

Lenguajes y Compiladores. Práctico 6 del 17/04/2019

Objetivos. Comprender las construcciones necesarias para dar semántica a output e input. Entender el orden subyacente a productos de dominios y uniones disjuntas de predomios. Poder determinar si dado un elemento d del dominio semántico existe un programa cuya denotación sea d .

Repaso.

- (1) Decida si la función $H: (\Sigma \rightarrow \Sigma'_{\perp}) \rightarrow (\Sigma \rightarrow \Sigma'_{\perp})$ es monótona y si es continua.

$$H f \sigma = \begin{cases} \sigma & \text{si } f = \perp \\ \perp & \text{en caso contrario} \end{cases}$$

- (2) Considere la función $G: (\Sigma \rightarrow \Sigma'_{\perp}) \rightarrow (\Sigma \rightarrow \Sigma'_{\perp})$ definida por

$$G f \sigma = \begin{cases} \perp & \text{si } f \text{ es parcial} \\ \sigma & \text{en caso contrario} \end{cases}$$

- (a) Decida si G es continua o no.
(b) Decida si existe un comando **while** b **do** c , tal que $G = F$, donde F es el funcional asociado al **while**.

Ejercicios.

- (1) Considerando el lenguaje con fallas y output, de un programa para cada posible comportamiento:
- cantidad finita de output y luego divergencia,
 - cantidad finita de output y luego falla,
 - cantidad finita de output y luego terminación,
 - cantidad infinita de output.
- (2) Dado el programa **while** $x > 0$ **do** $!x; c$, calcule su semántica denotacional, considerando los casos
- $c \equiv \text{if } x > 0 \text{ then skip else fail}$
 - $c \equiv \text{if } x > 0 \text{ then fail else skip}$
- (3) Demostrar o refutar las siguientes equivalencias usando semántica denotacional:
- $?x; ?y \equiv ?y; ?x$.
 - $?x; z := x \equiv ?z$.
 - newvar** $x := e$ **in** $(?x; z := x) \equiv ?z$.
- (4) Sea c un programa que no incluya fallas, outputs, ni inputs tal que $\{x, y\} \cap FV(c) = \emptyset$. Determine si es válida la siguiente igualdad:

$$?x; c; !x \equiv ?y; c; !y$$

- (5) Describa mediante un diagrama de Hasse las relaciones de orden que se establecen entre los siguientes elementos de Ω :

- $\iota_{in}(\lambda n. \iota_{out}(n, \perp))$

- $\iota_{out}(0, \perp)$,
- $\iota_{in}(\lambda n. \perp)$,
- $\iota_{in}(f)$, donde $f n = \begin{cases} \perp & \text{si } n < 0 \\ \iota_{out}(n, \perp) & \text{caso contrario} \end{cases}$

(6) Dé un programa, y justifique su elección, cuya semántica sea el supremo de la cadena:

$$w_0 = \perp, \quad w_{i+1} = \iota_{in}(\lambda n. \iota_{out}(n, w_i))$$

(7) Considere los programas de la forma **while true do** ($?x; c$). La cadena $F^i \perp \sigma$ de la semántica del **while**, ¿será siempre una cadena interesante en Ω ? Justifique su respuesta.

(8) Dado el programa $P \equiv$

```
newvar x := x + 1 in
  while x > 0 do ?x; if y > 0 then fail else !x
```

(a) Calcular la semántica denotacional de P , en un estado σ tal que $\sigma y > 0$.

(b) Considere el caso $\sigma y \leq 0$. Calcule $F \perp$ y $F^2 \perp$. Puede dar una expresión general para $\mathbf{Y}_{\Sigma \rightarrow \Sigma \perp} F$.

(9) Demostrar o refutar las siguientes equivalencias usando semántica denotacional

(a) $?x; \mathbf{while} \ b \ \mathbf{do} \ !x; ?x \ \mathbf{od}; !x \equiv \mathbf{while} \ b \ \mathbf{do} \ ?x; !x \ \mathbf{od}$.

(b) $?x; \mathbf{while} \ b \ \mathbf{do} \ !x; ?x \ \mathbf{od}; !x \equiv ?x; !x; \mathbf{while} \ b \ \mathbf{do} \ ?x; !x \ \mathbf{od}$.