Introduction to formal semantics of programming languages

ADA 2024/25

Dep. de Matemática Universidade de Aveiro
Alexandre Madeira
(madeira@ua.pt)

September 29, 2024

Outline

The imperative language While

The language While

For the purposes of this UC, we will introduce a prototype imperative programming language that contains the basic ingredients of any "standard" imperative programming language, without procedures or advanced data structures.

The imperative language While

Syntactical categories of While

Categogy	Domain	Meta-variables
Integers	$\mathbb{Z} = \{\cdots -1, 0, 1, \ldots\}$	z
Truth values	$\mathbb{B} = \{\texttt{true}, \texttt{false}\}$	t
Variables	$\texttt{Var} = \{x, y, \ldots\}$	x
Arithmetic expressions	`AExp`	a
Boolean expressions	`BExp`	b
commands	`Cmd`	c

The imperative language While

Arithmetic expressions AExp

$$a ::= z \mid x \mid a + a \mid a - a \mid a * a$$

Examples

- $3 + x * y$
- $x * ((y - 4) * 0)$
- ...

The imperative language While

Boolean expressions BExp

$$b ::= t \mid a = a \mid a > a \mid \neg b \mid b \wedge b \mid b \vee b,\ t \in \mathbb{B}$$

where $a \in$ AExp

Examples

- $\neg(x = 3) \vee (x = y)$
- $true \wedge (x = 3 \vee x = y)$
- ...

The imperative language While

Commands Cmd

$$c ::= \mathit{skip} \mid x := a \mid c; c \mid \texttt{if } b \texttt{ then } c \texttt{ else } c \mid \texttt{while } b \texttt{ do } c$$

where $a \in \texttt{AExp}$ and $b \in \texttt{BExp}$

Examples

- $\texttt{if } x > 0 \texttt{ then } x := 1 \texttt{ else } x := -x$
- $x := 3; (\texttt{if } x > 0 \texttt{ then } x := 1 \texttt{ else } x := -x)$
- $(\texttt{if } x > 0 \texttt{ then } x := 1 \texttt{ else } x := -x); x := 3$

The imperative language While

Exercise 1

Try to represent some other common commands (e.g. `repeat c until b` *) on means of the base programs of* `While`*.*

The imperative language While

Recommended readings

- book of Winskel, Chap 1 and Sec 2.1

Outline

To provide a formal interpretation to While

- Objective: given a `While` program, interpret it in a mathematical structure
- This is done in a **structural way**, using **inference rules** in a **natural deduction style**

$$\frac{\textbf{premisses}}{\textbf{conclusions}}\ \textbf{*conditions* (rule name)}$$

To provide a formal interpretation to While

- Objective: given a `While` program, interpret it in a mathematical structure
- This is done in a **structural way**, using **inference rules** in a **natural deduction style**

$$\frac{\textbf{premisses}}{\textbf{conclusions}}\ \textbf{*conditions* (rule name)}$$

- Let us introduce:
  - a notion of **state**
  - rules to **interpret arithmetic and Boolean expressions**
  - rules to **interpret commands**

(Recalling) natural deduction

Natural Deduction system for Propositional Logic

$$\frac{\varphi \quad \psi}{\varphi \wedge \psi}\wedge\mathrm{I} \qquad \frac{\varphi \wedge \psi}{\varphi}\wedge\mathrm{E}_1 \qquad \frac{\varphi \wedge \psi}{\psi}\wedge\mathrm{E}_2$$

$$\frac{\varphi}{\varphi \vee \psi}\vee\mathrm{I}_1 \qquad \frac{\psi}{\varphi \vee \psi}\vee\mathrm{I}_2 \qquad \frac{\varphi \vee \psi \quad \begin{array}{|c|}\hline \varphi \\ \vdots \\ \theta \\ \hline \end{array} \quad \begin{array}{|c|}\hline \psi \\ \vdots \\ \theta \\ \hline \end{array}}{\theta}\vee\mathrm{E}$$

$$\frac{\begin{array}{|c|}\hline \varphi \\ \vdots \\ \psi \\ \hline \end{array}}{\varphi \to \psi}\Rightarrow\mathrm{I} \qquad \frac{\varphi \quad \varphi \to \psi}{\psi}\Rightarrow\mathrm{E}$$

$$\frac{\bot}{\varphi}\bot\mathrm{E} \qquad \frac{}{\varphi \vee \neg\varphi}\mathrm{EM}$$

States

Definition 1

A program state (in the variables Var) is a function

$$\sigma : \texttt{Var} \to \mathbb{Z}$$

The state space of a program is the set

$$\Sigma = \{\sigma \mid \sigma : \texttt{Var} \to \mathbb{Z}\}$$

States

Definition 1

A program state (in the variables Var) is a function

$$\sigma : \texttt{Var} \to \mathbb{Z}$$

The state space of a program is the set

$$\Sigma = \{\sigma \mid \sigma : \texttt{Var} \to \mathbb{Z}\}$$

- $\sigma(x)$ denotes the value of the variable x in the state σ

Interpretation of Arithmetic expressions AExp

Interpretation of an expression $a \in \mathtt{AExp}$ in a state $\sigma \in \Sigma$

$$\langle a, \sigma \rangle \to z$$

means:

the expression a is evaluated in the state σ as z

Interpretation of Arithmetic expressions AExp

Axioms

$$\frac{}{\langle z, \sigma\rangle \to z} \qquad \frac{}{\langle x, \sigma\rangle \to \sigma(x)}$$

Interpretation of Arithmetic expressions AExp

Axioms

$$\frac{}{\langle z, \sigma\rangle \to z} \quad \frac{}{\langle x, \sigma\rangle \to \sigma(x)}$$

Inference Rules

$$\frac{\langle a_1, \sigma\rangle \to z_1 \quad \langle a_2, \sigma\rangle \to z_2}{\langle a_1 + a_2, \sigma\rangle \to z} \text{ where } z = z_1 + z_2$$

Interpretation of Arithmetic expressions AExp

Axioms

$$\frac{}{\langle z, \sigma\rangle \to z} \qquad \frac{}{\langle x, \sigma\rangle \to \sigma(x)}$$

Inference Rules

$$\frac{\langle a_1, \sigma\rangle \to z_1 \quad \langle a_2, \sigma\rangle \to z_2}{\langle a_1 + a_2, \sigma\rangle \to z} \text{ where } z = z_1 + z_2$$

$$\frac{\langle a_1, \sigma\rangle \to z_1 \quad \langle a_2, \sigma\rangle \to z_2}{\langle a_1 - a_2, \sigma\rangle \to z} \text{ where } z = z_1 - z_2$$

Interpretation of Arithmetic expressions AExp

Axioms

$$\frac{}{\langle z, \sigma\rangle \to z} \qquad \frac{}{\langle x, \sigma\rangle \to \sigma(x)}$$

Inference Rules

$$\frac{\langle a_1, \sigma\rangle \to z_1 \quad \langle a_2, \sigma\rangle \to z_2}{\langle a_1 + a_2, \sigma\rangle \to z} \text{ where } z = z_1 + z_2$$

$$\frac{\langle a_1, \sigma\rangle \to z_1 \quad \langle a_2, \sigma\rangle \to z_2}{\langle a_1 - a_2, \sigma\rangle \to z} \text{ where } z = z_1 - z_2$$

$$\frac{\langle a_1, \sigma\rangle \to z_1 \quad \langle a_2, \sigma\rangle \to z_2}{\langle a_1 * a_2, \sigma\rangle \to z} \text{ where } z = z_1 * z_2$$

Interpretation of Arithmetic expressions AExp

Axioms

$$\frac{}{\langle z, \sigma\rangle \to z} \qquad \frac{}{\langle x, \sigma\rangle \to \sigma(x)}$$

Inference Rules

$$\frac{\langle a_1, \sigma\rangle \to z_1 \quad \langle a_2, \sigma\rangle \to z_2}{\langle a_1 + a_2, \sigma\rangle \to z} \text{ where } z = z_1 + z_2$$

$$\frac{\langle a_1, \sigma\rangle \to z_1 \quad \langle a_2, \sigma\rangle \to z_2}{\langle a_1 - a_2, \sigma\rangle \to z} \text{ where } z = z_1 - z_2$$

$$\frac{\langle a_1, \sigma\rangle \to z_1 \quad \langle a_2, \sigma\rangle \to z_2}{\langle a_1 * a_2, \sigma\rangle \to z} \text{ where } z = z_1 * z_2$$

Example

Interpretation of expression $(x + y) - 1$ in a state σ such that $\sigma(x) = 1$ and $\sigma(y) = 0$

Interpretation of Arithmetic expressions AExp

Axioms

$$\frac{}{\langle z, \sigma\rangle \to z} \qquad \frac{}{\langle x, \sigma\rangle \to \sigma(x)}$$

Inference Rules

$$\frac{\langle a_1, \sigma\rangle \to z_1 \quad \langle a_2, \sigma\rangle \to z_2}{\langle a_1 + a_2, \sigma\rangle \to z} \text{ where } z = z_1 + z_2$$

$$\frac{\langle a_1, \sigma\rangle \to z_1 \quad \langle a_2, \sigma\rangle \to z_2}{\langle a_1 - a_2, \sigma\rangle \to z} \text{ where } z = z_1 - z_2$$

$$\frac{\langle a_1, \sigma\rangle \to z_1 \quad \langle a_2, \sigma\rangle \to z_2}{\langle a_1 * a_2, \sigma\rangle \to z} \text{ where } z = z_1 * z_2$$

Example

Interpretation of expression $(x + y) - 1$ in a state σ such that $\sigma(x) = 1$ and $\sigma(y) = 0$:

$$\frac{\dfrac{\dfrac{}{\langle x,\sigma\rangle \to 1}\,\sigma(x)=1 \quad \dfrac{}{\langle y,\sigma\rangle \to 0}\,\sigma(y)=0}{\langle x+y,\sigma\rangle \to 1}\,1 = 1 + 0 \quad \dfrac{}{\langle 1,\sigma\rangle \to 1}\,0 = 1 - 1}{\langle (x + y) - 1, \sigma\rangle \to 0}$$

Interpretation of Arithmetic expressions AExp

Exercise 2

1. *For* $\sigma(x) = 3$ *and* $\sigma(y) = 9$, *evaluate the expression* $(x + 3) * (y - 2)$ *in* σ
2. *For* $\sigma'(x) = 0$, *evaluate the expression* $(x + 5) + (7 + 9)$ *in* σ'

Interpretation of Arithmetic expressions AExp

Theorem 1

Let $a \in \mathtt{AExp}$ and $\sigma, \sigma' \in \Sigma$ such that, for any $x \in fv(a)$[a], $\sigma(x) = \sigma'(x)$. Then, for each $z \in \mathbb{Z}$,

$$\langle a, \sigma \rangle \to z \text{ iff } \langle a, \sigma' \rangle \to z$$

[a] The function $fv : \mathtt{AExp} \to \mathcal{P}(\mathtt{Var})$ that collects all the variable that occours in a expression, was defined in a previous class

Interpretation of Arithmetic expressions AExp

Theorem 1

Let $a \in \mathit{AExp}$ and $\sigma, \sigma' \in \Sigma$ such that, for any $x \in fv(a)$[a], $\sigma(x) = \sigma'(x)$. Then, for each $z \in \mathbb{Z}$,

$$\langle a, \sigma \rangle \to z \text{ iff } \langle a, \sigma' \rangle \to z$$

[a] The function $fv : \mathtt{AExp} \to \mathcal{P}(\mathtt{Var})$ that collects all the variable that occours in a expression, was defined in a previous class

Exercise 3

Prove the result using structural induction over AExp.

Interpretation of Boolean Expressions BExp

$$\texttt{BExp} \ni b ::= t \mid a = a \mid a > a \mid \neg b \mid b \wedge b \mid b \vee b,\ t \in \mathbb{B}; a \in \texttt{AExp}$$

Axioms

Interpretation of Boolean Expressions BExp

$$\texttt{BExp} \ni b ::= t \mid a = a \mid a > a \mid \neg b \mid b \wedge b \mid b \vee b,\ t \in \mathbb{B}; a \in \texttt{AExp}$$

Axioms $\frac{}{\langle t, \sigma \rangle \to t}$

Inference Rules

Interpretation of Boolean Expressions BExp

$$\texttt{BExp} \ni b ::= t \mid a = a \mid a > a \mid \neg b \mid b \wedge b \mid b \vee b,\ t \in \mathbb{B}; a \in \texttt{AExp}$$

Axioms $\frac{}{\langle t,\sigma\rangle \to t}$

Inference Rules

$$\frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 = a_2,\sigma\rangle \to \texttt{true}} \text{ if } z_1 = z_2 \qquad \frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 = a_2,\sigma\rangle \to \texttt{false}} \text{ if } z_1 \neq z_2$$

Interpretation of Boolean Expressions BExp

$$\texttt{BExp} \ni b ::= t \mid a = a \mid a > a \mid \neg b \mid b \wedge b \mid b \vee b,\ t \in \mathbb{B}; a \in \texttt{AExp}$$

Axioms $\frac{}{\langle t,\sigma\rangle \to t}$

Inference Rules

$$\frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 = a_2,\sigma\rangle \to \texttt{true}} \text{ if } z_1 = z_2 \qquad \frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 = a_2,\sigma\rangle \to \texttt{false}} \text{ if } z_1 \neq z_2$$

$$\frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 > a_2,\sigma\rangle \to \texttt{true}} \text{ if } z_1 > z_2 \qquad \frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 > a_2,\sigma\rangle \to \texttt{false}} \text{ se } z_1 \leq z_2$$

Interpretation of Boolean Expressions BExp

$$\texttt{BExp} \ni b ::= t \mid a = a \mid a > a \mid \neg b \mid b \wedge b \mid b \vee b,\ t \in \mathbb{B}; a \in \texttt{AExp}$$

Axioms $\dfrac{}{\langle t,\sigma\rangle \to t}$

Inference Rules

$$\frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 = a_2,\sigma\rangle \to \texttt{true}} \text{ if } z_1 = z_2 \qquad \frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 = a_2,\sigma\rangle \to \texttt{false}} \text{ if } z_1 \neq z_2$$

$$\frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 > a_2,\sigma\rangle \to \texttt{true}} \text{ if } z_1 > z_2 \qquad \frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 > a_2,\sigma\rangle \to \texttt{false}} \text{ se } z_1 \leq z_2$$

$$\frac{\langle b,\sigma\rangle \to \texttt{false}}{\langle \neg b,\sigma\rangle \to \texttt{true}} \qquad \frac{\langle b,\sigma\rangle \to \texttt{true}}{\langle \neg b,\sigma\rangle \to \texttt{false}}$$

Interpretation of Boolean Expressions BExp

$$\texttt{BExp} \ni b ::= t \mid a = a \mid a > a \mid \neg b \mid b \wedge b \mid b \vee b,\ t \in \mathbb{B}; a \in \texttt{AExp}$$

Axioms $\dfrac{}{\langle t,\sigma\rangle \to t}$

Inference Rules

$$\frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 = a_2,\sigma\rangle \to \texttt{true}} \text{ if } z_1 = z_2 \qquad \frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 = a_2,\sigma\rangle \to \texttt{false}} \text{ if } z_1 \neq z_2$$

$$\frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 > a_2,\sigma\rangle \to \texttt{true}} \text{ if } z_1 > z_2 \qquad \frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 > a_2,\sigma\rangle \to \texttt{false}} \text{ se } z_1 \leq z_2$$

$$\frac{\langle b,\sigma\rangle \to \texttt{false}}{\langle \neg b,\sigma\rangle \to \texttt{true}} \qquad \frac{\langle b,\sigma\rangle \to \texttt{true}}{\langle \neg b,\sigma\rangle \to \texttt{false}}$$

$$\frac{\langle b_1,\sigma\rangle \to \texttt{true} \quad \langle b_2,\sigma\rangle \to \texttt{true}}{\langle b_1 \wedge b_2,\sigma\rangle \to \texttt{true}} \qquad \frac{\langle b_1,\sigma\rangle \to \texttt{true} \quad \langle b_2,\sigma\rangle \to \texttt{false}}{\langle b_1 \wedge b_2,\sigma\rangle \to \texttt{false}}$$

Interpretation of Boolean Expressions BExp

$$\texttt{BExp} \ni b ::= t \mid a = a \mid a > a \mid \neg b \mid b \wedge b \mid b \vee b,\ t \in \mathbb{B}; a \in \texttt{AExp}$$

Axioms $\dfrac{}{\langle t,\sigma\rangle \to t}$

Inference Rules

$$\frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 = a_2,\sigma\rangle \to \texttt{true}} \text{ if } z_1 = z_2 \qquad \frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 = a_2,\sigma\rangle \to \texttt{false}} \text{ if } z_1 \neq z_2$$

$$\frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 > a_2,\sigma\rangle \to \texttt{true}} \text{ if } z_1 > z_2 \qquad \frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 > a_2,\sigma\rangle \to \texttt{false}} \text{ se } z_1 \leq z_2$$

$$\frac{\langle b,\sigma\rangle \to \texttt{false}}{\langle \neg b,\sigma\rangle \to \texttt{true}} \qquad \frac{\langle b,\sigma\rangle \to \texttt{true}}{\langle \neg b,\sigma\rangle \to \texttt{false}}$$

$$\frac{\langle b_1,\sigma\rangle \to \texttt{true} \quad \langle b_2,\sigma\rangle \to \texttt{true}}{\langle b_1 \wedge b_2,\sigma\rangle \to \texttt{true}} \qquad \frac{\langle b_1,\sigma\rangle \to \texttt{true} \quad \langle b_2,\sigma\rangle \to \texttt{false}}{\langle b_1 \wedge b_2,\sigma\rangle \to \texttt{false}}$$

$$\frac{\langle b_1,\sigma\rangle \to \texttt{false} \quad \langle b_2,\sigma\rangle \to \texttt{true}}{\langle b_1 \wedge b_2,\sigma\rangle \to \texttt{false}} \qquad \frac{\langle b_1,\sigma\rangle \to \texttt{false} \quad \langle b_2,\sigma\rangle \to \texttt{false}}{\langle b_1 \wedge b_2,\sigma\rangle \to \texttt{false}}$$

Interpretation of Boolean Expressions BExp

$$\texttt{BExp} \ni b ::= t \mid a = a \mid a > a \mid \neg b \mid b \wedge b \mid b \vee b,\ t \in \mathbb{B}; a \in \texttt{AExp}$$

Axioms $\dfrac{}{\langle t,\sigma\rangle \to t}$

Inference Rules

$$\frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 = a_2,\sigma\rangle \to \texttt{true}} \text{ if } z_1 = z_2 \qquad \frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 = a_2,\sigma\rangle \to \texttt{false}} \text{ if } z_1 \neq z_2$$

$$\frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 > a_2,\sigma\rangle \to \texttt{true}} \text{ if } z_1 > z_2 \qquad \frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 > a_2,\sigma\rangle \to \texttt{false}} \text{ se } z_1 \leq z_2$$

$$\frac{\langle b,\sigma\rangle \to \texttt{false}}{\langle \neg b,\sigma\rangle \to \texttt{true}} \qquad \frac{\langle b,\sigma\rangle \to \texttt{true}}{\langle \neg b,\sigma\rangle \to \texttt{false}}$$

$$\frac{\langle b_1,\sigma\rangle \to \texttt{true} \quad \langle b_2,\sigma\rangle \to \texttt{true}}{\langle b_1 \wedge b_2,\sigma\rangle \to \texttt{true}} \qquad \frac{\langle b_1,\sigma\rangle \to \texttt{true} \quad \langle b_2,\sigma\rangle \to \texttt{false}}{\langle b_1 \wedge b_2,\sigma\rangle \to \texttt{false}}$$

$$\frac{\langle b_1,\sigma\rangle \to \texttt{false} \quad \langle b_2,\sigma\rangle \to \texttt{true}}{\langle b_1 \wedge b_2,\sigma\rangle \to \texttt{false}} \qquad \frac{\langle b_1,\sigma\rangle \to \texttt{false} \quad \langle b_2,\sigma\rangle \to \texttt{false}}{\langle b_1 \wedge b_2,\sigma\rangle \to \texttt{false}}$$

$$\frac{\langle b_1,\sigma\rangle \to \texttt{true} \quad \langle b_2,\sigma\rangle \to \texttt{true}}{\langle b_1 \vee b_2,\sigma\rangle \to \texttt{true}} \qquad \frac{\langle b_1,\sigma\rangle \to \texttt{true} \quad \langle b_2,\sigma\rangle \to \texttt{false}}{\langle b_1 \vee b_2,\sigma\rangle \to \texttt{true}}$$

Interpretation of Boolean Expressions BExp

$$\texttt{BExp} \ni b ::= t \mid a = a \mid a > a \mid \neg b \mid b \wedge b \mid b \vee b,\ t \in \mathbb{B}; a \in \texttt{AExp}$$

Axioms $\dfrac{}{\langle t,\sigma\rangle \to t}$

Inference Rules

$$\frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 = a_2,\sigma\rangle \to \texttt{true}} \text{ if } z_1 = z_2 \qquad \frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 = a_2,\sigma\rangle \to \texttt{false}} \text{ if } z_1 \neq z_2$$

$$\frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 > a_2,\sigma\rangle \to \texttt{true}} \text{ if } z_1 > z_2 \qquad \frac{\langle a_1,\sigma\rangle \to z_1 \quad \langle a_2,\sigma\rangle \to z_2}{\langle a_1 > a_2,\sigma\rangle \to \texttt{false}} \text{ se } z_1 \leq z_2$$

$$\frac{\langle b,\sigma\rangle \to \texttt{false}}{\langle \neg b,\sigma\rangle \to \texttt{true}} \qquad \frac{\langle b,\sigma\rangle \to \texttt{true}}{\langle \neg b,\sigma\rangle \to \texttt{false}}$$

$$\frac{\langle b_1,\sigma\rangle \to \texttt{true} \quad \langle b_2,\sigma\rangle \to \texttt{true}}{\langle b_1 \wedge b_2,\sigma\rangle \to \texttt{true}} \qquad \frac{\langle b_1,\sigma\rangle \to \texttt{true} \quad \langle b_2,\sigma\rangle \to \texttt{false}}{\langle b_1 \wedge b_2,\sigma\rangle \to \texttt{false}}$$

$$\frac{\langle b_1,\sigma\rangle \to \texttt{false} \quad \langle b_2,\sigma\rangle \to \texttt{true}}{\langle b_1 \wedge b_2,\sigma\rangle \to \texttt{false}} \qquad \frac{\langle b_1,\sigma\rangle \to \texttt{false} \quad \langle b_2,\sigma\rangle \to \texttt{false}}{\langle b_1 \wedge b_2,\sigma\rangle \to \texttt{false}}$$

$$\frac{\langle b_1,\sigma\rangle \to \texttt{true} \quad \langle b_2,\sigma\rangle \to \texttt{true}}{\langle b_1 \vee b_2,\sigma\rangle \to \texttt{true}} \qquad \frac{\langle b_1,\sigma\rangle \to \texttt{true} \quad \langle b_2,\sigma\rangle \to \texttt{false}}{\langle b_1 \vee b_2,\sigma\rangle \to \texttt{true}}$$

$$\frac{\langle b_1,\sigma\rangle \to \texttt{false} \quad \langle b_2,\sigma\rangle \to \texttt{true}}{\langle b_1 \vee b_2,\sigma\rangle \to \texttt{true}} \qquad \frac{\langle b_1,\sigma\rangle \to \texttt{false} \quad \langle b_2,\sigma\rangle \to \texttt{false}}{\langle b_1 \vee b_2,\sigma\rangle \to \texttt{false}}$$

Interpretation of Boolean Expressions BExp

Exercise 4

For σ such that $\sigma(x) = 2$ and $\sigma(y) = 5$, interpret the Boolean expressions

1. $(y = 3) \vee (x = 2)$
2. $\neg(x = y)$

Interpretation of Boolean Expressions BExp

More efficient strategies to evaluate expressions?

To apply the rule

$$\frac{\langle b_1, \sigma\rangle \to \texttt{false} \quad \langle b_2, \sigma\rangle \to \texttt{true}}{\langle b_1 \wedge b_2, \sigma\rangle \to \texttt{false}}$$

it would be enough to evaluate b_1, i.e.

$$\frac{\langle b_1, \sigma\rangle \to \texttt{false}}{\langle b_1 \wedge b_2, \sigma\rangle \to \texttt{false}}$$

Exercise 5

Introduce a more efficient set of inference rules to evaluate BExp expressions

Interpretation of commands Cmd

Updates on states

Against of what happens with the interpretation of arithmetic and Boolean expressions, the commands execution change the states of a program. E.g.

$$\sigma(x) = 25 \xrightarrow{x := 3} \sigma'(x) = 3$$

Interpretation of commands Cmd

Updates on states

Against of what happens with the interpretation of arithmetic and Boolean expressions, the commands execution change the states of a program. E.g.

$$\sigma(x) = 25 \xrightarrow{x := 3} \sigma'(x) = 3$$

The **execution step**

$$\langle \texttt{x:=3}, \sigma \rangle \to \sigma'$$

transforms the state σ in a state σ' where, for any $y \in \texttt{Var}$,

$$\sigma'(y) = \begin{cases} 3 & y = x \\ \sigma(y) & y \neq x \end{cases}$$

Interpretation of commands Cmd

Notation

The expression

$$\langle c, \sigma \rangle \to \sigma'$$

denotes that the (full) execution of command c in state σ terminates in final state σ'.

Update function

Let $\sigma \in \Sigma$, $x, y \in \texttt{Var}$ and $z \in \mathbb{Z}$.

$$\sigma[x \leftarrow z](y) = \begin{cases} z & y = x \\ \sigma(y) & y \neq x \end{cases}$$

Interpretation of commands Cmd

Theorem 2

Let $z_1, z_2 \in \mathbb{Z}$ and $x, y \in \mathit{Var}$. Then,

$$(\sigma[x \leftarrow z_1])[x \leftarrow z_2] = \sigma[x \leftarrow z_2]$$

Proof.

Exercise! □

Interpretation of commands Cmd

Theorem 2

Let $z_1, z_2 \in \mathbb{Z}$ and $x, y \in \mathit{Var}$. Then,

$$(\sigma[x \leftarrow z_1])[x \leftarrow z_2] = \sigma[x \leftarrow z_2]$$

Proof.

Exercise! □

Corollary 2

Let $z_1, \ldots, z_n \in \mathbb{Z}$ and $x, y \in \mathit{Var}$. Then,

$$(\ldots((\sigma[x \leftarrow z_1])[x \leftarrow z_2])\ldots)[x \leftarrow z_n])(y) = \sigma[x \leftarrow z_n](y)$$

Interpretation of commands Cmd

$\texttt{Cmd} \ni c ::= \texttt{skip} \mid x := a \mid c; c \mid \texttt{if } b \texttt{ then } c \texttt{ else } c \mid \texttt{while } b \texttt{ do } c$

Commands interpretation

Axioms

Interpretation of commands Cmd

$\texttt{Cmd} \ni c ::= \texttt{skip} \mid x := a \mid c; c \mid \texttt{if } b \texttt{ then } c \texttt{ else } c \mid \texttt{while } b \texttt{ do } c$

Commands interpretation

Axioms

$(skip) \dfrac{}{\langle \texttt{skip}, \sigma \rangle \to \sigma}$

Inference rules

Interpretation of commands Cmd

$\texttt{Cmd} \ni c ::= \texttt{skip} \mid x := a \mid c; c \mid \texttt{if } b \texttt{ then } c \texttt{ else } c \mid \texttt{while } b \texttt{ do } c$

Commands interpretation

Axioms

$$(skip)\ \frac{}{\langle \texttt{skip}, \sigma \rangle \rightarrow \sigma}$$

Inference rules

$$(\text{assign})\ \frac{\langle a, \sigma \rangle \rightarrow z}{\langle x := a, \sigma \rangle \rightarrow \sigma[x \leftarrow z]}$$

Interpretation of commands Cmd

$\texttt{Cmd} \ni c ::= \texttt{skip} \mid x := a \mid c; c \mid \texttt{if } b \texttt{ then } c \texttt{ else } c \mid \texttt{while } b \texttt{ do } c$

Commands interpretation

Axioms

$$(skip)\ \frac{}{\langle \texttt{skip}, \sigma \rangle \rightarrow \sigma}$$

Inference rules

$$(\text{assign})\ \frac{\langle a, \sigma \rangle \rightarrow z}{\langle x := a, \sigma \rangle \rightarrow \sigma[x \leftarrow z]}$$

$$(\text{seq})\ \frac{\langle c_1, \sigma \rangle \rightarrow \sigma' \quad \langle c_2, \sigma' \rangle \rightarrow \sigma''}{\langle c_1; c_2, \sigma \rangle \rightarrow \sigma''}$$

Interpretation of commands Cmd

$\texttt{Cmd} \ni c ::= \texttt{skip} \mid x := a \mid c; c \mid \texttt{if } b \texttt{ then } c \texttt{ else } c \mid \texttt{while } b \texttt{ do } c$

Commands interpretation

Axioms

$$(skip)\ \frac{}{\langle \texttt{skip}, \sigma \rangle \to \sigma}$$

Inference rules

$$(\text{assign})\ \frac{\langle a, \sigma \rangle \to z}{\langle x := a, \sigma \rangle \to \sigma[x \leftarrow z]}$$

$$(\text{seq})\ \frac{\langle c_1, \sigma \rangle \to \sigma' \quad \langle c_2, \sigma' \rangle \to \sigma''}{\langle c_1; c_2, \sigma \rangle \to \sigma''}$$

$$(\text{if-t})\ \frac{\langle b, \sigma \rangle \to \texttt{true} \quad \langle c_1, \sigma \rangle \to \sigma'}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma \rangle \to \sigma'}$$

Interpretation of commands Cmd

$\texttt{Cmd} \ni c ::= \texttt{skip} \mid x := a \mid c; c \mid \texttt{if } b \texttt{ then } c \texttt{ else } c \mid \texttt{while } b \texttt{ do } c$

Commands interpretation

Axioms

$$(skip)\ \frac{}{\langle \texttt{skip}, \sigma \rangle \to \sigma}$$

Inference rules

$$(\text{assign})\ \frac{\langle a, \sigma \rangle \to z}{\langle x := a, \sigma \rangle \to \sigma[x \leftarrow z]}$$

$$(\text{seq})\ \frac{\langle c_1, \sigma \rangle \to \sigma' \quad \langle c_2, \sigma' \rangle \to \sigma''}{\langle c_1; c_2, \sigma \rangle \to \sigma''}$$

$$(\text{if-t})\ \frac{\langle b, \sigma \rangle \to \texttt{true} \quad \langle c_1, \sigma \rangle \to \sigma'}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma \rangle \to \sigma'} \qquad (\text{if-f})\ \frac{\langle b, \sigma \rangle \to \texttt{false} \quad \langle c_2, \sigma \rangle \to \sigma'}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma \rangle \to \sigma'}$$

Interpretation of commands Cmd

$\texttt{Cmd} \ni c ::= \texttt{skip} \mid x := a \mid c; c \mid \texttt{if } b \texttt{ then } c \texttt{ else } c \mid \texttt{while } b \texttt{ do } c$

Commands interpretation

Axioms

$$(skip)\ \frac{}{\langle \texttt{skip}, \sigma \rangle \to \sigma}$$

Inference rules

$$(\text{assign})\ \frac{\langle a, \sigma \rangle \to z}{\langle x := a, \sigma \rangle \to \sigma[x \leftarrow z]}$$

$$(\text{seq})\ \frac{\langle c_1, \sigma \rangle \to \sigma' \quad \langle c_2, \sigma' \rangle \to \sigma''}{\langle c_1; c_2, \sigma \rangle \to \sigma''}$$

$$(\text{if-t})\ \frac{\langle b, \sigma \rangle \to \texttt{true} \quad \langle c_1, \sigma \rangle \to \sigma'}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma \rangle \to \sigma'} \qquad (\text{if-f})\ \frac{\langle b, \sigma \rangle \to \texttt{false} \quad \langle c_2, \sigma \rangle \to \sigma'}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma \rangle \to \sigma'}$$

$$(\text{wh-t})\ \frac{\langle b, \sigma \rangle \to \texttt{true} \quad \langle c, \sigma \rangle \to \sigma' \quad \langle \texttt{while } b \texttt{ do } c, \sigma' \rangle \to \sigma''}{\langle \texttt{while } b \texttt{ do } c, \sigma \rangle \to \sigma''}$$

Interpretation of commands Cmd

$\texttt{Cmd} \ni c ::= \texttt{skip} \mid x := a \mid c; c \mid \texttt{if } b \texttt{ then } c \texttt{ else } c \mid \texttt{while } b \texttt{ do } c$

Commands interpretation

Axioms

$$(skip)\ \frac{}{\langle \texttt{skip}, \sigma \rangle \to \sigma}$$

Inference rules

$$(\text{assign})\ \frac{\langle a, \sigma \rangle \to z}{\langle x := a, \sigma \rangle \to \sigma[x \leftarrow z]}$$

$$(\text{seq})\ \frac{\langle c_1, \sigma \rangle \to \sigma' \quad \langle c_2, \sigma' \rangle \to \sigma''}{\langle c_1; c_2, \sigma \rangle \to \sigma''}$$

$$(\text{if-t})\ \frac{\langle b, \sigma \rangle \to \texttt{true} \quad \langle c_1, \sigma \rangle \to \sigma'}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma \rangle \to \sigma'} \qquad (\text{if-f})\ \frac{\langle b, \sigma \rangle \to \texttt{false} \quad \langle c_2, \sigma \rangle \to \sigma'}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma \rangle \to \sigma'}$$

$$(\text{wh-t})\ \frac{\langle b, \sigma \rangle \to \texttt{true} \quad \langle c, \sigma \rangle \to \sigma' \quad \langle \texttt{while } b \texttt{ do } c, \sigma' \rangle \to \sigma''}{\langle \texttt{while } b \texttt{ do } c, \sigma \rangle \to \sigma''}$$

$$(\text{wh-f})\ \frac{\langle b, \sigma \rangle \to \texttt{false}}{\langle \texttt{while } b \texttt{ do } c, \sigma \rangle \to \sigma}$$

Interpretation of commands Cmd

Example

Interpretation of program $x := 1;\ y := 3$ in a state σ

Interpretation of commands Cmd

Example

Interpretation of program $x := 1;\ y := 3$ in a state σ

$$\dfrac{\dfrac{\overline{\langle 1,\sigma\rangle \to 1}}{\langle x := 1,\sigma\rangle \to \sigma[x \leftarrow 1]}(assign) \quad \dfrac{\overline{\langle 3,\sigma[x\leftarrow 1]\rangle \to 3}}{\langle y := 3,\sigma[x\leftarrow 3]\rangle \to \sigma[x\leftarrow 1][y\leftarrow 3]}(assign)}{\langle x := 1;\ y := 3,\sigma\rangle \to \sigma[x \leftarrow 1][y \leftarrow 3]}(seq)$$

Interpretation of commands Cmd

Example

Interpretation of program $x := 1;\ y := 3$ in a state σ

$$\frac{\dfrac{\overline{\langle 1,\sigma\rangle \to 1}}{\langle x := 1,\sigma\rangle \to \sigma[x \leftarrow 1]}(assign) \quad \dfrac{\overline{\langle 3,\sigma[x\leftarrow 1]\rangle \to 3}}{\langle y := 3,\sigma[x\leftarrow 3]\rangle \to \sigma[x\leftarrow 1][y\leftarrow 3]}(assign)}{\langle x := 1;\ y := 3,\sigma\rangle \to \sigma[x \leftarrow 1][y \leftarrow 3]}(seq)$$

Exercise 6

Interpret the programs

$$if\ (2 = 3)\ then\ (x := 1;\ y := 3)\ else\ skip$$

and

$$if\ (2 = 3)\ then\ skip\ else\ (x := 1;\ y := 3)$$

Exercises

Exercise 7

Extend the base language `While` *with an operator*

$$\texttt{repeat } c \texttt{ until } b$$

It shall behave as expected (see the executions of $\omega \equiv \texttt{repeat } (x := x + 1) \texttt{ until } (x > 2)$

$$\sigma_0(x) = 1 \xrightarrow{\omega} \sigma_1(x) = 2 \xrightarrow{\omega} \sigma_2(x) = 3$$

$$\sigma_0(x) = 5 \xrightarrow{\omega} \sigma_1(x) = 6$$

1. *Extend the set of inference rules for this new command.*
2. *Derive the executions of* $\omega \equiv \texttt{repeat } (x := x + 1) \texttt{ until } (x > 2)$ *from a state* σ *such that* $\sigma(x) = 1$ *and from a state* σ' *such that* $\sigma'(x) = 5$.

Proving generic properties about programs

Definition 3 (Operational Equivalence)

Let $c, c' \in \texttt{Cmd}$. The programs c and c' are **operationally equivalent**, in symbols $c \sim c'$, iff for any $\sigma \in \Sigma$,

$$\langle c, \sigma \rangle \to \sigma' \text{ iff } \langle c', \sigma \rangle \to \sigma'$$

Proving generic properties about programs

Definition 3 (Operational Equivalence)

Let $c, c' \in \texttt{Cmd}$. The programs c and c' are **operationally equivalent**, in symbols $c \sim c'$, iff for any $\sigma \in \Sigma$,

$$\langle c, \sigma \rangle \to \sigma' \text{ iff } \langle c', \sigma \rangle \to \sigma'$$

Exercise 8

Show that

- $c_1; (c_2; c_3) \sim (c_1; c_2); c_3$
- $c_1; c_2 \not\sim c_2; c_1$
- $(\ x\ :=y;\ x\ :=1) \sim (\ x\ :=1)$
- $(\texttt{repeat}\ c\ \ \texttt{until}\ \ b\ ) \not\sim (\texttt{while}\ \neg b\ \texttt{do}\ c\ )$, *by assuming the semantic rules for* $\texttt{repeat}\ b\ \ \texttt{until}\ \ c$ *introduced in the previous Exercise 7*

Proving generic properties about programs

Theorem 3

Let $\omega \equiv \texttt{while}\ b\ \texttt{do}\ c$. *Then*

$$\omega \sim \texttt{if}\ b\ \texttt{then}\ (c; \omega)\ \texttt{else}\ \texttt{skip}$$

Proof.

Exercise! □

Proving generic properties about programs

Exercise 9

Let us revisit the inference rules for the command

$$\texttt{repeat}\ c\ \texttt{until}\ b$$

suggested in Exercise 7.

- *Define now this operator using the operators of* `While`

Proving generic properties about programs

Exercise 9

Let us revisit the inference rules for the command

$$\texttt{repeat}\ c\ \ \texttt{until}\ \ b$$

suggested in Exercise 7.

- *Define now this operator using the operators of* While
- *Prove that* $(\texttt{repeat}\ c\ \ \texttt{until}\ \ b) \sim (c; \texttt{while}\ \neg b\ \texttt{do}\ c)$ *using the rules introduced in the Exercise 7*

Proving generic properties about programs

Theorem 4

The execution of programs `While` *is deterministic, i.e. for any* $c \in \mathtt{Cmd}$ *and for any states* $\sigma, \sigma', \sigma'' \in \Sigma$,

$$\text{if } \langle c, \sigma \rangle \to \sigma' \text{ and } \langle c, \sigma \rangle \to \sigma'' \text{ then } \sigma' = \sigma''$$

About termination of programs

Exercise 10

Try to derive the program

$$\texttt{while true do skip}$$

About termination of programs

Exercise 10

Try to derive the program

$$\texttt{while true do skip}$$

There exist σ *and* σ' *such that*

$$\langle \texttt{while true do skip} , \sigma \rangle \to \sigma' ?$$

Big Steps Semantics Operational functional

Definition 4

The **functional of the operational semantics**

$$\mathfrak{B}[\![\_]\!] : \mathtt{Cmd} \to (\Sigma \dashrightarrow \Sigma)$$

assigns to every statement $c \in \mathtt{Cmd}$, a partial function on states which is defined as follows:

$$\mathfrak{B}[\![c]\!] : \Sigma \dashrightarrow \Sigma$$

$$\mathfrak{B}[\![c]\!](\sigma) = \begin{cases} \sigma' & \langle c, \sigma \rangle \to \sigma' \\ \mathit{undefined} & \text{otherwise} \end{cases}$$

Exercises

Exercise 11

Determine

$$\mathfrak{B}[\![\texttt{while true do skip}]\!]$$

Exercise 12

For $\sigma \in \Sigma$, determine

- $\mathfrak{B}[\![z := x;\ x := y;\ y := z]\!](\sigma)$

Exercises

Exercise 11

Determine

$$\mathfrak{B}[\![\texttt{while true do skip}]\!]$$

Exercise 12

For $\sigma \in \Sigma$, determine

- $\mathfrak{B}[\![z := x;\ x := y;\ y := z]\!](\sigma)$

Let σ_1 such that $\sigma_1(x) = 3$ and $\sigma_1(y) = 4$. Then, determine:

- $\mathfrak{B}[\![z := x;\ x := y;\ y := z]\!](\sigma_1)(x)$ *and*
- $\mathfrak{B}[\![z := x;\ x := y;\ y := z]\!](\sigma_1)(y)$

Exercises

Exercise 11

Determine

$$\mathfrak{B}[\![\mathit{while\ true\ do\ skip}]\!]$$

Exercise 12

For $\sigma \in \Sigma$, determine

- $\mathfrak{B}[\![z := x;\ x := y;\ y := z]\!](\sigma)$

Let σ_1 such that $\sigma_1(x) = 3$ and $\sigma_1(y) = 4$. Then, determine:

- $\mathfrak{B}[\![z := x;\ x := y;\ y := z]\!](\sigma_1)(x)$ *and*
- $\mathfrak{B}[\![z := x;\ x := y;\ y := z]\!](\sigma_1)(y)$

Exercise 13

Let $\sigma \in \Sigma$ such that $\sigma(x) = 3$. Calculate:
$\mathfrak{B}[\![y := 1; \mathit{while}\ \neg(x = 1)\ \mathit{do}\ (\ y := y \times x;\ x := x - 1\ )]\!](\sigma)(x)$

Outline

Small-steps Operational Semantics

Big-steps Vs Small-steps Operational Semantics

- Big steps semantics describes the effect of the execution of a complete programa c
- Small steps semantics emphasizes the individual steps of such executions

Big Steps Semantics

$$\langle c, \sigma \rangle \to \sigma'$$

Small Steps Semantics

$$\langle c, \sigma \rangle \Rightarrow \langle c', \sigma' \rangle$$

Small-steps Operational Semantics

$\texttt{Cmd} \ni c ::= \texttt{skip} \mid x := a \mid c; c \mid \texttt{if } b \texttt{ then } c \texttt{ else } c \mid \texttt{while } b \texttt{ do } c$

Inference Rules

$$\text{(assign)} \frac{\langle a,\sigma\rangle \to z}{\langle x := a,\sigma\rangle \Rightarrow \sigma[x \leftarrow z]} \qquad \text{(skip)} \frac{}{\langle \texttt{skip},\sigma\rangle \Rightarrow \sigma}$$

Small-steps Operational Semantics

$\texttt{Cmd} \ni c ::= \texttt{skip} \mid x := a \mid c; c \mid \texttt{if } b \texttt{ then } c \texttt{ else } c \mid \texttt{while } b \texttt{ do } c$

Inference Rules

(assign) $\dfrac{\langle a,\sigma\rangle \to z}{\langle x := a,\sigma\rangle \Rightarrow \sigma[x \leftarrow z]}$ (skip) $\dfrac{}{\langle \texttt{skip},\sigma\rangle \Rightarrow \sigma}$

(seq 1) $\dfrac{\langle c_1,\sigma_1\rangle \Rightarrow \sigma_2}{\langle c_1;c_2,\sigma_1\rangle \Rightarrow \langle c_2,\sigma_2\rangle}$ (seq 2) $\dfrac{\langle c_1,\sigma_1\rangle \Rightarrow \langle c_1',\sigma_2\rangle}{\langle c_1;c_2,\sigma_1\rangle \Rightarrow \langle c_1';c_2,\sigma_2\rangle}$

Small-steps Operational Semantics

$\texttt{Cmd} \ni c ::= \texttt{skip} \mid x := a \mid c; c \mid \texttt{if } b \texttt{ then } c \texttt{ else } c \mid \texttt{while } b \texttt{ do } c$

Inference Rules

(assign) $\dfrac{\langle a,\sigma\rangle \to z}{\langle x := a,\sigma\rangle \Rightarrow \sigma[x \leftarrow z]}$ (skip) $\dfrac{}{\langle \texttt{skip},\sigma\rangle \Rightarrow \sigma}$

(seq 1) $\dfrac{\langle c_1,\sigma_1\rangle \Rightarrow \sigma_2}{\langle c_1;c_2,\sigma_1\rangle \Rightarrow \langle c_2,\sigma_2\rangle}$ (seq 2) $\dfrac{\langle c_1,\sigma_1\rangle \Rightarrow \langle c_1',\sigma_2\rangle}{\langle c_1;c_2,\sigma_1\rangle \Rightarrow \langle c_1';c_2,\sigma_2\rangle}$

(if-t) $\dfrac{\langle b,\sigma\rangle \to \texttt{true}}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2,\sigma\rangle \Rightarrow \langle c_1,\sigma\rangle}$

Small-steps Operational Semantics

$\texttt{Cmd} \ni c ::= \texttt{skip} \mid x := a \mid c; c \mid \texttt{if } b \texttt{ then } c \texttt{ else } c \mid \texttt{while } b \texttt{ do } c$

Inference Rules

(assign) $\dfrac{\langle a,\sigma\rangle \to z}{\langle x := a,\sigma\rangle \Rightarrow \sigma[x \leftarrow z]}$ (skip) $\dfrac{}{\langle \texttt{skip},\sigma\rangle \Rightarrow \sigma}$

(seq 1) $\dfrac{\langle c_1,\sigma_1\rangle \Rightarrow \sigma_2}{\langle c_1;c_2,\sigma_1\rangle \Rightarrow \langle c_2,\sigma_2\rangle}$ (seq 2) $\dfrac{\langle c_1,\sigma_1\rangle \Rightarrow \langle c_1',\sigma_2\rangle}{\langle c_1;c_2,\sigma_1\rangle \Rightarrow \langle c_1';c_2,\sigma_2\rangle}$

(if-t) $\dfrac{\langle b,\sigma\rangle \to \texttt{true}}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2,\sigma\rangle \Rightarrow \langle c_1,\sigma\rangle}$ (if-f) $\dfrac{\langle b,\sigma\rangle \to \texttt{false}}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2,\sigma\rangle \Rightarrow \langle c_2,\sigma\rangle}$

Small-steps Operational Semantics

$$\texttt{Cmd} \ni c ::= \texttt{skip} \mid x := a \mid c; c \mid \texttt{if } b \texttt{ then } c \texttt{ else } c \mid \texttt{while } b \texttt{ do } c$$

Inference Rules

(assign) $\dfrac{\langle a,\sigma\rangle \to z}{\langle x := a,\sigma\rangle \Rightarrow \sigma[x \leftarrow z]}$ (skip) $\dfrac{}{\langle \texttt{skip},\sigma\rangle \Rightarrow \sigma}$

(seq 1) $\dfrac{\langle c_1,\sigma_1\rangle \Rightarrow \sigma_2}{\langle c_1;c_2,\sigma_1\rangle \Rightarrow \langle c_2,\sigma_2\rangle}$ (seq 2) $\dfrac{\langle c_1,\sigma_1\rangle \Rightarrow \langle c_1',\sigma_2\rangle}{\langle c_1;c_2,\sigma_1\rangle \Rightarrow \langle c_1';c_2,\sigma_2\rangle}$

(if-t) $\dfrac{\langle b,\sigma\rangle \to \texttt{true}}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2,\sigma\rangle \Rightarrow \langle c_1,\sigma\rangle}$ (if-f) $\dfrac{\langle b,\sigma\rangle \to \texttt{false}}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2,\sigma\rangle \Rightarrow \langle c_2,\sigma\rangle}$

(while) $\dfrac{}{\langle \texttt{while } b \texttt{ do } c ,\sigma\rangle \Rightarrow \langle \texttt{if } b \texttt{ then } (c;\texttt{while } b \texttt{ do } c) \texttt{ else skip},\sigma\rangle}$

Example 5

The derivation of the program $x := 1;\ y := 2;\ z := 3$ from a given σ:

Example 5

The derivation of the program $x := 1;\ y := 2;\ z := 3$ from a given σ:

$$(1)\ \frac{\dfrac{\dfrac{\langle 1,\sigma\rangle \to 1}{\langle\ x := 1,\sigma\rangle\ \Rightarrow\ \sigma[x \leftarrow 1]}}{\langle\ x := 1;\ y := 2,\sigma\rangle \Rightarrow \langle\ y := 2,\sigma[x \leftarrow 1]\rangle}}{\langle\ x\ := 1;\ y\ := 2;\ z\ := 3, \sigma\rangle\ \Rightarrow\ \langle\ y\ := 2;\ z\ := 3, \sigma[x \leftarrow 1]\rangle}$$

Example 5

The derivation of the program $x := 1;\ y := 2;\ z := 3$ from a given σ:

$$(1)\ \frac{\dfrac{\dfrac{\langle 1,\sigma\rangle \to 1}{\langle\ x := 1,\sigma\rangle\ \Rightarrow\ \sigma[x\leftarrow 1]}}{\langle\ x := 1;\ y := 2,\sigma\rangle \Rightarrow \langle\ y := 2,\sigma[x\leftarrow 1]\rangle}}{\langle\ x := 1;\ y := 2;\ z := 3,\sigma\rangle\ \Rightarrow\ \langle\ y := 2;\ z := 3,\sigma[x \leftarrow 1]\rangle}$$

$$(2)\ \frac{\dfrac{\langle 2,\sigma[x\leftarrow 1]\rangle \to 2}{\langle\ y := 2,\sigma[x\leftarrow 1]\rangle \Rightarrow \langle\sigma[x\leftarrow 1][y\leftarrow 2]\rangle}}{\langle\ y := 2;\ z := 3,\sigma[x \leftarrow 1]\rangle\ \Rightarrow\ \langle\ z := 3,\sigma[x \leftarrow 1][y \leftarrow 2]\rangle}$$

Example 5

The derivation of the program $x := 1;\ y := 2;\ z := 3$ from a given σ:

$$(1)\ \frac{\dfrac{\dfrac{\langle 1,\sigma\rangle \to 1}{\langle\ x := 1,\sigma\rangle\ \Rightarrow\ \sigma[x\leftarrow 1]}}{\langle\ x := 1;\ y := 2,\sigma\rangle\ \Rightarrow\ \langle\ y := 2,\sigma[x\leftarrow 1]\rangle}}{\langle\ x := 1;\ y := 2;\ z := 3,\sigma\rangle\ \Rightarrow\ \langle\ y := 2;\ z := 3,\sigma[x \leftarrow 1]\rangle}$$

$$(2)\ \frac{\dfrac{\dfrac{\langle 2,\sigma[x\leftarrow 1]\rangle \to 2}{\langle\ y := 2,\sigma[x\leftarrow 1]\rangle\ \Rightarrow\ \langle\sigma[x\leftarrow 1][y\leftarrow 2]\rangle}}{\langle\ y := 2;\ z := 3,\sigma[x \leftarrow 1]\rangle\ \Rightarrow\ \langle\ z := 3,\sigma[x \leftarrow 1][y \leftarrow 2]\rangle}}{}$$

$$(3)\ \frac{\dfrac{\dfrac{\langle 3,\sigma[x\leftarrow 1][y\leftarrow 2]\rangle \to 3}{\langle\ z := 3,\sigma[x\leftarrow 1][y\leftarrow 2]\ \Rightarrow\ \sigma[x\leftarrow 1][y\leftarrow 2][z\leftarrow 3]}}{\langle\ z := 3,\sigma[x \leftarrow 1][y \leftarrow 2]\rangle\ \Rightarrow\ \sigma[x \leftarrow 1][y \leftarrow 2][z \leftarrow 3]}}{}$$

Example 5

The derivation of the program $x := 1;\ y := 2;\ z := 3$ from a given σ:

$$(1)\ \frac{\dfrac{\dfrac{\langle 1,\sigma\rangle \to 1}{\langle\ x := 1,\sigma\rangle\ \Rightarrow\ \sigma[x\leftarrow 1]}}{\langle\ x := 1;\ y := 2,\sigma\rangle\ \Rightarrow\ \langle\ y := 2,\sigma[x\leftarrow 1]\rangle}}{\langle\ x := 1;\ y := 2;\ z := 3,\sigma\rangle\ \Rightarrow\ \langle\ y := 2;\ z := 3,\sigma[x \leftarrow 1]\rangle}$$

$$(2)\ \frac{\dfrac{\dfrac{\langle 2,\sigma[x\leftarrow 1]\rangle \to 2}{\langle\ y := 2,\sigma[x\leftarrow 1]\rangle\ \Rightarrow\ \langle\sigma[x\leftarrow 1][y\leftarrow 2]\rangle}}{\langle\ y := 2;\ z := 3,\sigma[x \leftarrow 1]\rangle\ \Rightarrow\ \langle\ z := 3,\sigma[x \leftarrow 1][y \leftarrow 2]\rangle}}{}$$

$$(3)\ \frac{\dfrac{\dfrac{\langle 3,\sigma[x\leftarrow 1][y\leftarrow 2]\rangle \to 3}{\langle\ z := 3,\sigma[x\leftarrow 1][y\leftarrow 2]\ \Rightarrow\ \sigma[x\leftarrow 1][y\leftarrow 2][z\leftarrow 3]}}{\langle\ z := 3,\sigma[x \leftarrow 1][y \leftarrow 2]\rangle\ \Rightarrow\ \sigma[x \leftarrow 1][y \leftarrow 2][z \leftarrow 3]}}{}$$

Therefore

$$\begin{aligned}
\langle\ x := 1;\ y := 2;\ z := 3,\sigma\rangle\ &\Rightarrow\ \langle\ y := 2;\ z := 3,\sigma[x \leftarrow 1]\rangle && (1)\\
&\Rightarrow\ \langle\ z := 3,\sigma[x \leftarrow 1][y \leftarrow 2]\rangle && (2)\\
&\Rightarrow\ \sigma[x \leftarrow 1][y \leftarrow 2][z \leftarrow 3] && (3)
\end{aligned}$$

Small-steps operational semantics

Exercise 14

From a state σ such that $\sigma(x) = 3$ derive the following programs:

- $\mathit{while}\ x < 5\ \mathit{do}\ x := x + 1$
- $y := 1; \mathit{while}\ \neg(x = 1)\ \mathit{do}\ (\ y := y \times x;\ x := x - 1\ )$
- $\mathit{while}\ \mathit{true}\ \mathit{do}\ \mathit{skip}$

Small-step Operational Semantics

Sequences and Derivation Sequences

A (possible infinite) set of configurations $\gamma_0, \gamma_1, \ldots$, such that $\gamma_0 = \langle c_0, \sigma_0 \rangle$, $\gamma_i = \langle c, \sigma \rangle$ for any $0 \leq i$ is called a **sequence of c in σ**.
A **derivation sequence of c in σ** is a sequence of c in σ that either

- the sequence is finite (of length $n \geq 0$), and γ_n is terminal, or
- the sequence is infinite.

Small-step Operational Semantics

Sequences and Derivation Sequences

A (possible infinite) set of configurations $\gamma_0, \gamma_1, \ldots$, such that $\gamma_0 = \langle c_0, \sigma_0 \rangle$, $\gamma_i = \langle c, \sigma \rangle$ for any $0 \leq i$ is called a **sequence of c in σ**.
A **derivation sequence of c in σ** is a sequence of c in σ that either

- the sequence is finite (of length $n \geq 0$), and γ_n is terminal, or
- the sequence is infinite.

We write:

- $\gamma \Rightarrow^i \gamma'$ is there is a sequence of length i from γ to γ'
- $\gamma \Rightarrow^* \gamma'$ is there is a finite sequence from γ to γ'

Properties of sequences

Note that

- $\gamma \Rightarrow \gamma'$ do not necessarily represents a derivations sequence
- for each step of the sequence, there is a derivation tree
- for each c and σ it is always possible to find (exactly) one derivation tree

[1] In `While` language, if terminates, terminates successfully. This is not always the case, if we consider some extensions in the language

Properties of sequences

Note that

- $\gamma \Rightarrow \gamma'$ do not necessarily represents a derivations sequence
- for each step of the sequence, there is a derivation tree
- for each c and σ it is always possible to find (exactly) one derivation tree

The execution of a c in σ:

- **loops** iff there is an infinite derivation sequence of c in σ
- **terminates** iff there is a finite derivation sequence of c in σ[1]
- **terminates successfully** iff there is a σ' such that $\langle c, \sigma \rangle \Rightarrow \sigma'$

[1] In `While` language, if terminates, terminates successfully. This is not always the case, if we consider some extensions in the language

A new proof strategy

Induction on the length of the derivation sequences

1. Prove that the property holds for all derivation sequences of length 0.
2. Prove that the property holds for all other derivation sequences: Assume that the property holds for all derivation sequences of length at most k (this is called the induction hypothesis) and show that it holds for derivation sequences of length $k + 1$.

The small-steps semantics functional

Lemma 6

If $\langle c_1; c_2, \sigma\rangle \Rightarrow^k \sigma'$, *then, there exists a state* σ_1 *and numbers* $k_1, k_2 \in \mathbb{N}$ *such that* $\langle c_1, \sigma\rangle \Rightarrow^{k_1} \sigma_1$, $\langle c_2, \sigma_1\rangle \Rightarrow^{k_2} \sigma'$ *and* $k = k_2 + k_2$.

The small-steps semantics functional

Lemma 6

If $\langle c_1; c_2, \sigma\rangle \Rightarrow^k \sigma'$*, then, there exists a state* σ_1 *and numbers* $k_1, k_2 \in \mathbb{N}$ *such that* $\langle c_1, \sigma\rangle \Rightarrow^{k_1} \sigma_1$, $\langle c_2, \sigma_1\rangle \Rightarrow^{k_2} \sigma'$ *and* $k = k_2 + k_2$.

Proof.

Exercise! (Proof by induction on the length of the derivation sequences) □

The small-steps semantics functional

Lemma 6

If $\langle c_1; c_2, \sigma\rangle \Rightarrow^k \sigma'$*, then, there exists a state* σ_1 *and numbers* $k_1, k_2 \in \mathbb{N}$ *such that* $\langle c_1, \sigma\rangle \Rightarrow^{k_1} \sigma_1$, $\langle c_2, \sigma_1\rangle \Rightarrow^{k_2} \sigma'$ *and* $k = k_2 + k_2$.

Proof.

Exercise! (Proof by induction on the length of the derivation sequences) □

Lemma 7

If $\langle c_1, \sigma_1\rangle \Rightarrow^k \sigma_2$ *then* $\langle c_1; c_2, \sigma_1\rangle \Rightarrow^k \langle c_2, \sigma_2\rangle$

The small-steps semantics functional

Lemma 6

If $\langle c_1; c_2, \sigma\rangle \Rightarrow^k \sigma'$*, then, there exists a state* σ_1 *and numbers* $k_1, k_2 \in \mathbb{N}$ *such that* $\langle c_1, \sigma\rangle \Rightarrow^{k_1} \sigma_1$, $\langle c_2, \sigma_1\rangle \Rightarrow^{k_2} \sigma'$ *and* $k = k_2 + k_2$.

Proof.

Exercise! (Proof by induction on the length of the derivation sequences) □

Lemma 7

If $\langle c_1, \sigma_1\rangle \Rightarrow^k \sigma_2$ *then* $\langle c_1; c_2, \sigma_1\rangle \Rightarrow^k \langle c_2, \sigma_2\rangle$

Small steps programs equivalence

Definition 8

The programs c_1 and c_2 are **semantically equivalent**, in symbols $c_1 \approx c_2$, if for any state $\sigma \in \Sigma$,

- $\langle c_1, \sigma \rangle \Rightarrow^* \gamma$ iff $\langle c_2, \sigma \rangle \Rightarrow^* \gamma$ if γ is terminal, or
- there is an infinite derivation sequence starting in $\langle c_1, \sigma \rangle$ iff there is one starting in $\langle c_2, \sigma \rangle$

Small steps programs equivalence

Exercise 15

Show that

- $c; \mathtt{skip} \approx c$
- $c_1; (c_2; c_3) \approx (c_1; c_2); c_3$

Exercise 16

Introduce small steps semantic rules for the operator

$$\mathtt{repeat}\ c\ \ \mathtt{until}\ \ b$$

and show that

$$\mathtt{repeat}\ c\ \ \mathtt{until}\ \ b \approx c; \mathtt{while}\ (\neg b)\ \mathtt{do}\ c$$

The small-steps semantics functional

Lemma 9

For every $c \in \mathit{Cmd}$ and for any states $\sigma, \sigma' \in \Sigma$, $k \in \mathbb{N}$,

$$\langle c, \sigma_1 \rangle \to \sigma_2 \textit{ implies } \langle c, \sigma_1 \rangle \Rightarrow^* \sigma_2$$

Proof.

Exercise! (Proof by structural induction over the derivation trees.) □

The small-steps semantics functional

Lemma 9

For every $c \in \mathit{Cmd}$ and for any states $\sigma, \sigma' \in \Sigma$, $k \in \mathbb{N}$,

$$\langle c, \sigma_1 \rangle \to \sigma_2 \text{ implies } \langle c, \sigma_1 \rangle \Rightarrow^* \sigma_2$$

Proof.

Exercise! (Proof by structural induction over the derivation trees.) □

Lemma 10

for any $c \in \mathit{Cmd}$, $\sigma, \sigma' \in \Sigma$ and $k \in \mathbb{N}$,

$$\text{if } \langle c, \sigma \rangle \Rightarrow^k \sigma' \text{ then } \langle c, \sigma \rangle \to \sigma'$$

The small-steps semantics functional

Lemma 9

For every $c \in \mathit{Cmd}$ and for any states $\sigma, \sigma' \in \Sigma$, $k \in \mathbb{N}$,

$$\langle c, \sigma_1 \rangle \to \sigma_2 \text{ implies } \langle c, \sigma_1 \rangle \Rightarrow^* \sigma_2$$

Proof.

Exercise! (Proof by structural induction over the derivation trees.) □

Lemma 10

for any $c \in \mathit{Cmd}$, $\sigma, \sigma' \in \Sigma$ and $k \in \mathbb{N}$,

$$\text{if } \langle c, \sigma \rangle \Rightarrow^k \sigma' \text{ then } \langle c, \sigma \rangle \to \sigma'$$

Proof.

Exercise! □

Small Steps Semantics Operational functional

Definition 11

The **functional of the small steps operational semantics**

$$\mathfrak{S}[\![\_]\!] : \mathtt{Cmd} \to (\Sigma \dashrightarrow \Sigma)$$

assigns to every statement $c \in \mathtt{Cmd}$, a partial states function which is defined as follows:

$$\mathfrak{S}[\![c]\!] : \Sigma \dashrightarrow \Sigma$$

$$\mathfrak{S}[\![c]\!](\sigma) = \begin{cases} \sigma' & \langle c, \sigma\rangle \Rightarrow^* \sigma' \\ \mathit{undefined} & \text{otherwise} \end{cases}$$

Equivalence of Big steps and Small steps semantics

Theorem 5

For any $c \in \mathit{Cmd}$,

$$\mathfrak{B}[\![c]\!] = \mathfrak{S}[\![c]\!]$$

i.e. for any $c \in \mathit{Cmd}$, $\sigma \in \Sigma$.

$$\mathfrak{B}[\![c]\!](\sigma) = \mathfrak{S}[\![c]\!](\sigma)$$

Equivalence of Big steps and Small steps semantics

Theorem 5

For any $c \in \mathit{Cmd}$,

$$\mathfrak{B}[\![c]\!] = \mathfrak{S}[\![c]\!]$$

i.e. for any $c \in \mathit{Cmd}$, $\sigma \in \Sigma$.

$$\mathfrak{B}[\![c]\!](\sigma) = \mathfrak{S}[\![c]\!](\sigma)$$

Proof.

Exercise!

The proof of this The proof follows from Lemmas 6,7 and 9. □

Introduction to formal semantics of programming languages

ADA 2024/25

Dep. de Matemática Universidade de Aveiro
Alexandre Madeira
(madeira@ua.pt)

September 29, 2024