



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  $\leftrightarrow$  validarEncuesta(th, ti)}  
}
```

1.1.2. Predicados y funciones auxiliares:

```
pred validarEncuesta (th:  $eph_h$ , ti:  $eph_i$ ) {  
  /* tabla hogares */  
  (esTabla(th, @largoItemHogar)  $\wedge_L$   
  ( $\forall h : hogar$ )( $h \in th \rightarrow_L$  (  
    codigoValidoh(th, ti, h)  $\wedge$  añoTrimestreCongruenteh(th, h)  $\wedge$  attEnRangoh(h)  
  )))  $\wedge$   
  /* tabla individuos */  
  (esTabla(ti, @largoItemIndividuo)  $\wedge_L$   
  ( $\forall i : individuo$ )( $i \in ti \rightarrow_L$  (  
    codigoValidoi(th, ti, i)  $\wedge$  añoTrimestreCongruentei(th, i)  $\wedge$  attEnRangoi(i)  $\wedge$   
    validarComponentei(ti, i)  
  )))  
}
```

```
pred codigoValidoh (th:  $eph_h$ , ti:  $eph_i$ , h: hogar) {  
  ( $\exists i : individuo$ )( $i \in ti \wedge_L$   
    h[@hogcodusu] = i[@indcodusu]  
  )  $\wedge$   
   $\neg$ ( $\exists h_2 : hogar$ )( $(h_2 \in th \wedge h_2 \neq h) \wedge_L$   
    h[@hogcodusu] = h2[@hogcodusu]  
  )  
}
```

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

```
pred attEnRangoh (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 codigoValidoi (th:  $eph_h$ , ti:  $eph_i$ , i: individuo) {  
  ( $\exists h : hogar$ )( $h \in th \wedge_L$   
    i[@indcodusu] = h[@hogcodusu]  
  )  $\wedge$   
   $\neg$ ( $\exists i_2 : individuo$ )( $(i_2 \in ti \wedge i_2 \neq i) \wedge_L$  (  
    i[@indcodusu] = i2[@indcodusu]  $\wedge$  i[@componente] = i2[@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 validarComponentei (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 *codigoValido* 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 *validarComponente_i* junto a *codigoValido_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 { $validarEncuesta(th, ti) \wedge (\exists h : hogar)(casaEnLaRegion(th, h, region))$ }  
  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 casaEnLaRegion (th:  $eph_h$ , h:  $hogar$ , region:  $dato$ ) {  
   $h \in th \wedge_L esHogarValido_{1,2}(h, region)$   
}
```

```
pred esHogarValido1,2 (h:  $hogar$ , region:  $dato$ ) {  
   $h[@region] = region \wedge h[@iv1] = 1$   
}
```

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

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

$$\sum_{h \in th} (\text{if } esHogarValido_{1,2}(h, region) \wedge h[@iv2] = habitaciones \text{ then } 1 \text{ else } 0 \text{ fi});$$

1.2.3. Observaciones:

- se hace uso del predicado *validarEncuesta* definido en 1.1.2.
- consideramos, mediante el predicado *casaEnLaRegion* 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 { $validarEncuesta(th, ti)$ }  
  Post { $|res| = 6 \wedge_L (\forall region : dato)(1 \leq region \leq 6 \rightarrow_L res[region - 1] = \%hacinado(th, ti, region))$ }  
}
```

1.3.2. Predicados y funciones auxiliares:

```
pred  $\Omega NoVacio_{1.3}$  (th:  $eph_h$ , region:  $dato$ ) {  
   $(\exists h : hogar)(h \in th \wedge_L esHogarValido_{1.3}(h, region))$   
}
```

```
pred  $esHogarValido_{1.3}$  (h:  $hogar$ , region:  $dato$ ) {  
   $h[@region] = region \wedge h[@mas\_500] = 0 \wedge h[@iv1] = 1$   
}
```

```
pred  $casaHacinada$  (ti:  $eph_i$ , h:  $hogar$ , region:  $dato$ ) {  
   $esHogarValido_{1.3}(h, region) \wedge \#individuosEnHogar(ti, h[@hogcodusu]) > 3 * h[@iv2]$   
}
```

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

$$\text{if } \Omega NoVacio_{1.3}(th, region) \text{ then } \frac{\sum_{h \in th} (\text{if } casaHacinada(ti, h, region) \text{ then } 1 \text{ else } 0 \text{ fi})}{\sum_{h \in th} (\text{if } esHogarValido_{1.3}(h, region) \text{ then } 1 \text{ else } 0 \text{ fi})} \text{ else } 0 \text{ fi ;}$$

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 $esHogarValido_{1.3}$.

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 {(validarEncuesta(t1h, t1i)  $\wedge$  validarEncuesta(t2h, t2i))  $\wedge_L$  comparacionValida(t1h, t1i, t2h, t2i)}  
  Post {res = true  $\iff$  %teleworking(t1h, t1i) < %teleworking(t2h, t2i)}  
}
```

1.4.2. Predicados y funciones auxiliares:

```
pred comparacionValida (t1h:  $eph_h$ , t1i:  $eph_i$ , t2h:  $eph_h$ , t2i:  $eph_i$ ) {  
  (t1h[0][@hogaño] = t2h[0][@hogaño] - 1  $\wedge$  t1h[0][@hogtrimestre] = t2h[0][@hogtrimestre])  
}
```

```
pred  $\Omega$ NoVacio1.4 (th:  $eph_h$ ) {  
  ( $\exists h$  : hogar)( $h \in th \wedge_L esHogarValido_{1.4}(h)$ )  
}
```

```
pred esHogarValido1.4 (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) {  
  esHogarValido1.4(th[indiceHogarPorCodusu(th, i[@indcodusu])])  
}
```

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

$$\text{if } \Omega NoVacio_{1.4}(th) \text{ then } \frac{\sum_{i \in ti} (\text{if } haceTeleworking(th, i) \text{ then } 1 \text{ else } 0 \text{ fi})}{\sum_{i \in ti} (\text{if } viveEnHogarValido(th, i) \text{ then } 1 \text{ else } 0 \text{ fi})} \text{ else } 0 \text{ fi};$$

1.4.3. Observaciones:

- se hace uso del predicado *indiceHogarPorCodusu* definido en 2.2. bajo la presunción de una encuesta válida.
- consideramos como comparación válida a aquella realizada entre encuestas de años consecutivos.
- la función auxiliar *%teleworking* considera como espacio de probabilidad (Ω) a todos los individuos que cumplan con el predicado *viveEnHogarValido*.

1.5. proc. costoSubsidioMejora

1.5.1. Especificación:

```
proc costoSubsidioMejora (in th:  $eph_i$ , in ti:  $eph_i$ , in monto:  $\mathbb{Z}$ , out res:  $\mathbb{Z}$ ) {  
  Pre { $validarEncuesta(th, ti) \wedge monto > 0$ }  
  Post { $res = monto * \sum_{h \in th} (if\ esHogarValido_{1.5}(ti, h) \text{ then } 1 \text{ else } 0 \text{ fi})$ }  
}
```

1.5.2. Predicados y funciones auxiliares:

```
pred esHogarValido1.5 (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.

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))  
}
```

2.2. Auxiliares Generales

```
aux #individuosEnHogar (ti: ephi, codusuh: dato) : ℤ = ∑i ∈ ti (if i[@indcodusu] = codusuh then 1 else 0 fi);
```

/ * indiceHogarPorCodusu asume codusu_h existe en la tabla y es único * /

```
aux indiceHogarPorCodusu (th: ephh, codusuh: dato) : ℤ = ∑h ∈ th if h[@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 : ℤ = itemIndividuo.ord(indtrimestre);  
aux @ch4 : ℤ = itemIndividuo.ord(ch4);  
aux @ch6 : ℤ = itemIndividuo.ord(ch6);  
aux @nivel_ed : ℤ = itemIndividuo.ord(nivel_ed);  
aux @cat_ocup : ℤ = itemIndividuo.ord(cat_ocup);  
aux @p47t : ℤ = itemIndividuo.ord(p47t);  
aux @ppo4g : ℤ = itemIndividuo.ord(ppo4g);  
  
aux @largoItemHogar : ℤ = 12;  
aux @largoitemIndividuo : ℤ = 10;
```