Trabajo Práctico 2: Especificación

"PokemonGOArgentina"

Índice

1. TAD Pokémon

TAD Pokémon es string

Fin TAD

2. TAD Jugador

TAD Jugador es nat

Fin TAD

3. TAD Coordenada

TAD Coordenada

igualdad observacional
$(\forall c, c' : \text{coor})\ (c =_{\text{obs}} c' \iff (latitud(c) =_{\text{obs}} latitud(c') \wedge longitud(c) =_{\text{obs}} longitud(c')))$

géneros coor

exporta coor, generadores, observadores, distEuclidea

usa Nat,Bool

observadores básicos

latitud	: coor	$\longrightarrow$	nat
longitud	: coor	$\longrightarrow$	nat

generadores

crearCoor	: nat $\times$ nat	$\longrightarrow$	coor

otras operaciones

distEuclidea	: coor $\times$ coor	$\longrightarrow$	nat	
coordenadaArriba	: coor	$\longrightarrow$	coor	
coordenadaAbajo	: coocr	$\longrightarrow$	coor	{latitud(c) >0}
coordenadaALaDerecha	: coor	$\longrightarrow$	coor	
coordenadaALaIzquierda	: coorc	$\longrightarrow$	coor	{longitud(c) >0}

axiomas ($\forall$ *la*, *lo*: nat, $\forall$ *c*1, *c*2: coor)

latitud(crearCoor(la, lo)) $\equiv$ la

longitud(crearCoor(la, lo)) $\equiv$ lo

distEuclidea(c1, c2) $\equiv$ **if** $latitud(c1) > latitud(c2)$ **then**
$(latitud(c1) - latitud(c2)) \times (latitud(c1) - latitud(c2))$
else
$(latitud(c2) - latitud(c1)) \times (latitud(c2) - latitud(c1))$
fi + **if** $longitud(c1) > longitud(c2)$ **then**
$(longitud(c1) - longitud(c2)) \times (longitud(c1) - longitud(c2))$
else
$(longitud(c2) - longitud(c1)) \times (longitud(c2) - longitud(c1))$
fi

coordenadaArriba(c) $\equiv$ crearCoor(latitud(c1)+1, longitud(c1))
coordenadaAbajo(c) $\equiv$ crearCoor(latitud(c1)-1, longitud(c1))
coordenadaALaDerecha(c) $\equiv$ crearCoor(latitud(c1), longitud(c1)+1)
coordenadaALaIzquierda(c) $\equiv$ crearCoor(latitud(c1), longitud(c1)-1)

Fin TAD

4. TAD Mapa

TAD Mapa

igualdad observacional

$(\forall m, m' : \text{map})\ (m =_{\text{obs}} m' \iff (coordenadas(m) =_{\text{obs}} coordenadas(m')))$

géneros map

exporta map, generadores, observadores, posExistente,hayCamino

usa Nat, Bool, Coordenada,conj(α)

observadores básicos

coordenadas	: map	$\longrightarrow$	conj(coor)

generadores

crearMapa	:	$\longrightarrow$	map
agregarCoor	: coor $\times$ map	$\longrightarrow$	map

otras operaciones

posExistente	: coor $\times$ map	$\longrightarrow$	bool
hayCamino	: coor c1 $\times$ coor c2 $\times$ map m	$\longrightarrow$	bool {c1 $\in$ coordenadas(m)$\wedge$c2 $\in$ coordenadas(m)}
existeCamino	: coor c1 $\times$ coor c2 $\times$ conj(coor) cs $\times$ map m	$\longrightarrow$	bool {c1 $\in$ coordenadas(m)$\wedge$c2 $\in$ coordenadas(m)$\wedge$ cs $\subseteq$ coordenadas(m)}

axiomas ($\forall$ $c1, c2$: coor, $\forall$ m: map, $\forall$ cs: conj(coor))

coordenadas(crearMapa)	$\equiv$	$\emptyset$
coordenadas(agregarCoor(c1, map))	$\equiv$	Ag(c1, coordenadas(map))
posExistente(c1, map)	$\equiv$	c1 $\in$ coordenadas(m)
hayCamino(c1, c2, map)	$\equiv$	existeCamino(c1,c2,coordenadas(m)-{ c1 },m)

existeCamino(c1, c2, cs, map) $\equiv$

```
if c1 = c2 then
    true
else
    if ∅?(cs) then
        false
    else
        existeCaminoPorArriba(c1,c2,cs,m) ∨
        existeCaminoPorAbajo(c1,c2,cs,m) ∨
        existeCaminoPorDerecha(c1,c2,cs,m) ∨
        existeCaminoPorIzquierda(c1,c2,cs,m)
    fi
fi
```

existeCaminoPorArriba(c1,c2,cs,m)	$\equiv$	coordenadaArriba(c1) $\in$ cs $\wedge_{\text{L}}$ existeCamino (coordenadaArriba(c1),c2,cs-{coordenadaArriba(c1)},m)
existeCaminoPorAbajo(c1,c2,cs,m)	$\equiv$	latitud(c1) $>$0 $\wedge_{\text{L}}$ coordenadaAbajo(c1)$\in$ cs $\wedge_{\text{L}}$ existeCamino(coordenadaAbajo(c1),c2, cs-{coordenadaAbajo(c1)},m)
existeCaminoPorDerecha(c1,c2,cs,m)	$\equiv$	coordenadaALaDerecha(c1)$\in$ cs $\wedge_{\text{L}}$ existeCamino(coordenadaALaDerecha(c1),c2, cs-{coordenadaALaDerecha(c1)},m)

existeCaminoPorIzquierda(c1,c2,cs,m) $\equiv$ longitud(c1) $>0 \wedge_{\text{L}}$ coordenadaALaIzquierda(c1) $\in$ cs $\wedge_{\text{L}}$
existeCamino(coordenadaALaIzquierda(c1),c2,
cs-{coordenadaALaIzquierda(c1)},m)

Fin TAD

5. TAD Juego

TAD Juego

igualdad observacional

$$(\forall j, j' : \text{juego}) \left(j =_{\text{obs}} j' \iff \left(\begin{array}{l} \text{mapa(j)} =_{\text{obs}} \text{mapa}(j') \wedge \text{jugadores(j)} =_{\text{obs}} \text{jugadores}(j') \wedge_{\text{L}} \\ (\forall\ \textit{jugador}\text{: e})\ (\ \text{e} \in \text{jugadores(j)} \Rightarrow_{\text{L}} \\ (\text{estaConectado(e,j)} =_{\text{obs}} \text{estaConectado(e,}j') \wedge \\ \text{sanciones(e,j)} =_{\text{obs}} \text{sanciones(e,}j') \wedge \\ \text{pokémons(e,j)} =_{\text{obs}} \text{pokémons(e,}j'))) \wedge_{\text{L}} \\ (\forall\ \textit{jugador}\text{: e})\ (\ \text{e} \in \text{jugadores(j)} \wedge_{\text{L}} \text{estaConectado(e,j)} \Rightarrow_{\text{L}} \\ \text{posicion(e,j)} =_{\text{obs}} \text{posicion(e,j')}) \wedge \\ \text{expulsados(j)} =_{\text{obs}} \text{expulsados(j')} \wedge \\ \text{posConPokémons(j)} =_{\text{obs}} \text{posConPokémons(j')} \wedge_{\text{L}} \\ (\forall\ \textit{coor}\text{: c})\ (\ \text{c} \in \text{posConPokémons(j)} \Rightarrow_{\text{L}} \\ \text{pokémonEnPos(c,j)} =_{\text{obs}} \text{pokémonEnPos(c,j')} \wedge \text{cantMovimientos-} \\ \text{ParaCaptura(c,j)} =_{\text{obs}} \text{cantMovimientosParaCaptura(c,j')}) \end{array} \right) \right)$$

géneros juego

exporta juego, generadores, observadores, puedoAgregarPokémon, hayPokémonCercano, posPókemonCercano, entrenadoresPosibles, indiceRareza, cantPokémonsTotales, cantMismaEspecie

usa Nat, Bool, Pokémon, Jugador, Coordenada, Mapa, conj(α), multiconj(α)

observadores básicos

mapa	: juego	$\longrightarrow$	map	
jugadores	: juego	$\longrightarrow$	conj(jugador)	
estaConectado	: jugador e $\times$ juego j	$\longrightarrow$	bool	{e $\in$ jugadores(j)}
sanciones	: jugador e $\times$ juego j	$\longrightarrow$	nat	{e $\in$ jugadores(j)}
posicion	: jugador e $\times$ juego j	$\longrightarrow$	coor {e $\in$ jugadores(j) $\wedge_{\text{L}}$ estaConectado(e, j)}	
pokémons	: jugador e $\times$ juego j	$\longrightarrow$	multiconj(pokémon)	{e $\in$ jugadores(j)}
expulsados	: juego	$\longrightarrow$	conj(jugador)	
posConPokémons	: juego	$\longrightarrow$	conj(coor)	
pokémonEnPos	: coor c $\times$ juego j	$\longrightarrow$	pokémon	{c $\in$ posConPokémons(j)}
cantMovimientosParaCaptura	: coor c $\times$ juego j	$\longrightarrow$	nat	{c $\in$ posConPokémons(j)}

generadores

crearJuego	: mapa	$\longrightarrow$	juego	
agregarPokémon	: pokémon p $\times$ coor c $\times$ juego j	$\longrightarrow$	juego	{puedoAgregarPokémon(c,j)}
agregarJugador	: juego j	$\longrightarrow$	juego	
conectarse	: jugador e $\times$ coor c $\times$ juego j	$\longrightarrow$	juego {e $\in$ jugadores(j) $\wedge_{\text{L}}$ $\neg$ estaConectado(e, j) $\wedge$ posExistente(c,mapa(j))}	
desconectarse	: jugador e $\times$ juego j	$\longrightarrow$	juego {e $\in$ jugadores(j) $\wedge_{\text{L}}$ estaConectado(e, j)}	
moverse	: jugador e $\times$ coor c $\times$ juego j	$\longrightarrow$	juego {e $\in$ jugadores(j) $\wedge_{\text{L}}$ estaConectado(e, j) $\wedge$ posExistente(c,mapa(j))}	

otras operaciones

ProxID	: juego	$\longrightarrow$	jugador
jugadoresConectados	: juego	$\longrightarrow$	conj(jugador)

soloLosConectados : conj(jugador) es × juego j ⟶ conj(jugador) {es ⊆ jugadores(j)}
puedoAgregarPokémon : coor c × juego j ⟶ bool
debeSancionarse : jugador e × coor c × juego j ⟶ bool {e ∈ jugadores(j)}
debeExpulsarse : jugador e × coor c × juego j ⟶ bool {e ∈ jugadores(j)}
hayPokémonCercano : coor c × juego j ⟶ bool
posPókemonCercano : coor c × juego j ⟶ coor {hayPokémonCercano(c,j)}
entrenadoresPosibles : coor c × conj(jugador) es × juego j ⟶ conj(jugador)
{hayPokémonCercano(c,j) ∧ es ⊆ jugadoresConectados(j)}
posDePokémonsACapturar : coor c × conj(coor) cs × juego j ⟶ conj(coor)
{cs ⊆ posConPokémons(j)}
seCapturo : coor c × coor c2 × juego j ⟶ bool {c ∈ posConPokémons(j)}
indiceRareza : pokémon p × juego j ⟶ nat {p ∈ todosLosPokémons(j)}
cantPokémonsTotales : juego ⟶ nat
todosLosPokémons : juego ⟶ multiconj(pokémon)
pokemonsSalvajes : conj(coor) cs × juego j ⟶ multiconj(pokémon)
{cs ⊆ posConPokémons(j)}
pokemonsCapturados : conj(jugador) es × juego j ⟶ multiconj(pokémon)
{es ⊆ jugadores(j)}
buscarPokémonsCercanos : coor c × conj(coor) cs × juego j ⟶ conj(coor)
cantMismaEspecie: pokémon × multiconj(pokémon) ⟶ nat

axiomas ∀ m: mapa, ∀ j: juego, ∀ $e, e2$: jugador, ∀ es: conj(jugador), ∀ $c, c2$: coor,∀ cs: conj(coor), ∀ p: Pokémon, ∀ ps: multiconj(Pokémon), ∀ k: nat

mapa(crearJuego(m)) ≡ m
mapa(agregarPokémon(p, c, j)) ≡ mapa(j)
mapa(agregarJugador(j)) ≡ mapa(j)
mapa(conectarse(e, c, j)) ≡ mapa(j)
mapa(desconectarse(e, j)) ≡ mapa(j)
mapa(moverse(e, c, j)) ≡ mapa(j)

jugadores(crearJuego(m)) ≡ ∅
jugadores(agregarPokémon(p, c, j)) ≡ jugadores(j)
jugadores(agregarJugador(j)) ≡ Ag(proxID(j), jugadores(j))
jugadores(conectarse(e, c, j)) ≡ jugadores(j)
jugadores(desconectarse(e, j)) ≡ jugadores(j)
jugadores(moverse(e, c, j)) ≡ **if** debeExpulsarse(e,c,j) **then**
jugadores(j)-{e}
else
jugadores(j)
fi

expulsados(crearJuego(m)) ≡ ∅
expulsados(agregarPokémon(p, c, j)) ≡ expulsados(j)
expulsados(agregarJugador(j)) ≡ expulsados(j)
expulsados(conectarse(e, c, j)) ≡ expulsados(j)

expulsados(desconectarse(e, j)) ≡ expulsados(j)
expulsados(moverse(e, c, j)) ≡ **if** debeExpulsarse(e,c,j) **then**
Ag(e,expulsados(j))
else
expulsados(j)
fi

estaConectado(e, agregarPokémon(p, c, j)) ≡ estaConectado(e, j)
estaConectado(e, agregarJugador(j)) ≡ **if** e = proxID(j) **then** false **else** estaConectado(e, j) **fi**
estaConectado(e, conectarse(e2, c, j)) ≡ **if** e = e2 **then** true **else** estaConectado(e, j) **fi**
estaConectado(e, desconectarse(e2, j)) ≡ **if** e = e2 **then** false **else** estaConectado(e, j) **fi**
estaConectado(e, moverse(e2, c, j)) ≡ estaConectado(e, j)

sanciones(e, agregarPokémon(p, c, j)) ≡ sanciones(e, j)
sanciones(e, agregarJugador(j)) ≡ **if** e = proxID(j) **then** ∅ **else** sanciones(e,j) **fi**
sanciones(e, conectarse(e2, c, j)) ≡ sanciones(e, j)
sanciones(e, desconectarse(e2, j)) ≡ sanciones(e, j)
sanciones(e, moverse(e2, c, j)) ≡ **if** e = e2 **then**
if debeSancionarse(e,c,j) **then** 1 **else** 0 **fi**
else
0
fi + sanciones(e,j)

posicion(e, agregarPokémon(p, c, j)) ≡ posicion(e, j)
posicion(e, agregarJugador(j)) ≡ posicion(e, j)
posicion(e, conectarse(e2, c, j)) ≡ **if** e = e2 **then** c **else** posicion(e, j) **fi**
posicion(e, desconectarse(e2,j)) ≡ posicion(e, j)
posicion(e, moverse(e2, c, j)) ≡ **if** e = e2 **then** c **else** posicion(e, j) **fi**

pokémons(e, agregarPokémon(p, c, j)) ≡ pokémons(e, j)
pokémons(e, agregarJugador(j)) ≡ **if** e = proxID(j) **then** ∅ **else** pokémons(e, j) **fi**
pokémons(e, conectarse(e2, c, j)) ≡ pokémons(e, j)
pokémons(e, desconectarse(e2, j)) ≡ pokémons(e, j)
pokémons(e, moverse(e2,c,j)) ≡ **if** e = e2 **then**
pokémons(e, j)
else
if hayPokémonCercano(posicion(e,j),j) $\wedge_{\text{L}}$
distEuclidea(posPókemonCercano(posicion(e,j),j),c, mapa(j))>4 **then**
if cantMovimientosParaCaptura(posPókemonCercano(posicion(e,j),j),j)>=9
∧ e = dameUno(entrenadoresPosibles(posPókemonCercano(
posicion(e,j),jugadoresConectados(j),j),j)) **then**
Ag(pokémonEnPos(PósPokémonCercano(posicion(e,j),j),j), pokémons(e,j))
else
pokémons(e, j)
fi
else
pokémons(e, j)
fi
fi

posConPokémons(crearJuego(m)) ≡ ∅

posConPokémons(agregarPokémon(p, c, j)) ≡ Ag(c, posConPokémons(j))

posConPokémons(agregarJugador(j)) ≡ posConPokémons(j)

posConPokémons(conectarse(e, c, j)) ≡ posConPokémons(j)

posConPokémons(desconectarse(e, j)) ≡ posConPokémons(j)

posConPokémons(moverse(e, c, j)) ≡ posConPokémons(mapa(j)) - posDePokémonsACapturar(c,posConPokémons(j),j)

pokémonEnPos(c,agregarPokémon(p, c2, j)) ≡ **if** c = c2 **then** p **else** pokémonEnPos(c,j) **fi**

pokémonEnPos(c,agregarJugador(j)) ≡ pokémonEnPos(c, j)

pokémonEnPos(c,conectarse(e, c2, j)) ≡ pokémonEnPos(c, j)

pokémonEnPos(c,desconectarse(e, j)) ≡ pokémonEnPos(c, j)

pokémonEnPos(c,moverse(e, c, j)) ≡ pokémonEnPos(c, j)

cantMovimientosParaCaptura(c,agregarPokémon(p, c2, j)) ≡ **if** c = c2 **then**
0
else
cantMovimientosParaCaptura(c,j)
fi

cantMovimientosParaCaptura(c,agregarJugador(j)) ≡ cantMovimientosParaCaptura(c,j)

cantMovimientosParaCaptura(c,conectarse(e, c2, j)) ≡ **if** hayPokémonCercano(c2,j) $\wedge_{\text{L}}$ posPokémonCercano(c2,j) = c **then**
0
else
cantMovimientosParaCaptura(c,j)
fi

cantMovimientosParaCaptura(c,desconectarse(e, j)) ≡ cantMovimientosParaCaptura(c,j)

cantMovimientosParaCaptura(c,moverse(e, c2, j)) ≡ **if** hayPokémonCercano(c2,j) $\wedge_{\text{L}}$ posPokémonCercano(c2,j) = c **then**
if hayPokémonCercano(posicion(e,j),j) $\wedge_{\text{L}}$ posPokémonCercano(posicion(e,j),j) = c **then**
cantMovimientosParaCaptura(c,j)
else
0
fi
else
cantMovimientosParaCaptura(c,j)+1
fi

proxID(j) ≡ **if** (#jugadores(j) + #expulsados(j)) = 0 **then**
0
else
(#jugadores(j) + #expulsados(j))+1
fi

jugadoresConectados(j) ≡ soloLosConectados(jugadores(j),j)

```
soloLosConectados(es,j) ≡ if ∅?(es) then
                            ∅
                          else
                            if estaConectado(dameUno(es),j) then
                              Ag(dameUno(es),soloLosConectados(sinUno(es),j))
                            else
                              soloLosConectados(sinUno(es),j)
                            fi
                          fi

puedoAgregarPokémon(c,j) ≡ posExistente(c,mapa(j)) ∧ ¬hayPokémonCercano(c,j)

debeSancionarse(e,c,j) ≡ ¬hayCamino(posición(e,j),c,mapa(j)) ∨ distEuclidea(posición(e,j),c,mapa(j))
                         >100

debeExpulsarse(e,c,j) ≡ debeSancionarse(e,c,j) ∧ sanciones(e,j) >= 4

hayPokémonCercano(c,j) ≡ ¬∅?(buscarPókemonsCercano(c,posConPokémons(j),j))

posPokémonCercano(c,j) ≡ dameUno(buscarPókemonsCercano(c,posConPokémons(j),j))

buscarPókemonsCercanos(c,cs,j) ≡ if ∅?(cs) then
                                   ∅
                                 else
                                   if distancia(c, dameUno(cs)) <= 25 then
                                     Ag(dameUno(cs),buscarPókemonsCercano(c,sinUno(cs),j)))
                                   else
                                     buscarPókemonsCercano(c,sinUno(cs),j))
                                   fi
                                 fi

entrenadoresPosibles(c,es,j) ≡ if ∅?(es) then
                                 ∅
                               else
                                 if hayPokémonCercano(posicion(dameUno(es),j),j) ∧_L
                                 posPokémonCercano(posicion(dameUno(es),j) = c ∧
                                 hayCamino(c,posicion(dameUno(es),j),mapa(j))
                                 then
                                   Ag(dameUno(es),entrenadoresPosibles(c,sinUno(es),j) )
                                 else
                                   entrenadoresPosibles(c,sinUno(es),j)
                                 fi
                               fi

posDePokémonsACapturar(c,cs,j) ≡ if ∅?(cs) then
                                   ∅
                                 else
                                   if seCapturo(dameUno(cs),c,j) then
                                     Ag(dameUno(cs), posDePokémonsACapturar(c, sinUno(Cs),j) )
                                   else
                                     posDePokémonsACapturar(c, sinUno(Cs),j))
                                   fi
                                 fi
```

seCapturo(c,c2,j) ≡ **if** hayPokémonCercano(c2,j) **then**
(posPokémonCercano(c2,j) ≠ c ∧ cantMovimientosParaCaptura(c,j) >= 9 ∧
¬∅?entrenadoresPosibles(c,jugadoresConectados(j),j))
else
(cantMovimientosParaCaptura(c,j) >= 9 ∧
¬∅?entrenadoresPosibles(c,jugadoresConectados(j),j))
fi

indiceRareza(p,j) ≡ 100 - (100 × cantMismaEspecie(p, j) / cantPokémonsTotales(j))

cantPokémonsTotales(j) ≡ #todosLosPokémons(j)

todosLosPokémons(j) ≡ pokemonsSalvajes(posConPokémons(j),j) ⋃ pokemonsCapturados(jugadores(j),j)

pokemonsSalvajes(cs,j) ≡ **if** ∅?(cs) **then**
∅
else
Ag(pokémonEnPos(DameUno(cs),j), pokemonsSalvajes(sinUno(cs),j))
fi

pokemonsCapturados(es,j) ≡ **if** ∅?(es) **then**
∅
else
pokémons(DameUno(es),j) ⋃ pokemonsCapturados(sinUno(es),j)
fi

cantMismaEspecie(p,j) ≡ #(p,todosLosPokémons(j))

Fin TAD