Solution – HW 3: Language Syntax, Semantics, Errors, Nondeterminism

CS 536: Science of Programming, Fall 2019

9/28, 9/29 pp. 1,2; 9/30 pp.1,2, 10/2 p.1

Lecture 5: Language Syntax/Operational Semantics

1. Here is one (of many possible) solutions: \( x := 1; j := 0; \text{while } j \leq m \text{ do } j := j + 1; x := x + 1; x := x + y \) od; \( j := j + 1 \) [10/2]

2. (Evaluate \( S \equiv \text{if } x > 0 \text{ then } x := x + 2 \times y; y := 3 \times y \text{ fi} \))

2a. \( \langle S, \{x = 2, y = 6\} \rangle \)

\( \rightarrow \langle \text{if } x > 0 \text{ then } x := x + 2 \times y; y := 3 \times y \text{ fi}, \{x = 2, y = 6\} \rangle \) // optional step (expanding \( S \))

\( \rightarrow \langle x := x + 2 \times y; y := 3 \times y, \{x = 2, y = 6\} \rangle \) // jump to start of true branch

\( \rightarrow \langle y := 3 \times y, \{x = 14, y = 6\} \rangle \) [9/29] // evaluate first assignment

\( \rightarrow \langle E, \{x = 14, y = 18\} \rangle \). // evaluate second assignment

Note 1: The comments weren't required. Note 2: The first step, where we expand \( S \), is an \( = \) not an \( \rightarrow \) because replacing \( S \) by the text it stands for is not a semantic operation. [9/29 rephrased]

2b. \( \langle S, \{x = -2, y = 8\} \rangle \rightarrow \langle \text{skip}, \{x = -2, y = 8\} \rangle \rightarrow \langle E, \{x = -2, y = 8\} \rangle \) [9/29]

Note the \( \text{skip} \) is required because an if-then statement is just an abbreviation for an if statement with \( \text{else skip} \).

3. (Evaluate loop) We have \( \sigma_0 = \{i = 1, x = 1, n = 5\} \) and \( W = \text{while } i \neq n \text{ do } S \text{ od} \) where \( S = i := i + 1; x := x + i \times i \) . One possible (kind of long) answer is

\( \langle W, \sigma_0 \rangle = \langle \text{while } i \neq n \text{ do } S \text{ od}, \sigma_0 \rangle \) // definition of \( W \)

\( \rightarrow \langle S; W, \sigma_0 \rangle \) // because \( \sigma_0 \models i \neq n \), the loop test

\( = \langle i := i + 1; x := x + i \times i; W, \sigma_0 \rangle \) // definition of \( S \)

\( \rightarrow \langle x := x + i \times i; W, \sigma_0[i \mapsto 2] \rangle \) // where \( \sigma_1 = \sigma_0[i \mapsto 2][x \mapsto 5] \)

\( \rightarrow \langle S; W, \sigma_1 \rangle \) // because \( \sigma_1 \models i \neq n \)

\( \rightarrow^2 \langle W, \sigma_2 \rangle \) // where \( \sigma_2 = \sigma_1[i \mapsto 3][x \mapsto 14] \models i \neq n \)

\( \rightarrow^3 \langle W, \sigma_3 \rangle \) // where \( \sigma_3 = \sigma_2[i \mapsto 4][x \mapsto 30] \models i \neq n \)

\( \rightarrow^2 \langle W, \sigma_4 \rangle \) // where \( \sigma_4 = \sigma_3[i \mapsto 5][x \mapsto 55] \models i \neq n \)

\( \rightarrow^2 \langle E, \sigma_5 \rangle \) // because \( \sigma_5 \neq i \neq n \).
Lecture 6: Denotational Semantics, Runtime Errs, Sequential Nondeterminism pt. 1

4. (Denotational semantics for Problem 2) [9/29]
   a. Since \( \langle S, \{x = 2, y = 6\} \rangle \rightarrow^* \langle E, \{x = 14, y = 18\} \rangle \), we have \( M(S, \{x = 2, y = 6\}) = \{\{x = 14, y = 18\}\} \).
   b. Since \( \langle S, \{x = -2, y = 8\} \rangle \rightarrow^* \langle E, \{x = -2, y = 8\} \rangle \), we have \( M(S, \{x = -2, y = 8\}) = \{\{x = -2, y = 8\}\} \).

5. (Diverging loop) We have \( W = \textbf{while} \ i \neq n \ \textbf{do} \ i := i+1; \ x := x+i \ \textbf{od} \). If we start with \( \sigma(i) < \sigma(n) \), the loop diverges; if we start with \( \sigma(i) \geq \sigma(n) \), we terminate, so the set we want is \( \{\sigma \in \Sigma \mid \sigma(i) > \sigma(n)\} \). [9/29]

6. (Deterministic program’s final state)
   a. If \( S \) is deterministic, then \( M(S, \sigma) \) is a singleton set \( \{\bot\} \), so for any program \( T \), \( \langle S; T, \sigma \rangle \rightarrow^* \langle E, \bot \rangle \). In English, since \( S \) doesn’t terminate, we can’t run \( T \), so \( S; T \) doesn’t terminate.
   b. Either \( \tau \in \Sigma \) or \( \tau = \bot \). Any member of \( \Sigma \) satisfies true so to get \( \tau \neq \top \), we must have \( \tau = \bot \).

Lecture 7: Sequential Nondeterminism pt. 2

7. (Nondeterministic program) [9/30; answer rewritten]
   - The set \( M(S, \sigma) \models \varphi \) iff every \( \tau \in M(S, \sigma) \) satisfies \( \varphi \).
   - Similarly, the set \( M(S, \sigma) \models \neg \varphi \) if every \( \tau \) in \( M(S, \sigma) \) satisfies \( \neg \varphi \).
   - But we’re given that \( M(S, \sigma) \models \varphi \) and \( M(S, \sigma) \models \neg \varphi \).
   - Then since \( M(S, \sigma) \models \varphi \), at least one \( \tau \) in \( M(S, \sigma) \) doesn’t satisfy \( \varphi \).
   - But we’re also given that \( \bot \not\in M(S, \sigma) \), so if \( \tau \) doesn’t satisfy \( \varphi \), it must satisfy \( \neg \varphi \).
   - Similarly, since \( M(S, \sigma) \models \neg \varphi \), we have that \( M(S, \sigma) \) contains at least one \( \tau \) that \( \models \varphi \).
   - So to get \( \bot \not\in M(S, \sigma) \), \( M(S, \sigma) \not\models \varphi \), and \( M(S, \sigma) \not\models \neg \varphi \), we must have at least two states in \( M(S, \sigma) \); one that \( \models \varphi \) and one that \( \not\models \neg \varphi \).
     - For a concrete example, if \( S = \textbf{if} \ T \rightarrow x := T \ \square \ T \rightarrow x := \textbf{F} \ \textbf{f} \ \textbf{i} \), then \( M(S, \emptyset) = \{\{x = T\}\} \).
     - \( \{x = F\} \) \not\models x and \( \not\models \neg x \).

8. (Nondeterministic loop that can both diverge and terminate) Basically, we need a loop where one guard lets us terminate and the other guard can cause divergence. A simple example of a \( W \) that does this is \( \textbf{do} \ x = 0 \rightarrow \textbf{sk} \{\square \} x = 0 \rightarrow x := -1 \ \textbf{od} \). Running this loop starting in state \( x = 0 \) diverges if we always choose the first guard and terminates in state \( \{x = -1\} \) if we ever choose the second guard.
a. Operationally, we diverge using execution path \( \langle W, \{x = 0\} \rangle \rightarrow \langle \textbf{skip}; W, \{x = 0\} \rangle \rightarrow \langle W, \{x = 0\} \rangle \rightarrow^2 \langle W, \{x = 0\} \rangle \rightarrow^2 \langle W, \{x = 0\} \rangle \rightarrow^2 \ldots \).

b. We terminate using path \( \langle W, \{x = 0\} \rangle \rightarrow \langle x := -1; W, \{x = 0\} \rangle \rightarrow \langle W, \{x = -1\} \rangle \rightarrow \langle E, \{x = -1\} \rangle \).

(This is the shortest path to termination: We can begin with chains of \( \langle W, \{x = 0\} \rangle \rightarrow^2 \langle W, \{x = 0\} \rangle \rightarrow^2 \ldots \) (as in part (a)) and join with the path above to get a loop that does more iterations before terminating.)