

Algoritmos y Estructura de Datos I

Segundo cuatrimestre de 2013

9 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(géneros(p));
  invariante génerosOrdenados : génerosOrd(géneros(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 : ( $\forall \text{dupla} \leftarrow \text{result}$ ) sinRepetidos(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);
}

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(nombresDePeliculas(res), elementosSinRepetir(nombres));
}

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 :  $\neg(\exists p \leftarrow \text{peliculasVistas}(\text{ticketsUsados}(\text{ts})) \text{ticketsPorPelicula}(p, \text{ticketsUsados}(\text{ts})) < \text{ticketsPorPelicula}(\text{result}, \text{ticketsUsados}(\text{ts})))$ ;
}

problema todosLosTicketsParaLaMismaSala (ts:[Ticket]) = result : Bool {
  asegura todosLosTicketsParaLaMismaSala : res == ( $\forall t \leftarrow [0..|ts| - 1]$ ) 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(tsi) == nueva;
  asegura mismasPelis : ( $\forall i \leftarrow [0..|pre(ts)|]$ ) pelicula(tsi) == pelicula(pre(ts)i);
  asegura ticketsSiguenUsados : ( $\forall i \leftarrow [0..|pre(ts)|]$ ) usado(tsi) == usado(pre(ts)i);
  asegura mismasSalas : ( $\forall i \leftarrow [0..|pre(ts)|]$ ) sala(pre(ts)i)  $\neq$  vieja sala(tsi) == sala(pre(ts)i);
}

```

```

}

tipo Cine {
  observador nombre (c: Cine) : String;
  observador películas (c: Cine) : [Películas];
  observador salas (c: Cine) : [Sala];
  observador sala (c: Cine, p: Película) : Sala;
    requiere  $p \in películas(c)$ ;
  observador espectadores (c: Cine, s: Sala) :  $\mathbb{Z}$ ;
    requiere  $s \in salas(c)$ ;
  observador ticketsVendidosSinUsar (c: Cine) : [Ticket];
  invariante sinPelículasRepetidas :  $sinRepetidos(nombresDePelículas(c))$ ;
  invariante sinSalasRepetidas :  $sinRepetidos(salas(c))$ ;
  invariante salasDeCineSonSalas :  $(\forall p \leftarrow películas(c)) sala(c, p) \in salas(c)$ ;
  invariante espectadoresNoNegativos :  $(\forall s \leftarrow salas(c)) espectadores(c, s) \geq 0$ ;
  invariante losTicketsVendidosEstanSinUsar :  $(\forall t \leftarrow ticketsVendidosSinUsar(c)) \neg usado(t)$ ;
  invariante salasConsistentes :  $sinRepetidos([sala(c, peli) | peli \leftarrow películas(c)])$ ;
  invariante losTicketsVendidosSonParaPelículasDelCine :  $(\forall t \leftarrow ticketsVendidosSinUsar(c))$ 
     $película(t) \in películas(c) \ \&\& \ sala(t) == sala(c, película(t))$ ;
}

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

problema agregarPelícula (c: Cine, p: Película, s: Sala) {
  requiere salaEntradaEnCine :  $s \in salas(c)$ ;
  requiere nuevaSalaVacía :  $\neg((\exists pe \leftarrow película(c)) sala(c, pe) == s)$ ;
  requiere nuevaPelícula :  $p \notin películas(c)$ ;
  modifica c;
  asegura películasSeMantienen :  $mismos(películas(pre(c)) ++ [p], películas(c))$ ;
  asegura salasSeMantienen :  $mismos(salas(c), salas(pre(c)))$ ;
  asegura nuevaPeliEnCorrectaSala :  $sala(c, p) == s$ ;
  asegura nombresSeMantienen :  $nombre(c) == nombre(pre(c))$ ;
  asegura salaParaTodasPelículasSeMantiene :  $(\forall p \leftarrow películas(pre(c))) sala(c, p) == sala(pre(c), p)$ ;
  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 películasSinLasDeSalasCerradas :  $mismos(películas(c), [peli | peli \leftarrow películas(pre(c)), peli \neq peliDeSala(s, c)])$ ;
  asegura salaParaTodasPelículasSeMantiene :  $(\forall p \leftarrow películas(pre(c))) sala(c, p) == sala(pre(c), p)$ ;
  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 :  $(\forall sa \leftarrow salas(pre(c))espectadores(pre(c), sa) == espectadores(c, sa)$ ;
}

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 cuenta (x: T, a: [T]) :  $\mathbb{Z} = long([y \mid y \leftarrow a, y == x])$ ;
aux mismos (a, b: [T]) : Bool =  $|a| == |b| \ \&\& \ (\forall c \in a) cuenta(c, a) == cuenta(c, b)$ ;
aux sinRepetidos (l: [T]) : Bool =  $(\forall i, j \leftarrow [0..|l|], i \neq j)l[i] \neq l[j]$ ;
aux nombresDePeliculas (c: Cine) : [String] =  $[nombre(p) \mid p \leftarrow peliculas(c)]$ ;
aux tienePeli (c: Cine, s: Sala) : Bool =  $(\exists p \leftarrow peliculas(c))sala(c, p) == s$ ;
aux obtenerGeneros (ps: [Pelicula]) : [Genero] =  $[genero(p) \mid p \leftarrow ps]$ ;
aux obtenerGenerosDupla (x: [(Genero, [Pelicula])]) : [Genero] =  $[prm(dupla) \mid dupla \leftarrow x]$ ;
aux generosOrd (gs: [Genero]) : Bool =  $(\forall i \leftarrow [0..|gs|], i \neq |gs|)ord(gs_i) \leq ord(gs_{i+1})$ ;

```

```

aux actoresOrd (as: [Actores]) : Bool = ( $\forall i \leftarrow [0..|as|, i \neq |gs|) \text{ord}(as_i) \leq \text{ord}(as_{i+1})$ );
aux ticketsUsados (s: [Ticket]) : [Ticket] = [ $x \mid x \leftarrow s, \text{usado}(x)$ ];
aux peliculasVistas (s: [Ticket]) : [Pelicula] = [ $\text{pelicula}(x) \mid x \leftarrow s$ ];
aux ticketsPorPelicula (p: Pelicula, tic: [Ticket]) :  $\mathbb{Z}$  = [ $t \mid t \leftarrow \text{tic}, \text{pelicula}(t) == p$ ];
aux pelisConExito (c: Cine, e:  $\mathbb{Z}$ ) : [Pelicula] = [ $\text{peli} \mid \text{sala} \leftarrow \text{salasConExito}(c, e),$ 
   $\text{peli} \leftarrow \text{peliculas}(c), \text{sala}(c, \text{peli}) == \text{sala}$ ];
aux elementosSinRepetir (s: [T]) : [T] = [ $s_i \mid i \leftarrow [0..|s|], \neg(\exists j \leftarrow [0..|s|], j \neq i) s_i == s_j$ ];
aux peliDeSala (s:sala, c: Cine) : Pelicula =  $\text{cab}([\text{peli} \mid \text{peli} \leftarrow \text{peliculas}(c), \text{sala}(c, \text{peli}) == s])$ ;
aux salaSinTicketsVendidosConMasDe (c: Cine, e:  $\mathbb{Z}$ ) : [Sala] = [ $\text{sala} \mid \text{sala} \leftarrow \text{salas}(c), (\forall t \leftarrow \text{ticketsVendidosSinUsar}(c))$ 
   $((\text{sala} \neq \text{sala}(t)) \&\& (\text{espectadores}(\text{sala}) \geq e))$ ];
aux salasConMasDe (c: Cine, e:  $\mathbb{Z}$ ) : [Sala] = [ $\text{sala} \mid \text{sala} \leftarrow \text{salas}(c), \text{espectadores}(c, \text{sala}) \geq e$ ];
aux pelisDeSalas (c: Cine, sal: [Sala]) : [Peliculas] = [ $\text{peliDeSala}(c, s) \mid s \leftarrow \text{sal}$ ];
aux seParecen (c: Cine, c': Cine, e:  $\mathbb{Z}$ ) : 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 =  $\text{nombre}(c) == \text{nombre}(c')$ ;
aux mismasSalas (c: Cine, c': Cine, e:  $\mathbb{Z}$ ) : Bool =  $\text{mismos}(\text{salas}(c), \text{salasSinTicketsVendidosConMasDe}(c', e))$ ;
aux mismasPeliculas (c: Cine, c': Cine, e:  $\mathbb{Z}$ ) : Bool =  $\text{mismas}(\text{peliculas}(c), \text{pelisDeSalas}(c', \text{salasConMasDe}(c', e)))$ ;
aux pelisEnMismasSalas (c: Cine, c': Cine) : Bool =  $(\forall p \leftarrow \text{peliculas}(c)) \text{sala}(c, p) == \text{sala}(c', p)$ ;
aux mismosEspectadores (c: Cine, c': Cine) : Bool =  $(\forall \text{sala} \leftarrow \text{salas}(c)) \text{espectadores}(c, \text{sala}) == \text{espectadores}(c', \text{sala})$ ;
aux mismosTickets (c: Cine, c': Cine) : Bool =  $\text{mismos}(\text{ticketsVendidosSinUsar}(c), \text{ticketsVendidosSinUsar}(c'))$ ;
aux nombrePelisDeCine (c: Cine) : [String] = [ $\text{nombre}(p) \mid p \leftarrow \text{peliculas}(c)$ ];
aux nombrePelisDeTicketsDeCine (c: Cine) : [String] = [ $\text{nombre}(\text{pelicula}(t)) \mid t \leftarrow \text{ticketsVendidosSinUsar}(c)$ ];
aux películaDeNombre (c: Cine, n: String) : Pelicula =  $\text{cab}([\text{peli} \mid \text{peli} \leftarrow \text{peliculas}(c), \text{nombre}(\text{peli}) == n])$ ;

```

2. Nota sobre los ejercicios 12, 13 y 14

El argumento utilizado al momento de la resolución de estos problemas es que no es posible cerrar salas que esten relacionadas con aquellos tickets vendidos sin usar, ya que el invariante lo prohíbe. Se podría haber optado también por eliminar los tickets de esas salas, pero fue considerado inapropiado, por lo tanto fue elegida la primera opción: no cerrar aquellas salas que tengan un ticket vendido sin usar asignado a ellas.