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; while j ≤ m do j := j+1; x := x+1; x := x+y od; j := j+1` [10/2]

2. (Evaluate $S \equiv$ **if** `x > 0` **then** `x := x+2*y; y := 3*y` **fi**)

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

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

$\rightarrow \langle$`x := x+2*y; y := 3*y`, $\{x = 2, y = 6\}\rangle$ // jump to start of true branch

$\rightarrow \langle$`y := 3*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. $\langle S, \{x = -2, y = 8\}\rangle \rightarrow \langle$**skip**$, \{x = -2, y = 8\}\rangle \rightarrow \langle E, \{x = -2, y = 8\}\rangle$ [9/29]

Note the **skip** is required because an if-then statement is just an abbreviation for an if statement with **else skip**.

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

$\langle W, \sigma_0\rangle = \langle$**while** $i \neq n$ **do** S **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*i;` $W, \sigma_0\rangle$ // definition of S

$\rightarrow \langle$`x := x+i*i;` $W, \sigma_0[i \mapsto 2]\rangle$

$\rightarrow \langle W, \sigma_1\rangle$ // 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 $\sigma_2 = \sigma_1[i \mapsto 3][x \mapsto 14] \models i \neq n$

$\rightarrow^2 \langle W, \sigma_3\rangle$ // where $\sigma_3 = \sigma_2[i \mapsto 4][x \mapsto 30] \models i \neq n$

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

$\rightarrow^2 \langle E, \sigma_4\rangle$ // because $\sigma_4 \not\models i \neq n$.

Lecture 6: Denotational Semantics, Runtime Errs, Sequential Nondeterminism pt. 1

4. (Denotational semantics for Problem 2) [9/29]

   4a. Since $\langle S, \{\mathtt{x} = 2, \mathtt{y} = 6\}\rangle \to^* \langle E, \{\mathtt{x} = 14, \mathtt{y} = 18\}\rangle$, we have $M(S, \{\mathtt{x} = 2, \mathtt{y} = 6\}) = \{\{\mathtt{x} = 14, \mathtt{y} = 18\}\}$.

   4b. Since $\langle S, \{\mathtt{x} = -2, \mathtt{y} = 8\}\rangle \to^* \langle E, \{\mathtt{x} = -2, \mathtt{y} = 8\}\rangle$, we have $M(S, \{\mathtt{x} = -2, \mathtt{y} = 8\}) = \{\{\mathtt{x} = -2, \mathtt{y} = 8\}\}$.

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

6. (Deterministic program's final state)

   a. If S is deterministic, then $M(S, \sigma)$ is a singleton set $\{\perp\}$, so for any program T, $\langle S;\ T, \sigma\rangle \to^* \langle E, \perp\rangle$. In English, since S doesn't terminate, we can't run T, so $S;\ T$ doesn't terminate.

   b. Either $\tau \in \Sigma$ or $\tau = \perp$. Any member of Σ satisfies true so to get $\tau \not\models \mathtt{T}$, we must have $\tau = \perp$.

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 φ.
   - Similarly, the set $M(S, \sigma) \models \neg\varphi$ if every τ in $M(S, \sigma)$ satisfies $\neg\varphi$.
   - But we're given that $M(S, \sigma) \not\models \varphi$ and $M(S, \sigma) \not\models \neg\varphi$.
   - Then since $M(S, \sigma) \not\models \varphi$, at least one τ in $M(S, \sigma)$ doesn't satisfy φ.
   - But we're also given that $\perp \notin M(S, \sigma)$, so if τ doesn't satisfy φ, it must satisfy $\neg\varphi$.
   - Similarly, since $M(S, \sigma) \not\models \neg\varphi$, we have that $M(S, \sigma)$ contains at least one τ that $\models \varphi$.
   - So to get $\perp \notin M(S, \sigma)$, $M(S, \sigma) \not\models \varphi$, 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 $\models \neg\varphi$,
     - For a concrete example, if $S \equiv$ **if** $\mathtt{T} \to \mathtt{x} := \mathtt{T}\ \square\ \mathtt{T} \to \mathtt{x} := \mathtt{F}$ **fi**, then $M(S, \varnothing) = \{\ \{\mathtt{x} = T\}, \{\mathtt{x} = F\}\ \} \not\models \mathtt{x}$ and $\not\models \neg\mathtt{x}$.

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 **do** $\mathtt{x} \geq 0 \to \mathtt{skip}\ \square\ \mathtt{x} \geq 0 \to \mathtt{x} := -1$ **od**. Running this loop starting in state $\mathtt{x} = 0$ diverges if we always choose the first guard and terminates in state $\{\mathtt{x} = -1\}$ if we ever choose the second guard.

a. Operationally, we diverge using execution path $\langle W, \{\mathtt{x} = 0\}\rangle \rightarrow \langle \mathbf{skip;}\ W, \{\mathtt{x} = 0\}\rangle \rightarrow \langle W, \{\mathtt{x} = 0\}\rangle \rightarrow^2 \langle W, \{\mathtt{x} = 0\}\rangle \rightarrow^2 \langle W, \{\mathtt{x} = 0\}\rangle \rightarrow^2 \dots$.

b. We terminate using path $\langle W, \{\mathtt{x} = 0\}\rangle \rightarrow \langle \mathtt{x := -1;}\ W, \{\mathtt{x} = 0\}\rangle \rightarrow \langle W, \{\mathtt{x} = -1\}\rangle \rightarrow \langle E, \{\mathtt{x} = -1\}\rangle$. (This is the shortest path to termination: We can begin with chains of $\langle W, \{\mathtt{x} = 0\}\rangle \rightarrow^2 \langle W, \{\mathtt{x} = 0\}\rangle \rightarrow^2 \dots$ (as in part (a)) and join with the path above to get a loop that does more iterations before terminating.)

