

## Semantik von Programmiersprachen – SS 2010

<http://pp.info.uni-karlsruhe.de/lehre/SS2010/semantik>

Blatt 7: Prozeduren

Besprechung: 01.06.2010

### 1. Welche der folgenden Aussagen sind richtig, welche falsch? (H)

- (a)  $[(p, i := i * 1; \text{call } p)] \vdash \langle \text{call } p, \sigma \rangle \Downarrow \sigma$
- (b)  $[(p, \text{skip}), \text{call } q]$  ist ein Programm der Sprache  $\text{While}_{PROC}$ .
- (c)  $\langle \text{skip}, \sigma \rangle$  ist die einzige blockierte Konfiguration der Small-Step-Semantik für  $\text{While}_{PROC}$ .
- (d) Wenn  $P \vdash \langle c, \sigma \rangle \xrightarrow{\infty}_1$ , dann enthält  $c$  eine `while`-Schleife.
- (e) Sei  $P \equiv [(p, y, x := 4; \text{result} := 5 * y)]$ ,  $E_0 \equiv [x \mapsto 0, y \mapsto 1]$  und  $s \equiv [\text{next} \mapsto 2]$ .  
Wenn  $P, E_0, E_0 \vdash \langle \{ \text{var } x = 3; y \leftarrow \text{call } p(x); y := y + x \}, s \rangle \Downarrow s'$ ,  
dann  $s'(E_0(x)) = 4$  und  $s'(E_0(y)) = 18$ .
- (f) Wenn  $P, E_0, E \vdash \langle c, s \rangle \Downarrow s'$ , dann  $s'(\text{next}) = s(\text{next})$ .

### 2. Small-Step-Semantik für Prozeduren mit Parametern (H)

Für Prozeduren mit einem Parameter ( $\text{While}_{PROC P}$ ) gibt es folgende Vorschläge für eine Aufrufregel der Small-Step-Semantik:

- (a) 
$$\frac{(p, x, c) \in P}{P \vdash \langle y \leftarrow \text{call } p(a), \sigma \rangle \rightarrow_1 \langle x := a; c; y := \text{result}, \sigma \rangle}$$
- (b) 
$$\frac{(p, x, c) \in P}{P \vdash \langle y \leftarrow \text{call } p(a), \sigma \rangle \rightarrow_1 \langle \{ \text{var } \text{result} = 0; \{ \text{var } x = a; c \}; y := \text{result} \}, \sigma \rangle}$$
- (c) 
$$\frac{(p, x, c) \in P}{P \vdash \langle y \leftarrow \text{call } p(a), \sigma \rangle \rightarrow_1 \langle \{ \text{var } x = a; \{ \text{var } \text{result} = 0; c; y := \text{result} \} \}, \sigma \rangle}$$
- (d) 
$$\frac{(p, x, c) \in P \quad z \text{ nicht verwendet in } P \text{ und } a}{P \vdash \langle y \leftarrow \text{call } p(a), \sigma \rangle \rightarrow_1 \langle \{ \text{var } z = 0; \{ \text{var } x = a; \{ \text{var } \text{result} = 0; c; z := \text{result} \} \}; y := z \}, \sigma \rangle}$$

Untersuchen Sie, in wie weit diese Regeln call-by-value Prozedurparameter und Rückgabewerte korrekt abbilden. Begründen Sie Ihre Antwort; ggf. mit Beispielprogrammen, die entsprechende Defizite aufzeigen. Welche dieser Regeln modellieren statische Variablenbindung?

### 3. Call by reference (Ü)

Die in der Vorlesung vorgestellte Big-Step-Semantik für Prozeduren mit einem Parameter wertet den übergebenen Parameter beim Aufruf aus und übergibt nur den Wert an die aufgerufene Prozedur. Insbesondere bleibt der Wert einer Variablen, die als Parameter übergeben wird, im

aufrufenden Kontext unverändert. Diese Parameterübergabeart heißt *call by value*. Daneben gibt es auch noch *call by reference*, bei der Änderungen am Parameterwert in der aufgerufenen Prozedur auch nach dem Ende des Aufrufs in der aufrufenden Prozedur sichtbar sind. Passen Sie in dieser Aufgabe die Big-Step-Semantik von  $\text{While}_{PROCP}$  entsprechend an:

- (a) Passen Sie die Syntax-Definition von  $\text{While}_{PROCP}$  an, so dass nur noch Variablen als Parameter verwendet können.
- (b) Ändern Sie die Big-Step-Semantik so, dass Parameter immer als call by reference übergeben werden.
- (c) Ändern Sie die Regeln erneut so, dass auch die Rückgabewertvariable mit call by reference übergeben wird.
- (d) Finden Sie ein Programm, bei dem sich das Verhalten in beiden Varianten von call by reference und call by value unterscheidet. Belegen Sie dies durch die Ableitungsbäume.