Semantics Of Programming Languages

Sergey Mechtaev
s.mechtaev@ucl.ac.uk

UCL Computer Science

The While Language

- Simple language for studying analyses
- A While program is a statement (or a sequence of statements)
- Elementary blocks (assignments, tests and $skip$ statements) are labelled

Syntactic Categories

$a \in AExp$ – arithmetic expressions

$b \in BExp$ – boolean expressions

$S \in$ Stmt – statements

$x, y \in Var$ – variables

$n \in Num$ – numerals

$l \in Lab$ – labels

$op_a \in Op_a$ – arithmetic operators

$op_b \in Op_b$ – boolean operators

$op_r \in Op_r$ – relational operators

Syntax

a = Arithmetic expression

b = Boolean expression

S = Statement (program)

$$a \ ::= x \mid n \mid a_1 \ op_a \ a_2$$

$$b \ ::= true \mid false \mid not \ b \mid b_1 \ op_b \ b_2 \mid a_1 \ op_r \ a_2$$

$$S \ ::= [x \ := a]^l \mid [skip]^l \mid S_1; S_2$$
$$\mid if \ [b]^l \ then \ S_1 \ else \ S_2 \mid while \ [b]^l \ do \ S$$

$[\dots]^l$ → elementary block

^{l} → label allows to identify the primitive constructs of a program

Example Program (Factorial)

$$
\begin{aligned}
&[y := x]^1;\\
&[z \ := 1]^2;\\
&while\ [y > 1]^3\ do\\
&\quad([z \ := z * y]^4;\\
&\quad\ [y \ := y - 1]^5);\\
&[y := 0]^6
\end{aligned}
$$

Formal Semantics

Program state is a mapping from variables to values (numbers):

$$\sigma \in State = Var \rightarrow Z$$

Configuration of the semantics is either a pair statement and state or it is just a state:

$$\langle S, \sigma \rangle \text{ or } \sigma$$

Transitions of the semantics are of the form:

$$\langle S, \sigma \rangle \rightarrow \sigma' \quad and \quad \langle S, \sigma \rangle \rightarrow \langle S', \sigma' \rangle$$

Formal Semantics

Semantics is:

- defined as a *sequence of transitions* between these configurations
- a description of how a piece of program syntax behaves when it is executed
- a description of how the syntax updates the state of the machine

Semantic brackets is semantic function that explain the meaning of the syntax

$$e.g. \qquad [\![six]\!] = 6$$

Semantic Functions

$N: Num \to Z$ Semantic function for Numerals

$A: Aexp \to (State \to Z)$ Semantic function for Arithmetic expressions

$\downarrow$

$State: Var \to Z$

$B: Bexp \to (State \to T)$ Semantic function for Boolean expressions

Semantics of Expressions

$$N: Num \rightarrow Z$$

$$N[\![6]\!] = 6$$

$$A: Aexp \rightarrow (State \rightarrow Z)$$

$$\downarrow$$

$$State: Var \rightarrow Z$$

$$A[\![x]\!]\sigma = \sigma(x)$$

$$A[\![a_1\ opa\ a_2]\!]\sigma = A[\![a_1]\!]\sigma\ opa\ A[\![a_2]\!]\sigma$$

$$B: Bexp \rightarrow (State \rightarrow T)$$

$$B[\![not\ b]\!]\sigma = \neg \mathrm{B}[\![b]\!]\sigma$$

$$B[\![b_1\ opb\ b_2]\!]\sigma = B[\![b_1]\!]\sigma\ opb\ B[\![b_2]\!]\sigma$$

$$B[\![a_1\ opr\ a_2]\!]\sigma = A[\![a_1]\!]\sigma\ opr\ A[\![a_2]\!]\sigma$$

Semantics of Expressions

Semantics of Numeral
= number represented by the Numeral

$$N[\![6]\!] = 6$$

$$A[\![x]\!]\sigma = \sigma(x)$$

$$A[\![a_1 \; opa \; a_2]\!]\sigma = A[\![a_1]\!]\sigma \; opa \; A[\![a_2]\!]\sigma$$

$$B[\![not \; b]\!]\sigma = \neg \mathrm{B}[\![b]\!]\sigma$$

$$B[\![b_1 \; opb \; b_2]\!]\sigma = B[\![b_1]\!]\sigma \; opb \; B[\![b_2]\!]\sigma$$

$$B[\![a_1 \; opr \; a_2]\!]\sigma = A[\![a_1]\!]\sigma \; opr \; A[\![a_2]\!]\sigma$$

Semantics of Expressions

$$N[\![6]\!] = 6$$

Semantics of variable x in state σ = value currently stored in the region of memory named by x (state σ)

$$A[\![x]\!]\sigma = \sigma(x)$$

$$A[\![a_1 \; opa \; a_2]\!]\sigma = A[\![a_1]\!]\sigma \; opa \; A[\![a_2]\!]\sigma$$

$$B[\![not \; b]\!]\sigma = \neg \mathrm{B}[\![b]\!]\sigma$$

$$B[\![b_1 \; opb \; b_2]\!]\sigma = B[\![b_1]\!]\sigma \; opb \; B[\![b_2]\!]\sigma$$

$$B[\![a_1 \; opr \; a_2]\!]\sigma = A[\![a_1]\!]\sigma \; opr \; A[\![a_2]\!]\sigma$$

Semantics of Expressions: *Example*

$$\mathcal{N}[\![6]\!] = 6$$

$$A[\![x]\!]\sigma = \sigma(x)$$

$$A[\![a_1 \; opa \; a_2]\!]\sigma = A[\![a_1]\!]\sigma \; opa \; A[\![a_2]\!]\sigma$$

$$\boxed{\begin{aligned} A[\![x+1]\!]\sigma &= A[\![x]\!]\sigma + A[\![1]\!]\sigma \\ &= \sigma(x) + N[\![1]\!] \\ &= 3+1 \\ &= 4 \end{aligned}}$$

$$B[\![not\ b]\!]\sigma = \neg \mathrm{B}[\![b]\!]\sigma$$

$$B[\![b_1 \; opb \; b_2]\!]\sigma = B[\![b_1]\!]\sigma \; opb \; B[\![b_2]\!]\sigma$$

$$B[\![a_1 \; opr \; a_2]\!]\sigma = A[\![a_1]\!]\sigma \; opr \; A[\![a_2]\!]\sigma$$

Structural Operational Semantics (SOS)

- *Small-step semantics* that specifies the behaviour of a program one step at a time.
- **Intermediate configurations** (if the execution from the statement S in the initial state σ is not complete):

$$\langle S, \sigma \rangle \rightarrow \underline{\langle S', \sigma' \rangle} \rightarrow \sigma''$$

- Describes the small progress from the initial configuration (initial state) to another intermediate configuration or, possibly, to the final state.

$$\langle S, \sigma \rangle \rightarrow \langle S', \sigma' \rangle \quad \text{or} \quad \langle S, \sigma \rangle \rightarrow \sigma'$$

- **Inference rules** justify each step (transition) until reaching a final configuration.

Structural Operational Semantics (SOS)

$[ass]$ $\quad \langle [x \coloneqq a]^l, \sigma \rangle \to \sigma[x \longmapsto \mathcal{A}[\![a]\!]\sigma]$

$[skip]$ $\quad \langle [skip]^l, \sigma \rangle \to \sigma$

$[seq_1]$ $$\frac{\langle S_1, \sigma \rangle \to \langle S'_1, \sigma' \rangle}{\langle S_1; S_2, \sigma \rangle \to \langle S'_1; S_2, \sigma' \rangle}$$

$[seq_2]$ $$\frac{\langle S_1, \sigma \rangle \to \sigma'}{\langle S_1; S_2, \sigma \rangle \to \langle S_2, \sigma' \rangle}$$

$[if_1]$ $\quad \langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma \rangle \to \langle S_1, \sigma \rangle \qquad if\ B[\![b]\!]\ \sigma = true$

$[if_2]$ $\quad \langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma \rangle \to \langle S_2, \sigma \rangle \qquad if\ B[\![b]\!]\ \sigma = false$

$[wh_1]$ $\quad \langle \text{while } [b]^l \text{ do } S, \sigma \rangle \to \langle (S; \text{while } [b]^l \text{ do } S), \sigma \rangle \qquad if\ B[\![b]\!]\ \sigma = true$

$[wh_2]$ $\quad \langle \text{while } [b]^l \text{ do } S, \sigma \rangle \to \sigma \qquad if\ B[\![b]\!]\ \sigma = false$

Structural Operational Semantics (SOS)

$[\boldsymbol{ass}]$ $\langle [x \coloneqq a]^l, \sigma \rangle \to \sigma[x \longmapsto \mathcal{A}[\![a]\!]\sigma]$

A variable identifier x is assigned an arithmetic expression a in the syntax $[x \coloneqq a]^l$ in the current state σ.
It makes a single step (transition) to a final state with the value of x updated to be the semantics (value) of a in the initial state σ.

$[skip]$ $\langle [skip]^l, \sigma \rangle \to \sigma$

$[seq_1]$ $$\frac{\langle S_1, \sigma \rangle \to \langle S'_1, \sigma' \rangle}{\langle S_1; S_2, \sigma \rangle \to \langle S'_1; S_2, \sigma' \rangle}$$

$[seq_2]$ $$\frac{\langle S_1, \sigma \rangle \to \sigma'}{\langle S_1; S_2, \sigma \rangle \to \langle S_2, \sigma' \rangle}$$

$[if_1]$ $\langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma \rangle \to \langle S_1, \sigma \rangle$ $\quad if\ \mathcal{B}[\![b]\!]\ \sigma = true$

$[if_2]$ $\langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma \rangle \to \langle S_2, \sigma \rangle$ $\quad if\ \mathcal{B}[\![b]\!]\ \sigma = false$

$[wh_1]$ $\langle \text{while } [b]^l \text{ do } S, \sigma \rangle \to \langle (S; \text{while } [b]^l \text{ do } S), \sigma \rangle$ $\quad if\ \mathcal{B}[\![b]\!]\ \sigma = true$

$[wh_2]$ $\langle \text{while } [b]^l \text{ do } S, \sigma \rangle \to \sigma$ $\quad if\ \mathcal{B}[\![b]\!]\ \sigma = false$

Structural Operational Semantics (SOS)

The program $skip$ is executed in the current state σ. It makes a single step to a final state σ.

$[ass]$ $\quad \langle [x := a]^l, \sigma \rangle \to \sigma[x \longmapsto \mathcal{A}[\![a]\!]\sigma]$

$[\boldsymbol{skip}]$ $\quad \langle [skip]^l, \sigma \rangle \to \sigma$

$[seq_1]$ $\quad \dfrac{\langle S_1, \sigma \rangle \to \langle S'_1, \sigma' \rangle}{\langle S_1; S_2, \sigma \rangle \to \langle S'_1; S_2, \sigma' \rangle}$

$[seq_2]$ $\quad \dfrac{\langle S_1, \sigma \rangle \to \sigma'}{\langle S_1; S_2, \sigma \rangle \to \langle S_2, \sigma' \rangle}$

$[if_1]$ $\quad \langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma \rangle \to \langle S_1, \sigma \rangle \qquad if\ \mathcal{B}[\![b]\!]\ \sigma = true$

$[if_2]$ $\quad \langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma \rangle \to \langle S_2, \sigma \rangle \qquad if\ \mathcal{B}[\![b]\!]\ \sigma = false$

$[wh_1]$ $\quad \langle \text{while } [b]^l \text{ do } S, \sigma \rangle \to \langle (S; \text{while } [b]^l \text{ do } S), \sigma \rangle \qquad if\ \mathcal{B}[\![b]\!]\ \sigma = true$

$[wh_2]$ $\quad \langle \text{while } [b]^l \text{ do } S, \sigma \rangle \to \sigma \qquad if\ \mathcal{B}[\![b]\!]\ \sigma = false$

Structural Operational Semantics (SOS)

Two rules for **"statements composition"**, one for each possible configuration we could get into (*intermediate configuration* or *final state*) after executing the statement S_1 in the initial state σ.

$[ass]$ $\quad \langle [x := a]^l, \sigma \rangle \to \sigma[x \longmapsto \mathcal{A}[\![a]\!]\sigma]$

$[skip]$ $\quad \langle [skip]^l, \sigma \rangle \to \sigma$

$[\boldsymbol{seq_1}]$ $\quad \dfrac{\langle S_1, \sigma \rangle \to \langle S'_1, \sigma' \rangle}{\langle S_1; S_2, \sigma \rangle \to \langle S'_1; S_2, \sigma' \rangle}$

$[\boldsymbol{seq_2}]$ $\quad \dfrac{\langle S_1, \sigma \rangle \to \sigma'}{\langle S_1; S_2, \sigma \rangle \to \langle S_2, \sigma' \rangle}$

$[if_1]$ $\quad \langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma \rangle \to \langle S_1, \sigma \rangle \qquad if \;\; \mathcal{B}[\![b]\!]\; \sigma = true$

$[if_2]$ $\quad \langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma \rangle \to \langle S_2, \sigma \rangle \qquad if \;\; \mathcal{B}[\![b]\!]\; \sigma = false$

$[wh_1]$ $\quad \langle \text{while } [b]^l \text{ do } S, \sigma \rangle \to \langle (S; \text{while } [b]^l \text{ do } S), \sigma \rangle \qquad if \;\; \mathcal{B}[\![b]\!]\; \sigma = true$

$[wh_2]$ $\quad \langle \text{while } [b]^l \text{ do } S, \sigma \rangle \to \sigma \qquad if \;\; \mathcal{B}[\![b]\!]\; \sigma = false$

Structural Operational Semantics (SOS)

The **transition** $\rightarrow$ is **true if**, when S_1 is executed in the initial state σ, without composing it with S_2, it does not take to a final state, but, in one small step, it is updated to $S'1$ and gets to an intermediate state σ'.

$[ass]$ $\quad \langle [x := a]^l, \sigma\rangle \rightarrow \sigma[x \longmapsto \mathcal{A}[\![a]\!]\sigma]$

$[skip]$ $\quad \langle [skip]^l, \sigma\rangle \rightarrow \sigma$

$[\boldsymbol{seq_1}]$ $\quad \dfrac{\langle S_1, \sigma\rangle \rightarrow \langle {S'}_1, \sigma'\rangle}{\langle S_1; S_2, \sigma\rangle \rightarrow \langle {S'}_1; S_2, \sigma'\rangle}$

$[seq_2]$ $\quad \dfrac{\langle S_1, \sigma\rangle \rightarrow \sigma'}{\langle S_1; S_2, \sigma\rangle \rightarrow \langle S_2, \sigma'\rangle}$

$[if_1]$ $\quad \langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma\rangle \rightarrow \langle S_1, \sigma\rangle \qquad \textit{if } \mathcal{B}[\![b]\!]\,\sigma = \textit{true}$

$[if_2]$ $\quad \langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma\rangle \rightarrow \langle S_2, \sigma\rangle \qquad \textit{if } \mathcal{B}[\![b]\!]\,\sigma = \textit{false}$

$[wh_1]$ $\quad \langle \text{while } [b]^l \text{ do } S, \sigma\rangle \rightarrow \langle (S; \text{while } [b]^l \text{ do } S), \sigma\rangle \qquad \textit{if } \mathcal{B}[\![b]\!]\,\sigma = \textit{true}$

$[wh_2]$ $\quad \langle \text{while } [b]^l \text{ do } S, \sigma\rangle \rightarrow \sigma \qquad \textit{if } \mathcal{B}[\![b]\!]\,\sigma = \textit{false}$

Structural Operational Semantics (SOS)

$[ass]$ $\langle [x := a]^l, \sigma \rangle \to \sigma[x \longmapsto \mathcal{A}[\![a]\!]\sigma]$

$[skip]$ $\langle [skip]^l, \sigma \rangle \to \sigma$

$[seq_1]$ $\dfrac{\langle S_1, \sigma \rangle \to \langle S'_1, \sigma' \rangle}{\langle S_1; S_2, \sigma \rangle \to \langle S'_1; S_2, \sigma' \rangle}$

$[\boldsymbol{seq_2}]$ $\dfrac{\langle S_1, \sigma \rangle \to \sigma'}{\langle S_1; S_2, \sigma \rangle \to \langle S_2, \sigma' \rangle}$

The **transition** $\to$ is **true if**, when S_1 is executed in state σ, it gets to a final state σ' in one small step.
Then when the composition $S_1; S_2$ is executed in σ, there is nothing of S_1 left to execute. Only S_2 needs to be executed in state σ'.

$[if_1]$ $\langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma \rangle \to \langle S_1, \sigma \rangle$ $\quad$ *if* $\mathcal{B}[\![b]\!]\ \sigma = \mathit{true}$

$[if_2]$ $\langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma \rangle \to \langle S_2, \sigma \rangle$ $\quad$ *if* $\mathcal{B}[\![b]\!]\ \sigma = \mathit{false}$

$[wh_1]$ $\langle \text{while } [b]^l \text{ do } S, \sigma \rangle \to \langle (S; \text{while } [b]^l \text{ do } S), \sigma \rangle$ $\quad$ *if* $\mathcal{B}[\![b]\!]\ \sigma = \mathit{true}$

$[wh_2]$ $\langle \text{while } [b]^l \text{ do } S, \sigma \rangle \to \sigma$ $\quad$ *if* $\mathcal{B}[\![b]\!]\ \sigma = \mathit{false}$

Structural Operational Semantics (SOS)

Two rules for **“conditionals”**, one for each possible configuration we could get into depending on which of the two branches is executed.

$[ass]$ $\quad \langle [x := a]^l, \sigma \rangle \to \sigma[x \longmapsto \mathcal{A}[\![a]\!]\sigma]$

$[skip]$ $\quad \langle [skip]^l, \sigma \rangle \to \sigma$

$[seq_1]$ $\quad \dfrac{\langle S_1, \sigma \rangle \to \langle S'_1, \sigma' \rangle}{\langle S_1; S_2, \sigma \rangle \to \langle S'_1; S_2, \sigma' \rangle}$

$[seq_2]$ $\quad \dfrac{\langle S_1, \sigma \rangle \to \sigma'}{\langle S_1; S_2, \sigma \rangle \to \langle S_2, \sigma' \rangle}$

$[\boldsymbol{if_1}]$ $\quad \langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma \rangle \to \langle S_1, \sigma \rangle$ $\quad if\ \mathcal{B}[\![b]\!]\ \sigma = true$

$[\boldsymbol{if_2}]$ $\quad \langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma \rangle \to \langle S_2, \sigma \rangle$ $\quad if\ \mathcal{B}[\![b]\!]\ \sigma = false$

$[wh_1]$ $\quad \langle \text{while } [b]^l \text{ do } S, \sigma \rangle \to \langle (S; \text{while } [b]^l \text{ do } S), \sigma \rangle$ $\quad if\ \mathcal{B}[\![b]\!]\ \sigma = true$

$[wh_2]$ $\quad \langle \text{while } [b]^l \text{ do } S, \sigma \rangle \to \sigma$ $\quad if\ \mathcal{B}[\![b]\!]\ \sigma = false$

Structural Operational Semantics (SOS)

The condition b is evaluated. If true (false), the next configuration equals to execute the true (false) branch, i.e. statement S_1 (S_2) in initial state σ.

$[ass]$ $\quad \langle [x := a]^l, \sigma\rangle \rightarrow \sigma[x \longmapsto \mathcal{A}[\![a]\!]\sigma]$

$[skip]$ $\quad \langle [skip]^l, \sigma\rangle \rightarrow \sigma$

$[seq_1]$ $\quad \dfrac{\langle S_1, \sigma\rangle \rightarrow \langle S'_1, \sigma'\rangle}{\langle S_1; S_2, \sigma\rangle \rightarrow \langle S'_1; S_2, \sigma'\rangle}$

$[seq_2]$ $\quad \dfrac{\langle S_1, \sigma\rangle \rightarrow \sigma'}{\langle S_1; S_2, \sigma\rangle \rightarrow \langle S_2, \sigma'\rangle}$

$[\boldsymbol{if_1}]$ $\quad \langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma\rangle \rightarrow \langle S_1, \sigma\rangle$ $\quad$ $if\ \mathcal{B}[\![b]\!]\ \sigma = true$

$[\boldsymbol{if_2}]$ $\quad \langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma\rangle \rightarrow \langle S_2, \sigma\rangle$ $\quad$ $if\ \mathcal{B}[\![b]\!]\ \sigma = false$

$[wh_1]$ $\quad \langle \text{while } [b]^l \text{ do } S, \sigma\rangle \rightarrow \langle (S; \text{while } [b]^l \text{ do } S), \sigma\rangle$ $\quad$ $if\ \mathcal{B}[\![b]\!]\ \sigma = true$

$[wh_2]$ $\quad \langle \text{while } [b]^l \text{ do } S, \sigma\rangle \rightarrow \sigma$ $\quad$ $if\ \mathcal{B}[\![b]\!]\ \sigma = false$

Structural Operational Semantics (SOS)

The body of the loop b is evaluated and two choices follows: executing the statement S at least once (and then try next iteration in the sequence) or not executing loop body at all and getting to a final state σ.

$[ass]$ $\quad \langle [x := a]^l, \sigma \rangle \to \sigma[x \longmapsto A[\![a]\!]\sigma]$

$[skip]$ $\quad \langle [skip]^l, \sigma \rangle \to \sigma$

$[seq_1]$ $\quad \dfrac{\langle S_1, \sigma \rangle \to \langle S'_1, \sigma' \rangle}{\langle S_1; S_2, \sigma \rangle \to \langle S'_1; S_2, \sigma' \rangle}$

$[seq_2]$ $\quad \dfrac{\langle S_1, \sigma \rangle \to \sigma'}{\langle S_1; S_2, \sigma \rangle \to \langle S_2, \sigma' \rangle}$

$[if_1]$ $\quad \langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma \rangle \to \langle S_1, \sigma \rangle \qquad if\ B[\![b]\!]\ \sigma = true$

$[if_2]$ $\quad \langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma \rangle \to \langle S_2, \sigma \rangle \qquad if\ B[\![b]\!]\ \sigma = false$

$[\boldsymbol{wh_1}]$ $\quad \langle \text{while } [b]^l \text{ do } S, \sigma \rangle \to \langle (S; \text{while } [b]^l \text{ do } S), \sigma \rangle \qquad if\ B[\![b]\!]\ \sigma = true$

$[\boldsymbol{wh_2}]$ $\quad \langle \text{while } [b]^l \text{ do } S, \sigma \rangle \to \sigma \qquad if\ B[\![b]\!]\ \sigma = false$

Derivation

A **derivation sequence** of configurations is:

- either a *finite* sequence of configurations $\langle S_1, \sigma_1\rangle, \ldots, \langle S_n, \sigma_n\rangle$ satisfying $\langle S_i, \sigma_i\rangle \rightarrow \langle S_{i+1}, \sigma_{i+1}\rangle$ for all $i \in [1, n)$, and a $\langle S_n, \sigma_n\rangle \rightarrow \sigma_{i+1}$ corresponding to a terminating computation (Terminating program)

- or an *infinite* sequence of configurations $\langle S_1, \sigma_1\rangle, \ldots, \langle S_n, \sigma_n\rangle, \ldots$ satisfying $\langle S_i, \sigma_i\rangle \rightarrow \langle S_{i+1}, \sigma_{i+1}\rangle$ for all $i > 0$ (Non-Terminating program)

formally defining the semantics (meaning) of a program.

Derivation: *Example*

$S \triangleq (z \coloneqq x; x \coloneqq y); y \coloneqq z$

The program S (sequence of composed statements in state σ) swaps the initial states of variables x and y.

$\sigma(x) = x_0$
$\sigma(y) = y_0$
$\sigma(z) = 0$

Derivation sequence (to demonstrate the correctness of S in a finite no of steps):

$$\langle (\mathrm{z} \coloneqq \mathrm{x}; \mathrm{x} \coloneqq \mathrm{y}); \mathrm{y} \coloneqq \mathrm{z}, \sigma \rangle$$
$$\to \langle x \coloneqq y; y \coloneqq z, \sigma[z \longmapsto x_0] \rangle$$
$$\to \langle y \coloneqq z; \sigma[\mathrm{z} \longmapsto \mathrm{x}_0, \mathrm{x} \longmapsto \mathrm{y}_0] \rangle$$
$$\to \sigma[z \longmapsto x_0, x \longmapsto y_0, y \longmapsto x_0]$$

Transition justified by the following inference rules:

$$\begin{array}{ll}
[ass] & \underline{\langle z \coloneqq x, \sigma \rangle \to \sigma[z \longmapsto x_0]} \\
[seq_1] & \underline{\langle z \coloneqq x, x \coloneqq y, \sigma \rangle \to \langle x \coloneqq y, \sigma[z \longmapsto x_0]} \\
[seq_2] & \langle (z \coloneqq x, x \coloneqq y); y \coloneqq z, \sigma \rangle \\
 & \to \langle x \coloneqq y; y \coloneqq z, \sigma[z \longmapsto x_0] \rangle
\end{array}$$