Principles of Programming Languages Lectures 6: Big-step Structural Operational Semantics

Andrei Arusoaie[1]

[1]Department of Computer Science

November 2, 2021

Outline

Structural Operational Semantics

Big-step SOS

- Configurations
- Arithmetic expressions
- Boolean expressions
- Statements

Structural Operational Semantics

A language designer should understand the existing design approaches:

- Big-step structural operational semantics
- Small-step structural operational semantics
- Denotational Semantics
- Modular operational semantics
- Reduction semantics with evaluation contexts
- Abstract Machines, the chemical abstract machine
- Axiomatic semantics

Slide credits: prof. Grigore Rosu, UIUC

Structural Operational Semantics

A language designer should understand the existing design approaches:

- Big-step structural operational semantics
- Small-step structural operational semantics
- Denotational Semantics
- Modular operational semantics
- Reduction semantics with evaluation contexts
- Abstract Machines, the chemical abstract machine
- Axiomatic semantics

Structural Operational Semantics (SOS)

- SOS was proposed by Plotkin in 1981
- Simple framework to describe the behaviour of the language constructs as inference rules
- Rules specify transitions between program configurations
- Configurations are tuples of various kinds of data structures (e.g., trees, sets, lists)
    - capture various components of program states (environment, memory, stacks, registers, etc.)

Big-step Structural Operational Semantics

- Probably the most natural way of defining structural operational semantics
- A.k.a. *natural semantics*, *relational semantics*, *evaluation semantics*
- Big-step SOS for a PL: a set of inference rules

IMP

The language that we use to illustrate SOS is IMP. It includes:

- Arithmetic expressions
- Boolean Expressions
- Statements: assignments, (possibly empty) blocks/sequences of statements, decisionals (if-then-else), loops (while)

Outline

Big-step SOS configurations

Configurations are tuples: $\langle \mathit{code}, \mathit{environment}, ... \rangle$.

They hold various information needed to execute a program:

- the code that needs to be executed;
- the current environment;
- the program stack;
- the current program counter;
- ...

Big-step SOS configurations

Configurations are tuples: $\langle \mathit{code}, \mathit{environment}, ... \rangle$.

They hold various information needed to execute a program:

- the code that needs to be executed;
- the current environment;
- the program stack;
- the current program counter;
- ...

IMP configurations

For instance, IMP configurations can store the following information:

- $\langle A, E\rangle$, where A has type `AExp` and E has type `Env`;
- $\langle N\rangle$, where N has type `nat`;
- $\langle B, E\rangle$, where B has type `BExp` and E has type `Env`;
- $\langle B\rangle$, where A has type `bool`;
- $\langle E\rangle$, where E has type `Env`;
- $\langle S, E\rangle$, where S has type `Stmt` and E has type `Env`;
- $\langle S\rangle$, where S has type `Stmt`.

Sequents

Big-step SOS sequents: relations over configurations of the form $C \Downarrow R$, where

- C is a configuration
- R is a *result configuration* that is obtained after the complete evaluation of C.

Result configurations are also called *irreducible configurations*.

Informally, a sequent says that a configuration C *evaluates/executes/transitions* to a configuration R.

Examples: $\langle 1, \sigma \rangle \Downarrow \langle 1 \rangle$, $\langle 1 + 2 \rangle \Downarrow \langle 3 \rangle$, $\langle x, \sigma \rangle \Downarrow \langle \sigma(x) \rangle$.

Sequents

Big-step SOS sequents: relations over configurations of the form $C \Downarrow R$, where

- C is a configuration
- R is a *result configuration* that is obtained after the complete evaluation of C.

Result configurations are also called *irreducible configurations*.

Informally, a sequent says that a configuration C *evaluates/executes/transitions* to a configuration R.

Examples: $\langle 1, \sigma \rangle \Downarrow \langle 1 \rangle$, $\langle 1 + 2 \rangle \Downarrow \langle 3 \rangle$, $\langle x, \sigma \rangle \Downarrow \langle \sigma(x) \rangle$.

Sequents

Big-step SOS sequents: relations over configurations of the form $C \Downarrow R$, where

- C is a configuration
- R is a *result configuration* that is obtained after the complete evaluation of C.

Result configurations are also called *irreducible configurations*.

Informally, a sequent says that a configuration C *evaluates/executes/transitions* to a configuration R.

Examples: $\langle 1, \sigma \rangle \Downarrow \langle 1 \rangle$, $\langle 1 + 2 \rangle \Downarrow \langle 3 \rangle$, $\langle x, \sigma \rangle \Downarrow \langle \sigma(x) \rangle$.

Sequents

Big-step SOS sequents: relations over configurations of the form $C \Downarrow R$, where

- C is a configuration
- R is a *result configuration* that is obtained after the complete evaluation of C.

Result configurations are also called *irreducible configurations*.

Informally, a sequent says that a configuration C *evaluates/executes/transitions* to a configuration R.

Examples: $\langle 1, \sigma\rangle \Downarrow \langle 1\rangle$, $\langle 1+2\rangle \Downarrow \langle 3\rangle$, $\langle x, \sigma\rangle \Downarrow \langle \sigma(x)\rangle$.

Sequents

Big-step SOS sequents: relations over configurations of the form $C \Downarrow R$, where

- C is a configuration
- R is a *result configuration* that is obtained after the complete evaluation of C.

Result configurations are also called *irreducible configurations*.

Informally, a sequent says that a configuration C *evaluates/executes/transitions* to a configuration R.

Examples: $\langle 1, \sigma \rangle \Downarrow \langle 1 \rangle$, $\langle 1 + 2 \rangle \Downarrow \langle 3 \rangle$, $\langle x, \sigma \rangle \Downarrow \langle \sigma(x) \rangle$.

Sequents

Big-step SOS sequents: relations over configurations of the form $C \Downarrow R$, where

- C is a configuration
- R is a *result configuration* that is obtained after the complete evaluation of C.

Result configurations are also called *irreducible configurations*.

Informally, a sequent says that a configuration C *evaluates/executes/transitions* to a configuration R.

Examples: $\langle 1, \sigma \rangle \Downarrow \langle 1 \rangle$, $\langle 1 + 2 \rangle \Downarrow \langle 3 \rangle$, $\langle x, \sigma \rangle \Downarrow \langle \sigma(x) \rangle$.

Rule schemata

The *Big-step SOS rules* have this form:

$$\frac{C_1 \Downarrow R_1 \qquad C_2 \Downarrow R_2 \qquad \cdots \qquad C_n \Downarrow R_n}{C_0 \Downarrow R_0},$$

where $C_0, C_1, \ldots, C_n$ are configurations and $R_0, R_1, \ldots, R_n$ are result configurations.

Here is an example of a big-step rule for IMP:

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

Rule schemata

The *Big-step SOS rules* have this form:

$$\frac{C_1 \Downarrow R_1 \qquad C_2 \Downarrow R_2 \qquad \cdots \qquad C_n \Downarrow R_n}{C_0 \Downarrow R_0},$$

where $C_0, C_1, \ldots, C_n$ are configurations and $R_0, R_1, \ldots, R_n$ are result configurations.

Here is an example of a big-step rule for IMP:

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

Outline

Big-step rules for IMP – 1: arithmetic expressions

BIGSTEP-CONST: $\langle i, \sigma \rangle \Downarrow \langle i \rangle$

BIGSTEP-LOOKUP: $\langle x, \sigma \rangle \Downarrow \langle \sigma(x) \rangle$ if $\sigma(x) \neq \bot$

BIGSTEP-ADD: $$\frac{\langle a_1, \sigma \rangle \Downarrow \langle i_1 \rangle \qquad \langle a_2, \sigma \rangle \Downarrow \langle i_2 \rangle}{\langle a_1 + a_2, \sigma \rangle \Downarrow \langle i_1 +_{nat} i_2 \rangle}$$

BIGSTEP-MUL: $$\frac{\langle a_1, \sigma \rangle \Downarrow \langle i_1 \rangle \qquad \langle a_2, \sigma \rangle \Downarrow \langle i_2 \rangle}{\langle a_1 * a_2, \sigma \rangle \Downarrow \langle i_1 *_{nat} i_2 \rangle}$$

Big-step rules for IMP – 1: arithmetic expressions

BIGSTEP-CONST: $\langle i, \sigma \rangle \Downarrow \langle i \rangle$

BIGSTEP-LOOKUP: $\langle x, \sigma \rangle \Downarrow \langle \sigma(x) \rangle$ if $\sigma(x) \neq \perp$

BIGSTEP-ADD: $\dfrac{\langle a_1, \sigma \rangle \Downarrow \langle i_1 \rangle \qquad \langle a_2, \sigma \rangle \Downarrow \langle i_2 \rangle}{\langle a_1 + a_2, \sigma \rangle \Downarrow \langle i_1 +_{nat} i_2 \rangle}$

BIGSTEP-MUL: $\dfrac{\langle a_1, \sigma \rangle \Downarrow \langle i_1 \rangle \qquad \langle a_2, \sigma \rangle \Downarrow \langle i_2 \rangle}{\langle a_1 * a_2, \sigma \rangle \Downarrow \langle i_1 *_{nat} i_2 \rangle}$

Big-step rules for IMP – 1: arithmetic expressions

BIGSTEP-CONST: $\langle i, \sigma\rangle \Downarrow \langle i\rangle$

BIGSTEP-LOOKUP: $\langle x, \sigma\rangle \Downarrow \langle \sigma(x)\rangle$ if $\sigma(x) \neq \perp$

BIGSTEP-ADD: $$\frac{\langle a_1, \sigma\rangle \Downarrow \langle i_1\rangle \qquad \langle a_2, \sigma\rangle \Downarrow \langle i_2\rangle}{\langle a_1 + a_2, \sigma\rangle \Downarrow \langle i_1 +_{nat} i_2\rangle}$$

BIGSTEP-MUL: $$\frac{\langle a_1, \sigma\rangle \Downarrow \langle i_1\rangle \qquad \langle a_2, \sigma\rangle \Downarrow \langle i_2\rangle}{\langle a_1 * a_2, \sigma\rangle \Downarrow \langle i_1 *_{nat} i_2\rangle}$$

Big-step rules for IMP – 1: arithmetic expressions

BIGSTEP-CONST: $\langle i, \sigma\rangle \Downarrow \langle i\rangle$

BIGSTEP-LOOKUP: $\langle x, \sigma\rangle \Downarrow \langle \sigma(x)\rangle$ if $\sigma(x) \neq \bot$

BIGSTEP-ADD: $$\frac{\langle a_1, \sigma\rangle \Downarrow \langle i_1\rangle \qquad \langle a_2, \sigma\rangle \Downarrow \langle i_2\rangle}{\langle a_1 + a_2, \sigma\rangle \Downarrow \langle i_1 +_{nat} i_2\rangle}$$

BIGSTEP-MUL: $$\frac{\langle a_1, \sigma\rangle \Downarrow \langle i_1\rangle \qquad \langle a_2, \sigma\rangle \Downarrow \langle i_2\rangle}{\langle a_1 * a_2, \sigma\rangle \Downarrow \langle i_1 *_{nat} i_2\rangle}$$

Derivation

- A derivation example where $\sigma(n) = 10$:

$$\dfrac{\dfrac{\cdot}{\langle 2, \sigma \rangle \Downarrow \langle 2 \rangle}\ \textsc{BigStep-Const} \qquad \dfrac{\cdot}{\langle n, \sigma \rangle \Downarrow \langle 10 \rangle}\ \textsc{BigStep-Lookup}}{\langle 2 + n, \sigma \rangle \Downarrow \langle 2 +_{nat} 10 \rangle}\ \textsc{BigStep-Add}$$

rule schema vs. *rule*?

The following *rule*:

$$\frac{\langle 2, \sigma\rangle \Downarrow \langle 2\rangle \quad \langle n, \sigma\rangle \Downarrow \langle 10\rangle}{\langle 2 \ +' \ n, \sigma\rangle \Downarrow \langle 2 +_{nat} 10\rangle}$$

is a an instance of the *rule schema*:

$$\frac{\langle a_1, \sigma\rangle \Downarrow \langle i_1\rangle \quad \langle a_2, \sigma\rangle \Downarrow \langle i_2\rangle}{\langle a_1 \ +' \ a_2, \sigma\rangle \Downarrow \langle i_1 +_{nat} i_2\rangle} \text{ BIGSTEP-ADD.}$$

On the other hand, this is not an instance of BIGSTEP-ADD:

$$\frac{\langle 2, \sigma\rangle \Downarrow \langle 2\rangle \quad \langle n, \sigma\rangle \Downarrow \langle 10\rangle}{\langle 2 \ +' \ n, \sigma\rangle \Downarrow \langle 2 +_{nat} 10, \sigma\rangle} \ ?$$

because $\langle 2 \ +' \ n, \sigma\rangle \Downarrow \langle 2 +_{nat} 10, \sigma\rangle$ is not an instance of the conclusion of BIGSTEP-ADD.

rule schema vs. *rule*?

The following *rule*:

$$\frac{\langle 2, \sigma\rangle \Downarrow \langle 2\rangle \quad \langle n, \sigma\rangle \Downarrow \langle 10\rangle}{\langle 2 \ \ +' \ \ n, \sigma\rangle \Downarrow \langle 2 +_{nat} 10\rangle}$$

is a an instance of the *rule schema*:

$$\frac{\langle a_1\,,\ \sigma\rangle \Downarrow \langle i_1\rangle \qquad \langle a_2,\ \sigma\rangle \Downarrow \langle i_2\rangle}{\langle a_1 \ \ +' \ \ a_2,\ \sigma\rangle \Downarrow \langle i_1 +_{nat} i_2\rangle}\ \textsc{BigStep-Add}.$$

On the other hand, this is not an instance of BIGSTEP-ADD:

$$\frac{\langle 2, \sigma\rangle \Downarrow \langle 2\rangle \quad \langle n, \sigma\rangle \Downarrow \langle 10\rangle}{\langle 2 \ \ +' \ \ n, \sigma\rangle \Downarrow \langle 2 +_{nat} 10, \sigma\rangle}\ ?$$

because $\langle 2 \ \ +' \ \ n, \sigma\rangle \Downarrow \langle 2 +_{nat} 10, \sigma\rangle$ is not an instance of the conclusion of BIGSTEP-ADD.

Outline

Big-step rules for IMP – 2: boolean expressions

BIGSTEP-TRUE: $\langle \texttt{btrue}, \sigma \rangle \Downarrow \langle \mathit{true} \rangle$

BIGSTEP-FALSE: $\langle \texttt{bfalse}, \sigma \rangle \Downarrow \langle \mathit{false} \rangle$

BIGSTEP-NOTTRUE: $$\frac{\langle b,\ \sigma \rangle \Downarrow \langle \mathit{true} \rangle}{\langle\, !\, b,\ \sigma \rangle \Downarrow \langle \mathit{false} \rangle}$$

BIGSTEP-NOTFALSE: $$\frac{\langle b,\ \sigma \rangle \Downarrow \langle \mathit{false} \rangle}{\langle\, !\, b,\ \sigma \rangle \Downarrow \langle \mathit{true} \rangle}$$

Big-step rules for IMP – 2: boolean expressions

BIGSTEP-ANDTRUE:
$$\frac{\langle b_1, \sigma\rangle \Downarrow \langle \mathit{true}\rangle \qquad \langle b_2, \sigma\rangle \Downarrow \langle b\rangle}{\langle b_1 \ \texttt{and} \ b_2, \sigma\rangle \Downarrow \langle b\rangle}$$

BIGSTEP-ANDFALSE:
$$\frac{\langle b_1, \sigma\rangle \Downarrow \langle \mathit{false}\rangle}{\langle b_1 \ \texttt{and} \ b_2, \sigma\rangle \Downarrow \langle \mathit{false}\rangle}$$

BIGSTEP-LT:
$$\frac{\langle a_1, \sigma\rangle \Downarrow \langle i_1\rangle \qquad \langle a_2, \sigma\rangle \Downarrow \langle i_2\rangle}{\langle a_1 \ < \ a_2, \sigma\rangle \Downarrow \langle i_1 <_{\mathit{nat}} i_2\rangle}$$

BIGSTEP-GT:
$$\frac{\langle a_1, \sigma\rangle \Downarrow \langle i_1\rangle \qquad \langle a_2, \sigma\rangle \Downarrow \langle i_2\rangle}{\langle a_1 \ > \ a_2, \sigma\rangle \Downarrow \langle i_1 >_{\mathit{nat}} i_2\rangle}$$

Derivation example

A derivation tree for $\langle 2 \ + \ n \ < \ 10, \sigma\rangle \Downarrow \langle (2 +_{nat} 10) <_{nat} 10\rangle$:

$$\dfrac{\dfrac{\dfrac{.}{\langle 2, \sigma\rangle \Downarrow \langle 2\rangle}\ \textsc{Const} \qquad \dfrac{.}{\langle n, \sigma\rangle \Downarrow \langle 10\rangle}\ \textsc{Lookup}}{\langle 2 \ + \ n, \sigma\rangle \Downarrow \langle 2 +_{nat} 10\rangle}\ \textsc{Add} \qquad \dfrac{.}{\langle 10, \sigma\rangle \Downarrow \langle 10\rangle}\ \textsc{Const}}{\langle 2 \ + \ n \ < \ 10, \sigma_1\rangle \Downarrow \langle (2 +_{nat} 10) <_{nat} 10\rangle}\ \textsc{Lt}$$

Outline

Structural Operational Semantics

Big-step SOS

- Configurations
- Arithmetic expressions
- Boolean expressions
- Statements

Big-step rules for IMP – 3: statements

BIGSTEP-ASSIGN:

$$\frac{\langle a, \sigma\rangle \Downarrow \langle i\rangle}{\langle x \;\texttt{::=}\; a, \sigma\rangle \Downarrow \langle \sigma[i/x]\rangle} \quad \textit{if } \sigma(x) \neq \bot$$

BIGSTEP-SEQ:

$$\frac{\langle s_1, \sigma\rangle \Downarrow \langle \sigma_1\rangle \qquad \langle s_2, \sigma_1\rangle \Downarrow \langle \sigma_2\rangle}{\langle s_1 \;\texttt{;;}\; s_2, \sigma\rangle \Downarrow \langle \sigma_2\rangle}$$

BIGSTEP-SKIP:

$$\langle \texttt{skip}, \sigma\rangle \Downarrow \langle \sigma\rangle$$

BIGSTEP-WHILEFALSE:

$$\frac{\langle b, \sigma\rangle \Downarrow \langle \mathit{false}\rangle}{\langle \texttt{while}\; b\; s, \sigma\rangle \Downarrow \langle \sigma\rangle}$$

BIGSTEP-WHILETRUE:

$$\frac{\langle b, \sigma\rangle \Downarrow \langle \mathit{true}\rangle \qquad \langle s \;\texttt{;;}\; \texttt{while}\; b\; s, \sigma\rangle \Downarrow \langle \sigma'\rangle}{\langle \texttt{while}\; b\; s, \sigma\rangle \Downarrow \langle \sigma'\rangle}$$

Exercise

Write a derivation for the sequent:

$$\langle i\texttt{::=}0\texttt{;;}\,\texttt{while}\ \ i{<}1\ (i\texttt{::=}i{+}1), \sigma\rangle \Downarrow \langle(\sigma[0/i])[1/i]\rangle.$$

The derivation is shown in the next slides

Derivation

$$\dfrac{\dfrac{\cdot}{\langle i\texttt{::=}0, \sigma\rangle \Downarrow \langle \sigma[0/i]\rangle}\ \textsc{Assign} \qquad A}{\langle i\texttt{::=}0\texttt{;;}\texttt{while }\ i{<}1\ (i\texttt{::=}i{+}1), \sigma\rangle \Downarrow \langle (\sigma[0/i])[1/i]\rangle}\ \textsc{Seq}$$

Derivation for A

$$
\dfrac{\dfrac{\dfrac{\cdot}{\langle i, \sigma[0/i]\rangle \Downarrow \langle 0\rangle}\ \textsc{Lookup} \qquad \dfrac{\cdot}{\langle 1, \sigma[0/i]\rangle \Downarrow \langle 1\rangle}\ \textsc{Const}}{\langle i{<}1, \sigma[0/i]\rangle \Downarrow \langle \mathit{true}\rangle}\ \textsc{Lt} \qquad B}{\langle \texttt{while}\ \ i{<}1\ (i ::= i{+}1), \sigma[0/i]\rangle \Downarrow \langle (\sigma[0/i])[1/i]\rangle}\ \textsc{WhileTrue}
$$

Derivation for B

$$
\cfrac{
  \cfrac{
    \cfrac{\cfrac{\cdot}{\langle i, \sigma[0/i]\rangle \Downarrow \langle 0\rangle}\ \textsc{Lookup} \qquad \cfrac{\cdot}{\langle 1, \sigma[0/i]\rangle \Downarrow \langle 1\rangle}\ \textsc{Const}}{\langle i{+}1\rangle \Downarrow \langle 1\rangle}\ \textsc{Add}
  }{\langle i ::= i{+}1\rangle \Downarrow \langle(\sigma[0/i])[1/i]\rangle}\ \textsc{Assign}
  \qquad C
}{\langle i ::= i{+}1\texttt{;;while}\ \ i{<}2\ (i ::= i{+}1), \sigma[0/i]\rangle \Downarrow \langle(\sigma[0/i])[1/i]\rangle}\ \textsc{Seq}
$$

Derivation for C

$$
\dfrac{\dfrac{\dfrac{\cdot}{\langle i, (\sigma[0/i])[1/i]\rangle \Downarrow \langle 1\rangle}\ \textsc{Lookup} \qquad \dfrac{\cdot}{\langle 1, (\sigma[0/i])[1/i]\rangle \Downarrow \langle 1\rangle}\ \textsc{Const}}{\langle i <' 1, (\sigma[0/i])[1/i]\rangle \Downarrow \langle \mathit{false}\rangle}\ \textsc{Lt}}{\langle \texttt{while}\ \ i{<}1\ (i::=i+1), (\sigma[0/i])[1/i]\rangle \Downarrow \langle(\sigma[0/i])[1/i]\rangle}\ \textsc{WhileFalse}
$$

Complete derivation

It is a real challenge to read the assembled derivation:

$$\dfrac{\dfrac{.}{\langle i\texttt{::=}0,\ \sigma\rangle \Downarrow \langle \sigma[0/i]\rangle} \qquad \dfrac{\dfrac{\dfrac{.}{\langle i,\ \sigma[0/i]\rangle \Downarrow \langle 0\rangle} \quad \dfrac{.}{\langle 1,\ \sigma[0/i]\rangle \Downarrow \langle 1\rangle}}{\langle i\texttt{<'}1,\ \sigma[0/i]\rangle \Downarrow \langle \mathit{true}\rangle} \qquad \dfrac{\dfrac{\dfrac{\dfrac{.}{\langle i,\ \sigma[0/i]\rangle \Downarrow \langle 0\rangle} \quad \dfrac{.}{\langle 1,\ \sigma[0/i]\rangle \Downarrow \langle 1\rangle}}{\langle i\texttt{+'}1\rangle \Downarrow \langle 1\rangle}}{\langle i\texttt{::=}i\texttt{+'}1\rangle \Downarrow \langle (\sigma[0/i])[1/i]\rangle} \quad \dfrac{\dfrac{\dfrac{.}{\langle i,\ (\sigma[0/i])[1/i]\rangle \Downarrow \langle 1\rangle} \quad \dfrac{.}{\langle 1,\ (\sigma[0/i])[1/i]\rangle \Downarrow \langle 1\rangle}}{\langle i\texttt{<'}1,\ (\sigma[0/i])[1/i]\rangle \Downarrow \langle \mathit{false}\rangle}}{\langle \mathsf{while}\ i\texttt{<'}1\ (i\texttt{::=}i\texttt{+'}1),\ (\sigma[0/i])[1/i]\rangle \Downarrow \langle (\sigma[0/i])[1/i]\rangle}}{\langle i\texttt{::=}i\texttt{+'}1\texttt{;;}\mathsf{while}\ i\texttt{<'}2\ (i\texttt{::=}i\texttt{+'}1),\ \sigma[0/i]\rangle \Downarrow \langle (\sigma[0/i])[1/i]\rangle}}{\langle \mathsf{while}\ i\texttt{<'}1\ (i\texttt{::=}i\texttt{+'}1),\ \sigma[0/i]\rangle \Downarrow \langle (\sigma[0/i])[1/i]\rangle}}{\langle i\texttt{::=}0\texttt{;;}\mathsf{while}\ i\texttt{<'}1\ (i\texttt{::=}i\texttt{+'}1),\ \sigma\rangle \Downarrow \langle (\sigma[0/i])[1/i]\rangle}$$

Bibliography

- Chapter Simple Imperative Programs, Section: Evaluation as relation in Software Foundations - Volume 1, Benjamin C. Pierce, Arthur Azevedo de Amorim, Chris Casinghino, Marco Gaboardi, Michael Greenberg, Cătălin Hriţcu, Vilhelm Sjöberg, Andrew Tolmach, Brent Yorgey `https://softwarefoundations.cis.upenn.edu/lf-current/Imp.html`