Algoritmos y Estructuras de Datos II

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

Trabajo Práctico 1

Órdenes de Producción

Grupo 6

Integrante	LU	Correo electrónico
Aronson, Alex	443/008	alexaronson@gmail.com
Bianchetti, Marcelo	378/08	bianchetti.ml@gmail.com
Pacosillo Mamani, Jhonny	796/06	jpacosillo@hotmail.com
Ravasi, Nicolás	53/08	nravasi@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

1. Signatura de los TADs

TAD WORKFLOW

géneros workflow

exporta workflow, generadores, observadores, workflowValido?, idTareaFinal, esTarea?

usa Nat, Tarea, Recursos, Bool

igualdad observacional

$$(\forall a, b : \text{workflow}) \left(a =_{\text{obs}} b \iff \begin{pmatrix} \text{dameIDTareas } (a) =_{\text{obs}} \text{dameIDTareas } (b) \wedge_{\text{L}} (\forall\ t\text{: tarea}) \\ \text{esTarea? } (a,\ t) \Rightarrow_{\text{L}} \text{verRecursosTarea } (a,t) =_{\text{obs}} \text{verRecursosTarea } (b,t) \wedge \text{verReqsTarea } (a,t) =_{\text{obs}} \text{verReqsTarea } (b,t) \end{pmatrix} \right)$$

observadores básicos

dameIDTareas	: workflow	$\longrightarrow$	conj(tarea)	
verRecursosTarea	: workflow w × tarea t	$\longrightarrow$	recursos	{esTarea? (w,t)}
verReqsTarea	: workflow w × tarea t	$\longrightarrow$	conj(nat)	{esTarea? (w,t)}

generadores

crear	: recursos	$\longrightarrow$	workflow
agTarea	: tarea t × recursos × conj(tarea) ct × workflow w	$\longrightarrow$	workflow
			{$0 < n \wedge \neg$ esTarea?(w,n) $\wedge \forall$ (x:tarea, $x \in ct$) esTarea?(w,x)}

otras operaciones

esTarea?	: workflow × nat	$\longrightarrow$	bool
workflowValido?	: workflow	$\longrightarrow$	bool
idTareaFinal	: workflow	$\longrightarrow$	conj(tarea)
todosLosReqs	: workflow × conj(tarea)	$\longrightarrow$	conj(tarea)
OtraOperacion	: param	$\longrightarrow$	resul

axiomas

$\forall w$: workflow, $\forall t,n$: tarea, $\forall r$: recursos, $\forall ct$: conj(tarea)

dameIDTareas (crear (r)) $\equiv$ Ag (0, $\emptyset$)
dameIDTareas (agTarea (t,r,ct,w)) $\equiv$ Ag (t, dameIDTareas (w))

verRecursosTarea (crear (r), n) $\equiv$ r
verRecursosTarea (agTarea (t,r,ct,w), n) $\equiv$ **if** $n = t$ **then** r **else** verRecursosTarea (w,n) **fi**

verReqsTarea (crear (r), n) $\equiv$ $\emptyset$
verReqsTarea (agTarea (t,r,ct,w), n) $\equiv$ **if** $n = t$ **then** ct **else** verReqsTarea (w,n) **fi**

esTarea? (w,t) $\equiv$ $t \in$ dameIDTareas (w)

workflowValido? (w) $\equiv$ $\emptyset$? (sinUno (idTareaFinal (w)))

idTareaFinal (w) $\equiv$ dameIDTareas (w) - todosLosReqs (w, idTareas(w))

todosLosReqs (w,ct) $\equiv$ **if** $\emptyset$? (ct) **then** $\emptyset$ **else** verReqsTarea(dameUno(ct), w) $\cup$ todosLosReqs (w, sinUno(ct)) **fi**

Fin TAD

TAD PRODUCCION

géneros produccion

exporta produccion, generadores, obsevadores, mostrarTareas, enEjecucion?, terminoOrden?, verRecursosDisponibles

usa Bool, Tarea, Orden, Recursos, Workflow, Conjunto

igualdad observacional

$$(\forall a,b: \text{produccion})\ (a =_{obs} b \Longleftrightarrow \begin{pmatrix} \text{verWorkflow } (a) =_{obs} \text{verWorkflow } (b) \wedge \text{verRecursos } (a) \\ =_{obs} \text{verRecursos } (b) \wedge \text{ultimaOrden } (a) =_{obs} \text{ultimaOrden} \\ (b) \wedge \text{tareasEjecutandose } (a) =_{obs} \text{tareasEjecutandose } (b) \\ \wedge_{L} (\forall\ ord\text{: orden, } t\text{: tarea}) (0 < ord \leq \text{ultimaOrden } (a) \wedge \\ \text{esTarea } (\text{verWorkflow } (p), t)) \Rightarrow_{L} \text{terminoTarea } (a,o,t) =_{obs} \\ \text{terminoTarea } (b,o,t) \end{pmatrix}$$

observadores básicos

verWorkflow : produccion $\longrightarrow$ workflow

verRecursos : produccion $\longrightarrow$ recursos

terminoTarea? : produccion p $\times$ orden ord $\times$ tarea t $\longrightarrow$ bool
$\{0 < ord \leq \text{ultimaOrden } (p) \wedge \text{esTarea? } (\text{verWorkflow } (p), t)\}$

tareasEjecutandose : produccion $\longrightarrow$ $\{\text{conj(fase)}\}$

ultimaOrden : produccion $\longrightarrow$ orden

generadores

crear : workflow w $\times$ recursos $\longrightarrow$ produccion $\{\text{workflowValido? } (w)\}$

agregarOrden : produccion p $\longrightarrow$ produccion

terminarTarea : produccion p $\times$ orden ord $\times$ tarea t $\longrightarrow$ produccion
$\{\text{fase}(ord,t) \in \text{tareasEjecutandose}(p)\}$

otras operaciones

terminoOrden? : produccion p $\times$ orden ord $\longrightarrow$ bool
$\{0 < ord \leq \text{ultimaOrden } (p)\}$

mostrarTareas : produccion p $\times$ orden ord $\longrightarrow$ dicc (tarea,string)
$\{0 < ord \leq \text{ultimaOrden } (p)\}$

generarDiccionarioTareas : produccion p $\times$ orden ord $\times$ conj(tarea) ct $\longrightarrow$ dicc (tarea,string)
$\{0 < ord \leq \text{ultimaOrden } (p) \wedge (\forall\ t\text{:tarea}, t \in ct)\ \text{esTarea? } (\text{verWorkflow } (p), t)\}$

enEjecucion? : produccion p $\times$ orden ord $\times$ tarea t $\longrightarrow$ bool
$\{0 < ord \leq \text{ultimaOrden}(p)\}$

terminaronTodosReq : produccion p $\times$ orden ord $\times$ conj(tarea) ct $\longrightarrow$ bool
$\{ord = 0 \wedge \emptyset?\ (\text{ct}) \vee 0 < ord \leq \text{ultimaOrden } (p) \wedge (\forall\ t\text{:tarea}, t \in ct)\ \text{esTarea? } (\text{verWorkflow } (p), t)\}$

hayRecursos : recursos $\times$ recursos $\longrightarrow$ bool

reqsSatisfechos : produccion $\longrightarrow$ conj(fase)

reqsSatisfechosAux : produccion $\times$ orden ord $\times$ conj(tarea) $\longrightarrow$ conj(fase)
$\{ord = 0 \wedge \emptyset?\ (\text{ct}) \vee 0 < ord \leq \text{ultimaOrden } (p) \wedge (\forall\ t\text{:tarea}, t \in ct)\ \text{esTarea? } (\text{verWorkflow } (p), t)\}$

tareasNuevasEjecutandose : recursos $\times$ workflow $\times$ conj(fase) $\longrightarrow$ conj(fase)

nuevosSatisfechos : produccion $\times$ orden $\times$ conj(tarea) $\times$ tarea $\longrightarrow$ conj(fase)
$\{ord = 0 \wedge \emptyset?\ (\text{ct}) \vee 0 < ord \leq \text{ultimaOrden } (p) \wedge (\forall\ t\text{:tarea}, t \in ct)\ \text{esTarea? } (\text{verWorkflow } (p), t)\}$

tareasCompletadasOrden : produccion $\times$ orden $\times$ conj(tarea) $\longrightarrow$ conj(tarea)
$\{ord \leq \text{ultimaOrden } (p) \wedge (\forall\ t\text{:tarea}, t \in ct)\ \text{esTarea? } (\text{verWorkflow } (p), t)\}$

verRecursosDisponibles : produccion $\longrightarrow$ recursos

recursosTareasEjecutandose: conj(fase)) $\times$ workflow $\longrightarrow$ recursos

axiomas

$\forall p$: produccion, $\forall w$: workflow, $\forall r,r1,r2$: recursos, $\forall t,t1,t2$: tarea, $\forall ord,ord1,ord2$: orden, $\forall ct$: conj(tarea), $\forall cf$: conj(fase)

verWorkflow (crear (w,r)) $\equiv$ w
verWorkflow (agregarOrden (p)) $\equiv$ verWorkflow (p)
verWorkflow (terminarTarea (p,ord,t)) $\equiv$ verWorkflow (p)

verRecursos (crear (w,r)) $\equiv$ r
verRecursos (agregarOrden (p)) $\equiv$ verRecursos (p)
verRecursos (terminarTarea (p,ord,t)) $\equiv$ verRecursos (p)

terminoTarea? (agregarOrden (p), ord, t) $\equiv$ **if** ultimaOrden(p) $<$ ord **then**
false
else
terminoTarea? (p,ord,t)
fi

terminoTarea? (terminarTarea (p,$ord1$,$t1$), $ord2$, $t2$) $\equiv$ **if** $ord1 = ord2 \wedge t1 = t2$ **then**
true
else
terminoTarea? (p,$ord2$,$t2$)
fi

ultimaOrden (crear (w,r)) $\equiv$ 0
ultimaOrden (agregarOrden (p)) $\equiv$ 1 + ultimaOrden(p)
ultimaOrden (terminarTarea (p,ord,t)) $\equiv$ ultimaOrden (p)

tareasEjecutandose (crear (w,r)) $\equiv$ $\emptyset$
tareasEjecutandose (agregarOrden (p)) $\equiv$ **if** (hayRecursos (r, verRecursosTarea (w, 0))) **then**
Ag (fase((ultimaOrden (p) + 1),0), tareasEjecutandose (p))
else
tareasEjecutandose (p)
fi

tareasEjecutandose (terminarTarea (p,ord,t)) $\equiv$ (tareasEjecutandose (p) - fase(ord,t)) $\cup$ tareasNuevasEjecutandose (agregarRecursosLiberados (p, t), verWorkflow (p), (actualizarRequerimientosSatisfechos (p, ord, t)))

agregarRecursosLiberados (p, t) $\equiv$ verRecursosDisponibles (p) $\cup$ verRecursosTarea (verWorkflow(p), t)

actualizarRequerimientosSatisfechos (p, ord, t) $\equiv$ (reqsSatisfechos (p) $\cup$ nuevosSatisfechos (verWorkflow (p), ord, dameIDTareas (verWorkflow (p) - Ag (tareasCompletadasOrden (p, ord, verIDTareas (verWorfklow(p))), t))))

tareasNuevasEjecutandose (r,w,cf) $\equiv$ **if** $\emptyset$? (cf) **then**
$\emptyset$
else
if hayRecursos (r, verRecursosTarea(w, tarea(dameUno(cf)))) **then**
Ag (dameUno (cf), tareasNuevasEjecutandose (r - (verRecursosTarea(w, (tarea(dameUno(cf))))), w, sinUno (cf))
else
tareasNuevasEjecutandose (r, w, sinUno (cf)
fi
fi

nuevosSatisfechos (p,ord,ct,t)) $\equiv$ **if** $\emptyset$? (ct) **then**
$\emptyset$
else
if terminaronTodosReq (p, ord, (verReqsTarea (verWorkflow (p), dameUno (ct)) - t) **then**
Ag (fase(ord,dameUno (ct)), nuevosSatisfechos (p, ord, sinUno (ct), t)
else
nuevosSatisfechos (p, ord, sinUno (ct), t)
fi
fi

tareasCompletadasOrden (p,ord,ct) $\equiv$ **if** $\emptyset$? (ct) **then**
$\emptyset$
else
if terminoTarea(dameUno (ct)) **then**
Ag (dameUno (ct), tareasCompletadasOrden (p, ord, sinUno (ct)))
else
tareasCompletadasOrden (p, ord, sinUno (ct))
fi
fi

terminoOrden? (p,ord) $\equiv$ terminoTarea? (p ,ord, dameUno (idTareaFinal (verWorkflow(p))))

mostrarTareas (p,ord) $\equiv$ generarDiccionarioTareas (p, ord, dameIDTareas (verWorkflow(p)))

generarDiccionarioTareas (p,ord,ct) $\equiv$ **if** $\emptyset$? (ct) **then**
vacio
else
if terminoTarea? (p, ord, dameUno (ct)) **then**
definir (dameUno (ct), 'Tarea Finalizada', generarDiccionarioTareas (p, ord, sinUno (ct)))
else
if enEjecucion? (p, ord, dameUno (ct) **then**
definir (dameUno (ct), Én Ejecucïï¿½n', generarDiccionarioTareas (p, ord, sinUno (ct)))
else
definir (dameUno (ct), 'Tarea No Iniciada', generarDiccionarioTareas (p, ord, sinUno (ct)))
fi
fi
fi

enEjecucion? (p,ord,t) $\equiv$ fase(ord,t) $\in$ tareasEjecutandose (p)

hayRecursos ($r1$,$r2$) $\equiv$ **if** $\emptyset$? ($r2$) **then** True **else** # (dameUno ($r2$), $r1$) $> 0 \wedge_{\text{L}}$ hayRecursos ($r1$ - dameUno ($r2$), sinUno ($r2$))

terminaronTodosReq (p,ord,ct) $\equiv$ **if** $\emptyset$? (cf) **then** $\emptyset$ **else** terminoTarea? (p, ord, dameUno (ct)) $\wedge$ terminaronTodosReq(p, ord, sinUno(ct))

reqsSatisfechos (p) $\equiv$ reqsSatisfechosAux (p, ultimaOrden(p), dameIDTareas (verWorkflow(p)))

reqsSatisfechosAux (p,ord,ct) $\equiv$ **if** $ord = 0$ **then**
$\emptyset$
else
if $\emptyset$? (ct) **then**
reqsSatisfechosAux (p, ord -1, dameIDTareas (verWorkflow (p)))
else
if terminaronTodosReq? (p, ord, verReqsTarea(verWorkflow, tarea(dameUno(c))) **then**
Ag (fase(ord, dameUno(ct)), reqsSatisfechosAux (p, ord, sinUno (ct)))
else
(reqsSatisfechosAux (p, ord, sinUno (c)))
fi
fi
fi

verRecursosDisponibles (p) $\equiv$ verRecursos (p) - recursosTareasEjecutandose (tareasEjecutandose (p), verWorkflow(p))

recursosTareasEjecutandose (cf,w)) $\equiv$ **if** $\emptyset$? (cf) **then** $\emptyset$ **else** verRecursosTarea (w, tarea (dameUno(cf))) $\cup$ recursosTareasEjecutandose (sinUno (cf), w)

Fin TAD

TAD Fase
Es un renombre de la tupla de $< nat, nat >$ cuyos observadores π_1 y π_2 fueron renombrados orden y tarea respectivamente, y el generador que toma un orden y una tarea y devuelve una fase fue renombrado fase

TAD Tarea
Es nat

TAD Orden
Es nat

TAD Recursos
Es $multiconjunto(nat)$

2. Informe

En este trabajo hicimos uso de una estructura de TADs en dos niveles. En primer lugar creamos el TAD Workflow, que representa la jerarquía de tareas necesarias para llevar a cabo una orden, en sí simboliza la tabla indicada en las consignas del TP con cada tarea cons sus respectivos recursos y tareas previas requeridas; el workflow se crea con una sola tarea, a la que le asignamos el ID 0, y que no requiere de otras tareas para comenzar. Luego se agregan tareas progresivamente, cada una sólo puede requerir como tareas anteriores aquellas que ya hayan sido agregadas al workflow previamente. No creímos necesario poner algún límite sobre el ID de cada tarea que se agregue, por lo tanto es técnicamente posible que por ejemplo la tarea final sea la 1 y una de las primeras la 24, si bien en la realidad eso sería poco claro, nos pareció que era muy restrictivo pedir más que eso y creímos que era mejor dejar esa decisión sobre cómo asignar los IDs en manos del usuario

El otro nivel de TAD es el representado por Produccion, que toma un Workflow al que requerimos que sea válido y un número de recursos que permanecerá invariable a lo largo de toda la ejecución. Para determinar si un Worklow es válido, tuvimos en cuenta lo pedido por la materia; ya que nuestrar reglas mismas de construcción de un workflow impedían tanto designar a más de una tarea como inicial como crear ciclos en los requerimientos de las tareas, solamente debíamos pedir que no hubiera más de una tarea final, lo que lo hicimos comprobando que no hubiera más de una tarea que no fuera requerida por ninguna otra.

Al agregarse órdenes vimos necesario el tener que distinguir las diferentes tareas de cada orden, y por eso creamos el TAD Fase, que si bien es simplemente un renombre de la tupla de dos elementos, nos ayuda a poder visualizar claramente a qué número de orden pertenece cada tarea. Ante la imposiblidad de elegir, dadas varias tareas posibles, cuál es la que empieza a ejecutarse cuando las condiciones están dadas, decidimos no especificar cuál es la tarea que se ejecuta en ese caso. Para lograr esto, apelamos a crearnos una lista de posibles tareas ejecutables, que simplemente toma todas las tareas cuyos requerimientos previos han sido satisfechos. Luego, se llama a la función dameUno que toma un elemento determinísticamente, pero sin que nosotros tengamos que especificar cuál. Dado ese elemento, chequeamos si los recursos disponibles en ese momento alcanzan para ejecutarla, y en caso de que ello suceda, la agregamos como tarea en ejecución, continuando en tanto en caso de ser agregada como en caso de que no verificando si hay más tareas por ejecutar (habiendo actualizado los recursos si la tarea entró en ejecución para marcar que hay menos recursos disponibles).

Para cumplir con lo pedido por el TP, creamos dos funciones que serán exportadas con el TAD que sirven para conocer el estado de las órdenes. Éstas son terminoOrden?, que simplemente consulta si la tarea final del workflow, dada una orden, terminó su ejecución; y mostrarTareas, que también dada una orden, devuelve un diccionario cuyas claves son todas las tareas de ese workflow y sus definiciones muestran el estado en el que tal tarea se encuentra, pudiendo ser este Tarea Finalizada, En Ejecución o Tarea No Iniciada, según corresponda.