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

Algoritmos y Estructura de Datos I

Segundo cuatrimestre de 2013

27 de septiembre de 2013

Especificación - Cine

1. Tipos

`tipo` Sala = $\mathbb{Z}$;
`tipo` Nombre = String;
`tipo` Actor = String;
`tipo` Espectadores = $\mathbb{Z}$;
`tipo` Es3D = Bool;
`tipo` Genero = Aventura, Comedia, Drama, Romántica, Terror;

2. Película

`tipo Pelicula` {
  `observador nombre` (p: Pelicula) : Nombre;
  `observador géneros` (p: Pelicula) : [Género];
  `observador actores` (p: Pelicula) : [Actor];
  `observador es3D` (p: Pelicula) : Bool;
  `invariante sinActoresRepetidos` : $sinRepetidos(actores(p))$;
  `invariante sinGénerosRepetidos` : $sinRepetidos(generos(p))$;
  `invariante génerosOrdenados` : $listaOrdenada(generos(p))$;
  `invariante actoresOrdenados` : $listaOrdenada(actores(p))$;
}

`problema nuevaP` (n: Nombre, gs: [Género], as: [Actor], b: Bool) = `result` : Pelicula {
  `requiere` $listaOrdenada(gs) \wedge listaOrdenada(as)$;
  `asegura` $nombre(result) == n \wedge es3D(result) == b$;
  `asegura` $((\forall g \leftarrow gs)g \in generos(result)) \wedge ((\forall g \leftarrow generos(result))g \in gs)$;
  `asegura` $((\forall a \leftarrow as)a \in actores(result)) \wedge ((\forall a \leftarrow actores(result))a \in as)$;
}

`problema nombreP` (p: Pelicula) = `result` : Nombre {
  `asegura` $result == nombre(p)$;
}

`problema generosP` (p: Pelicula) = `result` : [Género] {
  `asegura` $result == generos(p)$;
}

`problema actoresP` (p: Pelicula) = `result` : [Actor] {
  `asegura` $result == actores(p)$;
}

`problema es3DP` (p: Pelicula) = `result` : Bool {
  `asegura` $result == es3D(p)$;
}

`problema agruparPelisPorGeneroP` (ps: [Pelicula]) = `result` : [(Genero, [Pelicula])] {
  `requiere sinPelisRepetidas` : $sinRepetidos(ps)$;
  `asegura unGrupoPorGenero` : $sinRepetidos(primeros(result))$;
  `asegura sinCopiasEnGrupos` : $(\forall d \leftarrow result)sinRepetidos(sgd(d))$;
  `asegura estanTodas` : $(\forall p \leftarrow ps, g \leftarrow generos(p))(\exists d \leftarrow result)prm(d) == g \wedge p \in sgd(d)$;
  `asegura noHayDeMas` : $(\forall d \leftarrow result)(|sgd(d)| > 0 \wedge (\forall p \leftarrow sgd(d))p \in ps \wedge prm(d) \in generos(p))$;
}

`problema generarSagaDePeliculasP` (as:[Actor], gs:[Genero], nombres:[Nombre]) = `result` : [Pelicula] {

  requiere nombresDistintos : $sinRepetidos(nombres)$;
  requiere ordenadas : $listaOrdenada(as) \land listaOrdenada(gs)$;
  asegura mismaLongitud : $|result| == |nombres|$;
  asegura unaPeliPorNombre : $(\forall n \leftarrow nombres)(\exists p \leftarrow result) nombre(p) == n$
    $\land\ mismos(generos(p), sacarRepetidos(gs)) \land mismos(actores(p), sacarRepetidos(as))$;
}

3. Ticket

tipo Ticket {
  observador película (t: Ticket) : Pelicula ;
  observador sala (t: Ticket) : Sala ;
  observador usado (t: Ticket) : Bool ;
}

problema nuevoT (p: Pelicula, s: Sala, u: Bool) = result : Ticket {
  asegura $pelicula(result) == p \land sala(result) == s \land usado(result) == u$;
}

problema peliculaT (t: Ticket) = result : Pelicula {
  asegura $result == pelicula(p)$;
}

problema salaT (t: Ticket) = result : Sala {
  asegura $result == sala(p)$;
}

problema usadoT (t: Ticket) = result : Bool {
  asegura $result == usado(p)$;
}

problema usarT (t: Ticket) = result : Ticket {
  asegura mismoTicket : $pelicula(result) == pelicula(t) \land sala(result) == sala(t)$;
  asegura ahoraUsado : $usado(result)$;
}

problema peliculaMenosVistaT (ts:[Ticket]) = result : Pelicula {
  requiere alMenosUna : $|ts| > 0$;
  asegura esValida : $(\exists t \leftarrow ts) pelicula(t) == result$;
  asegura esLaMenosVista : $(\forall t \leftarrow ts) sumaUsado(result, ts) \leq sumaUsado(t, ts)$;
}

problema todosLosTicketsParaLaMismaSalaT (ts:[Ticket]) = result : Bool {
  asegura $result \leftrightarrow (\forall i \leftarrow [0..|ts|), j \leftarrow [0..|ts|)) sala(ts_i) == sala(ts_j)$;
}

problema cambiarSalaT (ts:[Ticket], vieja : Sala, nueva : Sala) = result : [Ticket] {
  asegura mismaLongitud : $|ts| == |result|$;
  asegura otrasNoCambian : $(\forall i \leftarrow [0...|ts|), sala(ts_i) \neq vieja) result_i == ts_i$;
  asegura viejaReemplazada : $(\forall i \leftarrow [0...|ts|), sala(ts_i) == vieja) pelicula(result_i) == pelicula(ts_i) \land$
    $usado(result_i) == usado(ts_i) \land sala(result_i) == nueva$;
}

4. Cine

tipo Cine {
  observador nombre (c: Cine) : Nombre ;
  observador películas (c: Cine) : [Peliculas] ;
  observador salas (c: Cine) : [Sala] ;
  observador sala (c: Cine, p: Pelicula) : Sala ;
      requiere $p \in peliculas(c)$;
  observador espectadores (c: Cine, s: Sala) : $\mathbb{Z}$;
      requiere $s \in salas(c)$;

```
  observador ticketsVendidosSinUsar (c: Cine) : [Ticket] ;
  invariante sinPeliculasRepetidas : sinRepetidos(nombresDePeliculas(c)) ;
  invariante sinSalasRepetidas : sinRepetidos(salas(c)) ;
  invariante salasDeCineSonSalas : (∀p ← peliculas(c))sala(c, p) ∈ salas(c) ;
  invariante salasSinPeliSinEspectadores : (∀s ← salas(c))¬tienePeli(c, s) ⇒ espectadores(c, s) == 0 ;
  invariante espectadoresNoNegativos : (∀s ← salas(c))espectadores(c, s) ≥ 0 ;
  invariante salasConsistentes : (∀p ← peliculas(c), q ← peliculas(c))sala(c, p) == sala(c, q) ⇒ p == q ;
  invariante losTicketsVendidosSonParaPeliculasDelCine : (∀t ← ticketsVendidosSinUsar(c))ticketOK(t, c) ;
  invariante losTicketsVendidosEstanSinUsar : (∀t ← ticketsVendidosSinUsar(c))¬usado(t) ;
}

problema nuevoC (n: Nombre) = result : Cine  {
  asegura nombre(result) == n ∧ |salas(result)| == 0 ;
}

problema nombreC (c: Cine) = result : Nombre  {
  asegura result == nombre(c) ;
}

problema peliculasC (c: Cine) = result : [Peliculas]  {
  asegura result == peliculas(c) ;
}

problema salasC (c: Cine) = result : [Sala]  {
  asegura result == salas(c) ;
}

problema espectadoresC (c: Cine, s:Sala) = result : ℤ  {
  requiere s ∈ salas(c) ;
  asegura result == espectadores(c, s) ;
}

problema salaC (c: Cine, p:Pelicula) = result : Sala  {
  requiere p ∈ peliculas(c) ;
  asegura result == sala(c, p) ;
}

problema ticketsVendidosC (c: Cine) = result : [Ticket]  {
  asegura result == ticketsVendidosSinUsar(c) ;
}

problema abrirSalaC (c: Cine, s : Sala) = result : Cine  {
  requiere salaNueva : ¬(s ∈ salas(c)) ;
  asegura mismoNombre : nombre(result) == nombre(c) ;
  asegura abreSala : mismos(salas(result), s : salas(c)) ;
  asegura mismasPelis : mismos(peliculas(result), peliculas(c)) ;
  asegura pelisEnMismaSala : (∀p ← peliculas(c))sala(result, p) == sala(c, p) ;
  asegura mismosEspectadores : (∀s ← salas(c))espectadores(result, s) == espectadores(c, s) ;
  asegura mismosTickets : mismos(ticketsVendidosSinUsar(result), ticketsVendidosSinUsar(c)) ;
}

problema agregarPeliculaC (c:Cine, p:Pelicula, s:Sala) = result : Cine  {
  requiere salaVacante : s ∈ salas(c) ∧ ¬(∃p2 ← peliculas(c))sala(p2) == s ;
  requiere peliNueva : ¬(∃p2 ← peliculas(c))nombre(p2) == nombre(p) ;
  asegura mismoNombre : nombre(result) == nombre(c) ;
  asegura mismasSalas : mismos(salas(result), salas(c)) ;
  asegura agregaPeli : mismos(peliculas(result), p : peliculas(c)) ;
  asegura otrasPelisEnMismaSala : (∀p2 ← peliculas(c))sala(result, p2) == sala(c, p2) ;
  asegura mismosEspectadores : (∀s ← salas(result))espectadores(result, s) == espectadores(c, s) ;
  asegura mismosTickets : mismos(ticketsVendidosSinUsar(result), ticketsVendidosSinUsar(c)) ;
  asegura salaPedida : sala(result, p) == s ;
}

problema cerrarSalaC (c:Cine,s:Sala) = result : Cine  {
  requiere salaValida : s ∈ salas(c) ;
  requiere ceroDemanda : ¬(∃t ← ticketsVendidosSinUsar(c))sala(t) == s ;
```

asegura mismoNombre : $nombre(result) == nombre(c)$;
asegura sacarSala : $mismos(s : salas(result), salas(c))$;
asegura mismasPelis : $mismos(peliculas(result), [\, p \mid p \leftarrow peliculas(c), sala(c,p) \neq s \,])$;
asegura pelisEnMismaSala : $(\forall p \leftarrow peliculas(result))sala(result,p) == sala(c,p)$;
asegura mismosEspectadores : $(\forall s \leftarrow salas(result))espectadores(result,s) == espectadores(c,s)$;
asegura mismosTickets : $mismos(ticketsVendidosSinUsar(result), ticketsVendidosSinUsar(c))$;
}

problema cerrarSalasC (c:Cine, e:$\mathbb{Z}$) = result : Cine {
requiere espectadoresNoNegativos : $e \geq 0$;
requiere ceroDemanda : $\neg(\exists t \leftarrow ticketsVendidosSinUsar(c))espectadores(c, sala(t)) < e$;
asegura respetaCotaDeEspectadores : $igualesCerrandoLasMenosVistas(c, result, e)$;
}

problema cerrarSalasDeLaCadenaC (cs:[Cine], e:$\mathbb{Z}$) = result : [Cine] {
requiere espectadoresNoNegativos : $e \geq 0$;
requiere ceroDemanda : $\neg(\exists c \leftarrow cs, t \leftarrow ticketsVendidosSinUsar(c))espectadores(c, sala(t)) < e$;
asegura mismaCantidad : $|result| == |cs|$;
asegura mismosCinesRespetandoCota : $(\forall c1 \leftarrow cs)(\exists c2 \leftarrow result)igualesCerrandoLasMenosVistas(c1, c2, e)$
$\wedge\, cuentaCines(c1, cs) == cuentaCines(c2, result)$;
}

problema peliculaC (c:Cine, s:Sala) = result : Pelicula {
requiere salaValida : $s \in salas(c) \wedge (\exists p \leftarrow peliculas(c))sala(c,p) == s$;
asegura esLaPeliDeLaSala : $result \in peliculas(c) \wedge sala(c, result) == s$;
}

problema venderTicketC (c:Cine, p:Pelicula) = result : (Cine,Ticket) {
requiere esPeliDelCine : $p \in peliculas(c)$;
asegura mismoNombre : $nombre(prm(result)) == nombre(c)$;
asegura mismasSalas : $mismos(salas(prm(result)), salas(c))$;
asegura mismasPelis : $mismos(peliculas(prm(result)), peliculas(c))$;
asegura pelisEnMismaSala : $(\forall p \leftarrow peliculas(c))sala(prm(result), p) == sala(c,p)$;
asegura mismosEspectadores : $(\forall s \leftarrow salas(c))espectadores(prm(result), s) == espectadores(c,s)$;
asegura ticketParaLaPeli : $pelicula(sgd(result)) == p \wedge \neg usado(sgd(result))$;
asegura ticketEnLaLista : $mismos(ticketsVendidosSinUsar(prm(result)), sgd(result) : ticketsVendidosSinUsar(c))$;
}

problema ingresarASalaC (c:Cine, s:Sala, t:Ticket) = result : (Cine,Ticket) {
requiere salaConsistente : $s == sala(t)$;
requiere ticketValido : $t \in ticketsVendidosSinUsar(c)$;
asegura mismoNombre : $nombre(prm(result)) == nombre(c)$;
asegura mismasSalas : $mismos(salas(prm(result)), salas(c))$;
asegura mismasPelis : $mismos(peliculas(prm(result)), peliculas(c))$;
asegura pelisEnMismaSala : $(\forall p \leftarrow peliculas(prm(result)))sala(prm(result), p) == sala(c,p)$;
asegura mismosEspectadores : $(\forall s2 \leftarrow salas(prm(result)), s2 \neq s)espectadores(prm(result), s) == espectadores(c,s)$;
asegura unEspectadorMas : $espectadores(prm(result), s) == espectadores(c,s) + 1$;
asegura usaElTicket : $mismos(t : ticketsVendidosSinUsar(prm(result)), ticketsVendidosSinUsar(c))$;
asegura elTicketSeUso : $sala(sgd(result)) == sala(t) \wedge pelicula(sgd(result)) == pelicula(t) \wedge usado(sgd(result))$;
}

problema pasarA3DUnaPeliculaC (c:Cine, nombre:Nombre) = result : (Cine,Pelicula) {
requiere estaLaPeli : $(\exists p \leftarrow peliculas(c))nombre(p) == nombre$;
asegura mismoNombre : $nombre(prm(result)) == nombre(c)$;
asegura mismasSalas : $mismos(salas(prm(result)), salas(c))$;
asegura laPeliEs3D : $igualExcepto3D(sgd(result), peliConNombre(c, nombre)) \wedge es3D(sgd(result))$;
asegura cambiaPeliA3D : $mismos(peliConNombre(c, nombre) : peliculas(prm(result)), sgd(result) : peliculas(c))$;
asegura pelisEnMismaSala : $(\forall p \leftarrow peliculas(c), nombre(p) \neq nombre)sala(prm(result), p) == sala(c,p)$;
asegura esaPeliEnMismaSala : $sala(prm(result), sgd(result)) == sala(c, peliConNombre(c, nombre))$;
asegura mismosEspectadores : $(\forall s \leftarrow salas(c))espectadores(prm(result), s) == espectadores(c,s)$;
asegura mismosTickets : $(\forall s \leftarrow salas(c))$
$contarSala(s, ticketsVendidosSinUsar(prm(result))) == contarSala(s, ticketsVendidosSinUsar(c))$;
}

5. Auxiliares

aux sinRepetidos (l: [T]) : Bool $= (\forall i, j \leftarrow [0..|l|), i \neq j) l_i \neq l_j$;
aux listaOrdenada (l: [T]) : Bool $= (\forall i \leftarrow [0..|l|-1)) l_i \leq l_{i+1}$;
aux ticketOK (t: Ticket, c:Cine) : Bool $= (\exists p \leftarrow peliculas(c), s \leftarrow salas(c)) p == pelicula(t) \wedge s == sala(t) \wedge sala(c,p) == s$;
aux primeros (ls : [(T_1, T_2)]) : [T_1] $= [\, prm(x) \mid x \leftarrow ls \,]$;
aux sacarRepetidos (l: [T]) : [T] $= [\, l_i \mid i \leftarrow [0..|l|), \neg(l_i \in l[0..i)) \,]$;
aux tienePeli (c: Cine, s:Sala) : Bool $= (\exists p \leftarrow pelicula(c)) sala(c,p) == s$;
aux sumaUsado (p : Pelicula, ts : [Ticket]) : $\mathbb{Z} = |[\, 1 \mid t \leftarrow ts, pelicula(t) == p \wedge usado(t) \,]|$;
aux cuentaCines (c:Cine,cs:[Cine]) : $\mathbb{Z} = |[\, 1 \mid c2 \leftarrow cs, cinesIguales(c,c2) \,]|$;
aux cinesIguales (c1,c2:Cine) : Bool $= nombre(c1) == nombre(c2) \wedge mismos(salas(c1), salas(c2))$
$\wedge\, mismos(peliculas(c1), peliculas(c2)) \wedge (\forall p \leftarrow peliculas(c1)) sala(c1,p) == sala(c2,p)$
$\wedge\, (\forall s \leftarrow salas(c1)) espectadores(c1,s) == espectadores(c2,s)$
$\wedge\, mismos(ticketsVendidosSinUsar(c1), ticketsVendidosSinUsar(c2))$;
aux igualesCerrandoLasMenosVistas (c1,c2:Cine,e:$\mathbb{Z}$) : Bool $= nombre(c1) == nombre(c2)$
$\wedge\, mismos(salas(c2), [\, s \mid s \leftarrow salas(c1), espectadores(c1,s) \geq e \,]$
$\wedge\, mismos(peliculas(c2), [\, p \mid p \leftarrow peliculas(c1), espectadores(c1, sala(c1,p)) \geq e \,])$
$\wedge\, (\forall p \leftarrow peliculas(c2)) sala(c2,p) == sala(c1,p)$
$\wedge\, (\forall s \leftarrow salas(c2)) espectadores(c2,s) == espectadores(c1,s)$
$\wedge\, mismos(ticketsVendidosSinUsar(c1), ticketsVendidosSinUsar(c2))$;
aux peliConNombre (c:Cine,n:Nombre) : Pelicula $= [\, p \mid p \leftarrow peliculas(c), nombre(p) == n \,]_0$;
aux igualExcepto3D (p1,p2 : Pelicula) : Bool $= nombre(p1) == nombre(p2) \wedge mismos(generos(p1), generos(p2))$
$\wedge\, mismos(actores(p1) actores(p2))$;
aux contarSala (s:Sala, ts:[Ticket]) : $\mathbb{Z} = |[\, 1 \mid t \leftarrow ts, sala(t) == s \,]|$;