



# TPE - Agricultura con drones

12 de abril de 2016

Algoritmos y Estructuras de Datos I

## Grupo 3

Integrante	LU	Correo electrónico
Sanmartin Sobol, Ruslan	275/14	rus1147@gmail.com
Garrett, Philip	318/14	garrett.phg@gmail.com
Belfiore, Rocío	332/14	rociobelfiore@gmail.com
Tuso, Bernardo	792/14	btuso.95@gmail.com



**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 :  $0 \leq i < prm(dimensiones(c)) \wedge 0 \leq j < sgd(dimensiones(c))$ ;

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

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 posPositiva0(posG)  $\wedge$  posPositiva0(posC);
  requiere posG  $\neq$  posC;
  requiere distancia((0, 0), posG)  $\leq$  100  $\wedge$  distancia(posG, 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 =  $(\forall i \leftarrow [1..|vueloRealizado(d)|])$ 
 $prm(vueloRealizado(d)_i) == prm(vueloRealizado(d)_{i-1}) \wedge (sgd(vueloRealizado(d)_i) == sgd(vueloRealizado(d)_{i-1}) - 1 \vee$ 
 $sgd(vueloRealizado(d)_i) == sgd(vueloRealizado(d)_{i-1}) + 1) \vee sgd(vueloRealizado(d)_i) == sgd(vueloRealizado(d)_{i-1})$ 
 $\wedge (prm(vueloRealizado(d)_i) == prm(vueloRealizado(d)_{i-1}) - 1 \vee prm(vueloRealizado(d)_i) == prm(vueloRealizado(d)_{i-1}) +$ 
 $1)$ ;

problema crearD (id: Id, pd:[Producto]) = res : Drone {
  asegura  $id(res) == id$ ;
  asegura  $bateria(res) == 100$ ;
  asegura  $\neg enVuelo(res)$ ;
  asegura  $mismos(productosDisponibles(res), pd)$ ;
}

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 :  $[(\mathbb{Z}, \mathbb{Z})]$  {
  asegura  $enVuelo(d) \Rightarrow mismos(res, cola(vueloRealizado(d)))$ ;
}

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

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

problema vueloEscaleraD (d: Drone) = res : Bool {
  asegura  $res == |vueloRealizado(d)| \geq 3 \wedge esEscalera(vueloRealizado(d))$ ;
}

aux esEscalera (ps:  $[(\mathbb{Z}, \mathbb{Z})]$ ) : Bool =  $primerosTresEscalera(ps) \wedge (\forall i \leftarrow [0..|ps| - 1]) comoLosPrimerosTres(ps, i)$ ;
aux primerosTresEscalera (ps:  $[(\mathbb{Z}, \mathbb{Z})]$ ) : Bool =  $|prm(restarPos(ps_2, ps_0))| == 1 \wedge |sgd(restarPos(ps_2, ps_0))| == 1$ ;
aux comoLosPrimerosTres (ps:  $[(\mathbb{Z}, \mathbb{Z})]$ , i:  $\mathbb{Z}$ ) : Bool =  $restarPos(ps_{i+1}, ps_i) == restarPos(ps_{(i \bmod 2)+1}, ps_{i \bmod 2})$ ;

```

```

problema vuelosCruzadosD (ds: [Drone]) = res : [(( $\mathbb{Z}$ , $\mathbb{Z}$ ), $\mathbb{Z}$ )] {
  requiere todosEnVuelo : ( $\forall d \leftarrow ds$ ) enVuelo(d);
  requiere igualDeLargos : ( $\forall d1 \leftarrow ds, d2 \leftarrow ds$ ) |vueloRealizado(d1)| == |vueloRealizado(d2)|;
  asegura hayCruce : ( $\forall j \leftarrow [0..|res|]$ ) sgd(resj) > 1;
  asegura estanTodosLosCruces : ( $\forall p \leftarrow posicionesRecorridas(ds), colisionesEnPosicion(ds, p) > 1$ )
    ( $\exists r \leftarrow res$ ) prm(r) = p;
  asegura cantidadDeCruces : ( $\forall j \leftarrow [0..|res|]$ ) sgd(resj) == colisionesEnPosicion(ds, prm(resj));
  asegura crucesSinRepetidos : sinRepetidos([prm(c) | c  $\leftarrow$  res]);
  asegura orden : ordenadoAsc(res)  $\vee$  ordenadoDesc(res);
}

aux colisionesEnPosicion (ds: [Drone], pos: ( $\mathbb{Z}$ ,  $\mathbb{Z}$ )) :  $\mathbb{Z}$  = |[d | d  $\leftarrow$  ds, i | i  $\leftarrow$  [0..|vueloRealizado(d)|],
vueloRealizado(d)i == pos  $\wedge$  colisionesEnInstanteEnPosicion(ds, pos, i) > 1]|;
aux colisionesEnInstanteEnPosicion (ds: [Drone], pos: ( $\mathbb{Z}$ ,  $\mathbb{Z}$ ), i:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  = |[d | d  $\leftarrow$  ds, vueloRealizado(d)i == pos]|;
aux ordenadoAsc (ps: [(( $\mathbb{Z}$ , $\mathbb{Z}$ ), $\mathbb{Z}$ ))] : Bool = ( $\forall p \leftarrow [0..|ps| - 1]$ ) sgd(resp)  $\leq$  sgd(resp+1);
aux ordenadoDesc (ps: [(( $\mathbb{Z}$ , $\mathbb{Z}$ ), $\mathbb{Z}$ ))] : Bool = ( $\forall p \leftarrow [0..|ps| - 1]$ ) sgd(resp)  $\geq$  sgd(resp+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 {
  requiere noHayDronesRepetidos : sinRepetidos([id(d) | d  $\leftarrow$  ds]);
  asegura mismoCampo : campo(res) == c;
  asegura mismaCantidadDeDrones : |ds| == |enjambreDrones(res)|;
  asegura mismoId : ( $\forall d1 \leftarrow$  ds)( $\exists d2 \leftarrow$  enjambreDrones(res)) id(d1) == id(d2);
  asegura dronesCargados : ( $\forall d \leftarrow$  enjambreDrones(res)) bateria(d) == 100;
  asegura estanEnElGranero : ( $\forall d \leftarrow$  enjambreDrones(res)) posicionActual(d) == posicionGranero(campo(s));
  asegura mismosProductos : ( $\forall d1 \leftarrow$  ds, d2  $\leftarrow$  enjambreDrones(res)) id(d1) == id(d2)
     $\Rightarrow$  mismos(productosDisponibles(d1), productosDisponibles(d2));
  asegura parcelasNoSensadas : ( $\forall p \leftarrow$  parcelasConCultivo(campo(s))) estadoDelCultivo(s, prm(p), sgd(p)) ==
    NoSensado;

}

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

}

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

}

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

}

problema crecerS (s: Sistema) {
  modifica s;
  asegura mismoCampo : campo(s) == campo(pre(s));
  asegura mismosDrones : mismosDrones(enjambreDrones(s), enjambreDrones(pre(s)));
  asegura ( $\forall p \leftarrow$  parcelasConCultivo(s)) deRecienSembradoAEnCrecimiento(pre(s), s, p)
     $\vee$  deEnCrecimientoAListoParaCosechar(pre(s), s, p)  $\vee$  mismoEstado(pre(s), s, p);

}

aux deRecienSembradoAEnCrecimiento (preS, s:Sistema, p: ( $\mathbb{Z}$ ,  $\mathbb{Z}$ )) : Bool = (estadoDelCultivo(preS, prm(p), sgd(p))
== RecienSembrado  $\wedge$  estadoDelCultivo(s, prm(p), sgd(p)) == EnCrecimiento);
aux deEnCrecimientoAListoParaCosechar (preS, s:Sistema, p: ( $\mathbb{Z}$ ,  $\mathbb{Z}$ )) : Bool = (estadoDelCultivo(preS, prm(p), sgd(p))
== EnCrecimiento  $\wedge$  estadoDelCultivo(s, prm(p), sgd(p)) == ListoParaCosechar);
aux mismoEstado (preS, s:Sistema, p: ( $\mathbb{Z}$ ,  $\mathbb{Z}$ )) : Bool = (estadoDelCultivo(preS, prm(p), sgd(p))
 $\notin$  [RecienSembrado, EnCrecimiento]  $\wedge$  estadoDelCultivo(s, prm(p), sgd(p)) == estadoDelCultivo(preS, prm(p), sgd(p)));

```

```

problema seVinoLaMalezaS (s: Sistema, ps: [(Z,Z)]) {
  requiere parcelasEnRango : ( $\forall p \leftarrow ps$ )  $enRango(dimensiones(campo(s)), prm(p), sgd(p))$ ;
  requiere parcelasConCultivo : ( $\forall p \leftarrow ps$ )  $contenido(campo(s), prm(p), sgd(p)) == Cultivo$ ;
  modifica s;
  asegura mismoCampo :  $campo(s) == campo(pre(s))$ ;
  asegura mismosDrones :  $mismosDrones(enjambreDrones(s), enjambreDrones(pre(s)))$ ;
  asegura mismoEstado : ( $\forall p \leftarrow parcelasConCultivo(campo(pre(s)))$ ,  $p \notin ps$ )
     $estadoDelCultivo(s, prm(p), sgd(p)) == estadoDelCultivo(pre(s), prm(p), sgd(p))$ ;
  asegura cultivoConMaleza : ( $\forall p \leftarrow ps$ )  $estadoDelCultivo(s, prm(p), sgd(p)) == ConMaleza$ ;
}

problema seExpandePlagaS (s: Sistema) {
  modifica s;
  asegura mismoCampo :  $campo(s) == campo(pre(s))$ ;
  asegura mismosDrones :  $mismosDrones(enjambreDrones(s), enjambreDrones(pre(s)))$ ;
  asegura estadoDelCultivo : ( $\forall p \leftarrow parcelasConCultivo(campo(s))$ )
    ( $seContagiaPlaga(pre(s), p) \wedge estadoDelCultivo(s, prm(p), sgd(p)) == ConPlaga$ )
     $\vee (\neg seContagiaPlaga(pre(s), p) \wedge estadoDelCultivo(s, prm(p), sgd(p)) == estadoDelCultivo(pre(s), prm(p), sgd(p)))$ ;
}

aux seContagiaPlaga (s: Sistema, p: (Z, Z)) : Bool = ( $\exists p' \leftarrow p : parcelasAdyacentesConCultivo(campo(s), p)$ )
 $estadoDelCultivo(s, prm(p'), sgd(p')) == ConPlaga$ ;

problema despegarS (s: Sistema, d: Drone) {
  requiere dronEnSistema :  $droneEnLista(enjambreDrones(s), d)$ ;
  requiere dronNoEnVuelo :  $\neg enVuelo(d)$ ;
  requiere algoDeBateria :  $bateria(d) > 0$ ;
  requiere parcelaLibre :  $\neg((\forall p \leftarrow parcelasAdyacentesAlGranero(campo(s)))$ 
    ( $\exists d' \leftarrow enjambreDrones(s)$ )  $posicionActual(d') == p$ );
  modifica s;
  asegura mismoCampo :  $campo(s) == campo(pre(s))$ ;
  asegura mismoEstado : ( $\forall p \leftarrow parcelasConCultivo(campo(s))$ )
     $estadoDelCultivo(s, prm(p), sgd(p)) == estadoDelCultivo(pre(s), prm(p), sgd(p))$ ;
  asegura mismaCantidadDeDrones :  $|enjambreDrones(s)| == |enjambreDrones(pre(s))|$ ;
  asegura noDmismosDrones : ( $\forall d' \leftarrow enjambreDrones(s)$ ,  $id(d') \neq id(d)$ ) ( $\exists d'' \leftarrow enjambreDrones(pre(s))$ )
     $mismoDrone(d', d'')$ ;
  asegura nuevoDrone : ( $\exists d' \leftarrow enjambreDrones(s)$ )  $id(d') == id(d) \wedge bateria(d') == bateria(d) - 1 \wedge enVuelo(d')$ 
     $\wedge |vueloRealizado(d')| == 2 \wedge vueloRealizado(d')_0 == posicionGranero(campo(s))$ 
     $\wedge mismos(productosDisponibles(d'), productosDisponibles(d))$ ;
}

problema listoParaCosecharS (s: Sistema) = res : Bool {
  asegura res ==  $[|p|p \leftarrow parcelasConCultivo(campo(s)), estadoDelCultivo(p) == ListoParaCosechar]| /$ 
     $|parcelasConCultivo(campo(s))| \geq 0,9$ ;
}

problema aterrizarYCargarBateriaS (s: Sistema, b: Z) {
  modifica s;
  asegura mismoCampo :  $campo(s) == campo(pre(s))$ ;
  asegura mismoEstado : ( $\forall p \leftarrow parcelasConCultivo(campo(s))$ )
     $estadoDelCultivo(s, prm(p), sgd(p)) == estadoDelCultivo(pre(s), prm(p), sgd(p))$ ;
  asegura mismaCantidadDeDrones :  $|enjambreDrones(s)| == |enjambreDrones(pre(s))|$ ;
  asegura dronesIguales : ( $\forall d \leftarrow enjambreDrones(pre(s))$ )
     $bateria(d) \geq b \Rightarrow droneEnLista(enjambreDrones(s), d)$ ;
  asegura dronesAterrizados : ( $\forall d \leftarrow enjambreDrones(pre(s))$ )
     $bateria(d) < b \Rightarrow (\exists d' \leftarrow enjambreDrones(s))$ 
     $id(d') == id(d) \wedge bateria(d') == 100 \wedge \neg enVuelo(d') \wedge (mismos(productosDisponibles(d'), productosDisponibles(d)))$ ;
}

```

```

problema fertilizarPorFilas (s: Sistema) {
  requiere unDronVolandoPorFila :  $\sinRepetidos([sgd(posicionActual(d)) | d \leftarrow dronesEnVuelo(s)])$ ;
  modifica s;
  asegura mismoCampo :  $campo(s) == campo(pre(s))$ ;
  asegura parcelasSinFertilizar :  $(\forall p \leftarrow parcelasConCultivo(campo(pre(s))), p \notin parcelasAFertilizar(pre(s)))$ 
     $estadoDelCultivo(s, prm(p), sgd(p)) == estadoDelCultivo(pre(s), prm(p), sgd(p))$ ;
  asegura parcelasFertilizadas :  $(\forall p \leftarrow parcelasAFertilizar(pre(s)))$ 
     $estadoDelCultivo(pre(s), prm(p), sgd(p)) \in [EnCrecimiento, RecienSembrado]$ 
     $\Rightarrow estadoDelCultivo(s, prm(p), sgd(p)) == ListoParaCosechar$ 
     $estadoDelCultivo(pre(s), prm(p), sgd(p)) \notin [EnCrecimiento, RecienSembrado]$ 
     $\Rightarrow estadoDelCultivo(s, prm(p), sgd(p)) == estadoDelCultivo(pre(s), prm(p), sgd(p))$ ;
  asegura mismaCantidadDeDrones :  $|enjambreDrones(s)| == |enjambreDrones(pre(s))|$ ;
  asegura siNoVuelanNoSeModifican :  $(\forall d \leftarrow enjambreDrones(pre(s)), \neg enVuelo(d))$ 
     $droneEnLista(enjambreDrones(s), d)$ ;
  asegura siVuelanFertilizan :  $(\forall d \leftarrow dronesEnVuelo(pre(s))) (\exists d' \leftarrow enjambreDrones(s))$ 
     $id(d') == id(d) \wedge bateria(d') == bateria(d) - |parcelasAFertilizarPorDrone(s, d)| \wedge \neg enVuelo(d')$ 
     $\wedge mismos(productosDisponibles(d'), eliminarFertilizante(d, |parcelasAFertilizarPorDrone(s, d)|))$ ;
}

aux parcelasAFertilizar (s: Sistema) :  $[(\mathbb{Z}, \mathbb{Z})] = [p | d \leftarrow enjambreDrones(s),$ 
 $p \leftarrow parcelasAFertilizarPorDrone(s, d), enVuelo(d)]$ ;
aux parcelasAFertilizarPorDrone (s: Sistema, d: Drone) :  $[(\mathbb{Z}, \mathbb{Z})] = [(i, sgd(posicionActual(d))) |$ 
 $i \leftarrow [fertilizaHastaColumna(s, d)..prm(posicionActual(d))]$ ;
aux fertilizaHastaColumna (s: Sistema, d: Drone) :  $\mathbb{Z} = max0([primerObstaculoEnColumna(campo(s), d) + 1,$ 
 $prm(posicionActual(d)) - bateria(d), prm(posicionActual(d)) - fertilizanteDisponible(d)])$ ;
aux primerObstaculoEnColumna (c: Campo, d: Drone) :  $\mathbb{Z} = max0([i | i \leftarrow [0..prm(posicionActual(d))],$ 
 $contenido(c, i, sgd(posicionActual(d))) \neq Cultivo])$ ;
aux eliminarFertilizante (d: Drone, cant:  $\mathbb{Z}$ ) :  $[Producto] = [p | p \leftarrow ps, p \neq Fertilizante]$ 
 $+ + [Fertilizante | i \leftarrow [0..|fertilizanteDisponible(d)| - cant]]$ ;

```

```

problema volarYSensarS (s: Sistema, d: Drone) {
  requiere droneEnSistema : droneEnLista(enjambreDrones(s), d) ;
  requiere algoDeBateria : bateria(d) > 0 ;
  requiere parcelaLibre :  $\neg((\forall p \leftarrow \text{parcelasAdyacentes}(\text{campo}(s), \text{posicionActual}(d)))$ 
     $(\exists d' \leftarrow \text{enjambreDrones}(s)) \text{posicionActual}(d') == p)$  ;
  modifica s ;
  asegura mismoCampo : campo(s) == campo(pre(s)) ;
  asegura mismaCantidadDeDrones :  $|\text{enjambreDrones}(s)| == |\text{enjambreDrones}(\text{pre}(s))|$  ;
  asegura mismosDronesSinD :  $(\forall d' \leftarrow \text{enjambreDrones}(\text{pre}(s)), d \neq d') \text{droneEnLista}(\text{enjambreDrones}(s), d')$  ;
  asegura nuevoD :  $(\exists d' \leftarrow \text{enjambreDrones}(s)) \text{id}(d') == \text{id}(d)$ 
     $\wedge \text{bateria}(d') == \text{bateria}(d) - 1 - \text{bateriaConsumida}(\text{pre}(s), d, d')$ 
     $\wedge \text{enVuelo}(d') \wedge |\text{vueloRealizado}(d')| == |\text{vueloRealizado}(d)| + 1$ 
     $\wedge |\text{productosAplicados}(d, d')| \leq 1$ 
     $\wedge \text{mismos}(\text{productosAplicados}(d, d') + +\text{productosDisponibles}(d'), \text{productosDisponibles}(d))$  ;
  asegura noDMismoEstadoDelCultivo :  $(\forall p \leftarrow \text{parcelasConCultivo}(\text{campo}(\text{pre}(s))),$ 
     $p \notin \text{parcelasAfectadas}(\text{pre}(s), d, \text{dronePorId}(s, d)))$ 
     $\text{esPosicionActualNoSensada}(s, \text{dronePorId}(s, d)) \Rightarrow (\text{estadoDelCultivo}(s, \text{prm}(p), \text{sgd}(p)) \neq \text{NoSensado})$ 
     $\wedge \neg \text{esPosicionActualNoSensada}(s, \text{dronePorId}(s, d)) \Rightarrow$ 
     $(\text{estadoDelCultivo}(s, \text{prm}(p), \text{sgd}(p)) == \text{estadoDelCultivo}(\text{pre}(s), \text{prm}(p), \text{sgd}(p)))$  ;
  asegura dNuevoEstadoDelCultivo :  $(\forall p \leftarrow \text{parcelasAfectadas}(\text{pre}(s), d, \text{dronePorId}(s, d)))$ 
     $\text{estadoDelCultivo}(s, \text{prm}(p), \text{sgd}(p)) == \text{RecienSembrado}$  ;
}

aux bateriaConsumida (s: Sistema, d1: Drone, d2: Drone) :  $\mathbb{Z} = \max(0 : [\text{sgd3}(i) | i \leftarrow$ 
impactoProductos(s, d1, productosAplicados(d1, d2))]) ;
aux parcelasAfectadas (s: Sistema, d1: Drone, d2: Drone) :  $[\mathbb{Z}] = [\text{pos} | i \leftarrow$ 
impactoProductos(s, d2, productosAplicados(d1, d2)), pos \leftarrow \text{trc3}(i)] ;
aux productosAplicados (d1: Drone, d2: Drone) :  $[\text{Producto}] = [p | p \leftarrow \text{productosDisponibles}(d1),$ 
p \notin productosDisponibles(d2)] ;
aux impactoProductos (s: Sistema, d: Drone, ps:  $[\text{Producto}]$ ) :  $[(\text{Producto}, \text{Carga}, [(\mathbb{Z}, \mathbb{Z})])] = [i | i \leftarrow \text{infoProductos}(s, d,$ 
prm3}(i) \in ps] ;
aux infoProductos (s: Sistema, d: Drone) :  $[(\text{Producto}, \text{Carga}, [(\mathbb{Z}, \mathbb{Z})])] = [$ 
(Plaguicida, bateria(d)div10, [posicionActual(d)]),
(PlaguicidaBajoConsumo, bateria(d)div20, [posicionActual(d)]),
(Herbicida, bateria(d)div20, [posicionActual(d)]),
(HerbicidaLargoAlcance, bateria(d)div20, posicionActual(d) : parcelasAdyacentesConMaleza(s, posicionActual(d)))] ;
aux esPosicionActualNoSensada (s: Sistema, d: Drone) : Bool = estadoDelCultivo(s, prm(posicionActual(d)),
sgd(posicionActual(d))) == NoSensado ;

```



## 5. Funciones Auxiliares

```

aux cuenta (x: T, a: [T]) :  $\mathbb{Z}$  =  $|[y|y \leftarrow a, y == x]|$ ;
aux distancia (a, b: (Ancho, Largo)) :  $\mathbb{Z}$  =  $|prm(a) - prm(b)| + |sgd(a) - sgd(b)|$ ;
aux sinRepetidos (xs: [T]) : Bool =  $(\forall x \leftarrow xs) cuenta(x, xs) == 1$ ;
aux mismos (a, b: [T]) : Bool =  $(|a| == |b|) \wedge (\forall c \leftarrow a) cuenta(c, a) == cuenta(c, b)$ ;
aux cola (a : [T]) : [T] =  $[a_i | i \leftarrow [1..|a|]]$ ;
aux restarPos (a, b: ( $\mathbb{Z}$ ,  $\mathbb{Z}$ )) : ( $\mathbb{Z}$ ,  $\mathbb{Z}$ ) =  $(prm(a) - prm(b), sgd(a) - sgd(b))$ ;
aux posPositiva0 (p: ( $\mathbb{Z}$ ,  $\mathbb{Z}$ )) : Bool =  $prm(p) \geq 0 \wedge sgd(p) \geq 0$ ;
aux max0 (zs: [ $\mathbb{Z}$ ]) :  $\mathbb{Z}$  =  $[z | z \leftarrow 0 : zs, (\forall z' \leftarrow zs) z \geq z']_0$ ;

```

### 5.1. Campo

```

aux alcanzableEn100 (posG: ( $\mathbb{Z}$ ,  $\mathbb{Z}$ ), a, l:  $\mathbb{Z}$ ) : Bool =  $(\forall i \leftarrow [0..a], j \leftarrow [0..l]) distancia(posG, (i, j)) \leq 100$ ;
aux enRango (dim: (Ancho, Largo), i, j :  $\mathbb{Z}$ ) : Bool =  $0 \leq i < prm(dim) \wedge 0 \leq j < sgd(dim)$ ;
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;
aux parcelasConCultivo (c: Campo) : [ $\mathbb{Z}$ ,  $\mathbb{Z}$ ] =  $[(i, j) | i \leftarrow [0..prm(dimensiones(c))], j \leftarrow [0..sgd(dimensiones(c))],$ 
contenido(c, i, j) == Cultivo];
aux parcelasAdyacentes (c: Campo, p: ( $\mathbb{Z}$ ,  $\mathbb{Z}$ )) : [ $\mathbb{Z}$ ,  $\mathbb{Z}$ ] =  $[(i, j) | i \leftarrow [0..prm(dimensiones(c))],$ 
j  $\leftarrow [0..sgd(dimensiones(c))], distancia((i, j), p) == 1]$ ;
aux parcelasAdyacentesAlGranero (c: Campo) : [ $\mathbb{Z}$ ,  $\mathbb{Z}$ ] = parcelasAdyacentes(c, posicionGranero(c));
aux parcelasAdyacentesConCultivo (c: Campo, p: ( $\mathbb{Z}$ ,  $\mathbb{Z}$ )) : [ $\mathbb{Z}$ ,  $\mathbb{Z}$ ] =  $[p' | p' \leftarrow parcelasConCultivo(c),$ 
distancia(p, p') == 1];

```

### 5.2. Drone

```

aux posicionesRecorridas (ds: [Drone]) : [ $\mathbb{Z}$ ,  $\mathbb{Z}$ ] =  $[p | d \leftarrow ds, p \leftarrow vueloRealizado(d)]$ ;
aux mismoDrone (d1: Drone, d2: Drone) : Bool =  $(id(d1) == id(d2)) \wedge (bateria(d1) == bateria(d2)) \wedge (enVuelo(d1) ==$ 
enVuelo(d2)) \wedge (vueloRealizado(d1) == vueloRealizado(d2)) \wedge (posicionActual(d1) == posicionActual(d2)) \wedge
(mismos(productosDisponibles(d1), productosDisponibles(d2)));
aux mismosDrones (ds1: [Drone], ds2: [Drone]) : Bool =  $(|ds1| == |ds2|) \wedge (sinRepetidos(ds1)) \wedge ((\forall d1 \leftarrow ds1)((\exists d2 \leftarrow$ 
ds2)mismoDrone(d1, d2)));
aux droneEnLista (ds: [Drone], d: Drone) : Bool =  $(\exists d' \leftarrow ds) mismoDrone(d, d')$ ;
aux fertilizanteDisponible (d: Drone) :  $\mathbb{Z}$  =  $[pd | pd \leftarrow productosDisponibles(d), pd == Fertilizante]$ ;
aux dronePorId (ds: [Drone], d: Drone) : Drone =  $[d' | d' \leftarrow ds, id(d') == id(d)]_0$ ;

```

### 5.3. Sistema

```

aux parcelasAdyacentesConMaleza (s: Sistema, p: ( $\mathbb{Z}$ ,  $\mathbb{Z}$ )) : [ $\mathbb{Z}$ ,  $\mathbb{Z}$ ] =  $[p' | p' \leftarrow parcelasConCultivo(campo(s)),$ 
distancia(p, p') == 1 \wedge estadoDelCultivo(s, prm(p), sgd(p)) == ConMaleza];

```

## 6. Aclaraciones

### 6.1. Funciones Auxiliares

Por una cuestión de claridad, las funciones auxiliares genéricas y las que corresponden a cada tipo (Campo, Drone, Sistema) se encuentran en la sección Funciones Auxiliares, mientras que las auxiliares específicas de cada problema se encuentran debajo de las resoluciones de los mismos.

### 6.2. Uso de *mismos* en *vueloRealizadoD*

Dado que el enunciado pide 'la lista de las posiciones de las parcelas que recorrió el dron', consideramos que el orden no importa.

### 6.3. Requiere $|vueloRealizado(d)| \geq 3$ en *vueloEscaleradoD*

Consideramos que una escalera sin escalones no es tal, por lo tanto si el recorrido del vuelo no llega a formar ningún escalón, decimos que no se trata de un vuelo escalonado.

### 6.4. Sobre *vuelosCruzadosD*

La colisión de varios drones en la misma posición en distintos instantes se podría resolver de varias maneras. Debido a la respuesta a nuestra consulta, en esta especificación se suma el número de colisiones. Es decir, si en la posición (2, 3) se cruzaron dos drones en un instante y cuatro drones en otro instante,  $((2, 3), 6) \in res$ .

Si un solo dron estuvo en (2, 3) en un instante y en otro instante hubo un cruce de dos drones en (2, 3),  $((2, 3), 2) \in res$ , dado que la presencia de un solo dron no se considera cruce.

### 6.5. Sobre la batería del dron en *despegarS*

En nuestra especificación decidimos NO asumir que el Drone va a tener algo de batería por estar en el granero, por lo tanto lo requerimos.

### 6.6. Sobre el uso de *pre(s)*

Si por algún motivo es lo mismo utilizar *s* o *pre(s)*, como por ejemplo en el problema *seVinoLaMalezaS :: mismoEstado*, elegimos utilizar *pre(s)* por una cuestión de claridad.

### 6.7. Sobre la interpretación de *seVinoLaMalezaS*

Interpretamos que 'todas las parcelas de cultivo de la lista *ps*' implica que todas las parcelas de la lista *ps* deben tener cultivo. También podría interpretarse que sólo se modifican las parcelas de la lista *ps* que tienen cultivo.

### 6.8. Sobre las auxiliares que devuelven el primer elemento $[T]_0$ de una lista

Pueden existir auxiliares que, si no se tiene en cuenta el problema desde el cual son referenciadas, parecen acceder al primer elemento de una lista posiblemente vacía. Somos conscientes de esto y prestamos especial atención a no romper ninguna función de este estilo :-)

### 6.9. Sobre *fertilizarPorFilas*

Dado que en la especificación del campo se dice que *prm(dimensiones(c))* corresponde al *Ancho* y *sgd(dimensiones(c))* corresponde al *Largo*, vamos a considerar que el primer elemento representa las columnas y el segundo elemento representa las filas.

En nuestra especificación decidimos no considerar la parcela actual en la que se encuentra el Drone, es decir que el mismo se desplaza hacia al oeste sin aplicar fertilizante y sin importar el contenido de esa parcela.

Finalmente, interpretamos que 'finalizar el recorrido' implica dejar de volar (y por tanto regresar al granero y limpiar el historial de vuelos). El dron queda con la batería y los productos sobrantes luego de fertilizar.

### 6.10. Sobre el consumo de batería en *volarYSensarS*

Dado que la carga de batería de un Drone es un entero y el gasto de batería de los productos es porcentual, realizamos una división entera que puede resultar en que el gasto de batería sea aproximado (no igual) al descrito en el enunciado.