Program Analysis
Operational Semantics (Part 5)
Plan for Today

- Motivation & preliminaries
- Abstract syntax of SIMP
- An abstract machine for SIMP
- Structural operation semantics for SIMP
  - Small-step semantics
  - Big-step semantics
Big-step Semantics

small-step semantics: transition relation = individual step of computation

now: transition relation = full evaluation

configurations remain the same

evaluation relation:

\[ (P, s) \Downarrow (P', s') \]

"evaluates to" = last config. of \( (P, s) \)'s evaluation sequ. (if \( P \) terminates)
(Some) axioms & rules

\[
\langle !x, s \rangle \Downarrow \langle n, s \rangle \quad \text{if } s(x) = n
\]

Almost as before:

\[
\langle E, s \rangle \Downarrow \langle n, s' \rangle
\]

\[
\langle \text{\texttt{:=}} E, s \rangle \Downarrow \langle \text{skip}, s'[x \mapsto n] \rangle
\]

\[
\langle C_1, s \rangle \Downarrow \langle \text{skip}, s' \rangle \quad \langle C_2, s' \rangle \Downarrow \langle \text{skip}, s'' \rangle
\]

\[
\langle C_1; C_2, s \rangle \Downarrow \langle \text{skip}, s'' \rangle
\]

\[
\text{(seq)}
\]

Much simpler:

1 rule instead of 2
Example \[ P = (x := !y ; x := !z) ; y := !z) \]
\[ s = \{ x \mapsto 0, y \mapsto 1, z \mapsto 2 \} \]

\[ \langle P, s \rangle \downarrow \langle \text{ship}, s' \rangle \]

where \[ s' = \{ x \mapsto 1, y \mapsto 2, z \mapsto 1 \} \]