DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

TPE - Agricultura con drones

21 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` : $0 \leq i < prm(dimensiones(c)) \land 0 \leq j < sgd(dimensiones(c))$;

`invariante dimensionesValidas` : $prm(dimensiones(c)) > 0 \land sgd(dimensiones(c)) > 0$;
`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 puntosValidos` : $0 \leq \mathsf{prm}(posG) \land 0 \leq \mathsf{sgd}(posG) \land 0 \leq \mathsf{prm}(posC) \land 0 \leq \mathsf{sgd}(posC)$;
`requiere puntosDistintos` : $\mathsf{prm}(posG) \neq \mathsf{prm}(posC) \lor \mathsf{sgd}(posG) \neq \mathsf{sgd}(posC)$;
`requiere aMenosDe100` : $distancia(posG, posC) \leq 100$;
`asegura` $posicionGranero(res) == posG(*)$;
`asegura` $posicionCasa(res) == posC(*)$;
}

- Falta requerir que distancia((0,0), posG) $\leq$ 100
- (*) enRango(dimensiones(res), prm(posG), sgd(posG)) $\land$ (contenido(res, prm(posG), sgd(posG)) == Granero) $\leftarrow$ mas declarativo

`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) : [(ℤ, ℤ)] ;
        observador posicionActual (d: Drone) : (ℤ, ℤ) ;
        observador productosDisponibles (d: Drone) : [Producto] ;

        invariante vuelosOk :
            enVuelo(d) ⇒ (|vueloRealizado(d)| > 0 ∧ posicionActual(d) == vueloRealizado(d)_{|vueloRealizado(d)|−1} ∧
            posicionesPositivas(d) ∧ movimientosOK(d)) ∧ ¬enVuelo(d) ⇒ |vueloRealizado(d)| == 0 ;
        invariante bateriaOk : 0 ≤ bateria(d) ≤ 100 ;
}
aux posicionesPositivas (d: Drone) : Bool = (∀i ← [0.. |vueloRealizado(d)|))prm(vueloRealizado(d)_i) ≥ 0 ∧
sgd(vueloRealizado(d)_i ≥ 0 ;
aux movimientosOK (d: Drone) : Bool = (∀i ← [1.. |vueloRealizado(d)|))
prm(vueloRealizado(d)_i) == prm(vueloRealizado(d)_{i−1}) ∧ (sgd(vueloRealizado(d)_i) == sgd(vueloRealizado(d)_{i−1}) − 1 ∨
sgd(vueloRealizado(d)_i) == sgd(vueloRealizado(d)_{i−1}) + 1) ∨ sgd(vueloRealizado(d)_i) == sgd(vueloRealizado(d)_{i−1})
∧(prm(vueloRealizado(d)_i) == prm(vueloRealizado(d)_{i−1})−1∨prm(vueloRealizado(d)_{i−1}) == prm(vueloRealizado(d)_{i−1})+
1) ;
problema crearD (i: ℤ, ps: [Producto]) = res : Drone {
        asegura bateria(res) == 100 ∧ id(res) == i ;
        asegura ¬enVuelo(res) ;
        asegura mismos(productosDisponibles(res), ps) ;
}

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

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

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

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

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

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

- usar mismos! no importa el orden

```
problema vueloEscaleradoD (d: Drone) = res : Bool {
        asegura res == enVuelo(d) ∧ movimientoAlternado(d) ∧ movimientoUnidireccional(d) ;
}

problema vuelosCruzadosD (ds: [Drone]) = res : [((ℤ, ℤ), ℤ)] {
        requiere todosEnVuelo : (∀d ∈ ds)enVuelo(d) ;
        requiere vuelosDeIgualLargo : (∀i ∈ [1.. |ds|)) |vueloRealizado(ds_i)| == |vueloRealizado(ds_{i−1})| ;
        asegura (ds == [ ] ∧ res == [ ]) ∨ (mismos(res, [cruce | i ← [0.. |vueloRealizado(ds_0)|),
            cruce ← [(posicion, |[1|d ← ds, vueloRealizado(d)_i == posicion]|)|
            posicion ← posicionesSinRepetir([vueloRealizado(d)_i | d ← ds])], sgd(cruce) > 1])) ;
```

`asegura enOrden` : $(\forall i \in [1..|res|))\mathsf{sgd}(res_i) \geq \mathsf{sgd}(res_{i-1}) \vee (\forall i \in [1..|res|))\mathsf{sgd}(res_i) \leq \mathsf{sgd}(res_{i-1})$ **;**

}

4. Sistema

```
tipo Sistema {
        observador campo (s: Sistema) : Campo ;
        observador estadoDelCultivo (s: Sistema, i, j: ℤ) : EstadoCultivo ;
            requiere enRango(dimensiones(s), i, j) ∧ contenido(campo(s), i, j) == Cultivo ;
        observador enjambreDrones (s: Sistema) : [Drone] ;

        invariante identificadoresUnicos : sinRepetidos([ id(d) | d ← enjambreDrones(s) ]) ;
        invariante unoPorParcela : (∀d, d' ← dronesEnVuelo(s), id(d) ≠ id(d'))posicionActual(d) ≠ posicionActual(d') ;
        invariante siNoVuelanEstanEnGranero : (∀d ← enjambreDrones(s), ¬enVuelo(d))
            posicionActual(d) == posicionGranero(campo(s)) ;
        invariante siEstanEnVueloElVueloEstaEnRango : (∀d ← dronesEnVuelo(s))(∀v ← vueloRealizado(d))
            enRango(dimensiones(campo(s), prm(v), sgd(v)) ;
}
    aux dronesEnVuelo (s: Sistema) : [Drone] = [ d | d ← enjambreDrones(s), enVuelo(d) ] ;
problema crearS (c: Campo, ds: [Drone]) = res : Sistema {
        asegura dronesEnGranero : (∀d ∈ ds) posicionActual(d) == posicionGranero(c)(*) ;
        asegura bateriaLlena : (∀d ∈ ds) bateria(d) == 100(*) ;
        asegura cultivosNoSensados : (∀i ∈ [0...prm(dimensiones(c))), j ← [0...sgd(dimensiones(c))))
            contenido(res, i, j) == Cultivo ⟶ estadoDelCultivo(c, i, j) == NoSensado(*) ;
        asegura (campo(res) == c ;
        asegura mismos(enjambreDrones(res), ds)(**) ;
}
```

- (*) no tiene sentido asegurar cosas sobre los parametros.
  En todo caso deberian ser requieres
- (**) mismos utiliza el '==' de dron!
- falta requerir ids de drones unicos y que no esten en vuelo

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

problema estadoDelCultivoS (s: Sistema, i, j: 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 : (∀i ∈ [0..prm(dimensiones(campo(s))))](∀j ∈ [0..sgd(dimensiones(campo(s))))
            contenido(campo(s), i, j) == Cultivo ⟶
            (estadoDelCultivo(pre(s), i, j) == RecienSembrado ⟶ estadoDelCultivo(s, i, j) == EnCrecimiento
            ∧ estadoDelCultivo(pre(s), i, j) == EnCrecimiento ⟶ estadoDelCultivo(s, i, j) == ListoParaCosechar
            ∧ (estadoDelCultivo(pre(s), i, j) ≠ RecienSembrado ∧ estadoDelCultivo(pre(s), i, j) ≠ EnCrecimiento)
            ⟶ estadoDelCultivo(pre(s), i, j) = estadoDelCultivo(s, i, j)) ;
}

problema seVinoLaMalezaS (s: Sistema, ps: [(ℤ, ℤ)]) {
        requiere (∀(i, j) ∈ ps) enRango(dimensiones(campo(s)), i, j) ;
        requiere (∀(i, j) ∈ 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 : $(\forall i \in [0..\text{prm}(dimensiones(campo(s))))(\forall j \in [0..\text{sgd}(dimensiones(campo(s)))))$
$\text{en}((i,j),ps) \longrightarrow estadoDelCultivo(s,i,j) == ConMaleza$
$\wedge \neg(\text{en}((i,j),ps)) \longrightarrow estadoDelCultivo(s,i,j) == estadoDelCultivo(pre(s),i,j)$;
}

problema seExpandePlagaS (s: Sistema) {
modifica s ;
asegura mismoCampo : $campo(\text{pre}(s)) == campo(s)$;
asegura mismosDrones : $mismos(enjambreDrones(\text{pre}(s)), enjambreDrones(s))$;
asegura expansiónDePlaga : $(\forall i \in [0..\text{prm}(dimensiones(campo(s))))(\forall j \in [0..\text{sgd}(dimensiones(campo(s))))$
$adyacenteAPlaga(s,i,j) \longrightarrow estadoDelCultivo(s,i,j) == ConPlaga$
$\wedge \neg(adyacenteAPlaga(s,i,j)) \longrightarrow estadoDelCultivo(s,i,j) == estadoDelCultivo(pre(s),i,j)$;
}

problema despegarS (s: Sistema, d: Drone) {
requiere $\neg enVuelo(d)$;
requiere $PosicionActual(d) == PosicionGranero(campo(s))$;
requiere $en(d, enjambreDrones(s))(*)$;
modifica $s, d(**)$;
asegura mismoCampo : $campo(\text{pre}(s)) == campo(s)$;
asegura mismoEstadoCultivo : $(\forall i \in [0..\text{prm}(dimensiones(campo(s))))(\forall j \in [0..\text{sgd}(dimensiones(campo(s))))$
$contenido(campo(s),i,j) \neq Cultivo \vee estadoDelCultivo(s,i,j) == estadoDelCultivo(pre(s),i,j))$;
asegura mismoDron : $id(d) == id(\text{pre}(d))$;
asegura mismaBateria : $bateria(d) == bateria(\text{pre}(d))$;
asegura mismaPosicion : $posicionActual(d) == posicionActual(\text{pre}(d))$;
asegura mismosProductos : $mismos(productosDisponibles(d), productosDisponibles(\text{pre}(d)))$;
asegura despegue : $enVuelo(d) \wedge vueloRealizado(d) == [posicionActual]$;
asegura mismoEnjambre : $mismos(enjambreDrones(s), d : [e \mid e \leftarrow enjambreDrones(\text{pre}(s)), e \neq \text{pre}(d)])$;
(***) }

- (*) 'en' usa el '==' de dron!
- (**)(***) hay que modificar el sistema s, no el dron!
- falta requerir que tenga al 100 la bateria

problema listoParaCosecharS (s: Sistema) = res : Bool {
asegura $res == (|[1|(i,j) \leftarrow cultivos(s), estadoDelCultivo(s,i,j) == ListoParaCosechar]| \div |cultivos(s)| \geq 0{,}9)$;
}

problema aterrizarYCargarBateriaS (s: Sistema, b: $\mathbb{Z}$) {
requiere $0 < b \leq 100$;
modifica s ;
asegura mismoCampo : $campo(\text{pre}(s)) == campo(s)$;
asegura mismoEstadoCultivo : $(\forall i \in [0..\text{prm}(dimensiones(campo(s))))(\forall j \in [0..\text{sgd}(dimensiones(campo(s))))$
$contenido(campo(s),i,j) \neq Cultivo \vee estadoDelCultivo(s,i,j) == estadoDelCultivo(pre(s),i,j))$;
asegura mismosDrones : $|enjambreDrones(s)| == |enjambreDrones(\text{pre}(s))|$;
asegura cambiosADrones : $(\forall d \in enjambreDrones(pre(s)))$
$(bateria(d) \leq b \longrightarrow ((\exists e \in enjambreDrones(s))$
$id(e) == id(d) \wedge bateria(e) == 100 \wedge \neg enVuelo(e) \wedge posicionActual(e) == posicionGranero(campo(s)) \wedge$
$vueloRealizado(d) == [\,] \wedge productosDisponibles(e) == productosDisponibles(d)(*))) \wedge (bateria(d) > b \longrightarrow$
$\text{en}(d, enjambreDrones(s)(**)))$;
}

- mismos!
- usa el -=”de dron!

problema fertilizarPorFilas (s: Sistema) {
requiere maxUnDronPorFila : $(\forall j \in [0..\text{sgd}(dimension(campo(s))))$
$|[1|i \leftarrow [0..\text{prm}(dimension(campo(s)))), \text{en}((i,j), posicionesDronesEnVuelo(s))]| < 2$;
modifica s ;
asegura mismoCampo : $campo(\text{pre}(s)) == campo(s)$;
asegura mismosDrones : $|enjambreDrones(s)| == |enjambreDrones(\text{pre}(s))|$
$\wedge (\forall d \in enjambreDrones(\text{pre}(s)))(\exists e \in enjambreDrones(s))id(e) == id(d)$;

asegura soloCambiarSiEnVuelo : $(\forall d \in enjambreDrones(\mathsf{pre}(s)))(\exists e \in enjambreDrones(s), id(e) == id(d))$
$enVuelo(d) == enVuelo(e) \land \neg enVuelo(d) \longrightarrow d == e$;
asegura vuelosAlOeste : $(\forall d \in enjambreDrones(\mathsf{pre}(s)))(\exists e \in enjambreDrones(s), id(e) == id(d))$
$\mathsf{prm}(posicionActual(e)) == \mathsf{prm}(posicionActual(d)) - |vueloDiferente(e, d)|$;
asegura gastoDeBateria : $(\forall d \in enjambreDrones(\mathsf{pre}(s)))(\exists e \in enjambreDrones(s), id(e) == id(d))$
$bateria(e) == (bateria(d) + |vueloDiferente(e, d)|)$;
asegura unFertilizantePorCultivo : $(\forall d \in enjambreDrones(\mathsf{pre}(s)))(\exists e \in enjambreDrones(s), id(e) == id(d))$
$mismos(productosDisponibles(d), productosDisponibles(e) ++ [Fertilizante \mid i \leftarrow [0..|vueloDiferente(e, d)|]])$;
asegura soloCultivos : $(\forall d \in enjambreDrones(\mathsf{pre}(s)))(\exists e \in enjambreDrones(s), id(e) == id(d))$
$(\forall (i, j) \in vueloDiferente(e, d))contenido(campo(s), i, j) == Cultivo$;
asegura condicionDeFinalizacion : $(\forall d \in enjambreDrones(s))$
$bateria(d) == 0 \lor \neg(\mathsf{en}(Fertilizante, productosDisponibles(d)))$
$\lor\ contenido(campo(s), \mathsf{prm}(posicionActual(d)), \mathsf{sgd}(posicionActual(d))) \neq Cultivo$
$\lor\ \mathsf{sgd}(posicionActual(d)) == 0$;
asegura fertilizarCampo : $(\forall d \in enjambreDrones(\mathsf{pre}(s)))$
$(\exists e \in enjambreDrones(s), id(e) == id(d))(\forall (i, j) \in vueloDiferente(e, d))$
$((\mathsf{en}(estadoDelCultivo(\mathsf{pre}(s), i, j), [RecienSembrado, EnCrecimiento])$
$\longrightarrow estadoDelCultivo(s, i, j) == ListoParaCosechar)$
$\land\ (\neg\mathsf{en}(estadoDelCultivo(\mathsf{pre}(s), i, j), [RecienSembrado, EnCrecimiento])$
$\longrightarrow estadoDelCultivo(s, i, j) == estadoDelCultivo(\mathsf{pre}(s), i, j))$;
}

- falta ver que pasa con vuelo realizado
- no dice que pasa con las posiciones que no cambian
- como cambia el estado del cultivo depende de que vuelo haya hecho cada dron!
- y si se chocan con el granero o la casa?

problema volarYSensarS (s: Sistema, d: Drone) {
requiere puedeMoverse : $bateria(d) > 0$;
requiere $enVuelo(d)$;
requiere $en(d, enjambreDrones(s))$;
modifica $s, d(***)$;
asegura mismoCampo : $campo(\mathsf{pre}(s)) == campo(s)$;
asegura mismoEnjambre : $mismos(enjambreDrones(s), d : [e \mid e \leftarrow enjambreDrones(\mathsf{pre}(s)), e \neq (*)\mathsf{pre}(d)])$;
asegura mismoDron : $id(d) == id(\mathsf{pre}(d))?$;
asegura aParcelaAdyacente : $|vueloDiferente(d, pre(d))| == 1$
$\land\ enRango(dimensiones(campo(s)), \mathsf{prm}(posicionActual(d)), \mathsf{sgd}(posicionActual(d)))$;
asegura sensado : $estadoDelCultivo(\mathsf{pre}(s), \mathsf{prm}(posicionActual(d)), \mathsf{sgd}(posicionActual(d))) == NoSensado \longrightarrow$
$(estadoDelCultivo(s, \mathsf{prm}(posicionActual(d)), \mathsf{sgd}(posicionActual(d))) \neq NoSensado \land noHaceNada(d, \mathsf{pre}(d)))$;
asegura siEstaBienNoHaceNada :
$\neg\mathsf{en}(estadoDelCultivo(\mathsf{pre}(s), \mathsf{prm}(posicionActual(d)), \mathsf{sgd}(posicionActual(d))), [ConMaleza, ConPlaga, NoSensado])$
$\longrightarrow (noHaceNada(d, \mathsf{pre}(d)) \land estadoDelCultivo(\mathsf{pre}(s), \mathsf{prm}(posicionActual(d)), \mathsf{sgd}(posicionActual(d))) ==$
$== estadoDelCultivo(s, \mathsf{prm}(posicionActual(d)), \mathsf{sgd}(posicionActual(d))))$;
asegura mataMalezas : $(**)estadoDelCultivo(\mathsf{pre}(s), \mathsf{prm}(posicionActual(d)), \mathsf{sgd}(posicionActual(d))) == ConMaleza$
$\longrightarrow (usaAlgunHerbicida(d, \mathsf{pre}(d))$
$\land\ estadoDelCultivo(s, \mathsf{prm}(posicionActual(d)), \mathsf{sgd}(posicionActual(d))) == RecienSembrado)$
$\lor\ ((\neg tieneHerbicida(d) \lor bateria(\mathsf{pre}(d)) < 6)$
$\land noHaceNada(d, \mathsf{pre}(d)) \land estadoDelCultivo(s, \mathsf{prm}(posicionActual(d)), \mathsf{sgd}(posicionActual(d))) == ConMaleza))$;
asegura mataPlagas : $estadoDelCultivo(\mathsf{pre}(s), \mathsf{prm}(posicionActual(d)), \mathsf{sgd}(posicionActual(d))) == ConPlaga$
$\longrightarrow (usaPlaguicidaComun(d, \mathsf{pre}(d))$
$\land\ estadoDelCultivo(s, \mathsf{prm}(posicionActual(d)), \mathsf{sgd}(posicionActual(d))) == RecienSembrado)$
$\lor\ (usaPlaguicidaBajoConsumo(d, \mathsf{pre}(d))$
$\land\ estadoDelCultivo(s, \mathsf{prm}(posicionActual(d)), \mathsf{sgd}(posicionActual(d))) == RecienSembrado)$
$\lor((tienePlaguicida(d) \land ((\neg tienePlaguicidaBajoConsumo(d) \land bateria(d) < 11) \lor (tienePlaguicidaBajoConsumo(d) \land$
$bateria(d) < 6))) \lor (\neg tienePlaguicida(d) \land \neg tienePlaguicidaBajoConsumo(d))) \land noHaceNada(d, \mathsf{pre}(d))$
$\land\ estadoDelCultivo(s, \mathsf{prm}(posicionActual(d)), \mathsf{sgd}(posicionActual(d))) == ConPlaga))$;
}

- (*) los drones no ¿? son los mismos! el drone d en s cambia!
- (**) si usa largo alcance, debe modificar tambien parcelas adyacentes
- (***) la idea es modificar el sistema s, no el drone d! habia que modificar en s a¿? drone d', donde id(d')==id(d)

5. Funciones Auxiliares

5.1. Campo

`aux posicionGranero` (c: Campo) : ($\mathbb{Z}$, $\mathbb{Z}$) $= [(i,j) \mid i \leftarrow [0..prm(dimensiones(c))), j \leftarrow [0..sgd(dimensiones(c))), contenido(c,i,j) == Granero]_0$;

`aux posicionCasa` (c: Campo) : ($\mathbb{Z}$, $\mathbb{Z}$) $= [(i,j) \mid i \leftarrow [0..prm(dimensiones(c))), j \leftarrow [0..sgd(dimensiones(c))), contenido(c,i,j) == Casa]_0$;

5.2. Drone

`aux movimientoAlternado` (d: Drone) : Bool $= (\forall d \in [2..\ |vueloRealizado(d))|)$
$\mathsf{prm}(vueloRealizado(d)_i) == \mathsf{prm}(vueloRealizado(d)_{i-1}) \Leftrightarrow |\mathsf{prm}(vueloRealizado(d)_i) - \mathsf{prm}(vueloRealizado(d)_{i-2})| == 1 \wedge$
$\mathsf{sgd}(vueloRealizado(d)_i) == \mathsf{sgd}(vueloRealizado(d)_{i-1}) \Leftrightarrow |\mathsf{sgd}(vueloRealizado(d)_i) - \mathsf{sgd}(vueloRealizado(d)_{i-2})| == 1$;

`aux movimientoUnidireccional` (d: Drone) : Bool $= (\forall d \in [3..\ |vueloRealizado(d))|)$
$|\mathsf{prm}(vueloRealizado(d)_i) - \mathsf{prm}(vueloRealizado(d)_{i-1})| == 1 \Leftrightarrow$
$|\mathsf{prm}(vueloRealizado(d)_i) - \mathsf{prm}(vueloRealizado(d)_{i-3})| == 2 \wedge$
$|\mathsf{sgd}(vueloRealizado(d)_i) - \mathsf{sgd}(vueloRealizado(d)_{i-1})| == 1 \Leftrightarrow$
$|\mathsf{sgd}(vueloRealizado(d)_i) - \mathsf{sgd}(vueloRealizado(d)_{i-3})| == 2$;

`aux posicionesSinRepetir` (ps: [($\mathbb{Z}$, $\mathbb{Z}$)]) : [($\mathbb{Z}$, $\mathbb{Z}$)] $= [ps_i \mid i \leftarrow [0..\ |ps|), (\forall j \in [0..i))\ ps_i \neq ps_j]$;

5.3. Sistema

`aux adyacenteAPlaga` (s: Sistema, (i,j): $\mathbb{Z}$) : Bool $=$
$\mathsf{en}(ConPlaga, [estadoDelCultivo(s,i,j) | (i,j) \leftarrow cultivosAdyacentes((i,j),s)])$;

`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) : [($\mathbb{Z}$, $\mathbb{Z}$)] $= [(i,j) | i \leftarrow [0..\mathsf{prm}(dimensiones(campo(s))))$
$j \leftarrow [0..\mathsf{sgd}(dimensiones(campo(s)))), contenido(campo(s),i,j) == Cultivo]$;

`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 $= \mathsf{en}(Herbicida, productosDisponibles(d))$
$\vee\ \mathsf{en}(HerbicidaLargoAlcance, productosDisponibles(d))$;

`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 $= \mathsf{en}(Plaguicida, productosDisponibles(d))$;

`aux tienePlaguicidaBajoConsumo` (d: Drone) : Bool $= \mathsf{en}(PlaguicidaBajoConsumo, productosDisponibles(d))$;