



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

TP de Especificación

Análisis Habitacional Argentino

8 de Septiembre de 2021

Algoritmos y Estructuras de Datos I

Grupo 02, comisión 11

Integrante	LU	Correo electrónico
Lakowsky, Manuel	511/21	mlakowsky@gmail.com
Lenardi, Juan Manuel	56/14	juanlenardi@gmail.com
Arienti, Federico	316/21	fa.arianti@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>

1. Problemas

1.1. proc. esEncuestaValida

1.1.1. Especificación:

```
proc esEncuestaValida (in th:  $eph_h$ , in ti:  $eph_i$ , out result: Bool) {  
  Pre {true}  
  Post {result = true  $\leftrightarrow$  sonTablasValidas(th, ti)}  
}
```

1.1.2. Predicados y funciones auxiliares:

```
pred sonTablasValidas (th:  $eph_h$ , ti:  $eph_i$ ) {  
  esTablaDeHogaresValida(th)  $\wedge$  esTablaDeIndividuosValida(ti)  
}
```

```
pred esTablaDeHogaresValida (th:  $eph_h$ , ti:  $eph_i$ ) {  
  esTabla(th, @largoItemHogar)  $\wedge_L$   
  ( $\forall i : \mathbb{Z}$ )( $0 \leq i < |th| \rightarrow_L$  (  
    codusuValido $_h$ (th, ti, i)  $\wedge$  añoYTrimestreCongruente $_h$ (th, th[i])  $\wedge$  attEnRango $_h$ (th[i])  
  ))  
}
```

```
pred esTablaDeIndividuosValida (th:  $eph_h$ , ti:  $eph_i$ ) {  
  esTabla(ti, @largoItemIndividuo)  $\wedge_L$   
  ( $\forall i : \mathbb{Z}$ )( $0 \leq i < |ti| \rightarrow_L$  (  
    codusuValido $_i$ (th, ti, i)  $\wedge$  añoYTrimestreCongruente $_i$ (th, ti[i])  $\wedge$  attEnRango $_i$ (ti[i])  $\wedge$   
    esComponenteValido(ti, ti[i])  
  ))  
}
```

```
pred codusuValido $_h$  (th:  $eph_h$ , ti:  $eph_i$ , i:  $\mathbb{Z}$ ) {  
  ( $\exists j : individuo$ )( $j \in ti \wedge_L$   
    th[i][@hogcodusu] = j[@indcodusu]  
  )  $\wedge$   
   $\neg(\exists j : \mathbb{Z})(0 \leq j < |th| \wedge i \neq j) \wedge_L$   
    th[i][@hogcodusu] = th[j][@hogcodusu]  
  )  
}
```

```
pred añoYTrimestreCongruente $_h$  (th:  $eph_h$ , h: hogar) {  
  h[@hogaño] = th[0][@hogaño]  $\wedge$  h[@hogtrimestre] = th[0][@hogtrimestre]  
}
```

```
pred attEnRango $_h$  (h: hogar) {  
   $0 \leq h[@hogcodusu] \wedge$   
   $1810 \leq h[@hogaño] \wedge$   
   $1 \leq h[@hogtrimestre] \leq 4 \wedge$   
   $-90 \leq h[@hoglatitud] \leq 90 \wedge$   
   $-180 \leq h[@hoglongitud] \leq 180 \wedge$   
   $1 \leq h[@ii7] \leq 3 \wedge$   
   $1 \leq h[@region] \leq 6 \wedge$   
   $0 \leq h[@mas_500] \leq 1 \wedge$   
   $1 \leq h[@iv1] \leq 5 \wedge$   
   $0 < h[@ii2] \leq h[@iv2] \wedge$   
   $1 \leq h[@ii3] \leq 2$   
}
```

```

pred codusuValidoi (th: ephh, ti: ephi, i: ℤ) {
  (∃h : hogar)(h ∈ th ∧L
    ti[i][@indcodusu] = h[@hogcodusu]
  ) ∧
  ¬(∃j : ℤ)((0 ≤ j < |th| ∧ i ≠ j) ∧L (
    ti[i][@indcodusu] = ti[j][@indcodusu] ∧ ti[i][@componente] = ti[j][@componente]
  ))
}

pred añoYTrimestreCongruentei (th: ephh, i: individuo) {
  i[@indaño] = th[0][@hogaño] ∧ i[@indtrimestre] = th[0][@hogtrimestre]
}

pred attEnRangoi (i: individuo) {
  0 ≤ i[@indcodusu] ∧
  1 ≤ i[@componente] ≤ 20 ∧
  1810 ≤ i[@indaño] ∧
  1 ≤ i[@indtrimestre] ≤ 4 ∧
  1 ≤ i[@ch4] ≤ 2 ∧
  0 ≤ i[@ch6] ∧
  0 ≤ i[@nivel_ed] ≤ 1 ∧
  -1 ≤ i[@estado] ≤ 1 ∧
  0 ≤ i[@cat_ocup] ≤ 4 ∧
  -1 ≤ i[@p47t] ∧
  1 ≤ i[@ppo4g] ≤ 10
}

pred esComponenteValido (ti: ephi, i: individuo) {
  i[@componente] = 1 ∨ (∃i2 : individuo)(i2 ∈ ti ∧L i[@componente] - 1 = i2[@componente])
}

```

1.1.3. Observaciones:

- se hace uso de diversos tipos y referencias definidos en 2.3 y 2.4.
- la función auxiliar *esTabla*, definida en 2.1., verifica que th y ti sean matrices del largo correcto y con al menos una entrada.
- los predicados *codusuValido* verifican, de forma cruzada, que los hogares tengan individuos asociados y viceversa, y que no estén repetidos.
- los predicados *añoYTrimestreCongruente* contrastan con la primer entrada de la tabla de hogares para asegurar la homogeneidad de los registros.
- el predicado *esComponenteValido* junto a *codusuValido_i*, y aplicado a todo individuo de la tabla, verifica que los componentes ocurran de forma continua, es decir sin saltos mayores a 1, a partir del primero. En consecuencia, basta con verificar éstos predicados, y que los componentes estén en el rango correcto para asegurar que no haya más de 20 individuos por hogar.
- consideramos que:
 - @hogcodusu y @indcodusu son estrictamente positivos.
 - @componente puede tomar valores entre 1 y 20 inclusive.
 - @hogaño y @indaño no pueden ser anteriores a la revolución de mayo.
 - @hogtrimestre y @indtrimestre toman valores entre 1 y 4 inclusive.
 - @hoglatitud representa la dirección *sur* con números negativos y *norte* con positivos.
 - @hoglongitud representa la dirección *oeste* con números negativos y *este* con positivos.
 - @ch6, al representar la edad, es mayor o igual a 0.
 - @iv2, la cantidad total de ambientes, es estrictamente mayor a 0.

1.2. proc. histHabitacional

1.2.1. Especificación:

```
proc histHabitacional (in th:  $eph_h$ , in ti:  $eph_i$ , in region:  $dato$ , out res:  $seq\langle\mathbb{Z}\rangle$ ) {  
  Pre { $sonTablasValidas(th, ti) \wedge 1 \leq region \leq 6 \wedge (\exists h : hogar)(laCasaEstaEnLaRegion(th, h, region))$ }  
  Post {  
     $maximoDeHabitaciones(th, region, res) \wedge$   
     $(\forall i : \mathbb{Z})(0 \leq i < |res| \rightarrow_L$   
       $res[i] = \#casasPorNroDeHabitaciones(th, k, i + 1)$   
    )}  
}
```

1.2.2. Predicados y funciones auxiliares:

```
pred laCasaEstaEnLaRegion (th:  $eph_h$ , h:  $hogar$ , region:  $dato$ ) {  
   $h \in th \wedge_L esHogarValidoParaHistograma(h, region)$   
}
```

```
pred esHogarValidoParaHistograma (h:  $hogar$ , region:  $dato$ ) {  
   $h[@region] = region \wedge h[@iv1] = 1$   
}
```

```
pred maximoDeHabitaciones (th:  $eph_h$ , region:  $dato$ , res:  $seq\langle\mathbb{Z}\rangle$ ) {  
   $(\exists h : hogar)(laCasaEstaEnLaRegion(th, h, region) \wedge_L$   
     $h[@iv2] = |res| \wedge (\forall h_2 : hogar)(laCasaEstaEnLaRegion(th, h_2, region) \rightarrow_L h[@iv2] \geq h_2[@iv2])$   
  )  
}
```

```
aux #casasPorNroDeHabitaciones (th:  $eph_h$ , region:  $dato$ , habitaciones:  $\mathbb{Z}$ ) :  $\mathbb{Z} =$ 
```

$$\sum_{h \in th} (\text{if } esHogarValidoParaHistograma(h, region) \wedge h[@iv2] = habitaciones \text{ then } 1 \text{ else } 0 \text{ fi});$$

1.2.3. Observaciones:

- se hace uso del predicado *sonTablasValidas* definido en 1.1.2.
- consideramos, mediante el predicado *laCasaEstaEnLaRegion* en la precondition, que no tiene sentido preguntarse sobre el histograma habitacional de una región si ésta no tiene hogares.
- el predicado *maximoDeHabitaciones* verifica que el largo de la resolución corresponda con la cantidad máxima de habitaciones en la tabla de hogares.

1.3. proc. laCasaEstaQuedandoChica

1.3.1. Especificación:

```
proc laCasaEstaQuedandoChica (in th:  $eph_h$ , in ti:  $eph_i$ , out res:  $seq(\mathbb{R})$ ) {  
  Pre { $sonTablasValidas(th, ti)$ }  
  Post { $|res| = 6 \wedge_L (\forall region : dato)(1 \leq region \leq 6 \longrightarrow_L res[region - 1] = \%hacinado(th, ti, region))$ }  
}
```

1.3.2. Predicados y funciones auxiliares:

```
pred  $\Omega NoVacioHacinamiento$  (th:  $eph_h$ , region:  $dato$ ) {  
   $(\exists h : hogar)(h \in th \wedge_L esHogarValidoParaHacinamiento(h, region))$   
}
```

```
pred  $esHogarValidoParaHacinamiento$  (h:  $hogar$ , region:  $dato$ ) {  
   $h[@region] = region \wedge h[@mas\_500] = 0 \wedge h[@iv1] = 1$   
}
```

```
pred  $casaHacinada$  (ti:  $eph_i$ , h:  $hogar$ , region:  $dato$ ) {  
   $esHogarValidoParaHacinamiento(h, region) \wedge \#individuosEnHogar(ti, h[@hogcodusu]) > 3 * h[@iv2]$   
}
```

```
aux  $\%hacinado$  (th:  $eph_h$ , ti:  $eph_i$ , region:  $dato$ ) :  $\mathbb{R} =$ 
```

```
  if  $\Omega NoVacioHacinamiento(th, region)$  then
```

$$\frac{\sum_{h \in th} (\text{if } casaHacinada(ti, h, region) \text{ then } 1 \text{ else } 0 \text{ fi})}{\sum_{h \in th} (\text{if } esHogarValidoParaHacinamiento(h, region) \text{ then } 1 \text{ else } 0 \text{ fi})}$$

```
  else 0 fi;
```

1.3.3. Observaciones:

- se hace uso de la función auxiliar $\#individuosEnHogar$ definida en 2.2.
- la función auxiliar $\%hacinado$ considera como espacio de probabilidad (Ω) a todos los hogares que cumplan con el predicado $esHogarValidoParaHacinamiento$.
- en el predicado $\%hacinado$ consideramos que si no hay hogares válidos en una región, entonces la proporción de hogares hacinados respecto a esa región es 0.

1.4. proc. creceElTeleworkingEnCiudadesGrandes

1.4.1. Especificación:

```
proc creceElTeleworkingEnCiudadesGrandes (in t1h:  $eph_h$ , in t1i:  $eph_i$ , in t2h:  $eph_h$ , in t2i:  $eph_i$ , out res: Bool) {  
  Pre {(sonTablasValidas(t1h, t1i)  $\wedge$  sonTablasValidas(t2h, t2i))  $\wedge_L$  esComparacionValida(t1h, t1i, t2h, t2i)}  
  Post {res = true  $\iff$  %teleworking(t1h, t1i) < %teleworking(t2h, t2i)}  
}
```

1.4.2. Predicados y funciones auxiliares:

```
pred esComparacionValida (t1h:  $eph_h$ , t1i:  $eph_i$ , t2h:  $eph_h$ , t2i:  $eph_i$ ) {  
  t1h[0][@hogtrimestre] = t2h[0][@hogtrimestre]  $\wedge$  t1h[0][@hogañño] < t2h[0][@hogañño]  
}
```

```
pred  $\Omega$ NoVacioTeleworking (th:  $eph_h$ ) {  
  ( $\exists h$  : hogar)( $h \in th \wedge_L$  esHogarValidoParaTeleworking(h))  
}
```

```
pred esHogarValidoParaTeleworking (h: hogar) {  
  h[@mas_500] = 1  $\wedge$  (h[@iv1] = 1  $\vee$  h[@iv1] = 2)  
}
```

```
pred haceTeleworking (th:  $eph_h$ , i: individuo) {  
  viveEnHogarValido(th, i)  $\wedge$  i[@ii3] = 1  $\wedge$  i[@ppo4g] = 6  
}
```

```
pred viveEnHogarValido (th:  $eph_h$ , i: individuo) {  
  esHogarValidoParaTeleworking(th[indiceHogarPorCodusu(th, i[@indcodusu])])  
}
```

```
aux %teleworking (th:  $eph_h$ , ti:  $eph_i$ ) :  $\mathbb{R}$  =
```

```
  if  $\Omega$ NoVacioTeleworking(th) then
```

$$\frac{\sum_{i \in ti} (\text{if } \text{haceTeleworking}(th, i) \text{ then } 1 \text{ else } 0 \text{ fi})}{\sum_{i \in ti} (\text{if } \text{viveEnHogarValido}(th, i) \text{ then } 1 \text{ else } 0 \text{ fi})}$$

```
  else 0 fi ;
```

1.4.3. Observaciones:

- se hace uso del predicado *indiceHogarPorCodusu* definido en 2.2. bajo la presunción de una encuesta válida.
- la función auxiliar *%teleworking* considera como espacio de probabilidad (Ω) a todos los individuos que cumplan con el predicado *viveEnHogarValido*.
- en el predicado *%teleworking* consideramos que si no hay hogares válidos para considerar, entonces la proporción de hogares respecto al total es 0.

1.5. proc. costoSubsidioMejora

1.5.1. Especificación:

```
proc costoSubsidioMejora (in th:  $eph_h$ , in ti:  $eph_i$ , in monto:  $\mathbb{Z}$ , out res:  $\mathbb{Z}$ ) {  
  Pre { $sonTablasValidas(th, ti) \wedge monto > 0$ }  
  Post { $res = monto * \sum_{h \in th} (if esHogarValidoParaSubsidio(ti, h) then 1 else 0 fi)$ }  
}
```

1.5.2. Predicados y funciones auxiliares:

```
pred esHogarValidoParaSubsidio (ti:  $eph_i$ , h: hogar) {  
   $h[@ii7] = 1 \wedge h[@iv1] = 1 \wedge \#individuosEnHogar(ti, h[@hogcodusu]) - 2 > h[@ii2]$   
}
```

1.5.3. Observaciones:

- consideramos que un subsidio es necesariamente un monto positivo y que, dado el objetivo final de la especificación debe ser mayor a 0.

1.6. proc. generarJoin

1.6.1. Especificación:

```
proc generarJoin (in th:  $eph_h$ , in ti:  $eph_i$ , out junta:  $joinHI$ ) {  
  Pre { $sonTablasValidas(th, ti)$ }  
  Post { $|junta| = |ti| \wedge sonDuplasValidas(th, ti, junta)$ }  
}
```

1.6.2. Predicados y funciones auxiliares:

```
pred sonDuplasValidas (th:  $eph_h$ , ti:  $eph_i$ , junta:  $joinHI$ ) {  
  hayDuplaParaTodoIndividuo(ti, junta)  $\wedge$   
  hogaresEnDuplaValidos(th, junta)  $\wedge$   
  codusuCoincide(junta)  
}
```

```
pred hayDuplaParaTodoIndividuo (ti:  $eph_i$ , junta:  $joinHI$ ) {  
  ( $\forall i : individuo$ )( $i \in ti \rightarrow_L$   
    ( $\exists j : \mathbb{Z}$ )( $0 \leq j < |junta| \wedge_L (junta[j])_1 = i$ )  
  )  
}
```

```
pred hogaresEnDuplaValidos (th:  $eph_h$ , junta:  $joinHI$ ) {  
  ( $\forall i : \mathbb{Z}$ )( $0 \leq i < |junta| \rightarrow_L$   
    ( $\exists h : hogar$ )( $h \in th \wedge (junta[i])_0 = h$ )  
  )  
}
```

```
pred codusuCoincide (junta:  $joinHI$ ) {  
  ( $\forall i : \mathbb{Z}$ )( $0 \leq i < |junta| \rightarrow_L$   
    ( $junta[i]_0[@hogcodusu] = (junta[i])_1[@indcodusu]$ )  
  )  
}
```

1.6.3. Observaciones:

- el predicado *hayDuplaParaTodoIndividuo* verifica tanto que estén todos los individuos de *ti* en la *junta* como que no hayan repetidos. Esto es así porque previamente se comprueba que $|junta| = |ti|$.

1.7. proc. ordenarRegionYTipo

1.7.1. Especificación:

```
proc ordenarRegionYTipo (inout th:  $eph_h$ , inout ti :  $eph_i$ ) {  
  Pre { $sonTablasValidas(th, ti) \wedge th = th_0 \wedge ti = ti_0$ }  
  Post { $lasTablasNoCambian(th, th_0, ti, ti_0) \wedge sonTablasOrdenadas(th, ti)$ }  
}
```

1.7.2. Predicados y funciones auxiliares:

```
pred lasTablasNoCambian (th:  $eph_h$ ,  $th_0$ :  $eph_h$ , ti:  $eph_i$ ,  $ti_0$ :  $eph_i$ ) {  
   $tienenLosMismosElementos(th, th_0) \wedge tienenLosMismosElementos(ti, ti_0)$   
}
```

```
pred sonTablasOrdenadas (th:  $eph_h$ , ti:  $eph_i$ ) {  
   $hogaresOrdenados(th) \wedge individuosOrdenados(th, ti)$   
}
```

```
pred hogaresOrdenados (th:  $eph_h$ ) {  
   $regionCreciente(th) \wedge codusuCreciente(th)$   
}
```

```
pred regionCreciente (th:  $eph_h$ ) {  
   $(\forall i : \mathbb{Z})(0 \leq i < |th| - 1 \rightarrow_L th[i][@region] \leq th[i + 1][@region])$   
}
```

```
pred codusuCreciente (th:  $eph_h$ ) {  
   $(\forall i : \mathbb{Z})(0 \leq i < |th| - 1 \wedge_L th[i][@region] = th[i + 1][@region]) \rightarrow_L$   
     $th[i][@hogcodusu] < th[i + 1][@hogcodusu]$   
  )  
}
```

```
pred individuosOrdenados (th:  $eph_h$ , ti:  $eph_i$ ) {  
   $codusuComoHogares(th, ti) \wedge componenteCreciente(ti)$   
}
```

```
pred codusuComoHogares (th:  $eph_h$ , ti:  $eph_i$ ) {  
   $(\forall i : \mathbb{Z})(0 \leq i < |th| - 1 \rightarrow_L$   
     $ordenadosDeADosCodusu(ti, th[i][@hogcodusu], th[i + 1][@hogcodusu])$   
  )  
}
```

```
pred ordenadosDeADosCodusu (ti:  $eph_i$ , cod1: dato, cod2: dato) {  
   $(\forall i, j : \mathbb{Z})(0 \leq i, j < |ti| \wedge_L$   
     $(ti[i][@indcodusu] = cod1 \wedge ti[j][@indcodusu] = cod2) \rightarrow$   
       $i < j$   
  )  
}
```

```
pred componenteCreciente (ti:  $eph_i$ ) {  
   $(\forall i : \mathbb{Z})(0 \leq i < |ti| - 1 \wedge_L ti[i][@indcodusu] = ti[i + 1][@indcodusu]) \rightarrow_L$   
     $ti[i][@componente] < ti[i + 1][@componente]$   
  )  
}
```

1.7.3. Observaciones:

- se hace uso del predicado *tienenLosMismosElementos* definido en 2.1.
- el predicado *ordenadosDeADosCodusu* considera que $cod1 < cod2$, ya que se evalúa luego de corroborar el predicado *hogaresOrdenados*. El mismo verifica que todo individuo en *ti* con el *cod1* tiene su índice menor al de todos los individuos en *ti* con el *cod2*.

- observamos que el caso $i = j$ resulta en el lado izquierdo de la implicación siendo falso. Dado que, necesariamente, $cod1 \neq cod2$.
- el predicado *codusuComoHogares* comprueba si *ordenadosDeADosCodusu* es verdadero o no para todos los *codusu* de a pares de hogares consecutivos en *th*. Considera que la tabla de hogares está ordenada por región y *codusu* creciente.
- ambos predicados, *codusuComoHogares* y *ordenadosDeADoscodusu*, funcionan en conjunto para verificar que, en la tabla de individuos ordenada, todos los individuos estén agrupados por *codusu*, y que estos sigan el mismo orden que el de los *hogcodusu* de la tabla de hogares ya ordenada.

1.8. proc. muestraHomogenea

1.8.1. Especificación:

```
proc muestraHomogenea (in th : ephh, in ti : ephi, out res : seq⟨hogar⟩) {  
  Pre {sonTablasValidas(th, ti)}  
  Post {  
    ((∃s : seq⟨hogar⟩)(esLaSecuenciaMasLarga(th, ti, s)) ∧ res = s) ∨  
    (¬(∃s : seq⟨hogar⟩)(esLaSecuenciaMasLarga(th, ti, s)) ∧ |res| = 0)  
  }  
}
```

1.8.2. Predicados y funciones auxiliares:

```
pred esLaSecuenciaMasLarga (th : ephh, ti : ephi, res : seq⟨hogar⟩) {  
  esSecuenciaHomogenea(th, ti, res) ∧  
  ¬(∃s : seq⟨hogar⟩)(esSecuenciaHomogenea(th, ti, s) ∧ |s| ≥ |res|)  
}
```

```
pred esSecuenciaHomogenea (th : ephh, ti : ephi, res : seq⟨hogar⟩) {  
  |res| ≥ 3 ∧  
  contieneHogaresValidos(th, res) ∧  
  ordenCrecienteEntreIngresos(ti, res) ∧  
  laDiferenciaEsConstante(ti, res)  
}
```

```
pred contieneHogaresValidos (th : ephh, res : seq⟨hogar⟩) {  
  (∀i : ℤ)(0 ≤ i < |res| →L res[i] ∈ th)  
}
```

```
pred ordenCrecienteEntreIngresos (ti : ephi, res : seq⟨hogar⟩) {  
  (∀i : ℤ)(0 ≤ i < |res| - 1 →L diferenciaEntreIngresosConsecutivos(ti, res, i) ≥ 0)  
}
```

```
aux diferenciaEntreIngresosConsecutivos (ti : ephi, res : seq⟨hogar⟩, i : ℤ) : ℤ =  
  ingresoPorHogar(ti, res[i + 1]) - ingresoPorHogar(ti, res[i]);
```

```
aux ingresoPorHogar (ti : ephi, h : hogar) : ℤ =
```

$$\sum_{i \in ti} \text{if } i[\text{@indcodusu}] = h[\text{@hogcodusu}] \text{ then } i[\text{@p47T}] \text{ else } 0 \text{ fi};$$

```
pred laDiferenciaEsConstante (ti : ephi, res : seq⟨hogar⟩) {  
  (∀i : ℤ)(0 ≤ i < |res| - 2 →L  
    diferenciaEntreIngresosConsecutivos(ti, res, i) =  
    diferenciaEntreIngresosConsecutivos(ti, res, i + 1)  
  )  
}
```

1.8.3. Observaciones:

■

1.9. proc. corregirRegion

1.9.1. Especificación:

```
proc corregirRegion (inout th : ephh, in ti : ephi) {  
  Pre {sonTablasValidas(th, ti) ∧ th = th0}  
  Post {|th| = |th0| ∧L (fueraDeGBAPermaneceSinCambios(th, th0) ∧ cambióGBAaPampeana(th, th0))}  
}
```

1.9.2. Predicados y funciones auxiliares:

```
pred fueraDeGBAPermaneceSinCambios (th, th0 : ephh) {  
  (∀i : ℤ)((0 ≤ i < |th| ∧ th0[i][@Region] ≠ 1) →L th[i] = th0[i])  
}
```

```
pred cambióGBAaPampeana (th, th0 : ephh) {  
  (∀i : ℤ)((0 ≤ i < |th| ∧ th0[i][@Region] = 1) →L estaEnRegionPampeana(th[i], th0[i]))  
}
```

```
pred estaEnRegionPampeana (h, h0 : hogar) {  
  (∀j : ℤ)((0 ≤ j < |h| ∧ j ≠ @Region) →L h[j] = h0[j]) ∧ h[@Region] = 5  
}
```

1.9.3. Observaciones:

- las condiciones suficientes en los predicados *fueraDeGBAPermaneceSinCambios* y *cambióGBAaPampeana* son complementarias, por lo que todo hogar que pertenece a th₀, pertenece a th (dada la salvedad que cambian las regiones). Y, dado que tienen el mismo largo, no puede haber hogares nuevos.

1.10. proc. histogramaDeAnillosConcentricos

1.10.1. Especificación:

```

proc histogramaDeAnillosConcentricos (in th:  $eph_h$ , in centro:  $\mathbb{Z} \times \mathbb{Z}$ , in distancias:  $seq\langle \mathbb{Z} \rangle$ , out res:  $seq\langle \mathbb{Z} \rangle$ ) {
  Pre {esTablaDeHogaresValida(th)  $\wedge$  esCentroValido(centro)  $\wedge$  sonDistanciasValidas(distancias)}
  Post {
    |res| = |distancias|  $\wedge_L$  (
      res[0] = #HogaresEnAnillo(th, centro, 0, distancias[0])  $\wedge$ 
      ( $\forall i : \mathbb{Z}$ )( $0 < i < |result| \rightarrow_L$ 
        result[i] = #HogaresEnAnillo(th, centro, distancias[i - 1], distancias[i])
      )
    )
  }
}

```

1.10.2. Predicados y funciones auxiliares:

```

pred esCentroValido (centro :  $\mathbb{Z} \times \mathbb{Z}$ ) {
   $-90 \leq centro_0 \leq 90 \wedge -180 \leq centro_1 \leq 180$ 
}

pred sonDistanciasValidas (s:  $seq\langle \mathbb{Z} \rangle$ ) {
   $|s| > 0 \wedge_L (s[0] > 0 \wedge (\forall i : \mathbb{Z})(0 \leq i < |s| - 1 \rightarrow_L s[i] < s[i + 1]))$ 
}

```

aux #*HogaresEnAnillo* (th : eph_h , centro: $\mathbb{Z} \times \mathbb{Z}$, desde: \mathbb{Z} , hasta: \mathbb{Z}) : \mathbb{Z} =

$$\sum_{h \in th} \text{if } \text{cuadrado}(\text{desde}) \leq \text{distancia}^2(h, \text{centro}) < \text{cuadrado}(\text{hasta}) \text{ then } 1 \text{ else } 0 \text{ fi};$$

aux cuadrado (n : \mathbb{Z}) : \mathbb{Z} = $n * n$;

aux distancia² (h: *hogar*, centro: $\mathbb{Z} \times \mathbb{Z}$) : \mathbb{Z} =

$$\text{cuadrado}(h[\text{@hoglatitud}] - \text{centro}_0) + \text{cuadrado}(h[\text{@hoglongitud}] - \text{centro}_1);$$

1.10.3. Observaciones:

- se hace uso del predicado *esTablaDeHogaresValida* definido en 1.1.2.
- dado que la pertenencia de un punto $P = (x, y)$ a un anillo concéntrico definido en el intervalo $[A, B)$, donde A y B denotan dos radios respecto al centro $C = (x_0, y_0)$, se define como:

$$A \leq \sqrt{(x - x_0)^2 + (y - y_0)^2} < B \quad (1)$$

por simple manipulación algebraica (elevando al cuadrado), la misma relación se mantiene para:

$$A^2 \leq (x - x_0)^2 + (y - y_0)^2 < B^2 \quad (2)$$

el predicado #*HogaresEnAnillo* hace uso de esta observación.

- cabe aclarar que el predicado *distancia*² devuelve necesariamente un entero, ya que \mathbb{Z} es un cuerpo respecto a la suma, resta y producto.

1.11. proc. quitarIndividuos

1.11.1. Especificación:

```
proc quitarIndividuos (inout th : ephh, inout ti : ephi, in busqueda : seq⟨(ItemIndividuo, dato)⟩, out result : (ephh, ephi))  
{  
  Pre {sonTablasValidas(th, ti) ∧ esBusquedaValida(busqueda) ∧ th = th0 ∧ ti = ti0}  
  Post {  
    tienenLosMismosElementos(th0, th ++ result0) ∧  
    tienenLosMismosElementos(ti0, ti ++ result1) ∧  
    losIndividuosEstanFiltrados(ti0, ti, result1, busqueda) ∧  
    losHogaresEstanFiltrados(th0, th, result0, ti0, busqueda)  
  }  
}
```

1.11.2. Predicados y funciones auxiliares:

```
pred esBusquedaValida (Q : seq⟨(ItemIndividuo, dato)⟩) {  
  /* Q = query */  
  (∀i : Z)(0 ≤ i < |Q| →L (  
    pideUnDatoValido(Q[i]) ∧ ¬(∃j : Z)((0 ≤ j < |Q| ∧ i ≠ j) ∧L (Q[i])0 = (Q[j])0))  
  )  
}
```

```
pred pideUnDatoValido (filtro : (ItemIndividuo, dato)) {  
  (itemIndividuo.ord(filtro0) = @indcodusu ∧ 0 ≤ filtro1) ∨  
  (itemIndividuo.ord(filtro0) = @componente ∧ 1 ≤ filtro1 ≤ 20) ∨  
  (itemIndividuo.ord(filtro0) = @indañño ∧ 1810 ≤ filtro1) ∨  
  (itemIndividuo.ord(filtro0) = @indtrimestre ∧ 1 ≤ filtro1 ≤ 4) ∨  
  (itemIndividuo.ord(filtro0) = @ch4 ∧ 1 ≤ filtro1 ≤ 2) ∨  
  (itemIndividuo.ord(filtro0) = @ch6 ∧ 0 ≤ filtro1) ∨  
  (itemIndividuo.ord(filtro0) = @nive_ed ∧ 0 ≤ filtro1 ≤ 1) ∨  
  (itemIndividuo.ord(filtro0) = @estado ∧ -1 ≤ filtro1 ≤ 1) ∨  
  (itemIndividuo.ord(filtro0) = @cat_ocup ∧ 0 ≤ filtro1 ≤ 4) ∨  
  (itemIndividuo.ord(filtro0) = @p47t ∧ -1 ≤ filtro1) ∨  
  (itemIndividuo.ord(filtro0) = @ppo4g ∧ 1 ≤ filtro1 ≤ 10)  
}
```

```
pred losIndividuosEstanFiltrados (original, filtrada, complemento : ephi, Q : seq⟨(ItemIndividuo, dato)⟩) {  
  (∀i : individuo)(i ∈ original →L (  
    (i ∈ complemento ∧ i ∉ filtrada) ⇔ esBusquedaExitosa(i, Q)  
  ))  
}
```

```
pred losHogaresEstanFiltrados (original, filtrada, complemento : ephh, ti : ephi, Q : seq⟨(ItemIndividuo, dato)⟩) {  
  (∀h : hogar)(h ∈ original →L (  
    (h ∈ complemento ∧ h ∉ filtrada) ⇔  
    (∀i : individuo)((i ∈ ti ∧ i[@indcodusu] = h[@hogcodusu]) →L (  
      esBusquedaExitosa(i, Q)  
    ))  
  ))  
}
```

```
pred esBusquedaExitosa (i : individuo, Q : seq⟨(ItemIndividuo, dato)⟩) {  
  (∀ filtro : (ItemIndividuo, dato))(filtro ∈ Q →L (  
    i[itemIndividuo.ord(filtro0)] = filtro1  
  ))  
}
```

1.11.3. Observaciones:

- se hace uso del predicado *tienenLosMismosElementos* definido en 2.1.

- el predicado *losHogaresEstanFiltrados* considera que un hogar debe ser filtrado sólo si todos los individuos que viven en él son filtrados.
- decidimos incorporar directamente el predicado *pideUnDatoValido* en esta sección, en vez de modularizar los predicados *attEnRango* definidos en 1.1.2., porque -para el alcance de este TPE- consideramos que complejizaría la lectura de ambos procedimientos en mayor medida de lo que les podría aportar.

2. Predicados y Auxiliares generales

2.1. Predicados Generales

```
pred esMatriz (s: seq⟨seq⟨T⟩⟩) {  
    (∀ fila : seq⟨T⟩)(fila ∈ s →L |fila| = |s[0]|)  
}  
  
pred esTabla (m: seq⟨seq⟨T⟩⟩, columnas: ℤ) {  
    |m| > 0 ∧L (|m[0]| = columnas ∧ esMatriz(m))  
}  
  
pred tienenLosMismosElementos (s1: seq⟨T⟩, s2: seq⟨T⟩) {  
    |s1| = |s2| ∧ (∀ i : T)(i ∈ s1 ⇔ i ∈ s2)  
}
```

2.2. Auxiliares Generales

```
aux #individuosEnHogar (ti: ephi, codusuh: dato) : ℤ = ∑i ∈ ti (if i[@indcodusu] = codusuh then 1 else 0 fi);  
  
/* indiceHogarPorCodusu asume codusuh existe en la tabla y es único */  
  
aux indiceHogarPorCodusu (th: ephh, codusuh: dato) : ℤ = ∑i=0|th|-1 if th[i][@hogcodusu] = codusuh then i else 0 fi;
```

2.3. Tipos y Enumerados

```
type dato = ℤ  
type individuo = seq⟨dato⟩  
type hogar = seq⟨dato⟩  
type ephi = seq⟨individuo⟩  
type ephh = seq⟨hogar⟩  
type joinHI = seq⟨hogar × individuo⟩  
  
enum ItemHogar {  
    hogcodusu, hogaño, hogtrimestre, hoglatitud, hoglongitud, ii7, region, mas_500, iv1, iv2, ii2, ii3  
}  
  
enum ItemIndividuo {  
    indcodusu, componente, indaño, indtrimestre, ch4, ch6, nivel_ed, cat_ocup, p47t, ppo4g  
}
```

2.4. Referencias

```
aux @hogcodusu : ℤ = itemHogar.ord(hogcodusu);  
aux @hogaño : ℤ = itemHogar.ord(hogaño);  
aux @hogtrimestre : ℤ = itemHogar.ord(hogtrimestre);  
aux @hoglatitud : ℤ = itemHogar.ord(hoglatitud);  
aux @hoglongitud : ℤ = itemHogar.ord(hoglongitud);  
aux @ii7 : ℤ = itemHogar.ord(ii7);  
aux @region : ℤ = itemHogar.ord(region);  
aux @mas_500 : ℤ = itemHogar.ord(mas_500);  
aux @iv1 : ℤ = itemHogar.ord(iv1);  
aux @iv2 : ℤ = itemHogar.ord(iv2);  
aux @ii2 : ℤ = itemHogar.ord(ii2);  
aux @ii3 : ℤ = itemHogar.ord(ii3);  
  
aux @indcodusu : ℤ = itemIndividuo.ord(indcodusu);  
aux @componente : ℤ = itemIndividuo.ord(componente);  
aux @indaño : ℤ = itemIndividuo.ord(indaño);
```



```

aux @indtrimestre :  $\mathbb{Z}$  = itemIndividuo.ord(indtrimestre);
aux @ch4 :  $\mathbb{Z}$  = itemIndividuo.ord(ch4);
aux @ch6 :  $\mathbb{Z}$  = itemIndividuo.ord(ch6);
aux @nivel_ed :  $\mathbb{Z}$  = itemIndividuo.ord(nivel_ed);
aux @cat_ocup :  $\mathbb{Z}$  = itemIndividuo.ord(cat_ocup);
aux @p47t :  $\mathbb{Z}$  = itemIndividuo.ord(p47t);
aux @ppo4g :  $\mathbb{Z}$  = itemIndividuo.ord(ppo4g);

aux @largoItemHogar :  $\mathbb{Z}$  = 12;
aux @largoitemIndividuo :  $\mathbb{Z}$  = 10;

```