DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

TPE - Agricultura con drones

8 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` $posicionCasa(res) == posC$;
}

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

`problema contenidoC` (c: Campo, i, j: $\mathbb{Z}$) = `res` : Parcela {
`requiere enRango` : $0 \leq i < \mathsf{prm}(dimensiones(c)) \land 0 \leq j < \mathsf{sgd}(dimensiones(c))$;
`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) : [(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 ;
        asegura id(res) == i ;
        asegura enVuelo(res) == False ;
        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 res == vueloRealizado(d) ;
}

problema posicionActualD (d: Drone) = res : (ℤ, 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) ∧ movimientoAlternado(d) ∧ movimientoUnidireccional(d) ;
}

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

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

}

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(c, i, j) == Cultivo ⇒ 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: 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 mismos(dameEstado(pre(s), RecienSembrado), dameEstado(s, EnCrecimiento)) ;
        asegura mismos(dameEstado(pre(s), EnCrecimiento), dameEstado(s, ListoParaCosechar)) ;
        asegura mismos(dameOtroEstado(pre(s), [RecienSembrado, EnCrecimiento, ListoParaCosechar]),
            dameOtroEstado(s, [RecienSembrado, EnCrecimiento, ListoParaCosechar]) ;
}

problema seVinoLaMalezaS (s: Sistema, ps: [(ℤ, ℤ)]) {
        requiere (∀(i, j) ← ps)enRango(dimensiones(campo(s)), i, j) ;
        modifica s ;
        asegura mismos(dameOtroEstado(pre(s), []), dameEstado(s, ConMaleza)) ;
}

problema seExpandePlagaS (s: Sistema) {
        modifica s ;
        asegura (∀(i, j) ← dameEstado(pre(s), ConPlaga))
            todasTienenEstado(parcelasAdyacentes(i, j, campo(s)), s, ConPlaga)) ;
        asegura mismos(noCAdyacente(pre(s)), noCAdyacente(s)) ;
}

problema despegarS (s: Sistema, d: Drone) {
        requiere ¬enVuelo(d) ;
        requiere bateria(d) = 100 ;
        modifica s ;
        asegura |vueloRealizado(d)| > 0 ;
```

}

problema listoParaCosecharS (s: Sistema) = res : Bool {
    asegura $|dameEstado(s, ListoParaCosechar)| / |dameOtroEstado(s, [])| \geq 0{,}9$;
}

problema aterrizarYCargarBateriaS (s: Sistema, b: $\mathbb{Z}$) {
    modifica s ;
    asegura $(\forall d \leftarrow (dronesConBateria(enjambreDrones(pre(s)), b))$
        $bateria(d) == 100 \wedge |vuelosRealizados(d)| == 0 \wedge \neg enVuelo(d)$;
}

problema fertilizarPorFilas (s: Sistema) {
    requiere $sinRepetidos(listaAbscisas(posicionesEnjambre(s)))$;
    modifica s ;
    asegura $(\forall d \leftarrow dejanDeFuncionar(pre(s)))(\forall e \leftarrow enjambreDrones(s))$
        $id(s) == id(e), posicionActual(e) == posicionGranero(campo(s))$;
    asegura $(\forall e \leftarrow enjambreDrones(s))(\forall d \leftarrow enjambreDrones(pre(s)))$
        $(coordAbscisa(e) < coordAbscisa(d) \vee posicionActual(e) == posicionGranero(campo(s)))$
        $\wedge (mismos(productosDisponibles(e), Fertilizante : productosDisponibles(d)))$;
}

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

5. Funciones Auxiliares

5.1. Campo

5.2. Drone

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

5.3. Sistema

aux dameEstado (s: Sistema, estado: EstadoCultivo) : $[(\mathbb{Z}, \mathbb{Z})] = [(i,j)|$
$i \leftarrow [0..\mathsf{prm}(dimensiones(campo(s)))), j \leftarrow [o..\mathsf{sgd}(dimensiones(campo(s)))),$
$contenido(campo(s), i, j) == Cultivo \wedge estadoDeCultivo(s, i, j) == estado]$;
aux dameOtroEstado (s: Sistema, fueraEstados: [EstadoCultivo]) : $[(\mathbb{Z}, \mathbb{Z})] = [(i,j)|$
$i \leftarrow [0..\mathsf{prm}(dimensiones(campo(s)))), j \leftarrow [o..\mathsf{sgd}(dimensiones(campo(s)))),$
$contenido(campo(s), i, j) == Cultivo \wedge (\forall estado \leftarrow fueraEstados) estadoDeCultivo(s, i, j) \neq estado]$;
aux parcelasAdyacentes ((i,j):($\mathbb{Z}$,$\mathbb{Z}$), c: Campo) : $[(\mathbb{Z}, \mathbb{Z})] = [(x,y)|$
$x \leftarrow [i-1, i+1], y \leftarrow [j-1, j+1], contenido(c, x, y) == Cultivo$
$\wedge enRango(dimensiones(c), x, y)]$;
aux todasTienenEstado (ps: $[(\mathbb{Z}, \mathbb{Z})]$, s: Sistema, estado: EstadoCultivo) : Bool $= (\forall (i,j) \leftarrow ps) estadoDeCultivo(s, i, j) ==$
$estado$;
aux noCAdyacente (s: Sistema) : $[(\mathbb{Z}, \mathbb{Z})] = [(x,y)|$
$x \leftarrow [0..\mathsf{prm}(dimensiones(campo(s)))), y \leftarrow [0..\mathsf{sgd}(dimensiones(campo(s)))),$
$(x,y) \notin parcelasAdyacentes((x,y), campo(s))]$;
aux dronesConBateria (ds: [Drone], b: $\mathbb{Z}$) : [Drone] $= [d|$
$d \leftarrow ds, bateria(d) \leq b]$;
aux posicionesEnjambre (s: Sistema) : $[(\mathbb{Z}, \mathbb{Z})] = [posicionActual(d)|$
$d \leftarrow enjambreDrones(s)]$;
aux listaAbscisas (cs: $[(\mathbb{Z}, \mathbb{Z})]$) : $[\mathbb{Z}] = [\mathsf{prm}(coord)|coord \leftarrow cs]$;
aux coordAbscisa (d: Drone) : $\mathbb{Z} = \mathsf{prm}(posicionActual(d))$;
aux dejanDeFuncionar (s: Sistema) : [Drone] $= [d|d \leftarrow enjambreDrones(s),$
$bateria(d) == 0$
$\vee\ \mathsf{prm}(posicionActual(d)) == 0$
$\vee\ Fertilizante \notin productosDisponibles(d)$
$\vee\ contenido(campo(s), \mathsf{prm}(posicionActual(d)), \mathsf{sgd}(posicionActual(d))) \neq Cultivo$;