Recursión Ultimate

Matías Barbeito (Cani)

Algoritmos y Estructuras de Datos II,
Departamento de Computación,
Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires

1 de septiembre de 2017

Ejercicio 1

TAD Diccionario(κ, σ)

géneros dicc(κ, σ)

observadores básicos

def?	:	$\kappa \times$ dicc(κ, σ)	$\longrightarrow$	bool	
obtener	:	κ c $\times$ dicc(κ, σ) d	$\longrightarrow$	σ	{def?(c, d)}

generadores

vacío	:		$\longrightarrow$	dicc(κ, σ)	
definir	:	$\kappa \times \sigma \times$ dicc(κ, σ)	$\longrightarrow$	dicc(κ, σ)	

otras operaciones

borrar	:	κ c $\times$ dicc(κ, σ) d	$\longrightarrow$	dicc(κ, σ)	{def?(c,d)}
claves	:	dicc(κ, σ)	$\longrightarrow$	conj(κ)	

axiomas $\forall d$: dicc(κ, σ), $\forall c, k$: κ, $\forall s$: σ

def?(c,vacío)	$\equiv$	false
def?(c, definir(k, s, d)	$\equiv$	$c = k \lor$ def?(c, d)
obtener(c, definir(k, s, d))	$\equiv$	**if** $c = k$ **then** s **else** obtener(c, d) **fi**

Fin TAD

Ejercicio 1

Dados dos diccionarios donde el significado de uno es la clave del otro, crear un nuevo diccionario que tenga las claves del primero y los significados del segundo, estando estos relacionados por el significado/clave en común.

$$\text{juntar} : \text{dicc}(k \times s_1) d_1 \times \text{dicc}(s_1 \times s_2) \longrightarrow \text{dicc}(k, s_2)$$

$$\{(\forall k\text{: } k \ \text{def?}(\text{k}, d_1) \Rightarrow_{\text{L}} \text{def?}(\text{obtener}(\text{k}, d_1), d_2)\}$$

TAD Diccionario

observadores básicos

def? : $\kappa \times \text{dicc}(\kappa, \sigma) \longrightarrow \text{bool}$

obtener : $\kappa\ c \times \text{dicc}(\kappa, \sigma)\ d \longrightarrow \sigma$ $\{\text{def?}(c, d)\}$

generadores

vacío : $\longrightarrow \text{dicc}(\kappa, \sigma)$

definir : $\kappa \times \sigma \times \text{dicc}(\kappa, \sigma) \longrightarrow \text{dicc}(\kappa, \sigma)$

otras operaciones

borrar : $\kappa\ c \times \text{dicc}(\kappa, \sigma)\ d \longrightarrow \text{dicc}(\kappa, \sigma)$ $\{\text{def?}(c,d)\}$

claves : $\text{dicc}(\kappa, \sigma) \longrightarrow \text{conj}(\kappa)$

Ejercicio 1

$$\text{juntar} \; : \; \text{dicc}(k \times s_1)d_1 \times \text{dicc}(s_1 \times s_2) \;\longrightarrow\; \text{dicc}(k,s_2)$$

$$\{(\forall k\colon k \; \text{def?}(\text{k}, d_1) \Rightarrow_{\text{L}} \text{def?}(\text{obtener}(\text{k}, d_1), d_2)\}$$

$$
\begin{array}{lll}
\text{juntar}(d_1, d_2) & \equiv & \textbf{if } \emptyset?(\text{claves}(d_1)) \;\; \textbf{then} \\
 & & \quad \text{vacío} \\
 & & \textbf{else} \\
 & & \quad \text{definir}(\\
 & & \quad \text{dameUno}(\text{claves}(d_1)), \\
 & & \quad \text{obtener}(\text{obtener}(\text{dameUno}(\text{claves}(d_1)), \quad d_1), \quad d_2), \\
 & & \quad \text{juntar}(\text{borrar}(\text{dameUno}(\text{claves}(d_1)), d_1), d_2) \\
 & & \textbf{fi}
\end{array}
$$

Ejercicio 2

Dado el TAD Caralibro que representa una red social, se nos pide axiomatizar la función *islaDeAmigos(u, c)* que dado un miembro de la red social nos devuelva el conjunto de usuarios que son alcanzables con la relación de amistad.

TAD Caralibro

géneros cl

observadores básicos

miembros : cl $\longrightarrow$ conj(usuario)

amigos : usuario u $\times$ cl c $\longrightarrow$ conj(usuario)

$\{$u $\in$ miembros(cl)$\}$

otras operaciones

islaDeAmigos : usuario u $\times$ cl c $\longrightarrow$ conj(usuario)

$\{$u $\in$ miembros(c)$\}$

Fin TAD

Ejercicio 2

$\text{islaDeAmigos} \;:\; \text{usuario } u \times \text{cl } c \;\longrightarrow\; \text{conj(usuario)} \qquad \{u \in \text{miembros}(c)\}$

$\text{amigosDeAmigos} \;:\; \text{conj(usuario) } us \times \text{cl } c \;\longrightarrow\; \text{conj(usuarios)}$

$\{us \subseteq \text{miembros}(c)\}$

$$
\begin{aligned}
\text{amigosDeAmigos}(us, c) \;\equiv\; & \textbf{if } \emptyset?(us) \textbf{ then} \\
& \quad \emptyset \\
& \textbf{else} \\
& \quad \text{amigos}(\text{dameUno}(us), c) \cup \{\text{dameUno}(us)\} \cup \\
& \quad \text{amigosDeAmigos}(\text{sinUno}(us), c) \\
& \textbf{fi}
\end{aligned}
$$

Ejercicio 2

islaDeAmigos : usuario u × cl c $\longrightarrow$ conj(usuario) $\{u \in \text{miembros}(c)\}$

amigosDeAmigos : conj(usuario) us × cl c $\longrightarrow$ conj(usuarios)

$\{us \subseteq \text{miembros}(c)\}$

islaDeVariosAmigos : conj(usuario) us × cl c $\longrightarrow$ conj(usuario)

$\{us \subseteq \text{miembros}(c)\}$

islaDeVariosAmigos(us, c) $\equiv$ **if** ?? **then**
us
else
islaDeVariosAmigos(amigosDeAmigos(us, c), c)
fi

islaDeAmigos(u, c) $\equiv$ islaDeVariosAmigos($\{$u$\}$, c)

Ejercicio 2

$\text{islaDeAmigos} \ : \ \text{usuario}\ u \times \text{cl}\ c \longrightarrow \text{conj(usuario)} \qquad \{\text{u} \in \text{miembros(c)}\}$

$\text{amigosDeAmigos} \ : \ \text{conj(usuario)}\ us \times \text{cl}\ c \longrightarrow \text{conj(usuarios)}$

$\{\text{us} \subseteq \text{miembros(c)}\}$

$\text{islaDeVariosAmigos} \ : \ \text{conj(usuario)}\ us \times \text{cl}\ c \longrightarrow \text{conj(usuario)}$

$\{\text{us} \subseteq \text{miembros(c)}\}$

$$
\begin{array}{lll}
\text{islaDeVariosAmigos(us, c)} & \equiv & \textbf{if}\ \text{amigosDeAmigos(us, c)} \subseteq \text{us}\ \ \textbf{then} \\
 & & \quad \text{us} \\
 & & \textbf{else} \\
 & & \quad \text{islaDeVariosAmigos(amigosDeAmigos(us, c),} \\
 & & \quad \text{c)} \\
 & & \textbf{fi}
\end{array}
$$

Ejercicio 2

islaDeAmigos : usuario u $\times$ cl c $\longrightarrow$ conj(usuario) {u $\in$ miembros(c)}

amigosDeAmigos : conj(usuario) us $\times$ cl c $\longrightarrow$ conj(usuarios)

{us $\subseteq$ miembros(c)}

islaDeVariosAmigos : conj(usuario) us $\times$ cl c $\longrightarrow$ conj(usuario)

{us $\subseteq$ miembros(c)}

islaDeAmigos(u, c) $\equiv$ islaDeVariosAmigos({u}, c)

Ejercicio 3

Ahora que sabemos cómo se componen las islas de amigos, nos piden especificar todas las cadenas de menor cantidad de saltos entre dos usuarios en una misma isla. Esto es, aquellas cadenas de amistad entre dos usuarios con largo equivalente al grado de separación entre ellos.

Ejercicio 3

$$\text{gradosDeSeparación} : \text{usuario } a \times \text{usuario } b \times \text{cl } c \longrightarrow \text{conj(secu(usuario))}$$
$$\text{gradosDeSeparación(a,b,c)} \equiv \text{minLargo(seguirCadenaA(a} \bullet <>, \text{ b, c))}$$

$$\text{seguirCadenaA} : \text{secu(usuario) } s \times \text{usuario } b \times \text{cl } c \longrightarrow \text{conj(secu(usuario))}$$

$$\text{cadenasExtendidas} : \text{secu(usuario) } s \times \text{conj(usuario) } us \times \text{usuario } b \times \text{cl } c \longrightarrow \text{conj(secu(usuario))}$$

Ejercicio 3

$$\text{seguirCadenaA} \; : \; \text{secu(usuario)}\; s \times \text{usuario}\; b \times \text{cl}\; c \longrightarrow \text{conj(secu(usuario))}$$

$$\begin{array}{l}
\text{seguirCadenaA(s, b, c)} \equiv \textbf{if } \text{fin(s)} = \text{b} \;\; \textbf{then} \\
\qquad \{\text{s}\} \\
\textbf{else} \\
\qquad \text{cadenasExtendidas(s, amigos(fin(s), c), b, c)} \\
\textbf{fi}
\end{array}$$

$$\text{cadenasExtendidas} \; : \; \text{secu(usuario)}\; s \times \text{conj(usuario)}\; us \times \text{usuario}\; b \times \text{cl}\; c \longrightarrow \text{conj(secu(usuario))}$$

$$\begin{array}{l}
\text{cadenasExtendidas(s,us,b,c)} \equiv \textbf{if } \emptyset?(\text{us}) \;\; \textbf{then} \\
\qquad \emptyset \\
\textbf{else} \\
\qquad \textbf{if } \neg \text{ está?(dameUno(us), s)} \;\; \textbf{then} \\
\qquad\qquad \text{seguirCadenaA(s} \circ \text{dameUno(us), b, c)} \\
\qquad\qquad \cup \text{ cadenasExtendidas(s, sinUno(us), b, c)} \\
\qquad \textbf{else} \\
\qquad\qquad \text{cadenasExtendidas(s, sinUno(us), b, c)} \\
\qquad \textbf{fi} \\
\textbf{fi}
\end{array}$$