Ejercitación 1er parcial
Algoritmos y Estructuras de Datos II

Jonathan Seijo

Departamento de Computación,
Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires

23 de abril de 2018

Especificación

Especificación

Enunciado - I

Técnicos a Domicilio (o simplemente "TaD"), es una empresa que provee servicio técnico para problemas de electricidad en hogares y empresas. TaD cuenta con un grupo de técnicos altamente capacitados para atender la demanda de sus clientes y tiene una estrategia de trabajo algo particular.

Cuando alguien solicita un técnico, la central de TaD verifica si alguno de sus técnicos se encuentra en la empresa y de ser así envía inmediatamente un técnico al domicilio de la persona. En caso de no haber técnicos disponibles en ese momento (i.e., todos se encuentran atendiendo algún pedido), el pedido queda *pendiente de asignación* a la espera de que algún técnico se desocupe.

Especificación

Enunciado - II

Por otro lado, cuando un técnico termina de resolver un problema, y antes de retirarse de ese domicilio, el técnico avisa por radio a la central que quedó disponible para otro trabajo.

Si existiesen en ese momento pedidos *pendientes de asignación*, la central le asigna al técnico el más cercano al domicilio en el que éste se encuentra y el técnico se dirige automáticamente hacia allí (si hay más de un pendiente a la misma distancia mínima, se asignará al pedido entre éstos que lleve más tiempo esperando). Por el contrario, de no haber trabajos pendientes, el técnico regresa a la central y queda disponible para futuros trabajos.

Especificación

Se pide:

Modelar con un TAD la empresa *Técnicos a Domicilio* descripta teniendo en cuenta además que interesa saber, dada una dirección, quiénes fueron los técnicos que la visitaron la mayor cantidad de veces (aun si todavía no resolvieron el inconveniente técnico).

Observación: Se puede asumir como dado el TAD Dirección que exporta el género *dirección* y la operación *dist*(d, d') que devuelve un *nat* que representa la distancia entre las direcciones d y d'.

Especificación

Consejo

Entendamos bien el enunciado antes de empezar a escribir.

Especificación

Consejo

Entendamos bien el enunciado antes de empezar a escribir.

- Si hay varios técnicos en la central y se solicita un técnico, ¿importa quién atiende el pedido?

Especificación

Consejo

Entendamos bien el enunciado antes de empezar a escribir.

- Si hay varios técnicos en la central y se solicita un técnico, ¿importa quién atiende el pedido?
- Si **no** hay técnicos disponibles pero hay pedidos pendientes, ¿importa el orden de los pedidos?

Especificación

Consejo

Entendamos bien el enunciado antes de empezar a escribir.

- Si hay varios técnicos en la central y se solicita un técnico, ¿importa quién atiende el pedido?
- Si **no** hay técnicos disponibles pero hay pedidos pendientes, ¿importa el orden de los pedidos?
- Si un técnico no está en la central, ¿importa saber dónde está?

Especificación

Consejo

Entendamos bien el enunciado antes de empezar a escribir.

- ¿Podemos identificar algún comportamiento automático? ¿En qué etapa de la resolución deberíamos tenerlo en cuenta?

Especificación

Consejo

Entendamos bien el enunciado antes de empezar a escribir.

- ¿Podemos identificar algún comportamiento automático? ¿En qué etapa de la resolución deberíamos tenerlo en cuenta?
- El enunciado dice: "*interesa saber, dada una dirección, quiénes fueron los técnicos que la visitaron la mayor cantidad de veces*".
  ¿Necesitamos tener en cuenta algún historial de visitas?

Especificación

Entendimos qué nos piden, ¿y ahora qué?

Especificación

Entendimos qué nos piden, ¿y ahora qué?

Especificación

Consejo

Antes de axiomatizar, pensemos y definamos cuáles son **todos** nuestros observadores y generadores.

¿Por dónde comenzamos?

Especificación

Consejo

Antes de axiomatizar, pensemos y definamos cuáles son **todos** nuestros observadores y generadores.

¿Por dónde comenzamos?

Consejo

Empecemos con los observadores.

(¿Podría empezar pensando en los generadores?)

Observadores

Observadores

observadores básicos

técnicos : tad $\longrightarrow$ conj(*tecnico*)
técnicosLibres : tad $\longrightarrow$ conj(*tecnico*)
pendientes : tad $\longrightarrow$ secu(*direccion*)
direcciónActual : tecnico t_1 $\times$ tad T $\longrightarrow$ direccion

$$\{t_1 \in \text{técnicos}(T) \land t_1 \notin \text{técnicosLibres}(T)\}$$

cantVisitas : direccion d $\times$ tecnico t_1 $\times$ tad T $\longrightarrow$ nat

$$\{t_1 \in \text{técnicos}(T)\}$$

Igualdad Observacional

El enunciado no me pide explícitamente una igualdad observacional ¿debería hacerla?

Igualdad Observacional

El enunciado no me pide explícitamente una igualdad observacional ¿debería hacerla?

SI

Igualdad Observacional

igualdad observacional

$$(\forall t, t' : \text{tad}) \left(t =_{\text{obs}} t' \iff \left(\begin{array}{l} \text{pendientes}(t) =_{\text{obs}} \text{pendientes}(t') \wedge \\ \text{técnicos}(t) =_{\text{obs}} \text{técnicos}(t') \wedge \\ \text{técnicosLibres}(t) =_{\text{obs}} \text{técnicosLibres}(t') \wedge_{\text{L}} (\\ (\forall\ p : \text{técnico})\ (p \in \text{técnicos}(t) \Rightarrow_{\text{L}} (\forall\ d : \text{dirección}) \\ (\text{cantVisitas}(t, p, d) =_{\text{obs}} \text{cantVisitas}(t', p, d)) \wedge \\ (p \in \text{técnicosLibres}(t) \Rightarrow_{\text{L}} \\ \text{direcciónActual}(t, p) =_{\text{obs}} \text{direcciónActual}(t', p))))) \end{array} \right) \right)$$

Generadores

Consejo

Identificar partes estáticas y partes dinámicas.

¿Los técnicos se mantienen iguales o cambian?

Generadores

Generadores

$$\text{iniciar} : \text{conj}(\textit{tecnico}) \longrightarrow \text{tad}$$

$$\text{solicitar} : \text{direccion} \longrightarrow \text{tad}$$

$$\text{finalizar} : \text{tecnico } t_1 \times \text{tad } T \longrightarrow \text{tad}$$

$$\{t_1 \in \text{técnicos}(T) \wedge t_1 \notin \text{técnicosLibres}(T)\}$$

Otras operaciones

Consejo

No olvidar que el enunciado puede pedir funciones específicas que no son observadores ni generadores.

En este caso, queremos conocer cuáles son los técnicos que más visitaron cierta dirección.

$$\text{másVisitaron} : \text{direccion} \times \text{tad} \longrightarrow \text{conj}(\textit{tecnico})$$

Otras operaciones

¿Estaría bien si la función “másVisitaron” es un observador?

Otras operaciones

¿Estaría bien si la función “másVisitaron” es un observador?

NO

Axiomas

Mi axiomatización es perfecta pero es **inentendible** ¿está bien?

Axiomas

Mi axiomatización es perfecta pero es **inentendible** ¿está bien?

NO

Axiomas

¿Es grave romper la congruencia?

Axiomas

¿Es grave romper la congruencia?

SI

Axiomas

¿Puedo axiomatizar **todo** en base a los generadores?

Axiomas

¿Puedo axiomatizar **todo** en base a los generadores?

Consejo

Usen los observadores siempre que puedan.
Mucho cuidado con la congruencia.

Axiomas

$$\text{técnicos}(\text{iniciar}(ts)) \equiv ts$$
$$\text{técnicos}(\text{solicitar}(d, T)) \equiv \text{técnicos}(T)$$
$$\text{técnicos}(\text{finalizar}(t_1, T)) \equiv \text{técnicos}(T)$$

Axiomas

$$\text{técnicosLibres}(\text{iniciar}(ts)) \equiv \emptyset$$

$$\begin{aligned}\text{técnicosLibres}(\text{solicitar}(d,\ T)) \equiv\ & \textbf{if } \#\ \text{técnicosLibres}(T) \geq 1\ \textbf{ then} \\ & \quad \text{sinUno}(\ \text{técnicosLibres}(T)\ ) \\ & \textbf{else} \\ & \quad \emptyset \\ & \textbf{fi}\end{aligned}$$

$$\begin{aligned}\text{técnicosLibres}(\text{finalizar}(t_1,\ T)) \equiv\ & \textbf{if } \text{vacía?}(\ \text{pendientes}(T)\ )\ \textbf{ then} \\ & \quad \text{Ag}(t_1,\ \text{técnicosLibres}(T)) \\ & \textbf{else} \\ & \quad \text{técnicosLibres}(T) \\ & \textbf{fi}\end{aligned}$$

Axiomas

$$\text{pendientes}(\text{iniciar}(ts)) \equiv <>$$

$$\begin{aligned}\text{pendientes}(\text{solicitar}(d,\ T)) \equiv\ & \textbf{if } \#(\ \text{técnicosLibres}(T)\ ) \geq 1 \ \textbf{ then} \\ & <> \\ & \textbf{else} \\ & \quad \text{pendientes}(T) \circ d \\ & \textbf{fi}\end{aligned}$$

$$\begin{aligned}\text{pendientes}(\text{finalizar}(t_1,\ T)) \equiv\ & \textbf{if } \text{vacía?}(\ \text{pendientes}(T)\ ) \ \textbf{ then} \\ & <> \\ & \textbf{else} \\ & \quad \text{borrar}(\text{MasCercana}(t_1, T), \text{pendientes}(T)) \\ & \textbf{fi}\end{aligned}$$

Axiomas

$$\begin{aligned}\text{dirActual}(t_1,\ \text{solicitar}(d,\ T)) \equiv\ & \textbf{if } (\#(\ \text{técnicosLibres}(T)\ ) \geq 1 \\ & \wedge_{\text{L}} \text{dameUno}(\text{técnicosLibres}(T)) \quad = \quad t_1) \\ & \textbf{then} \\ & \quad d \\ & \textbf{else} \\ & \quad \text{dirActual}(t_1,\ T) \\ & \textbf{fi}\end{aligned}$$

$$\begin{aligned}\text{dirActual}(t_1,\ \text{finalizar}(t_2,\ T)) \equiv\ & \textbf{if } t_1 = t_2 \ \textbf{ then} \\ & \quad \text{MasCercana}(t_1,\ T) \\ & \textbf{else} \\ & \quad \text{dirActual}(t_1,\ T) \\ & \textbf{fi}\end{aligned}$$

Axiomas

$$\text{cantVisit}(d_1,\ t_1,\ \text{iniciar}(ts)) \equiv 0$$

$$\begin{aligned}\text{cantVisit}(d_1,\ t_1,\ \text{solicitar}(d_2,\ T)) \equiv\ & \textbf{if } (\#(\ \text{técnicosLibres}(T)\ ) \geq 1 \\ & \wedge_{\text{L}} t_1 = \text{dameUno}(\text{técnicosLibres}(T)) \\ & \wedge d_1 = d_2) \\ & \textbf{then} \\ & \quad 1 + \text{cantVisit}(d_1,\ t_1,\ T) \\ & \textbf{else} \\ & \quad \text{cantVisit}(d_1,\ t_1,\ T) \\ & \textbf{fi}\end{aligned}$$

Axiomas

$$
\begin{aligned}
\text{cantVisit}(d_1,\ t_1,\ \text{finalizar}(t_2,\ T)) \equiv\ & \textbf{if}\ t_1 = t_2 \\
& \wedge\ \text{long}(\ \text{pendientes}(T)\ ) \geq 1 \\
& \wedge_{\text{L}} \text{MasCercana}(t_1,\ T) = d_1\ \ \textbf{then} \\
& \quad 1 + \text{cantVisit}(d_1,\ t_1,\ T) \\
& \textbf{else} \\
& \quad \text{cantVisit}(d_1,\ t_1,\ T) \\
& \textbf{fi}
\end{aligned}
$$

Axiomas

$$\text{MasCercana} : \text{tecnico } t_1 \times \text{tad } T \longrightarrow \text{direccion}$$

$$\left\{ \begin{array}{l} t_1 \in \text{técnicos}(T) \\ \wedge \neg \text{vacia}(\text{pendientes}(T)) \end{array} \right\} \wedge t_1 \notin \text{técnicosLibres}(T) \Big\}$$

$$\text{MasCercana}(t_1,\ T) \equiv \text{MasCerca}(\ \text{dirActual}(t_1,\ T),\ \text{pendientes}(T)\ )$$

Axiomas

$$\text{MasCerca} : \text{direccion } d \times \text{secu}(\textit{direccion})\ ds \longrightarrow \text{direccion})$$

$$\{\neg\ \text{vacia}(\text{pendientes}(T))\}$$

$$\begin{aligned}
\text{MasCerca}(d,\ ds) \equiv\ & \textbf{if } \text{long}(ds) = 1 \textbf{ then} \\
& \quad \text{prim}(ds) \\
& \textbf{else} \\
& \quad \textbf{if } \text{dist}(d,\text{prim}(ds)) \leq \text{dist}(d,\text{MasCerca}(d,\text{fin}(ds))) \\
& \quad \textbf{then} \\
& \quad\quad \text{prim}(ds) \\
& \quad \textbf{else} \\
& \quad\quad \text{MasCerca}(d,\ \text{fin}(ds)) \\
& \quad \textbf{fi} \\
& \textbf{fi}
\end{aligned}$$

Axiomas

$$\text{másVisitaron} \ : \ \text{direccion} \times \text{tad} \ \longrightarrow \ \text{conj}(\textit{tecnico})$$

$$\begin{aligned} \text{másVisitaron}(d, \ T) \ \equiv \ & \text{LosDeCantVisitas}(d, \\ & \text{maxCantVisitas}(d, \ \text{técnicos}(T), \ T), \ \text{técnicos}(T)) \end{aligned}$$

Axiomas

maxCantVisitas : direccion $\times$ conj(*tecnico*) *ts* $\times$ tad $\longrightarrow$ nat

$\{\neg\emptyset?(\text{ts})\}$

maxCantVisitas(d,ts,T) $\equiv$ **if** $\#(ts) = 1$ **then**
cantVisit(d, dameUno(ts), T)
else
if (cantVisit(d, dameUno(ts), T) $\geq$
maxCantVisitas(d,sinUno(ts),T))
then
cantVisit(d, dameUno(ts), T)
else
maxCantVisitas(d,sinUno(ts),T))
fi
fi

Axiomas

LosDeCantVisitas : direccion $\times$ nat $\times$ conj(*tecnico*) *ts* $\longrightarrow$ nat

$\{\neg\emptyset?(\text{ts})\}$

LosDeCantVisitas(*d*,*n*,*ts*) $\equiv$ **if** $\emptyset?(\text{ts})$ **then**
$\emptyset$
else
if cantVisit(*d*, dameUno(*ts*), *T*) = *n*
then
Ag(dameUno(*ts*),
LosDeCantVisitas(*d*, *n*, sinUno(*ts*)))
else
LosDeCantVisitas(*d*, *n*, sinUno(*ts*))
fi
fi

Complejidad

Complejidad

¿Qué necesito para resolver ejercicios de complejidad?

Consejo

Entender bien las definiciones de $\mathcal{O}$, Ω y Θ

Complejidad

¿$\mathcal{O}$ **significa** peor caso?

Complejidad

¿$\mathcal{O}$ **significa** peor caso?

NO

Complejidad

¿$\mathcal{O}$ **significa** peor caso?

NO

Podríamos usar $\mathcal{O}$ para referirmos al mejor caso.
(Ver item 4 del ejercicio)

Complejidad (1)

Enunciado

Si $f \in O(n)$ y $g(n) = n^2$, entonces $f \circ g \in O(g)$.

Complejidad (1)

Enunciado

Si $f \in O(n)$ y $g(n) = n^2$, entonces $f \circ g \in O(g)$.

Verdadero

- $f(n) \in \mathcal{O}(n)$
- $\exists\ c\ /\ f(n) \leq c.n \qquad \forall n \geq n_0$
- Como $n^2 \geq n$, $f(n^2) \leq c.n^2$
- $g(n) = n^2$, entonces $f(g(n)) \leq c.g(n)$
- $f \circ g \in \mathcal{O}(g)$

Complejidad (2)

Enunciado

Si $f \in O(n)$ y $g(n) = n^2$, entonces $g \circ f \in O(f)$.

Complejidad (2)

Enunciado

Si $f \in O(n)$ y $g(n) = n^2$, entonces $g \circ f \in O(f)$.

Falso

- Tomemos f(n) = n
- Supongamos que g(f(n)) = O(f)
- Entonces, g(n) = O(n)
- Es decir, n^2 = O(n), **ABS**

Complejidad (3)

Enunciado

La complejidad temporal del *peor caso* del Algoritmo es $\Theta(n^2)$.

Muestra para cada número,
cuántos menores consecutivos hay entre el inicio y su posición

```
1:  function MostrarMenoresConsecutivos(arreglo de enteros A)
2:      int i, total;
3:      for i := 0 ... Long(A) - 1 do
4:          j := 0;
5:          while j < i && A[j] < A[i] do
6:              j := j + 1;
7:          end while
8:          imprimir j;
9:      end for
10: end function
```

Complejidad (3)

Enunciado

La complejidad temporal del *peor caso* del Algoritmo es $\Theta(n^2)$.

Complejidad (3)

Enunciado

La complejidad temporal del *peor caso* del Algoritmo es $\Theta(n^2)$.

Verdadero

- En peor caso el algoritmo realiza $(\sum_{i=0}^{n-1} i)$ operaciones.

$$\sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}$$

- $\frac{n(n-1)}{2} = \frac{n^2}{2} - \frac{n}{2} \leq n^2$, es $O(n^2)$

Complejidad (3)

Veamos que también es $\Omega(n^2)$

Queremos ver si existe C > 0, tq para todo $n \geq n_0$ se verifique:

- $\frac{n^2}{2} - \frac{n}{2} \geq C.n^2$
- $\frac{1}{2} - \frac{1}{2n} \geq C$
- Tomando límite en n, la expresión tiende a $\frac{1}{2}$, por lo que existe algun C > 0 que verifica (a partir de cierto n_0).

Como es $O(n^2)$ y $\Omega(n^2)$, es $\Theta(n^2)$

Complejidad (4)

Enunciado

La complejidad temporal del *mejor caso* del Algoritmo es $O(n^2)$.

Muestra para cada número,
cuántos menores consecutivos hay entre el inicio y su posición

```
1:  function MostrarMenoresConsecutivos(arreglo de enteros A)
2:      int i, total;
3:      for i := 0 ... Long(A) - 1 do
4:          j := 0;
5:          while j < i && A[j] < A[i] do
6:              j := j + 1;
7:          end while
8:          imprimir j;
9:      end for
10: end function
```

Complejidad (4)

Enunciado

La complejidad temporal del *mejor caso* del Algoritmo es $O(n^2)$.

Complejidad (4)

Enunciado

La complejidad temporal del *mejor caso* del Algoritmo es $O(n^2)$.

Verdadero

- En el mejor caso no entramos por el ciclo central.
- En este caso la complejidad es $\theta(n)$. En particular es $O(n)$.
- Como $O(n) \subset O(n^2)$, el algoritmo es $O(n^2)$ en mejor caso.

Rep y Abs

Rep

Rep: Invariante de representación

Es una función booleana con dominio en el "mundo funcional" que da *true* cuando recibe una instancia válida.

Consejo

Debemos tener en cuenta:

- Restricciones de los generadores (¿Qué instancias se pueden formar?)
- Decisiones de diseño
- Coherencia de información redundante

Rep

¿Cómo se escribe el Rep formalmente?

- Rep: $\widehat{estr} \rightarrow$ bool
- $(\forall e : \widehat{estr})$ Rep(e) $\equiv$ (1) $\wedge$ (2) $\wedge_{\mathrm{L}}$(3) $\wedge$ ..

1. Predicado 1
2. Predicado 2
3. Predicado 3

Abs

Abs: función de abstracción

Nos dice cómo se relaciona una instancia de mi estructura con una instancia del TAD que estoy diseñando

Consejo

Utilizar los observadores para establecer la relación.

Consejo

Cuidado con los " $\Longleftrightarrow$ "

Enunciado

Enunciado

La organización Redes en Paralelo (o simplemente "ReP") agrupa varias redes sociales en un mismo sitio. El objetivo del sitio es que sus usuarios puedan comunicarse con sus "amistades" de distintas redes sociales, utilizando un único portal.

Para ello, ReP registra a qué redes sociales (dentro de un conjunto predeterminado de redes) pertenece cada uno de sus usuarios y monitorea las relaciones de amistad presentes en estas redes. Dos usuarios serán "amigos" en ReP si y sólo si son amigos en alguna de las redes monitoreadas.

ReP brinda además algunos servicios extra como videollamadas, compartir archivos, etc., pero sólo ofrece estos servicios a aquellas relaciones de amistad que sean suficientemente fuertes. Para ello se define que dos usuarios son *super amigos* si son amigos en 3 o más redes sociales.

Enunciado

observadores básicos

$$\begin{array}{lll} \text{usuarios} & : \text{rep} & \longrightarrow \text{conj(usuario)} \\ \text{redes} & : \text{rep} & \longrightarrow \text{conj(red)} \\ \text{miembro} & : \text{rep } r \times \text{usuario } u \times \text{red } t & \longrightarrow \text{bool} \end{array}$$

$$\{u \in \text{usuarios}(r) \land t \in \text{redes}(r)\}$$

$$\text{amigosEn} : \text{rep } r \times \text{usuario } u \times \text{usuario } u' \times \text{red } t \longrightarrow \text{bool}$$

$$\{\{u, u'\} \subseteq \text{usuarios}(r) \land t \in \text{redes}(r)\}$$

Enunciado

generadores

$$\begin{array}{lll} \text{iniciar} & : \text{conj(red)} & \longrightarrow \text{rep} \\ \text{altaUsuario} & : \text{rep } r \times \text{usuario } u & \longrightarrow \text{rep} \end{array}$$

$$\{u \notin \text{usuarios}(r)\}$$

$$\text{altaEnRed} \quad : \text{rep } r \times \text{usuario } u \times \text{red } t \longrightarrow \text{rep}$$

$$\{u \in \text{usuarios}(r) \land t \in \text{redes}(r)\}$$

$$\text{amistadEnRed} \quad : \text{rep } r \times \text{usuario } u \times \text{usuario } u' \times \text{red } t \longrightarrow \text{rep}$$

$$\left\{ \begin{array}{l} u \neq u' \land \{u, u'\} \subseteq \text{usuarios}(r) \land t \in \text{redes}(r) \land_{\text{L}} \\ \text{miembro}(r, u, t) \land \text{miembro}(r, u', t) \land \neg\text{amigosEn}(r, u, u', t) \end{array} \right\}$$

Enunciado

otras operaciones

superAmigos : rep $r \times$ usuario $u \times$ usuario $u' \longrightarrow$ bool

$\{\{u, u'\} \subseteq \text{usuarios}(r)\}$

Enunciado

Se decidió utilizar la siguiente estructura para representar el TAD:

ReP **se representa con** estr

donde estr es tupla $\langle$ *usuarios*: conj(usuario),
redes: conj(red),
membresías: dicc(usuario,conj(red)),
amigos: dicc(usuario, conj(tupla$\langle$red,usuario$\rangle$))),
superAmigos: dicc(usuario,conj(usuario))$\rangle$)

En esta estructura, *usuarios* y *redes* almacenan los usuarios y las redes registradas en el ReP y *membresías* registra las redes a las que está suscripto cada usuario. Por otro lado para cada usuario *u*, *amigos* guarda las relaciones de amistad entre *u* y otros usuarios en diferentes redes sociales y *superAmigos* registra el conjunto de usuarios que son amigos de *u* en al menos 3 redes diferentes.

Rep

Se pide escribir en castellano y formalmente el invariante de representación.
Posible solución:

1. Las claves de *membresías* son usuarios válidos.

   claves(e.*membresias*) $=$ e.*usuarios*

2. Las membresias de todos los usuarios son redes válidas.

   ($\forall u$: usuario) (def?(u, e.*membresias*) $\Rightarrow_{\mathrm{L}}$
   obtener(u, e.*membresias*) $\subseteq$ e.*redes*)

Rep

3. Las claves de *amigos* son usuarios válidos.

   claves(e.*amigos*) = e.*usuarios*

4. Los datos de las tuplas de amigos están bien formadas (datos en el sistema, y reciprocidad de amistades)

$$
\begin{array}{l}
(\forall u\text{: usuario})\ (\text{def?}(u, \text{e.}amigos) \Rightarrow_{\text{L}} \\
\quad (\forall t\text{: tupla<red,usuario>})(\ \text{t} \in \text{obtener}(u, \text{e.}amigos) \Rightarrow_{\text{L}} \\
\qquad \pi_2(\text{t}) \in \text{e.}usuarios \wedge \\
\qquad \pi_1(\text{t}) \in \text{obtener}(\text{u}, \text{e.}membresias) \wedge \\
\qquad \pi_1(\text{t}) \in \text{obtener}(\pi_2(\text{t}), \text{e.}membresias) \wedge \\
\qquad <\pi_1(\text{t}), \text{u}> \in \text{obtener}(\pi_2(\text{t}), \text{e.}amigos) \\
\quad) \\
)
\end{array}
$$

Rep

5. Las claves de *superamigos* son usuarios válidos.

   claves(e.*superamigos*) $=$ e.*usuarios*

6. Los *superamigos* son usuarios válidos.

   ($\forall u$: usuario) (def?(u, e.*superamigos*) $\Rightarrow_{\mathrm{L}}$
   obtener(u, e.*superamigos*) $\subseteq$ e.*usuarios*)

Rep

7. Los amigos de u con mas de 3 apariciones están en superamigos.

$(\forall u$: usuario) (def?(u, e.*amigos*) $\Rightarrow_{\mathrm{L}}$
    $(\forall u_2$: usuario)
    CantApariciones(u_2, obtener(u, e.*amigos*)) $\geq 3 \Rightarrow_{\mathrm{L}}$
        $u_2 \in$ obtener(u, e.*superamigos*)
)
)

Rep

8. Los superamigos son amigos de u con mas de 3 apariciones

$$
\begin{array}{l}
(\forall u\text{: usuario})\ (\text{def?}(u, \text{e.}\mathit{superamigos}) \Rightarrow_{\text{L}} \\
\quad (\forall u_2\text{: usuario}) \\
\quad u_2 \in \text{obtener}(u, \text{e.}\mathit{superamigos}) \Rightarrow_{\text{L}} \\
\quad\quad \text{CantApariciones}(u_2, \text{obtener}(u, \text{e.}\mathit{amigos})) \geq 3 \\
\quad) \\
)
\end{array}
$$

Abs

Abs

Abs: estr $e \rightarrow$ ReP $\quad \{\ Rep(e)\ \}$

$Abs(e) = r$ /

usuarios(r) = e.usuarios $\wedge$

redes(r) = e.redes $\wedge_{\mathrm{L}}$

$(\forall u$: usuario$)(\forall t$: red$)$ ($u \in$ usuarios$(r) \wedge t \in$ redes$(r) \Rightarrow_{\mathrm{L}}$

(miembro$(r, u, t) \Leftrightarrow t \in$ obtener(u, e.membresías) $\wedge$

$(\forall u'$: usuario$)$

(amigosEn$(r, u, u', t) \Leftrightarrow < r, u' > \in$ obtener(u, e.amigos))

)

)

Fin

:)