

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(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 : ( $\forall \text{dupla} \leftarrow \text{result}$ ) 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 peliEnTickets : result  $\in$  peliculasVistas(ticketsUsados(ts));
  asegura esLaPeliMenosVista :  $\neg((\exists p \leftarrow \text{peliculasVistas}(\text{ticketsUsados}(\text{ts})))$ 
    ticketsPorPelicula(p, ticketsUsados(ts)) < ticketsPorPelicula(result, ticketsUsados(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) == viejasala(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)  $\neq$  viejasala(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 :  $sinRepetidos([sala(c, peli) \mid peli \leftarrow películas(c)])$ ;
  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 mismoNombre :  $nombre(res) == n$ ;
  asegura salasVacias :  $salas(res) == []$ ;
}

problema agregarPelícula (c: Cine, p: Película, s: Sala) {
  requiere esSala :  $s \in salas(c)$ ;
  requiere nuevaSalaVacía :  $\neg((\exists pe \leftarrow película(c)) sala(c, pe) == s)$ ;
  requiere nuevaPelícula :  $p \notin películas(c)$ ;
  modifica c;
  asegura mismosNombres :  $nombre(c) == nombre(pre(c))$ ;
  asegura mismasPelis :  $mismos(películas(pre(c)) ++ [p], películas(c))$ ;
  asegura mismasSalas :  $mismos(salas(c), salas(pre(c)))$ ;
  asegura peliEnSala :  $sala(c, p) == s$ ;
  asegura pelisEnMismasSalas :  $(\forall p \leftarrow películas(pre(c))) sala(c, p) == sala(pre(c), p)$ ;
  asegura mismosEspectadores :  $(\forall s \leftarrow salas(c)) espectadores(c, p) == espectadores(pre(c), p)$ ;
  asegura mismosTickets :  $mismos(ticketsVendidosSinUsar(c) == ticketsVendidosSinUsar(pre(c)))$ ;
}

problema cerrarSala (c: Cine, s: Sala) {
  requiere esSala :  $s \in salas(c)$ ;
  requiere salaSinTickets :  $(\forall t \leftarrow ticketsVendidosSinUsar(c)) sala(t) \neq s$ ;
  modifica c;
  asegura mismoNombre :  $nombre(c) == nombre(pre(c))$ ;
  asegura mismasSalas :  $mismos(salas(c), [sal \mid sal \leftarrow salas(pre(c)), sal \neq s])$ ;
  asegura mismasPelis :  $mismos(películas(c), [peli \mid peli \leftarrow películas(pre(c)), peli \neq peliDeSala(s, c)])$ ;
  asegura pelisEnMismasSalas :  $(\forall p \leftarrow películas(pre(c))) sala(c, p) == sala(pre(c), p)$ ;
  asegura mismosEspectadores :  $(\forall s \leftarrow salas(c)) espectadores(c, p) == espectadores(pre(c), p)$ ;
  asegura mismosTickets :  $mismos(ticketsVendidosSinUsar(c), ticketsVendidosSinUsar(pre(c)))$ ;
}

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 mismosCines :  $|cs| == |pre(cs)|$ ;
  asegura cinesSeParecen :  $(\forall c' \leftarrow pre(cs)) (\exists c \leftarrow cs) seParecen(c, c', e)$ ;
}

```

```

problema pelicula (c: Cine, s: Sala) = result : Pelicula {
  requiere esSala :  $s \in salas(c)$ ;
  requiere tienePeli :  $(\exists p \leftarrow peliculas(c))sala(c, p) == s$ ;
  asegura esPeli :  $result \in peliculas(c)$ ;
  asegura peliEnSala :  $sala(c, result) == s$ ;
}

problema venderTicket (c: Cine, p: Pelicula) = result : Ticket {
  requiere esPelicula :  $p \in peliculas(c)$ ;
  modifica c;
  asegura mismoNombre :  $nombre(c) == nombre(pre(c))$ ;
  asegura mismasSalas :  $mismos(salas(pre(c)), salas(c))$ ;
  asegura mismasPelis :  $mismos(peliculas(pre(c)), peliculas(c))$ ;
  asegura pelisEnMismasSalas :  $(\forall pe \leftarrow peliculas(pre(c))sala(pre(c), pe) == sala(c, pe)$ ;
  asegura mismosEspectadores :  $(\forall sa \leftarrow salas(pre(c))espectadores(pre(c), pe) == espectadores(c, pe)$ ;
  asegura mismosTickets :  $mismos(ticketsVendidosSinUsar(pre(c) ++ [result]), ticketsVendidosSinUsar(pre(c))$ ;
}

problema ingresarASala (c: Cine, s: Sala, t: Ticket) {
  requiere esSala :  $s \in salas(c)$ ;
  requiere salaTicketEsSala :  $sala(t) == s$ ;
  requiere ticketSinUsar :  $t \in ticketsVendidosSinUsar(c)$ ;
  modifica c;
  modifica t;
  asegura mismaPeli :  $pelicula(t) == pelicula(pre(t))$ ;
  asegura mismaSala :  $sala(t) == sala(pre(t))$ ;
  asegura ticketUsado :  $usado(t)$ ;
  asegura ingresaEspectador :  $espectadores(c, s) == espectadores(pre(c), s) + 1$ ;
  asegura mismasSalas :  $mismos(salas(c), salas(pre(c)))$ ;
  asegura mismoNombre :  $nombre(c) == nombre(pre(c))$ ;
  asegura mismasPelis :  $mismos(peliculas(c), peliculas(pre(c)))$ ;
  asegura pelisEnMismasSalas :  $(\forall p \leftarrow peliculas(pre(c))sala(c, p) == sala(pre(c), p)$ ;
  asegura mismosEspectadores :  $(\forall sa \leftarrow salas(pre(c)), sa \neq s)espectadores(c, sa) == espectadores(pre(c), sa)$ ;
  asegura mismosTickets :  $mismos(ticketsVendidosSinUsar(c),$ 
     $[ti \mid ti \leftarrow ticketsVendidosSinUsar(pre(c)),$ 
     $ti \neq t])$ ;
}

problema pasarA3DUnaPelicula (c: Cine, nombre: String) = result : Pelicula {
  requiere laPeliculaExiste :  $(\exists p \leftarrow peliculas(c))nombre(p) == nombre$ ;
  modifica c;
  asegura mismoNombrePeli :  $nombre(result) == nombre$ ;
  asegura peli3D :  $es3D(result)$ ;
  asegura mismosGeneros :  $generos(result) == generos(peliculaDeNombre(pre(c), nombre))$ ;
  asegura mismosActores :  $actores(result) == actores(peliculaDeNombre(pre(c), nombre))$ ;
  asegura mismaSala :  $sala(c, result) == sala(c, peliculaDeNombre(pre(c), nombre))$ ;
  asegura mismoNombre :  $nombre(c) == nombre(pre(c))$ ;
  asegura mismasSalas :  $salas(c) == salas(pre(c))$ ;
  asegura mismasPelis :  $mismos(nombrePelisDeCine(c), nombrePelisDeCine(pre(c)))$ ;
  asegura pelisEnMismasSalas :  $(\forall p \leftarrow peliculas(c), nombre(p) \neq nombre)sala(c, p) == sala(pre(c), p)$ ;
  asegura mismosEspectadores :  $(\forall s \leftarrow salas(c))espectadores(c, s) == espectadores(pre(c), s)$ ;
  asegura mismosTickets :  $mismos(nombrePelisDeTicketsDeCine(c), nombrePelisDeTicketsDeCine(pre(c)))$ ;
}

```

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 = ( $\forall i \leftarrow [0..|gs|], i \neq |gs| \text{ord}(gs_i) \leq \text{ord}(gs_{i+1})$ );
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] = [ $x \mid x \leftarrow s, usado(x)$ ];
aux peliculasVistas (s: [Ticket]) : [Pelicula] = [ $pelicula(x) \mid x \leftarrow s$ ];
aux ticketsPorPelicula (p: Pelicula, tic: [Ticket]) :  $\mathbb{Z}$  = [ $t \mid t \leftarrow tic, pelicula(t) == p$ ];
aux pelisConExito (c: Cine, e:  $\mathbb{Z}$ ) : [Pelicula] = [ $pelis \mid sala \leftarrow salasConExito(c, e),$ 
   $pelis \leftarrow peliculas(c), sala(c, pelis) == sala$ ];
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 =  $cab([pelis \mid pelis \leftarrow peliculas(c), sala(c, pelis) == s])$ ;
aux salaSinTicketsVendidosConMasDe (c: Cine, e:  $\mathbb{Z}$ ) : [Sala] = [ $sala \mid sala \leftarrow salas(c), (\forall t \leftarrow ticketsVendidosSinUsar(c))$ 
   $((sala \neq sala(t)) \&\& (espectadores(sala) \geq e))$ ];
aux salasConMasDe (c: Cine, e:  $\mathbb{Z}$ ) : [Sala] = [ $sala \mid sala \leftarrow salas(c), espectadores(c, sala) \geq e$ ];
aux pelisDeSalas (c: Cine, sal: [Sala]) : [Peliculas] = [ $pelisDeSala(c, s) \mid s \leftarrow sal$ ];
aux mismoNombre (c, c': Cine) : Bool =  $nombre(c) == nombre(c')$ ;
aux mismasSalas (c, c': Cine, e:  $\mathbb{Z}$ ) : Bool =  $mismos(salas(c), salasSinTicketsVendidosConMasDe(c', e))$ ;
aux mismasPeliculas (c, c': Cine, e:  $\mathbb{Z}$ ) : Bool =  $mismos(peliculas(c), pelisDeSalas(c', salasConMasDe(c', e)))$ ;
aux pelisEnMismasSalas (c, c': Cine) : Bool = ( $\forall p \leftarrow peliculas(c)) sala(c, p) == sala(c', p)$ ;
aux mismosEspectadores (c, c': Cine) : Bool = ( $\forall sala \leftarrow salas(c)) espectadores(c, sala) == espectadores(c', sala)$ ;
aux mismosTickets (c, c': Cine) : Bool =  $mismos(ticketsVendidosSinUsar(c), ticketsVendidosSinUsar(c'))$ ;
aux nombrePelisDeCine (c: Cine) : [String] = [ $nombre(p) \mid p \leftarrow peliculas(c)$ ];
aux nombrePelisDeTicketsDeCine (c: Cine) : [String] = [ $nombre(pelicula(t)) \mid t \leftarrow ticketsVendidosSinUsar(c)$ ];
aux peliculaDeNombre (c: Cine, n: String) : Pelicula =  $cab([pelis \mid pelis \leftarrow peliculas(c), nombre(pelis) == n])$ ;
aux seParecen (c, c': Cine, e:  $\mathbb{Z}$ ) : Bool =
  mismoNombre(c, c')&&
  mismasSalas(c, c', e)&&
  mismasPeliculas(c, c')&&
  pelisEnMismasSalas(c, c')&&
  mismosEspectadores(c, c')&&
  mismosTickets(c, c');

```

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.