#### Operational Semantics

#### KC Sivaramakrishnan

Spring 2025





## Language

```
Numbers n \in \mathbb{N}
Variables x \in \text{Strings}
Expressions e ::= n \mid x \mid e + e \mid e - e \mid e \times e
Commands c ::= \text{skip} \mid x \leftarrow e \mid c; c \mid \text{if } e \text{ then } c \text{ else } c \mid \text{while } e \text{ do } c
```

# Big Step Semantics

Says what the result is only when the program terminates

# Small Step Semantics

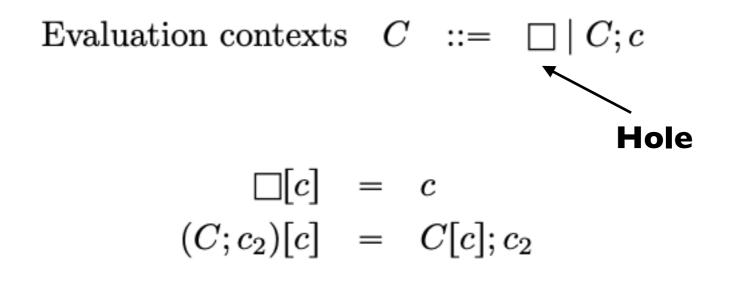
- Big step semantics only says something about terminating programs
  - Program may not terminate (web server)
  - Unclear how to model concurrent interleaving of threads
- Says what the result is when the program takes a single step
  - ◆ Can model non-termination, concurrency

$$\frac{(v,c_1) \rightarrow (v',c_1')}{(v,x \leftarrow e) \rightarrow (v[x \mapsto \llbracket e \rrbracket v], \mathsf{skip})} \quad \frac{(v,c_1) \rightarrow (v',c_1')}{(v,c_1;c_2) \rightarrow (v',c_1';c_2)} \quad \frac{[v] v \neq 0}{(v,\mathsf{if}\ e\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2) \rightarrow (v,c_1)} \quad \frac{[v] v = 0}{(v,\mathsf{if}\ e\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2) \rightarrow (v,c_2)} \\ \frac{[v] v \neq 0}{(v,\mathsf{while}\ e\ \mathsf{do}\ c_1) \rightarrow (v,c_1;\mathsf{while}\ e\ \mathsf{do}\ c_1)} \quad \frac{[v] v = 0}{(v,\mathsf{while}\ e\ \mathsf{do}\ c_1) \rightarrow (v,\mathsf{skip})}$$

### Congruence rules are tedious

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

#### Contextual Small-step Semantics



Plugging a Hole

# Contextual Small-step Semantics for cmd

# Contextual Small-step Semantics for cmd

```
THEOREM 7.11. There exists valuation v such that (\bullet[input \mapsto 2], factorial) \rightarrow_c^*
(v, \mathsf{skip}) and v(\mathsf{output}) = 2.
     PROOF.
          (\bullet[input \mapsto 2], output \leftarrow 1; factorial\_loop)
         (\bullet[input \mapsto 2], (\Box; factorial\_loop)[output \leftarrow 1])
  \rightarrow_{c} (\bullet[input \mapsto 2][output \mapsto 1], skip; factorial_loop)
         (\bullet[input \mapsto 2][output \mapsto 1], \square[skip; factorial\_loop])
        (\bullet[input \mapsto 2][output \mapsto 1], factorial\_loop)
         (\bullet[input \mapsto 2][output \mapsto 1], \square[factorial\_loop])
  \rightarrow_{c} (\bullet[input \mapsto 2][output \mapsto 1], (output \leftarrow output \times input; input \leftarrow input - 1); factorial_loop)
         (\bullet[input \mapsto 2][output \mapsto 1], ((\Box; input \leftarrow input - 1); factorial\_loop)[output \leftarrow output \times input])
        (\bullet[input \mapsto 2][output \mapsto 2], (skip; input \leftarrow input - 1); factorial\_loop)
         (\bullet[\mathtt{input} \mapsto 2][\mathtt{output} \mapsto 2], (\square; \mathtt{factorial\_loop})[\mathtt{skip}; \mathtt{input} \leftarrow \mathtt{input} - 1)]
  \rightarrow_{c} (\bullet[input \mapsto 2][output \mapsto 2], input \leftarrow input -1; factorial_loop)
          (\bullet[input \mapsto 2][output \mapsto 2], (\Box; factorial\_loop)[input \leftarrow input - 1])
         (\bullet[input \mapsto 1][output \mapsto 2], skip; factorial\_loop)
          (\bullet[input \mapsto 1][output \mapsto 2], \square[skip; factorial\_loop])
 \rightarrow_{c}^{*}
  \rightarrow_{c} (\bullet[input \mapsto 0][output \mapsto 2], skip)
```

Clearly the final valuation assigns output to 2.

## Context Payoff: Concurrency

Commands 
$$c ::= \ldots |c||c$$

Evaluation contexts 
$$C ::= \ldots |C||c||c||C$$

$$(v, \mathsf{skip}||c) \to_0 (v, c)$$