

Lenguajes y Compiladores. Práctico 4 del 09/04/2018

Objetivos: Comprender las ecuaciones semánticas para el lenguaje imperativo simple. Utilizar semántica denotacional para comparar si dos programas son equivalentes o no. Computar las cadenas de sucesivas ejecuciones de un ciclo, identificar el caso general de la cadena y probar por inducción que es correcto. Definir nuevas construcciones para el lenguaje imperativo, dando sus ecuaciones semánticas. Relacionar valores del dominio semántico $\Sigma \rightarrow \Sigma_{\perp}$ con la denotación de programas. Conocer los teoremas de sustitución, renombre y coincidencia y utilizarlos para concluir si dos programas son equivalentes o no.

Recordar que utilizamos tipografía sans-serif (como en x) para variables concretas y tipografía serif (como en e) para meta-variables.

Repaso. Decida si los siguientes argumentos son correctos:

- (1) Sea D un dominio y $f: D \rightarrow D$. Si f es continua, entonces f tiene un menor punto fijo.
- (2) Sea D un dominio y $f: D \rightarrow D$. Si f es continua, entonces \perp es el menor punto fijo de f .
- (3) Sea D un dominio y sea $f_1 \sqsubseteq f_2 \sqsubseteq f_3 \sqsubseteq \dots$ una cadena en $D \rightarrow D$. Si el supremo de la cadena pertenece a la cadena, entonces existe un índice k tal que $f_k = f_{k+1} = f_{k+2}$.

Ejercicios.

- (1) Utilizar la semántica denotacional para demostrar o refutar las siguientes equivalencias:
 - (a) $c; \mathbf{skip} \equiv c$
 - (b) $c_1; (c_2; c_3) \equiv (c_1; c_2); c_3$
 - (c) $(\mathbf{if } b \mathbf{ then } c_0 \mathbf{ else } c_1); c_2 \equiv \mathbf{if } b \mathbf{ then } c_0; c_2 \mathbf{ else } c_1; c_2$
 - (d) $c_2; (\mathbf{if } b \mathbf{ then } c_0 \mathbf{ else } c_1) \equiv \mathbf{if } b \mathbf{ then } c_2; c_0 \mathbf{ else } c_2; c_1$
 - (e) $x := y; z := w \equiv z := w; x := y$
- (2) Utilizar la semántica denotacional para demostrar o refutar las siguientes equivalencias:
 - (a) $\mathbf{newvar } x := e \mathbf{ in skip} \equiv \mathbf{skip}$
 - (b) $\mathbf{newvar } x := e \mathbf{ in } y := x \equiv y := e$
 - (c) $\mathbf{newvar } x := e_1 \mathbf{ in } (\mathbf{newvar } y := e_2 \mathbf{ in } c) \equiv \mathbf{newvar } y := e_2 \mathbf{ in } (\mathbf{newvar } x := e_1 \mathbf{ in } c)$
- (3) Teniendo en cuenta los ejercicios anteriores, discuta en grupo las siguientes afirmaciones:
 - (a) El parser puede eliminar toda ocurrencia de **skip**.
 - (b) El parser puede elegir inclinar las secuencias de más de dos comandos hacia la derecha o hacia la izquierda.
- (4) Considere el comando **while true do** $x := x - 1$
 - (a) Dar la función F que define su semántica. Calcular la expresión más sencilla que pueda para F .
 - (b) Existe algún n tal que $F^n \perp_{\Sigma \rightarrow \Sigma_{\perp}}$ no sea idénticamente \perp ?
 - (c) Considere la cadena en $\Sigma \rightarrow \Sigma_{\perp}$ dada por

$$\omega_i \sigma = \begin{cases} \sigma & \text{si } 0 \leq \sigma x \leq i \\ \perp & \text{caso contrario} \end{cases}$$

Es sabido que la continuidad de F garantiza la igualdad $F(\bigsqcup \omega_i) = \bigsqcup F\omega_i$. Compruebe la misma calculando cada miembro de la igualdad para el caso de la cadena dada.

- (5) Calcule la semántica denotacional de los siguientes comandos:
- while** $x < 2$ **do** **if** $x < 0$ **then** $x := 0$ **else** $x := x + 1$
 - while** $x < 2$ **do** **if** $y = 0$ **then** $x := x + 1$ **else** **skip**
- (6) Suponga que $\llbracket \mathbf{while} \ b \ \mathbf{do} \ c \rrbracket \sigma \neq \perp$.
- Demuestre que existe $n \geq 0$ tal que $F^n \perp \sigma \neq \perp$.
 - Demuestre que si $\sigma' = \llbracket \mathbf{while} \ b \ \mathbf{do} \ c \rrbracket \sigma$ entonces $\neg \llbracket b \rrbracket \sigma'$
- (7) Demostrar o refutar las siguientes equivalencias usando semántica denotacional:
- while** **false** **do** $c \equiv \mathbf{skip}$
 - while** b **do** $c \equiv \mathbf{while} \ b \ \mathbf{do} \ (c; c)$
 - $(\mathbf{while} \ b \ \mathbf{do} \ c) ; \mathbf{if} \ b \ \mathbf{then} \ c_0 \ \mathbf{else} \ c_1 \equiv (\mathbf{while} \ b \ \mathbf{do} \ c) ; c_1$
- (8) Considerar las siguientes definiciones como syntactic sugar del comando **for** $v := e_0$ **to** e_1 **do** c :
- $v := e_0; \mathbf{while} \ v \leq e_1 \ \mathbf{do} \ c; v := v + 1$.
 - newvar** $v := e_0$ **in** $\mathbf{while} \ v \leq e_1 \ \mathbf{do} \ c; v := v + 1$.
 - newvar** $w := e_1$ **in** **newvar** $v := e_0$ **in** $\mathbf{while} \ v \leq w \ \mathbf{do} \ c; v := v + 1$
- ¿Hay alguna que pueda considerarse satisfactoria? Justificar.
- (9) Enunciar de manera completa el teorema de coincidencia y demostrar el caso **while**.
- (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 $\llbracket c_0; c_1 \rrbracket = \llbracket c_1; c_0 \rrbracket$.
- (11) Considere los siguientes comandos:
- $$c_0 \doteq \mathbf{newvar} \ x := x + y \ \mathbf{in} \\ c; \mathbf{while} \ x > 0 \ \mathbf{do} \ y := y + 1; \ x := x - 1$$
- $$c_1 \doteq \mathbf{newvar} \ y := x + z \ \mathbf{in} \\ c; \mathbf{while} \ y > 0 \ \mathbf{do} \ z := z + 1; \ y := y - 1$$
- Asuma que c es un comando que satisface $(FV \ c) \cap \{x, y, z\} = \emptyset$. Formule la relación que existe entre estos comandos (vistos como funciones que transforman estados), y pruebe tal relación sin calcular semántica.
- (12) Considere la siguiente cadena en $\Sigma \rightarrow \Sigma_{\perp}$. Decida si existe un programa cuya semántica sea el supremo de la cadena. $f_i \sigma = \begin{cases} \sigma & \text{si } \sigma \times \leq \sigma y \\ \perp & \text{en caso contrario} \end{cases}$