Algoritmos y Estructuras de Datos II
Primer parcial – 27 de septiembre de 2013

Aclaraciones

- 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 **MB**, **B**, **R** o **M**, y podrá recuperarse independientemente de los demás. Podrán aprobarse los parciales de la materia con hasta un ejercicio **R** (regular) y uno **M** (mal) **entre ambos exámenes**, siempre que ninguno de dichos ejercicios sea un ejercicio 1. Para más detalles ver la sección *Cursada* del sitio Web.

Ej. 1. Especificación

Los discos rígidos guardan la información de a bloques que se numeran de manera secuencial. Un *file system* (abreviado *fs*) es una estructura de datos que indica para cada archivo, en dónde se encuentra almacenada su información. Nos interesa especificar un *fs* que permita las siguientes operaciones:

- Crear un *fs* indicando el tamaño (en bloques) del disco sobre el que se crea.
- Ingresar un archivo, indicando su nombre y su tamaño (en bloques).
- Dado el nombre de un archivo, poder consultar qué bloques le corresponden.
- Dado un bloque, poder consultar el nombre del archivo allí alojado.

Debido a que la política específica de asignación de bloques será definida más adelante (en la implementación), nuestra especificación deberá ser lo más general posible respecto de qué bloques se le asignan a cada archivo. Lo único que se sabe con certeza a esta altura es que un bloque del disco puede alojar como máximo a un archivo.

Ej. 2. Inducción estructural

El TAD SecuenciaRaraConDobles modela cierta estructura similar a las secuencias de naturales, pero con la particularidad de que cada nodo puede ser normal o *doble* (léase, contener dos nats en lugar de uno).

Sus generadores son los siguientes:

$$\begin{array}{lll} \text{<>} & : & \longrightarrow \text{zecu} \\ \text{ag} & : \text{nat} \times \text{zecu} & \longrightarrow \text{zecu} \\ \text{agDoble} & : \text{nat} \times \text{nat} \times \text{zecu} & \longrightarrow \text{zecu} \end{array}$$

De sus operaciones, las relevantes para este ejercicio son:

$$\begin{array}{lll} \text{doblesOrd?} : \text{zecu} \longrightarrow \text{bool} & & \\ \quad ord_0)\ \text{doblesOrd?}(\text{<>}) & \equiv & \text{true} \\ \quad ord_1)\ \text{doblesOrd?}(\text{ag}(a, z)) & \equiv & \text{doblesOrd?}(z) \\ \quad ord_2)\ \text{doblesOrd?}(\text{agDoble}(a, b, z)) & \equiv & a \leq b \land \text{doblesOrd?}(z) \end{array}$$

$$\begin{array}{lll} \text{totalMax} : \text{zecu} \longrightarrow \text{nat} & & \\ \quad tot_0)\ \text{totalMax}(\text{<>}) & \equiv & 0 \\ \quad tot_1)\ \text{totalMax}(\text{ag}(a, z)) & \equiv & a + \text{totalMax}(z) \\ \quad tot_2)\ \text{totalMax}(\text{agDoble}(a, b, z)) & \equiv & \textbf{if } a \geq b \textbf{ then } a \textbf{ else } b \textbf{ fi} + \text{totalMax}(z) \end{array}$$

$$\begin{array}{lll} \text{aplanarSeg} : \text{zecu} \longrightarrow \text{zecu} & & \\ \quad apl_0)\ \text{aplanarSeg}(\text{<>}) & \equiv & \text{<>} \\ \quad apl_1)\ \text{aplanarSeg}(\text{ag}(a, z)) & \equiv & \text{ag}(a, \text{aplanarSeg}(z)) \\ \quad apl_2)\ \text{aplanarSeg}(\text{agDoble}(a, b, z)) & \equiv & \text{ag}(b, \text{aplanarSeg}(z)) \end{array}$$

Interesa demostrar por inducción estructural la siguiente propiedad:

$$\big(\forall z : \text{zecu}\big)\ \big(\text{doblesOrd?}(z) \implies \text{totalMax}(z) \equiv \text{totalMax}(\text{aplanarSeg}(z))\big)$$

a) Escribir el predicado unario. Luego exhibir el esquema completo de inducción a utilizar. Marcar **claramente** CB(s), PI(s), HI(s), TI(s) y alcance de cada cuantificador involucrado.

b) Plantear el/los caso(s) base y resolverlo(s), justificando cada paso de la demostración.

c) Plantear el/los paso(s) inductivo(s) y resolverlo(s), justificando cada paso de la demostración.

Ej. 3. Diseño

Una empresa autopartista almacena información sobre piezas de automóviles. Cada pieza corresponde a cierto modelo de automóvil. Algunas piezas están a su vez compuestas por subpiezas, que por supuesto deben servir para los mismos modelos. La parte de la especificación que nos interesa es:

TAD PARTE **ES** NAT
TAD MODELO **ES** STRING

TAD AUTOPARTISTA

observadores básicos

modelos	: autopartista	$\longrightarrow$	conj(modelo)	
partes	: autopartista	$\longrightarrow$	conj(parte)	
modelo	: autopartista a × parte p	$\longrightarrow$	modelo	$\{p \in \text{partes}(a)\}$
subPartes	: autopartista a × parte p	$\longrightarrow$	conj(parte)	$\{p \in \text{partes}(a)\}$

generadores

fundarAutopartista	:	$\longrightarrow$	autopartista	
registrarModelo	: autopartista a × modelo m	$\longrightarrow$	autopartista	$\{m \notin \text{modelos}(a)\}$
registrarParte	: autopartista a × modelo m × parte p	$\longrightarrow$	autopartista	$\{m \in \text{modelos}(a) \wedge p \notin \text{partes}(a)\}$
registrarSubParte	: autopartista a × parte p × parte sp	$\longrightarrow$	autopartista	

$$\left\{\begin{array}{l} p \in \text{partes}(a) \wedge sp \in \text{partes}(a) \wedge_{\text{L}} \text{modelo}(sp) = \text{modelo}(p) \wedge \\ \quad p \notin \text{clausuraDeSubPartes}(a, sp) \wedge (\forall p' : \text{parte})(p' \in \text{partes}(a) \Rightarrow_{\text{L}} sp \notin \text{subPartes}(a, p')) \end{array}\right\}$$

axiomas

modelos(fundarAutopartista())	$\equiv$	$\emptyset$
modelos(registrarModelo(a, m))	$\equiv$	Ag(m, modelos(a))
...		
partes(fundarAutopartista())	$\equiv$	$\emptyset$
partes(registrarParte(a, m, p))	$\equiv$	Ag(p, partes(a))
...		
modelo(registrarParte(a, m, p), p')	$\equiv$	**if** $p = p'$ **then** m **else** modelo(a, p') **fi**
...		
subPartes(registrarParte(a, m, p), p')	$\equiv$	**if** $p = p'$ **then** $\emptyset$ **else** subPartes(a, p') **fi**
subPartes(registrarSubParte(a, p, sp), p')	$\equiv$	**if** $p = p'$ **then** Ag(sp, $\emptyset$) **else** $\emptyset$ **fi** $\cup$ subPartes(a, p')

Fin TAD

donde clausuraDeSubPartes : autopartista a × parte $p \longrightarrow$ conj(parte) devuelve no solamente las subpartes de p sino también sus subsubpartes, subsubsubpartes, y así hasta incluir todo lo alcanzable desde p (excepto p).

Para representar el género imperativo Autopartista se decidió utilizar la siguiente estructura:

Autopartista **se representa con** estr, donde

estr es tupla ⟨ *modelos*: conj(modelo),
partes: dicc(parte, tupla<modelo, conj(parte)>),
parte_madre: dicc(parte, parte),
tiene_subpartes: dicc(parte, bool) ⟩

donde *tiene_subpartes* indica qué partes tienen subpartes, *parte_madre* indica, dada una subparte, cuál es la parte a la que pertenece (en forma directa, no transitivamente), y *partes* indica el modelo al que pertenece la parte así como también el conjunto con la totalidad (ahora sí, transitivamente) de sus subpartes.

a) Escribir en castellano el invariante de representación.

b,c) Escribir formalmente *b)* el invariante de representación y *c)* la función de abstracción.

Ayuda: Puede suponerse dada una función *hayCamino* : parte d × parte h × dicc(parte, parte) $\longrightarrow$ bool que determine si es posible llegar desde d hasta h mediante una o más aplicaciones de la operación *obtener*.