Programs, Semantics, Errors, Nondeterminism

CS 536: Science of Programming, Spring 2018

Due Wed Feb 14, 11:59 pm — No Late Assignments

2/15: solved

A. Instructions

- You can work together in groups of ≤ 4. Submit your work on Blackboard. Submit one copy, under the name of one person in the group (doesn't matter who). Include the names and A-IDs of everyone in the group (including the submitter) inside that copy.
- Solution will be posted Thu, Feb 15 to maximize the time you have to study it before Exam 1 (Mon, Feb 19).

B. Why?

- Our simple programming language is a model for the kind of constructs seen in actual languages.
- Our programs stand for state transformers.

A. Why?

- Program semantics can be understood using operational, step-by-step evaluation of program / state snapshots or using denotational, state-transformation semantics.
- Runtime errors cause failure of normal program execution.
- Nondeterminism can help us develop programs without worrying about overlapping cases.

B. Outcomes

After this homework, you should be able to

- Translate simple programs into our language and calculate their meaning;
- Write simple predicate functions.
- Use operational semantics to describe step-by-step execution of programs in our language.
- Give the denotational semantics of a program in a state.
- Say when and how evaluation of an expression or program fails due to a runtime error.
- Evaluate a nondeterministic **`if-fi`** and **`do-od`**.

C. Problems [100 points total]

Part 1: Programs and Operational Semantics

For Problems 1 – 3, translate the given C-like programs into our programming language.

1. [6 points] `for (j = m, x = 1; j >= 1; ) x *= y[--j];`
2. [6 points] `j = m; x = 1; while (--j >= 0) x *= y[j];`
3. [6 points] `x = 1; j = 0; while (j++ < m) x *= y[j];`

4. [12 = 2 * 6 points] Let $\sigma = \{(\mathtt{x}, \alpha), (\mathtt{y}, \beta)\}$ (or in ASCII, let σ = {(x, α), (y, β)}). Evaluate each $\langle S, \sigma\rangle$ below to completion.

   a. $S \equiv$ `t := x ; x := y; y := t`

   b. $S \equiv$ **if** `x < 0` **then** `y := -x` **else** `y := x` **fi**

5. [13 points] Let $S \equiv$ `s := 0;` **while** $\mathtt{n} \geq 0$ **do** S_1 **od** where the loop body $S_1 \equiv$ `s := s+n; n := n-1`.

   a. [3 points] If $\sigma(\mathtt{n}) < 0$, what are the operational semantics of S in σ?

   b. [4 points] Give the operational semantics of the loop body S_1 in an arbitrary state τ.

   c. [6 points] Let $\sigma(\mathtt{n}) = 3$. Write out the complete execution sequence for $\langle S, \sigma\rangle$. Use multi-step execution as in Example 10 in Lecture 6.

6. [7 points] First study this definition of a predicate function $\mathtt{Match}(\mathtt{b}, \mathtt{b}', \mathtt{x}, \mathtt{y}, \mathtt{n})$:

$$\mathtt{Match}(\mathtt{b}, \mathtt{b}', \mathtt{x}, \mathtt{y}, \mathtt{n}) \equiv \forall\, \mathtt{k} \,.\, 0 \leq \mathtt{k} < \mathtt{n} \rightarrow \mathtt{b}[\mathtt{x+k}] = \mathtt{b}'[\mathtt{y+k}]$$

Then $\sigma \vDash \mathtt{Match}(\mathtt{b}, \mathtt{b}', \mathtt{x}, \mathtt{y}, \mathtt{n})$ when in σ, the n elements $\mathtt{b}[\mathtt{x..x+n-1}]$ are equal to $\mathtt{b}'[\mathtt{y..y+n-1}]$, element-wise ($\mathtt{b}[\mathtt{x}] = \mathtt{b}'[\mathtt{y}]$, $\mathtt{b}[\mathtt{x+1}] = \mathtt{b}'[\mathtt{y+1}]$, etc). E.g., if $\mathtt{b} = (0, 1, 2, 3, 2, 3)$ and $\mathtt{b}' = (1, 0, 1, 2, 3, 2)$, then all three of $\mathtt{Match}(\mathtt{b}, \mathtt{b}', 0, 1, 4)$, $\mathtt{Match}(\mathtt{b}, \mathtt{b}', 2, 3, 2)$, and $\mathtt{Match}(\mathtt{b}, \mathtt{b}', 4, 3, 2)$ are true. (More technically, if $\sigma(\mathtt{b})$ and $\sigma(\mathtt{b}')$ are as given then $\sigma \vDash \mathtt{Match}(\mathtt{b}, \mathtt{b}', 0, 1, 4)$, etc.)

For this problem, modify the definition for `Match` to get a predicate function $\mathtt{Reverse}(\mathtt{b}, \mathtt{b}', \mathtt{x}, \mathtt{y}, \mathtt{n})$ that checks for the `b` and $\mathtt{b}'$ subsequences being reverses of each other. I.e., we now want $\mathtt{b}[\mathtt{x..x+n-1}]$ (with indexes increasing) to equal $\mathtt{b}'[\mathtt{y+n-1..y}]$ (with indexes decreasing). E.g., For the same `b` and `b'`, $\mathtt{Reverse}(\mathtt{b}, \mathtt{b}', 0, 0, 2)$ and $\mathtt{Reverse}(\mathtt{b}, \mathtt{b}', 2, 3, 3)$ are true but $\mathtt{Reverse}(\mathtt{b}, \mathtt{b}', 0, 0, 3)$ is false.

There are multiple recursive and non-recursive definitions for `Match`; give a non-recursive one.

Part 2: Denotational Semantics, Errors and Nondeterminism

For Questions 1 – 4, calculate the denotational semantics $M(\textit{statement}, \textit{state})$ using the denotational semantics rules given in Lecture 6: $M(v := e, \sigma) = \sigma[v \mapsto \sigma(e)]$, $M(S_1\mathtt{;}\ S_2, \sigma) = M(S_2, \tau) = M(S_2, M(S_1, \sigma))$, etc. Note: Integer division truncates, as does `sqrt(...)`. If the meaning is some flavor of $\perp$, be specific and say $\perp_d$ or $\perp_e$.

1. [5 points] Calculate $M($**if** `x odd` **then** `x := x-1` **fi;** `x := x/2`$, \sigma)$ where σ is an arbitrary state. Your answer will be symbolic and have two cases. Note `x odd` $\equiv$ `x % 2 = 1`.

2. [5 points] Calculate $M(W, \{\mathtt{x} = 0, \mathtt{y} = 1\})$ where $W \equiv$ **while** $\mathtt{x} \neq 3$ **do** S **od** and $S \equiv$ `x := x+1; y := y+y`. Use the technique of characterizing the sequence of states seen at the loop test. Note it will help if you calculate $M(S, \tau)$ for an arbitrary τ.

3. [6 points] Calculate $M(W, \{\mathtt{x} = 4, \mathtt{y} = 1\})$ where W is as in the previous problem.

4. [20 = 4*5 points] Calculate $M(S, \sigma)$ where $S \equiv$ `v` := `b[x];` `w` := `x/v;` `y` := `sqrt(w)`, $\sigma = \{$`b` = (3, 0, −2), `x` = $\alpha\}$, and a value for α as below.

   a. $\alpha = -1$ b. $\alpha = 0$ c. $\alpha = 1$ d. $\alpha = 2$

5. [5 points] What are the semantic similarities and differences among IF_1, IF_2, and IF_3. where
   - $IF_1 \equiv$ **`if`** $B_1 \rightarrow S_1 \square B_2 \rightarrow S_2$ **`fi`**
   - $IF_2 \equiv$ **`if`** $B_2 \rightarrow S_2 \square B_1 \rightarrow S_1$ **`fi`** and
   - $IF_3 \equiv$ **`if`** $B_1 \rightarrow S_1 \square B_2 \rightarrow S_2 \square \neg B_1 \wedge \neg B_2 \rightarrow$ **`skip fi`**

6. [9 points] Let $S \equiv$ **`do`** `j` < 3 → `k` := `k+1;` `j` := `j+1` $\square$ `j` < 5 → `k` := `k-1;` `j` := `j+1` **`od`**, and let $\sigma = \{$`j` = 0, `k` = 0$\}$. What is $M(S, \sigma)$? (It will be a subset of $\Sigma_\perp$.) You can argue informally ("After one iteration, `j` = 1 and `k` = … or …".)

Solution to Homework 2 — Programs, Semantics, Errors, Nondeterminism

Spring 2018

Part 1

(program translation)

1. `j := m ; x := 1; while j >= 1 do j := j - 1; x := x*y[j] od`
2. `j := m; x := 1; j := j-1; while j >= 0 do x := x*y[j]; j := j - 1 od`
3. `x := 1; j := 0; while j < m do j := j + 1; x := x*y[j] od; j := j + 1`

4. (Examples of operational execution)

4a. (Here's a very detailed solution.) Let $\sigma = \{(\mathtt{x}, \alpha), (\mathtt{y}, \beta)\}$ and $S \equiv$ `t := x ; x := y; y := t`, then

$$\begin{aligned}
\langle S, \sigma\rangle &= \langle \mathtt{t := x\ ;\ x := y;\ y := t}, \sigma\rangle \\
&\to \langle \mathtt{x := y;\ y := t}, \sigma[\mathtt{t} \mapsto \sigma(\mathtt{x})]\rangle \\
&= \langle \mathtt{x := y;\ y := t}, \sigma[\mathtt{t} \mapsto \alpha]\rangle \\
&\to \langle \mathtt{y := t}, \sigma_1[\mathtt{x} \mapsto \sigma_1(\mathtt{y})\rangle && \text{where } \sigma_1 = \sigma[\mathtt{t} \mapsto \alpha] \\
&= \langle \mathtt{y := t}, \sigma_1[\mathtt{x} \mapsto \beta]\rangle && \text{bec. } \sigma_1(\mathtt{y}) = \sigma[\mathtt{t} \mapsto \alpha](\mathtt{y}) = \sigma(\mathtt{y}) = \beta \\
&= \langle \mathtt{y := t}, \sigma_2\rangle && \text{where } \sigma_2 = \sigma_1[\mathtt{x} \mapsto \beta] = \sigma[\mathtt{t} \mapsto \alpha][\mathtt{x} \mapsto \beta] \\
&\to \langle E, \sigma_2[\mathtt{y} \mapsto \sigma_2(\mathtt{t})]\rangle \\
&= \langle E, \sigma_2[\mathtt{y} \mapsto \alpha]\rangle && \text{bec. } \sigma_2(\mathtt{t}) = \sigma_1[\mathtt{x} \mapsto \beta](\mathtt{t}) = \sigma_1(\mathtt{t}) = \sigma[\mathtt{t} \mapsto \alpha](\mathtt{t}) = \alpha
\end{aligned}$$

4b. (This solution has less detail.) Let $\sigma = \{(\mathtt{x}, \alpha), (\mathtt{y}, \beta)\}$ and $S \equiv$ `if x < 0 then y := -x else y := x fi`. If $\alpha < 0$, then $\langle S, \sigma\rangle \to \langle \mathtt{y := -x}, \sigma\rangle \to \langle E, \sigma[\mathtt{y} \mapsto -\alpha]\rangle$. Otherwise, if $\alpha \geq 0$, then $\langle S, \sigma\rangle \to \langle \mathtt{y := x}, \sigma\rangle \to \langle E, \sigma[\mathtt{y} \mapsto \alpha]\rangle$.

5. (Operational semantics of loops) As in the problem statement, let $S \equiv$ `s := 0;` W, where $W \equiv$ **while** $\mathtt{n} \geq 0$ **do** S_1 **od** and $S_1 \equiv$ `s := s+n; n := n-1`.

5a. If $\sigma(\mathtt{n}) < 0$, then $\langle S, \sigma\rangle \to \langle W, \sigma[\mathtt{s} \mapsto 0]\rangle \to \langle E, \sigma[\mathtt{s} \mapsto 0]\rangle$

5b. Let's look at the general semantics of the loop body S_1: In an arbitrary state $\tau[\mathtt{s} \mapsto \alpha][\mathtt{n} \mapsto \beta]$, we get $\langle S_1, \tau[\mathtt{s} \mapsto \alpha][\mathtt{n} \mapsto \beta]\rangle = \langle \mathtt{s := s+n;\ n := n-1}, \tau[\mathtt{s} \mapsto \alpha][\mathtt{n} \mapsto \beta]\rangle \to \langle \mathtt{n := n-1}, \tau[\mathtt{n} \mapsto \beta][\mathtt{s} \mapsto \alpha+\beta]\rangle \to \langle E, \tau[\mathtt{s} \mapsto \alpha+\beta][\mathtt{n} \mapsto \beta-1]\rangle$.

5c. If $\sigma(\mathtt{n}) = 3$, then

$$\begin{aligned}
\langle S, \sigma\rangle &\to \langle W, \sigma_0\rangle && \text{where } \sigma_0 = \sigma[\mathtt{s} \mapsto 0][\mathtt{n} \mapsto 3] \\
&\to \langle S_1; W, \sigma_0\rangle && \text{bec. } \sigma_0(\mathtt{n} \geq 0) = (3 \geq 0) = T \\
&\to^* \langle W, \sigma_1\rangle && \text{where } \sigma_1 = \sigma_0[\mathtt{s} \mapsto 0+3][\mathtt{n} \mapsto 3-1] = \sigma[\mathtt{s} \mapsto 3][\mathtt{n} \mapsto 2] \\
&\to \langle S_1; W, \sigma_1\rangle && \text{bec. } \sigma_1(\mathtt{n} \geq 0) = (2 \geq 0) = T \\
&\to^* \langle W, \sigma_2\rangle && \text{where } \sigma_2 = \sigma_1[\mathtt{s} \mapsto 3+2][\mathtt{n} \mapsto 2-1] = \sigma[\mathtt{s} \mapsto 5][\mathtt{n} \mapsto 1] \\
&\to \langle S_1; W, \sigma_2\rangle && \text{bec. } \sigma_2(\mathtt{n} \geq 0) = (1 \geq 0) = T \\
&\to^* \langle W, \sigma_3\rangle && \text{where } \sigma_3 = \sigma_2[\mathtt{s} \mapsto 5+1][\mathtt{n} \mapsto 1-1] = \sigma[\mathtt{s} \mapsto 6][\mathtt{n} \mapsto 0] \\
&\to \langle S_1; W, \sigma_3\rangle && \text{bec. } \sigma_3(\mathtt{n} \geq 0) = (0 \geq 0) = T \\
&\to^* \langle W, \sigma_4\rangle && \text{where } \sigma_4 = \sigma_3[\mathtt{s} \mapsto 6+0][\mathtt{n} \mapsto 0-1] = \sigma[\mathtt{s} \mapsto 6][\mathtt{n} \mapsto -1] \\
&\to \langle E, \sigma_4\rangle && \text{bec. } \sigma_4(\mathtt{n} \geq 0) = (-1 \geq 0) = F
\end{aligned}$$

6. With $\mathtt{Match}(\mathtt{b}, \mathtt{b}', \mathtt{x}, \mathtt{y}, \mathtt{n}) \equiv \forall\, \mathtt{k} \,.\, 0 \le \mathtt{k} < \mathtt{n} \to \mathtt{b[x+k]} = \mathtt{b'[y+k]}$, we test $\mathtt{b[x+k]} = \mathtt{b'[y+k]}$ because they are the $\mathtt{k}$'th elements from the *left* ends of the $\mathtt{b}$ and $\mathtt{b}'$ segments. `Reverse` needs to test the $\mathtt{k}$'th element left of $\mathtt{b[x]}$ with the $\mathtt{k}$'th element *right* of $\mathtt{b'[y+n-2]}$. I.e., we should test $\mathtt{b[x+k]} = \mathtt{b'[y+n-1-k]}$. Quantifying, we get $\mathtt{Reverse}(\mathtt{b}, \mathtt{b}', \mathtt{x}, \mathtt{y}, \mathtt{n}) \equiv \forall\, \mathtt{k} \,.\, 0 \le \mathtt{k} < \mathtt{n} \to \mathtt{b[x+k]} = \mathtt{b'[y+n-1-k]+0}$.

Part 2 — For 1 - 4, recalculate using M(S, σ) = … rules — also

1. Let $\alpha = \sigma(\mathtt{x})$ and let $S \equiv \mathbf{if}\ \mathtt{x\ odd}\ \mathbf{then}\ \mathtt{x := x-1}\ \mathbf{fi;}\ \mathtt{x := x/2}$.
   - If α is odd, then $\langle S, \sigma\rangle \to \langle \mathtt{x := x-1;\ x := x/2}, \sigma\rangle \to \langle \mathtt{x := x/2}, \sigma[\mathtt{x} \mapsto \alpha-1]\rangle \to \langle E, \sigma[\mathtt{x} \mapsto (\alpha-1)/2]\rangle$
   - If α is even then $\langle S, \sigma\rangle \to \langle \mathbf{skip;}\ \mathtt{x := x/2}, \sigma\rangle \to \langle \mathtt{x := x/2}, \sigma\rangle \to \langle E, \sigma[\mathtt{x} \mapsto \alpha\,/\,2]\rangle$

   So $M(S, \sigma) = \sigma[\mathtt{x} \mapsto (\alpha-1)/2]$ (if α is odd) or $\sigma[\mathtt{x} \mapsto \alpha\,/\,2]$ (if α is even). Equivalently, $M(S, \sigma) = \sigma[\mathtt{x} \mapsto \beta]$ where $\alpha = 2\beta$ or $2\beta + 1$.

Check #2: are these values right?

2. We're given $W \equiv \mathbf{while}\ \mathtt{x} \ne 3\ \mathbf{do}\ S\ \mathbf{od}$ and $S \equiv \mathtt{x := x+1;\ y := y+y}$.
   In any arbitrary state $\tau[\mathtt{x} \mapsto \delta][\mathtt{y} \mapsto \eta]$, the loop body ends with $M(S, \tau[\mathtt{x} \mapsto \delta][\mathtt{y} \mapsto \eta]) =$
   $M(\mathtt{x := x+1;\ y := y+y}, \tau[\mathtt{x} \mapsto \delta][\mathtt{y} \mapsto \eta]) = \tau[\mathtt{x} \mapsto \delta+1][\mathtt{y} \mapsto 2\eta]$. Then,

$\langle W, \{\mathtt{x} = 0, \mathtt{y} = 1\}\rangle$	Continue loop, since $\{\mathtt{x} = 0, \mathtt{y} = 1\}(\mathtt{x} \ne 3) = \mathrm{T}$
$\to^3 \langle W, \{\mathtt{x} = 1, \mathtt{y} = 2\}\rangle$	$\{\mathtt{x} = 1, \ldots\}(\mathtt{x} \ne 3) = \mathrm{T}$, so continue
$\to^3 \langle W, \{\mathtt{x} = 2, \mathtt{y} = 4\}\rangle$	$\{\mathtt{x} = 2, \ldots\}(\mathtt{x} \ne 3) = \mathrm{T}$, so continue
$\to^3 \langle W, \{\mathtt{x} = 3, \mathtt{y} = 8\}\rangle$	$\{\mathtt{x} = 3, \ldots\}(\mathtt{x} \ne 3) = \mathrm{F}$, so loop terminates

   So $M(W, \{\mathtt{x} = 0, \mathtt{y} = 1\}) = \{\mathtt{x} = 3, \mathtt{y} = 8\}$.

3. Given W and S as in the previous problem, we have the sequence

   $\langle W, \{\mathtt{x} = 4, \mathtt{y} = 1\}\rangle$
   $\to^3 \langle W, \{\mathtt{x} = 5, \mathtt{y} = 2\}\rangle$
   $\to^3 \langle W, \{\mathtt{x} = 6, \mathtt{y} = 6\}\rangle$
   $\to^3 \langle W, \{\mathtt{x} = 7, \mathtt{y} = 12\}\rangle \to \ldots$

   where in the general case we have $\langle W, \{\mathtt{x} = \beta + 4, \mathtt{y} = 2\ \hat{}\ \beta\}$ for all $\beta \in \mathbb{N}$. In all these configurations, the value of $\mathtt{x} \ne 3$, so we have an infinite sequence, so $M(W, \{\mathtt{x} = 4, \mathtt{y} = 1\}) = \perp_d$.

4. (Runtime errors) We have $S \equiv \mathtt{v := b[x];\ w := x/v;\ y := sqrt(w)}$ and $\sigma = \{\mathtt{b} = (3, 0, -2), \mathtt{x} = \alpha\}$

   4a. If $\sigma(\mathtt{x}) = \alpha = -1$, then $\sigma(\mathtt{b[x]}) = \perp_e$, since the array index $\mathtt{x}$ is -1, which is illegal. So
   $\langle \mathtt{v := b[x];\ w := x/v;\ y := sqrt(w)}, \sigma\rangle \to \langle \mathtt{w := x/v;\ y := sqrt(w)}, \perp_e\rangle \to \langle E, \perp_e\rangle$.

4b. If $\sigma(\texttt{x}) = \alpha = 0$, then $\langle \texttt{v := b[x]; w := x/v; y := sqrt(w)}, \sigma\rangle$

$\rightarrow \langle \texttt{w := x/v; y := sqrt(w)}, \sigma[\texttt{v} \mapsto 3]\rangle$

$\rightarrow \langle \texttt{y := sqrt(w)}, \sigma[\texttt{v} \mapsto 3]\,[\texttt{x} \mapsto 0]\rangle$

$\rightarrow \langle E, \sigma[\texttt{v} \mapsto 3]\,[\texttt{x} \mapsto 0]\,[\texttt{y} \mapsto 0]\,\rangle$, so $M(S, \sigma) = \sigma[\texttt{v} \mapsto 3]\,[\texttt{x} \mapsto 0]\,[\texttt{y} \mapsto 0]$.

4c. If $\sigma(\texttt{x}) = \alpha = 1$, then $\langle \texttt{v := b[x]; w := x/v; y := sqrt(w)}, \sigma\rangle$

$\rightarrow \langle \texttt{w := x/v; y := sqrt(w)}, \sigma[\texttt{v} \mapsto 0]\rangle$

$\rightarrow \langle E, \bot_e\}$ (evaluation of `x/v` fails because `v` is zero in $\sigma[\texttt{v} \mapsto 0]$)

So $M(S, \sigma) = \bot_e$.

4d. If $\sigma(\texttt{x}) = \alpha = 2$ then $\langle \texttt{v := b[x]; w := x/v; y := sqrt(w)}, \sigma\rangle$

$\rightarrow \langle \texttt{w := x/v; y := sqrt(w)}, \sigma[\texttt{v} \mapsto -2]\rangle$ (in σ, $\texttt{x} = 2$ and $\texttt{b[2]} = -2$)

$\rightarrow \langle \texttt{y := sqrt(w)}, \sigma[\texttt{v} \mapsto -2]\,[\texttt{w} \mapsto -1]\,\rangle$ (the value of $\texttt{x} = 2$ in the current state)

$\rightarrow \langle E, \bot_e\}$ (evaluation of `sqrt(w)` fails because $\texttt{w} = -1$)

So $M(S, \sigma) = \bot_e$.

5. IF_1 and IF_2 have the same semantics because the order of the guarded commands doesn't matter. (If both tests are true we choose nondeterministically from the set of statements $\{S_1, S_2\}$.) IF_1 and IF_2 have the same semantics as IF_3 when one of B_1 and B_2 is true and the other is false. If both B_1 and B_2 are false then IF_1 and IF_2 both fail (produce $\bot_e$) but IF_3 executes **`skip`**.

6. We execute the loop for 5 iterations, since both guarded commands increment `j` by 1. The iterations where $\texttt{j} < 3$ can execute either guarded command body, so for $\texttt{j} = 0$, 1, and 2, we nondeterministically increment or decrement `k` by 1, so `k` is $0 \pm 1 \pm 1 \pm 1$ (by which I mean the value of $\texttt{k} \in \{0+1+1+1, 0+1+1-1, \ldots\}$). The set of possible final states turns out to be $\{\{\texttt{j} = 5, \texttt{k} = \beta\} \mid \beta \in \{-5, -3, -1, 1\}\}$:

$\texttt{j} = \ldots$	$\texttt{k} \in \ldots$
0	$\{0\}$
1	$\{0\} \pm 1 = \{-1, 1\}$
2	$\{-1, 1\} \pm 1 = \{-2, 0, 2\}$
3	$\{-2, 0, 2\} \pm 1 = \{-3, -1, 1, 3\}$
4	$\{-3, -1, 1, 3\} - 1 = \{-4, -2, 0, 2\}$
5	$\{-4, -2, 0, 2\} - 1 = \{-5, -3, -1, 1\}$

