Algoritmos y Estructuras de Datos II

Primer Cuatrimestre de 2015

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

Trabajo Práctico 1

Especificación DCNet

Grupo 24

Integrante	LU	Correo electrónico
Fernando Frassia	340/13	ferfrassia@gmail.com
Rodrigo Seoane Quilne	910/11	seoane.raq@gmail.com
Sebastian Matias Giambastiani	916/12	sebastian.giambastiani@hotmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. Aclaraciones

• TDC son todos los caminos completos posibles de una computadora a otra.

• CAMINOPARCIAL genera una secuencia de tuplas (c1, i1, i2, c2). En el último elemento de la secuencia, se encuentra la información de dónde está el paquete actualmente (c1), cuál es la próxima computadora a la que irá (c2), y cuales son las interfaces que las conectan (i1 e i2 respectivamente). Notar que caminoParcial no distingue entre un paquete que llegó a su destino y un paquete que está en la última computadora antes de llegar a su destino. Esto es por un tema de tipos (dado que la última computadora no puede ser agregada como una tupla (c, i, i, c), por eso es que existe π_3 en el observador infoP, el cual devuelve true cuando p está en su destino, y false cuando no. También vale aclarar que en π_1 se encuentra su computadora origen (es decir, donde se encoló), y en π_2 su computadora destino.

• Se decidió que las computadoras no puedan estar conectadas consigo mismas

• Si bien el observador caminoParcial devuelve el camino mínimo de un paquete (a medida que este viaja), se decidió que el camino mínimo a seguir (caminoTotal) se genere en la instancia en que se encola dicho paquete.

• METOINTERFACES toma una Secu(Compu) y devuelve una Secu(Compu, Interfaz, Interfaz, Compu). No logramos que entrara la signatura en la hoja. Hay un pequeño abuso de notación al no escribir la palabra TUPLA pero es porque así iba a entrar menos en la hoja y por ende entenderse menos.

• ENCOLAR no representa una cola ni trabaja con el TAD COLA, sí hay un comportamiento tipo 'cola' en su axiomatización. La razón de su nombre es porque es como que 'encola' paquetes en una computadora. Pero no hicimos una Cola y tampoco nos interesó observar, desde los observadores, quienes son los paquetes en espera (sí precisamos esta información en un momento y la obtuvimos mediante otra operación). Solo nos interesó saber cuántos había en espera en cada computadora.

• En la restricción de ENCOLAR, la operación hayConexión?(laRed(d), c_1, c_2) ya chequea que c_1 y c_2 sean distintas. Por eso no está explicito en la restricción.

• PAQUETESDE y PAQUETESDE2 devuelven el conjunto de paquetes 'en transito' de una computadora, es decir, no toman en cuenta los que sí están en esa computadora pero ya llegaron a su destino.

2. Renombres

INTERFAZ es NAT

COMPU es NAT

PAQUETE es TUPLA(NAT, NAT), EL π_1 DE LA TUPLA ES LA PRIORIDAD DEL PAQUETE, Y EL π_2 ES SU IDENTIFICADOR PERSONAL

3. Operaciones Auxiliares

otras operaciones

aConjunto	: Secu(α)	$\longrightarrow$	Conj(α)	
siguiente	: Secu(α) $s1$ × Secu(α) $s2$	$\longrightarrow$	α	{long(s_1) < long(s_2)}
masCorto	: Conj(Secu(α)) cs	$\longrightarrow$	Secu(α)	{¬∅?(cs)}
agATodos	: α × Conj(Secu(α))	$\longrightarrow$	Conj(Secu(α))	

axiomas $\forall e$: α, $\forall c$: Conj(α), $\forall s, s1, s2$: Secu(α). $\forall cs$: Conj(Secu(α))

aConjunto(s) ≡ **if** vacia?(s) **then** ∅ **else** Ag(prim(s), aConjunto(fin(s))) **fi**

siguiente(s_1, s_2) ≡ **if** vacia?(s_1) **then** prim(s_2) **else** siguiente(fin(s_1), fin(s_2)) **fi**

```
masCorto(cs) ≡ if #(cs) = 1 then
                   dameUno(cs)
               else
                   if long(dameUno(cs)) < long(dameUno(SinUno(cs))) then
                       masCorto(cs - dameUno(sinUno(cs)))
                   else
                       masCorto(sinUno(cs))
                   fi
               fi
```

agATodos(e, cs) ≡ **if** ∅?(cs) **then** ∅ **else** Ag((e • dameUno(cs)), agATodos(e, sinUno(cs))) **fi**

4. TAD Red

TAD RED

igualdad observacional

$$(\forall r1, r2 : \text{Red})\ \left(r1 =_{\text{obs}} r2 \iff \left(\begin{array}{l} \text{compus}(r_1) =_{\text{obs}} \text{compus}(r_2) \wedge_{\text{L}} \\ (\forall\ \text{c: Compu}) \\ [(\text{c} \in \text{compus}(r_1) \Rightarrow_{\text{L}} (\text{interfacesDe}(r_1, \text{c}) =_{\text{obs}} \\ \text{interfacesDe}(r_2, \text{c}))) \wedge_{\text{L}} \\ (\forall\ \text{i:Interfaz}) \\ [(\text{c} \in \text{compus}(r_1) \wedge_{\text{L}} \text{i} \in \text{interfacesDe}(r_1, \text{c}) \Rightarrow_{\text{L}} (\text{iLibre?}(r_1, \\ \text{c, i}) =_{\text{obs}} \text{iLibre?}(r_2, \text{c, i}))) \wedge_{\text{L}} \\ (\text{c} \in \text{compus}(r_1) \wedge_{\text{L}} \text{i} \in \text{interfacesDe}(r_1, \text{c}) \wedge_{\text{L}} \neg \text{iLibre?}(r_1, \\ \text{c, i}) \Rightarrow_{\text{L}} (\text{conectadoA}(r_1, \text{c, i}) =_{\text{obs}} \text{conectadoA}(r_2, \text{c, i})))]] \end{array} \right) \right)$$

géneros Red

exporta Red, generadores, observadores

usa BOOL, CONJ(α), COMPU, INTERFAZ, SECU(α)

observadores básicos

compus	: Red	$\longrightarrow$ Conj(Compu)	
interfacesDe	: Red r × Compu c	$\longrightarrow$ Conj(Interfaz)	{c ∈ compus(r)}
conectadoA	: Red r × Compu c × Interfaz i	$\longrightarrow$ Compu	{c ∈ compus(r) $\wedge_{\text{L}}$ i ∈ interfacesDe(r,c) $\wedge_{\text{L}}$ ¬ iLibre?(r, c, i)}
iLibre?	: Red r × Compu c × Interfaz i	$\longrightarrow$ Bool	{c ∈ compus(r) $\wedge_{\text{L}}$ i ∈ interfacesDe(r, c)}

generadores

crear	:	$\longrightarrow$ Red	
agCompu	: Red r × Compu c × Conj(Interfaz)	$\longrightarrow$ Red	{c ∉ compus(r)}
conectar	: Red r × Compu $c1$ × Compu $c2$ × Interfaz $i1$ × Interfaz $i2$	$\longrightarrow$ Red	$\left\{ \begin{array}{l} c_1 \neq c_2 \wedge \{c_1, c_2\} \subseteq \text{compus(r)} \wedge_{\text{L}} c_1 \notin \text{compusDirectas(r,}c_2) \wedge c_2 \notin \text{compusDirectas(r,}c_1) \wedge ((i_1 \\ \in \text{interfacesDe(r,}c_1) \wedge i_2 \in \text{interfacesDe(r,}c_2)) \wedge_{\text{L}} \text{iLibre?(r,}c_1\text{,}i_1) \wedge \text{iLibre?(r,}c_2\text{,}i_2)) \end{array} \right\}$

otras operaciones

iOcupadas	: Red r × Compu c	$\longrightarrow$ Conj(Interfaz)	{c ∈ compus(r)}
iOcupadas2	: Red r × Compu c × Conj(Interfaz) ci	$\longrightarrow$ Conj(Interfaz)	{c ∈ compus(r) $\wedge_{\text{L}}$ $c_i \subseteq$ interfacesDe(r,c)}
hayConexion?	: Red r × Compu $c1$ × Compu $c2$	$\longrightarrow$ Bool	{{c_1, c_2} ⊆ compus(r)}
hayConexion2?	: Red r × Compu $c1$ × Compu $c2$ × Conj(Compu) cc	$\longrightarrow$ Bool	{({c_1, c_2} ∪ cc) ⊆ compus(r)}
conjHayConexion?	: Red r × Conj(Compu) $cc1$ × Compu c × Conj(Compu) $cc2$	$\longrightarrow$ Bool	{({c} ∪ cc1 ∪ cc2) ⊆ compus(r)}
compusDirectas	: Red r × Compu c	$\longrightarrow$ Conj(Compu)	{c ∈ compus(r)}
compusDirectas2	: Red r × Compu c × Conj(Interfaz) ci	$\longrightarrow$ Conj(Compu)	{c ∈ compus(r) $\wedge_{\text{L}}$ ci ⊆ iOcupadas(r, c)}
TDC	: Red r × Compu $c1$ × Compu $c2$	$\longrightarrow$ Conj(Secu(Compu))	{{c_1, c_2} ⊆ compus(r)}
TDC2	: Red r × Compu $c1$ × Compu $c2$ × Secu(Compu) sc	$\longrightarrow$ Conj(Secu(Compu))	{{c_1, c_2} ⊆ compus(r)}
conjTDC	: Red r × Conj(Compu) cc × Compu c × Secu(Compu) sc	$\longrightarrow$ Conj(Secu(Compu))	{Ag(c, cc) ⊆ compus(r)}
suInterfaz	: Red r × Compu $c1$ × Compu $c2$	$\longrightarrow$ Interfaz	{{c_1, c_2} ⊆ compus(r) $\wedge_{\text{L}}$ $c_2 \in$ compusDirectas(r, c_1)}
suInterfaz2	: Red r × Compu $c1$ × Compu $c2$ × Conj(Interfaz) ci	$\longrightarrow$ Interfaz	{{c_1, c_2} ⊆ compus(r) $\wedge_{\text{L}}$ ci ⊆ iOcupadas(r, c_1)}
metoInterfaces	: Red r × Secu(Compu) sc	$\longrightarrow$ Secu(Compu, Interfaz, Interfaz	{¬vacia?(sc)}

axiomas $\forall r$: Red, $\forall c, c1, c2, c3$: Compu, $\forall i, i1, i2, i3$: Interfaz, $\forall ci$: Conj(Interfaz), $\forall cc, cc1, cc2$: Conj(Compu),

$\forall sc$: Secu(Compu)

compus(crear()) $\equiv \emptyset$

compus(AgCompu(r, c, ci)) $\equiv$ Ag(c , compus(r))

compus(conectar(r, c_1, c_2, i_1, i_2)) $\equiv$ compus(r)

interfacesDe(agCompu(r, c_1, ci), c_2) $\equiv$ **if** $c_1 = c_2$ **then** ci **else** interfacesDe(r, c_2) **fi**

interfacesDe(conectar(r, c_1, c_2, i_1, i_2), c_3) $\equiv$ interfacesDe(r, c_3)

conectadoA(agCompu(r, c_1, ci), c_2, i_2) $\equiv$ conectadoA(r, c_2, i_2)

conectadoA(conectar(r, c_1, c_2, i_1, i_2), c_3, i_3) $\equiv$ **if** $c_3 = c_1 \wedge i_3 = i_1$ **then**
c_2
else
if $c_3 = c_2 \wedge i_3 = i_2$ **then**
c_1
else
conectadoA(r, c_3, i_3)
fi
fi

iLibre?(agCompu(r, c_1, ci), c_2, i) $\equiv$ **if** $c_1 = c_2$ **then** true **else** iLibre?(r, c_2, i) **fi**

iLibre?(conectar(r, c_1, c_2, i_1, i_2), c_3, i_3) $\equiv$ **if** $c_3 = c_1 \wedge i_3 = i_1$ **then**
false
else
if $c_3 = c_2 \wedge i_3 = i_2$ **then**
false
else
iLibre?(r, c_3, i_3)
fi
fi

iOcupadas(r, c) $\equiv$ iOcupadas2(r, c, interfacesDe(r, c))

iOcupadas2(r, c, ci) $\equiv$ **if** $\emptyset$?(ci) **then**
$\emptyset$
else
if $\neg$ iLibre?(r, c, dameUno(ci)) **then**
Ag(dameUno(ci), iOcupadas2(r, c, sinUno(ci)))
else
iOcupadas2(r, c, sinUno(ci))
fi
fi

hayConexion?(r, c_1, c_2) $\equiv$ hayConexion2?(r, c_1, c_2, $\emptyset$)

hayConexion2?(r, c_1, c_2, cc) $\equiv$ **if** $c_2 \in$ compusDirectas(r, c_1) **then**
true
else
conjHayConexion?(r, compusDirectas(r, c_1) - cc, c_2, Ag(c_1, cc))
fi

conjHayConexion?(r, cc_1, c, cc_2) $\equiv$ **if** $\emptyset$?(cc_1) **then**
false
else
hayConexion2?(r, dameUno(cc_1), c, cc_2) $\vee$ conjHayConexion?(r, sinUno(cc_1), c, cc_2)
fi

compusDirectas(r, c) $\equiv$ compusDirectas2(r, c, iOcupadas(r, c))

compusDirectas2(r, c, ci) $\equiv$ **if** $\emptyset$?(ci) **then**
$\emptyset$
else
Ag((conectadoA(r, c, dameUno(ci))), compusDirectas2(r, c, sinUno(ci)))
fi

TDC(r, c_1, c_2) $\equiv$ agATodos(c_1, TDC2(r, c_1, c_2, <>))

TDC2(r, c_1, c_2, sc) $\equiv$ **if** $c_2 \in$ compusDirectas(r, c_1) **then**
{sc & <c_1, c_2>}
else
conjTDC(r, compusDirectas(r, c_1) - aConjunto(sc), c_2, sc $\circ$ c_1)
fi

conjTDC(r, cc, c, sc) $\equiv$ **if** $\emptyset$?(cc) **then**
$\emptyset$
else
TDC2(r, dameUno(cc), c, sc) $\bigcup$ conjTDC(r, sinUno(cc), c, sc)
fi

suInterfaz(r, c_1, c_2) $\equiv$ suInterfaz2(r, c_1, c_2, iOcupadas(r, c_1))

suInterfaz2(r, c_1, c_2, ci) $\equiv$ **if** conectadoA(r, c_1, dameUno(ci)) = c_2 **then**
dameUno(ci)
else
suInterfaz2(r, c_1, c_2, sinUno(ci))
fi

metoInterfaces(r, sc) $\equiv$ **if** vacia?(fin(sc)) **then**
<>
else
(prim(sc), suInterfaz(r, prim(sc), prim(fin(sc))), suInterfaz(r, prim(fin(sc)), prim(sc)), prim(fin(sc))) • metoInterfaces(r, fin(sc))
fi

Fin TAD

5. TAD DCNet

TAD DCNET

igualdad observacional

$$(\forall d1, d2 : \text{DCNet}) \left(d1 =_{\text{obs}} d2 \iff \left(\begin{array}{l} \text{laRed}(d_1) =_{\text{obs}} \text{laRed}(d_2) \wedge \\ \text{paquetes}(d_1) =_{\text{obs}} \text{paquetes}(d_2) \wedge_{\text{L}} \\ [(\forall \text{c: Compu}) \ (\text{c} \in \text{compus}(\text{laRed}(d_1)) \Rightarrow_{\text{L}} \\ (\text{cantEnviados}(d_1, \text{c}) =_{\text{obs}} \text{cantEnviados}(d_2, \text{c}))) \\ \wedge \\ (\forall \text{p: Paquete}) \ (\text{p} \in \text{paquetes}(d_1) \Rightarrow_{\text{L}} (\text{infoP}(d_1, \\ \text{p}) =_{\text{obs}} \text{infoP}(d_2, \text{p}) \wedge \text{caminoParcial}(d_1, \text{p}) =_{\text{obs}} \\ \text{caminoParcial}(d_2, \text{p})))] \end{array} \right) \right)$$

géneros DCNet

exporta DCNet, generadores, observadores

usa RED, SECU(α), CONJUNTO(α), NAT, BOOL, PAQUETE, TUPLA

observadores básicos

laRed	: DCNet	$\longrightarrow$	Red	
paquetes	: DCNet	$\longrightarrow$	Conj(Paquete)	
infoP	: DCNet d × Paquete p	$\longrightarrow$	(Compu, Compu, Bool)	{p ∈ paquetes(d)}
cantEnviados	: DCNet d × Compu c	$\longrightarrow$	Nat	{c ∈ compus(laRed(d))}
caminoParcial	: DCNet d × Paquete p	$\longrightarrow$	Secu(Compu, Interfaz, Interfaz, Compu)	{p ∈ paquetes(d)}

generadores

iniciar	: Red	$\longrightarrow$	DCNet	
encolar	: DCNet d × Compu $c1$ × Compu $c2$ × Paquete p	$\longrightarrow$	DCNet	$\left\{ \begin{array}{l} [(\forall\ p_2\text{: Paquete}) \ (p_2 \in \text{paquetes(d)} \Rightarrow \pi_2(p_2) \neq \pi_2(\text{p}))] \wedge \{c_1, c_2\} \subseteq \text{compus(laRed(d))} \wedge_{\text{L}} \text{hay-} \\ \text{Conexion?(laRed(d)}, c_1, c_2) \end{array} \right\}$
pasarSeg	: DCNet	$\longrightarrow$	DCNet	

otras operaciones

masEnviante	: DCNet d	$\longrightarrow$	Compu	{¬∅?(compus(laRed(d)))}
masEnviante2	: DCNet d × Conj(Compu) cc	$\longrightarrow$	Compu	{cc ⊆ compus(laRed(d)) ∧ ¬ ∅?(cc)}
conjMasEnviante	: DCNet d × Conj(Compu) cc × Compu c	$\longrightarrow$	Conj(Compu)	{Ag(c, cc) ⊆ compus(laRed(d))}
caminoTotal	: DCNet d × Paquete p	$\longrightarrow$	Secu(Compu, Interfaz, Interfaz, Compu)	{p ∈ paquetes(d)}
paquetesDe	: DCNet × Compu	$\longrightarrow$	Conj(Paquete)	
paquetesDe2	: DCNet d × Compu × Conj(Paquete) cp	$\longrightarrow$	Conj(Paquete)	{cp ⊆ paquetes(d)}
masPrioritario	: Conj(Paquete) cp	$\longrightarrow$	Paquete	{¬∅?(cp)}

axiomas $\forall r$: Red, $\forall d$: DCNet, $\forall c, c1, c2, c3$: Compu, $\forall p$: Paquete, $\forall cc$: Conj(Compu), $\forall cp$: Conj(Paquete)

laRed(iniciar(r))	≡	r
laRed(encolar(d, c_1, c_2, p))	≡	laRed(d)
laRed(pasarSeg(d))	≡	laRed(d)
paquetes(iniciar(r))	≡	∅
paquetes(encolar(d, c_1, c_2, p))	≡	Ag(p, paquetes(d))
paquetes(pasarSeg(d))	≡	paquetes(d)
infoP(encolar(d, c_1, c_2, p_1), p_2)	≡	**if** $p_1 = p_2$ **then** (c_1, c_2, false) **else** infoP(d, p_2) **fi**

infoP(pasarSeg(d), p) ≡ **if** π_4(ult(caminoParcial(d, p))) = π_2(infoP(d, p)) ∧ ¬π_3(infoP(d, p)) $\wedge_{\text{L}}$ masPrioritario(paquetesDe(d, π_1(ult(caminoParcial(d, p))))) = p **then** (π_1(infoP(d, p)), π_2(infoP(d, p)), true) **else** infoP(d, p) **fi**

cantEnviados(iniciar(r), c) ≡ 0

cantEnviados(encolar(d, c_1, c_2, p), c_3) ≡ cantEnviados(d, c_3)

cantEnviados(pasarSeg(d), c) ≡ **if** #paquetesDe(d, c) = 0 **then** cantEnviados(d, c) **else** cantEnviados(d, c) + 1 **fi**

caminoParcial(encolar(d, c_1, c_2, p_1), p_2) ≡ **if** $p_1 = p_2$ **then** <prim(caminoTotal(encolar(d, c_1, c_2, p_1), p_2))> **else** caminoParcial(d, p_2) **fi**

caminoParcial(pasarSeg(d), p) ≡ **if** π_4(ult(caminoParcial(d, p))) = π_2(infoP(d, p)) **then** caminoParcial(d, p) **else** **if** masPrioritario(paquetesDe(d, π_1(ult(caminoParcial(d, p))))) = p **then** caminoParcial(d, p) ∘ siguiente(caminoParcial(d, p), caminoTotal(d, p)) **else** caminoParcial(d, p) **fi** **fi**

masEnviante(d) ≡ dameUno(conjMasEnviante(d, compus(laRed(d)), masEnviante2(d, compus(laRed(d)))))

masEnviante2(d, cc) ≡ **if** #(cc) = 1 **then** dameUno(cc) **else** **if** cantEnviados(d, dameUno(cc)) ≥ cantEnviados(d, dameUno(sinUno(cc))) **then** masEnviante2(d, cc - dameUno(sinUno(cc))) **else** masEnviante2(d, cc - dameUno(cc)) **fi** **fi**

conjMasEnviante(d, cc, c) ≡ **if** ∅?(cc) **then** ∅ **else** **if** cantEnviados(d, dameUno(cc)) = cantEnviados(d, c) **then** Ag(dameUno(cc), conjMasEnviante(d, sinUno(cc), c)) **else** conjMasEnviante(d, sinUno(cc), c) **fi** **fi**

caminoTotal(d, p) ≡ metoInterfaces(laRed(d), masCorto(TDC(laRed(d), π_1(infoP(d, p)), π_2(infoP(d, p)))))

paquetesDe(d, c) ≡ paquetesDe2(d, c, paquetes(d))

paquetesDe2(d, c, cp) ≡ **if** $\emptyset?$(cp) **then**
$\emptyset$
else
if π_1(ult(caminoParcial(d, dameUno(cp)))) = c $\wedge$
$\neg\pi_3$(infoP(d, dameUno(cp)))
then
Ag(dameUno(cp), paquetesDe2(d, c, sinUno(cp)))
else
paquetesDe2(d, c, sinUno(cp))
fi
fi

masPrioritario(cp) ≡ **if** #(cp) = 1 **then**
dameUno(cp)
else
if π_1(dameUno(cp)) > π_1(dameUno(sinUno(cp))) **then**
masPrioritario(cp - dameUno(sinUno(cp)))
else
masPrioritario(cp - dameUno(cp))
fi
fi

Fin TAD