Operational Semantics

KC Sivaramakrishnan
Spring 2021

IIT MADRAS

Language

$$
\begin{array}{rccl}
\text{Numbers} & n & \in & \mathbb{N} \\
\text{Variables} & x & \in & \mathsf{Strings} \\
\text{Expressions} & e & ::= & n \mid x \mid e + e \mid e - e \mid e \times e \\
\text{Commands} & c & ::= & \mathsf{skip} \mid x \leftarrow e \mid c; c \mid \mathsf{if}\ e\ \mathsf{then}\ c\ \mathsf{else}\ c \mid \mathsf{while}\ e\ \mathsf{do}\ c
\end{array}
$$

Big Step Semantics

- Says what the result is when the program terminates

$$\frac{}{(v, \mathsf{skip}) \Downarrow v} \quad \frac{}{(v, x \leftarrow e) \Downarrow v[x \mapsto [\![e]\!]v]} \quad \frac{(v, c_1) \Downarrow v_1 \quad (v_1, c_2) \Downarrow v_2}{(v, c_1; c_2) \Downarrow v_2}$$

$$\frac{[\![e]\!]v \neq 0 \quad (v, c_1) \Downarrow v'}{(v, \mathsf{if}\ e\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2) \Downarrow v'} \quad \frac{[\![e]\!]v = 0 \quad (v, c_2) \Downarrow v'}{(v, \mathsf{if}\ e\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2) \Downarrow v'}$$

$$\frac{[\![e]\!]v \neq 0 \quad (v, c_1) \Downarrow v_1 \quad (v_1, \mathsf{while}\ e\ \mathsf{do}\ c_1) \Downarrow v_2}{(v, \mathsf{while}\ e\ \mathsf{do}\ c_1) \Downarrow v_2} \quad \frac{[\![e]\!]v = 0}{(v, \mathsf{while}\ e\ \mathsf{do}\ c_1) \Downarrow v}$$

Small Step Semantics

- Big step semantics only says something about *terminating* programs
  - Program may not terminate (web server)
  - Unclear how to model concurrent interleaving of threads
- Says what the result is when the program takes a *single step*
  - Can model non-termination, concurrency

$$\frac{}{(v, x \leftarrow e) \rightarrow (v[x \mapsto [\![e]\!]v], \mathsf{skip})} \qquad \frac{(v, c_1) \rightarrow (v', c_1')}{(v, c_1; c_2) \rightarrow (v', c_1'; c_2)} \qquad \frac{}{(v, \mathsf{skip}; c_2) \rightarrow (v, c_2)}$$

$$\frac{[\![e]\!]v \neq 0}{(v, \mathsf{if}\ e\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2) \rightarrow (v, c_1)} \qquad \frac{[\![e]\!]v = 0}{(v, \mathsf{if}\ e\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2) \rightarrow (v, c_2)}$$

$$\frac{[\![e]\!]v \neq 0}{(v, \mathsf{while}\ e\ \mathsf{do}\ c_1) \rightarrow (v, c_1; \mathsf{while}\ e\ \mathsf{do}\ c_1)} \qquad \frac{[\![e]\!]v = 0}{(v, \mathsf{while}\ e\ \mathsf{do}\ c_1) \rightarrow (v, \mathsf{skip})}$$

Congruence rules are tedious

$$\frac{(v, c_1) \rightarrow (v', c_1')}{(v, c_1; c_2) \rightarrow (v', c_1'; c_2)}$$

Contextual Small-step Semantics

$$\text{Evaluation contexts} \quad C \quad ::= \quad \square \mid C; c$$

Hole

$$\begin{aligned} \square[c] &= c \\ (C; c_2)[c] &= C[c]; c_2 \end{aligned}$$

Plugging a Hole

Contextual Small-step Semantics for cmd

$$\frac{}{(v, x \leftarrow e) \rightarrow_0 (v[x \mapsto [\![e]\!]v], \textsf{skip})} \quad \frac{}{(v, \textsf{skip}; c_2) \rightarrow_0 (v, c_2)}$$

$$\frac{[\![e]\!]v \neq 0}{(v, \textsf{if } e \textsf{ then } c_1 \textsf{ else } c_2) \rightarrow_0 (v, c_1)} \quad \frac{[\![e]\!]v = 0}{(v, \textsf{if } e \textsf{ then } c_1 \textsf{ else } c_2) \rightarrow_0 (v, c_2)}$$

$$\frac{[\![e]\!]v \neq 0}{(v, \textsf{while } e \textsf{ do } c_1) \rightarrow_0 (v, c_1; \textsf{while } e \textsf{ do } c_1)} \quad \frac{[\![e]\!]v = 0}{(v, \textsf{while } e \textsf{ do } c_1) \rightarrow_0 (v, \textsf{skip})}$$

$$\frac{(v, c) \rightarrow_0 (v', c')}{(v, C[c]) \rightarrow_{\textsf{c}} (v', C[c'])}$$

Contextual Small-step Semantics for cmd

THEOREM 7.11. *There exists valuation v such that* $(\bullet[\texttt{input} \mapsto 2], \texttt{factorial}) \rightarrow_{\mathsf{c}}^{*} (v, \mathsf{skip})$ *and* $v(\texttt{output}) = 2$.

PROOF.

$$
\begin{array}{ll}
 & (\bullet[\texttt{input} \mapsto 2], \texttt{output} \leftarrow 1; \texttt{factorial\_loop}) \\
= & (\bullet[\texttt{input} \mapsto 2], (\Box; \texttt{factorial\_loop})[\texttt{output} \leftarrow 1]) \\
\rightarrow_{\mathsf{c}} & (\bullet[\texttt{input} \mapsto 2][\texttt{output} \mapsto 1], \mathsf{skip}; \texttt{factorial\_loop}) \\
= & (\bullet[\texttt{input} \mapsto 2][\texttt{output} \mapsto 1], \Box[\mathsf{skip}; \texttt{factorial\_loop}]) \\
\rightarrow_{\mathsf{c}} & (\bullet[\texttt{input} \mapsto 2][\texttt{output} \mapsto 1], \texttt{factorial\_loop}) \\
= & (\bullet[\texttt{input} \mapsto 2][\texttt{output} \mapsto 1], \Box[\texttt{factorial\_loop}]) \\
\rightarrow_{\mathsf{c}} & (\bullet[\texttt{input} \mapsto 2][\texttt{output} \mapsto 1], (\texttt{output} \leftarrow \texttt{output} \times \texttt{input}; \texttt{input} \leftarrow \texttt{input} - 1); \texttt{factorial\_loop}) \\
= & (\bullet[\texttt{input} \mapsto 2][\texttt{output} \mapsto 1], ((\Box; \texttt{input} \leftarrow \texttt{input} - 1); \texttt{factorial\_loop})[\texttt{output} \leftarrow \texttt{output} \times \texttt{input}]) \\
\rightarrow_{\mathsf{c}} & (\bullet[\texttt{input} \mapsto 2][\texttt{output} \mapsto 2], (\mathsf{skip}; \texttt{input} \leftarrow \texttt{input} - 1); \texttt{factorial\_loop}) \\
= & (\bullet[\texttt{input} \mapsto 2][\texttt{output} \mapsto 2], (\Box; \texttt{factorial\_loop})[\mathsf{skip}; \texttt{input} \leftarrow \texttt{input} - 1)] \\
\rightarrow_{\mathsf{c}} & (\bullet[\texttt{input} \mapsto 2][\texttt{output} \mapsto 2], \texttt{input} \leftarrow \texttt{input} - 1; \texttt{factorial\_loop}) \\
= & (\bullet[\texttt{input} \mapsto 2][\texttt{output} \mapsto 2], (\Box; \texttt{factorial\_loop})[\texttt{input} \leftarrow \texttt{input} - 1]) \\
\rightarrow_{\mathsf{c}} & (\bullet[\texttt{input} \mapsto 1][\texttt{output} \mapsto 2], \mathsf{skip}; \texttt{factorial\_loop}) \\
= & (\bullet[\texttt{input} \mapsto 1][\texttt{output} \mapsto 2], \Box[\mathsf{skip}; \texttt{factorial\_loop}]) \\
\rightarrow_{\mathsf{c}}^{*} & \ldots \\
\rightarrow_{\mathsf{c}} & (\bullet[\texttt{input} \mapsto 0][\texttt{output} \mapsto 2], \mathsf{skip})
\end{array}
$$

Clearly the final valuation assigns output to 2. □

Equivalence

7.3.1. Equivalence of Small-Step, With and Without Evaluation Contexts. This new semantics formulation is equivalent to the other two, as we establish now.

THEOREM 7.12. *If* $(v, c) \to (v', c')$*, then* $(v, c) \to_{\mathsf{c}} (v', c')$.

PROOF. By induction on the derivation of $(v, c) \to (v', c')$. □

LEMMA 7.13. *If* $(v, c) \to_0 (v', c')$*, then* $(v, c) \to (v', c')$.

PROOF. By cases on the derivation of $(v, c) \to_0 (v', c')$. □

LEMMA 7.14. *If* $(v, c) \to_0 (v', c')$*, then* $(v, C[c]) \to (v', C[c'])$.

PROOF. By induction on the structure of evaluation context C, appealing to the last lemma. □

THEOREM 7.15. *If* $(v, c) \to_{\mathsf{c}} (v', c')$*, then* $(v, c) \to (v', c')$.

PROOF. By inversion on the derivation of $(v, c) \to_{\mathsf{c}} (v', c')$, followed by an appeal to the last lemma. □

Context Payoff: Concurrency

$$\text{Commands} \quad c \quad ::= \quad \ldots \mid c||c$$

$$\text{Evaluation contexts} \quad C \quad ::= \quad \ldots \mid C||c \mid c||C$$

$$\frac{}{(v, \mathsf{skip}||c) \rightarrow_0 (v, c)}$$