



TPE - Agricultura con drones

13 de abril de 2016

Algoritmos y Estructuras de Datos I

Grupo 15

Integrante	LU	Correo electrónico
Szperling, Sebastián	763/15	zebaszp@gmail.com
Barylko, Roni	750/15	ronibarylko@hotmail.com
Giudice, Carlos	694/15	Carlosr.giudice@gmail.com
López Segura, Florencia	759/13	fsegura@dc.uba.ar



Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (+54 +11) 4576-3300

<http://www.exactas.uba.ar>

1. Tipos

```
tipo Id =  $\mathbb{Z}$ ;
tipo Carga =  $\mathbb{Z}$ ;
tipo Ancho =  $\mathbb{Z}$ ;
tipo Largo =  $\mathbb{Z}$ ;
tipo Parcela = Cultivo, Granero, Casa;
tipo Producto = Fertilizante, Plaguicida, PlaguicidaBajoConsumo, Herbicida, HerbicidaLargoAlcance;
tipo EstadoCultivo = ReciénSembrado, EnCrecimiento, ListoParaCosechar, ConMaleza, ConPlaga, NoSensado;
```

2. Campo

```
tipo Campo {
  observador dimensiones (c: Campo) : (Ancho, Largo);
  observador contenido (c: Campo, i, j:  $\mathbb{Z}$ ) : Parcela;
  requiere enRango :  $0 \leq i < \text{prm}(\text{dimensiones}(c)) \wedge 0 \leq j < \text{sgd}(\text{dimensiones}(c))$ ;

  invariante dimensionesValidas :  $\text{prm}(\text{dimensiones}(c)) > 0 \wedge \text{sgd}(\text{dimensiones}(c)) > 0$ ;
  invariante unaSolaCasa :  $|\{(i, j) | i \leftarrow [0..\text{prm}(\text{dimensiones}(c))], j \leftarrow [0..\text{sgd}(\text{dimensiones}(c))], \text{contenido}(c, i, j) == \text{Casa}\}| == 1$ ;
  invariante unSoloGranero :  $|\{(i, j) | i \leftarrow [0..\text{prm}(\text{dimensiones}(c))], j \leftarrow [0..\text{sgd}(\text{dimensiones}(c))], \text{contenido}(c, i, j) == \text{Granero}\}| == 1$ ;
  invariante algoDeCultivo :  $|\{(i, j) | i \leftarrow [0..\text{prm}(\text{dimensiones}(c))], j \leftarrow [0..\text{sgd}(\text{dimensiones}(c))], \text{contenido}(c, i, j) == \text{Cultivo}\}| \geq 1$ ;
  invariante posicionesAlcanzables :  $\text{posicionesAlcanzablesEn100}(c)$ ;
}

aux posicionesAlcanzablesEn100 (c: Campo) : Bool =
  alcanzableEn100(posicionGranero(c), prm(dimensiones(c)), sgd(dimensiones(c)));

problema crearC (posG, posC: ( $\mathbb{Z}$ ,  $\mathbb{Z}$ )) = res : Campo {
  requiere puntosValidos :  $0 \leq \text{prm}(\text{posG}) \wedge 0 \leq \text{sgd}(\text{posG}) \wedge 0 \leq \text{prm}(\text{posC}) \wedge 0 \leq \text{sgd}(\text{posC})$ ;
  requiere puntosDistintos :  $\text{prm}(\text{posG}) \neq \text{prm}(\text{posC}) \vee \text{sgd}(\text{posG}) \neq \text{sgd}(\text{posC})$ ;
  requiere aMenosDe100 :  $\text{distancia}(\text{posG}, \text{posC}) \leq 100$ ;
  asegura posicionGranero(res) == posG;
  asegura posicionCasa(res) == posC;
}

problema dimensionesC (c: Campo) = res : (Ancho, Largo) {
  asegura res == dimensiones(c);
}

problema contenidoC (c: Campo, i, j:  $\mathbb{Z}$ ) = res : Parcela {
  requiere enRango(dimensiones(c), i, j);
  asegura res == contenido(c, i, j);
}
```

3. Drone

```

tipo Drone {
  observador id (d: Drone) : Id;
  observador bateria (d: Drone) : Carga;
  observador enVuelo (d: Drone) : Bool;
  observador vueloRealizado (d: Drone) :  $[(\mathbb{Z}, \mathbb{Z})]$ ;
  observador posicionActual (d: Drone) :  $(\mathbb{Z}, \mathbb{Z})$ ;
  observador productosDisponibles (d: Drone) : [Producto];

  invariante vuelosOk :
    enVuelo(d)  $\Rightarrow$  ( $|vueloRealizado(d)| > 0 \wedge posicionActual(d) == vueloRealizado(d)_{|vueloRealizado(d)|-1} \wedge$ 
    posicionesPositivas(d)  $\wedge$  movimientosOK(d))  $\wedge \neg enVuelo(d) \Rightarrow |vueloRealizado(d)| == 0$ ;
  invariante bateriaOk :  $0 \leq bateria(d) \leq 100$ ;
}

aux posicionesPositivas (d: Drone) : Bool = ( $\forall i \leftarrow [0..|vueloRealizado(d)|)$ )  $prm(vueloRealizado(d)_i) \geq 0 \wedge$ 
sgd(vueloRealizado(d)i)  $\geq 0$ ;
aux movimientosOK (d: Drone) : Bool = ( $\forall i \leftarrow [1..|vueloRealizado(d)|)$ )
prm(vueloRealizado(d)i) == prm(vueloRealizado(d)i-1)  $\wedge$  sgd(vueloRealizado(d)i) == sgd(vueloRealizado(d)i-1) - 1  $\vee$ 
sgd(vueloRealizado(d)i) == sgd(vueloRealizado(d)i-1) + 1  $\vee$  sgd(vueloRealizado(d)i) == sgd(vueloRealizado(d)i-1)
 $\wedge$  (prm(vueloRealizado(d)i) == prm(vueloRealizado(d)i-1) - 1  $\vee$  prm(vueloRealizado(d)i-1) == prm(vueloRealizado(d)i-1) + 1);

problema crearD (i:  $\mathbb{Z}$ , ps: [Producto]) = res : Drone {
  asegura bateria(res) == 100  $\wedge$  id(res) == i;
  asegura  $\neg enVuelo$ (res);
  asegura mismos(productosDisponibles(res), ps);
}

problema idD (d: Drone) = res :  $\mathbb{Z}$  {
  asegura res == id(d);
}

problema bateriaD (d: Drone) = res :  $\mathbb{Z}$  {
  asegura res == bateria(d);
}

problema enVueloD (d: Drone) = res : Bool {
  asegura res == enVuelo(d);
}

problema vueloRealizadoD (d: Drone) = res :  $[(\mathbb{Z}, \mathbb{Z})]$  {
  requiere enVuelo(d);
  asegura mismos(res, [vueloRealizado(d)i |  $i \leftarrow [1..|vueloRealizado(d)|]$ ]);
}

problema posicionActualD (d: Drone) = res :  $(\mathbb{Z}, \mathbb{Z})$  {
  asegura res == posicionActual(d);
}

problema productosDisponiblesD (d: Drone) = res : [Producto] {
  asegura res == productosDisponibles(d);
}

problema vueloEscaleradoD (d: Drone) = res : Bool {
  asegura res == enVuelo(d)  $\wedge$  movimientoAlternado(d)  $\wedge$  movimientoUnidireccional(d);
}

problema vuelosCruzadosD (ds: [Drone]) = res :  $[(\mathbb{Z}, \mathbb{Z}), \mathbb{Z}]$  {
  requiere todosEnVuelo : ( $\forall d \in ds$ ) enVuelo(d);
  requiere vuelosDeIgualLargo : ( $\forall i \in [1..|ds|]$ )  $|vueloRealizado(ds_i)| == |vueloRealizado(ds_{i-1})|$ ;
  asegura (ds == []  $\wedge$  res == [])  $\vee$  (mismos(res, [cruce |  $i \leftarrow [0..|vueloRealizado(ds_0)|]$ ,
    cruce  $\leftarrow [(posicion, [|1 | d \leftarrow ds, vueloRealizado(d)_i == posicion])]$  |
    posicion  $\leftarrow posicionesSinRepetir([vueloRealizado(d)_i | d \leftarrow ds])$ ], sgd(cruce) > 1));
  asegura enOrden : ( $\forall i \in [1..|res|]$ ) sgd(resi)  $\geq$  sgd(resi-1)  $\vee$  ( $\forall i \in [1..|res|]$ ) sgd(resi)  $\leq$  sgd(resi-1);
}

```

4. Sistema

```

tipo Sistema {
  observador campo (s: Sistema) : Campo;
  observador estadoDelCultivo (s: Sistema, i, j:  $\mathbb{Z}$ ) : EstadoCultivo;
  requiere enRango(dimensiones(s), i, j)  $\wedge$  contenido(campo(s), i, j) == Cultivo;
  observador enjambreDrones (s: Sistema) : [Drone];

  invariante identificadoresUnicos : sinRepetidos([id(d) | d  $\leftarrow$  enjambreDrones(s)]);
  invariante unoPorParcela : ( $\forall d, d' \leftarrow$  dronesEnVuelo(s), id(d)  $\neq$  id(d')) posicionActual(d)  $\neq$  posicionActual(d');
  invariante siNoVuelanEstanEnGranero : ( $\forall d \leftarrow$  enjambreDrones(s),  $\neg$ enVuelo(d))
    posicionActual(d) == posicionGranero(campo(s));
  invariante siEstanEnVueloElVueloEstaEnRango : ( $\forall d \leftarrow$  dronesEnVuelo(s))( $\forall v \leftarrow$  vueloRealizado(d))
    enRango(dimensiones(campo(s), prm(v), sgd(v)));
}

aux dronesEnVuelo (s: Sistema) : [Drone] = [d | d  $\leftarrow$  enjambreDrones(s), enVuelo(d)];

problema crearS (c: Campo, ds: [Drone]) = res : Sistema {
  asegura dronesEnGranero : ( $\forall d \in ds$ ) posicionActual(d) == posicionGranero(c);
  asegura bateriaLlena : ( $\forall d \in ds$ ) bateria(d) == 100;
  asegura cultivosNoSensados : ( $\forall i \in [0..prm(dimensiones(c))], j \leftarrow [0..sgd(dimensiones(c))])$ 
    contenido(c, i, j) == Cultivo  $\longrightarrow$  estadoDelCultivo(c, i, j) == NoSensado;
  asegura (campo(res) == c);
  asegura mismos(enjambreDrones(res), ds);
}

problema campoS (s: Sistema) = res : Campo {
  asegura res == campo(s);
}

problema estadoDelCultivoS (s: Sistema, i, j:  $\mathbb{Z}$ ) = res : EstadoCultivo {
  requiere enRango(dimensiones(campo(s)), i, j);
  requiere contenido(campo(s), i, j) == Cultivo;
  asegura res == estadoDelCultivo(s, i, j);
}

problema enjambreDronesS (s: Sistema) = res : [Drone] {
  asegura mismos(res, enjambreDrones(s));
}

problema crecerS (s: Sistema) {
  modifica s;
  asegura mismoCampo : campo(pre(s)) == campo(s);
  asegura mismosDrones : mismos(enjambreDrones(pre(s)), enjambreDrones(s));
  asegura crecimientoCultivo : ( $\forall i \in [0..prm(dimensiones(campo(s)))])(\forall j \in [0..sgd(dimensiones(campo(s)))]$ )
    contenido(campo(s), i, j) == Cultivo  $\longrightarrow$ 
    (estadoDelCultivo(pre(s), i, j) == RecienSembrado  $\longrightarrow$  estadoDelCultivo(s, i, j) == EnCrecimiento
     $\wedge$  estadoDelCultivo(pre(s), i, j) == EnCrecimiento  $\longrightarrow$  estadoDelCultivo(s, i, j) == ListoParaCosechar
     $\wedge$  (estadoDelCultivo(pre(s), i, j)  $\neq$  RecienSembrado  $\wedge$  estadoDelCultivo(pre(s), i, j)  $\neq$  EnCrecimiento)
     $\longrightarrow$  estadoDelCultivo(pre(s), i, j) = estadoDelCultivo(s, i, j));
}

problema seVinoLaMalezaS (s: Sistema, ps: [( $\mathbb{Z}$ ,  $\mathbb{Z}$ ))] {
  requiere ( $\forall (i, j) \in ps$ ) enRango(dimensiones(campo(s)), i, j);
  requiere ( $\forall (i, j) \in ps$ ) contenido(campo(s), i, j) == Cultivo;
  modifica s;
  asegura mismoCampo : campo(pre(s)) == campo(s);
  asegura mismosDrones : mismos(enjambreDrones(pre(s)), enjambreDrones(s));
  asegura enmalezado : ( $\forall i \in [0..prm(dimensiones(campo(s)))])(\forall j \in [0..sgd(dimensiones(campo(s)))]$ )
    en((i, j), ps)  $\longrightarrow$  estadoDelCultivo(s, i, j) == ConMaleza
     $\wedge$   $\neg$ (en((i, j), ps))  $\longrightarrow$  estadoDelCultivo(s, i, j) == estadoDelCultivo(pre(s), i, j);
}

problema seExpandePlagaS (s: Sistema) {
  modifica s;
}

```

```

asegura mismoCampo : campo(pre(s)) == campo(s);
asegura mismosDrones : mismos(enjambreDrones(pre(s)), enjambreDrones(s));
asegura expansiónDePlaga : (∀i ∈ [0..prm(dimensiones(campo(s)))])(∀j ∈ [0..sgd(dimensiones(campo(s)))]
    adyacenteAPlaga(s, i, j) → estadoDelCultivo(s, i, j) == ConPlaga
    ∧ ¬(adyacenteAPlaga(s, i, j)) → estadoDelCultivo(s, i, j) == estadoDelCultivo(pre(s), i, j);
}

problema despegarS (s: Sistema, d: Drone) {
    requiere ¬enVuelo(d);
    requiere PosicionActual(d) == PosicionGranero(campo(s));
    requiere en(d, enjambreDrones(s));
    modifica s, d;
    asegura mismoCampo : campo(pre(s)) == campo(s);
    asegura mismoEstadoCultivo : (∀i ∈ [0..prm(dimensiones(campo(s)))])(∀j ∈ [0..sgd(dimensiones(campo(s)))]
        contenido(campo(s), i, j) ≠ Cultivo ∨ estadoDelCultivo(s, i, j) == estadoDelCultivo(pre(s), i, j));
    asegura mismoDron : id(d) == id(pre(d));
    asegura mismaBateria : bateria(d) == bateria(pre(d));
    asegura mismaPosicion : posicionActual(d) == posicionActual(pre(d));
    asegura mismosProductos : mismos(productosDisponibles(d), productosDisponibles(pre(d)));
    asegura despegue : enVuelo(d) ∧ vueloRealizado(d) == [posicionActual];
    asegura mismoEnjambre : mismos(enjambreDrones(s), d : [e | e ← enjambreDrones(pre(s)), e ≠ pre(d)]);
}

problema listoParaCosecharS (s: Sistema) = res : Bool {
    asegura res == ([1 | (i, j) ← cultivos(s), estadoDelCultivo(s, i, j) == ListoParaCosechar] ÷ |cultivos(s)| ≥ 0,9);
}

problema aterrizarYCargarBateriaS (s: Sistema, b: ℤ) {
    requiere 0 < b ≤ 100;
    modifica s;
    asegura mismoCampo : campo(pre(s)) == campo(s);
    asegura mismoEstadoCultivo : (∀i ∈ [0..prm(dimensiones(campo(s)))])(∀j ∈ [0..sgd(dimensiones(campo(s)))]
        contenido(campo(s), i, j) ≠ Cultivo ∨ estadoDelCultivo(s, i, j) == estadoDelCultivo(pre(s), i, j));
    asegura mismosDrones : |enjambreDrones(s)| == |enjambreDrones(pre(s))|;
    asegura cambiosADrones : (∀d ∈ enjambreDrones(pre(s)))
        (bateria(d) ≤ b → ((∃e ∈ enjambreDrones(s))
            id(e) == id(d) ∧ bateria(e) == 100 ∧ ¬enVuelo(e) ∧ posicionActual(e) == posicionGranero(campo(s)) ∧
            vueloRealizado(d) == [ ] ∧ productosDisponibles(e) == productosDisponibles(d))) ∧ (bateria(d) > b →
            en(d, enjambreDrones(s)));
}

problema fertilizarPorFilas (s: Sistema) {
    requiere maxUnDronPorFila : (∀j ∈ [0..sgd(dimension(campo(s)))]
        |[1 | i ← [0..prm(dimension(campo(s)))]], en((i, j), posicionesDronesEnVuelo(s))]| < 2;
    modifica s;
    asegura mismoCampo : campo(pre(s)) == campo(s);
    asegura mismosDrones : |enjambreDrones(s)| == |enjambreDrones(pre(s))|
        ∧ (∀d ∈ enjambreDrones(pre(s)))(∃e ∈ enjambreDrones(s))id(e) == id(d);
    asegura soloCambiarSiEnVuelo : (∀d ∈ enjambreDrones(pre(s)))(∃e ∈ enjambreDrones(s), id(e) == id(d))
        enVuelo(d) == enVuelo(e) ∧ ¬enVuelo(d) → d == e;
    asegura vuelosAlOeste : (∀d ∈ enjambreDrones(pre(s)))(∃e ∈ enjambreDrones(s), id(e) == id(d))
        prm(posicionActual(e)) == prm(posicionActual(d)) - |vueloDiferente(e, d)|;
    asegura gastoDeBateria : (∀d ∈ enjambreDrones(pre(s)))(∃e ∈ enjambreDrones(s), id(e) == id(d))
        bateria(e) == (bateria(d) + |vueloDiferente(e, d)|);
    asegura unFertilizantePorCultivo : (∀d ∈ enjambreDrones(pre(s)))(∃e ∈ enjambreDrones(s), id(e) == id(d))
        mismos(productosDisponibles(d), productosDisponibles(e) + +[Fertilizante | i ← [0..|vueloDiferente(e, d)|]]);
    asegura soloCultivos : (∀d ∈ enjambreDrones(pre(s)))(∃e ∈ enjambreDrones(s), id(e) == id(d))
        (∀(i, j) ∈ vueloDiferente(e, d))contenido(campo(s), i, j) == Cultivo;
    asegura condicionDeFinalizacion : (∀d ∈ enjambreDrones(s))
        bateria(d) == 0 ∨ ¬(en(Fertilizante, productosDisponibles(d))
            ∨ contenido(campo(s), prm(posicionActual(d)), sgd(posicionActual(d))) ≠ Cultivo
            ∨ sgd(posicionActual(d)) == 0;
    asegura fertilizarCampo : (∀d ∈ enjambreDrones(pre(s)))
        (∃e ∈ enjambreDrones(s), id(e) == id(d))(∀(i, j) ∈ vueloDiferente(e, d))

```

```

    ((en(estadoDelCultivo(pre(s), i, j), [RecienSembrado, EnCrecimiento])
    → estadoDelCultivo(s, i, j) == ListoParaCosechar)
    ∧ (¬en(estadoDelCultivo(pre(s), i, j), [RecienSembrado, EnCrecimiento])
    → estadoDelCultivo(s, i, j) == estadoDelCultivo(pre(s), i, j)) ;
}

problema volarYSensarS (s: Sistema, d: Drone) {
    requiere puedeMoverse : bateria(d) > 0 ;
    requiere enVuelo(d) ;
    requiere en(d, enjambreDrones(s)) ;
    modifica s, d ;
    asegura mismoCampo : campo(pre(s)) == campo(s) ;
    asegura mismoEnjambre : mismos(enjambreDrones(s), d : [e | e ← enjambreDrones(pre(s)), e ≠ pre(d)]) ;
    asegura mismoDron : id(d) == id(pre(d)) ;
    asegura aParcelaAdyacente : |vueloDiferente(d, pre(d))| == 1
        ∧ enRango(dimensiones(campo(s)), prm(posicionActual(d)), sgd(posicionActual(d))) ;
    asegura sensado : estadoDelCultivo(pre(s), prm(posicionActual(d)), sgd(posicionActual(d))) == NoSensado →
        (estadoDelCultivo(s, prm(posicionActual(d)), sgd(posicionActual(d))) ≠ NoSensado ∧ noHaceNada(d, pre(d)) ;
    asegura siEstaBienNoHaceNada :
        ¬en(estadoDelCultivo(pre(s), prm(posicionActual(d)), sgd(posicionActual(d))), [ConMaleza, ConPlaga, NoSensado])
        → (noHaceNada(d, pre(d)) ∧ estadoDelCultivo(pre(s), prm(posicionActual(d)), sgd(posicionActual(d))) ==
        == estadoDelCultivo(s, prm(posicionActual(d)), sgd(posicionActual(d)))) ;
    asegura mataMalezas : estadoDelCultivo(pre(s), prm(posicionActual(d)), sgd(posicionActual(d))) == ConMaleza
        → (usaAlgunHerbicida(d, pre(d))
        ∧ estadoDelCultivo(s, prm(posicionActual(d)), sgd(posicionActual(d))) == RecienSembrado)
        ∨ ((¬tieneHerbicida(d) ∨ bateria(pre(d)) < 6)
        ∧ noHaceNada(d, pre(d)) ∧ estadoDelCultivo(s, prm(posicionActual(d)), sgd(posicionActual(d))) == ConMaleza)) ;
    asegura mataPlagas : estadoDelCultivo(pre(s), prm(posicionActual(d)), sgd(posicionActual(d))) == ConPlaga
        → (usaPlaguicidaComun(d, pre(d))
        ∧ estadoDelCultivo(s, prm(posicionActual(d)), sgd(posicionActual(d))) == RecienSembrado)
        ∨ (usaPlaguicidaBajoConsumo(d, pre(d))
        ∧ estadoDelCultivo(s, prm(posicionActual(d)), sgd(posicionActual(d))) == RecienSembrado)
        ∨ ((tienePlaguicida(d) ∧ ((¬tienePlaguicidaBajoConsumo(d) ∧ bateria(d) < 11) ∨ (tienePlaguicidaBajoConsumo(d) ∧
        bateria(d) < 6))) ∨ (¬tienePlaguicida(d) ∧ ¬tienePlaguicidaBajoConsumo(d)) ∧ noHaceNada(d, pre(d))
        ∧ estadoDelCultivo(s, prm(posicionActual(d)), sgd(posicionActual(d))) == ConPlaga)) ;
}

```

5. Funciones Auxiliares

5.1. Campo

```
aux posicionGranero (c: Campo) : ( $\mathbb{Z}$ ,  $\mathbb{Z}$ ) = [(i, j) | i  $\leftarrow$  [0..prm(dimensiones(c))], j  $\leftarrow$  [0..sgd(dimensiones(c))],  
contenido(c, i, j) == Granero]0 ;  
aux posicionCasa (c: Campo) : ( $\mathbb{Z}$ ,  $\mathbb{Z}$ ) = [(i, j) | i  $\leftarrow$  [0..prm(dimensiones(c))], j  $\leftarrow$  [0..sgd(dimensiones(c))],  
contenido(c, i, j) == Casa]0 ;
```

5.2. Drone

```
aux movimientoAlternado (d: Drone) : Bool = ( $\forall d \in [2..|vueloRealizado(d)|]$ )  
prm(vueloRealizado(d)i) == prm(vueloRealizado(d)i-1)  $\Leftrightarrow$  |prm(vueloRealizado(d)i) - prm(vueloRealizado(d)i-2)| == 1  $\wedge$   
sgd(vueloRealizado(d)i) == sgd(vueloRealizado(d)i-1)  $\Leftrightarrow$  |sgd(vueloRealizado(d)i) - sgd(vueloRealizado(d)i-2)| == 1 ;  
aux movimientoUnidireccional (d: Drone) : Bool = ( $\forall d \in [3..|vueloRealizado(d)|]$ )  
|prm(vueloRealizado(d)i) - prm(vueloRealizado(d)i-1)| == 1  $\Leftrightarrow$   
|prm(vueloRealizado(d)i) - prm(vueloRealizado(d)i-3)| == 2  $\wedge$   
|sgd(vueloRealizado(d)i) - sgd(vueloRealizado(d)i-1)| == 1  $\Leftrightarrow$   
|sgd(vueloRealizado(d)i) - sgd(vueloRealizado(d)i-3)| == 2 ;  
aux posicionesSinRepetir (ps: [( $\mathbb{Z}$ ,  $\mathbb{Z}$ )]) : [( $\mathbb{Z}$ ,  $\mathbb{Z}$ )] = [psi | i  $\leftarrow$  [0..|ps|], ( $\forall j \in [0..i)$ ) psi  $\neq$  psj] ;
```

5.3. Sistema

```
aux adyacenteAPlaga (s: Sistema, (i, j):  $\mathbb{Z}$ ) : Bool =  
en(ConPlaga, [estadoDelCultivo(s, i, j) | (i, j)  $\leftarrow$  cultivosAdyacentes((i, j), s)]) ;  
aux cultivosAdyacentes ((i, j): ( $\mathbb{Z}$ ,  $\mathbb{Z}$ ), s: Sistema) : [( $\mathbb{Z}$ ,  $\mathbb{Z}$ )] = [(x, y) | (x, y)  $\leftarrow$  [(i, j + 1), (i + 1, j), (i, j - 1), (i - 1, j)],  
enRango(dimensiones(campo(s)), x, y)  $\wedge$  contenido(campo(s), x, y) == Cultivo] ;  
aux cultivos (s: Sistema) : [( $\mathbb{Z}$ ,  $\mathbb{Z}$ )] = [(i, j) | i  $\leftarrow$  [0..prm(dimensiones(campo(s)))]  
j  $\leftarrow$  [0..sgd(dimensiones(campo(s)))]], contenido(campo(s), i, j) == Cultivo] ;  
aux posicionesDronesEnVuelo (s: Sistema) : [( $\mathbb{Z}$ ,  $\mathbb{Z}$ )] = [posicionActual(d) | d  $\leftarrow$  enjambreDrones(s), enVuelo(d)] ;  
aux vueloDiferente (e, d: Drone) : [( $\mathbb{Z}$ ,  $\mathbb{Z}$ )] = [vueloRealizado(e)i | i  $\leftarrow$  [|vueloRealizado(d)|..|vueloRealizado(e)|]] ;  
aux noHaceNada (e, d: Drone) : Bool = mismos(productosDisponibles(d), productosDisponibles(e))  
 $\vee$  bateria(d) - 1 == bateria(e) ;  
aux usaAlgunHerbicida (e, d: Drone) : Bool = (mismos(productosDisponibles(d), Herbicida : productosDisponibles(e))  $\vee$   
mismos(productosDisponibles(d), HerbicidaLargoAlcance : productosDisponibles(e)))  $\wedge$  bateria(e) == bateria(d) - 6 ;  
aux tieneHerbicida (d: Drone) : Bool = en(Herbicida, productosDisponibles(d))  
 $\vee$  en(HerbicidaLargoAlcance, productosDisponibles(d)) ;  
aux usaPlaguicidaComun (e, d: Drone) : Bool = mismos(productosDisponibles(d), Plaguicida : productosDisponibles(e))  $\wedge$   
bateria(e) == bateria(d) - 11 ;  
aux usaPlaguicidaBajoConsumo (e, d: Drone) : Bool = mismos(productosDisponibles(d), PlaguicidaBajoConsumo :  
productosDisponibles(e))  $\wedge$  bateria(e) == bateria(d) - 6 ;  
aux tienePlaguicida (d: Drone) : Bool = en(Plaguicida, productosDisponibles(d)) ;  
aux tienePlaguicidaBajoConsumo (d: Drone) : Bool = en(PlaguicidaBajoConsumo, productosDisponibles(d)) ;
```