# IMP: a simple imperative language

## CS 1520 (Spring 2025)

Harvard University

Tuesday, February 11, 2025

# Today, we learn to

- define operational semantics for a simple imperative language
- prove equivalence between commands
- perform arguments on proof trees
- perform induction over derivation without counterpart over structure

# IMP syntax

- arithmetic expressions

$$a \in \mathbf{Aexp}$$

- boolean expressions

$$b \in \mathbf{Bexp}$$

- commands

$$c \in \mathbf{Com}$$

# IMP syntax

$$a ::= x \mid n \mid a_1 + a_2 \mid a_1 \times a_2$$

$$b ::= \textbf{true} \mid \textbf{false} \mid a_1 < a_2$$

$$\begin{aligned} c ::= {} & \textbf{skip} \mid x := a \mid c_1; c_2 \\ & \mid \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \\ & \mid \textbf{while } b \textbf{ do } c \end{aligned}$$

# Small-step operational semantics

# Small-step operational semantics

- configurations of the form:

# Small-step operational semantics

- configurations of the form:
  - $< a, \sigma >$

# Small-step operational semantics

- configurations of the form:
  - $< a, \sigma >$
  - $< b, \sigma >$

# Small-step operational semantics

- configurations of the form:
  - $< a, \sigma >$
  - $< b, \sigma >$
  - $< c, \sigma >$

# Small-step operational semantics

- configurations of the form:
  - $< a, \sigma >$
  - $< b, \sigma >$
  - $< c, \sigma >$
- final configurations of the form:

# Small-step operational semantics

- configurations of the form:
  - $< a, \sigma >$
  - $< b, \sigma >$
  - $< c, \sigma >$
- final configurations of the form:
  - $< n, \sigma >$

# Small-step operational semantics

- configurations of the form:
  - $< a, \sigma >$
  - $< b, \sigma >$
  - $< c, \sigma >$
- final configurations of the form:
  - $< n, \sigma >$
  - $< \mathbf{true}, \sigma >$, $< \mathbf{false}, \sigma >$

# Small-step operational semantics

- configurations of the form:
  - $< a, \sigma >$
  - $< b, \sigma >$
  - $< c, \sigma >$

- final configurations of the form:
  - $< n, \sigma >$
  - $< \mathbf{true}, \sigma >$, $< \mathbf{false}, \sigma >$
  - $< \mathbf{skip}, \sigma >$

# 3 different small-step operational semantics relations

# 3 different small-step operational semantics relations

$$\longrightarrow_{\mathbf{Aexp}} \subseteq \,?$$

$$\longrightarrow_{\mathbf{Bexp}} \subseteq \,?$$

$$\longrightarrow_{\mathbf{Com}} \subseteq \,?$$

# 3 different small-step operational semantics relations

$$\longrightarrow_{\mathbf{Aexp}} \subseteq \mathbf{Aexp} \times \mathbf{Store} \times \mathbf{Aexp} \times \mathbf{Store}$$

$$\longrightarrow_{\mathbf{Bexp}} \subseteq \mathbf{Bexp} \times \mathbf{Store} \times \mathbf{Bexp} \times \mathbf{Store}$$

$$\longrightarrow_{\mathbf{Com}} \subseteq \mathbf{Com} \times \mathbf{Store} \times \mathbf{Com} \times \mathbf{Store}$$

# 3 different small-step operational semantics relations

$$\longrightarrow_{\mathbf{Aexp}} \subseteq (\mathbf{Aexp} \times \mathbf{Store}) \times (\mathbf{Aexp} \times \mathbf{Store})$$

$$\longrightarrow_{\mathbf{Bexp}} \subseteq (\mathbf{Bexp} \times \mathbf{Store}) \times (\mathbf{Bexp} \times \mathbf{Store})$$

$$\longrightarrow_{\mathbf{Com}} \subseteq (\mathbf{Com} \times \mathbf{Store}) \times (\mathbf{Com} \times \mathbf{Store})$$

# 3 different small-step operational semantics relations

$$(\mathbf{Aexp} \times \mathbf{Store}) \longrightarrow_{\mathbf{Aexp}} (\mathbf{Aexp} \times \mathbf{Store})$$

$$(\mathbf{Bexp} \times \mathbf{Store}) \longrightarrow_{\mathbf{Bexp}} (\mathbf{Bexp} \times \mathbf{Store})$$

$$(\mathbf{Com} \times \mathbf{Store}) \longrightarrow_{\mathbf{Com}} (\mathbf{Com} \times \mathbf{Store})$$

# Arithmetic expressions (1/2)

$$\frac{}{< x, \sigma > \longrightarrow < n, \sigma >} \text{ where } n = \sigma(x)$$

$$\frac{< a_1, \sigma > \longrightarrow < a_1', \sigma >}{< a_1 + a_2, \sigma > \longrightarrow < a_1' + a_2, \sigma >}$$

$$\frac{< a_2, \sigma > \longrightarrow < a_2', \sigma >}{< n + a_2, \sigma > \longrightarrow < n + a_2', \sigma >}$$

$$\frac{}{< n + m, \sigma > \longrightarrow < p, \sigma >} \text{ where } p = n + m$$

# Arithmetic expressions (2/2)

$$\frac{< a_1, \sigma > \longrightarrow < a_1', \sigma >}{< a_1 \times a_2, \sigma > \longrightarrow < a_1' \times a_2, \sigma >}$$

$$\frac{< a_2, \sigma > \longrightarrow < a_2', \sigma >}{< n \times a_2, \sigma > \longrightarrow < n \times a_2', \sigma >}$$

$$\frac{}{< n \times m, \sigma > \longrightarrow < p, \sigma >} \text{ where } p = n \times m$$

# Boolean expressions

$$\frac{< a_1, \sigma > \longrightarrow < a_1', \sigma >}{< a_1 < a_2, \sigma > \longrightarrow < a_1' < a_2, \sigma >}$$

$$\frac{< a_2, \sigma > \longrightarrow < a_2', \sigma >}{< n < a_2, \sigma > \longrightarrow < n < a_2', \sigma >}$$

$$\frac{}{< n < m, \sigma > \longrightarrow < \textbf{true}, \sigma >} \text{ where } n < m$$

$$\frac{}{< n < m, \sigma > \longrightarrow < \textbf{false}, \sigma >} \text{ where } n \geq m$$

# Commands (1/3)

$$\frac{< a, \sigma > \longrightarrow < a', \sigma >}{< x := a, \sigma > \longrightarrow < x := a', \sigma >}$$

$$\frac{}{< x := n, \sigma > \longrightarrow < \textbf{skip}, \sigma[x \mapsto n] >}$$

$$\frac{< c_1, \sigma > \longrightarrow < c_1', \sigma' >}{< c_1; c_2, \sigma > \longrightarrow < c_1'; c_2, \sigma' >}$$

$$\frac{}{< \textbf{skip}; c_2, \sigma > \longrightarrow < c_2, \sigma >}$$

# Commands (2/3)

$$\frac{< b, \sigma > \longrightarrow < b', \sigma >}{\begin{array}{c}< \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2, \sigma > \longrightarrow \\ < \textbf{if } b' \textbf{ then } c_1 \textbf{ else } c_2, \sigma >\end{array}}$$

$$\frac{}{< \textbf{if true then } c_1 \textbf{ else } c_2, \sigma > \longrightarrow < c_1, \sigma >}$$

$$\frac{}{< \textbf{if false then } c_1 \textbf{ else } c_2, \sigma > \longrightarrow < c_2, \sigma >}$$

# Commands (3/3)

$$\frac{}{<\ \textbf{while}\ b\ \textbf{do}\ c, \sigma > \longrightarrow \\ <\ \textbf{if}\ b\ \textbf{then}\ (c; \textbf{while}\ b\ \textbf{do}\ c)\ \textbf{else skip}, \sigma >}$$

# Small-step execution

$$
\begin{array}{lll}
 & < \mathrm{foo} := 3; \textbf{while } \mathrm{foo} < 4 \textbf{ do } \mathrm{foo} := \mathrm{foo} + 5, \sigma > & \\
\longrightarrow & < \textbf{skip}; \textbf{while } \mathrm{foo} < 4 \textbf{ do } \mathrm{foo} := \mathrm{foo} + 5, \sigma' > & \text{where } \sigma' = \sigma[\mathrm{foo} \mapsto 3] \\
\longrightarrow & < \textbf{while } \mathrm{foo} < 4 \textbf{ do } \mathrm{foo} := \mathrm{foo} + 5, \sigma' > & \\
\longrightarrow & < \textbf{if } \mathrm{foo} < 4 \textbf{ then } (\mathrm{foo} := \mathrm{foo} + 5; W) \textbf{ else skip}, \sigma' > & \\
\longrightarrow & < \textbf{if } 3 < 4 \textbf{ then } (\mathrm{foo} := \mathrm{foo} + 5; W) \textbf{ else skip}, \sigma' > & \\
\longrightarrow & < \textbf{if true then } (\mathrm{foo} := \mathrm{foo} + 5; W) \textbf{ else skip}, \sigma' > & \\
\longrightarrow & < \mathrm{foo} := \mathrm{foo} + 5; \textbf{while } \mathrm{foo} < 4 \textbf{ do } \mathrm{foo} := \mathrm{foo} + 5, \sigma' > & \\
\longrightarrow & < \mathrm{foo} := 3 + 5; \textbf{while } \mathrm{foo} < 4 \textbf{ do } \mathrm{foo} := \mathrm{foo} + 5, \sigma' > & \\
\longrightarrow & < \mathrm{foo} := 8; \textbf{while } \mathrm{foo} < 4 \textbf{ do } \mathrm{foo} := \mathrm{foo} + 5, \sigma' > & \\
\longrightarrow & < \textbf{skip}; \textbf{while } \mathrm{foo} < 4 \textbf{ do } \mathrm{foo} := \mathrm{foo} + 5, \sigma'' > & \text{where } \sigma'' = \sigma'[\mathrm{foo} \mapsto 8] \\
\longrightarrow & < \textbf{while } \mathrm{foo} < 4 \textbf{ do } \mathrm{foo} := \mathrm{foo} + 5, \sigma'' > & \\
\longrightarrow & < \textbf{if } \mathrm{foo} < 4 \textbf{ then } (\mathrm{foo} := \mathrm{foo} + 5; W) \textbf{ else skip}, \sigma'' > & \\
\longrightarrow & < \textbf{if } 8 < 4 \textbf{ then } (\mathrm{foo} := \mathrm{foo} + 5; W) \textbf{ else skip}, \sigma'' > & \\
\longrightarrow & < \textbf{if false then } (\mathrm{foo} := \mathrm{foo} + 5; W) \textbf{ else skip}, \sigma'' > & \\
\longrightarrow & < \textbf{skip}, \sigma'' > &
\end{array}
$$

(where $W$ is an abbreviation for the while loop $\textbf{while } \mathrm{foo} < 4 \textbf{ do } \mathrm{foo} := \mathrm{foo} + 5$).

# Large-step operational semantics

# Large-step operational semantics

$$\Downarrow_{\textbf{Aexp}} \subseteq \ ?$$

$$\Downarrow_{\textbf{Bexp}} \subseteq \ ?$$

$$\Downarrow_{\textbf{Com}} \subseteq \ ?$$

# Large-step operational semantics

$$\Downarrow_{\mathbf{Aexp}} \subseteq \mathbf{Aexp} \times \mathbf{Store} \times \mathbf{Int}$$

$$\Downarrow_{\mathbf{Bexp}} \subseteq \mathbf{Bexp} \times \mathbf{Store} \times \mathbf{Bool}$$

$$\Downarrow_{\mathbf{Com}} \subseteq \mathbf{Com} \times \mathbf{Store} \times \mathbf{Store}$$

# Large-step operational semantics

$$\Downarrow_{\mathbf{Aexp}} \subseteq (\mathbf{Aexp} \times \mathbf{Store}) \times \mathbf{Int}$$

$$\Downarrow_{\mathbf{Bexp}} \subseteq (\mathbf{Bexp} \times \mathbf{Store}) \times \mathbf{Bool}$$

$$\Downarrow_{\mathbf{Com}} \subseteq (\mathbf{Com} \times \mathbf{Store}) \times \mathbf{Store}$$

# Large-step operational semantics

$$(\textbf{Aexp} \times \textbf{Store}) \Downarrow_{\textbf{Aexp}} \textbf{Int}$$

$$(\textbf{Bexp} \times \textbf{Store}) \Downarrow_{\textbf{Bexp}} \textbf{Bool}$$

$$(\textbf{Com} \times \textbf{Store}) \Downarrow_{\textbf{Com}} \textbf{Store}$$

# Arithmetic expressions

$$\frac{}{< n, \sigma > \Downarrow n} \qquad \frac{}{< x, \sigma > \Downarrow n} \text{ where } \sigma(x) = n$$

$$\frac{< a_1, \sigma > \Downarrow n_1 \qquad < a_2, \sigma > \Downarrow n_2}{< a_1 + a_2, \sigma > \Downarrow n} \text{ where } n = n_1 + n_2$$

$$\frac{< a_1, \sigma > \Downarrow n_1 \qquad < a_2, \sigma > \Downarrow n_2}{< a_1 \times a_2, \sigma > \Downarrow n} \text{ where } n = n_1 \times n_2$$

# Boolean expressions

$$\frac{}{< \mathbf{true}, \sigma > \Downarrow \mathbf{true}} \qquad \frac{}{< \mathbf{false}, \sigma > \Downarrow \mathbf{false}}$$

$$\frac{< a_1, \sigma > \Downarrow n_1 \qquad < a_2, \sigma > \Downarrow n_2}{< a_1 < a_2, \sigma > \Downarrow \mathbf{true}} \text{ where } n_1 < n_2$$

$$\frac{< a_1, \sigma > \Downarrow n_1 \qquad < a_2, \sigma > \Downarrow n_2}{< a_1 < a_2, \sigma > \Downarrow \mathbf{false}} \text{ where } n_1 \geq n_2$$

# Commands (1/2)

$$\textsc{Skip}\ \frac{}{<\textbf{skip}, \sigma>\Downarrow \sigma}$$

$$\textsc{Asg}\ \frac{<a, \sigma>\Downarrow n}{<x := a, \sigma>\Downarrow \sigma[x \mapsto n]}$$

$$\textsc{Seq}\ \frac{<c_1, \sigma>\Downarrow \sigma' \qquad <c_2, \sigma'>\Downarrow \sigma''}{<c_1; c_2, \sigma>\Downarrow \sigma''}$$

$$\textsc{If-T}\ \frac{<b, \sigma>\Downarrow \textbf{true} \qquad <c_1, \sigma>\Downarrow \sigma'}{<\textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2, \sigma>\Downarrow \sigma'}$$

$$\textsc{If-F}\ \frac{<b, \sigma>\Downarrow \textbf{false} \qquad <c_2, \sigma>\Downarrow \sigma'}{<\textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2, \sigma>\Downarrow \sigma'}$$

# Commands (2/2)

$$\textsc{While-F}\ \frac{<b, \sigma> \Downarrow \textbf{false}}{<\textbf{while}\ b\ \textbf{do}\ c, \sigma> \Downarrow \sigma}$$

$$\textsc{While-T}\ \frac{<b, \sigma> \Downarrow \textbf{true} \qquad <c, \sigma> \Downarrow \sigma' \qquad <\textbf{while}\ b\ \textbf{do}\ c, \sigma'> \Downarrow \sigma''}{<\textbf{while}\ b\ \textbf{do}\ c, \sigma> \Downarrow \sigma''}$$

# Command equivalence

The small-step operational semantics suggest that the loop **while** $b$ **do** $c$ should be equivalent to the command **if** $b$ **then** $(c; \textbf{while } b \textbf{ do } c)$ **else skip**. Can we show that this indeed the case when the language is defined using the above large-step evaluation?

# Equivalence of commands

Two commands $c$ and $c'$ are equivalent
written $c \sim c'$
if, for any stores $\sigma$ and $\sigma'$, we have

$$< c, \sigma > \Downarrow \sigma' \iff < c', \sigma > \Downarrow \sigma'.$$

# Theorem

For all $b \in \mathbf{Bexp}$ and $c \in \mathbf{Com}$ we have

$$\mathbf{while}\ b\ \mathbf{do}\ c$$

$$\sim$$

$$\mathbf{if}\ b\ \mathbf{then}\ (c; \mathbf{while}\ b\ \mathbf{do}\ c)\ \mathbf{else\ skip}$$

# Proof

Let $W$ be an abbreviation for **while** $b$ **do** $c$. We want to show that for all stores $\sigma, \sigma'$, we have:

$$< W, \sigma > \Downarrow \sigma' \iff < \textbf{if } b \textbf{ then } (c; W) \textbf{ else skip}, \sigma > \Downarrow \sigma'$$

For this, we must show that both directions ($\Longrightarrow$ and $\Longleftarrow$) hold. We'll show only direction $\Longrightarrow$; the other is similar.

Assume that $\sigma$ and $\sigma'$ are stores such that $< W, \sigma > \Downarrow \sigma'$. It means that there is some derivation that proves for this fact. Inspecting the evaluation rules, we see that there are two possible rules whose conclusions match this fact: $\textsc{While-F}$ and $\textsc{While-T}$. We analyze each of them in turn.

# Case While-F (1/2)

The derivation must look like the following.

$$\textsc{While-F}\ \frac{\dfrac{\vdots^{1}}{< b, \sigma > \Downarrow \textbf{false}}}{< W, \sigma > \Downarrow \sigma}$$

Here, we use $\vdots^{1}$ to refer to the derivation of $< b, \sigma > \Downarrow$ **false**. Note that in this case, $\sigma' = \sigma$.

# Case $\textsc{While-F}$ (2/2)

We can use $\vdots^1$ to derive a proof tree showing that the evaluation of **if** $b$ **then** $(c; W)$ **else skip** yields the same final state $\sigma$:

$$\textsc{If-F}\ \dfrac{\dfrac{\vdots^1}{< b, \sigma > \Downarrow \textbf{false}} \qquad \textsc{Skip}\ \dfrac{}{< \textbf{skip}, \sigma > \Downarrow \sigma}}{< \textbf{if}\ b\ \textbf{then}\ (c; W)\ \textbf{else skip}, \sigma > \Downarrow \sigma}$$

# Case While-T (1/2)

In this case, the derivation has the following form.

$$\textsc{While-T}\ \dfrac{\dfrac{\vdots^{2}}{< b, \sigma > \Downarrow \textbf{true}} \quad \dfrac{\vdots^{3}}{< c, \sigma > \Downarrow \sigma''} \quad \dfrac{\vdots^{4}}{< W, \sigma'' > \Downarrow \sigma'}}{< W, \sigma > \Downarrow \sigma'}$$

# Case $\textsc{While-T}$ (2/2)

We can use subderivations $\vdots^2$, $\vdots^3$, and $\vdots^4$ to show that the evaluation of **if** $b$ **then** $(c; W)$ **else skip** yields the same final state $\sigma$.

$$
\textsc{If-T}\;\dfrac{\dfrac{\vdots^2}{< b, \sigma > \Downarrow \textbf{true}} \qquad \textsc{Seq}\;\dfrac{\dfrac{\vdots^3}{< c, \sigma > \Downarrow \sigma''} \qquad \dfrac{\vdots^4}{< W, \sigma'' > \Downarrow \sigma'}}{< c; W, \sigma > \Downarrow \sigma'}}{< \textbf{if}\ b\ \textbf{then}\ (c; W)\ \textbf{else skip}, \sigma > \Downarrow \sigma'}
$$

# Break

- Add $\wedge$ to boolean expressions.
- Contrast the design of While in small-step and large-step. Can one style be used for the other? Can you mix small-step and large-step?
- How do you prove that **while true do skip** never terminates? In small-step? In large-step?
- Define and sketch proof for large-step determinism of commands.

# $\wedge$ extending grammar

$$b ::= \ldots \mid b_1 \wedge b_2$$
$$t ::= \textbf{true} \mid \textbf{false}$$

# $\wedge$ extending large-step semantics

$$\frac{< b_1, \sigma > \Downarrow t_1 \qquad < b_2, \sigma > \Downarrow t_2}{< b_1 \wedge b_2, \sigma > \Downarrow t_3}$$

where $t_3$ is **true**
if $t_1$ and $t_2$ are **true**,
and **false** otherwise

# $\wedge$ extending large-step semantics (alternative left-first-sequential)

$$\frac{< b_1, \sigma > \Downarrow \textbf{false}}{< b_1 \wedge b_2, \sigma > \Downarrow \textbf{false}}$$

$$\frac{< b_1, \sigma > \Downarrow \textbf{true} \qquad < b_2, \sigma > \Downarrow \textbf{false}}{< b_1 \wedge b_2, \sigma > \Downarrow \textbf{false}}$$

$$\frac{< b_1, \sigma > \Downarrow \textbf{true} \qquad < b_2, \sigma > \Downarrow \textbf{true}}{< b_1 \wedge b_2, \sigma > \Downarrow \textbf{true}}$$

# Alternative large-step rule for While

$$\frac{< \textbf{if } b \textbf{ then } (c; \textbf{while } b \textbf{ do } c) \textbf{ else skip}, \sigma > \Downarrow \sigma'}{< \textbf{while } b \textbf{ do } c, \sigma > \Downarrow \sigma'}$$

# Determinism

For all commands $c \in \mathbf{Com}$ and stores $\sigma, \sigma_1, \sigma_2 \in \mathbf{Store}$,
if $< c, \sigma > \Downarrow \sigma_1$ and $< c, \sigma > \Downarrow \sigma_2$ then $\sigma_1 = \sigma_2$.

# Proof Sketch for Determinism

By induction on the derivation of $< c, \sigma >\Downarrow \sigma_1$.
The inductive hypothesis $P$ is

$$P(< c, \sigma >\Downarrow \sigma_1) = \forall \sigma_2 \in \textbf{Store}, \\ \text{if } < c, \sigma >\Downarrow \sigma_2 \text{ then } \sigma_1 = \sigma_2.$$

We have a derivation for $< c, \sigma >\Downarrow \sigma_1$, for some $c$, $\sigma$, and $\sigma_1$. Assume that the inductive hypothesis holds for any subderivation $< c', \sigma' >\Downarrow \sigma''$ used in the derivation of $< c, \sigma >\Downarrow \sigma_1$.
Assume that for some $\sigma_2$ we have $< c, \sigma >\Downarrow \sigma_2$.
We need to show that $\sigma_1 = \sigma_2$.

# Case $\textsc{If-T}$ (1/2)

$$\textsc{If-T}\ \dfrac{\dfrac{\vdots}{< b, \sigma >\Downarrow \textbf{true}} \qquad \dfrac{\vdots}{< c_1, \sigma >\Downarrow \sigma_1}}{< \textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2, \sigma >\Downarrow \sigma_1},$$

and we have $c \equiv \textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2$.
The last rule used in the derivation of $< c, \sigma >\Downarrow \sigma_2$ must be either $\textsc{If-T}$ or $\textsc{If-F}$ (since these are the only rules that can be used to derive a conclusion of the form $< \textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2, \sigma >\Downarrow \sigma_2$). But by the determinism of boolean expressions, we must have $< b, \sigma >\Downarrow \textbf{true}$, and so the derivation of $< c, \sigma >\Downarrow \sigma_2$ must have the following form...

# Case IF-T (2/2)

$$\text{IF-T}\ \frac{\dfrac{\vdots}{< b, \sigma > \Downarrow \textbf{true}} \qquad \dfrac{\vdots}{< c_1, \sigma > \Downarrow \sigma_2}}{< \textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2, \sigma > \Downarrow \sigma_2}$$

The result holds by the inductive hypothesis applied

to the derivation $\dfrac{\vdots}{< c_1, \sigma > \Downarrow \sigma_1}$ .

# Case $\textsc{While-T}$ (1/3)

Here we have

$$\textsc{While-T}\ \dfrac{\dfrac{\vdots}{< b, \sigma > \Downarrow \textbf{true}} \quad \dfrac{\vdots}{< c_1, \sigma > \Downarrow \sigma'} \quad \dfrac{\vdots}{< c, \sigma' > \Downarrow \sigma_1}}{< \textbf{while}\ b\ \textbf{do}\ c_1, \sigma > \Downarrow \sigma_1},$$

and we have $c \equiv \textbf{while}\ b\ \textbf{do}\ c_1$. The last rule used in the derivation of $< c, \sigma > \Downarrow \sigma_2$ must also be $\textsc{While-T}$ (by the determinism of boolean expressions), and so we have...

# Case While-T (2/3)

$$
\textsc{While-T}\ \dfrac{\dfrac{\vdots}{< b, \sigma > \Downarrow \textbf{true}} \quad \dfrac{\vdots}{< c_1, \sigma > \Downarrow \sigma''} \quad \dfrac{\vdots}{< c, \sigma'' > \Downarrow \sigma_2}}{< \textbf{while}\ b\ \textbf{do}\ c_1, \sigma > \Downarrow \sigma_2}\ .
$$

By the inductive hypothesis applied to the

derivation $\dfrac{\vdots}{< c_1, \sigma > \Downarrow \sigma'}$ , we have $\sigma' = \sigma''$...

# Case WHILE-T (3/3)

By another application of the inductive hypothesis,

to the derivation $\dfrac{\vdots}{< c, \sigma' > \Downarrow \sigma_1}$, we have $\sigma_1 = \sigma_2$ and the result holds.

# Comment on Case WHILE-T

Even though the command $c \equiv$ **while** $b$ **do** $c_1$ appears in the derivation of $<$ **while** $b$ **do** $c_1, \sigma >\Downarrow \sigma_1$, we do not run in to problems, as the induction is over the *derivation*, not over the structure of the command.