

Lenguajes y Compiladores. Práctico 5 del 10/04/2019

Objetivos: Comprender las modificaciones necesarias que se deben realizar al dominio semántico para poder extender el lenguaje con un mecanismo de fallas y manejo de fallas. Poder construir trazas (i.e. derivaciones) de cómo se ejecuta un programa utilizando la semántica operacional. Poder relacionar la semántica denotacional con la operacional probando su equivalencia.

Recordar que los símbolos en cursiva representan metavariables. Por ejemplo, si en un ejercicio involucra los símbolos v y w , debe considerarse el caso en que ambas metavariables denoten la misma variable concreta.

Repaso.

- (1) Escriba un teorema de coincidencia para la semántica operacional.
- (2) Después de terminar con los ejercicios, pruebe el teorema de coincidencia (sin hacer ninguna inducción).

Ejercicios.

- (1) Dado el programa $P \equiv$

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

Caracterice (sin calcular la semántica) los estados σ en los cuales P se comporta como **skip**.

- (2) Demostrar o refutar las siguientes equivalencias usando semántica denotacional:
 - (a) $c; \mathbf{while\ true\ do\ skip} \equiv \mathbf{while\ true\ do\ skip}$.
 - (b) $c; \mathbf{fail} \equiv \mathbf{fail}$.
 - (c) $\mathbf{newvar\ } v := e \mathbf{\ in\ } v := v + 1; \mathbf{fail} \equiv$
 $\mathbf{newvar\ } w := e \mathbf{\ in\ } w := w + 1; \mathbf{fail}$.
 - (d) $\mathbf{while\ } b \mathbf{\ do\ fail} \equiv \mathbf{if\ } b \mathbf{\ then\ fail\ else\ skip}$.
 - (e) $x := 0; \mathbf{catch\ } x := 1 \mathbf{\ in\ while\ } x < 1 \mathbf{\ do\ fail} \equiv$
 $x := 0; \mathbf{while\ } x < 1 \mathbf{\ do\ catch\ } x := 1 \mathbf{\ in\ fail}$

Considere el punto (a) para el lenguaje sin fallas: ¿la respuesta es la misma que para el lenguaje con fallas?

- (3) Computar la semántica operacional en estados σ tales que $\sigma x = 2, \sigma y = 1$. de los siguientes programas
 - (1) $y := x + y; \mathbf{if\ } y > 0 \mathbf{\ then\ } x := x - 1 \mathbf{\ else\ skip}$
 - (2) $\mathbf{while\ } x > 0 \mathbf{\ do}$
 $y := x + y; \mathbf{if\ } y > 0 \mathbf{\ then\ } x := x - 1 \mathbf{\ else\ skip}$

- (4) Computar la semántica operacional del siguiente programa en un estado arbitrario

```
newvar x := 2 in
  while x > 0 do
    y := x + y; if y > 0 then x := x - 1 else skip
```

- (5) Pruebe:

(a) Si $\langle c_0, \sigma \rangle \rightarrow^* \sigma'$, entonces $\langle c_0; c_1, \sigma \rangle \rightarrow^* \langle c_1, \sigma' \rangle$

(b) Si $\langle c, [\sigma|x : \llbracket e \rrbracket \sigma] \rangle \rightarrow^* \sigma'$, entonces

$$\langle \mathbf{newvar} \ x := e \ \mathbf{in} \ c, \sigma \rangle \rightarrow^* [\sigma'|x : \sigma x].$$

(c) Si $\langle c, [\sigma|x : \llbracket e \rrbracket \sigma] \rangle \rightarrow^* \langle c', \sigma' \rangle$, entonces

$$\langle \mathbf{newvar} \ x := e \ \mathbf{in} \ c, \sigma \rangle \rightarrow^* \langle \mathbf{newvar} \ x := \sigma'x \ \mathbf{in} \ c', [\sigma'|x : \sigma x] \rangle$$

- (6) Demostrar la corrección de la semántica operacional one-step (incluyendo **fail**) respecto de la semántica denotacional, demostrando simultáneamente

(a) $\langle c, \sigma \rangle \rightarrow \sigma' \implies \llbracket c \rrbracket \sigma = \sigma'$.

(b) $\langle c, \sigma \rangle \rightarrow \langle \mathbf{abort}, \sigma' \rangle \implies \llbracket c \rrbracket \sigma = \langle \mathbf{abort}, \sigma' \rangle$.

(c) $\langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle \implies \llbracket c \rrbracket \sigma = \llbracket c' \rrbracket \sigma'$.

Use inducción en la derivación.

- (7) Demostrar simultáneamente:

(a) $\langle c, \sigma \rangle \rightarrow^* \sigma' \iff \llbracket c \rrbracket \sigma = \sigma'$.

(b) $\langle c, \sigma \rangle \rightarrow^* \langle \mathbf{abort}, \sigma' \rangle \iff \llbracket c \rrbracket \sigma = \langle \mathbf{abort}, \sigma' \rangle$.

Use inducción en la estructura de c . Tendrá que usar los resultados probados en el ejercicio 5. El caso **while** requiere a su vez una inducción en el mínimo n tal que $F^n(\perp)\sigma \neq \perp$.

- (8) Demostrar la equivalencia entre la semántica denotacional y la operacional del lenguaje imperativo simple incluyendo **fail**. Debe salir inmediatamente de los dos ejercicios anteriores.

- (9) (opcional) Proponga una semántica operacional big-step $\Rightarrow \subseteq \Gamma_n \times \Gamma_t$ para el lenguaje LIS, tal que

$$\langle c, \sigma \rangle \Rightarrow \sigma' \quad \text{sii} \quad \langle c, \sigma \rangle \rightarrow^* \sigma' .$$

Discutan cómo probarían la equivalencia; prueben la equivalencia en grupo (quienes hacen una dirección, no hacen la otra; y viceversa)

Laboratorio 4

Implementar la semántica denotacional para el lenguaje imperativo simple con fallas (LIS + Fallas) extendiendo el lenguaje imperativo extremadamente simple (LIES) con `while`, `newvar`, `fail` y `catch`. El enunciado completo y el archivo a completar se puede descargar de la wiki de la materia: https://wiki.cs.famaf.unc.edu.ar/doku.php?id=compiladores:main#lab_2019.