university of groningen

Basic Approaches to the Semantics of Computation (BaSC)

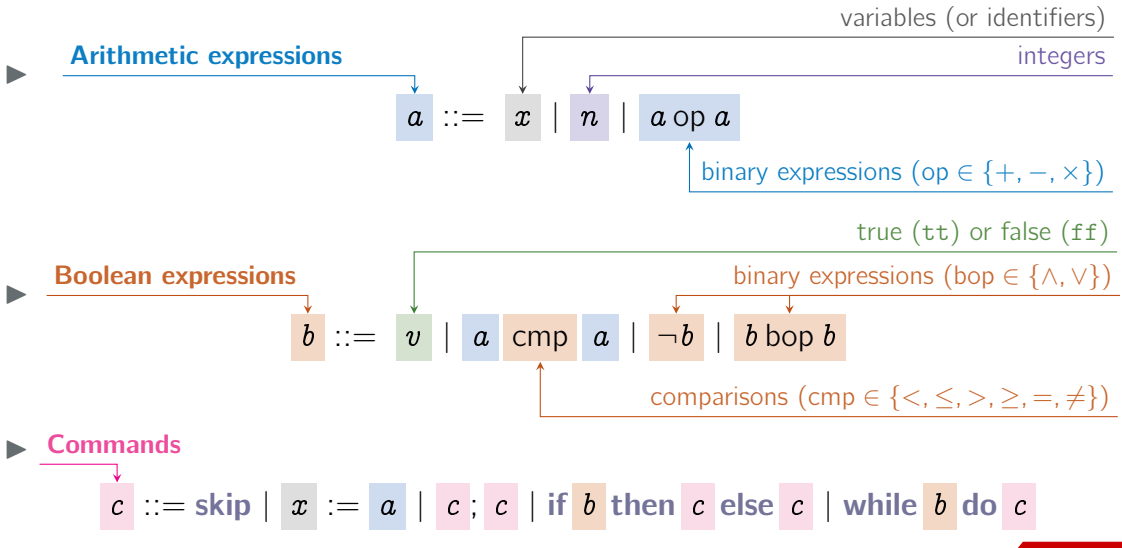
Lecture 4: IMP

Jorge A. Pérez

Bernoulli Institute for Mathematics, Computer Science, and AI
University of Groningen, Groningen, the Netherlands

November 25, 2025

IMP: A Language in Three Layers



Requirements

- In the previous slide, what do colors actually mean?
- Clearly, there are strings that we don't want:
  - $\neg \texttt{1985}$ (negation only defined on booleans)
  - $\texttt{apple} \leq \texttt{orange}$ (comparisons only defined on arithmetic exps)
  - $\texttt{41}; \texttt{42}$ (concatenation only defined on commands)
  - **while** $\texttt{apple}$ **do** $x := x + \texttt{tt}$ (loops only defined on boolean conditions)
- Beyond excluding nonsensical terms, we want to give formal semantics exclusively to terms that make sense
- "terms that make sense" = well-sorted terms

Terms Over a Signature

Assume:

$$S = \{s_1, s_2, \ldots\} \qquad \text{a set of sorts (think: types)}$$
$$\Sigma = \{\Sigma_{s_1,\ldots,s_n,s}\}_{s_1,\ldots,s_n,s \in S} \qquad \text{a many-sorted signature}$$

- This way, we have

$$f \in \Sigma_{s_1,\ldots,s_n,s} \qquad f : (s_1 \times \cdots \times s_n) \to s$$

- The set of all terms of sort s, denoted $T_{\Sigma,s}$ is the least set such that:
  - if $c \in \Sigma_{\epsilon,s}$ then $c \in T_{\Sigma,s}$
  - if $f \in \Sigma_{s_1,\ldots,s_n,s}$ and $\forall i.\, t_i \in T_{\Sigma,s_i}$ then $f(t_1,\ldots,t_n) \in T_{\Sigma,s}$
- $T_\Sigma = \{T_{\Sigma,s}\}_{s \in S}$ denotes the set of all well-sorted terms

Last Time: Arithmetic Expressions

$$x \in \mathsf{Ide} \qquad n \in \mathbb{Z} \qquad \mathsf{op} \in \{+, -, \times\}$$

$$a \in \mathit{Aexp} ::= x \mid n \mid a \,\mathsf{op}\, a$$

$$S \triangleq \{\mathsf{Aexp}\}$$

$$\Sigma_{\epsilon,\mathsf{Aexp}} \triangleq \mathsf{Ide} \cup \mathbb{Z} \qquad \Sigma_{\mathsf{Aexp}\,\mathsf{Aexp},\mathsf{Aexp}} \triangleq \{+, \times, -\}$$

Boolean Expressions

$$x \in \mathrm{Ide} \qquad n \in \mathbb{Z} \qquad v \in \mathbb{B}$$
$$\mathrm{op} \in \{+, -, \times\} \qquad \mathrm{bop} \in \{\wedge, \vee\} \qquad \mathrm{cmp} \in \{<, \leq, >, \geq, =, \neq\}$$

$$a \in \mathit{Aexp} ::= x \mid n \mid a\ \mathrm{op}\ a$$
$$b \in \mathit{Bexp} ::= v \mid a\ \mathrm{cmp}\ a \mid \neg b \mid b\ \mathrm{bop}\ b$$

$$S \triangleq \{\mathrm{Aexp}, \mathrm{Bexp}\}$$

$$\Sigma_{\epsilon,\mathrm{Aexp}} \triangleq \mathrm{Ide} \cup \mathbb{Z} \qquad \Sigma_{\mathrm{Aexp}\,\mathrm{Aexp},\mathrm{Aexp}} \triangleq \{+, \times, -\}$$
$$\Sigma_{\epsilon,\mathrm{Bexp}} \triangleq \mathbb{B} \qquad \Sigma_{\mathrm{Aexp}\,\mathrm{Aexp},\mathrm{Bexp}} \triangleq \{<, \leq, >, \geq, =, \neq\}$$
$$\Sigma_{\mathrm{Bexp},\mathrm{Bexp}} \triangleq \{\neg\} \qquad \Sigma_{\mathrm{Bexp}\,\mathrm{Bexp},\mathrm{Bexp}} \triangleq \{\vee, \wedge\}$$

IMP Semantics (Expressions)

$$\frac{}{\langle x, \sigma\rangle \longrightarrow \sigma(x)} \qquad \frac{}{\langle n, \sigma\rangle \longrightarrow n} \qquad \frac{\langle a_0, \sigma\rangle \longrightarrow n_0 \quad \langle a_1, \sigma\rangle \longrightarrow n_1}{\langle a_0 \text{ op } a_1, \sigma\rangle \longrightarrow n_0 \; op \; n_1}$$

$$\frac{}{\langle v, \sigma\rangle \longrightarrow v} \qquad \frac{\langle b, \sigma\rangle \longrightarrow v}{\langle \neg b, \sigma\rangle \longrightarrow \neg v} \qquad \frac{\langle a_0, \sigma\rangle \longrightarrow n_0 \quad \langle a_1, \sigma\rangle \longrightarrow n_1}{\langle a_0 \text{ cmp } a_1, \sigma\rangle \longrightarrow n_0 \; cmp \; n_1}$$

$$\frac{\langle b_0, \sigma\rangle \longrightarrow v_0 \quad \langle b_1, \sigma\rangle \longrightarrow v_1}{\langle b_0 \text{ bop } b_1, \sigma\rangle \longrightarrow v_0 \; bop \; v_1}$$

IMP Semantics (Expressions)

$$\frac{}{\langle x, \sigma\rangle \longrightarrow \sigma(x)} \qquad \frac{}{\langle n, \sigma\rangle \longrightarrow n} \qquad \frac{\langle a_0, \sigma\rangle \longrightarrow n_0 \quad \langle a_1, \sigma\rangle \longrightarrow n_1}{\langle a_0 \text{ op } a_1, \sigma\rangle \longrightarrow n_0 \; op \; n_1}$$

$$\frac{}{\langle v, \sigma\rangle \longrightarrow v} \qquad \frac{\langle b, \sigma\rangle \longrightarrow v}{\langle \neg b, \sigma\rangle \longrightarrow \neg v} \qquad \frac{\langle a_0, \sigma\rangle \longrightarrow n_0 \quad \langle a_1, \sigma\rangle \longrightarrow n_1}{\langle a_0 \text{ cmp } a_1, \sigma\rangle \longrightarrow n_0 \; cmp \; n_1}$$

$$\frac{\langle b_0, \sigma\rangle \longrightarrow v_0 \quad \langle b_1, \sigma\rangle \longrightarrow v_1}{\langle b_0 \text{ bop } b_1, \sigma\rangle \longrightarrow v_0 \; bop \; v_1}$$

These rules define two different well-formed formulas:

- $\langle a, \sigma\rangle \longrightarrow n$, with $a \in Aexp, n \in \mathbb{Z}$
- $\langle b, \sigma\rangle \longrightarrow v$, with $b \in Bexp, v \in \mathbb{B}$

IMP Semantics (Expressions)

$$\frac{}{\langle x, \sigma\rangle \longrightarrow \sigma(x)} \qquad \frac{}{\langle n, \sigma\rangle \longrightarrow n} \qquad \frac{\langle a_0, \sigma\rangle \longrightarrow n_0 \quad \langle a_1, \sigma\rangle \longrightarrow n_1}{\langle a_0 \text{ op } a_1, \sigma\rangle \longrightarrow n_0 \; op \; n_1}$$

$$\frac{}{\langle v, \sigma\rangle \longrightarrow v} \qquad \frac{\langle b, \sigma\rangle \longrightarrow v}{\langle \neg b, \sigma\rangle \longrightarrow \neg v} \qquad \frac{\langle a_0, \sigma\rangle \longrightarrow n_0 \quad \langle a_1, \sigma\rangle \longrightarrow n_1}{\langle a_0 \text{ cmp } a_1, \sigma\rangle \longrightarrow n_0 \; cmp \; n_1}$$

$$\frac{\langle b_0, \sigma\rangle \longrightarrow v_0 \quad \langle b_1, \sigma\rangle \longrightarrow v_1}{\langle b_0 \text{ bop } b_1, \sigma\rangle \longrightarrow v_0 \; bop \; v_1}$$

A termination property (expanded):

- $P(b) \triangleq \forall \sigma \in \mathbb{M}.\, \exists v \in \mathbb{B}.\, \langle b, \sigma\rangle \longrightarrow v$

$\forall b.P(b)?$

Termination: Base Case (1 of 2)

$$\forall v \in \mathbb{B}.\, P(v)$$

Termination: Base Case (1 of 2)

$$\forall v \in \mathbb{B}.\, P(v)$$

Take some arbitrary $v \in \mathbb{B}$. We must prove:

$$P(v) \triangleq \forall \sigma.\, \exists u.\, \langle v, \sigma \rangle \longrightarrow u$$

Termination: Base Case (1 of 2)

$$\forall v \in \mathbb{B}.\, P(v)$$

Take some arbitrary $v \in \mathbb{B}$. We must prove:

$$P(v) \triangleq \forall \sigma.\, \exists u.\, \langle v, \sigma \rangle \longrightarrow u$$

- Let $\sigma \in \mathbb{M}$ be some arbitrary memory.
  Consider the goal $\langle v, \sigma \rangle \longrightarrow u$, where u is the only variable.

Termination: Base Case (1 of 2)

$$\forall v \in \mathbb{B}.\, P(v)$$

Take some arbitrary $v \in \mathbb{B}$. We must prove:

$$P(v) \triangleq \forall \sigma.\, \exists u.\, \langle v, \sigma \rangle \longrightarrow u$$

- Let $\sigma \in \mathbb{M}$ be some arbitrary memory.
  Consider the goal $\langle v, \sigma \rangle \longrightarrow u$, where u is the only variable.
- By rule $\dfrac{}{\langle v, \sigma \rangle \longrightarrow v}$ we have $\langle v, \sigma \rangle \longrightarrow u \nwarrow_{[u=v]} \square$

Termination: Base Case (1 of 2)

$$\forall v \in \mathbb{B}.\, P(v)$$

Take some arbitrary $v \in \mathbb{B}$. We must prove:

$$P(v) \triangleq \forall \sigma.\, \exists u.\, \langle v, \sigma \rangle \longrightarrow u$$

- Let $\sigma \in \mathbb{M}$ be some arbitrary memory.
  Consider the goal $\langle v, \sigma \rangle \longrightarrow u$, where u is the only variable.
- By rule $\dfrac{}{\langle v, \sigma \rangle \longrightarrow v}$ we have $\langle v, \sigma \rangle \longrightarrow u \nwarrow_{[u=v]} \square$
- We are done by taking $u = v$.

Termination: Another Case

$$\forall a_0, a_1 . P(a_0 \text{ cmp } a_1)$$

Termination: Another Case

$$\forall a_0, a_1 . P(a_0 \text{ cmp } a_1)$$

Take some arbitrary a_0, a_1. We must prove:
$P(a_0 \text{ cmp } a_1) \triangleq \forall \sigma . \exists v . \langle a_0 \text{ cmp } a_1, \sigma \rangle \longrightarrow v$

- Let $\sigma \in \mathbb{M}$ be some memory. Consider the goal $\langle a_0 \text{ op } a_1, \sigma \rangle \longrightarrow v$.

Termination: Another Case

$$\forall a_0, a_1. P(a_0 \text{ cmp } a_1)$$

Take some arbitrary a_0, a_1. We must prove:
$P(a_0 \text{ cmp } a_1) \triangleq \forall \sigma. \exists v. \langle a_0 \text{ cmp } a_1, \sigma \rangle \longrightarrow v$

- Let $\sigma \in \mathbb{M}$ be some memory. Consider the goal $\langle a_0 \text{ op } a_1, \sigma \rangle \longrightarrow v$.
- By rule $\dfrac{\langle a_0, \sigma \rangle \longrightarrow n_0 \quad \langle a_1, \sigma \rangle \longrightarrow n_1}{\langle a_0 \text{ cmp } a_1, \sigma \rangle \longrightarrow n_0 \; cmp \; n_1}$ we have
  $\langle a_0 \text{ cmp } a_1, \sigma \rangle \longrightarrow v \nwarrow_{[v = n_0 \, cmp \, n_1]} \langle a_0, \sigma \rangle \longrightarrow n_0, \langle a_1, \sigma \rangle \longrightarrow n_1$

Termination: Another Base Case

$$\forall a_0, a_1 . P(a_0 \text{ cmp } a_1)$$

Take some arbitrary a_0, a_1. We must prove:
$P(a_0 \text{ cmp } a_1) \triangleq \forall \sigma . \exists v . \langle a_0 \text{ cmp } a_1, \sigma \rangle \longrightarrow v$

- Let $\sigma \in \mathbb{M}$ be some memory. Consider the goal $\langle a_0 \text{ op } a_1, \sigma \rangle \longrightarrow v$.
- By rule $\dfrac{\langle a_0, \sigma \rangle \longrightarrow n_0 \quad \langle a_1, \sigma \rangle \longrightarrow n_1}{\langle a_0 \text{ cmp } a_1, \sigma \rangle \longrightarrow n_0 \; cmp \; n_1}$ we have
  $\langle a_0 \text{ cmp } a_1, \sigma \rangle \longrightarrow v \nwarrow_{[v = n_0 \, cmp \, n_1]} \langle a_0, \sigma \rangle \longrightarrow n_0, \langle a_1, \sigma \rangle \longrightarrow n_1$
- By the termination of arithmetic expressions (not IH!), such n_0, n_1 exist

Termination: Another Base Case

$$\forall a_0, a_1.\ P(a_0 \text{ cmp } a_1)$$

Take some arbitrary a_0, a_1. We must prove:
$P(a_0 \text{ cmp } a_1) \triangleq \forall \sigma.\ \exists v.\ \langle a_0 \text{ cmp } a_1, \sigma \rangle \longrightarrow v$

- Let $\sigma \in \mathbb{M}$ be some memory. Consider the goal $\langle a_0 \text{ op } a_1, \sigma \rangle \longrightarrow v$.
- By rule $\dfrac{\langle a_0, \sigma \rangle \longrightarrow n_0 \quad \langle a_1, \sigma \rangle \longrightarrow n_1}{\langle a_0 \text{ cmp } a_1, \sigma \rangle \longrightarrow n_0 \ cmp \ n_1}$ we have
  $\langle a_0 \text{ cmp } a_1, \sigma \rangle \longrightarrow v \nwarrow_{[v = n_0 \ cmp \ n_1]} \langle a_0, \sigma \rangle \longrightarrow n_0, \langle a_1, \sigma \rangle \longrightarrow n_1$
- By the termination of arithmetic expressions (not IH!), such n_0, n_1 exist
- We are done by taking $v = n_0 \ cmp \ n_1$.

Termination: Inductive Cases

$$\forall b.\, P(b) \quad \Rightarrow \quad P(\neg b)$$

$$\forall b_0, b_1.\, (P(b_0) \wedge P(b_1)) \Rightarrow P(b_0 \text{ bop } b_1)$$

Exercise: Complete these cases yourself.

Commands

$$a \in Aexp ::= x \mid n \mid a \operatorname{op} a \qquad b \in Bexp ::= v \mid a \operatorname{cmp} a \mid \neg b \mid b \operatorname{bop} b$$
$$c \in Com ::= \textbf{skip} \mid x := a \mid c; c \mid \textbf{if } b \textbf{ then } c \textbf{ else } c \mid \textbf{while } b \textbf{ do } c$$

$$S \triangleq \{\mathsf{Aexp}, \mathsf{Bexp}, \mathsf{Com}\}$$

$$\Sigma_{\epsilon,\mathsf{Aexp}} \triangleq \mathsf{Ide} \cup \mathbb{Z} \qquad \Sigma_{\mathsf{Aexp}\,\mathsf{Aexp},\mathsf{Aexp}} \triangleq \{+, \times, -\}$$
$$\Sigma_{\epsilon,\mathsf{Bexp}} \triangleq \mathbb{B} \qquad \Sigma_{\mathsf{Aexp}\,\mathsf{Aexp},\mathsf{Bexp}} \triangleq \{<, \leq, >, \geq, =, \neq\}$$
$$\Sigma_{\mathsf{Bexp},\mathsf{Bexp}} \triangleq \{\neg\} \qquad \Sigma_{\mathsf{Bexp}\,\mathsf{Bexp},\mathsf{Bexp}} \triangleq \{\vee, \wedge\}$$
$$\Sigma_{\epsilon,\mathsf{Com}} \triangleq \{\textbf{skip}\} \qquad \Sigma_{\mathsf{Aexp},\mathsf{Com}} \triangleq \{x ::= \mid x \in \mathsf{Ide}\}$$
$$\Sigma_{\mathsf{Com}\,\mathsf{Com},\mathsf{Com}} \triangleq \{\,;\,\} \qquad \Sigma_{\mathsf{Bexp}\,\mathsf{Com}\,\mathsf{Com},\mathsf{Com}} \triangleq \{\textbf{if } \cdot \textbf{ then } \cdot \textbf{ else } \cdot\}$$
$$\Sigma_{\mathsf{Bexp}\,\mathsf{Com},\mathsf{Com}} \triangleq \{\textbf{while } \cdot \textbf{ do } \cdot\}$$

Memories

- A mapping from identifiers (locations / variables) to integers:

$$\mathbb{M} \triangleq \{\sigma : \mathsf{Ide} \to \mathbb{Z} \mid \sigma(x) \neq 0\}$$

σ has finite support

Memories

- A mapping from identifiers (locations / variables) to integers:

$$\mathbb{M} \triangleq \{\sigma : \mathsf{Ide} \to \mathbb{Z} \mid \sigma(x) \neq 0\}$$

σ has finite support

- Example: Memory $\sigma = (42/x, 37/y)$ is defined only for x and y.
  We write $\sigma(x) = 42$ and $\sigma(y) = 37$. For any other $z \in \mathsf{Ide}$, $\sigma(z) = 0$.

Memories

- A mapping from identifiers (locations / variables) to integers:

$$\mathbb{M} \triangleq \{\sigma : \mathsf{Ide} \to \mathbb{Z} \mid \sigma(x) \neq 0\}$$

σ has finite support

- Example: Memory $\sigma = (42/x, 37/y)$ is defined only for x and y. We write $\sigma(x) = 42$ and $\sigma(y) = 37$. For any other $z \in \mathsf{Ide}$, $\sigma(z) = 0$.
- Formally:

$$(n_1/x_1, \cdots, n_k/x_k) : \mathsf{Ide} \to \mathbb{Z}$$

all different

$$(n_1/x_1, \cdots, n_k/x_k)(x) \triangleq \begin{cases} n_i & \text{if } x = x_i, \\ 0 & \text{otherwise.} \end{cases}$$

Memories

- A mapping from identifiers (locations / variables) to integers:

$$\mathbb{M} \triangleq \{\sigma : \mathsf{Ide} \to \mathbb{Z} \mid \sigma(x) \neq 0\}$$

σ has finite support

- Example: Memory $\sigma = (42/x, 37/y)$ is defined only for x and y. We write $\sigma(x) = 42$ and $\sigma(y) = 37$. For any other $z \in \mathsf{Ide}$, $\sigma(z) = 0$.
- Formally:

$$(n_1/x_1, \cdots, n_k/x_k) : \mathsf{Ide} \to \mathbb{Z}$$

all different

$$(n_1/x_1, \cdots, n_k/x_k)(x) \triangleq \begin{cases} n_i & \text{if } x = x_i, \\ 0 & \text{otherwise.} \end{cases}$$

- The initial memory is $\sigma_0 \triangleq ()$.

Memory Updates

$$\sigma[n/y](x) \triangleq \begin{cases} n & \text{if } x = y, \\ \sigma(x) & \text{otherwise.} \end{cases}$$

Memory Updates

$$\sigma[n/y](x) \triangleq \begin{cases} n & \text{if } x = y, \\ \sigma(x) & \text{otherwise.} \end{cases}$$

Example

Suppose $\sigma = (42/x, 37/y)$.

- $\sigma[74/y]$ represents an update of σ: a revised mapping.
  Then $\sigma[74/y](y) = 74$ (actual update) and $\sigma[74/y](x) = 42$ (as before)
- $\sigma' = \sigma[\sigma(x) + 1/x]$ adds 1 to the current value of x. Hence, $\sigma'(x) = 43$.
- $\sigma[55/z]$ updates σ by instantiating a new variable: an "extended mapping".

Memory Updates

$$\sigma[n/y](x) \triangleq \begin{cases} n & \text{if } x = y, \\ \sigma(x) & \text{otherwise.} \end{cases}$$

Some properties:

1. $\forall \sigma, m, n, y.\, \sigma[m/y][n/y] = \sigma[n/y]$. We have:

$$\sigma[m/y][n/y](x) \triangleq \begin{cases} n & \text{if } x = y, \\ \sigma[m/y](x) = \sigma(x) & \text{otherwise.} \end{cases}$$

Memory Updates

$$\sigma[n/y](x) \triangleq \begin{cases} n & \text{if } x = y, \\ \sigma(x) & \text{otherwise.} \end{cases}$$

Some properties:

1. $\forall \sigma, m, n, y.\, \sigma[m/y][n/y] = \sigma[n/y]$. We have:

$$\sigma[m/y][n/y](x) \triangleq \begin{cases} n & \text{if } x = y, \\ \sigma[m/y](x) = \sigma(x) & \text{otherwise.} \end{cases}$$

2. $\forall \sigma, m, n, y, z.\, y \neq z \Rightarrow \sigma[n/y][m/z] = \sigma[m/z][n/y]$.
   We use the simpler notation $\sigma[n/y, m/z]$ in such cases.

Memory Updates

$$\sigma[n/y](x) \triangleq \begin{cases} n & \text{if } x = y, \\ \sigma(x) & \text{otherwise.} \end{cases}$$

Some properties:

1. $\forall \sigma, m, n, y \,.\, \sigma[m/y][n/y] = \sigma[n/y]$. We have:

$$\sigma[m/y][n/y](x) \triangleq \begin{cases} n & \text{if } x = y, \\ \sigma[m/y](x) = \sigma(x) & \text{otherwise.} \end{cases}$$

2. $\forall \sigma, m, n, y, z \,.\, y \neq z \Rightarrow \sigma[n/y][m/z] = \sigma[m/z][n/y]$.
   We use the simpler notation $\sigma[n/y, m/z]$ in such cases.
3. Also: $(n_1/x_1, \cdots, n_k/x_k) = \sigma_0[n_1/x_1, \cdots, n_k/x_k]$.

IMP Semantics (Commands)

$$\frac{}{\langle \textbf{skip}, \sigma \rangle \longrightarrow \sigma} \qquad \frac{\langle a, \sigma \rangle \longrightarrow n}{\langle x := a, \sigma \rangle \longrightarrow \sigma[n/x]} \qquad \frac{\langle c_0, \sigma \rangle \longrightarrow \sigma'' \quad \langle c_1, \sigma'' \rangle \longrightarrow \sigma'}{\langle c_0; c_1, \sigma \rangle \longrightarrow \sigma'}$$

$$\frac{\langle b, \sigma \rangle \longrightarrow \texttt{ff} \quad \langle c_1, \sigma \rangle \longrightarrow \sigma'}{\langle \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1, \sigma \rangle \longrightarrow \sigma'} \qquad \frac{\langle b, \sigma \rangle \longrightarrow \texttt{tt} \quad \langle c_0, \sigma \rangle \longrightarrow \sigma'}{\langle \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1, \sigma \rangle \longrightarrow \sigma'}$$

$$\frac{\langle b, \sigma \rangle \longrightarrow \texttt{ff}}{\langle \textbf{while } b \textbf{ do } c, \sigma \rangle \longrightarrow \sigma}$$

$$\frac{\langle b, \sigma \rangle \longrightarrow \texttt{tt} \quad \langle c, \sigma \rangle \longrightarrow \sigma'' \quad \langle \textbf{while } b \textbf{ do } c, \sigma'' \rangle \longrightarrow \sigma'}{\langle \textbf{while } b \textbf{ do } c, \sigma \rangle \longrightarrow \sigma'}$$

In this case, the rules define well-formed formulas of the form $\langle c, \sigma \rangle \longrightarrow \sigma'$.

Example

A command c:

$$\underbrace{\underbrace{\underbrace{x := 0}_{c_3}; \underbrace{\textbf{while}\ 0 \leq y\ \textbf{do}\ \underbrace{\underbrace{x := x + (2 \times y) + 1}_{c_3}; \underbrace{y := y - 1}_{c_4}}_{c_2}}_{c_1}}_{c}$$

What does c do?

Example

A command c:

$$\underbrace{\underbrace{\underbrace{x := 0; \textbf{while}\ 0 \leq y\ \textbf{do}\ \underbrace{\underbrace{x := x + (2 \times y) + 1}_{c_3}; \underbrace{y := y - 1}_{c_4}}_{c_2}}_{c_1}}_{c}$$

What does c do?

Let us derive as goal-oriented derivation for $\langle c, (27/x, 2/y)\rangle \longrightarrow \sigma$.

Goal-Oriented Derivation (Excerpt)

$$
\begin{array}{rl}
 & \langle c, (27/x, 2/y)\rangle \longrightarrow \sigma \\
\nwarrow & \langle x := 0, (27/x, 2/y)\rangle \longrightarrow \sigma_1, \langle c_1, \sigma_1\rangle \longrightarrow \sigma \\
\nwarrow_{\sigma_1=(27/x,2/y)[n/x]} & \langle x := 0, (27/x, 2/y)\rangle \longrightarrow n, \langle c_1, (27/x, 2/y)[n/x]\rangle \longrightarrow \sigma \\
\nwarrow_{n=0,\sigma_1=(0/x,2/y)} & \langle c_1, (0/x, 2/y)\rangle \longrightarrow \sigma \\
\nwarrow & \langle 0 \leq y, (0/x, 2/y)\rangle \longrightarrow \mathtt{tt}, \langle c_2, (0/x, 2/y)\rangle \longrightarrow \sigma_2, \langle c_1, \sigma_2\rangle \longrightarrow \sigma \\
 & \vdots \\
\nwarrow_{\sigma=(9/x,-1/y)} & \langle 0 \leq y, (9/x, -1/y)\rangle \longrightarrow \mathtt{ff}
\end{array}
$$

Look Closely

Do you notice something odd about this rule?

$$\frac{\langle b, \sigma\rangle \longrightarrow \texttt{tt} \quad \langle c, \sigma\rangle \longrightarrow \sigma'' \quad \langle \textbf{while } b \textbf{ do } c, \sigma''\rangle \longrightarrow \sigma'}{\langle \textbf{while } b \textbf{ do } c, \sigma\rangle \longrightarrow \sigma'}$$

Look Closely

Do you notice something odd about this rule?

this premise is as complex as the conclusion!

$$\frac{\langle b, \sigma\rangle \longrightarrow \texttt{tt} \quad \langle c, \sigma\rangle \longrightarrow \sigma'' \quad \langle \textbf{while } b \textbf{ do } c, \sigma''\rangle \longrightarrow \sigma'}{\langle \textbf{while } b \textbf{ do } c, \sigma\rangle \longrightarrow \sigma'}$$

Look Closely

Do you notice something odd about this rule?

this premise is as complex as the conclusion!

$$\frac{\langle b, \sigma\rangle \longrightarrow \texttt{tt} \quad \langle c, \sigma\rangle \longrightarrow \sigma'' \quad \langle \textbf{while } b \textbf{ do } c, \sigma''\rangle \longrightarrow \sigma'}{\langle \textbf{while } b \textbf{ do } c, \sigma\rangle \longrightarrow \sigma'}$$

A recursive definition!

IMP Semantics (Commands)

$$\frac{}{\langle \textbf{skip}, \sigma\rangle \longrightarrow \sigma} \qquad \frac{\langle a, \sigma\rangle \longrightarrow n}{\langle x := a, \sigma\rangle \longrightarrow \sigma[n/x]} \qquad \frac{\langle c_0, \sigma\rangle \longrightarrow \sigma'' \quad \langle c_1, \sigma''\rangle \longrightarrow \sigma'}{\langle c_0; c_1, \sigma\rangle \longrightarrow \sigma'}$$

$$\frac{\langle b, \sigma\rangle \longrightarrow \texttt{ff} \quad \langle c_1, \sigma\rangle \longrightarrow \sigma'}{\langle \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1, \sigma\rangle \longrightarrow \sigma'} \qquad \frac{\langle b, \sigma\rangle \longrightarrow \texttt{tt} \quad \langle c_0, \sigma\rangle \longrightarrow \sigma'}{\langle \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1, \sigma\rangle \longrightarrow \sigma'}$$

$$\frac{\langle b, \sigma\rangle \longrightarrow \texttt{ff}}{\langle \textbf{while } b \textbf{ do } c, \sigma\rangle \longrightarrow \sigma} \qquad \frac{\langle b, \sigma\rangle \longrightarrow \texttt{tt} \quad \langle c, \sigma\rangle \longrightarrow \sigma'' \quad \langle \textbf{while } b \textbf{ do } c, \sigma''\rangle \longrightarrow \sigma'}{\langle \textbf{while } b \textbf{ do } c, \sigma\rangle \longrightarrow \sigma'}$$

A termination property (further expanded):

- $P(c) \triangleq \forall \sigma \in \mathbb{M}.\, \exists \sigma' \in \mathbb{M}.\, \langle c, \sigma\rangle \longrightarrow \sigma'$ $\qquad \forall c.P(c)?$

Termination?

$$P(c) \triangleq \forall \sigma \in \mathbb{M}.\, \exists \sigma' \in \mathbb{M}.\, \langle c, \sigma \rangle \longrightarrow \sigma' \qquad \forall c.P(c)?$$

- Consider $w \triangleq$ **while** $\mathtt{tt}$ **do skip**.
  We have:

$$\langle w, \sigma \rangle \longrightarrow \sigma'$$
$$\nwarrow \langle \mathtt{tt}, \sigma \rangle \longrightarrow \mathtt{tt}, \langle \textbf{skip}, \sigma \rangle \longrightarrow \sigma'', \langle w, \sigma'' \rangle \longrightarrow \sigma'$$
$$\nwarrow \langle \textbf{skip}, \sigma \rangle \longrightarrow \sigma'', \langle w, \sigma'' \rangle \longrightarrow \sigma'$$
$$\nwarrow_{\sigma''=\sigma} \langle w, \sigma \rangle \longrightarrow \sigma'$$

- We reached the same goal from which we started!
  No options to backtrack: $\langle \mathtt{tt}, \sigma \rangle \not\longrightarrow \mathtt{ff}$. Cannot complete the derivation!
- We conclude $\neg P(w)$: we found a counter-example to termination.

Divergence

- $\langle c, \sigma \rangle \not\longrightarrow$ means $\neg \exists \sigma'. \langle c, \sigma \rangle \longrightarrow \sigma'$.
- We say that c diverges on σ.
- Divergence is difficult to detect and handle — it is undecidable!

Divergence: Example

Consider $w \triangleq \textbf{while}\, x > 0\, \textbf{do}\, x := x + 1$, and some arbitrary σ. Two possibilities:

Divergence: Example

Consider $w \triangleq \textbf{while}\, x > 0\, \textbf{do}\, x := x + 1$, and some arbitrary σ. Two possibilities:

- If $\sigma(x) \leq 0$:
  $\langle w, \sigma \rangle \longrightarrow \sigma' \nwarrow_{\sigma' = \sigma} \langle x > 0, \sigma \rangle \longrightarrow \texttt{ff} \nwarrow^{*} \Box$.
  Hence, $\langle w, \sigma \rangle \longrightarrow \sigma$.

Divergence: Example

Consider $w \triangleq \textbf{while}\, x > 0\, \textbf{do}\, x := x + 1$, and some arbitrary σ. Two possibilities:

- If $\sigma(x) \leq 0$:
  $\langle w, \sigma\rangle \longrightarrow \sigma' \nwarrow_{\sigma'=\sigma} \langle x > 0, \sigma\rangle \longrightarrow \texttt{ff} \nwarrow^* \Box$.
  Hence, $\langle w, \sigma\rangle \longrightarrow \sigma$.
- If $\sigma(x) > 0$:

$$\langle w, \sigma\rangle \longrightarrow \sigma'$$

$$\nwarrow \langle x > 0, \sigma\rangle \longrightarrow \texttt{tt}, \langle x := x + 1, \sigma\rangle \longrightarrow \sigma'', \langle w, \sigma''\rangle \longrightarrow \sigma'$$

$$\nwarrow^* \langle x := x + 1, \sigma\rangle \longrightarrow \sigma'', \langle w, \sigma''\rangle \longrightarrow \sigma'$$

$$\nwarrow_{\sigma''=\sigma[n/x]} \langle x := x + 1, \sigma\rangle \longrightarrow n, \langle w, \sigma[n/x]\rangle \longrightarrow \sigma'$$

$$\nwarrow^*_{n=\sigma(x)+1} \langle w, \sigma[\sigma(x) + 1/x]\rangle \longrightarrow \sigma'$$

Divergence: Example

Consider $w \triangleq \textbf{while}\, x > 0\, \textbf{do}\, x := x + 1$, and some arbitrary σ. Two possibilities:

- If $\sigma(x) \leq 0$:
  $\langle w, \sigma\rangle \longrightarrow \sigma' \nwarrow_{\sigma'=\sigma} \langle x > 0, \sigma\rangle \longrightarrow \texttt{ff} \nwarrow^* \Box$.
  Hence, $\langle w, \sigma\rangle \longrightarrow \sigma$.
- If $\sigma(x) > 0$:

$$\langle w, \sigma\rangle \longrightarrow \sigma'$$

$$\nwarrow \langle x > 0, \sigma\rangle \longrightarrow \texttt{tt}, \langle x := x + 1, \sigma\rangle \longrightarrow \sigma'', \langle w, \sigma''\rangle \longrightarrow \sigma'$$

$$\nwarrow^* \langle x := x + 1, \sigma\rangle \longrightarrow \sigma'', \langle w, \sigma''\rangle \longrightarrow \sigma'$$

$$\nwarrow_{\sigma''=\sigma[n/x]} \langle x := x + 1, \sigma\rangle \longrightarrow n, \langle w, \sigma[n/x]\rangle \longrightarrow \sigma'$$

$$\nwarrow^*_{n=\sigma(x)+1} \langle w, \sigma[\sigma(x) + 1/x]\rangle \longrightarrow \sigma'$$

Not the same goal we started from! How to prove divergence?

Proving Divergence

Consider $w \triangleq \textbf{while}\ b\ \textbf{do}\ c$. Proof idea: find a set of memories $S \subseteq \mathbb{M}$ such that

1. $\forall \sigma \in S. \langle b, \sigma \rangle \longrightarrow \mathtt{tt}$
2. $\forall \sigma \in S. \forall \sigma' \in \mathbb{M}. \langle c, \sigma \rangle \longrightarrow \sigma' \Rightarrow \sigma' \in S$

Then we can conclude $\forall \sigma \in S. \langle w, \sigma \rangle \not\longrightarrow$.

Proving Divergence

Consider $w \triangleq \textbf{while}\ b\ \textbf{do}\ c$. Proof idea: find a set of memories $S \subseteq \mathbb{M}$ such that

1. $\forall \sigma \in S.\, \langle b, \sigma \rangle \longrightarrow \mathtt{tt}$
2. $\forall \sigma \in S. \forall \sigma' \in \mathbb{M}.\, \langle c, \sigma \rangle \longrightarrow \sigma' \Rightarrow \sigma' \in S$

Then we can conclude $\forall \sigma \in S.\, \langle w, \sigma \rangle \not\longrightarrow$.

Equivalently, as a rule:

$$\frac{\sigma \in S \qquad \forall \sigma \in S.\, \langle b, \sigma \rangle \longrightarrow \mathtt{tt} \qquad \forall \sigma \in S. \forall \sigma' \in \mathbb{M}.\, (\langle c, \sigma \rangle \longrightarrow \sigma' \Rightarrow \sigma' \in S)}{\langle w, \sigma \rangle \not\longrightarrow}$$

Note:

- If $\langle c, \sigma \rangle \not\longrightarrow$ then $\langle c, \sigma \rangle \longrightarrow \sigma' \Rightarrow \sigma' \in S$ trivially holds.
- Proving divergence is not always possible (nor easy)

Divergence: Example (Revisited)

Consider $w \triangleq \textbf{while}\ x > 0\ \textbf{do}\ x := x + 1$, and some arbitrary σ. Two possibilities:

- If $\sigma(x) \leq 0$, then we proceed to show $\langle w, \sigma \rangle \longrightarrow \sigma$ (as before).
- If $\sigma(x) > 0$, we define

$$S \triangleq \{\sigma \mid \sigma(x) > 0\}$$

Divergence: Example (Revisited)

Consider $w \triangleq \textbf{while}\ x > 0\ \textbf{do}\ x := x + 1$, and some arbitrary σ. Two possibilities:

- If $\sigma(x) \leq 0$, then we proceed to show $\langle w, \sigma \rangle \longrightarrow \sigma$ (as before).
- If $\sigma(x) > 0$, we define

$$S \triangleq \{\sigma \mid \sigma(x) > 0\}$$

We check the two conditions:

1. $\forall \sigma \in S.\, \langle x > 0, \sigma \rangle \longrightarrow \mathtt{tt}$ ✓
2. $\forall \sigma \in S. \forall \sigma'.\, \langle x := x + 1, \sigma \rangle \longrightarrow \sigma' \Rightarrow \sigma' \in S$ ✓

Divergence: Example (Revisited)

Consider $w \triangleq \textbf{while}\ x > 0\ \textbf{do}\ x := x + 1$, and some arbitrary σ. Two possibilities:

- If $\sigma(x) \leq 0$, then we proceed to show $\langle w, \sigma \rangle \longrightarrow \sigma$ (as before).
- If $\sigma(x) > 0$, we define

$$S \triangleq \{\sigma \mid \sigma(x) > 0\}$$

We check the two conditions:

1. $\forall \sigma \in S.\, \langle x > 0, \sigma \rangle \longrightarrow \mathtt{tt}$ ✓
2. $\forall \sigma \in S. \forall \sigma'.\, \langle x := x + 1, \sigma \rangle \longrightarrow \sigma' \Rightarrow \sigma' \in S$ ✓

In fact:

$$\langle x := x + 1, \sigma \rangle \longrightarrow \sigma' \Rightarrow \sigma' = \sigma[\sigma(x) + 1/x]$$
$$\sigma(x) > 0 \Rightarrow \sigma[\sigma(x) + 1/x](x) = \sigma(x) + 1 > 0$$

Therefore, if $\sigma(x) > 0$ then $\langle w, \sigma \rangle \not\longrightarrow$.

IMP Semantics (Commands)

$$\frac{}{\langle \mathtt{skip}, \sigma\rangle \longrightarrow \sigma} \qquad \frac{\langle a, \sigma\rangle \longrightarrow n}{\langle x := a, \sigma\rangle \longrightarrow \sigma[n/x]} \qquad \frac{\langle c_0, \sigma\rangle \longrightarrow \sigma'' \quad \langle c_1, \sigma''\rangle \longrightarrow \sigma'}{\langle c_0; c_1, \sigma\rangle \longrightarrow \sigma'}$$

$$\frac{\langle b, \sigma\rangle \longrightarrow \mathtt{ff} \quad \langle c_1, \sigma\rangle \longrightarrow \sigma'}{\langle \mathtt{if}\ b\ \mathtt{then}\ c_0\ \mathtt{else}\ c_1, \sigma\rangle \longrightarrow \sigma'} \qquad \frac{\langle b, \sigma\rangle \longrightarrow \mathtt{tt} \quad \langle c_0, \sigma\rangle \longrightarrow \sigma'}{\langle \mathtt{if}\ b\ \mathtt{then}\ c_0\ \mathtt{else}\ c_1, \sigma\rangle \longrightarrow \sigma'}$$

$$\frac{\langle b, \sigma\rangle \longrightarrow \mathtt{ff}}{\langle \mathtt{while}\ b\ \mathtt{do}\ c, \sigma\rangle \longrightarrow \sigma} \qquad \frac{\langle b, \sigma\rangle \longrightarrow \mathtt{tt} \quad \langle c, \sigma\rangle \longrightarrow \sigma'' \quad \langle \mathtt{while}\ b\ \mathtt{do}\ c, \sigma''\rangle \longrightarrow \sigma'}{\langle \mathtt{while}\ b\ \mathtt{do}\ c, \sigma\rangle \longrightarrow \sigma'}$$

We have seen determinacy for arithmetic expressions. For commands:

- $P(c) \triangleq \forall \sigma, \sigma_1, \sigma_2 \in \mathbb{M}.\, (\langle c, \sigma\rangle \longrightarrow \sigma_1 \wedge \langle c, \sigma\rangle \longrightarrow \sigma_2) \Rightarrow \sigma_1 = \sigma_2$
- $\forall c.P(c)$?

Structural Induction for Commands

Given the syntax of commands:

$$c \in Com ::= \textbf{skip} \mid x := a \mid c; c \mid \textbf{if } b \textbf{ then } c \textbf{ else } c \mid \textbf{while } b \textbf{ do } c$$

We have that structural induction is as follows:

$$\frac{\begin{array}{c} P(\textbf{skip}) \qquad \forall x, a.\, P(x := a) \\ \forall c_0, c_1.\, P(c_0) \wedge P(c_1) \Rightarrow P(c_0; c_1) \\ \forall b, c_0, c_1.\, P(c_0) \wedge P(c_1) \Rightarrow P(\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1) \\ \forall b, c.\, P(c) \Rightarrow P(\textbf{while } b \textbf{ do } c) \end{array}}{\forall c \in Com.\, P(c)}$$

Proving Determinacy

We have two base cases:

$$P(\textbf{skip})$$
$$\forall x,\, a.\, P(x := a)$$

and three inductive cases:

$$\forall c_0,\, c_1.\, P(c_0) \wedge P(c_1) \Rightarrow P(c_0;\, c_1)$$
$$\forall b,\, c_0,\, c_1.\, P(c_0) \wedge P(c_1) \Rightarrow P(\textbf{if}\ b\ \textbf{then}\ c_0\ \textbf{else}\ c_1)$$
$$\forall b,\, c.\, P(c) \Rightarrow P(\textbf{while}\ b\ \textbf{do}\ c)$$

Proving Determinacy

We have two base cases:

$$P(\textbf{skip})$$
$$\forall x, a.\, P(x := a)$$

and three inductive cases:

$$\forall c_0, c_1.\, P(c_0) \wedge P(c_1) \Rightarrow P(c_0; c_1)$$
$$\forall b, c_0, c_1.\, P(c_0) \wedge P(c_1) \Rightarrow P(\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1)$$
$$\forall b, c.\, P(c) \Rightarrow P(\textbf{while } b \textbf{ do } c)$$

The case for **while** b **do** c fails, due to the recursive definition of its semantics.
We need a more convenient proof principle: rule induction.

Program Equivalence

- Symbolic manipulation of programs is possible if we can guarantee that their semantics is preserved
- Useful to study program transformations (interpreters, compilers, code optimization, refactoring, etc)
- But, when are two commands equivalent?

Operational Equivalence

A first attempt, using the default initial memory σ_0:

$$c_1 \sim_{\mathrm{o}} c_2 \quad \triangleq \forall \sigma.\, (\langle c_1, \sigma_0 \rangle \longrightarrow \sigma \Leftrightarrow \langle c_2, \sigma_0 \rangle \longrightarrow \sigma)$$

Operational Equivalence

A first attempt, using the default initial memory σ_0:

$$c_1 \sim_o c_2 \quad \triangleq \forall \sigma. \left(\langle c_1, \sigma_0 \rangle \longrightarrow \sigma \Leftrightarrow \langle c_2, \sigma_0 \rangle \longrightarrow \sigma\right)$$

Consider the commands $x := y + 1$ and $x := 1$.

Operational Equivalence

A first attempt, using the default initial memory σ_0:

$$c_1 \sim_o c_2 \quad \triangleq \forall \sigma. \left(\langle c_1, \sigma_0 \rangle \longrightarrow \sigma \Leftrightarrow \langle c_2, \sigma_0 \rangle \longrightarrow \sigma\right)$$

Consider the commands $x := y + 1$ and $x := 1$.

- Clearly, $x := y + 1 \sim_o x := 1$

Operational Equivalence

A first attempt, using the default initial memory σ_0:

$$c_1 \sim_o c_2 \quad \triangleq \forall \sigma.\, (\langle c_1, \sigma_0 \rangle \longrightarrow \sigma \Leftrightarrow \langle c_2, \sigma_0 \rangle \longrightarrow \sigma)$$

Consider the commands $x := y + 1$ and $x := 1$.

- Clearly, $x := y + 1 \sim_o x := 1$
- Let $C[\bullet] \triangleq y := 1; [\bullet]$ be a context: an environment for commands.

Operational Equivalence

A first attempt, using the default initial memory σ_0:

$$c_1 \sim_o c_2 \quad \triangleq \forall \sigma.\, (\langle c_1, \sigma_0 \rangle \longrightarrow \sigma \Leftrightarrow \langle c_2, \sigma_0 \rangle \longrightarrow \sigma)$$

Consider the commands $x := y + 1$ and $x := 1$.

- Clearly, $x := y + 1 \sim_o x := 1$
- Let $C[\bullet] \triangleq y := 1; [\bullet]$ be a context: an environment for commands.
- Given $C[\bullet]$, we can "plug in" arbitrary commands in $\bullet$. We obtain:
  $C[x := y + 1] \triangleq y := 1; x := y + 1 \qquad C[x := 1] \triangleq y := 1; x := 1$

Operational Equivalence

A first attempt, using the default initial memory σ_0:

$$c_1 \sim_o c_2 \quad \triangleq \forall \sigma.\, (\langle c_1, \sigma_0 \rangle \longrightarrow \sigma \Leftrightarrow \langle c_2, \sigma_0 \rangle \longrightarrow \sigma)$$

Consider the commands $x := y + 1$ and $x := 1$.

- Clearly, $x := y + 1 \sim_o x := 1$
- Let $C[\bullet] \triangleq y := 1; [\bullet]$ be a context: an environment for commands.
- Given $C[\bullet]$, we can "plug in" arbitrary commands in $\bullet$. We obtain:
  $C[x := y + 1] \triangleq y := 1; x := y + 1 \qquad C[x := 1] \triangleq y := 1; x := 1$
- However, the resulting commands are not equivalent, as we have

$$y := 1; x := y + 1 \not\sim_o y := 1; x := 1$$

- There is a context that does not preserve the equivalence!

Operational Equivalence

Another attempt, using any initial memory :

$$c_1 \sim_{\mathrm{o}} c_2 \triangleq \forall \sigma, \sigma'. \left(\langle c_1, \sigma \rangle \longrightarrow \sigma' \Leftrightarrow \langle c_2, \sigma \rangle \longrightarrow \sigma'\right)$$

We have that this definition of $\sim_{\mathrm{o}}$ is

- an equivalence (reflexive, symmetric, transitive)
- a congruence: it is preserved by every context!

The End