

CASILLERO es NAT

MOVIMIENTO es NAT

CONTINENTE es STRING

JUGADOR es NAT

## 1. TAD TABLERO

TAD TABLERO

géneros      tablero

exporta      tablero, generadores, observadores, continentes, todosLosMovs

usa          BOOL, NAT, CASILLERO, MOVIMIENTO, CONTINENTE, MULTICONJEXT( $\alpha$ ), CONJ( $\alpha$ )

igualdad observacional

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

generadores

*//crearTablero: crea un tablero con dos casilleros. El 1 es del primer continente, el 2 del segundo; el primer movimiento va desde el primer casillero hasta el segundo, y el segundo movimiento hace lo opuesto.*

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

*//agregarCasillero: grega un casillero del continente k conectado por el movimiento m al casillero c. El movimiento m' conecta c al casillero creado. Las restricciones en k aseguran que se cumpla el agrupamiento de continentes, las restricciones sobre m y m' aseguran que se cumpla la unicidad de movimientos y la necesidad de dos movimientos asegura que se cumpla la simetría.*

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

*//conectar: conecta los casilleros pasados como parámetros con el movimiento m del primero al segundo y m' del segundo al primero.*

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: agrega un movimiento en un solo sentido. Requiere que los casilleros ya estén conectados para que no se rompa la simetría.*

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

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

### otras operaciones

*//todosLosMovs: devuelve un conjunto con todos los movimientos que salen del casillero  $c$  a cualquier casillero del tablero.*

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

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

$\text{continentes} : \text{tablero} \longrightarrow \text{conj}(\text{continente})$

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

**axiomas**  $\forall t: \text{tablero}$

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

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

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

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

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

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

```

**Fin TAD**

## 2. TAD PARTIDA

**TAD PARTIDA**

<b>géneros</b>	partida
<b>exporta</b>	partida, generadores, observadores, jugadoresActivos, jugadoresEliminados, terminada?, ganador, casillerosDominados, casillerosDisputados, casillerosVacíos
<b>usa</b>	BOOL, NAT, CASILLERO, JUGADOR, TABLERO, MOVIMIENTO, CONTINENTE, MULTICONJEXT( $\alpha$ ), CONJ( $\alpha$ ), SECUEXT( $\alpha$ )

**igualdad observacional**

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

**observadores básicos**

tablero : partida  $\rightarrow$  tablero

#jugadores : partida  $\rightarrow$  nat

//fichasEnCasillero: las fichas de cada jugador están representadas por su cardinal en un multiconjunto. Una aparición de  $j$  en fichasEnCasillero representa una ficha de  $j$  en el casillero dado. Esta convención con los multiconjuntos se mantendrá durante toda la especificación para representar las fichas en cada casillero.

fichasEnCasillero : casillero  $c \times$  partida  $p \longrightarrow$  multiconjExt(jugador)  $\{c \leq \#casilleros(tablero(p))\}$   
 misión : jugador  $j \times$  partida  $p \longrightarrow$  continente  $\{j \leq \#jugadores(p)\}$

*//fichasPuestas: devuelve la cantidad de fichas que puso  $j$  a lo largo de todo el partido.*

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

### generadores

*//crearPartida: crea una nueva partida cuyo tablero es  $t$ . La cantidad de jugadores es  $js$  y las secuencias  $cs$  y  $ks$  indican el casillero donde coloca la primera ficha y el continente que cada jugador debe conquistar respectivamente. Al  $i$ -ésimo jugador le corresponde la  $(i-1)$ -ésima posición de cada secuencia, ya que el primer jugador es el 1 y los índices de las secuencias empiezan en 0.*

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

$$\left\{ \begin{array}{l} 2 \leq js \wedge \text{long}(cs) = \text{long}(ks) = js \wedge \text{sinRepetidos}(cs) \wedge ((\forall k : \text{string}) \text{está?}(k, ks) \Rightarrow) \\ k \in \text{continentes}(t)) \wedge ((\forall c : \text{nat}) \text{está?}(c, cs) \Rightarrow c \leq \#casilleros(t)) \end{array} \right\}$$

*//agregarFicha: agrega una ficha del jugador  $j$  en el casillero  $c$ . Requiere que esté vacío o dominado por  $j$ .*

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

$$\left\{ \begin{array}{l} j \leq \#jugadores(p) \wedge_L \text{estáActivo?}(j, p) \wedge \neg \text{terminada?}(p) \wedge c \leq \#casilleros(tablero(p)) \wedge_L \\ \text{jugadoresEnCasillero}(c, p) = \emptyset \vee \text{jugadoresEnCasillero}(c, p) = \{j\} \end{array} \right\}$$

*//mover: el jugador  $j$  realiza la acción de movimiento  $m$  con  $n$  fichas. El funcionamiento se detalla más profundamente en los axiomas.*

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

### 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: dado un casillero  $c$ , devuelve un multiconjunto con todas las fichas de  $j$  de todos los casilleros del tablero que con el movimiento  $m$  podrían llegar a  $c$ .*

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

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

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

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

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

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

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

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

*//contarNoDominadosHasta: esta función sirve de auxiliar para saber si el jugador  $j$  completó o no su misión. Al hacer recursión sobre el parámetro  $n$ , devuelve la cantidad de casilleros numerados entre 1 y  $n$  que pertenecen al continente que  $j$  debe conquistar y no son dominados por  $j$ .*

$\text{contarNoDominadosHasta} : \text{jugador } j \times \text{nat } n \times \text{partida } p \longrightarrow \text{nat} \quad \{j \leq \#\text{jugadores}(p)\}$   
 $\text{estáDominado?} : \text{casillero } c \times \text{partida } p \longrightarrow \text{bool} \quad \{c \leq \#\text{casilleros}(\text{tablero}(p))\}$   
 $\text{estáDisputado?} : \text{casillero } c \times \text{partida } p \longrightarrow \text{bool} \quad \{c \leq \#\text{casilleros}(\text{tablero}(p))\}$   
 $\text{jugadoresEnCasillero} : \text{casillero } c \times \text{partida } p \longrightarrow \text{conj}(\text{jugador}) \quad \{c \leq \#\text{casilleros}(\text{tablero}(p))\}$   
 $\text{dominadoPor} : \text{jugador } j \times \text{casillero } c \times \text{partida } p \longrightarrow \text{bool} \quad \{j \leq \#\text{jugadores}(p) \wedge c \leq \#\text{casilleros}(\text{tablero}(p))\}$   
 $\text{dameDominados} : \text{nat} \times \text{partida} \longrightarrow \text{conj}(\text{casillero})$   
 $\text{dameDisputados} : \text{nat} \times \text{partida} \longrightarrow \text{conj}(\text{casillero})$   
 $\text{dameVacíos} : \text{nat} \times \text{partida} \longrightarrow \text{conj}(\text{casillero})$   
 $\text{maxiFourcade} : \text{nat} \times \text{partida} \longrightarrow \text{jugador}$

**axiomas**  $\forall p: \text{partida}$

### Observadores

$\text{tablero}(\text{crearPartida}(t, js, cs, ks)) \equiv t$   
 $\text{tablero}(\text{agregarFicha}(j, c, p)) \equiv \text{tablero}(p)$   
 $\text{tablero}(\text{mover}(j, m, n, p)) \equiv \text{tablero}(p)$   
 $\#\text{jugadores}(\text{crearPartida}(t, js, cs, ks)) \equiv js$   
 $\#\text{jugadores}(\text{agregarFicha}(j, c, p)) \equiv \#\text{jugadores}(p)$   
 $\#\text{jugadores}(\text{mover}(j, m, n, p)) \equiv \#\text{jugadores}(p)$   
 $\text{fichasEnCasillero}(c, \text{crearPartida}(t, js, cs, ks)) \equiv \text{if } \text{está?}(c, cs) \text{ then } \{\text{suc}(\text{posición}(c, cs))\} \text{ else } \emptyset \text{ fi}$   
 $\text{fichasEnCasillero}(c, \text{agregarFicha}(j, \tilde{c}, p)) \equiv \text{fichasEnCasillero}(c, p) \cup (\text{if } c = \tilde{c} \text{ then } \{j\} \text{ else } \emptyset \text{ fi})$

*//fichasEnCasillero: si el casillero es dominado por el jugador que hizo el movimiento, tendrá  $n$  fichas menos de éste, o 0 en caso de que hubiera menos de  $n$  fichas. Si el casillero estaba disputado, algún jugador tendrá una ficha menos. En cualquier caso se agregan todas las fichas posibles de  $j$  a través de los casilleros dominados por  $j$  que se conecten a  $c$  con el movimiento  $m$ .*

$\text{fichasEnCasillero}(c, \text{mover}(j, m, n, p)) \equiv \text{if } \text{estáDominado?}(c, p) \text{ then}$   
 $\quad \text{if } j \in \text{fichasEnCasillero}(c, p) \wedge m \in \text{todosLosMovs}(c, \text{tablero}(p))$   
 $\quad \text{then}$   
 $\quad \quad (\text{fichasEnCasillero}(c, p) - \text{agNVeces}(j, n, \emptyset)) \cup$   
 $\quad \quad \text{fichasVecinasDeJ}(c, j, m, p)$   
 $\quad \text{else}$   
 $\quad \quad \text{fichasEnCasillero}(c, p) \cup \text{fichasVecinasDeJ}(c, j, m, p)$   
 $\quad \text{fi}$   
 $\text{else}$   
 $\quad \text{if } \text{estáDisputado?}(\text{fichasEnCasillero}(c, p)) \text{ then}$   
 $\quad \quad \text{sinUno}(\text{fichasEnCasillero}(c, p)) \cup \text{fichasVecinasDeJ}(c, j, m, p)$   
 $\quad \text{else}$   
 $\quad \quad \text{fichasVecinasDeJ}(c, j, m, p)$   
 $\quad \text{fi}$   
 $\text{fi}$   
 $\text{misión}(j, \text{crearPartida}(t, js, cs, ks)) \equiv ks[j - 1]$   
 $\text{misión}(j, \text{agregarFicha}(j, c, p)) \equiv \text{misión}(j, p)$   
 $\text{misión}(j, \text{mover}(j, m, n, p)) \equiv \text{misión}(j, p)$   
 $\text{fichasPuestas}(j, \text{crearPartida}(t, js, cs, ks)) \equiv 1$   
 $\text{fichasPuestas}(j, \text{agregarFicha}(\tilde{j}, c, p)) \equiv \text{fichasPuestas}(j, p) + (\text{if } j = \tilde{j} \text{ then } 1 \text{ else } 0 \text{ fi})$

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

### 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**  
          $agNVeces(j, \min(\#(j, fichasEnCasillero(cn, tablero(p))), n), \emptyset)$   
          $\cup fichasVecinasDeJ(j, cn - 1, m, c, n, p)$   
     **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**  
      $0 < \#(j, fichasEnCasillero(n, p)) \vee$   
      $tieneFichasEnAlgúnCasillero(j, n - 1, p)$   
**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( $n, \text{tablero}(p)$ ) = misión( $j, p$ ) then
    suc(contarNoDominadosHasta( $j, n - 1, p$ ))
    else
        contarNoDominadosHasta( $j, n - 1, p$ )
    fi

dominadoPor( $j, n, p$ )  $\equiv$  estáDominado?( $c, p$ )  $\wedge j \in \text{jugadoresEnCasillero}(c, p)$ 
estáDominado?( $c, p$ )  $\equiv$  #( $\text{jugadoresEnCasillero}(c, p)$ ) = 1
estáDisputado?( $c, p$ )  $\equiv$  #( $\text{jugadoresEnCasillero}(c, p)$ ) > 1
jugadoresEnCasillero( $c, p$ )  $\equiv$  aConj(fichasEnCasillero( $c, p$ ))

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

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

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

```

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

```

maxiFourcade( $n, p$ )  $\equiv$  if jugadoresActivos( $p$ ) =  $\{n\} \vee \text{completóLaMisión?}(n, p)$  then
     $n$ 
    else
        maxiFourcade( $n - 1, p$ )
    fi

```

**Fin TAD**

### 3. Extensiones de otros TADs

#### 3.1. TAD SECUEXT extiende SECUENCIA

**TAD SECUEXT**

(...)

otras operaciones

$\bullet[\bullet] : \text{secuExt}(\alpha) \times \text{nat} \longrightarrow \alpha$   $\{n < \text{long}(s)\}$   
 $\text{sinRepetidos?} : \text{secuExt}(\alpha) \longrightarrow \text{bool}$

**axiomas**

$s[n] \equiv \text{if } n = 0 \text{ then } \text{prim}(s) \text{ else } \text{fin}(s)[n - 1] \text{ fi}$   
 $\text{sinRepetidos?}(s) \equiv \text{if } \text{vacía?}(s) \text{ then } \text{true} \text{ else } \neg \text{está?}(\text{prim}(s), \text{fin}(s)) \wedge \text{sinRepetidos?}(\text{fin}(s)) \text{ fi}$

**Fin TAD**

### 3.2. TAD MULTICONJEXT extiende MULTICONJUNTO

**TAD MULTICONJEXT**

(...)

**otras operaciones**

$\text{agNVeces} : \alpha \times \text{nat} \times \text{multiconjExt}(\alpha) \longrightarrow \text{multiconjExt}(\alpha)$   
 $\text{aConj} : \text{multiconjExt}(\alpha) \longrightarrow \text{conj}(\alpha)$

**axiomas**

$\text{agNVeces}(a, n, c) \equiv \text{if } n = 0 \text{ then } c \text{ else } \text{Ag}(a, \text{agNVeces}(a, n - 1, c)) \text{ fi}$   
 $\text{aConj}(c) \equiv \text{if } \emptyset?(c) \text{ then } \emptyset \text{ else } \text{Ag}(\text{dameUno}(c), \text{aConj}(\text{sinUno}(c))) \text{ fi}$

**Fin TAD**