



TPE - Agricultura con drones

6 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
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 :  $0 \leq i < \text{prm}(\text{dimensiones}(c)) \wedge 0 \leq j < \text{sgd}(\text{dimensiones}(c))$ ;
  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 (i:  $\mathbb{Z}$ , ps: [Producto]) = res : Drone {
  asegura bateria(res) == 100;
  asegura id(res) == i;
  asegura enVuelo(res) == False;
  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 res == 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 vueloEscaleraD (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 : ( $\forall d \leftarrow ds$ ) enVuelo(d);
  requiere vuelosDeIgualLargo : ( $\forall i \leftarrow [1..|ds|]) |vueloRealizado(ds_i)| == |vueloRealizado(ds_{i-1})|$ ;
  asegura crucesValidos : ( $\forall cruce \leftarrow res$ ) sgd(cruce)  $> 1$ ;
  asegura |ds| == 0  $\vee$  (mismos(res, [cruce]_i  $\leftarrow [0..|vueloRealizado(ds_0)|]$ ,
    cruce  $\leftarrow [(posicion, |[1|d \leftarrow ds, vueloRealizado(d)_i == posicion]|)$ 
    posicion  $\leftarrow sinRepetidos([vueloRealizado(d)_i | d \leftarrow ds])$ , sgd(cruce)  $> 1$ ));
}

```

$\}$ $\text{asegura enOrden} : (\forall i \leftarrow [1..|res|]) \text{sgd}(res_i) \geq \text{sgd}(res_{i-1}) \vee (\forall i \leftarrow [1..|res|]) \text{sgd}(res_i) \leq \text{sgd}(res_{i-1}) ;$

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 :  $sinRepetidos([id(d) \mid d \leftarrow enjambreDrones(s)])$ ;
  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] =  $[d \mid d \leftarrow enjambreDrones(s), enVuelo(d)]$ ;

problema crearS (c: Campo, ds: [Drone]) = res : Sistema {
}

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

problema estadoDelCultivoS (s: Sistema, i, j:  $\mathbb{Z}$ ) = res : EstadoCultivo {
}

problema enjambreDronesS (s: Sistema) = res : [Drone] {
}

problema crecerS (s: Sistema) {
}

problema seVinoLaMalezaS (s: Sistema, ps:  $[(\mathbb{Z}, \mathbb{Z})]$ ) {
}

problema seExpandePlagaS (s: Sistema) {
}

problema despegarS (s: Sistema, d: Drone) {
}

problema listoParaCosecharS (s: Sistema) = res : Bool {
}

problema aterrizarYCargarBateriaS (s: Sistema, b:  $\mathbb{Z}$ ) {
}

problema fertilizarPorFilas (s: Sistema) {
}

problema volarYSensarS (s: Sistema, d: Drone) {
}
```

5. Funciones Auxiliares

5.1. Campo

5.2. Drone

```
aux movimientoAlternado (d: Drone) : Bool = (∀d ← [2..|vueloRealizado(d)|])
prm(vueloRealizado(d)i) == prm(vueloRealizado(d)i-1) ⇔ |prm(vueloRealizado(d)i) - prm(vueloRealizado(d)i-2)| == 1 ∧
sgd(vueloRealizado(d)i) == sgd(vueloRealizado(d)i-1) ⇔ |sgd(vueloRealizado(d)i) - sgd(vueloRealizado(d)i-2)| == 1 ;
aux movimientoUnidireccional (d: Drone) : Bool = (∀d ← [3..|vueloRealizado(d)|])
|prm(vueloRealizado(d)i) - prm(vueloRealizado(d)i-1)| == 1 ⇔
|prm(vueloRealizado(d)i) - prm(vueloRealizado(d)i-3)| == 2 ∧
|sgd(vueloRealizado(d)i) - sgd(vueloRealizado(d)i-1)| == 1 ⇔
|sgd(vueloRealizado(d)i) - sgd(vueloRealizado(d)i-3)| == 2 ;
```

5.3. Sistema