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

Algoritmos y Estructura de Datos I

Segundo cuatrimestre de 2012

01 de Octubre de 2012

Especificación del TPF - Hoteles Funcionales v1.3

1. Tipos

```
tipo Fecha = ℤ;
tipo DNI = ℤ;
tipo Dinero = ℤ;
tipo Cadena = String;
tipo Nombre = String;
tipo Provincia = String;
tipo CheckIn = (DNI, Fecha);
tipo CheckOut = (DNI, Fecha);
tipo TipoHabitacion = Simple, Doble, Triple, Cuadruple;
tipo Accesorio = Jacuzzi, LCD, PS3, DVD, Pelotero, Inflable;
```

2. Reserva

```
tipo Reserva {
  observador documento (r: Reserva) : DNI;
  observador fechaDesde (r: Reserva) : Fecha;
  observador fechaHasta (r: Reserva) : Fecha;
  observador tipo (r: Reserva) : TipoHabitacion;
  observador confirmada (r: Reserva) : Bool;

  invariante NoAntesDeDespues : fechaHasta(r) > fechaDesde(r);
}

problema nuevaR (d: DNI, fd, fh: Fecha, t: TipoHabitacion) = result : Reserva {
  requiere fh > fd;
  asegura confirmada(result) == False;
  asegura fechaDesde(result) == fd;
  asegura fechaHasta(result) == fh;
  asegura documento(result) == d;
  asegura tipo(result) == t;
}

problema documentoR (r: Reserva) = result : DNI {
  asegura result == documento(r);
}

problema fechaDesdeR (r: Reserva) = result : Fecha {
  asegura result == fechaDesde(r);
}

problema fechaHastaR (r: Reserva) = result : Fecha {
  asegura result == fechaHasta(r);
}

problema tipoR (r: Reserva) = result : tipoHabitacion {
  asegura result == tipo(r);
}

problema confirmadaR (r: Reserva) = result : Bool {
  asegura result == confirmada(r);
}
```

```
problema confirmarR (r: Reserva) = result : Reserva {
  asegura confirmada(result) == True;
  asegura fechaDesde(result) == fechaDesde(r);
  asegura fechaHasta(result) == fechaHasta(r);
  asegura documento(result) == documento(r);
  asegura tipo(result) == tipo(r);
}
```

3. Habitación

```
tipo Habitacion {
  observador numero (h: Habitacion) : ℤ;
  observador tipo (h: Habitacion) : TipoHabitacion;
  observador accesorios (h: Habitacion) : [Accesorio];

  invariante sinAccesoriosRepetidos : sinRepetidos(accesorios(h));
  invariante accesoriosOrdenada : ordenada(accesorios(h));
}

problema nuevaH (n: ℤ, t: TipoHabitacion, as: [Accesorio]) = result : Habitacion {
  requiere sinAccesoriosRepetidos : sinRepetidos(as);
  requiere accesoriosOrdenada : ordenada(as);
  asegura numero(result) == n;
  asegura tipo(result) == t;
  asegura accesorios(result) == as;
}

problema numeroH (h: habitacion) = result : ℤ {
  asegura numero(h) == result;
}

problema tipoH (h: habitacion) = result : tipoHabitacion {
  asegura tipo(h) == result;
}

problema accesoriosH (h: habitacion) = result : [Accesorio] {
  asegura accesorios(h) == result;
}

problema pretencionesDePopStarH (as: [Accesorio], hs: [Habitacion]) = result : [Habitacion] {
  asegura (∀r ← result)r ∈ hs;
  asegura (∀r ← result)tieneMasAccesoriosQueTodas(r, as, hs);
  asegura (∀h ← hs, tieneMasAccesoriosQueTodas(h, as, hs))h ∈ result;
  aux cantidadDeAccesoriosEnHabitacion (as: [Accesorio], h: Habitacion) : ℤ = |[ a | a ← as, a ∈ accesorios(h) ]|;
  aux tieneMasAccesoriosQueTodas (h: Habitacion, as: [Accesorio], hs: [Habitacion]) : Bool =
      (∀x ← hs)cantidadDeAccesoriosEnHabitacion(as, x) ≤ cantidadDeAccesoriosEnHabitacion(as, h);
}
```

4. Hotel

```
tipo Hotel {
  observador nombre (h: Hotel) : Nombre ;
  observador cadena (h: Hotel) : Cadena ;
  observador huespedes (h: Hotel) : [DNI] ;
  observador habitaciones (h: Hotel) : [Habitacion] ;
  observador ingresos (h: Hotel) : [(CheckIn, Habitacion)] ;
  observador salidas (h: Hotel) : [CheckOut] ;
  observador reservas (h: Hotel) : [Reserva] ;
  observador tarifaHabitacionXDia (h: Hotel) : [(TipoHabitacion, Dinero)] ;
  observador precioAccesorio (h: Hotel) : [(Accesorio, Dinero)] ;
  invariante habitacionesValidas : (∀c ← ingresos(h))snd(c) ∈ habitaciones(h) ;
  invariante cuantosHay : |huespedes(h)| == |ingresos(h)| − |salidas(h)| ;
  invariante sinHuespedesRepeditos : sinRepetidos(huespedes(h)) ;
  invariante siEstanNoSeFueron : ∀d ← huespedes(h)|ingresosDe(h, d)| == |salidasDe(h, d)| + 1 ;
  invariante siSeVaEntro : (∀o ← salidas(h))existeUnIngresoSinSalida(h, o) ;
  invariante estanAlMenosUnDia : (∀o ← salidas(h))¬existeUnIngresoElMismoDia(h, dniCheckOut(o), fechaCheckOut(o)) ;
  invariante noEntranDosVeces : (∀i ← ingresos(h))¬existeUnIngresoElMismoDia(h, dniCheckIn(i), fechaCheckIn(i)) ;
  invariante reservasValidas : (∀r ← reservas(h))existeUnaHabitacionDelTipo(h, tipo(r)) ;
  invariante sinTarifasRepetidas : sinRepetidos([prm(t)|t ← tarifaHabitacionXDia(h)]) ;
  invariante sinPreciosRepetidos : sinRepetidos([prm(p)|p ← precioAccesorio(h)]) ;
  invariante tarifasPositivas : (∀t ← tarifaHabitacionXDia(h))snd(t) > 0 ;
  invariante preciosPositivos : (∀p ← precioAccesorio(h))snd(p) > 0 ;
  invariante noValeAcaparar : (∀r1, r2 ← reservas(h), tipo(r1) == tipo(r2) ∧ documento(r1) == documento(r2) ∧
      fechaDesde(r1) == fechaDesde(r2))r1 == r2 ;
  invariante noVuelveElMismoDia : ¬(((∃ci ← ingresos(h))(∃co ← salidas(h))
      dniCheckIn(prm(ci)) == dniCheckOut(co) ∧ fechaCheckIn(prm(ci)) == fechaCheckOut(co))) ;
  invariante preciosSeCorresponden : (∀h′ ← habitaciones(h), a ← accesorios(h′))
      (∃p ← prcioAccesorio(h))a == prm(p) ;
  invariante tiposDeHabSeCorresponden : (∀h′ ← habitaciones(h))(∃p ← tarifaHabitacionXDia(h))tipo(h′) == prm(p) ;
  invariante sinRepetidos(reservas(h)) ;
}

problema nuevoH (n: Nombre, c: Cadena, hs: [Habitacion], thxd: [(TipoHabitacion, Dinero)], pa: [(Accesorio, Dinero)] ) =
result : Hotel {
  requiere sinTarifasRepetidas : sinRepetidos([ prm(t) | t ← thxd ]) ;
  requiere sinPreciosRepetidos : sinRepetidos([ prm(p) | p ← pa ]) ;
  requiere tarifasPositivas : (∀t ← thxd)snd(t) > 0 ;
  requiere preciosPositivos : (∀p ← pa)snd(t) > 0 ;
  requiere sinHabsRepetidas : sinRepetidos(hs) ;
  requiere preciosSeCorresponden : (∀h ← hs, a ← accesorios(h))(∃p ← pa)a == prm(p) ;
  requiere tiposDeHabSeCorresponden : (∀h ← hs)(∃p ← thxd)tipo(h) == prm(p) ;
  asegura nombre(result) == n ;
  asegura cadena(result) == c ;
  asegura mismos(habitaciones(result), hs) ;
  asegura |reservas(result)| == 0 ;
  asegura |huespedes(result)| == 0 ;
  asegura |ingresos(result)| == 0 ;
  asegura |salidas(result)| == 0 ;
  asegura mismos(tarifaHabitacionXDia(result), thxd) ;
  asegura mismos(precioAccesorio(result), pa) ;
}

problema nombreH (h: Hotel) = result : Nombre {
  asegura nombre(h) == result ;
}

problema cadenaH (h: Hotel) = result : Cadena {
  asegura cadena(h) == result ;
}

problema huespedesH (h: Hotel) = result : [DNI] {
  asegura mismos(huespedes(h), result) ;
```

}

problema habitacionesH (h: Hotel) = result : [Habitacion] {
  asegura $mismos(habitaciones(h), result)$;
}

problema ingresosH (h: Hotel) = result : [(checkIn, Habitacion)] {
  asegura $mismos(ingresos(h), result)$;
}

problema salidasH (h: Hotel) = result : [CheckOut] {
  asegura $mismos(salidas(h), result)$;
}

problema reservasH (h: Hotel) = result : [Reserva] {
  asegura $mismos(reservas(h), result)$;
}

problema tarifaXDiaH (h: Hotel, t: TipoHabitacion) = result : Dinero {
  requiere $(\exists p \leftarrow tarifaHabitacionXDia(h))prm(p) == t$;
  asegura $result == cab([\, snd(p) \mid p \leftarrow tarifaHabitacionXDia(h), prm(p) == t \,])$;
}

problema precioAccesorioH (h: Hotel, a: Accesorio) = result : Dinero {
  requiere $(\exists p \leftarrow precioAccesorio(h))prm(p) == a$;
  asegura $result == cab([\, snd(p) \mid p \leftarrow precioAccesorio(h), prm(p) == a \,])$;
}

problema venderH (h: Hotel, c: Cadena) = result : Hotel {
  asegura $cambioDeCadena(h, result, c)$;
}

problema hacerReservaH (h: Hotel, r: Reserva) = result : Hotel {
  requiere $existeUnaHabitacionDelTipo(h, tipo(r))$;
  requiere noAcapara : $(\forall r' \leftarrow reservas(h), tipo(r) == tipo(r'),$
    $documento(r) == documento(r'), fechaDesde(r) == fechaDesde(r'))r == r'$;
  requiere $\neg(\exists r' \leftarrow reservas(h))r == r'$;

  asegura $nombre(h) == nombre(result)$;
  asegura $cadena(h) == cadena(result)$;
  asegura $mismos(huespedes(h), huespedes(result))$;
  asegura $mismos(habitaciones(h), habitaciones(result))$;
  asegura $mismos(ingresos(h), ingresos(result))$;
  asegura $mismos(salidas(h), salidas(result))$;
  asegura $mismos(r : reservas(h), reservas(result))$;
  asegura $mismos(tarifaHabitacionXDia(h), tarifaHabitacionXDia(result))$;
  asegura $mismos(precioAccesorio(h), precioAccesorio(result))$;
}

problema sobreReservadoH (h: Hotel, f: Fecha) = result : Bool {
  asegura $result == capacidad(h) < plazasReservadas(reservasXFecha(h, f))$;
  aux reservasXFecha (h: Hotel, f: Fecha) : [Reserva] $= [\, r \mid r \leftarrow reservas(h), fechaDesde(r) \leq f \leq fechaHasta(r) \,]$;
  aux plazasReservadas (rs: [Reserva]) : $\mathbb{Z} = sum([\, cantidadHuespedes(tipo(r)) \mid r \leftarrow rs \,])$;
}

problema registrarHuespedH (h: Hotel, d: DNI, f:Fecha, a: Habitacion) = result : Hotel {
  requiere $a \in habitaciones(h)$;
  requiere $d \notin huespedes(h)$;
  requiere habitacionLibre : $(\forall i \in ingresos(h), snd(i) == a)\ tieneCheckout(h, prm(i))$;
  requiere $existeReserva(h, d, f, tipo(a), false)$;
  requiere superaUltimoCheckout :
    $|salidasDe(h, d)| > 0 \longrightarrow max([\, fechaCheckOut(co) \mid co \leftarrow salidasDe(h, d) \,]) < f$;

  asegura $nombre(h) == nombre(result)$;
  asegura $cadena(h) == cadena(result)$;
  asegura $mismos(habitaciones(h), habitaciones(result))$;
  asegura $mismos(d : huespedes(h), huespedes(result))$;

asegura $mismos(((d, f), a) : ingresos(h), ingresos(result))$;
asegura $mismos(salidas(h), salidas(result))$;
asegura $|reservas(h)| == |reservas(result)|$;
asegura $|reservasConfirmadas(h)| == |reservasConfirmadas(result)| - 1$;
asegura $|reservasSinConfirmar(h)| == |reservasSinConfirmar(result)| + 1$;
asegura $mismos(tarifaHabitacion(h), tarifaHabitacion(result))$;
asegura $mismos(precioAccesorio(h), precioAccesorio(result))$;
asegura otrasReservasIguales : $(\forall r \in reservas(h), r \neq dameReserva(h, d, f, tipo(a), false))r \in reservas(result)$;
asegura $existeReserva(result, d, f, tipo(a), true)$;

aux existeReserva (h: Hotel, d: DNI, f: Fecha, t: TipoHabitacion, c: Bool) : Bool =
$(\exists r \leftarrow reservas(h))documento(r) == d \wedge fechaDesde(r) == f \wedge tipo(r) == t \wedge confirmada(r) == c$;
aux dameReserva (h: Hotel, d: DNI, f: Fecha, t: Tipo, c: Bool) : Reserva =
$cab([r | r \leftarrow reservas(h), documento(r) == d, fechaDesde(r) == f, tipo(r) == t, confirmada(r) == c])$;
aux tieneCheckOut (h: Hotel, ci : CheckIn) : Bool =
$(\exists co \in salidasDe(h, dniCheckIn(ci)), fechaCheckIn(ci) < fechaCheckOut(co))$
$\neg(\exists ci' \in ingresosDe(h, dniCheckIn(ci)), ci \neq ci')fechaCheckIn(ci) < fechaCheckIn(ci') < fechaCheckOut(co)$;
aux reservasConfirmadas (h: Hotel) : [Reserva] $= [\, r \mid r \leftarrow rs, confirmada(r) \,])$;
aux reservasSinConfirmar (h: Hotel) : [Reserva] $= [\, r \mid r \leftarrow rs, \neg confirmada(r) \,])$;
}

problema desRegistrarHuespedH (h: Hotel, d: DNI, f:Fecha) = result : Hotel {
requiere $d \in huespedes(h)$;
requiere $fechaMayorAlUltimoIngreso(f, d, h)$;

asegura $nombre(h) == nombre(result)$;
asegura $cadena(h) == cadena(result)$;
asegura $mismos(habitaciones(h), habitaciones(result))$;
asegura $mismos(huespedes(result), sacar(d, huespedes(h)))$;
asegura $mismos(ingresos(h), ingresos(result))$;
asegura $mismos((d, f) : salidas(h), salidas(result))$;
asegura $mismos(reservas(h), reservas(result))$;
asegura $mismos(tarifaHabitacion(h), tarifaHabitacion(result))$;
asegura $mismos(precioAccesorio(h), precioAccesorio(result))$;

aux sacar (x: T, xs: [T]) : [T] $= [\, k \mid k \leftarrow xs, k \neq x \,]$;
aux fechaMayorAlUltimoIngreso (f: Fecha, d: DNI, h: Hotel) : Bool $= f > max([\, fechaCheckIn(c) \mid c \leftarrow ingresosDe(h, d) \,])$;
}

problema huespedesConPalabraH (h: Hotel) = result : [DNI] {
asegura $mismos(result, sacarRepetidos(buscarHuespedesDePalabra(h)))$;

aux buscarHuespedesDePalabra (h: Hotel) : [DNI] $= [\, dniCheckOut(o) \mid i \leftarrow ingresos(h),$
$o \leftarrow salidasDe(h, dniCheckIn(prm(i))), noHaySalidaEnElMedio(prm(i), o, h),$
$existeReserva(h, dniCheckIn(prm(i)), fechaCheckIn(prm(i)), fechaCheckOut(o), tipo(sgd(i)), true) \,]$;
aux existeReserva (h: Hotel, d: DNI, fd: Fecha, fh: Fecha, t: TipoHabitacion, c: Bool) : Bool =
$(\exists r \leftarrow reservas(h))documento(r) == d \wedge fechaDesde(r) == fd \wedge tipo(r) == t \wedge fechaHasta(r) == fh \wedge confirmada(r) == c$;
}

problema calcularCuentaH (h: Hotel, i: CheckIn, o: CheckOut, hb: Habitacion) = result : Dinero {
requiere $dniCheckIn(i) == dniCheckOut(o)$;
requiere $fechaCheckIn(i) < fechaCheckOut(o)$;
requiere $(i, hb) \in ingresos(h)$;
requiere $o \in salidas(h)$;
requiere $noHaySalidaEnElMedio(i, o, h)$;

asegura $result == precioXAccesorios(accesorios(hb), h) + (tarifaXHabitacion(h, tipo(hb)) * dias(i, o))$;

aux precioXAccesorios (as: [Accesorio], h: Hotel) : Dinero =
$sum([\, snd(x) \mid a \leftarrow as, x \leftarrow precioAccesorio(h), prm(x) == a \,])$;
aux dias (i: CheckIn, o: CheckOut) : $\mathbb{Z} = fechaCheckOut(o) - fechaCheckIn(i)$;
aux tarifaXHabitacion (h: Hotel, t: TipoHabitacion) : Dinero $= [snd(d) | d \leftarrow tarifaHabitacionXDia(h), prm(d) == t]_0$;
}

problema reservasSolapadasH (h: Hotel, d: DNI) = result : Bool {

```
  asegura result == (∃r ← reservas(h), documento(r) == d)seSolapa(h, r);
  aux seSolapa (h: Hotel, r: Reserva) : Bool = (∃r' ← reservas(h), documento(r) == documento(r'), r ≠ r')
      entreFechas(fechaDesde(r), fechaHasta(r), fechaDesde(r')) ∨
      entreFechas(fechaDesde(r), fechaHasta(r), fechaHasta(r'));
  aux entreFechas (fd, fh, f: Fecha) : Bool = fd <= f <= fh;
}
```

5. MinisterioDeTurismo

```
tipo MinisterioDeTurismo {
  observador secretarias (m: MinisterioDeTurismo) : [Provincia] ;
  observador registro (m: MinisterioDeTurismo, p: Provincia ) : [Hotel] ;
       requiere p ∈ secretarias(m) ;
  observador cadenasDeHoteles (m: MinisterioDeTurismo) : [[Hotel]] ;
  invariante sinHotelesRepetidas : (∀xs ← cadenasDeHoteles(m))sinRepetidosH(xs) ;
  invariante cadenasConHoteles : (∀xs ← cadenasDeHoteles(m))|xs| > 0 ;
  invariante sinCadenasRepetidas : sinRepetidos([ cadena(xs_0) | xs ← cadenasDeHoteles(m) ]) ;
  invariante sinProvinciasRepetidas : sinRepetidos(secretarias(m)) ;
  invariante cadenasBienFormadas : (∀xs ← cadenasDeHoteles(m))sonDeLaMismaCadena(xs) ;
  invariante sinNombresRepetidosEnCadenas : (∀xs ← cadenasDeHoteles(m))sinRepetidos(todosLosNombres(xs)) ;
  invariante hotelesConsistentes : mismosHoteles(aplanar(cadenasDeHoteles(m)), todosLosHoteles(m)) ;
}

problema nuevoM (ps: [Provincia]) = result : MinisterioDeTurismo {
  requiere sinRepetidos(ps) ;
  asegura mismos(secretarias(result), ps) ;
  asegura |cadenasDeHoteles(result)| == 0 ;
  asegura (∀p ← ps)|registro(result, p)| == 0 ;
}

problema secretariasM (m: MinisterioDeTurismo) = result : [Provincia] {
  asegura mismos(secretarias(m), result) ;
}

problema registroM (m: MinisterioDeTurismo, p: Provincia) = result : [Hotel] {
  requiere p ∈ secretarias(m) ;
  asegura mismosHoteles(registro(m, p), result) ;
}

problema cadenasDeHotelesM (m: MinisterioDeTurismo) = result : [[Hotel]] {
  asegura |cadenasDeHoteles(m)| == |result| ;
  asegura (∀hs ← result)(∃hs' ← cadenasDeHoteles(m))mismosHoteles(hs, hs') ;
  asegura (∀hs ← cadenasDeHoteles(m))(∃hs' ← result)mismosHoteles(hs, hs') ;
}

problema agregarHotelM (m: MinisterioDeTurismo, h: Hotel, p: Provincia) = result : MinisterioDeTurismo {
  requiere p ∈ secretarias(m) ;
  requiere ¬(∃h' ← todosLosHoteles(m))hotelesIguales(h, h') ;
  requiere sinNombresRepetidosEnCadenas : sinRepetidos(todosLosNombres(h : hotelesXCadena(cadena(h), m))) ;

  asegura mismos(secretarias(m), secretarias(result)) ;
  asegura registrosIgualesSalvoP : (∀p' ← secretarias(m), p' ≠ p)
     hotelesIgales(registro(m, p), registro(result, p)) ;
  asegura PtieneAH : mismosHoteles(registro(result, p), h : registro(m, p)) ;

  aux hotelesXCadena (c: Cadena, m: MinisterioDeTurismo) : [Hotel] =
     [ h | hs ← cadenasDeHoteles(m), h ← hs, cadena(h) == c ]) ;
}

problema cadenasAmarretasM (m: MinisterioDeTurismo) = result : [Cadena] {
  asegura mismos(result, cadenasQueEnMenosProvinciasEstan(m)) ;
  aux cadenasQueEnMenosProvinciasEstan (m: MinisterioDeTurismo) : [Cadena] =
     [ c | c ← todasLasCadenas(m), estaEnCantidadMinima(c, m) ] ;
  aux estaEnCantidadMinima (c: Cadena, m: MinisterioDeTurismo) : Bool =
     (∀c2 ← todasLasCadenas(m))cantidadDeProvinciasConPresencia(c, m) ≤ cantidadDeProvinciasConPresencia(c2, m) ;
  aux cantidadDeProvinciasConPresencia (c: Cadena, m: MinisterioDeTurismo) : ℤ =
     |[ p | p ← secretarias(m), (∃h ← registro(m, p))cadena(h) == c ]| ;
}

problema fusionAutorizadaM (m: MinisterioDeTurismo, c1: Cadena, c2: Cadena) = result : MinisterioDeTurismo {
  requiere cadenasRegistradas : existeCadena(c1, m) ∧ existeCadena(c2, m) ;
  requiere fusionPosible :
     sinRepetidos(todosLosNombres(hotelesXCadena(c1, m)) + +todosLosNombres(hotelesXCadena(c2, m))) ;
```

asegura mismasProvincias : $mismos(secretarias(result), secretarias(m))$;
asegura noSePerdieronRegistros : $(\forall p \leftarrow secretarias(m))|registro(m,p)| == |registro(result,p)|$;
asegura noSePerdieronHoteles : $|todosLosHoteles(m)| == |todosLosHoteles(result)|$;
asegura losOtrosHotelesNoCambiaron :
$mismosHoteles([\, h \mid h \leftarrow todosLosHoteles(m), cadena(h) \neq c1 \wedge cadena(h) \neq c2\,], [\, h \mid h \leftarrow todosLosHoteles(result), cadena(h$
$c1 \wedge cadena(h) \neq c2\,]$;
asegura noPerdimosCadenas : $|todasLasCadenas(m)| == |todasLasCadenas(result)| + 1$;
asegura otrasCadenas :
$(\forall c \leftarrow todasLasCadenas(m), c \neq c1 \wedge c \neq c2) mismosHoteles(hotelesXCadena(c,m), hotelesXCadena(c, pre(m)))$;
asegura chauC2 : $|hotelesXCadena(c2, result)| == 0$;
asegura holaC1yC2 : $|hotelesXCadena(c1, result)| == |hotelesXCadena(c1,m) + +hotelesXCadena(c2,m)|$;
asegura cambioDeCadena :
$(\forall h \leftarrow todosLosHoteles(m), cadena(h) == c2)) \exists h2 \leftarrow todosLosHoteles(result) cambioDeCadena(h, h2, c1)$;
asegura noCambioDeCadena : $(\forall h \leftarrow todosLosHoteles(m, cadena(h) == c1))(\exists h2 \leftarrow todosLosHoteles(m)) hotelesIguales(h2, h)$
asegura hotelesQueExistenMantienenProvincia : $(\forall s \leftarrow secretarias(m))$
$(\forall h \leftarrow registro(m,s), cadena(h) \neq c2)$
$(\exists h' \leftarrow registro(result, s) hotelesIguales(h, h')$;
asegura hotelesQueCambiaronMantienenProvincia : $(\forall s \leftarrow secretarias(m))$
$(\forall h \leftarrow registro(m,s), cadena(h) == c2)$
$(\exists h' \leftarrow registro(result, s) cambioDeCadena(h, h', c1)$;

aux existeCadena (c: Cadena, m: MinisterioDeTurismo) : Bool =
$(\exists hs \leftarrow cadenasDeHoteles(m))(\exists h \leftarrow hs) cadena(h) == c$;
aux hotelesXCadena (c: Cadena, m: MinisterioDeTurismo) : [Hotel] =
$[\, h \mid hs \leftarrow cadenaDeHoteles(m), h \leftarrow hs, cadena(h) == c\,]$;
aux provinciasXCadena (c: Cadena, m: MinisterioDeTurismo) : [Provincia] =
$sacarRepetidos([\, p \mid p \leftarrow secretarias(m), (\exists h \leftarrow registro(m,p)) cadena(h) == c\,])$;
}

6. Auxiliares

aux dniCheckIn (c: CheckIn) : DNI $= prm(c)$;

aux dniCheckOut (c: CheckOut) : DNI $= prm(c)$;

aux fechaCheckIn (c: CheckIn) : Fecha $= snd(c)$;

aux fechaCheckOut (c: CheckOut) : Fecha $= snd(c)$;

aux ingresosDe (h: Hotel, d: DNI) : [CheckIn] $= [x | x \leftarrow ingresos(h), dniCheckIn(prm(x)) == d]$;

aux salidasDe (h: Hotel, d: DNI) : [CheckOut] $= [x | x \leftarrow salidas(h), dniCheckOut(prm(x)) == d]$;

aux noHaySalidaEnElMedio (i: CheckIn, o: CheckOut, h: Hotel) : Bool $= \neg((\exists co \leftarrow salidasDe(h, dniCheckOut(o)))$
$fechaCheckIn(i) < fechaCheckOut(co) < fechaCheckOut(o))$;

aux existeUnIngresoSinSalida (h: Hotel, o1: CheckOut) : Bool =
$(\exists i \leftarrow ingresosDe(h, dniCheckOut(o1)), fechaCheckIn(i) < fechaCheckOut(o1))$
$\neg(\exists o2 \leftarrow salidasDe(h, dniCheckOut(o1)), fechaCheckIn(i) < fechaCheckOut(o2) < fechaCheckOut(o1)$;

aux existeUnIngresoElMismoDia (h: Hotel, d: DNI, f: Fecha) : Bool =
$(\exists i \leftarrow ingresosDe(h,d)) fechaCheckIn(i) == f$;

aux existeUnaHabitacionDelTipo (h: Hotel, t: TipoHabitacion) : Bool $= (\exists x \leftarrow habitaciones(h)) tipo(x) == t$;

aux capacidad (h: Hotel) : $\mathbb{Z} = \sum [\, cantidadHuespedes(tipo(h)) \mid h \leftarrow habitaciones(h)\,]$;

aux cantidadHuespedes (t: TipoHabitacion) : $\mathbb{Z}$ =
if $t == Simple$ then 1 else (if $t == Doble$ then 2 else (if $t == Triple$ then 3 else 4)) ;

aux sonDeLaMismaCadena (hs: [Hotel]) : Bool $= (\forall h1, h2 \leftarrow hs) cadena(h1) == cadena(h2)$;

aux todosLosHoteles (m: MinisterioDeTurismo) : [Hotel] $= [\, h \mid p \leftarrow secretarias(m), h \leftarrow registro(m,p)\,]$;

aux todosLosNombres (hs: [Hotel]) : [String] $= [\, nombre(h) \mid h \leftarrow hs\,]$;

aux todasLasCadenas (m: MinisterioDeTurismo) : [Cadena] $= sacarRepetidos([\, cadena(h) \mid h \leftarrow todosLosHoteles(m)\,])$;

aux aplanar (xss: [[T]]) : [T] $= [\, x \mid xs \leftarrow xss, x \leftarrow xs\,]$;

aux ordenada (l:[T]) : Bool $= (\forall i \leftarrow [0..|l| - 1)) l_i \leq l_{i+1}$;

aux sinRepetidos (l: [T]) : Bool $= (\forall i, j \leftarrow [0..|l|), i \neq j) l_i \neq l_j$;

aux sinRepetidosH (l: [Hotel]) : Bool $= (\forall i, j \leftarrow [0..|l|), i \neq j) \neg hotelesIguales(l_i, l_j)$;

aux sacarRepetidos (l: [T]) : Bool $= (\forall i \leftarrow [0..|l|)) l_i \notin l_[i+1..longitud(l)-1)$;

aux hotelesIguales (h, h' : Hotel) : Bool $= \{$
$nombre(h) == nombre(h') \wedge$
$cadena(h) == cadena(h') \wedge$
$mismos(huespedes(h), huespedes(h')) \wedge$
$mismos(habitaciones(h), habitaciones(h')) \wedge$
$mismos(ingresos(h), ingresos(h')) \wedge$
$mismos(salidas(h), salidas(h')) \wedge$
$mismos(reservas(h), reservas(h')) \wedge$
$mismos(tarifaHabitacionXDia(h), tarifaHabitacionXDia(h')) \wedge$
$mismos(precioAccesorio(h), precioAccesorio(h'))$
$\}$;

aux cambioDeCadena (h1: Hotel, h2:Hotel, c: Cadena) : Bool $= \{nombre(h1) == nombre(h2) \wedge$
$cadena(h2) == c \wedge$
$mismos(huespedes(h1), huespedes(h2)) \wedge$
$mismos(habitaciones(h1), habitaciones(h2)) \wedge$
$mismos(ingresos(h1), ingresos(h2)) \wedge$
$mismos(salidas(h1), salidas(h2)) \wedge$
$mismos(reservas(h1), reservas(h2)) \wedge$
$mismos(tarifaHabitacionXDia(h1), tarifaHabitacionXDia(h2)) \wedge$
$mismos(precioAccesorio(h1), precioAccesorio(h2))$
$\}$;

aux mismosHoteles (hs, hs': [Hotel]) : Bool $= |hs| == |hs'| \wedge$
$(\forall h \leftarrow hs)(\exists h' \leftarrow hs') hotelesIguales(h, h') \wedge$
$(\forall h \leftarrow hs')(\exists h' \leftarrow hs) hotelesIguales(h, h')$;