Lenguajes y Compiladores - Guía 4 - Año 2012

Contenidos: Semántica denotacional del lenguaje imperativo simple

(1) Utilizar la semántica denotacional para demostrar o refutar las siguientes equivalencias:
   (a) $c; \textbf{skip} \equiv c$
   (b) $(\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1); c_2 \equiv \textbf{if } b \textbf{ then } c_0; c_2 \textbf{ else } c_1; c_2$
   (c) $c_2; (\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1) \equiv \textbf{if } b \textbf{ then } c_2; c_0 \textbf{ else } c_2; c_1$
   (d) $\mathsf{x} := \mathsf{x} \equiv \textbf{skip}$
   (e) $\mathsf{x} := \mathsf{y} \ ; \ \mathsf{z} := \mathsf{y} \equiv \mathsf{z} := \mathsf{y} \ ; \ \mathsf{x} := \mathsf{y}$

(2) Utilizar la semántica denotacional para demostrar o refutar las siguientes equivalencias:
   (a) $\textbf{newvar } \mathsf{x} := e \textbf{ in skip} \equiv \textbf{skip}$
   (b) $\textbf{newvar } \mathsf{x} := e \textbf{ in } \mathsf{y} := \mathsf{x} \equiv \mathsf{y} := e$
   (c) $\textbf{newvar } \mathsf{x} := e_1 \textbf{ in } (\textbf{newvar } \mathsf{y} := e_2 \textbf{ in } c) \equiv$
   $\textbf{newvar } \mathsf{y} := e_2 \textbf{ in } (\textbf{newvar } \mathsf{x} := e_1 \textbf{ in } c)$

(3) Enunciar de manera completa y demostrar de manera detallada el Teorema de Renombre para el lenguaje imperativo simple considerando sólo asignación, **skip**, secuencia, **if** y **newvar**. (O sea el lenguaje imperativo simple sin **while**.)

(4) Considere los siguientes programas
   (a) $\textbf{while } \mathsf{x} < 2 \textbf{ do if } \mathsf{x} < 0 \textbf{ then } \mathsf{x} := 0 \textbf{ else } \mathsf{x} := \mathsf{x} + 1$
   (b) $\textbf{while } \mathsf{x} < 2 \textbf{ do if } \mathsf{y} = 0 \textbf{ then } \mathsf{x} := \mathsf{x} + 1 \textbf{ else skip}$
   Determine en cada caso si existe n tal que $F^n\bot = F^{n+1}\bot$. Luego calcule la semántica denotacional.

(5) Suponga que $[\![\textbf{while } b \textbf{ do } c]\!]\sigma \neq \bot$.
   (a) Demuestre que existe $n \geq 0$ tal que $F^n \bot \sigma \neq \bot$.
   (b) Demuestre que si $\sigma' = [\![\textbf{while } b \textbf{ do } c]\!]\sigma$ entonces $\neg[\![b]\!]\sigma'$

(6) Demostrar o refutar las siguientes equivalencias usando semántica denotacional:
   (a) $\textbf{while false do } c \equiv \textbf{skip}$
   (b) $\textbf{while } b \textbf{ do } c \equiv \textbf{while } b \textbf{ do } (c; c)$
   (c) $(\textbf{while } b \textbf{ do } c) \ ; \ \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1 \equiv (\textbf{while } b \textbf{ do } c) \ ; c_1$

(7) Considerar las siguientes definiciones como syntactic sugar del comando $\textbf{for } v := e_0 \textbf{ to } e_1 \textbf{ do } c$:
   (a) $v := e_0; \textbf{while } v \leq e_1 \textbf{ do } c; v := v + 1$.

(b) **newvar** $v := e_0$ **in while** $v \leq e_1$ **do** $c;\ v := v + 1$.

(c) **newvar** $w := e_1$ **in**
newvar $v := e_0$ **in while** $v \leq w$ **do** $c; v := v + 1$

¿Hay alguna que pueda considerarse satisfactoria? Justificar.

(8) Demostrar el caso **while** del Teorema de Renombre.

(9) Definir la sintaxis y la semántica denotacional del comando **repeat** c **until** b satisfaciendo

repeat c **until** $b\ \equiv$
$c;$ **if not** b **then** (**repeat** c **until** b) **else skip**

Luego probar o refutar su equivalencia con el comando
$c;$ **while not** b **do** c.

(10) Usando el Teorema de coincidencia para comandos probar que para todo par de comandos c_0, c_1, si

$$FV\ c_0 \cap FA\ c_1 = FV\ c_1 \cap FA\ c_0 = \emptyset,$$

entonces $[\![c_0; c_1]\!] = [\![c_1; c_0]\!]$.

(11) Considere los siguientes comandos:

$c_0 \doteq$ **newvar** $\mathsf{x} := \mathsf{x} + \mathsf{y}$ **in**
$c;$ **while** $\mathsf{x} > 0$ **do** $\mathsf{y} := \mathsf{y} + 1;\ \mathsf{x} := \mathsf{x} - 1$

$c_1 \doteq$ **newvar** $\mathsf{y} := \mathsf{x} + \mathsf{z}$ **in**
$c;$ **while** $\mathsf{y} > 0$ **do** $\mathsf{z} := \mathsf{z} + 1;\ \mathsf{y} := \mathsf{y} - 1$

Asuma que c es un comando que satisface $(FV\ c) \cap \{\mathsf{x}, \mathsf{y}, \mathsf{z}\} = \emptyset$. Formule la relación que existe entre estos comandos (vistos como funciones que transforman estados), y pruebe tal relación sin calcular semántica.