

## Algoritmos y Estructura de Datos I

Segundo cuatrimestre de 2013

8 de septiembre de 2013

## TPE Cine

```

tipo Actor = String;
tipo Sala =  $\mathbb{Z}$ ;
tipo género = Aventura, Comedia, Drama, Romántica, Terror;

tipo Pelicula {
  observador nombre (p: Pelicula) : String;
  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 : generosOrd(generos(p));
  invariante actoresOrdenados : actoresOrd(actores(p));
}

problema agruparPelisPorGenero (ps:[Pelicula]) = result : [(Genero, [Pelicula])] {
  asegura generoSalidaEntrada : ( $\forall \text{dupla} \leftarrow \text{result}$ ) prm(dupla)  $\in$  obtenerGeneros(ps);
  asegura pelisSalidaEntrada : ( $\forall \text{dupla} \leftarrow \text{result}, \text{peli} \leftarrow \text{sgd}(\text{dupla})$ ) peli  $\in$  ps;
  asegura pelisSalidaSinRepetir : ( $\forall \text{dupla} \leftarrow \text{result}$ ) sinRepetir(sgd(dupla));
  asegura generosSalidaEnPelis : ( $\forall \text{dupla} \leftarrow \text{result}, \text{peli} \leftarrow \text{sgd}(\text{dupla})$ ) prm(dupla)  $\in$  generos(peli);
  asegura generosSalidaSinRepetir : sinRepetidos(obtenerGenerosDupla(result));
  asegura pelisEntradaEnSalida : ( $\forall \text{peli} \leftarrow \text{ps}, \text{dupla} \leftarrow \text{result}, \text{prm}(\text{dupla}) \in \text{generos}(\text{peli})$ ) peli  $\in$  sgd(dupla);
  asegura generosEntradaEnSalida : ( $\forall \text{genero} \leftarrow \text{obtenerGeneros}(\text{ps})$ ) genero  $\in$  obtenerGenerosDupla(result);
  aux obtenerGeneros (ps: [Pelicula]) : [Genero] = [genero(p) | p  $\leftarrow$  ps];
  aux obtenerGenerosDupla (x: [(Genero, [Pelicula])]) : [Genero] = [prm(dupla) | dupla  $\leftarrow$  x];
}

problema generarSagaDePeliculas (as:[Actor], gs:[Genero], nombres:[String]) = result : [Pelicula] {
  asegura mismosGeneros : ( $\forall \text{peli} \leftarrow \text{result}$ ) mismos(generos(peli), elementosSinRepetir(gs));
  asegura mismosActores : ( $\forall \text{peli} \leftarrow \text{result}$ ) mismos(actores(peli), elementosSinRepetir(as));
  asegura todosNombreEnNombres : mismos(listaNombrePelis(res), elementosSinRepetir(nombres));
  aux listaNombrePelis (listaPelis: [Pelicula]) : [String] = [nombre(peli) | peli  $\leftarrow$  listaPelis];
  aux elementosSinRepetir (s: [T]) : [T] = [si | i  $\leftarrow$  [0..s]],  $\neg(\exists j \leftarrow [0..s], j  $\neq$  i) si == sj];
}

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

problema películaMenosVista (ts : [Ticket]) = result : Bool {
  requiere ts  $\neq$  [];
  requiere ticketsUsados(ts)  $\neq$  [];
  asegura peliEstaEnTickets : result  $\in$  peliculasVistas(ticketsUsados(ts));
  asegura esLaPeliMenosVista :  $\neg(\exists p \leftarrow \text{peliculasVistas}(\text{ticketsUsados}(\text{ts})) \text{ticketsPorPelicula}(p, \text{ticketsUsados}(\text{ts})) < \text{ticketsPorPelicula}(\text{result}, \text{ticketsUsados}(\text{ts})))$ ;
  aux ticketsUsados (s: [Ticket]) : [Ticket] = [x | x  $\leftarrow$  s, usado(x)];
  aux peliculasVistas (s: [Ticket]) : [Pelicula] = [pelicula(x) | x  $\leftarrow$  s];
  aux ticketsPorPelicula (p: Pelicula, tic: [Ticket]) :  $\mathbb{Z}$  = [t | t  $\leftarrow$  tic, pelicula(t) == p];
}

problema todosLosTicketsParaLaMismaSala (ts:[Ticket]) = result : Bool {
  asegura todosLosTicketsParaLaMismaSala : res == ( $\forall t \leftarrow [0..\text{ts}] - 1$ ) sala(tsi) == sala(tsi+1);
}$ 
```

```

problema cambiarSala (ts:[Ticket], vieja: Sala, nueva: Sala) {
  modifica ts;
  asegura mismoNumeroTickets : |ts| == |pre(ts)|;
  asegura cambioDeSala : ( $\forall i \leftarrow [0..|pre(ts)|], sala(pre(ts)_i) == vieja.sala(ts_i == nueva;$ 
  asegura mismasPelis : ( $\forall i \leftarrow [0..|pre(ts)|] pelicula(ts_i) == pelicula(pre(ts)_i);$ 
  asegura ticketsSiguenUsados : ( $\forall i \leftarrow [0..|pre(ts)|] usado(ts_i) == usado(pre(ts)_i);$ 
  asegura mismasSalas : ( $\forall i \leftarrow [0..|pre(ts)|], sala(pre(ts)_i \neq vieja.sala(ts_i) == sala(pre(ts)_i);$ 
}

tipo Cine {
  observador nombre (c: Cine) : String;
  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 : ( $\forall p \leftarrow peliculas(c) sala(c, p) \in salas(c);$ 
  invariante espectadoresNoNegativos : ( $\forall s \leftarrow salas(c) espectadores(c, s) \geq 0;$ 
  invariante losTicketsVendidosEstanSinUsar : ( $\forall t \leftarrow ticketsVendidosSinUsar(c) \neg usado(t);$ 
  invariante salasConsistentes :  $sinRepetidos([sala(c, peli) | peli \leftarrow peliculas(c)]);$ 
  invariante losTicketsVendidosSonParaPeliculasDelCine : ( $\forall t \leftarrow ticketsVendidosSinUsar(c)$ 
     $pelicula(t) \in peliculas(c) \ \&\& \ sala(t) == sala(c, pelicula(t));$ 
}

problema cineVacio (n: String) = result : Cine {
  asegura nombreCineEntradaEsSalida :  $nombre(res) == n;$ 
  asegura salasCineVacias :  $salas(res) == [];$ 
}

problema agregarPelicula (c: Cine, p: Pelicula, s: Sala) {
  requiere salaEntradaEnCine :  $s \in salas(c);$ 
  requiere nuevaSalaVacía :  $\neg((\exists pe \leftarrow pelicula(c)) sala(c, pe) == s);$ 
  requiere nuevaPelicula :  $p \notin peliculas(c);$ 
  modifica c;
  asegura peliculasSeMantienen :  $mismos(peliculas(pre(c)) ++ [p], peliculas(c));$ 
  asegura salasSeMantienen :  $mismos(salas(c), salas(pre(c));$ 
  asegura nuevaPeliEnCorrectaSala :  $sala(c, p) == s;$ 
  asegura nombresSeMantienen :  $nombre(c) == nombre(pre(c));$ 
  asegura salaParaTodasPeliculasSeMantiene : ( $\forall p \leftarrow peliculas(pre(c)) sala(c, p) == sala(pre(c), p);$ 
  asegura espectadoresSeMantienen : ( $\forall s \leftarrow salas(c) espectadores(c, p) == espectadores(pre(c), p);$ 
  asegura noSeVendieronMasTickets :  $mismos(ticketsVendidosSinUsar(c) == ticketsVendidosSinUsar(pre(c));$ 
}

problema cerrarSala (c: Cine, s: Sala) {
  requiere salaEsteEnElCine :  $s \in salas(c);$ 
  requiere noSeCierreSalaConTicketsVendidos : ( $\forall t \leftarrow ticketsVendidosSinUsar(c) sala(t) \neq s;$ 
  modifica c;
  asegura salasNuevas :  $mismos(salas(c), [sal | sal \leftarrow salas(pre(c)), sal \neq s]);$ 
  asegura nombreSeMantiene :  $nombre(c) == nombre(pre(c));$ 
  asegura peliculasSinLasDeSalasCerradas :  $mismos(peliculas(c), [peli | peli \leftarrow peliculas(pre(c)), peli \neq peliDeSala(s, c)]);$ 
  asegura salaParaTodasPeliculasSeMantiene : ( $\forall p \leftarrow peliculas(pre(c)) sala(c, p) == sala(pre(c), p);$ 
  asegura espectadoresSeMantienen : ( $\forall s \leftarrow salas(c) espectadores(c, p) == espectadores(pre(c), p);$ 
  asegura noSeVendieronMasTickets :  $mismos(ticketsVendidosSinUsar(c), ticketsVendidosSinUsar(pre(c));$ 
}

problema cerrarSalas (c: Cine, e:  $\mathbb{Z}$ ) {
  requiere  $e \geq 0;$ 
  modifica c;
  asegura seParecen( $c, pre(c), e$ );
}

```

```

problema cerrarSalasDeLaCadena (cs: [Cine], e:  $\mathbb{Z}$ ) {
  requiere  $e \geq 0$ ;
  modifica cs;
  asegura mismaListaDeCines :  $long(cs) == long(pre(cs))$ ;
  asegura cinesSeParecen :  $(\forall c' \leftarrow pre(cs))(\exists c \leftarrow cs) seParecen(c, c', e)$ ;
}

problema pelicula (c: Cine, s: Sala) = result : Pelicula {
  requiere salaEnElCine :  $s \in salas(c)$ ;
  requiere todasSalasConPeli :  $(\exists p \leftarrow peliculas(c)) sala(c, p) == s$ ;
  asegura peliResultadoEstaEnElCine :  $result \in peliculas(c)$ ;
  asegura aLaPeliResultadoLeCorrespondeSalaEntrada :  $sala(c, result) == s$ ;
}

problema venderTicket (c: Cine, p: Pelicula) = result : Ticket {
  requiere peliculaEnElCine :  $p \in peliculas(c)$ ;
  modifica c;
  asegura sumoTicketANoUsados :  $mismos(ticketsVendidosSinUsar(pre(c) ++ [result], ticketsVendidosSinUsar(pre(c)))$ ;
  asegura noCambiaNombreDelCine :  $nombre(c) == nombre(pre(c))$ ;
  asegura salasDelCineNoCambian :  $mismos(salas(pre(c)), salas(c))$ ;
  asegura peliculasDelCineNoCambian :  $mismos(peliculas(pre(c)), peliculas(c))$ ;
  asegura salasDeLasPelisNoCambian :  $(\forall pe \leftarrow peliculas(pre(c)) sala(pre(c), pe) == sala(c, pe)$ ;
  asegura espectadoresDeLasSalasNoCambian :  $(\forall sa \leftarrow salas(pre(c)) espectadores(pre(c), pe) == espectadores(c, pe)$ ;
}

problema ingresarASala (c: Cine, s: Sala, t: Ticket) {
  requiere salaEntradaEnCine :  $s \in salas(c)$ ;
  requiere salaTicketIgualEntrada :  $salta(t) == s$ ;
  requiere ticketEstaEntreLosNoUsados :  $t \in ticketsVendidosSinUsar(c)$ ;
  modifica c;
  modifica t;
  asegura peliculaDelTicketNoCambia :  $pelicula(t) == pelicula(pre(t))$ ;
  asegura salaDelTicketNoCambia :  $sala(t) == sala(pre(t))$ ;
  asegura marcarElTicketComoUsado :  $usado(t)$ ;
  asegura sumarUnEspectador :  $espectadores(c, s) == espectadores(pre(c), s) + 1$ ;
  asegura sacoElTicketDeSinUsar :  $mismos(ticketsVendidosSinUsar(c), [ti \mid ti \leftarrow ticketsVendidosSinUsar(pre(c)),$ 
     $ti \neq t])$ ;
  asegura salasNoCambian :  $mismos(salas(c), salas(pre(c)))$ ;
  asegura nombreDelCineNoCambia :  $nombre(c) == nombre(pre(c))$ ;
  asegura peliculasDelCineNoCambia :  $mismos(peliculas(c), peliculas(pre(c)))$ ;
  asegura pelisNoCambianSala :  $(\forall p \leftarrow peliculas(pre(c)) sala(c, p) == sala(pre(c), p)$ ;
  asegura espectadoresNoCambianDeOtrasSalas :  $(\forall sa \leftarrow salas(pre(c)), sa \neq s) espectadores(c, sa) == espectadores(pre(c), sa)$ ;
}

problema pasarA3DUnaPelicula (c: Cine, nombre: String) = result : Pelicula {
  requiere laPeliculaExiste :  $(\exists p \leftarrow peliculas(c)) nombre(p) == nombre$ ;
  modifica c;
  asegura nombreResultadoIgualEntrada :  $nombre(result) == nombre$ ;
  asegura peli3D :  $es3D(result)$ ;
  asegura generoResultadoIgualEntrada :  $generos(result) == generos(peliculaDeNombre(pre(c), nombre))$ ;
  asegura actoresResultadoIgualEntrada :  $actores(result) == actores(peliculaDeNombre(pre(c), nombre))$ ;
  asegura salaResultadoIgualEntrada :  $sala(c, result) == sala(c, (peliculaDeNombre(pre(c), nombre)))$ ;
  asegura nombreNoCambia :  $nombre(c) == nombre(pre(c))$ ;
  asegura salasNoCambian :  $salas(c) == salas(pre(c))$ ;
  asegura lasPelisNoCambian :  $mismos(nombrePelisDeCine(c), nombrePelisDeCine(pre(c)))$ ;
  asegura lasSalasPorPeliculaNoCambian :  $(\forall p \leftarrow peliculas(c), nombre(p) \neq nombre) sala(c, p) == sala(pre(c), p)$ ;
  asegura losEspectadoresNoCambian :  $(\forall s \leftarrow salas(c)) espectadores(c, s) == espectadores(pre(c), s)$ ;
  asegura losTicketsNoCambian :  $mismos(nombrePelisDeTicketsDeCine(c), nombrePelisDeTicketsDeCine(pre(c)))$ ;
}

```

## 1. Auxiliares

```
aux obtenerGeneros (ps: [Pelicula]) : [Genero] = [genero(p) | p ← ps];
aux obtenerGenerosDupla (x: [(Genero,[Pelicula])]) : [Genero] = [prm(dupla) | dupla ← x];
aux generosOrd (gs: [Genero]) : Bool = (∀i ← [0..|gs|], i ≠ |gs|) ord(gsi) ≤ ord(gsi+1);
aux actoresOrd (as: [Actores]) : Bool = (∀i ← [0..|as|], i ≠ |gs|) ord(asi) ≤ ord(asi+1);
aux pelisConExito (c: Cine, e: ℤ) : [Pelicula] = [peli | sala ← salasConExito(c, e), peli ← peliculas(c), sala(c, peli) == sala];
aux nombresDePeliculas (c: Cine) : [String] = [nombre(p) | p ← peliculas(c)];
aux tienePeli (c: Cine, s: Sala) : Bool = (∃p ← pelicula(c)) sala(c, p) = s;
aux seParecen (c: Cine, c': Cine, e: ℤ) : Bool =
  nombresIguales(c, c') &&
  mismasSalas(c, c', e) &&
  mismasPeliculas(c, c') &&
  pelisEnMismasSalas(c, c') &&
  mismosEspectadores(c, c') &&
  mismosTickets(c, c');
aux nombresIguales (c: Cine, c': Cine) : Bool = nombre(c) == nombre(c');
aux mismasSalas (c: Cine, c': Cine, e: ℤ) : Bool = mismos(salas(c), salasSinTicketsVendidosConMasDe(c', e));
aux mismasPeliculas (c: Cine, c': Cine, e: ℤ) : Bool = mismas(peliculas(c), pelisDeSalas(c', salasConMasDe(c', e)));
aux pelisEnMismasSalas (c: Cine, c': Cine) : Bool = (∀p ← peliculas(c)) sala(c, p) == sala(c', p);
aux mismosEspectadores (c: Cine, c': Cine) : Bool = (∀sala ← salas(c)) espectadores(c, sala) == espectadores(c', sala);
aux mismosTickets (c: Cine, c': Cine) : Bool = mismos(ticketsVendidosSinUsar(c), ticketsVendidosSinUsar(c'));
aux nombrePelisDeCine (c: Cine) : [String] = [nombre(p) | p ← peliculas(c)];
aux nombrePelisDeTicketsDeCine (c: Cine) : [String] = [nombre(pelicula(t)) | t ← ticketsVendidosSinUsar(c)];
aux películaDeNombre (c: Cine, n: String) : Pelicula = cab[peleli | peleli ← peliculas(c), nombre(peleli) == n];
```