DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico de Especificación (TPE)

Análisis Habitacional Argentino

26 de octubre de 2023 — Algoritmos y Estructuras de Datos I

Integrante	LU	Correo electrónico
Guibaudo, Camila	682/17	cami_sol_guibaudo@hotmail.com
Licari, Leandro	276/21	leandrolicari2000@gmail.com
Suarez, Antony	792/21	sebastsuar@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

Índice

1. Preliminares

Definición de tipos de datos:

- type $individuo = seq\langle dato \rangle$
- type $hogar = seq\langle dato \rangle$
- type $eph_i = seq\langle individuo \rangle$
- type $= eph_h = seq\langle hogar \rangle$
- type $joinHI = seq\langle hogar \cdot individuo \rangle$

Enumeración de atributos de individuo y hogar:

1. enum ItemIndividuo { INDCODUSU, COMPONENTE, INDAÑO, INDTRIMESTRE, CH4, CH6, NIVEL_ED, ESTADO, CAT_OCUP, P47T, PP04G }
2. enum ItemHogar { HOGCODUSU, HOGAÑO, HOGTRIMESTRE, HOGLATITUD, HOGLONGITUD, II7, REGION, MAS_500, IV1, IV2, II2, II3 }

Funciones generales.

Número	Auxiliar numérico
	Tabla individuo
0	`aux @indCodusu` : $\mathbb{Z} = ord(INDCODUSU)$;
1	`aux @componente` : $\mathbb{Z} = ord(COMPONENTE)$;
2	`aux @indAño` : $\mathbb{Z} = ord(INDAO)$;
3	`aux @indTrimestre` : $\mathbb{Z} = ord(INDTRIMESTRE)$;
4	`aux @genero` : $\mathbb{Z} = ord(CH4)$;
5	`aux @edad` : $\mathbb{Z} = ord(CH6)$;
6	`aux @nivelEducativo` : $\mathbb{Z} = ord(NIVEL\_ED)$;
7	`aux @estado` : $\mathbb{Z} = ord(ESTADO)$;
8	`aux @categoriaOcupacional` : $\mathbb{Z} = ord(CAT\_OCUP)$;
9	`aux @ingreso` : $\mathbb{Z} = ord(P47T)$;
10	`aux @lugarDeTrabajo` : $\mathbb{Z} = ord(PP04G)$;
	Tabla hogar
0	`aux @hogCodusu` : $\mathbb{Z} = ord(HOGCODUSU)$;
1	`aux @añoHogar` : $\mathbb{Z} = ord(HOGAO)$;
2	`aux @hogTrimestre` : $\mathbb{Z} = ord(HOGTRIMESTRE)$;
3	`aux @hogLatitud` : $\mathbb{Z} = ord(HOGLATITUD)$;
4	`aux @hogLongitud` : $\mathbb{Z} = ord(HOGLONGITUD)$;
5	`aux @regimenDeTenencia` : $\mathbb{Z} = ord(II7)$;
6	`aux @region` : $\mathbb{Z} = ord(REGION)$;
7	`aux @enCiudadGrande` : $\mathbb{Z} = ord(MAS\_500)$;
8	`aux @tipoHogar` : $\mathbb{Z} = ord(IV1)$;
9	`aux @cantidadAmbientes` : $\mathbb{Z} = ord(IV2)$;
10	`aux @cantidadHabitaciones` : $\mathbb{Z} = ord(II2)$;
11	`aux @tieneAmbienteExclusivoDeTrabajo` : $\mathbb{Z} = ord(II3)$;

Cuadro 1: Funciones auxiliares para describir atributos de individuo y hogar

- /* El predicado rango comprueba si un entero dado, está entre 0 y la longitud de la secuencia -1.*/
  `pred rango` (s: $seq\langle T \rangle$) {

$(\forall i : \mathbb{Z})(0 \leq i < |s|)$

}

- `aux apariciones` (s : $seq\langle T \rangle$, c: T) : $\mathbb{Z}$ $= \sum_{i=0}^{|s|-1}$(if $s[i] = c$ then 1 else 0 fi) **;**

2. Primera parte

2.1. Ejercicio 1

`proc esEncuestaVálida` (**in** th : eph_h, **in** ti : eph_i, **out** result : Bool) {
`Pre` {true}
`Post` $\{result = true \iff validarEncuesta(th, ti)\}$
}

- /* El predicado validarEncuesta decide si una encuesta es válida.*/
  `pred validarEncuesta` (th : eph_h,ti : eph_i) {
  $esMatriz(th) \land esMatriz(ti) validarExistencia(th, 12) \land validarExistencia(ti, 11) \land$
  $validarTodosLosRangos(th, ti) \land sinHogaresRepetidos(th) \land sinIndividuosRepetidos(ti) \land$
  $existenciaAsociacion(th, ti) \land existenciaAsociacion(ti, th) \land verificarCantIndividuos(th, ti) \land$
  $mismoAoTrimTHTI(th, ti) \land validarHabitaciones(th)$
  }
- /* El predicado validarTodosLosRangos si cada dato de las matrices th y ti están en un rango válido.*/
  `pred validarTodosLosRangos` (th : eph_h,ti : eph_i) {
  $validarRangos(th, @hogAo, 1869, 2021) \land validarRangos(th, @hogTrimestre, 1, 4) \land$
  $validarRangos(th, @hogLatitud, -90, 90) \land validarRangos(th, @hogLongitud, -180, 180) \land$
  $validarRangos(th, @regimenDeTenencia, 1, 3) \land validarRangos(th, @region, 1, 6) \land$
  $validarRangos(th, @enCiudadGrande, 0, 1) \land validarRangos(th, @tipoHogar, 1, 5) \land$
  $validarRangos(th, @tieneAmbienteExclusivoDeTrabajo, 1, 2) \land validarRangos(ti, @indTrimestre, 1, 4) \land$
  $validarRangos(ti, @genero, 1, 2) \land validarRangos(ti, @edad, 0, 130) \land validarRangos(ti, @nivelEducativo, 0, 1) \land$
  $validarRangos(ti, @estado, -1, 1) \land validarRangos(ti, @categoriaOcupacional, 0, 4) \land$
  $validarRangos(ti, @lugarDeTrabajo, 1, 10) \land validarNaturales(th, ti)$
  }
- /* El predicado validarNaturales decide si unos datos que tienen que ser naturales, lo son.*/
  `pred validarNaturales` (th : eph_h,ti : eph_i) {
  $(\forall i : \mathbb{Z})(rango(th, i) \longrightarrow_L (th[i][@hogCodusu] > 0) \land_L (th[i][@cantidadAmbientes] > 0)) \land ((\forall j : \mathbb{Z})(rango(ti, j) \longrightarrow_L$
  $(ti[j][@indCodusu] > 0) \land_L (ti[j][@componente] > 0) \land_L (ti[j][@ingreso] > -2)))$
  }
- /* El predicado esMatriz decide si una secuencia de secuencias es una matriz.*/
  `pred esMatriz` (m : $seq\langle seq\langle \mathbb{Z} \rangle \rangle$) {
  $(\forall i : \mathbb{Z})(rango(m, i) \longrightarrow_L |m[i]| > 0 \land$
  $(\forall j : \mathbb{Z})(rango(m, j) \longrightarrow_L |m[i]| = |m[j]|))$
  }
- /* El predicado validarRangos decide si un dato pasado por parámetro está en un rango válido.*/
  `pred validarRangos` (m : $seq\langle seq\langle \mathbb{Z} \rangle \rangle$, n, min, max : $\mathbb{Z}$) {
  $(\forall i : \mathbb{Z})(rango(m, i) \longrightarrow_L (min \leq m[i][n]) \leq max)$
  }
- /* El predicado validarExistencia valida el largo de una matriz, y que tenga la cantidad correcta de datos.*/
  `pred validarExistencia` (m: $seq\langle seq\langle \mathbb{Z} \rangle \rangle$, n: $\mathbb{Z}$) {
  $esMatriz(m) \land |m[0]| = n$
  }
- /* El predicado existenciaAsociacion valida si no hay individuos sin hogares ni hogares sin individuos.*/
  `pred existenciaAsociacion` (a, b: $seq\langle seq\langle \mathbb{Z} \rangle \rangle$) {
  $(\forall i : \mathbb{Z})(rango(a, i)) \longrightarrow_L ((\exists j : \mathbb{Z})(rango(b, j)) \longrightarrow_L a[i][@codigoHogar] = b[j][@codigoIndHogar]))$
  }

- /* El predicado sinHogaresRepetidos verifica hogar por hogar, para comprobar si se repite alguno. */
  `pred sinHogaresRepetidos` (th : eph_h) {

  $(\forall i : \mathbb{Z})(rango(th, i) \longrightarrow_L (apariciones(th, th[i]) = 1))$

  }

- /* El predicado sinIndividuosRepetidos valida que en un mismo hogar no se repitan individuos. */
  `pred sinIndividuosRepetidos` (ti : eph_i) {

  $(\forall i : \mathbb{Z})(\forall j : \mathbb{Z})((rango(ti, u) \land rango(ti, j)) \land (i \neq j) \land_L (ti[i][@codigoIndhogar] = ti[j][@codigoIndhogar])) \longrightarrow_L$
  $ti[i][@componente] \neq ti[i][@componente]))$

  }

- /* El predicado mismoAñoTrimTHTI verifica que la primer fila de th y ti tengan mismo año y trimestre, y luego que cada tabla tenga el mismo año y trimestre que la primer fila de cada una. */
  `pred mismoAñoTrimTHTI` (th : eph_h,ti : eph_i) {

  $th[0][@aoHogar] = ti[0][@aoIndividuo] \land$
  $th[0][@trimestreHogar] = th[0][@trimestreIndividuo] \land$
  $validarAoTrim(th, @hogAo, @hogTrimestre) \land_L validarAoTrim(ti, @indAo, @indTrimestre)$

  }

- /* El predicado validarAñoTrim verifica que para todas las filas, tengan el mismo año y trimestre. */
  `pred validarAñoTrim` (m: $seq\langle seq\langle \mathbb{Z} \rangle\rangle$, a : $\mathbb{Z}$, b : $\mathbb{Z}$) {

  $(\forall i : \mathbb{Z})(0 \leq i \leq (|m| - 1) \longrightarrow_L (m[i][a] = m[i + 1][a]) \land_L (m[i][b] = m[i + 1][b])$

  }

- /* El predicado verificarCantIndividuos valida que el total de habitantes por hogar sea menor o igual a 20. */
  `pred verificarCantIndividuos` (th : eph_h,ti : eph_i) {

  $(\forall i : \mathbb{Z})(rango(th, i) \longrightarrow_L (\sum_{j=0}^{|ti|-1}(\mathsf{if}\ th[i][@hogCodusu] = ti[j][@indCodusu])\ \mathsf{then}\ 1\ \mathsf{else}\ 0\ \mathsf{fi}) \leq 20))$

  }

- /* El predicado validarHabitaciones verifica que la cantidad total de habitaciones por hogar, sea menor a la cantidad de ambientes, y que por lo menos haya una habitación. */
  `pred validarHabitaciones` (th : eph_h) {

  $(\forall i : \mathbb{Z})(rango(th, i) \longrightarrow_L th[i][@cantidadAmbientes] \geq th[i][@cantidadHabitaciones]) \land_L$
  $(th[i][@cantidadHabitaciones] > 0)$

  }

2.2. Ejercicio 2

proc histHabitacional (**in** th : eph_h, **in** ti : eph_i,**in** region : $\mathbb{Z}$, **out** res : $seq\langle\mathbb{Z}\rangle$) {
Pre $\{validarEncuesta(th, ti) \wedge (1 \leq region \leq 6)\}$
Post $\{validarCantHabs(th, res, region) \wedge validarMax(th, |res|, region)\}$
}

- /* El predicado validarCantHabs verifica que en la secuencia res, en cada posición i de res, haya la cantidad de hogares en la región dada de tipo casa y con i habitaciones. */
  pred validarCantHabs (th : eph_h, res : $seq\langle\mathbb{Z}\rangle$, region : $\mathbb{Z}$) {
  $(\forall i : \mathbb{Z})(rango(res, i) \longrightarrow_L (res[i] = \sum_{j=0}^{|th|-1}(\text{if } cumpleCondicion(th, j, region) \wedge_L$
  $i = th[j][@cantidadHabitaciones] \text{ then } 1 \text{ else } 0 \text{ fi})))$
  }
- /* El predicado validarMax valida que no exista algún hogar que sea de tipo casa y sea de la región dada, que tenga mayor cantidad de habitaciones que res[m] dónde m es el máximo de habitaciones que cumplen esas condiciones.*/
  pred validarMax (th : eph_h, longRes, region : $\mathbb{Z}$) {
  $\neg((\exists j : \mathbb{Z})(rango(th, j) \longrightarrow_L cumpleCondicion(th, j, region) \wedge_L th[j][@cantidadHabitaciones]) > longRes))$
  }
- /* El predicado cumpleCondición verifica que un hogar dado, sea de tipo casa y de la región dada. */
  pred cumpleCondicion (th : eph_h, j, region : $\mathbb{Z}$) {
  $th[j][@region] = region \wedge th[j][@tipoHogar] = 1$
  }

2.3. Ejercicio 3

proc laCasaEstaQuedandoChica (**in** th : eph_h, **in** ti : eph_i, **out** res : $seq\langle\mathbb{R}\rangle$) {
Pre $\{validarEncuesta(th, ti)\}$
Post $\{(|res| = 7) \wedge (proporcionCorrecta(res) \wedge (res[0] = 0) \wedge (\forall i : \mathbb{Z})(0 < i < |res|) \longrightarrow (res[i] = cocienteHogares(th, ti, i)))\}$
}

- /* Verifica que cada elemento de la secuencia res, sea un valor de porcentaje válido. */
  pred proporcionCorrecta (res : $seq\langle\mathbb{R}\rangle$) {
  $(\forall i : \mathbb{Z})(rango(res, i) \longrightarrow (0 \leq res[i] \leq 1)$
  }
- /* Valida si un hogar se localiza en una ciudad con más de 500000 habitantes. */
  pred hogarEnCiudadGrande (h : hogar) {
  $(h[@enCiudadGrande] = 1)$
  }
- aux cocienteHogares (th : eph_h, ti : eph_i, r : $\mathbb{Z}$) : $\mathbb{R}$ = $hogaresConHacinamiento(th, ti, r))/(hogaresXRegion(th, r)$;
- aux cocienteIndividuos (th : eph_h, ti : eph_i, h : $\mathbb{Z}$) : $\mathbb{R}$ = if $th[h][@cantidadHabitaciones] = 0$ then $(individuosMismoHogar(ti,$
- /* Cantidad de habitantes de un hogar dado. */
  aux individuosMismoHogar (ti : eph_i, codHogar : $\mathbb{Z}$) : $\mathbb{Z}$ =
  $\sum_{p=0}^{|ti|-1}$(if $ti[p][@indCodusu] = codHogar$ then 1 else 0 fi) ;
- aux hogaresXRegion (th : eph_h, reg : $\mathbb{Z}$) : $\mathbb{Z}$ = $\sum_{i=0}^{|th|-1}$(if $th[i][@region]) = reg$ then 1 else 0 fi) ;
- aux hogaresConHacinamiento (th : eph_h, ti : eph_i, r : $\mathbb{Z}$) : $\mathbb{Z}$ =
  $\sum_{h=0}^{|th|-1}$(if $(\neg(hogarEnciudadGrande(th[h]))) \wedge_L th[h][@region] = r \wedge cocienteIndividuos(th, ti, h) > 3)$
  then 1 else 0 fi) ;

2.4. Ejercicio 4

proc `creceElTeleworkingEnCudadesGrandesl` (**in** t1h : eph_h, **in** t1i : eph_i, **in** t2h : eph_h, **in** t2i : eph_i, **out** res : Bool) {
`Pre` $\{validarEncuesta(t1h, t1i) \land validarEncuesta(t2h, t2i) \land$
$(t1h[0][@hogAo] < t2h[0][@hogAo]) \land$
$(t1h[0][@hogTrimestre] = t2h[0][@hogTrimestre])\}$
`Post` $\{res = true \leftrightarrow (proporcion(t1h, t1i) < proporcion(t2h, t2i))\}$
}

- /* Si no existen hogares en ciudades con más de 500000 habitantes el divisor sería 0. */
`aux proporción` (th : eph_h, ti : eph_i) : $\mathbb{R}$ =
if $hogaresEnCiudadGrande(th, ti) \neq 0$ then $individuosTeleworking(th, ti)/hogaresEnCiudadGrande(th, ti)$ else 0 fi ;

- /* Cantidad de individuos que trabajan en su hogar. */
`aux individuosTeleworking` (th : eph_h, ti : eph_i) : $\mathbb{Z}$ =
$\sum_{p=0}^{|ti|-1}$(if $(hogarEnCiudadGrande(th[hogarAsociado(th, ti[p])]) \land_L personaTrabajaEnCasa(ti[p], hogarAsociado(th, ti[p]))$
then 1 else 0 fi) ;

- /* Da el numero i de indexación fila, correspondiente al hogar asociado a un individuo. */
`aux hogarAsociado` (th : eph_h, p : individuo) : $\mathbb{Z}$ =
$\sum_{i=0}^{|th|-1}$ if $(p[@codigoindHogar] = th[i][@codigoHogar])$ then i else 0 fi) ;

- /* Valida si un individuo trabaja desde su casa y si tiene un lugar de trabajo en su hogar. */
`pred personaTrabajaEnCasa` (p : individuo, h : hogar) {
$h[@tieneAmbienteExclusivoDeTrabajo] = 1 \land$
$(h[@tipoHogar] = 1 \lor (h[@tipoHogar] = 2) \land$
$p[@estado] = 1 \land p[@lugarDeTrabajo] = 6$
}

- /* Cantidad de hogares que pertenecen a ciudades con más de 500000 habitantes. */
`aux #hogaresCiudadGrande` (th : eph_h, ti : eph_i) : $\mathbb{Z}$ =
$\sum_{i=0}^{|ti|-1}$(if $HogarEnCiudadGrande(hogarAsociado(th, ti[i]))$ then 1 else 0 fi) ;

2.5. Ejercicio 5

proc costoSubsidioMejora (**in** th : eph_h, **in** ti : eph_i, **in** monto : $\mathbb{Z}$, **out** res : $\mathbb{Z}$) {
Pre $\{validarEncuesta(th, ti) \wedge_L (monto > 0)\}$
Post $\{res = \sum_{i=0}^{|th|-1}(\text{if } hogaresParaSubsidio(th, ti, i) \text{ then } monto \text{ else } 0 \text{ fi})\}$
}

- /* El predicado hogaresParaSubsidio verifica si un hogar cumple con las condiciones para que se le otorgue un subsidio. */
  pred hogaresParaSubsidio (th : eph_h,ti : eph_i, i : $\mathbb{Z}$) {
  $(th[i][@tipoHogar] = 1) \wedge_L (th[i][@regimenDeTenencia] = 1) \wedge_L$
  $(th[i][@cantidadHabitaciones] < (cantHabitantesDelHogar(th, ti, i) - 2))$
  }
- /* El predicado cantHabitantesDelHogar devuelve la cantidad de habitantes de un hogar dado. */
  aux cantHabitantesDelHogar (th : eph_h, ti : eph_i, i : $\mathbb{Z}$) : $\mathbb{Z}$ =
  $\sum_{j=0}^{|ti|}(\text{if } th[i][@hogCodusu] = ti[j][@indCodusu] \text{ then } 1 \text{ else } 0 \text{ fi})$;

3. Segunda Parte

3.1. Ejercicio 6

proc `generarJoin` (**in** th : eph_h, **out** ti : eph_i, **out** junta : joinHI) {
`Pre` $\{validarEncuesta(th, ti)\}$
`Post` $\{(|junta| = |ti|) \land duplaValida(junta) \land elementosEquivalentes(th, ti, junta)\}$
}

- /* individuosMismoHogar: 3.8, hogarAsociado: 4.3 */
  /*como en ti cada individuo es unico, implica que en los individuos en junta son unicos */
  `pred elementosEquivalentes` (th: eph_h, ti: eph_i, junta: joinHI) {
  $((\forall j : \mathbb{Z})(rango(th, j) \longrightarrow_L \#aparicionesHogDupla(junta, th[j]) = individuosMismoHogar(ti, th[j][@hogCodusu])) \land$
  $(\forall i : \mathbb{Z})(rango(ti, i) \longrightarrow_L \#aparicionesHogDupla(junta, ti[i]) = 1)$
  }
- /* Apariciones de p, $p \in seq\langle\mathbb{Z}\rangle$ en $n$esimo lugar */
  `aux #aparicionesIndDupla` (junta: joinHI, s: $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z} = \sum_{i=0}^{|junta|-1}$(if $esPermutacion(junta[i]_1, s)$ then 1 else 0 fi) ;
- /* Apariciones de p, $p \in seq\langle\mathbb{Z}\rangle$ en $n$esimo lugar */
  `aux #aparicionesHogDupla` (junta: joinHI, s: $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z} = \sum_{i=0}^{|junta|-1}$(if $esPermutacion(junta[i]_0, s)$ then 1 else 0 fi) ;
  `pred esPermutacion` (S1 : $seq\langle T\rangle$, S2 : $seq\langle T\rangle$, e: T) {
  $(\forall e : T)(apariciones(S1, e) = apariciones(S2, e)$
  }
- /* Verifica cada dupla tiene un individuo con su hogar*/
  `pred duplaValida` (junta:joinHI) {
  $\prod_{i=0}^{|junta|-1}$(if $junta[i]_0[@hogcodusu] = junta[i]_1[@indCodusu]$ then 1 else 0 fi) $= 1$
  }

3.2. Ejercicio 7

proc ordenarRegionYTipo (**inout** th: eph_h, **inout** ti: eph_i) {
Pre {$validarEncuesta(th, ti) \land th = th_0 \land ti = ti_0$}
Post {$listaHogaresOrdenada(th) \land listaIndividuosOrdenada(th, ti) \land |th| = |th_0| \land |ti| = |ti_0| \land$
$mismosElementos(th, th_0) \land mismosElementos(ti, ti_0)$}
}

- pred listaHogaresOrdenada (th: eph_h) {
  $estaOrdenadaPorRegion(th) \land estaOrdenadaPorCodigo(th)$
  }
- pred estaOrdenadaPorRegion (th: eph_h) {
  $(\forall i : \mathbb{Z})(\forall j : \mathbb{Z})((0 \leq i < |th| \land 0 \leq j < |th|) \longrightarrow_L (th[i][@region] \leq th[j][@region] \longrightarrow i \leq j))$
  }
- pred estaOrdenadaPorCodigo (th: eph_h) {
  $(\forall i : \mathbb{Z})(\forall j : \mathbb{Z})((0 \leq i < |th| \land 0 \leq j < |th| \land_L pertenecenALaMismaRegion(i, j, th)) \longrightarrow_L$
  $((th[i][@hogCodusu] \leq th[j][@hogCodusu]) \longrightarrow i \leq j))$
  }
- pred pertenecenALaMismaRegion (fila1: $\mathbb{Z}$, fila2: $\mathbb{Z}$, th: eph_h) {
  $th[fila1][@region] = th[fila2][@region]$
  }
- /* El predicado valida si todos los individuos están ordenados por hogares y por componente. */
  pred listaIndividuosOrdenada (th: eph_h, ti: eph_i) {
  $estaOrdenadaPorHogares(th, ti) \land estaOrdenadaPorComponentes(ti)$
  }
- /* Verifica que todos los individuos estén ordenados por sus hogares, según cómo los hogares quedaron ordenados en la tabla th */
  pred estaOrdenadaPorHogares (th: eph_h, ti: eph_i) {
  $(\forall i : \mathbb{Z})(\forall j : \mathbb{Z})((0 \leq i < |ti| \land 0 \leq j < |ti|)$
  $\longrightarrow_L (aparecePrimeroEnTh(th, ti[i][@indCodusu], ti[j][@indCodusu]) \longrightarrow i \leq j))$
  }
- /* Dados un codusu1 y un codusu2 me dice si la fila de th correspondiente a codusu1 aparece antes que la fila correspondiente a codusu2.*/
  pred aparecePrimeroEnTh (th: eph_h, codusu1: $\mathbb{Z}$, codusu2: $\mathbb{Z}$) {
  $(\exists i : \mathbb{Z})(\exists j : \mathbb{Z})(0 \leq i < |th| \land 0 \leq j < |th| \land i \neq j$
  $\land_L th[i][@hogCodusu] = codusu1 \land_L th[j][@hogCodusu] = codusu2 \land i < j)$
  }
- /*Verifica si todos los individuos pertenecientes a un mismo hogar, están ordenados por su componente. */
  pred estaOrdenadaPorComponentes (ti: eph_i) {
  $(\forall i : \mathbb{Z})(\forall j : \mathbb{Z})((0 \leq i < |ti| \land 0 \leq j < |ti| \land pertenecenAlMismoHogar(i, j, ti)) \longrightarrow_L$
  $((ti[i][@indCodusu] \leq ti[j][@indCodusu]) \longrightarrow i \leq j))$
  }
- pred pertenecenAlMismoHogar (fila1: $\mathbb{Z}$, fila2: $\mathbb{Z}$, ti: eph_i) {
  $ti[fila1][@indCodusu] = ti[fila2][@indCodusu]$
  }
- pred mismosElementos (t1,t2: $seq\langle seq\langle \mathbb{Z} \rangle \rangle$) {
  $(\forall i : \mathbb{Z})(0 \leq i < |t1| \longrightarrow_L apariciones(t1[i], t1) = apariciones(t1[i], t2))$
  }

3.3. Ejercicio 8

proc `muestraHomogenea` (**in** th : eph_h, **out** ti : eph_i, **out** res : $seq\langle hogar\rangle$) {
`Pre` $\{validarEncuesta(th, ti)\}$
`Post` $\{((\exists s : seq < hogar >)(incluido(s, th) \land |s| \geq 3 \land laDifDeIngresosEsIgualParaHogaresConsec(s, th, ti)) \land res = s) \lor ((\neg(\exists s : seq < hogar >)(incluido(s, th) \land |s| \geq 3 \land laDifDeIngresosEsIgualParaHogaresConsec(s, th, ti)) \land res =<>) \land esLaMayorPosible(res) \land estaOrdenada(res)\}$
}

- /* El predicado incluido valida si en una secuencia de hogares, cada hogar pertenece a la th. */
  `pred incluido` (s: $seq < hogar >$, th: eph_h) {
  $(\forall i : \mathbb{Z})((0 \leq i < |s|) \longrightarrow_L s[i] \in th)$
  }
- `pred laDifDeIngresosEsIgualParaHogaresConsec` (s: $seq < hogar >$, th: eph_h, ti: eph_i) {
  $(\forall i : \mathbb{Z})(\forall j : \mathbb{Z})((0 \leq i < |s| - 1 \land 0 \leq j < |s| - 1) \longrightarrow_L$
  $((ingresoDelHogar(s[i + 1][@hogCodusu], th, ti) - ingresoDelHogar(s[i][@hogCodusu], th, ti)) =$
  $(ingresoDelHogar(s[j + 1][@hogCodusu], th, ti) - ingresoDelHogar(s[j][@hogCodusu], th, ti))))$
  }
- /* Dado un hogCodusu, devuelve la cantidad total de ingresos de ese hogar, considerando a los ingresos de todos sus integrantes. */
  `aux ingresoDelHogar` (codusu: $\mathbb{Z}$, ti: eph_i) : $\mathbb{Z}$ =
  $\sum_{i=0}^{|ti|-1}$(if $ti[i][@indCodusu] = codusu \land ti[i][@ingreso] \neq -1$) then $ti[i][@ingreso]$ else 0 fi) ;
- `pred esLaMayorPosible` (s: $seq < hogar >$, th: eph_h, ti: eph_i) {
  $\neg(\exists s2$:seq<hogar>$)(incluido(s2, th) \land laDifDeIngresosEsIgualParaHogaresConsec(s2, th, ti) \land |s2| \geq 3 \land |s2| > |s|)$
  }
- `pred estaOrdenada` (s: $seq < hogar >$, th: eph_h, ti: eph_i) {
  $(\forall i : \mathbb{Z})(\forall j : \mathbb{Z})((0 \leq i < |s| \land 0 \leq j < |s| \land i < j) \longrightarrow_L$
  $(ingresoDelHogar(s[i][@hogCodusu], th, ti) < ingresoDelHogar(s[j][@hogCodusu], th, ti)))$
  }

3.4. Ejercicio 9

proc corregirRegion (**inout** th : eph_h, **in** ti : eph_i) {
Pre $\{validarEncuesta(th, ti) \land th = th_0\}$
Post $\{|th| = |th_0| \land_L noExisteHogarEnGBA(th) \land comparacionPorHogar(th_0, th) \land comparacionPorCantidad(th_0, th)\}$
}

- /* Valido que no existen hogares en la región GBA. */
  pred noExisteHogarEnGBA (th: eph_0) {
  $(\forall i : \mathbb{Z})((0 \leq i < |th|) \longrightarrow_L th[i][@region] \neq 1)$
  }
- /*Si s2 cambiando el elemento en i posición por e, es igual a s1*/
  pred isSetAt (s1 : $seq\langle T \rangle$, s2 : $seq\langle T \rangle$, i: $\mathbb{Z}$, e: T) {
  $((0 \leq i < |s2|) \longrightarrow_L (s1 = setAt(s2, i, e)) \land \neg(0 \leq i < |s2|) \longrightarrow s1 = s2))$
  }
- /* al revisar por indexacion implica que th mantiene el orden de th_0 */
  pred comparacionPorHogar (th_0 : eph_h, th : eph_h) {
  $(\forall i : \mathbb{Z})((0 \leq i < |th|) \longrightarrow_L (th_0[i] = th[i] \lor isSetAt(th_0[i], th[i], @region, 1)))$
  }
- /* Valido que la cantidad de hogares en la región Pampeana, de la nueva th corregida, es igual a la cantidad de hogares de la región Pampeana + la cantidad de hogares de la región GBA de la tabla th sin corregir.*/
  pred comparacionPorCantidad (th_0 : eph_h, th : eph_h) {
  $\#hogaresPorRegion(th, 5) = \#hogaresPorRegion(th_0, 5) + \#hogaresPorRegion(th_0, 1)$
  }
- aux #hogaresPorRegion (th: eph_h, n: $\mathbb{Z}$) : $\mathbb{Z}$ = $\sum_{i=0}^{|th|-1}$(if $th[i][@region] = n$ then 1 else 0 fi) ;

3.5. Ejercicio 10

proc `histogramaDeAnillosConcentricos` (**in** th : eph_h, **in** centro : $\mathbb{Z} \times \mathbb{Z}$, **in** distancias : $seq\langle\mathbb{Z}\rangle$, **out** result : $seq\langle\mathbb{Z}\rangle$) {
`Pre` $\{validarTH(th) \land (apariciones(distancia, 0) = 0) \land esCreciente(distancias) \land (-90 \leq centro_0 \leq 90) \land (-180 \leq centro_1 \leq 180)\}$
`Post` $\{|distancias| = |result| \land (result[0] = (hogaresEnXAnillo(th, 0, distancias[0], centro)) \land ((\forall i : \mathbb{Z})((1 \leq i < |result|) \longrightarrow_L (result[i] = (hogaresEnXAnillo(th, distancias[i-1], distancias[i])))))\}$
}

- `pred validarTH` (th : eph_h) {

  $esMatriz(th) \land validarExistencia(th, 12) \land validarRangos(th, @hogAo, 1869, 2021) \land validarRangos(th, @hogTrimestre, 1, 4) \land validarRangos(th, @hogLatitud, -90, 90) \land validarRangos(th, @hogLongitud, -180, 180) \land validarRangos(th, @regimenDeTenencia, 1, 3) \land validarRangos(th, @region, 1, 6) \land validarRangos(th, @enCiudadGrande, 0, 1) \land validarRangos(th, @tipoHogar, 1, 5) \land validarRangos(th, @tieneAmbienteExclusivoDeTrabajo, 1, 2) \land (\forall i : \mathbb{Z})(rango(th, i) \longrightarrow_L (th[i][@hogCodusu] > 0) \land_L (th[i][@cantidadAmbientes] > 0)) \land validarHabitaciones(th) \land sinHogaresRepetidos(th)$

  }

- `pred esCreciente` (s : $seq\langle\mathbb{Z}\rangle$) {

  $(\forall i : \mathbb{Z})((0 \leq i \leq |s| - 2) \longrightarrow_L (s[i] < s[i+1]))$

  }

- /* Devuelve la cantidad de hogares que se encuentran a una distancia d (dMin <= d <dMax) del centro.*/
  `aux hogaresEnXAnillo` (th : eph_h, dMin : $\mathbb{Z}$, dMax : $\mathbb{Z}$, centro : $\mathbb{Z} \times \mathbb{Z}$) : $\mathbb{Z}$ =
  $\sum_{i=0}^{|th|-1}$(if $dMin \leq distancia(centro_0, centro_1, th[i][@hogLatitud], th[i][@hogLongitud]) \leq dMax$ then 1 else 0 fi) ;

- /* Devuelve la distancia que hay entre un hogar y una ubicación. */
  `aux distancia` (latC : $\mathbb{Z}$, lonC : $\mathbb{Z}$, latH : $\mathbb{Z}$, lonH : $\mathbb{Z}$) : $\mathbb{R}$ =
  $\sqrt{(latC - latH)^2 + (lonC - lonH)^2}$;

3.6. Ejercicio 11

proc quitarIndividuos (**inout** th : eph_h, **inout** ti : eph_i, **in** busqueda : $seq\langle(ItemIndividuo, dato)\rangle$, **out** result : ($eph_h$,$eph_i$))
{
Pre $\{validarEncuesta(th, ti) \land validarTerminoDeBusqueda(busqueda) \land noHayItemsRepetidos(busqueda) \land th = th0 \land ti = ti0\}$
Post $\{validarIndividuosQuitados(result, busqueda) \land validarHogaresQuitados(result) \land$
$(\forall hq : \mathbb{Z})(rango(result_0, hq) \longrightarrow_L (apariciones(th, result_0[hq]) = 0)) \land$
$(\forall iq : \mathbb{Z})(rango(result_1, iq) \longrightarrow_L (apariciones(ti, result_1[iq]) = 0)) \land$
$(|th0| = |th| + |result_0|) \land (|ti0| = |ti| + |result_1|) \land$
$verificoContenidoFinal(th0, th) \land verificoContenidoFinal(th0, result_0) \land$
$verificoContenidoFinal(ti0, ti) \land verificoContenidoFinal(ti0, result_1)\}$
}

- /* *Veo si el índice de itemIndividuo que me pasan por búsqueda coincide con el índice de algún item de individuo, y si coincide, corroboro que su valor correspondiente, esté en un rango válido.* */
  pred validarTerminoDeBusqueda (busqueda : $seq\langle(ItemIndividuo, dato)\rangle$) {
  $(\forall i : \mathbb{Z})(rango(busqueda, i) \longrightarrow_L$
  $((busqueda[i]_0 = @indCodusu \longrightarrow (busqueda[i]_1 > 0)) \lor$
  $(busqueda[i]_0 = @componente \longrightarrow (busqueda[i]_1 > 0)) \lor$
  $(busqueda[i]_0 = @indAo \longrightarrow (1869 \leq busqueda[i]_1 \leq 2021)) \lor$
  $(busqueda[i]_0 = @indTrimestre \longrightarrow (1 \leq busqueda[i]_1 \leq 4)) \lor$
  $(busqueda[i]_0 = @genero \longrightarrow (1 \leq busqueda[i]_1 \leq 2)) \lor$
  $(busqueda[i]_0 = @edad \longrightarrow (0 \leq busqueda[i]_1 \leq 130)) \lor$
  $(busqueda[i]_0 = @nivelEducativo \longrightarrow (0 \leq busqueda[i]_1 \leq 1)) \lor$
  $(busqueda[i]_0 = @estado \longrightarrow (-1 \leq busqueda[i]_1 \leq 1)) \lor$
  $(busqueda[i]_0 = @categoriaOcupacional \longrightarrow (-1 \leq busqueda[i]_1)) \lor$
  $(busqueda[i]_0 = @ingreso \longrightarrow (0 \leq busqueda[i]_1 \leq 1)) \lor$
  $(busqueda[i]_0 = @lugarDeTrabajo \longrightarrow (1 \leq busqueda[i]_1 \leq 10))))$
  }
- pred noHayItemsRepetidos (busqueda : $seq\langle(ItemIndividuo, dato)\rangle$) {
  $(\forall i : \mathbb{Z})(rango(busqueda, i) \longrightarrow_L (\neg(\exists j : \mathbb{Z})(rango(busqueda, j) \land (i \neq j) \land_L busqueda[i]_0 = busqueda[j]_0))$
  }
- /* *Verifico que cada individuo quitado pertenezca a un hogar, asociado a algún individuo que cumpla todos los términos de la búsqueda.* */
  pred validarIndividuosQuitados (result : (eph_h,eph_i), busqueda : $seq\langle(ItemIndividuo, dato)\rangle$) {
  $(\forall i : \mathbb{Z})(rango(result_1, i) \longrightarrow_L ((\exists h : \mathbb{Z})(rango(result_1, h) \land$
  $(result_1[i][@indCodusu] = result_1[h][@indCodusu]) \land_L$
  $((\forall k : \mathbb{Z})(rango(busqueda, k) \longrightarrow_L (result_1[h][busqueda[k]_0)] = busqueda[k]_1)))))$
  }
- /* *Valido que todos los hogares quitados, le corresponden por lo menos a un individuo.* */
  pred validarHogaresQuitados (result : (eph_h,eph_i)) {
  $(\forall i : \mathbb{Z})(rango(result_0, i) \longrightarrow_L ((\exists j : \mathbb{Z})(rango(result_1, j) \land result_0[i][@codigoHogar] = result_1[j][@codigoIndHogar]))$
  }
- /* Recibo th0 o ti0, y verifico que los contenidos de los outputs coincidan. */
  pred verificoContenidoFinal (m : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$, tabla : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
  $(\forall i : \mathbb{Z})(rango(tabla, i) \longrightarrow_L ((\exists j : \mathbb{Z})(rango(m, j) \land_L tabla[i] = m[j]))$
  }