The SPL Language: Syntax

$x \in \texttt{Var}$ set of integer variables
$v \in \texttt{Z}$ set of integer constants
$\ell \in \texttt{Lab}$ set of labels

$a \in \texttt{AExp}$ set of arithmetic expressions
$b \in \texttt{BExp}$ set of boolean expressions
$s \in \texttt{Stmt}$ set of statements

x, a, b, s are called meta-variables

$a ::= v \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2$

$b ::= \texttt{true} \mid \texttt{false} \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \mid a_1 = a_2 \mid a_1 \leq a_2$

$s ::= x := a^\ell \mid \texttt{skip}^\ell \mid s_1 ; s_2 \mid \texttt{if } b^\ell \texttt{ then } s_1 \texttt{ else } s_2 \mid \texttt{while } b^\ell \texttt{ do } s$

- variables are not declared
- expressions have no side-effects, all side-effects in statements
- only basic statements: no functions, heap, exceptions,…
- semantics usually specified at abstract syntax level

Operational Semantics

- Specifies how expressions and statements should be evaluated

- Evaluation depends on the shape of the expression/statement:
  - `1,2,3`, … do not evaluate any further
  - `x + y` is evaluated further

- Think of it as an interpreter

Operational Semantics

- Evaluation depends on values of variables
  - what does x + y evaluate to ?
  - depends on the values of x and y

- Values of variables at any moment in time are given by a function $\sigma \in$ Store = Var $\longrightarrow$ Z
  - Z is the set of integers
  - to simplify presentation we assume Store denotes total functions
  - if σ is such that `x ↦ 5` and `y ↦ 3`, then `x + y` is `8`

Operational Semantics for SPL

- Configurations: $c \in \Sigma$ where $\Sigma = (\text{Stmt} \times \text{Store}) \cup \text{Store}$
  - $<S, \sigma>$ is a configuration
  - σ is also a configuration: a terminal configuration. All other configurations are non-terminal

- Transitions: $\rightarrow \subseteq \Sigma \times \Sigma$
  - steps between configurations

- Transition system: $(\Sigma, \longrightarrow, I, F)$
  - $I \subseteq \Sigma$: initial configurations
  - $F \subseteq$ Store: final configurations

Operational Semantics for SPL

- We write $c \rightarrow c'$ when $(c, c') \in \rightarrow$
- $\rightarrow^*$ denotes the reflexive transitive closure of the relation $\rightarrow$. We say $c \rightarrow^* c'$ when:
  - $c = c_0$ and $c_n = c'$
  - there is a sequence $c_0 \rightarrow c_1 \rightarrow \dots c_n$ for some $n \geq 0$

Notation: Rules of Inference

These are called evaluation rules

$$\frac{\text{Hypothesis}_1 \;\ldots\; \text{Hypothesis}_n}{\text{Conclusion}}$$

Example:

$$\frac{\text{A is true} \qquad\qquad \text{B is true}}{\text{A} \wedge \text{B is true}}$$

Evaluation rules with no premises are called axioms

$$\frac{}{\text{Conclusion}}$$

Next: operational semantics of SPL

Operational Semantics of SPL

- There are two kinds: big-step and small-step

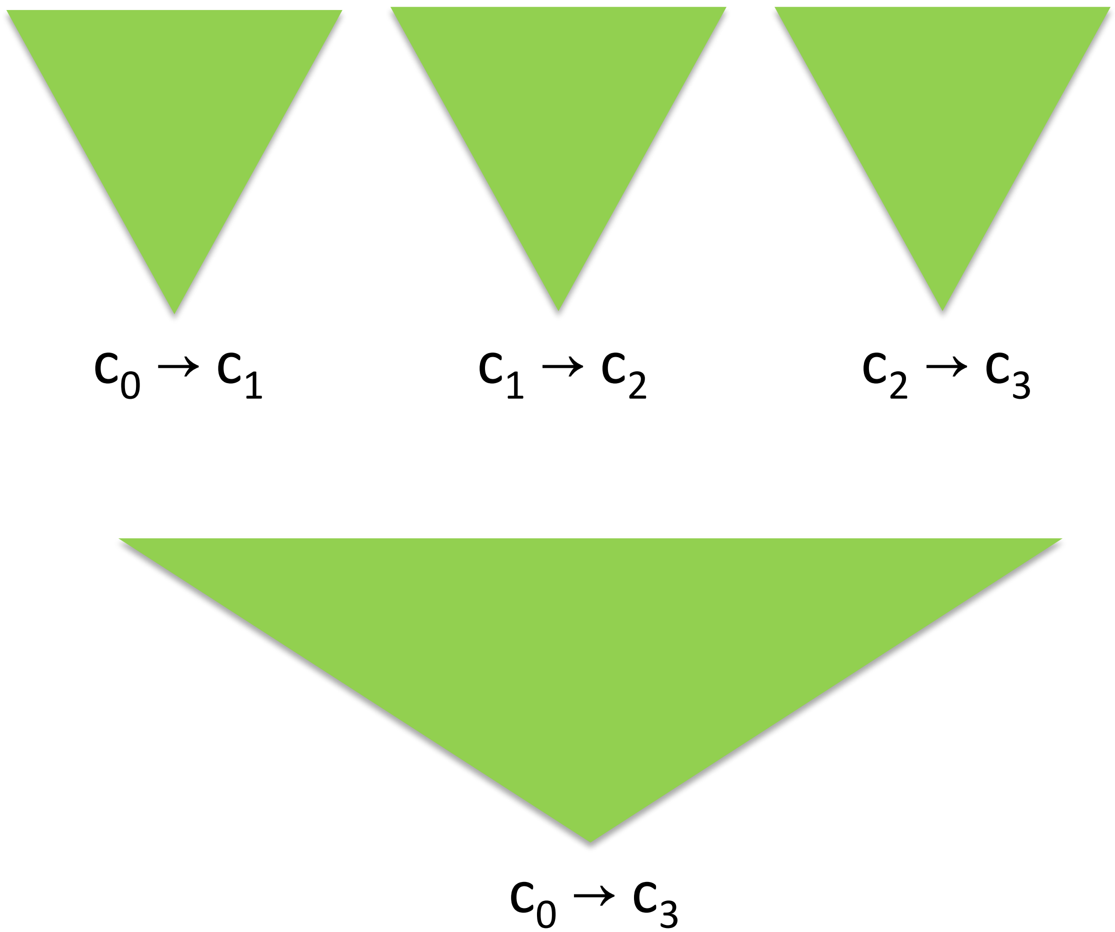
- Big-step
  - $c \longrightarrow c'$ describes the entire computation

- Small-step
  - $c \longrightarrow c'$ describes a single step of a larger computation

Small Step vs. Big Step

small step

big step



Operational Semantics of SPL

Next, we will give semantics of SPL. The statements will be evaluated in a small-step style, while the expressions will be evaluated in big-step style.

Auxiliary Relations

- To describe the semantics of AExp and BExp we use two auxiliary relations

$$\texttt{for AExp: } \Downarrow_{\mathrm{a}} \subseteq (\texttt{AExp} \times \texttt{Store}) \times \texttt{Z}$$
$$\texttt{for BExp: } \Downarrow_{\mathrm{b}} \subseteq (\texttt{BExp} \times \texttt{Store}) \times \{\texttt{true, false}\}$$

- Judgments such as

$$\langle \mathrm{a} , \sigma \rangle \Downarrow_{\mathrm{a}} \mathrm{v}$$

are read as: “expression *a* evaluates to *v* in store σ”
Boolean expressions read similarly

Evaluation rules for AExp

$$\frac{\langle a_1 , \sigma \rangle \Downarrow_a v_1 \qquad \langle a_2 , \sigma \rangle \Downarrow_a v_2}{\langle a_1 + a_2 , \sigma \rangle \Downarrow_a v_1 + v_2}$$

$$\frac{}{\langle x , \sigma \rangle \Downarrow_a \sigma(x)}$$

Evaluation rules for BExp

$$\frac{\langle a_1, \sigma\rangle \Downarrow_a v_1 \qquad \langle a_2, \sigma\rangle \Downarrow_a v_2}{\langle a_1 \leq a_2, \sigma\rangle \Downarrow_b bv} \quad bv \text{ is } v_1 \leq v_2$$

$$\frac{\langle a_1, \sigma\rangle \Downarrow_a v_1 \qquad \langle a_2, \sigma\rangle \Downarrow_a v_2}{\langle a_1 = a_2, \sigma\rangle \Downarrow_b bv} \quad bv \text{ is } v_1 == v_2$$

$$\frac{}{\langle b_1 \wedge b_2, \sigma\rangle \Downarrow_b ???}$$

What about this ?

Evaluation rules for BExp

$$\frac{\langle a_1, \sigma\rangle \Downarrow_a v_1 \qquad \langle a_2, \sigma\rangle \Downarrow_a v_2}{\langle a_1 \leq a_2, \sigma\rangle \Downarrow_b bv} \quad bv \text{ is } v_1 \leq v_2$$

$$\frac{\langle a_1, \sigma\rangle \Downarrow_a v_1 \qquad \langle a_2, \sigma\rangle \Downarrow_a v_2}{\langle a_1 = a_2, \sigma\rangle \Downarrow_b bv} \quad bv \text{ is } v_1 == v_2$$

$$\frac{\langle b_1, \sigma\rangle \Downarrow_b \text{true} \qquad \langle b_2, \sigma\rangle \Downarrow_b \text{true}}{\langle b_1 \land b_2, \sigma\rangle \Downarrow_b \text{true}}$$

short-circuit evaluation

$$\frac{\langle b_1, \sigma\rangle \Downarrow_b \text{false}}{\langle b_1 \land b_2, \sigma\rangle \Downarrow_b \text{false}} \qquad \frac{\langle b_2, \sigma\rangle \Downarrow_b \text{false}}{\langle b_1 \land b_2, \sigma\rangle \Downarrow_b \text{false}}$$

How to read the rules

- Top-down: like inference rules
  - If we know hypothesis holds, conclusion holds
  - If $\langle x , \sigma \rangle \Downarrow_a 5$ and $\langle y , \sigma \rangle \Downarrow_a 6$ then $\langle x + y , \sigma \rangle \Downarrow_a 11$

- Bottom-up: read by inversion
  - Suppose we want to evaluate $\langle x + y , \sigma \rangle \Downarrow_a$
  - Lets look at rules with conclusion that has $\langle x + y , \sigma \rangle$
  - Here: only 1 rule has it as a conclusion (the addition rule)
  - Repeat a recursive tree-walk

Example: Derivation Tree

Evaluate this: $\langle\, (x + 3) * (y + 4)\, , \sigma \rangle$ where σ: $x \mapsto 1$, $y \mapsto 2$

Example: Derivation Tree

Evaluate this: $\langle\, (x + 3) \ast (y + 4)\,,\, \sigma\rangle$ where σ: $x \mapsto 1$, $y \mapsto 2$

$$\frac{}{\langle x\,,\,\sigma\rangle \Downarrow_a 1}$$

$$\frac{}{\langle y\,,\,\sigma\rangle \Downarrow_a 2}$$

$$\frac{\langle x\,,\,\sigma\rangle \Downarrow_a 1 \qquad \langle 3\,,\,\sigma\rangle \Downarrow_a 3}{\langle x + 3\,,\,\sigma\rangle \Downarrow_a 4}$$

$$\frac{\langle y\,,\,\sigma\rangle \Downarrow_a 2 \qquad \langle 4\,,\,\sigma\rangle \Downarrow_a 4}{\langle y + 4\,,\,\sigma\rangle \Downarrow_a 6}$$

$$\frac{\langle x + 3, \sigma\rangle \Downarrow_a 4 \qquad \langle y + 4\,,\,\sigma\rangle \Downarrow_a 6}{\langle\, (x + 3) \ast (y + 4)\,,\,\sigma\rangle \Downarrow_a 24}$$

Evaluation of Statements

- Evaluating a statement produces a new store
  - $\langle \mathrm{s}, \sigma \rangle \rightarrow \langle \mathrm{s}', \sigma' \rangle$

- Evaluation order is important
  - In s_1 ; s_2 s_1 is evaluated before s_2
  - In if true then s_1 else s_2 s_2 is not evaluated

- Some constructs have multiple rules
  - conditionals and while

Evaluation rules for Stmt I

$$\frac{\langle s_1 , \sigma \rangle \to \langle s_2 , \sigma_1 \rangle}{\langle s_1 \, ; \, s_3 , \sigma \rangle \to \langle s_2 \, ; \, s_3 , \sigma_1 \rangle} \qquad \frac{\langle s_1 , \sigma \rangle \to \sigma_1}{\langle s_1 \, ; \, s_2 , \sigma \rangle \to \langle s_2 , \sigma_1 \rangle} \qquad \frac{}{\langle \text{skip}, \sigma \rangle \to \sigma}$$

$$\frac{\langle a , \sigma \rangle \Downarrow_a v}{\langle x := a, \sigma \rangle \to \langle x := v , \sigma \rangle} \qquad \frac{}{\langle x := v, \sigma \rangle \to \sigma[x \mapsto v]}$$

assignment not a single step

Evaluation rules for Stmt II

$$\frac{}{\langle \text{if true then } s_1 \text{ else } s_2 , \sigma \rangle \to \langle s_1 , \sigma \rangle} \qquad \frac{}{\langle \text{if false then } s_1 \text{ else } s_2 , \sigma \rangle \to \langle s_2 , \sigma \rangle}$$

$$\frac{\langle b_1 , \sigma \rangle \Downarrow_b \text{bv}}{\langle \text{if } b_1 \text{ then } s_1 \text{ else } s_2 , \sigma \rangle \to \langle \text{if bv then } s_1 \text{ else } s_2 , \sigma \rangle}$$

$$\frac{}{\langle \text{while b do s}, \sigma \rangle \to ???}$$

Evaluation rules for Stmt II

$$\frac{}{\langle \text{if true then } s_1 \text{ else } s_2, \sigma\rangle \rightarrow \langle s_1, \sigma\rangle} \qquad \frac{}{\langle \text{if false then } s_1 \text{ else } s_2, \sigma\rangle \rightarrow \langle s_2, \sigma\rangle}$$

$$\frac{\langle b_1, \sigma\rangle \Downarrow_b \text{bv}}{\langle \text{if } b_1 \text{ then } s_1 \text{ else } s_2, \sigma\rangle \rightarrow \langle \text{if bv then } s_1 \text{ else } s_2, \sigma\rangle}$$

$$\frac{}{\langle \text{while b do s}, \sigma\rangle \rightarrow \langle \text{if b then (s; while b do s) else skip}, \sigma\rangle}$$

‘while’ expressed in terms of ‘if’

Sequences

Note that for a program S_0 the steps are formed via the relation $\rightarrow$

That is, sequences are $\langle S_0, \sigma_0 \rangle \rightarrow \langle S_1, \sigma_1 \rangle \rightarrow \ldots$

The relations $\Downarrow_a$ or $\Downarrow_b$ are only used to justify the step with $\rightarrow$

In other words, $\Downarrow_a$ or $\Downarrow_b$ are only used to build the relation $\rightarrow$