fi

dameVacíos(n, p) $\equiv$ **if** $n = 0$ **then**
$\emptyset$
else
if $\emptyset$?(jugadoresEnCasillero(n, p)) **then**
$\{n\}\cup$ dameVacíos$(n-1, p)$
else
dameVacíos$(n-1, p)$
fi
fi

//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. Funciones auxiliares de otros TADs

agNVeces : $\alpha \times$ nat $\times$ multiconj(α) $\longrightarrow$ multiconj(α)

agNVeces(a, n, c) $\equiv$ **if** $n = 0$ **then** c **else** Ag(a,agNVeces$(a, n-1, c)$) **fi**

$\bullet[\bullet]$: secu(α) $\times$ nat $\longrightarrow$ α $\{n <$ long$(s)\}$

s[n] $\equiv$ **if** $n = 0$ **then** prim(s) **else** fin$(s)[n-1]$ **fi**

aConj : multiconj(α) $\longrightarrow$ conj(α)

aConj(c) $\equiv$ **if** $\emptyset?(c)$ **then** $\emptyset$ **else** Ag(dameUno(c),aConj(sinUno(c))) **fi**

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

sinRepetidos?(s) $\equiv$ **if** vacía?(s) **then** true **else** $\neg$ está?(prim(s),fin(s)) $\wedge$ sinRepetidos?(fin(s)) **fi**

4. TAD Tupla(α_1, ..., α_n)

TAD Tupla(α_1, ..., α_n)

igualdad observacional

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

parámetros formales

géneros α_1, ..., α_n

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

exporta tupla, generadores, observadores

observadores básicos

Π_1 : tupla($\alpha_1, \ldots, \alpha_n$) $\longrightarrow$ α_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: α_1 ... $\forall$ 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

5. TAD Secuencia(α)

TAD Secuencia(α)

igualdad observacional

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

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) $\times$ α $\longrightarrow$ 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) $\wedge_L$ (e = prim(s) $\vee$ está?(e, fin(s)))

Fin TAD

6. TAD Conjunto(α)

TAD Conjunto(α)

igualdad observacional

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

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

$\bullet \in \bullet$: $\alpha \times$ conj(α) $\longrightarrow$ bool

generadores

$\emptyset$: $\longrightarrow$ conj(α)

Ag : $\alpha \times$ conj(α) $\longrightarrow$ conj(α)

otras operaciones

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

axiomas $\forall\ c, d$: $\text{conj}(\alpha)$, $\forall\ a, b$: α

$a \in \emptyset$	$\equiv$	false
$a \in \text{Ag}(b,\ c)$	$\equiv$	$(a = b) \vee (a \in c)$
$\emptyset?(\emptyset)$	$\equiv$	true
$\emptyset?(\text{Ag}(b,\ c))$	$\equiv$	false
$\#(\emptyset)$	$\equiv$	0
$\#(\text{Ag}(a,\ c))$	$\equiv$	$1 + \#(c - \{\ a\ \}\ )$
$c - \{a\}$	$\equiv$	$c - \text{Ag}(a,\ \emptyset)$
$\emptyset \cup c$	$\equiv$	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** $\text{Ag}(a,\ c \cap d)$ **else** $c \cap d$ **fi**
$\text{dameUno}(c) \in c$	$\equiv$	true
$\text{sinUno}(c)$	$\equiv$	$c - \{\text{dameUno}(c)\}$
$c \subseteq d$	$\equiv$	$c \cap d = c$
$\emptyset - c$	$\equiv$	$\emptyset$
$\text{Ag}(a,\ c) - d$	$\equiv$	**if** $a \in d$ **then** $c - d$ **else** $\text{Ag}(a,\ c - d)$ **fi**

Fin TAD

7. TAD Multiconjunto(α)

TAD Multiconjunto(α)

igualdad observacional

$$(\forall c, c' : \text{multiconj}(\alpha))\ \left(c =_{\text{obs}} c' \Longleftrightarrow ((\forall a : \alpha)(\#(a, c) =_{\text{obs}} \#(a, c')))\right)$$

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

$\#$: $\alpha \times$ multiconj(α) $\longrightarrow$ nat

generadores

$\emptyset$: $\longrightarrow$ multiconj(α)

Ag : $\alpha \times$ multiconj(α) $\longrightarrow$ multiconj(α)

otras operaciones

$\bullet \in \bullet$: $\alpha \times$ multiconj(α) $\longrightarrow$ bool

$\emptyset?$: multiconj(α) $\longrightarrow$ bool

$\#$: multiconj(α) $\longrightarrow$ nat

$\bullet - \{\bullet\}$: multiconj(α) $\times\ \alpha$ $\longrightarrow$ multiconj(α)

$\bullet \cup \bullet$: multiconj(α) $\times$ multiconj(α) $\longrightarrow$ multiconj(α)

$\bullet \cap \bullet$: multiconj(α) $\times$ multiconj(α) $\longrightarrow$ multiconj(α)

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

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

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

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

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

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

$\emptyset?(\emptyset) \equiv$ true

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

$\#(\emptyset) \equiv 0$

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

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

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

$\emptyset \cup c \equiv c$

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

$\emptyset \cap c \equiv \emptyset$

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

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

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

Fin TAD

8. TAD Arreglo dimensionable(α)

TAD Arreglo dimensionable(α)

igualdad observacional

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

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 {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 < tam(a)}

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

tam(crearArreglo(n)) $\equiv$ n

tam(a [n] $\leftarrow$ e) $\equiv$ tam(a)

definido(crearArreglo(n), m)) $\equiv$ false

definido(a [n] $\leftarrow$ e, m) $\equiv$ $n = m$ $\vee$ definido?(a, m)

(a [n] $\leftarrow$ e) [m] $\equiv$ **if** $n = m$ **then** e **else** a [m] **fi**

Fin TAD

9. TAD Pila(α)

TAD Pila(α)

igualdad observacional

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

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 {$\neg$ vacía?(p)}

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

generadores

vacía : $\longrightarrow$ pila(α)

apilar : α $\times$ 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

10. 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

11. TAD Árbol binario(α)

TAD Árbol binario(α)

igualdad observacional

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

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

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

12. TAD Diccionario(clave, significado)

TAD Diccionario(clave, significado)

igualdad observacional

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

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 × dicc(clave, significado)	⟶ bool	
obtener	: clave c × dicc(clave, significado) d	⟶ significado	{def?(c, d)}

generadores

vacío	:	⟶ dicc(clave, significado)
definir	: clave × significado × dicc(clave, significado)	⟶ dicc(clave, significado)

otras operaciones

borrar	: clave c × dicc(clave, significado) d	⟶ dicc(clave, significado)	{def?(c,d)}
claves	: dicc(clave, significado)	⟶ conj(clave)	

axiomas ∀ d: dicc(clave, significado), ∀ c, k: clave, ∀ s: significado

def?(c,vacío) ≡ false

def?(c, definir(k, s, d) ≡ $c = k \vee$ def?(c, d)

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

borrar(c, definir(k, s, d)) ≡ **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) ≡ ∅

claves(definir(c,s,d)) ≡ Ag(c, claves(d))

Fin TAD

13. TAD Cola de prioridad(α)

TAD Cola de prioridad(α)

igualdad observacional

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

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

vacía	:	$\longrightarrow$	colaPrior(α)
encolar	: $\alpha \times$ 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) $\vee_L$ proximo(c) $< e$ **then** e **else** próximo(c) **fi**
desencolar(encolar(e, c))	$\equiv$	**if** vacía?(c) $\vee_L$ proximo(c) $< e$ **then** c **else** encolar(e, desencolar(c)) **fi**

Fin TAD

[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: $((a < b \wedge b < c) \Rightarrow a < c)$ para todo $a, b, c : \alpha$

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