

Algoritmos y Estructuras de Datos II

Primer parcial - Miércoles 3 de mayo de 2017

Aclaraciones

Configuración: Agustina

1	2	3	T
P	P	P	P

¡EXCELENTE!

- El parcial es a libro abierto.
- Cada ejercicio debe entregarse en hojas separadas.
- Incluir en cada hoja el número de orden asignado, número de hoja, apellido y nombre.
- Al entregar el parcial, completar el resto de las columnas en la planilla.
- Cada ejercicio se calificará con Promocionado, Aprobado, Regular, o Insuficiente.
- El parcial estará aprobado si: (el ejercicio 1 tiene al menos A) y (el ejercicio 3 tiene al menos una A o (el ejercicio 3 tiene una R y el 2 al menos una A)).

Ej. 1. Especificación

El Centro de Gestión de Trámites (CGT) de la universidad cuenta con una intrincada red de ventanillas en las cuales se realizan distintos tipos de trámites. → x PARÁMETRO!

En cualquier momento pueden llegar personas al CGT, las cuales saben exactamente qué ventanillas tienen que visitar y en qué orden deben hacerlo, con lo cual se dirigen directamente a la primera de las ventanillas que debe visitar. Cuando una persona llega a una ventanilla, saca un número y espera a ser llamada. Una persona no visita más de una vez una misma ventanilla durante su estadía en el CGT.

Los empleados de las ventanillas no se toman tiempo libre, con lo cual cuando terminan de atender a una persona llaman inmediatamente a la que le sigue en la cola de espera. Si no hay nadie esperando, el empleado queda disponible para atender a la siguiente persona que llegue a la ventanilla. Por otro lado, cuando una persona termina de ser atendida en una ventanilla avanza hacia la siguiente ventanilla en su itinerario (a menos que no le queden trámites por hacer, en cuyo caso la persona se retira del establecimiento).

Modelar con un TAD el Centro de Gestión de Trámites descripto teniendo en cuenta que interesa saber:

- La composición actual de la cola en cada ventanilla, es decir, las personas que se encuentran esperando (o siendo atendidas) y en qué orden.
- La ventanilla que atendió a más personas.

Ej. 2. Complejidad → lo mejor para el final!

Discutir la veracidad de las siguientes afirmaciones, justificando adecuadamente en cada caso:

- $\Omega(n) \subseteq O(n^2)$
- $O(n^2) \subseteq \Omega(n)$
- La complejidad temporal del *mejor caso* del Algoritmo 1 es $O(n^2)$.
- La complejidad temporal del *peor caso* del Algoritmo 1 es $\Omega(n)$.

Algoritmo 1 Cuenta cuántos divisores tiene en el arreglo cada número par del mismo

```

1: function DIVISORESDEPARES(arreglo A)
2:   int i, total;           }  $\Theta(1)$  (1º elem)
3:   total := 0;             }  $\Theta(1)$ 
4:   for i := 0 ... Long(A) - 1 do }  $\Theta(n) = O(1)$ 
5:     if 2 divide a A[i] then
6:       for j := 0 ... Long(A) - 1 do }  $\Theta(n)$ 
7:         if A[j] divide a A[i] then }  $\Theta(n)$ 
8:           total := total + 1; }  $\Theta(n)$ 
9:         end if }  $\Theta(n)$ 
10:      end for }  $\Theta(n)$ 
11:    end if }  $\Theta(n)$ 
12:  end for }  $\Theta(n)$ 
13:  return total;
14: end function
  
```

$\Theta(n) = O(n)$

Observación: Consideramos que las verificaciones de divisibilidad en el algoritmo son operaciones elementales.

Ej. 3. Diseño

El sitio REMATAZOS permite a sus usuarios publicar artículos para rematarlos al mejor postor. Mientras una publicación esté activa, los usuarios pueden ofertar un monto por el artículo (siempre mayor que el monto de la máxima oferta hasta el momento). Cada publicación cuenta con un *precio de venta automática* definido por el dueño de la misma (y oculto a la comunidad) y la duración de cada publicación es indefinida. Si en algún momento, una oferta alcanza o supera el precio de venta automática, el remate finaliza automáticamente concretándose así la venta. Por otro lado, el dueño del artículo puede decidir finalizar el remate en cualquier momento y en caso de haber alguna oferta, la venta se concreta al mejor postor. En cualquier caso, no se guardan registros de las publicaciones finalizadas. El siguiente TAD modela el sitio de remates descripto (aunque se omiten las axiomatizaciones).

TADs USUARIO y PUB son NAT

TAD OFERTA es TUPLA(USUARIO,NAT)

TAD REMATAZOS

observadores básicos

usuarios	:	rematazos	→ conj(usuario)
publicaciones	:	rematazos $r \times$ usuario u	→ conj(pub)
precioVentaAutom	:	rematazos $r \times$ pub p	→ nat
ofertas	:	rematazos $r \times$ pub p	→ secu(oferta)

$\{u \in \text{usuarios}(r)\}$
 $\{\text{publicacionActiva}(r,p)\}$
 $\{\text{publicacionActiva}(r,p)\}$

generadores

iniciar	:		→ rematazos
altaUsuario	:	rematazos $r \times$ usuario u	→ rematazos
publicar	:	rematazos $r \times$ usuario $u \times$ pub $p \times$ nat pr	→ rematazos
			$\{u \in \text{usuarios}(r) \wedge \neg \text{publicacionActiva}(r,p)\}$
ofertar	:	rematazos $r \times$ pub $p \times$ oferta of $\{\Pi_1(of) \in \text{usuarios}(r) \setminus \{\text{dueño}(r,p)\} \wedge (\text{vacía?}(ofertas(r,p)) \vee_L \Pi_2(of) > \Pi_2(\text{ultimo}(ofertas(r,p))))\}$	→ rematazos
finalizar	:	rematazos $r \times$ pub p	→ rematazos $\{\text{publicacionActiva}(r,p)\}$

otras operaciones

dueño	:	rematazos $r \times$ pub p	→ usuario
-------	---	------------------------------	-----------

$\{\text{publicacionActiva}(r,p)\}$

predicados

$$\text{publicacionActiva}(r,p) \equiv (\exists u : \text{usuario}) (u \in \text{usuarios}(r) \wedge_L p \in \text{publicaciones}(r,u))$$

Fin TAD

Se decidió utilizar la siguiente estructura para representar el TAD:

REMATAZOS se representa con estr

donde estr es tupla (*usuarios*: conj(usuario),
publicaciones: dicc(usuario,conj(tupla(pub,nat))),
dueños: dicc(pub, usuario),
ofertas: dicc(pub,secu(oferta)))

En esta estructura, *usuarios* almacena los usuarios del sitio, *publicaciones* registra las publicaciones activas de cada usuario (junto con sus precios de venta automática) y *dueños* indica qué usuario publicó cada artículo. Por otro lado, *ofertas* almacena todas las ofertas hechas (en orden) para cada publicación.

Teniendo en cuenta el TAD REMATAZOS y la estructura elegida para su representación se pide:

- Escribir en castellano el invariante de representación.
- Escribir formalmente el invariante de representación.
- Escribir formalmente la función de abstracción.

1 2 3
 P | P D O MARCU

Algoritmos y Estructuras de Datos II N° de orden 14
 1er parcial - Ejercicio 1 Hoja 1

Aclaración: persona podría modelarse con muchos TADs diferentes. El fin de evitar la sobreespecificación y dado que las operaciones que se hacen con persona dentro de cgt son válidas para prácticamente cualquier tipo, esa decisión se va a dejar para la etapa de diseño.

TAD CGT

géneros
uso
exporta

cgt
Bool, NAT, COLA(α), PERSONA, CONJ(α), SECU(nat)
cgt, igualdad observacional, observadores, generadores, atendióMás

igualdad observacional

$$(\forall c, c' : \text{cgt}) \quad c =_{\text{obs}} c' \Leftrightarrow \left(\begin{array}{l} \# \text{ventanillas}(c) =_{\text{obs}} \# \text{ventanillas}(c') \wedge \\ (\forall n : \text{nat}) \quad 1 \leq n \leq \# \text{ventanillas}(c) \Rightarrow \\ \text{libre}(c, n) =_{\text{obs}} \text{libre}(c', n) \wedge \\ \# \text{atendidos}(c, n) =_{\text{obs}} \# \text{atendidos}(c', n) \wedge \\ \text{ocupada?}(c, n) =_{\text{obs}} \text{ocupada?}(c', n) \wedge \\ (\text{ocupada?}(c, n) \Rightarrow \text{cliente}(c, n) =_{\text{obs}} \text{cliente}(c', n)) \\ \wedge \left(\begin{array}{l} (\forall p : \text{persona}) \quad p \in \text{todasLasPersonas}(c) \Rightarrow \\ \text{ventanillasPend}(c, p) =_{\text{obs}} \text{ventanillasPend}(c', p) \end{array} \right) \end{array} \right)$$

observadores básicos

$$\begin{array}{ll} \# \text{ventanillas} : \text{cgt} \rightarrow \text{nat} & \{ 1 \leq \text{ventanilla} \leq \# \text{ventanillas}(c) \} \\ \text{libre} : \text{cgt } c \times \text{nat ventanilla} \rightarrow \text{cola(persona)} & \{ 1 \leq \text{ventanilla} \leq \# \text{ventanillas}(c) \} \\ \# \text{atendidos} : \text{cgt } c \times \text{nat ventanilla} \rightarrow \text{nat} & \{ 1 \leq \text{ventanilla} \leq \# \text{ventanillas}(c) \} \\ \text{ocupada?} : \text{cgt } c \times \text{nat ventanilla} \rightarrow \text{bool} & \{ 1 \leq \text{ventanilla} \leq \# \text{ventanillas}(c) \} \\ \text{cliente} : \text{cgt } c \times \text{nat ventanilla} \rightarrow \text{persona} & \{ 1 \leq \text{ventanilla} \leq \# \text{ventanillas}(c) \wedge \text{ocupada?}(c, \text{ventanilla}) \} \\ \text{ventanillasPend} : \text{cgt } c \times \text{persona } p \rightarrow \text{secu(nat)} & \{ p \in \text{todasLasPersonas}(c) \} \end{array}$$

generadores

$$\begin{array}{ll} \text{abrirCGT} : \text{nat contVentanillas} \rightarrow \text{cgt} & \{ 1 \leq \text{contVentanillas} \} \quad (*) \\ \text{llegarPersona} : \text{cgt } c \times \text{persona } p \times \text{secu(nat) ventanillas} \rightarrow \text{cgt} & \\ \quad \{ p \notin \text{todasLasPersonas}(c) \wedge \neg \text{vacío?}(\text{ventanillas}) \wedge \neg \text{sinRepetidos}(\text{ventanillas}) \} & \\ \quad \{ (\forall n : \text{nat}) \text{ está?}(n, \text{ventanillas}) \Rightarrow 1 \leq n \leq \# \text{ventanillas}(c) \} & \\ \text{atender} : \text{cgt } c \times \text{nat ventanilla} \rightarrow \text{cgt} & \\ \quad \{ 1 \leq \text{ventanilla} \leq \# \text{ventanillas}(c) \wedge \neg \text{ocupada?}(c, \text{ventanilla}) \} & \end{array}$$

otras operaciones

$$\begin{array}{ll} \text{atendióMás} : \text{cgt} \rightarrow \text{nat} & \{ \text{PARA } \forall n \text{ TIENEA SENTIDO UNIQUÍTICO} \} \\ \text{atendióMásAux} : \text{cgt } c \times \text{nat ventsAMirar} \rightarrow \text{nat} & \{ 1 \leq \text{ventsAMirar} \leq \# \text{ventanillas}(c) \} \\ \text{todasLasPersonas} : \text{cgt } c \rightarrow \text{conj(persona)} & \\ \text{todasLasPersonasAux} : \text{cgt } c \times \text{nat ventsAMirar} \rightarrow \text{conj(persona)} & \\ \quad \{ 1 \leq \text{ventsAMirar} \leq \# \text{ventanillas}(c) \} & \end{array}$$

Parar que sea únicamente 1 persona

Continúa en la carilla siguiente

Axiomas

$$\#ventanillas(\text{nuevo}(CET(n)), v) \equiv n$$

$$\#ventanillas(\text{llegarPersona}(c, p, vents), v) \equiv \#ventanillas(c)$$

$$\#ventanillas(\text{atender}(c, vent), v) \equiv \#ventanillas(c)$$

$$filz(\text{nuevo}(CET(n)), v) \equiv \text{vacío}$$

$$filz(\text{llegarPersona}(c, p, vents), v) \equiv \begin{array}{l} \text{if } v = \text{prim}(vents) \text{ then} \\ \quad \text{if } \neg \text{ocupada?}(c, v) \text{ then} \\ \quad \quad \text{vacío} \\ \quad \text{else} \\ \quad \quad \text{encolar}(p, filz(c, v)) \end{array}$$

fi

else

$$filz(c, v)$$

fi

$$filz(\text{atender}(c, v), v) \equiv$$

$$\begin{array}{l} \text{if } \neg \text{vacío?}(\text{ventanillasPend}(\text{cliente}(c, v))) \wedge v = \text{prim}(\text{ventanillasPend}(\text{cliente}(c, v))) \text{ then} \\ \quad \text{encolar}(\text{cliente}(c, v), filz(c, v)) \end{array}$$

else

$$filz(c, v) \text{ Esta función tiene un error. Ver corrección en (*)}$$

fi

$$\#\text{atendidos}(\text{nuevo}(CET(n)), v) \equiv 0$$

$$\#\text{atendidos}(\text{llegarPersona}(c, p, vents), v) \equiv \#\text{atendidos}(c, v)$$

$$\#\text{atendidos}(\text{atender}(c, v), v) \equiv (\text{if } v = \tilde{v} \text{ then } 1 \text{ else } 0) + \#\text{atendidos}(c, v).$$

$$\text{ocupada?}(\text{nuevo}(CET(n)), v) \equiv \text{false}$$

$$\text{ocupada?}(\text{llegarPersona}(c, p, vents), v) \equiv$$

$$\begin{array}{l} \text{if } v = \text{prim}(vents) \text{ then true else } \text{ocupada?}(c, v) \end{array}$$

// Si v es la primera ventanilla que tiene que ir la nueva persona, seguro que ve a estar ocupada, ya sea porque lo estaba antes o porque estabas libre e inmediatamente se puso a atender al cliente. Si v es cualquier otra ventanilla, la llegada del nuevo cliente no la afecta en nada. (1)

$$\text{ocupada?}(\text{atender}(c, v), v) \equiv$$

$$\begin{array}{l} \text{if } v = \tilde{v} \text{ then} \\ \quad \text{if } \text{vacío?}(filz(c, v)) \text{ then} \end{array}$$

// si no hay a quién atender, queda libre

false

else

// en caso contrario atiende al próximo

true

fi

else

$$\begin{array}{l} \text{if } \neg \text{vacío?}(\text{ventanillasPend}(\text{cliente}(c, v))) \wedge v = \text{prim}(\text{ventanillasPend}(\text{cliente}(c, v))) \text{ then} \\ \quad \text{true} \end{array}$$

// reasignamiento en el logo al (1)

else

$$\text{ocupada?}(c, v)$$

fi

fi

Continúa en la hoja siguiente

Algoritmos y Estructuras de Datos II
1er parcial - Ejercicio 1

Nº de orden 14

Floja 2

cliente(llegarPersonal(c, p, rents), v) ≡

if \neg ocupada?(c, v) \wedge v = prim(rents) then p else cliente(c, v) fi

cliente(atender(c, v), v) ≡

if v = \tilde{v} then proximo(fila(v)) else cliente(c, v) fi

// si la fila estuviera vacía, v pasaría a estar desocupada y no estaría llamando a esta función debido a sus restricciones, por lo que puedo usar proximo(fila(c, v))

ventanillasPend(llegarPersonal(c, \tilde{p} , rents), p) ≡ if $p = \tilde{p}$ then

if \neg ocupada?(c, prim(rents)) then
fin(rents)

else

rents

fi

else
ventanillasPend(c, p)

fi

ventanillasPend(atender(c, v), p) ≡ if $p = \text{cliente}(c, v)$ then

fin(ventanillasPend(c, p))

else

ventanillasPend(c, p)

fi

atendióMás(c) ≡ atendióMásAux(c, #ventanillas(c))

atendióMásAux(c, vs) ≡ if vs = 1 then

1

else

if #atendidos(c, vs) > #atendidos(c, atendióMásAux(c, vs-1)) then

vs

else

atendióMásAux(c, vs-1)

fi

fi

todasLasPersonas(c) ≡ todasLasPersonasAux(c, #ventanillas(c))

todasLasPersonasAux(c, vs) ≡

((if ocupada?(c, vs) then {cliente(c, vs)} else \emptyset) \cup colo4Conj(fila(c, vs)))

\cup (if vs = 1 then \emptyset else todasLasPersonasAux(c, vs-1) fi)

Fin TAD

Continúa en la carilla siguiente

Funciones auxiliares de otros TADS

sinRepetidos: set α \rightarrow bool

```

sinRepetidos(s) ≡ if vacío? (s) then
    true
  else
    ~esté? (prim(s), fin(s)) ∧ sinRepetidos (fin(s))
fi

```

coleAlonj: colz α \rightarrow conj α

```

coleAConj(c) ≡ if vacío? (c) then
    φ
  else
    Ag (próximo(c), coleAConj (desencolar(c)))
fi

```

(*) No permite la existencia de CGTs con 0 ventanillas porque agregarían costos borde engorrosos de manejar y no aportan nada al modelado que se pide en la consigna.

(*) Función corregida:

```

filz (atender(c, v), v) ≡
if v = v then
  if ~vacío? (filz(c, v)) then
    desencolar (filz(c, v))
  else
    vacío
  fi
else
  if ~vacío? (ventanillasPend(c, cliente(c, v))) ∧ prim(ventanillasPend(c, cliente(c, v))) = v then
    encolar (cliente(c, v); filz(c, v))
  else
    filz(c, v)
  fi
fi

```

↑
FALTA IF ATENDER? (c, v)

1. Falso. Contraejemplo:

Sea $f(n) = n^3$. Veamos que $f \in \Omega(n)$:

$$\Omega(n) = \{g: \mathbb{N} \rightarrow \mathbb{N}, (\exists c, n_0) / (\forall n \geq n_0): g(n) \geq cn\}$$

Sean $c=1, n_0=1$. $n^3 \geq n \Leftrightarrow n^2(n-1) \geq 0$, válido $\forall n \geq 1 \Rightarrow$
 $\Rightarrow f \in \Omega(n)$.

Aun embargo:

Supongamos que $f \in O(n^2)$.

$$O(n^2) = \{g: \mathbb{N} \rightarrow \mathbb{N}, (\exists c, n_0) / (\forall n \geq n_0): g(n) \leq cn^2\}$$

Entonces $(\exists c, n_0) / (\forall n \geq n_0): n^3 \leq cn^2$

$$n^3 \leq cn^2 \Leftrightarrow n^3 - cn^2 \leq 0 \Leftrightarrow n^2(n-c) \leq 0 \Leftrightarrow n-c \leq 0 \Leftrightarrow n \leq c$$

Pero entonces, para cualquier c que elijamos, vamos a estar acotando n por una constante; esto va a ser imposible que la condición se cumpla con cualquier $n \geq n_0$, pues debería valer para n arbitrariamente grande.

$$\therefore (\exists c, n_0) / (\forall n \geq n_0): n^3 \leq cn^2 \Leftrightarrow n^3 \notin O(n^2)$$

$$\therefore \exists f \in \Omega(n) / f \notin O(n^2) \Rightarrow \Omega(n) \notin O(n^2).$$

2. Falso. Contraejemplo:

Sea $f(n) = 1$. Veamos que $f \in O(n^2)$:

$$\text{Sean } c=1, n_0=1. (\forall n \geq 1): n^2 \geq 1 \Rightarrow f \in O(n^2).$$

Supongamos que $f \in \Omega(n)$.

Entonces $(\exists c, n_0) / (\forall n \geq n_0): 1 \geq cn$.

$$1 \geq cn \Leftrightarrow n \leq \frac{1}{c}$$

Nuevamente estamos imponiendo una cota superior para n que va a hacer imposible que la condición valga para cualquier n mayor o igual a n_0 , sin importar nuestra elección de c y n_0 .

$$\therefore (\exists c, n_0) / (\forall n \geq n_0): 1 \geq cn \Leftrightarrow 1 \notin \Omega(n)$$

$$\therefore \exists f \in O(n^2) / f \notin \Omega(n) \Rightarrow O(n^2) \notin \Omega(n).$$

Observemos primero que todas las operaciones, tanto las de las guardas como las que están adentro de la función y de cada ciclo, son elementales ($O(1)$). Asignaciones, declaraciones, acceso a arreglos, sumas de enteros y verificaciones de divisibilidad. Entonces, la complejidad temporal se puede calcular viendo cuántas veces se ejecuta cada ciclo.

3. Verdadero. Justificación:

Consideremos un arreglo que sólo tiene elementos impares (el mejor caso).

El ciclo que empieza en la línea 4 se ejecuta si o sí n veces, donde n es la longitud del arreglo.

El bloque `if` de la línea 5 no se ejecuta en ninguna iteración del ciclo.

Por lo tanto, sólo hicimos n operaciones elementales (más las constantes de antes del ciclo). La complejidad es

$$O(1) + O(1) + n \cdot O(1) = n \in O(n^2).$$

(Ya demostramos en clase que $n \in O(n^2)$).

∴ El mejor caso de este algoritmo es $O(n^2)$.

4. Verdadero. Justificación:

Consideremos un arreglo que sólo tiene elementos pares (el peor caso).

El ciclo de la línea 4 también se ejecuta n veces, pero esta vez el bloque `if` se ejecuta en todas las iteraciones.

El ciclo dentro de este bloque se ejecuta n veces independientemente de los valores del arreglo (siempre q cuando se haya entrado al bloque, por supuesto) y la instrucción de adentro es $O(1)$. Entonces la complejidad es

$$O(1) + O(1) + n \cdot (n \cdot (O(1) \cdot O(1))) = n \cdot O(n)$$
$$= n^2 \in \Omega(n).$$

(Con $c=0$, $n_0=1$ y una demostración análoga a la del punto 1 se ve que $n^2 \in \Omega(n)$).

∴ El peor caso del algoritmo es $\Omega(n)$.

- a) 1. Los usuarios de e.usuarios son los mismos que los de e.publicaciones
 2. Los usuarios de e.usuarios incluyen a los de e.dueños
 3. Las publicaciones de e.publicaciones son las mismas que las de e.dueños
 4. Las publicaciones son las mismas que las de e.ofertas
 5. Para cada usuario v , las publicaciones de v lo tienen como significado en e.dueños
 6. Para cada publicación p , p está entre las publicaciones que le corresponden a su dueño en e.publicaciones
 7. Ninguna publicación tiene una oferta mayor o igual a su precio de venta automática (de lo contrario ya no estaría activa)
 8. Ningún usuario ofertó en una publicación propia Faltan predicados, ver (*)
~~Falta: PUES NO SE REPITE.~~
 - b) Rep: estr → bool
- Rep(e) ≡ true \Leftrightarrow
1. $e.usuarios = \text{claves}(e.publicaciones) \wedge$
 2. $\text{significados}(e.dueños) \subseteq e.usuarios \wedge$
 3. $\text{claves}(e.dueños) = \text{damePrimeros}(\text{UnirConjuntos}(\text{significados}(e.publicaciones)))$
 4. $\text{claves}(e.ofertas) = \text{claves}(e.dueños) \wedge (*)$
 5. $(\forall v: \text{usuario}) \text{ def?}(v, e.publicaciones) \Rightarrow ((\forall p: \text{pub})$
 $p \in \text{damePrimeros}(\text{obtener}(v, e.publicaciones)) \Rightarrow \text{obtener}(p, e.dueños) = v) \wedge$
 6. $(\forall p: \text{pub}) \text{ def?}(p, e.dueños) \Rightarrow$
 $(p \in \text{damePrimeros}(\text{obtener}(\text{obtener}(p, e.dueños), e.publicaciones))) \wedge$
 7. $(\forall p: \text{pub}) \text{ def?}(p, e.ofertas) \Rightarrow ((\forall o: \text{oferta}) \text{ esté?}(o, \text{obtener}(p, e.ofertas)) \Rightarrow$
 $\Pi_2(o) < \text{buscarPrecioPub}(p, \text{UnirConjuntos}(\text{significados}(e.publicaciones)))) \wedge$
 8. $(\forall v: \text{usuario}) \text{ def?}(v, e.usuarios) \Rightarrow ((\forall p: \text{pub}) p \in \text{damePrimeros}(\text{obtener}(v, e.publicaciones)) \Rightarrow$
 $((\forall o: \text{oferta}) \text{ esté?}(o, \text{obtener}(p, e.ofertas)) \Rightarrow \Pi_1(o) \neq v)) \wedge \text{ver}(\star)$

(*) Observar que esto implica la igualdad entre las publicaciones de e.ofertas y las de e.publicaciones por el punto anterior.

Continúa en la carilla siguiente

Funciones auxiliares

significados: $\text{dicc}(\kappa, \sigma) \rightarrow \text{conj}(\sigma)$

$\text{significados}(d) \equiv \text{if } d = \text{vacío} \text{ then}$

\emptyset
else

Ag(obtener(dameUno(claves(d)), d), significados(borrar(dameUno(d), d)))

fi

UnirConjuntos: $\text{conj}(\text{conj}(\alpha)) \rightarrow \text{conj}(\alpha)$

$\text{unirConjuntos}(c) \equiv \text{if } \emptyset ? (c) \text{ then}$

\emptyset
else

dameUno(c) \cup unirConjuntos(sinUno(c))

fi

damePrimeros: $\text{conj}(\text{tuple}(\alpha, \beta)) \rightarrow \text{conj}(\beta)$

$\text{damePrimeros}(c) \equiv \text{if } \emptyset ? (c) \text{ then}$

\emptyset
else

Ag($\Pi_1(\text{dameUno}(c)), \text{damePrimeros}(\text{sinUno}(c))$)

fi

buscarPrecioPub: $\text{pub } p \times \text{conj}(\text{tuple}(\text{pub}, \text{nat})) c \rightarrow \text{nat} \quad \{p \in \text{damePrimeros}(c)\}$

$\text{buscarPrecioPub}(p, c) \equiv \text{if } \#(c) = 1 \text{ then}$

$\Pi_2(\text{dameUno}(c))$

else

if $\Pi_1(\text{dameUno}(c)) = p$ then

$\Pi_2(\text{dameUno}(c))$

else

buscarPrecioPub(p, sinUno(c))

fi

fi

c) Abs: estr e \rightarrow rematzos

$\{\text{Rep}(e)\}$

$(\forall e: \text{estr}) \text{ Abs}(e) =_{\text{obs}} r: \text{rematzos} \}$

$\text{usuarios}(r) =_{\text{obs}} e \cdot \text{usuarios} \wedge_L$

$((\forall u: \text{usuario}) \vee e \in \text{usuarios}(r) \Rightarrow (\text{publicaciones}(r, u) =_{\text{obs}} \text{damePrimeros}(\text{obtener}(u, e, \text{publicaciones})))$

$\wedge_L ((\forall p: \text{pub}) \text{ } p \in \text{publicaciones}(r, u) \Rightarrow \text{precioVentaAutom}(r, p) =_{\text{obs}}$
 $\text{buscarPrecioPub}(p, \text{obtener}(u, e, \text{publicaciones}))$

$\wedge \text{ofertes}(r, p) =_{\text{obs}} \text{obtener}(p, e, \text{ofertes})) \big) \big)$

Continúa en la página siguiente

(*) Faltó agregar los siguientes predicados:

a) 9. Cada oferta es mayor a la anterior

b) 9. $(\forall p : pub) \text{ def?}(p, e.\text{ofertas}) \Rightarrow \text{ordenad2PorSegunda}(\text{obtener}(p, e.\text{ofertas}))$

$\text{ordenad2PorSegunda} : \text{secu}(\text{tupla}\langle \text{usuario}, \text{nat} \rangle) \rightarrow \text{bool}$

$\text{ordenad2PorSegunda}(s) \equiv \text{if } \text{long}(s) \leq 1 \text{ then}$

 true

 else

$\Pi_2(\text{prim}(s)) < \Pi_2(\text{prim}(\text{fin}(s)))$

$\wedge \text{ordenad2PorSegunda}(\text{fin}(s))$

 fi

