capturaPokemon? : jugador j × posiciones ps × sistema s ⟶ bool
{j ∈ jugadoresConectados(jugadores(s), s) ∧ ps ⊆ posiciones(mapa(s))}

pokemonACapturar : jugador j × posiciones ps × sistema s ⟶ bool
{j ∈ jugadoresConectados(jugadores(s), s) ∧ ps ⊆ posiciones(mapa(s))}

jugadoresConectados : conj(jugador) js × sistema s ⟶ conj(jugador)
{js ⊆ jugadores(s)}

pokemonsCerca? : posicion p × posiciones ps × sistema s ⟶ bool
{p ∈ posiciones(mapa(s)) ∧ ps ⊆ posiciones(mapa(s))}

axiomas ∀ m: mapa, ∀ s: sistema, ∀ j, j': jugador, ∀ p, p': posicion, ∀ pk: pokemon, ∀ n, m: nat,
∀ js: conj(jugador), ∀ ps: conj(posicion)

mapa(crearSistema(m)) ≡ m
mapa(registrarJugador(s,j)) ≡ mapa(s)
mapa(conectarJugador(s,j,p)) ≡ mapa(s)
mapa(desconectarJugador(s,j)) ≡ mapa(s)
mapa(moverJugador(s,j,p)) ≡ mapa(s)
mapa(agregarPokemon(s,pk,p)) ≡ mapa(s)
jugadores(crearSistema(m)) ≡ ∅
jugadores(registrarJugador(s,j)) ≡ Ag(j,jugadores(s))
jugadores(conectarJugador(s,j,p)) ≡ jugadores(s)
jugadores(desconectarJugador(s,j)) ≡ jugadores(s)
jugadores(moverJugador(s,j,p)) ≡ jugadores(s)
jugadores(agregarPokemon(s,pk,p)) ≡ jugadores(s)
jugadoresConectados(js, s) ≡ **if** vacio?(js) **then**
∅
else
if ¬eliminado?(dameUno(js)) ∧$_L$ conectado?(dameUno(js)) **then**
Ag(dameUnos(js), jugadoresConectados(sinUno(js), s))
else
jugadoresConectados(sinUno(js), s)
fi
fi
conectado?(j, registrarJugador(s,j')) ≡ **if** j = j' **then** false **else** conectado?(j,s) **fi**
conectado?(j, conectarJugador(s,j',p)) ≡ **if** j = j' **then** true **else** conectado?(j,s) **fi**
conectado?(j, desconectarJugador(s,j')) ≡ **if** j = j' **then** false **else** conectado?(j,s) **fi**
conectado?(j, moverJugador(s,j',p)) ≡ conectado?(j,s)
conectado?(j, agregarPokemon(s,pk,p)) ≡ conectado?(j,s)
posicionJugador(j, registrarJugador(s,j')) ≡ posicionJugador(j,s)
posicionJugador(j, conectarJugador(s,j',p)) ≡ **if** j = j' **then** p **else** posicionJugador(j,s) **fi**
posicionJugador(j, desconectarJugador(s,j')) ≡ posicionJugador(j,s)
posicionJugador(j, moverJugador(s,j',p)) ≡ **if** j = j' **then** p **else** posicionJugador(j,s) **fi**
posicionJugador(j, agregarPokemon(s,pk,p)) ≡ posicionJugador(j,s)
pokemonsCapturados(j, registrarJugador(s,j')) ≡ **if** j = j' **then** ∅ **else** pokemonsCapturados(j,s) **fi**
pokemonsCapturados(j, conectarJugador(s,j',p)) ≡ pokemonsCapturados(j,s)
pokemonsCapturados(j, desconectarJugador(s,j')) ≡ pokemonsCapturados(j,s)
pokemonsCapturados(j, agregarPokemon(s,pk,p)) ≡ pokemonsCapturados(j,s)
pokemonsCapturados(j, moverJugador(s,j',p)) ≡ **if** conectado?(j) ∧$_L$ capturaPokemon?(j, posiciones(mapa(s)), moverJugador(s,j',p)) **then**
Ag(pokemonACapturar(j, posiciones(mapa(s)), moverJugador(s,j',p)), pokemonsCapturados(j, s))
else
pokemonsCapturados(j, s)
fi

X Le tienen que pasar la distancia antes del movimiento (5 en este caso), ya que como es comportamiento automático, una vez que hubo un movimiento tal que el pokémon debería ser capturado por el jugador eso ocurre instantáneamente.

capturaPokemon?(j, ps, s) ≡ **if** vacio?(ps) **then**
false
else
if hayPokemon?(dameUno(ps), s) ∧ distancia(posicionJugador(j, s), dameUno(ps)) ≤ 2 **then**
¬∅?(jugadoresDisponiblesACapturar(jugadoresConectados(jugadores(s), s), dameUno(ps), s)) ∧$_L$
j = dameUno(jugadoresDisponiblesACapturar(jugadoresConectados(jugadores(s), s), dameUno(ps), s))
else
capturaPokemon?(j, sinUno(ps), s)
fi
fi

pokemonACapturar(j, ps, s) ≡ **if** hayPokemon?(dameUno(ps), s) ∧ distancia(posicionJugador(j, s), dameUno(ps)) ≤ 2 **then**
pokemonEnPos(dameUno(ps), s)
else
capturaPokemon?(j, sinUno(ps), s)
fi

jugadoresDisponiblesACapturar(js, p, s) ≡ **if** vacio?(js) **then**
∅
else
if movsLejosDePos(dameUno(js), s) = 10 **then**
Ag(dameUno(js), jugadoresDisponiblesACapturar(sinUno(js, p, s)))
else
jugadoresDisponiblesACapturar(sinUno(js, p, s))
fi
fi

movsLejosDePos(j, p, registrarJugador(s,j')) ≡ movsLejosDePos(j, p, s)
movsLejosDePos(j, p, conectarJugador(s,j',p')) ≡ **if** j = j' **then** 0 **else** movsLejosDePos(j, p, s) **fi**
movsLejosDePos(j, p, desconectarJugador(s,j')) ≡ movsLejosDePos(j, p, s)
movsLejosDePos(j, p, agregarPokemon(s,pk,p')) ≡ **if** p = p' **then** 0 **else** movsLejosDePos(j, p, s) **fi**
movsLejosDePos(j, p, moverJugador(s,j',p')) ≡ **if** distancia(p, posicionJugador(j, s)) ≤ 2 **then**
if j = j' **then**
if distancia(p, p') ≤ 2 **then**
movsLejosDePos(j, p ,s)
else
0
fi
else
if distancia(p, p') ≤ 2 **then**
if distancia(p, posicionJugador(j', s)) ≤ 2 **then**
movsLejosDePos(j, p ,s)
else
0
fi
else
1 + movsLejosDePos(j, p ,s)
fi
fi
else
0
fi

sancionesJugador(j, registrarJugador(s,j')) ≡ **if** j = j' **then** 0 **else** sancionesJugador(j,s) **fi**
sancionesJugador(j, conectarJugador(s,j',p)) ≡ sancionesJugador(j,s)
sancionesJugador(j, desconectarJugador(s,j')) ≡ sancionesJugador(j,s)

✗ Entre hayPokemon?, alguienCapturaPokemon y capturaPokemon? hay una dependencia circular. Esto es un problema porque nunca se reduce la instancia del sistema, así que la evaluación entra en un ciclo infinito que no termina.

✓ sancionesJugador(j, moverJugador(s,j',p)) ≡ **if** j = j' ∧$_L$ (10 ≤ distancia(p, posicionJugador(j,s)) ∨ (¬conexion?(p, posicionJugador(j,s), mapa(s)))) **then**
1 + sancionesJugador(j,s)
else
sancionesJugador(j,s)
fi

sancionesJugador(j, agregarPokemon(s,pk,p)) ≡ sancionesJugador(j,s)

✓ hayPokemon?(p, crearSistema(m)) ≡ false

✓ hayPokemon?(p, registrarJugador(s,j)) ≡ hayPokemon?(p,s)

hayPokemon?(p, conectarJugador(s,j,p')) ≡ hayPokemon?(p,s)

hayPokemon?(p, desconectarJugador(s,j)) ≡ hayPokemon?(p,s)

✗ hayPokemon?(p, moverJugador(s,j,p')) ≡ **if** hayPokemon?(p,s) **then**
¬alguienCapturaPokemon(jugadoresConectados(jugadores(s), s), p, moverJugador(s,j,p'))
else
false
fi
→ debería llamarse con s!

✓ hayPokemon?(p, agregarPokemon(s,pk,p')) ≡ **if** p = p' **then** true **else** hayPokemon?(p,s) **fi**

alguienCapturaPokemon(js, p, s) ≡ **if** vacio?(js) **then**
false
else
if capturaPokemon?(dameUno(js), p, s) **then**
true
else
capturaPokemon?(sinUno(js), p, s)
fi
fi

✓ pokemonEnPos(p, registrarJugador(s,j)) ≡ pokemonEnPos(p,s)

✓ pokemonEnPos(p, conectarJugador(s,j,p')) ≡ pokemonEnPos(p,s)

pokemonEnPos(p, desconectarJugador(s,j)) ≡ pokemonEnPos(p,s)

pokemonEnPos(p, moverJugador(s,j,p')) ≡ pokemonEnPos(p,s)

pokemonEnPos(p, agregarPokemon(s,pk,p')) ≡ **if** p = p' **then** pk **else** pokemonEnPos(p,s) **fi**

pokemonsCerca?(p, ps, s) ≡ **if** vacio?(ps) **then**
false
else
if ¬(p = dameUno(ps)) ∧ distancia(p, dameUno(ps)) < 5 ∧ hayPokemon?(dameUno(ps), s) **then**
true
else
pokemonsCerca?(p, sinUno(ps), s)
fi
fi

✓ eliminado?(j,s) ≡ sancionesJugador(j,s) = 5

✓ rareza(pk, s) ≡ 100 - 100 × (#(pk,todosLosPokemons(s)) ÷ #(todosLosPokemons(s)))

✓ todosLosPokemons(s) ≡ pokemonsEnArea(posiciones(mapa(s)),s) ∪ pokemonsCapturadosPorGrupo(jugadores(s),s)

pokemonsEnArea(ps, s) ≡ **if** vacio?(ps) **then**
∅
else
if hayPokemon?(dameUno(ps), s) **then**
Ag(pokemonEnPos(dameUno(ps), s), pokemonsEnArea(sinUno(ps),s))
else
pokemonsEnArea(sinUno(ps),s)
fi
fi