STRUCTURAL OPERATIONAL SEMANTICS

Grigore Rosu

CS422 – Programming Language Semantics

Conventional Semantic Approaches

A language designer should understand the existing design approaches, techniques and tools, to know what is possible and how, or to come up with better ones. This part of the course will cover the two major PL semantic approaches:

- Big-step structural operational semantics (Big-step SOS)
- Small-step structural operational semantics (Small-step SOS)
- Denotational semantics
- Modular structural operational semantics (Modular SOS)
- Reduction semantics with evaluation contexts
- Abstract Machines
- The chemical abstract machine
- Axiomatic semantics

Many other semantic approaches covered in CS522

IMP

A simple imperative language

IMP – A Simple Imperative Language

We will exemplify the conventional semantic approaches by means of IMP, a very simple non-procedural imperative language, with

- Arithmetic expressions
- Boolean expressions
- Assignment statements
- Conditional statements
- While loop statements
- Blocks

IMP Syntax

$$
\begin{array}{rcl}
Int & ::= & \text{the domain of (unbounded) integer numbers, with usual operations on them} \\
Bool & ::= & \text{the domain of Booleans} \\
Id & ::= & \text{standard identifiers} \\
AExp & ::= & Int \\
 & | & Id \\
 & | & AExp \ \texttt{+} \ AExp \\
 & | & AExp \ \texttt{/} \ AExp \\
BExp & ::= & Bool \\
 & | & AExp \ \texttt{<=} \ AExp \\
 & | & \texttt{!} \ BExp \\
 & | & BExp \ \texttt{\&\&} \ BExp \\
Block & ::= & \{\} \\
 & | & \{ \ Stmt \ \} \\
Stmt & ::= & Block \\
 & | & Id \ \texttt{=} \ AExp \ \texttt{;} \\
 & | & Stmt \ Stmt \\
 & | & \texttt{if} \ \texttt{(} \ BExp \ \texttt{)} \ Block \ \texttt{else} \ Block \\
 & | & \texttt{while} \ \texttt{(} \ BExp \ \texttt{)} \ Block \\
Pgm & ::= & \texttt{int} \ \mathbf{List}\{Id\} \ \texttt{;} \ Stmt
\end{array}
$$

Suppose that, for demonstration purposes, we want “+” and “/” to be non-deterministically strict, “<=“ to be sequentially strict, and “&&” to be short-circuited

Comma-separated list of identifiers

IMP State

- Most semantics need some notion of *state*. A state holds all the semantic ingredients to fully define the meaning of a given program or fragment of program.
- For IMP, a state is a *partial finite-domain function* from identifiers to integers (i.e., a function defined only on a finite subset of identifiers and undefined on the rest), written using a half-arrow:

$$\sigma : Id \rightharpoondown Int$$

- We let *State* denote the set of such functions, and may write it

$$[Id \rightharpoondown Int]^{finite}$$

or

$$\mathbf{Map}\{Id \mapsto Int\}$$

Lookup, Update and Initialization

- We may write states by enumerating each identifier binding. For example, the following state binds x to 8 and y to 0:

$$\sigma = x \mapsto 8, y \mapsto 0$$

- Typical state operations are lookup, update and initialization
- *Lookup*

$$\_(\_) : State \times Id \to Int$$

- *Update*

$$\_[\_/\_] : State \times Int \times Id \to State$$

- *Initialization*

$$\_ \mapsto \_ : \mathbf{List}\{Id\} \times Int \to State$$

BIG-STEP SOS

Big-step structural operational semantics

Big-Step Structural Operational Semantics (Big-Step SOS)

- Gilles Kahn (1987), under the name *natural semantics*. Also known as *relational semantics*, or *evaluation semantics*. We can regard a big-step SOS as a recursive interpreter, telling for a fragment of code and state what it evaluates to.
- *Configuration*: tuple containing code and semantic ingredients
  - E.g., $\langle a_1, \sigma \rangle$ $\langle a_1 + a_2, \sigma \rangle$ $\langle i_1 \rangle$ $\langle i_1 +_{Int} i_2 \rangle$ $\langle \sigma \rangle$
- *Sequent*: Pair of configurations, to be *derived* or *proved*
  - E.g., $\langle a_1, \sigma \rangle \Downarrow \langle i_1 \rangle$ $\langle a_1 + a_2, \sigma \rangle \Downarrow \langle i_1 +_{Int} i_2 \rangle$
- *Rule*: Tells how to derive a sequent from others
  - E.g.,

Read "evaluates to"

Premises

$$\frac{\langle a_1, \sigma \rangle \Downarrow \langle i_1 \rangle \qquad \langle a_2, \sigma \rangle \Downarrow \langle i_2 \rangle}{\langle a_1 + a_2, \sigma \rangle \Downarrow \langle i_1 +_{Int} i_2 \rangle}$$

Conclusion

May omit line when no premises

Big-Step SOS of IMP - Arithmetic

$$\langle i, \sigma \rangle \Downarrow \langle i \rangle \qquad (\textsc{BigStep-Int})$$

State lookup

$$\langle x, \sigma \rangle \Downarrow \langle \sigma(x) \rangle \quad \text{if } \sigma(x) \neq \bot \qquad (\textsc{BigStep-Lookup})$$

Read: "provided that a_1 evaluates to i_1 in σ and a_2 evaluates to i_2 in σ, then $a_1 + a_2$ evaluates to the integer sum of i_1 and i_2 in σ

$$\frac{\langle a_1, \sigma \rangle \Downarrow \langle i_1 \rangle \qquad \langle a_2, \sigma \rangle \Downarrow \langle i_2 \rangle}{\langle a_1 + a_2, \sigma \rangle \Downarrow \langle i_1 +_{Int} i_2 \rangle} \qquad (\textsc{BigStep-Add})$$

$$\frac{\langle a_1, \sigma \rangle \Downarrow \langle i_1 \rangle \qquad \langle a_2, \sigma \rangle \Downarrow \langle i_2 \rangle}{\langle a_1 \,/\, a_2, \sigma \rangle \Downarrow \langle i_1 \,/_{Int}\, i_2 \rangle} \quad \text{if } i_2 \neq 0 \qquad (\textsc{BigStep-Div})$$

Side condition ensures rule will never apply when a_2 evaluates to 0

Big-Step SOS of IMP - Boolean

$$\langle t, \sigma \rangle \Downarrow \langle t \rangle \qquad (\textsc{BigStep-Bool})$$

$$\frac{\langle a_1, \sigma \rangle \Downarrow \langle i_1 \rangle \qquad \langle a_2, \sigma \rangle \Downarrow \langle i_2 \rangle}{\langle a_1 \texttt{ <= } a_2, \sigma \rangle \Downarrow \langle i_1 \leq_{Int} i_2 \rangle} \qquad (\textsc{BigStep-Leq})$$

$$\frac{\langle b, \sigma \rangle \Downarrow \langle \texttt{true} \rangle}{\langle \texttt{!}\, b, \sigma \rangle \Downarrow \langle \texttt{false} \rangle} \qquad (\textsc{BigStep-Not-True})$$

$$\frac{\langle b, \sigma \rangle \Downarrow \langle \texttt{false} \rangle}{\langle \texttt{!}\, b, \sigma \rangle \Downarrow \langle \texttt{true} \rangle} \qquad (\textsc{BigStep-Not-False})$$

$$\frac{\langle b_1, \sigma \rangle \Downarrow \langle \texttt{false} \rangle}{\langle b_1 \texttt{ \&\& } b_2, \sigma \rangle \Downarrow \langle \texttt{false} \rangle} \qquad (\textsc{BigStep-And-False})$$

$$\frac{\langle b_1, \sigma \rangle \Downarrow \langle \texttt{true} \rangle \qquad \langle b_2, \sigma \rangle \Downarrow \langle t \rangle}{\langle b_1 \texttt{ \&\& } b_2, \sigma \rangle \Downarrow \langle t \rangle} \qquad (\textsc{BigStep-And-True})$$

Big-Step SOS of IMP - Statements

$$\langle \{\}, \sigma \rangle \Downarrow \langle \sigma \rangle \qquad \text{(BigStep-Empty-Block)}$$

$$\frac{\langle s, \sigma \rangle \Downarrow \langle \sigma' \rangle}{\langle \{\, s \,\}, \sigma \rangle \Downarrow \langle \sigma' \rangle} \qquad \text{(BigStep-Block)}$$

State update

$$\frac{\langle a, \sigma \rangle \Downarrow \langle i \rangle}{\langle x = a;, \sigma \rangle \Downarrow \langle \sigma[i/x] \rangle} \quad \text{if } \sigma(x) \neq \bot \qquad \text{(BigStep-Asgn)}$$

$$\frac{\langle s_1, \sigma \rangle \Downarrow \langle \sigma_1 \rangle \qquad \langle s_2, \sigma_1 \rangle \Downarrow \langle \sigma_2 \rangle}{\langle s_1\ s_2, \sigma \rangle \Downarrow \langle \sigma_2 \rangle} \qquad \text{(BigStep-Seq)}$$

$$\frac{\langle b, \sigma \rangle \Downarrow \langle \mathtt{true} \rangle \qquad \langle s_1, \sigma \rangle \Downarrow \langle \sigma_1 \rangle}{\langle \mathtt{if}\ (b)\ s_1\ \mathtt{else}\ s_2, \sigma \rangle \Downarrow \langle \sigma_1 \rangle} \qquad \text{(BigStep-If-True)}$$

$$\frac{\langle b, \sigma \rangle \Downarrow \langle \mathtt{false} \rangle \qquad \langle s_2, \sigma \rangle \Downarrow \langle \sigma_2 \rangle}{\langle \mathtt{if}\ (b)\ s_1\ \mathtt{else}\ s_2, \sigma \rangle \Downarrow \langle \sigma_2 \rangle} \qquad \text{(BigStep-If-False)}$$

$$\frac{\langle b, \sigma \rangle \Downarrow \langle \mathtt{false} \rangle}{\langle \mathtt{while}\ (b)\ s, \sigma \rangle \Downarrow \langle \sigma \rangle} \qquad \text{(BigStep-While-False)}$$

$$\frac{\langle b, \sigma \rangle \Downarrow \langle \mathtt{true} \rangle \qquad \langle s\ \mathtt{while}\ (b)\ s, \sigma \rangle \Downarrow \langle \sigma' \rangle}{\langle \mathtt{while}\ (b)\ s, \sigma \rangle \Downarrow \langle \sigma' \rangle} \qquad \text{(BigStep-While-True)}$$

Big-Step SOS of IMP - Programs

State initialization

$$\frac{\langle s, \boxed{xl \mapsto 0} \rangle \Downarrow \langle \sigma \rangle}{\langle \texttt{int}\ xl;\ s \rangle \Downarrow \langle \sigma \rangle} \qquad (\textsc{BigStep-Var})$$

Big-Step Rule Instances

- Rules are schemas, allowing recursively enumerable many instances; side conditions filter out instances
  - E.g., these are correct instances of the rule for division

$$\frac{\langle x, (x \mapsto 8, y \mapsto 0)\rangle \Downarrow \langle 8\rangle \qquad \langle 2, (x \mapsto 8, y \mapsto 0)\rangle \Downarrow \langle 2\rangle}{\langle x/2, (x \mapsto 8, y \mapsto 0)\rangle \Downarrow \langle 4\rangle}$$

$$\frac{\langle x, (x \mapsto 8, y \mapsto 0)\rangle \Downarrow \langle 8\rangle \qquad \langle 2, (x \mapsto 8, y \mapsto 0)\rangle \Downarrow \langle 4\rangle}{\langle x/2, (x \mapsto 8, y \mapsto 0)\rangle \Downarrow \langle 2\rangle}$$

The second may look suspicious, but it is not. Normally, one should never be able to apply it, because one cannot prove its hypotheses

  - However, the following is *not* a correct instance (no matter what ? is):

$$\frac{\langle x, (x \mapsto 8, y \mapsto 0)\rangle \Downarrow \langle 8\rangle \qquad \langle y, (x \mapsto 8, y \mapsto 0)\rangle \Downarrow \langle 0\rangle}{\langle x/y, (x \mapsto 8, y \mapsto 0)\rangle \Downarrow \langle ?\rangle}$$

Big-Step SOS Derivation

The following is a valid proof derivation, or proof tree, using the big-step SOS proof system of IMP above.
Suppose that x and y are identifiers and $\sigma(x)=8$ and $\sigma(y)=0$.

$$\dfrac{\dfrac{\cdot}{\langle \mathrm{x},\sigma\rangle \Downarrow \langle 8\rangle}\,(\textsc{BigStep-Lookup}) \qquad \dfrac{\dfrac{\dfrac{\cdot}{\langle \mathrm{y},\sigma\rangle \Downarrow \langle 0\rangle}\,(\textsc{BigStep-Lookup}) \quad \dfrac{\cdot}{\langle \mathrm{x},\sigma\rangle \Downarrow \langle 8\rangle}\,(\textsc{BigStep-Lookup})}{\langle \mathrm{y} \,/\, \mathrm{x},\sigma\rangle \Downarrow \langle 0\rangle}\,(\textsc{BigStep-Div}) \quad \dfrac{\cdot}{\langle 2,\sigma\rangle \Downarrow \langle 2\rangle}\,(\textsc{BigStep-Int})}{\langle \mathrm{y} \,/\, \mathrm{x} + 2,\sigma\rangle \Downarrow \langle 2\rangle}\,(\textsc{BigStep-Add})}{\langle \mathrm{x} \,/\, (\mathrm{y} \,/\, \mathrm{x} + 2),\sigma\rangle \Downarrow \langle 4\rangle}\,(\textsc{BigStep-Div})$$

Big-Step SOS for Type Systems

- Big-Step SOS is routinely used to define type systems for programming languages
- The idea is that a fragment of code c, in a given *type environment* Γ, can be assigned a certain type τ. We typically write

$$\Gamma \vdash c : \tau$$

instead of

$$\langle c, \Gamma \rangle \Downarrow \langle \tau \rangle$$

- Since all variables in IMP have integer type, Γ can be replaced by a list of untyped variables in our case. In general, however, a type environment Γ contains *typed variables*, that is, pairs "$x : \tau$".

Typing Arithmetic Expressions

$$xl \vdash i : int \qquad (\textsc{BigStepTypeSystem-Int})$$

$$xl \vdash x : int \quad \text{if } x \in xl \qquad (\textsc{BigStepTypeSystem-Lookup})$$

$$\frac{xl \vdash a_1 : int \qquad xl \vdash a_2 : int}{xl \vdash a_1 + a_2 : int} \qquad (\textsc{BigStepTypeSystem-Add})$$

$$\frac{xl \vdash a_1 : int \qquad xl \vdash a_2 : int}{xl \vdash a_1 \;/\; a_2 : int} \qquad (\textsc{BigStepTypeSystem-Div})$$

$$xl \vdash t : bool \quad \text{if } t \in \{\texttt{true}, \texttt{false}\} \qquad (\textsc{BigStepTypeSystem-Bool})$$

Typing Boolean Expressions

$$\frac{xl \vdash a_1 : int \qquad xl \vdash a_2 : int}{xl \vdash a_1 \texttt{<=} a_2 : bool} \quad \text{(BigStepTypeSystem-Leq)}$$

$$\frac{xl \vdash b : bool}{xl \vdash \texttt{!}\, b : bool} \quad \text{(BigStepTypeSystem-Not)}$$

$$\frac{xl \vdash b_1 : bool \qquad xl \vdash b_2 : bool}{xl \vdash b_1 \texttt{\&\&}\, b_2 : bool} \quad \text{(BigStepTypeSystem-And)}$$

$$xl \vdash \{\} : block \quad \text{(BigStepTypeSystem-Empty-Block)}$$

Typing Statements

The type of *s* can be either *block* or *stmt*

$$\frac{xl \vdash s : \boxed{\tau}}{xl \vdash \{\, s \,\} : block} \quad \text{if } \tau \in \{block, stmt\} \qquad (\textsc{BigStepTypeSystem-Block})$$

$$\frac{xl \vdash a : int}{xl \vdash x = a\,; : stmt} \quad \text{if } x \in xl \qquad (\textsc{BigStepTypeSystem-Asgn})$$

$$\frac{xl \vdash s_1 : \boxed{\tau_1} \qquad xl \vdash s_2 : \boxed{\tau_2}}{xl \vdash s_1 \;\, s_2 : stmt} \quad \text{if } \tau_1, \tau_2 \in \{block, stmt\} \qquad (\textsc{BigStepTypeSystem-Seq})$$

$$\frac{xl \vdash b : bool \qquad xl \vdash s_1 : block \qquad xl \vdash s_2 : block}{xl \vdash \texttt{if}\ (b)\ s_1\ \texttt{else}\ s_2 : stmt} \qquad (\textsc{BigStepTypeSystem-If})$$

$$\frac{xl \vdash b : bool \qquad xl \vdash s : block}{xl \vdash \texttt{while}\ (b)\ s : stmt} \qquad (\textsc{BigStepTypeSystem-While})$$

Typing Programs

$$\frac{xl \vdash s : \boxed{\tau}}{\vdash \mathtt{int}\ xl\,;\ s : pgm} \quad \text{if } \tau \in \{block, stmt\} \qquad (\textsc{BigStepTypeSystem-Pgm})$$

Big-Step SOS Type Derivation

Like the big-step rules for the concrete semantics of IMP, the ones for its type system are also rule schemas. We next show a proof derivation for the well-typed-ness of an IMP program that adds all the numbers from 1 to 100:

$$\frac{\dfrac{tree_1 \qquad \dfrac{tree_2 \qquad tree_3}{\texttt{n,s} \vdash (\texttt{while (!(n <= 0)) \{s = s + n; n = n + -1;\}}) : stmt}}{\texttt{n,s} \vdash (\texttt{n = 100; s = 0; while (!(n <= 0)) \{s = s + n; n = n + -1;\}}) : stmt}}{\vdash (\texttt{int n,s; n = 100; s = 0; while (!(n <= 0)) \{s = s + n; n = n + -1;\}}) : pgm}$$

where

Big-Step SOS Type Derivation

$$tree_1 \quad = \quad \left\{ \quad \dfrac{\dfrac{\dfrac{\cdot}{\texttt{n,s} \vdash \texttt{100} : \mathit{int}}}{\texttt{n,s} \vdash (\texttt{n = 100;}) : \mathit{stmt}} \qquad \dfrac{\dfrac{\cdot}{\texttt{n,s} \vdash \texttt{0} : \mathit{int}}}{\texttt{n,s} \vdash (\texttt{s = 0;}) : \mathit{stmt}}}{\texttt{n,s} \vdash (\texttt{n = 100; s = 0;}) : \mathit{stmt}} \right.$$

$$tree_2 \quad = \quad \left\{ \quad \dfrac{\dfrac{\dfrac{\cdot}{\texttt{n,s} \vdash \texttt{n} : \mathit{int}} \qquad \dfrac{\cdot}{\texttt{n,s} \vdash \texttt{0} : \mathit{int}}}{\texttt{n,s} \vdash (\texttt{n <= 0}) : \mathit{bool}}}{\texttt{n,s} \vdash (\texttt{!(n <= 0)}) : \mathit{bool}} \right.$$

Big-Step SOS Type Derivation

$$
tree_3 \quad = \quad \left\{ \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\cdot}{\texttt{n,s} \vdash \texttt{s} : \mathit{int}} \qquad \dfrac{\cdot}{\texttt{n,s} \vdash \texttt{n} : \mathit{int}}}{\texttt{n,s} \vdash (\texttt{s + n}) : \mathit{int}}}{\texttt{n,s} \vdash (\texttt{s = s + n;}) : \mathit{stmt}} \qquad \dfrac{\dfrac{\dfrac{\cdot}{\texttt{n,s} \vdash \texttt{n} : \mathit{int}} \qquad \dfrac{\cdot}{\texttt{n,s} \vdash \texttt{-1} : \mathit{int}}}{\texttt{n,s} \vdash (\texttt{n + -1}) : \mathit{int}}}{\texttt{n,s} \vdash (\texttt{n = n + -1;}) : \mathit{stmt}}}{\texttt{n,s} \vdash (\texttt{s = s + n; n = n + -1;}) : \mathit{stmt}}}{\texttt{n,s} \vdash \{\ \texttt{s = s + n; n = n + -1;}\ \} : \mathit{block}} \right.
$$

SMALL-STEP SOS

Small-step structural operational semantics

Small-Step Structural Operational Semantics (Small-Step SOS)

- Gordon Plotkin (1981)
- Also known as *transitional semantics*, or *reduction semantics*
- One can regard a small-step SOS as a device capable of executing a program step-by-step
- *Configuration*: tuple containing code and semantic ingredients
  - E.g., $\langle a, \sigma \rangle$ $\langle b, \sigma \rangle$ $\langle s, \sigma \rangle$ $\langle p \rangle$
- *Sequent (transition)*: Pair of configurations, to be *derived (proved)*
  - E.g., $\langle a_1, \sigma \rangle \to \langle a_1', \sigma \rangle$ $\langle a_1 + a_2, \sigma \rangle \to \langle a_1' + a_2, \sigma \rangle$
- *Rule*: Tells how to derive a sequent from others
  - E.g.,

$$\frac{\langle a_1, \sigma \rangle \to \langle a_1', \sigma \rangle}{\langle a_1 + a_2, \sigma \rangle \to \langle a_1' + a_2, \sigma \rangle}$$

Small-Step SOS of IMP - Arithmetic

State lookup

$$\langle x, \sigma \rangle \rightarrow \langle \sigma(x), \sigma \rangle \quad \text{if } \sigma(x) \neq \bot \qquad (\textsc{SmallStep-Lookup})$$

$$\frac{\langle a_1, \sigma \rangle \rightarrow \langle a_1', \sigma \rangle}{\langle a_1 + a_2, \sigma \rangle \rightarrow \langle a_1' + a_2, \sigma \rangle} \qquad (\textsc{SmallStep-Add-Arg1})$$

$$\frac{\langle a_2, \sigma \rangle \rightarrow \langle a_2', \sigma \rangle}{\langle a_1 + a_2, \sigma \rangle \rightarrow \langle a_1 + a_2', \sigma \rangle} \qquad (\textsc{SmallStep-Add-Arg2})$$

$$\langle i_1 + i_2, \sigma \rangle \rightarrow \langle i_1 +_{\mathit{Int}} i_2, \sigma \rangle \qquad (\textsc{SmallStep-Add})$$

+ is non-deterministic (its arguments can evaluate in any order, and interleaved

Small-Step SOS of IMP - Arithmetic

$$\frac{\langle a_1, \sigma \rangle \to \langle a_1', \sigma \rangle}{\langle a_1 \;/\; a_2, \sigma \rangle \to \langle a_1' \;/\; a_2, \sigma \rangle} \qquad (\textsc{SmallStep-Div-Arg1})$$

$$\frac{\langle a_2, \sigma \rangle \to \langle a_2', \sigma \rangle}{\langle a_1 \;/\; a_2, \sigma \rangle \to \langle a_1 \;/\; a_2', \sigma \rangle} \qquad (\textsc{SmallStep-Div-Arg2})$$

$$\langle i_1 \;/\; i_2, \sigma \rangle \to \langle i_1 \;/_{\mathit{Int}}\; i_2, \sigma \rangle \quad \text{if } i_2 \neq 0 \qquad (\textsc{SmallStep-Div})$$

Side condition ensures rule will never apply when denominator is 0

/ is also non-deterministic

Small-Step SOS of IMP - Boolean

$$\frac{\langle a_1, \sigma \rangle \to \langle a_1', \sigma \rangle}{\langle a_1 \texttt{<=} a_2, \sigma \rangle \to \langle a_1' \texttt{<=} a_2, \sigma \rangle} \qquad (\textsc{SmallStep-Leq-Arg1})$$

$$\frac{\langle a_2, \sigma \rangle \to \langle a_2', \sigma \rangle}{\langle \boxed{i_1} \texttt{<=} a_2, \sigma \rangle \to \langle i_1 \texttt{<=} a_2', \sigma \rangle} \qquad (\textsc{SmallStep-Leq-Arg2})$$

$$\langle i_1 \texttt{<=} i_2, \sigma \rangle \to \langle i_1 \leq_{Int} i_2, \sigma \rangle \qquad (\textsc{SmallStep-Leq})$$

Ensures sequential strictness

Small-Step SOS of IMP - Boolean

$$\frac{\langle b, \sigma \rangle \to \langle b', \sigma \rangle}{\langle !\, b, \sigma \rangle \to \langle !\, b', \sigma \rangle} \qquad (\textsc{SmallStep-Not-Arg})$$

$$\langle !\, \texttt{true}, \sigma \rangle \to \langle \texttt{false}, \sigma \rangle \qquad (\textsc{SmallStep-Not-True})$$

$$\langle !\, \texttt{false}, \sigma \rangle \to \langle \texttt{true}, \sigma \rangle \qquad (\textsc{SmallStep-Not-False})$$

$$\frac{\langle b_1, \sigma \rangle \to \langle b_1', \sigma \rangle}{\langle b_1 \;\&\&\; b_2, \sigma \rangle \to \langle b_1' \;\&\&\; b_2, \sigma \rangle} \qquad (\textsc{SmallStep-And-Arg1})$$

$$\langle \texttt{false} \;\&\&\; b_2, \sigma \rangle \to \langle \texttt{false}, \sigma \rangle \qquad (\textsc{SmallStep-And-False})$$

$$\langle \texttt{true} \;\&\&\; b_2, \sigma \rangle \to \langle b_2, \sigma \rangle \qquad (\textsc{SmallStep-And-True})$$

Short-circuit semantics

Small-Step SOS Derivation

The following is a valid proof derivation, or proof tree, using the small-step SOS proof system for expressions of IMP above. Suppose that *x* and *y* are identifiers and $\sigma(x)=1$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\cdot}{\langle x, \sigma\rangle \to \langle 1, \sigma\rangle}\;(\textsc{SmallStep-Lookup})}{\langle y \,/\, x, \sigma\rangle \to \langle y \,/\, 1, \sigma\rangle}\;(\textsc{SmallStep-Div-Arg2})}{\langle x + (y \,/\, x), \sigma\rangle \to \langle x + (y \,/\, 1), \sigma\rangle}\;(\textsc{SmallStep-Add-Arg2})}{\langle (x + (y \,/\, x)) \mathrel{<=} x, \sigma\rangle \to \langle (x + (y \,/\, 1)) \mathrel{<=} x, \sigma\rangle}\;(\textsc{SmallStep-Leq-Arg1})$$

Small-Step SOS of IMP - Statements

$$\langle \{\, s \,\}, \sigma \rangle \to \langle s, \sigma \rangle \qquad \text{(SmallStep-Block)}$$

$$\frac{\langle a, \sigma \rangle \to \langle a', \sigma \rangle}{\langle x = a\,;, \sigma \rangle \to \langle x = a'\,;, \sigma \rangle} \qquad \text{(SmallStep-Asgn-Arg2)}$$

State update

$$\langle x = i\,;, \sigma \rangle \to \langle \{\}, \boxed{\sigma[i/x]} \rangle \quad \text{if } \sigma(x) \neq \bot \qquad \text{(SmallStep-Asgn)}$$

$$\frac{\langle s_1, \sigma \rangle \to \langle s_1', \sigma' \rangle}{\langle s_1 \; s_2, \sigma \rangle \to \langle s_1' \; s_2, \sigma' \rangle} \qquad \text{(SmallStep-Seq-Arg1)}$$

$$\langle \{\} \; s_2, \sigma \rangle \to \langle s_2, \sigma \rangle \qquad \text{(SmallStep-Seq-Empty-Block)}$$

Small-Step SOS of IMP - Statements

$$\frac{\langle b, \sigma \rangle \to \langle b', \sigma \rangle}{\langle \texttt{if } (b)\ s_1 \texttt{ else } s_2, \sigma \rangle \to \langle \texttt{if } (b')\ s_1 \texttt{ else } s_2, \sigma \rangle} \quad (\textsc{SmallStep-If-Arg1})$$

$$\langle \texttt{if (true) } s_1 \texttt{ else } s_2, \sigma \rangle \to \langle s_1, \sigma \rangle \quad (\textsc{SmallStep-If-True})$$

$$\langle \texttt{if (false) } s_1 \texttt{ else } s_2, \sigma \rangle \to \langle s_2, \sigma \rangle \quad (\textsc{SmallStep-If-False})$$

$$\langle \texttt{while } (b)\ s, \sigma \rangle \to \langle \texttt{if } (b)\ \{\ s \texttt{ while } (b)\ s\ \} \texttt{ else } \{\}, \sigma \rangle \quad (\textsc{SmallStep-While})$$

$$\langle \texttt{int } xl;\ s \rangle \to \langle s, xl \mapsto 0 \rangle \quad (\textsc{SmallStep-Var})$$

State initialization