UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Algoritmos y Estructura de Datos I

Segundo cuatrimestre de 2016

5 de septiembre de 2016

TPE OJOTA (Organización de Juegos Olímpicos Tp de Algoritmos 1) v1.0

1. Auxiliares definidas por nosotros

`aux participa` (j:JJOO,a:Atleta) : Bool $= (\exists c \leftarrow competencias(j))a \in participantes(c)$;
`aux paises` (j:JJOO) : [Pais] $= [\, nacionalidad(a) \mid a \leftarrow atletas(j) \,]$;

2. Tipos

`tipo` Deporte = String ;
`tipo` Pais = String ;
`tipo` Sexo = Femenino, Masculino ;

3. Atleta

`tipo Atleta` {
`observador nombre` (a: Atleta) : String ;
`observador sexo` (a: Atleta) : Sexo ;
`observador añoNacimiento` (a: Atleta) : $\mathbb{Z}$;
`observador nacionalidad` (a: Atleta) : Pais ;
`observador ciaNumber` (a: Atleta) : $\mathbb{Z}$;
`observador deportes` (a: Atleta) : [Deporte] ;
`observador capacidad` (a: Atleta, d: Deporte) : $\mathbb{Z}$;
`requiere` $d \in deportes(a)$;

`invariante` $|deportes(a)| > 0$;
`invariante` $sinRepetidos(deportes(a))$;
`invariante` $ordenada(deportes(a))$;
`invariante capacidadEnRango` : $(\forall d \leftarrow deportes(a))0 \leq capacidad(a,d) \leq 100$;
}

`problema especialidad` (a: Atleta) = `res` : Deporte {
`asegura` $result \in deportes(a)$;
`asegura` $(\forall d \leftarrow deportes(a))capacidad(a,d) \leq capacidad(a,res)$;
}

`problema entrenarNuevoDeporte` (a: Atleta, d: Deporte, c: $\mathbb{Z}$) {
`requiere` $\neg(d \in deportes(a))$;
`requiere` $0 \leq c \leq 100$;

`modifica` a ;

`asegura` $nombre(a) == nombre(\mathsf{pre}(a))$;
`asegura` $sexo(a) == sexo(\mathsf{pre}(a))$;
`asegura` $añoNacimiento(a) == añoNacimiento(\mathsf{pre}(a))$;
`asegura` $nacionalidad(a) == nacionalidad(\mathsf{pre}(a))$;
`asegura` $ciaNumber(a) == ciaNumber(\mathsf{pre}(a))$;
`asegura SeAgregoDeporte` : $mismos(deportes(a), d : deportes(\mathsf{pre}(a))) \wedge ordenada(deportes(a))$;
`asegura LosOtrosNoCambiaron` : $(\forall x \leftarrow deportes(a), x \neq d)capacidad(a,x) == capacidad(\mathsf{pre}(a),x)$;
`asegura CapacidadCorrecta` : $capacidad(a,d) == c$;
}

4. Competencia

```
tipo Competencia {
        observador categoria (c: Competencia) : (Deporte, Sexo) ;
        observador participantes (c: Competencia) : [Atleta] ;
        observador finalizada (c: Competencia) : Bool ;
        observador ranking (c: Competencia) : [Atleta] ;
             requiere finalizada(c) ;
        observador lesTocoControlAntiDoping (c: Competencia) : [Atleta] ;
             requiere finalizada(c) ;
        observador leDioPositivo (c: Competencia, a: Atleta) : Bool ;
             requiere finalizada(c) ∧ a ∈ lesTocoControlAntiDoping(c) ;

        invariante participaUnaSolaVez : sinRepetidos(ciaNumbers(participantes(c))) ;
        invariante participantesPertenecenACat :
            (∀p ← participantes(c))prm(categoria(c)) ∈ deportes(p) ∧ sgd(categoria(c)) == sexo(p) ;
        invariante elRankingEsDeParticipantesYNoHayRepetidos :
            finalizada(c) ⇒ incluida(ranking(c), participantes(c)) ;
        invariante seControlanParticipantesYNoHayRepetidos :
            finalizada(c) ⇒ incluida(lesTocoControlAntiDoping(c), participantes(c)) ;
}

problema finalizarCompetencia (c: Competencia, posiciones: [Atleta], control: [(Atleta, Bool)]) {
        requiere ¬finalizada(c) ;
        requiere PosicionesCumplenInvariante : incluida(posiciones, participantes(c)) ;
        requiere ControladosCumplenInvariante : incluida(atletasControlados(control), participantes(c)) ;

        modifica c ;

        asegura (categoria(c) == categoria(pre(c))) ∧ mismos(participantes(c), participantes(pre(c))) ;
        asegura finalizada(c) ;
        asegura RankingCorrecto : ranking(c) == posiciones ;
        asegura ControlCorrecto : mismos(lesTocoControlAntiDoping(c), atletasControlados(control)) ;
        asegura ResultadoAntiDopingCorrecto : (∀a ← control)sgd(a) == leDioPositivo(c, prm(a)) ;

        aux atletasControlados (c:[(Atleta,Bool)]) : [Atleta] = [ prm(a) | a ← c ] ;
}

problema linfordChristie (c: Competencia, a: Atleta) {
        requiere ¬finalizada(c) ;
        requiere a ∈ participantes(c) ;

        modifica c ;

        asegura ¬finalizada(c) ∧ categoria(c) == categoria(pre(c)) ;
        asegura AtletaDescalificado : mismos(participantes(pre(c)), a : participantes(c)) ;
}

problema gananLosMasCapaces (c: Competencia) = res : Bool {
        requiere finalizada(c) ;

        asegura res == (∀i ← [1.. |ranking(c)| − 1))
            capacidadCompetencia(c, ranking(c)_i) ≥ capacidadCompetencia(c, ranking(c)_{i+1}) ;

        aux capacidadCompetencia (c:Competencia,a:Atleta) : ℤ = capacidad(a, prm(categoria(c))) ;
}

problema sancionarTramposos (c: Competencia) {
        requiere finalizada(c) ;

        modifica c ;

        asegura finalizada(c) ∧ categoria(c) == categoria(pre(c)) ∧ mismos(participantes(c), participantes(pre(c))) ;
        asegura ranking(c) == rankingSinTramposos(pre(c)) ;
        asegura mismos(lesTocoControlAntiDoping(c), lesTocoControlAntiDoping(pre(c))) ;
        asegura (∀a ← lesTocoControlAntiDoping(c))leDioPositivo(c, a) == leDioPositivo(pre(c), a) ;

        aux rankingSinTramposos (c:Competencia) : [Atleta] = [ a | a ← ranking(c),
            a ∈ lesTocoControlAntiDoping(c) → leDioPositivo(c, a) == False ] ;
}
```

5. JJOO

```
tipo JJOO {
        observador año (j: JJOO) : ℤ ;
        observador atletas (j: JJOO) : [Atleta] ;
        observador cantDias (j: JJOO) : ℤ ;
        observador cronograma (j: JJOO, dia: ℤ) : [Competencia] ;
             requiere 1 ≤ dia ≤ cantDias(j) ;
        observador jornadaActual (j: JJOO) : ℤ ;

        invariante atletasUnicos : sinRepetidos(ciaNumbers(atletas(j))) ;
        invariante unaDeCadaCategoria : (∀i, k ← [0..|competencias(j)|), i ≠ k)
            categoria(competencias(j)_i) ≠ categoria(competencias(j)_k) ;
        invariante competidoresInscriptos : (∀c ← competencias(j))incluida(participantes(c), atletas(j)) ;
        invariante jornadaValida : 1 ≤ jornadaActual(j) ≤ cantDias(j) ;
        invariante finalizadasSiiYaPasoElDia : lasPasadasFinalizaron(j) ∧ lasQueNoPasaronNoFinalizaron(j) ;
}

problema dePaseo (j: JJOO) = res : [Atleta]  {
        asegura mismos(res, noParticipan(j)) ;
        aux noParticipan (j:JJOO) : [Atleta] = [ a | a ← atletas(j), ¬participa(j, a) ] ;
}

problema medallero (j: JJOO) = res : [(Pais, [ℤ])]  {
        asegura OrdenadoPorOro : (∀i ← [0..|res| − 1))sgd(res_i)_0 ≥ sgd(res_{i+1})_0 ;
        asegura OrdenadoPorPlata : (∀i ← [0..|res| − 1), sgd(res_i)_0 == sgd(res_{i+1})_0)sgd(res_i)_1 ≥ sgd(res_{i+1})_1 ;
        asegura OrdenadoPorBronce : (∀i ← [0..|res| − 1), sgd(res_i)_1 == sgd(res_{i+1})_1)sgd(res_i)_2 ≥ sgd(res_{i+1})_2 ;
        asegura NoHayPaisesRepetidos : sinRepetidos([ prm(p) | p ← res ]) ;
        asegura TodosGanaronMedallas : (∀p ← res) ∑ sgd(p) > 0 ;
        asegura EstanTodosLosQueGanaron : (∀p ← paises(j), totalMedallasPais(j, p) > 0)((∃q ← res)prm(q) == p) ;
        asegura CantidadCorrectaDeMedallas : (∀p ← res)
            ((totalOroPais(j, prm(p)) == sgd(p)_0) ∧ (totalPlataPais(j, prm(p)) == sgd(p)_1)
            ∧ (totalBroncePais(j, prm(p)) == sgd(p)_2)) ;

        aux totalOroPais (j:JJOO,p:Pais) : ℤ = |[ c | c ← competencias(j), finalizada(c) ∧ ganoOroPais(c, p) ]| ;
        aux totalPlataPais (j:JJOO,p:Pais) : ℤ = |[ c | c ← competencias(j), finalizada(c) ∧ ganoPlataPais(c, p) ]| ;
        aux totalBroncePais (j:JJOO,p:Pais) : ℤ = |[ c | c ← competencias(j), finalizada(c) ∧ ganoBroncePais(c, p) ]| ;
        aux totalMedallasPais (j:JJOO,p:Pais) : ℤ = totalOroPais(j, p) + totalPlataPais(j, p) + totalBroncePais(j, p) ;
        aux ganoOroPais (c:Competencia,p:Pais) : Bool = |ranking(c)| ≥ 1 ∧ nacionalidad(ranking(c)_0) == pais ;
        aux ganoPlataPais (c:Competencia,p:Pais) : Bool = |ranking(c)| ≥ 2 ∧ nacionalidad(ranking(c)_1) == pais ;
        aux ganoBroncePais (c:Competencia,p:Pais) : Bool = |ranking(c)| ≥ 3 ∧ nacionalidad(ranking(c)_2) == pais ;
}

problema boicotPorDisciplina (j: JJOO, cat: (Deporte, Sexo), p: Pais) = res : ℤ  {
        requiere CategoriaValida : (∃c ← competencias(j))categoria(c) == cat ;

        modifica j ;

        asegura año(j) == año(pre(j))∧cantDias(j) == cantDias(pre(j))∧jornadaActual(j) == jornadaActual(pre(j))∧
            mismos(atletas(j), atletas(pre(j))) ;
        asegura NoModificaOtrasCompetencias : (∀d ← [1..cantDias(j)])
            ((∀c_1 ← cronograma(pre(j), d), categoria(c_1) ≠ cat)((∃c_2 ← cronograma(j, d))mismaCompetencia(c_1, c_2))) ;
        asegura NoCreaCompetenciasNuevas : (∀d ← [1..cantDias(j)])
            ((∀c_1 ← cronograma(j, d), categoria(c_1) ≠ cat)((∃c_2 ← cronograma(pre(j), d))mismaCompetencia(c_1, c_2))) ;
        asegura (∀d ← [1..cantDias(j)]) |cronograma(j, d)| == |cronograma(pre(j), d)| ;
        asegura CatFueModificada : (∃c_1 ← competencias(j), categoria(c_1) == cat)
            ((∃c_2 ← competencias(pre(j)), categoria(c_2) == cat)
            (mismos(participantes(c_1), sacarPais(p, participantes(c_2))))
            ∧ (finalizada(c_1) == finalizada(c_2))
            ∧ (finalizada(c_1) → ((ranking(c_1) == sacarPais(p, ranking(c_2)))
            ∧ (mismos(lesTocoControlAntiDoping(c_1), sacarPais(p, lesTocoControlAntiDoping(c_2))))
            ∧ ((∀a ← lesTocoControlAntiDoping(c_1))leDioPositivo(c_1, a) == leDioPositivo(c_2, a))))) ;
        asegura ResultadoCorrecto : res == |[ a | a ← atletas(j), nacionalidad(a) == p ∧ participa(j, a) ]| ;

        aux sacarPais (p:Pais,l:[Atleta]) : [Atleta] = [ a | a ← l, nacionalidad(a) ≠ p ] ;
```

aux mismaCompetencia (c_1,c_2:Competencia) : Bool $= (categoria(c_1) == categoria(c_2))$
$\wedge\ (mismos(participantes(c_2), participantes(c2))) \wedge (finalizada(c_1) == finalizada(c_2))$
$\wedge\ (finalizada(c_1) \rightarrow ((ranking(c_1) == ranking(c_2))$
$\wedge\ (mismos(lesTocoControlAntiDoping(c_1), lesTocoControlAntiDoping(c_2))$
$\wedge\ ((\forall a \leftarrow lesTocoControlAntiDoping(c_1))leDioPositivo(c_1, a) == leDioPositivo(c_2, a))))$;
}

problema losMasFracasados (j: JJOO, p: Pais) = res : [Atleta] {
asegura Participaron : $(\forall a \leftarrow res)a \in losQueMasParticiparon(j, p)$;
asegura NoGanaronNada : $(\forall a \leftarrow res)totalMedallasAtleta(j, a) == 0$;
asegura EstanTodos : $(\forall a \leftarrow losQueMasParticiparon(j, p), totalMedallasAtleta(j, a) == 0)a \in res$;

aux losQueMasParticiparon (j:JJOO,p:Pais) : [Atleta] $= [\, a \mid a \leftarrow atletas(j), nacionalidad(a) == p$
$\wedge\ ((\forall b \leftarrow atletas(j), nacionalidad(b) == p)cantCompetencias(j, a) \geq cantCompetencias(j, b))\,]$;
aux cantCompetencias (j:JJOO,a:Atleta) : $\mathbb{Z} = |[\, c \mid c \leftarrow competencias(j), a \in participantes(c)\,]|$;
aux totalMedallasAtleta (j:JJOO,a:Atleta) : $\mathbb{Z} =$
$|[\, c \mid c \leftarrow competencias(j), finalizada(c) \wedge ganoMedallaAtleta(c, a)\,]|$;
aux ganoMedallaAtleta (c:Competencia,a:Atleta) : Bool $= (|ranking(c)| \geq 1 \wedge ranking(c)_0 == a)$
$\vee\ (|ranking(c)| \geq 2 \wedge ranking(c)_1 == a) \vee (|ranking(c)| \geq 3 \wedge ranking(c)_2 == a)$;
}

problema liuSong (j: JJOO, a: Atleta, p: País) {
requiere $a \in atletas(j)$;

modifica j ;

asegura $año(j) == año(\mathsf{pre}(j)) \wedge cantDias(j) == cantDias(\mathsf{pre}(j)) \wedge jornadaActual(j) == jornadaActual(\mathsf{pre}(j))$;
asegura SoloCambioElAtletaDeseado : $(\forall d \leftarrow [1..cantDias(j)])$
$((\forall c_1 \leftarrow cronograma(\mathsf{pre}(j), d))((\exists c_2 \leftarrow cronograma(j, d))mismaCompNacCamb(c_1, c_2, a, p))$
$\wedge\ |cronograma(j, d)| == |cronograma(\mathsf{pre}(j), d)|)$;

aux mismoNacCamb (a_1,a_2:Atleta,p:Pais) : Bool $= nombre(a_1) == nombre(a_2) \wedge sexo(a_1) == sexo(a_2)$
$\wedge\ añoNacimiento(a_1) == añoNacimiento(a_2) \wedge nacionalidad(a_2) == p \wedge ciaNumber(a_1) == ciaNumber(a_2)$
$\wedge\ deportes(a_1) == deportes(a_2) \wedge ((\forall d \leftarrow deportes(a_1))capacidad(a_1, d) == capacidad(a_2, d))$;
aux mismosAtlNacCamb (l_1,l_2:[Atleta],a:Atleta,p:Pais) : Bool $=$
$((\forall a_1 \leftarrow l_1, ciaNumber(a_1) \neq ciaNumber(a))((\exists a_2 \leftarrow l_2)a_1 == a_2))$
$\wedge\ ((\exists a_1 \leftarrow l_1, ciaNumber(a_1) == ciaNumber(a_2)) \rightarrow ((\exists a_2 \leftarrow l_2)mismosNacCamb(a_1, a_2, p)))$
$\wedge\ |l_1| == |l_2|$;
aux mismaCompNacCamb (c_1,c_2:Competencia,a:Atleta,p:Pais) : Bool $=$
$categoria(c_1) == categoria(c_2) \wedge mismosAtlNacCamb(participantes(c_1), participantes(c_2), a, p)$
$\wedge\ finalizada(c_1) == finalizada(c_2)$
$\wedge\ (finalizada(c_1) \rightarrow (mismosAtlNacCamb(ranking(c_1), ranking(c_2), a, p)$
$\wedge\ mismoOrden(ranking(c_1), ranking(c_2))$
$\wedge\ mismosAtlNacCamb(lesTocoControlAntiDoping(c_1), lesTocoControlAntiDoping(c_2), a, p)$
$\wedge\ ((\forall b \leftarrow lesTocoControlAntiDoping(c_1), ciaNumber(b) \neq ciaNumber(a))$
$leDioPositivo(c_1, b) == leDioPositivo(c_2, b))$
$\wedge((\exists a_1 \leftarrow lesTocoControlAntiDoping(c_1), a_2 \leftarrow lesTocoControlAntiDoping(c_2))mismoNacCamb(a_1, a_2, p)) \rightarrow$
$leDioPositivo(c_1, a_1) == leDioPositivo(c_2, a_2)))$;
aux mismoOrden (l,s:[Atleta]) : Bool $= (\forall i \leftarrow [0..|l|))ciaNumber(l_i) == ciaNumber(s_i)$;
}

problema stevenBradbury (j: JJOO) = res : Atleta {
asegura $res \in ganaronOro(j)$;
asegura MenosCapaz : $(\forall a \leftarrow ganaronOro(j))menorCapOro(j, a) \geq menorCapOro(j, res)$;

aux menorCapOro (j:JJOO,a:Atleta) : $\mathbb{Z} = minimo(capacidadOro(j, a))$;
aux capacidadOro (j:JJOO,a:Atleta) : $[\mathbb{Z}] = [\, capacidad(a, d) \mid d \leftarrow deportesOro(j, a)\,]$;
aux deportesOro (j:JJOO,a:Atleta) : [Deporte] $=$
$[\, prm(categoria(c)) \mid c \leftarrow competencias(j), finalizada(c) \wedge ganoOroAtleta(c, a)\,]$;
aux ganoOroAtleta (c:Competencia,a:Atleta) : Bool $= |ranking(c)| \geq 1 \wedge ranking(c)_0 == a$;
}

problema uyOrdenadoAsíHayUnPatrón (j: JJOO) = res : Bool {
}

problema sequíaOlímpica (j: JJOO) = res : [País] {
}

`problema transcurrirDia` (j: JJOO) {
}

6. Auxiliares

`aux ciaNumbers` (as: [Atleta]) : $[\mathbb{Z}] = [\, ciaNumber(a) \mid a \leftarrow as \,]$;
`aux competencias` (j: JJOO) : [Competencia] $= [\, c \mid d \leftarrow [1..cantDias(j)], c \leftarrow cronograma(j,d) \,]$;
`aux incluida` (l_1, l_2:[T]) : Bool $= (\forall x \leftarrow l_1) cuenta(x, l_1) \leq cuenta(x, l_2)$;
`aux lasPasadasFinalizaron` (j: JJOO) : Bool $= (\forall d \leftarrow [1..jornadaActual(j)))(\forall c \leftarrow cronograma(j,d)) finalizada(c)$;
`aux lasQueNoPasaronNoFinalizaron` (j: JJOO) : Bool $=$
$(\forall d \leftarrow (jornadaActual(j)..cantDias(j)])(\forall c \leftarrow cronograma(j,d)) \neg finalizada(c)$;
`aux ordenada` (l:[T]) : Bool $= (\forall i \leftarrow [0..|l|-1)) l_i \leq l_{i+1}$;
`aux sinRepetidos` (l: [T]) : Bool $= (\forall i, j \leftarrow [0..|l|), i \neq j) l_i \neq l_j$;