Principles of Programming Languages Lecture 7: Small-step Structural Operational Semantics

Andrei Arusoaie[1]

[1]Department of Computer Science

November 9, 2021

Outline

Structural Operational Semantics

A language designer should understand the existing design approaches:

- Big-step structural operational semantics (discussed last week)
- 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.)

Small-step Structural Operational Semantics

- A variant of SOS that captures the notion of one computational step
- A.k.a. *reduction semantics*, *transition semantics*, *one-step operational semantics*
- Small-step SOS for a PL: a set of small-step inference rules

Small-step Structural Operational Semantics

- A variant of SOS that captures the notion of one computational step
- A.k.a. *reduction semantics*, *transition semantics*, *one-step operational semantics*
- Small-step SOS for a PL: a set of small-step inference rules
- The transitions between small-step *configurations* is denoted by a simple arrow "$\rightarrow$': it captures only *one* computation step at a time, while the big-step transition denoted by $\Downarrow$ captures *all* computations in a single transition

Why do we need Small-step SOS?

- Big-step SOS issues:
  - non-deterministic behaviour

```
int main() {
        int x = 1;
        return (x = 1) + (x = 2);
}
```

Why do we need Small-step SOS?

- Big-step SOS issues:
  - non-deterministic behaviour

```
int main() {
        int x = 1;
        return (x = 1) + (x = 2);
}
```

  - cannot observe intermediate states (e.g., interleaving in concurrent languages)

Why do we need Small-step SOS?

- Big-step SOS issues:
  - non-deterministic behaviour

```
int main() {
        int x = 1;
        return (x = 1) + (x = 2);
}
```

  - cannot observe intermediate states (e.g., interleaving in concurrent languages)
  - cannot naturally define the evaluation of expressions like `2 + nil` in languages where nats and lists can be mixed

Small-step SOS

Difference w.r.t. Big-step SOS:

Small-step SOS

Difference w.r.t. Big-step SOS:

- A small-step SOS transition encodes **only one** step

Small-step SOS

Difference w.r.t. Big-step SOS:

- A small-step SOS transition encodes **only one** step
- A big-step SOS transition can be simulated using multiple small-steps (transitive closure)

Small-step SOS

Difference w.r.t. Big-step SOS:

- A small-step SOS transition encodes **only one** step
- A big-step SOS transition can be simulated using multiple small-steps (transitive closure)
- Since big-step defines all computation steps in one transition it cannot capture non-determinism by default

Small-step SOS

Difference w.r.t. Big-step SOS:

- A small-step SOS transition encodes **only one** step
- A big-step SOS transition can be simulated using multiple small-steps (transitive closure)
- Since big-step defines all computation steps in one transition it cannot capture non-determinism by default
- Direct control over what to execute and when

Small-step SOS

Difference w.r.t. Big-step SOS:

- A small-step SOS transition encodes **only one** step
- A big-step SOS transition can be simulated using multiple small-steps (transitive closure)
- Since big-step defines all computation steps in one transition it cannot capture non-determinism by default
- Direct control over what to execute and when
- Therefore, small-step SOS is finer-grain than big-step SOS

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

Configurations

The small-step SOS configurations for IMP are a subset of the big-step SOS configurations:

- $\langle A, E\rangle$, where A is of type `AExp` and E is of type `Env`;
- $\langle B, E\rangle$, where B is of type `BExp` and E is of type `Env`;
- $\langle S, E\rangle$, where S is of type `Stmt` and E is of type `Env`.

Sequents and rules

- The small-step SOS sequents are binary relations over configurations: $C \to C'$
- $C \to C'$: the configuration C' is obtained from C after one step of computation

Sequents and rules

- The small-step SOS sequents are binary relations over configurations: $C \to C'$
- $C \to C'$: the configuration C' is obtained from C after one step of computation
- Small-step SOS rules have the following general form:

$$\frac{C_1 \to C_1' \quad C_2 \to C_2' \quad \cdots \quad C_n \to C_n'}{C_0 \to C_0'} \ \textit{cond.}$$

Outline

Small-step SOS

There are some noticeable differences w.r.t. big-step SOS:

- The lookup rule:
  Small-step: $\langle x, \sigma\rangle \Rightarrow \langle \sigma(x), \sigma\rangle$ vs. Big-step: $\langle x, \sigma\rangle \Downarrow \langle \sigma(x)\rangle$;
- There is no rule for evaluating constants;
- More rules for evaluating addition/multiplication of arithmetic expressions!

Small-step SOS

There are some noticeable differences w.r.t. big-step SOS:

- The lookup rule:
  Small-step: $\langle x, \sigma \rangle \Rightarrow \langle \sigma(x), \sigma \rangle$ vs. Big-step: $\langle x, \sigma \rangle \Downarrow \langle \sigma(x) \rangle$;
- There is no rule for evaluating constants;
- More rules for evaluating addition/multiplication of arithmetic expressions!
  - The reason is that both operands of an addition/multiplication can be evaluated using the one step relation $\langle a, \sigma \rangle \to \langle a', \sigma \rangle$.

Small-step SOS

There are some noticeable differences w.r.t. big-step SOS:

- The lookup rule:
  Small-step: $\langle x, \sigma \rangle \Rightarrow \langle \sigma(x), \sigma \rangle$ vs. Big-step: $\langle x, \sigma \rangle \Downarrow \langle \sigma(x) \rangle$;
- There is no rule for evaluating constants;
- More rules for evaluating addition/multiplication of arithmetic expressions!
  - The reason is that both operands of an addition/multiplication can be evaluated using the one step relation $\langle a, \sigma \rangle \rightarrow \langle a', \sigma \rangle$.
  - Therefore, one can evaluate either the first argument or the second argument $\rightarrow$ two rules - one for each operand

Small-step SOS

There are some noticeable differences w.r.t. big-step SOS:

- The lookup rule:
  Small-step: $\langle x, \sigma\rangle \Rightarrow \langle \sigma(x), \sigma\rangle$ vs. Big-step: $\langle x, \sigma\rangle \Downarrow \langle \sigma(x)\rangle$;
- There is no rule for evaluating constants;
- More rules for evaluating addition/multiplication of arithmetic expressions!
  - The reason is that both operands of an addition/multiplication can be evaluated using the one step relation $\langle a, \sigma\rangle \to \langle a', \sigma\rangle$.
  - Therefore, one can evaluate either the first argument or the second argument $\to$ two rules - one for each operand
  - Moreover, when both operands are completely evaluated we use an additional rule to obtain the result.

Small-step rules for IMP – 1: arithmetic expressions

Lookup and Addition:

SMALLSTEP-LOOKUP: $\langle x, \sigma \rangle \rightarrow \langle \sigma(x), \sigma \rangle$ if $\sigma(x) \neq \perp$

SMALLSTEP-ADD1: $$\frac{\langle a_1, \sigma \rangle \rightarrow \langle a_1', \sigma \rangle}{\langle a_1 \;+'\; a_2, \sigma \rangle \rightarrow \langle a_1' \;+'\; a_2, \sigma \rangle}$$

SMALLSTEP-ADD2: $$\frac{\langle a_2, \sigma \rangle \rightarrow \langle a_2', \sigma \rangle}{\langle a_1 \;+'\; a_2, \sigma \rangle \rightarrow \langle a_1 \;+'\; a_2', \sigma \rangle}$$

SMALLSTEP-ADD: $\langle i_1 \;+'\; i_2, \sigma \rangle \rightarrow \langle i_1 +_{nat} i_2, \sigma \rangle$

Note: + is non-deterministic!

Small-step rules for IMP – 1: arithmetic expressions

Multiplication:

$$\text{SMALLSTEP-MUL1:}\quad \frac{\langle a_1,\ \sigma\rangle \rightarrow \langle a_1',\sigma\rangle}{\langle a_1\ \ \star'\ \ a_2,\ \sigma\rangle \rightarrow \langle a_1'\ \ \star'\ \ a_2,\sigma\rangle}$$

$$\text{SMALLSTEP-MUL2:}\quad \frac{\langle a_2,\ \sigma\rangle \rightarrow \langle a_2',\sigma\rangle}{\langle a_1\ \ \star'\ \ a_2,\ \sigma\rangle \rightarrow \langle a_1\ \ \star'\ \ a_2',\sigma\rangle}$$

$$\text{SMALLSTEP-MUL:}\quad \langle i_1\ \ \star'\ \ i_2,\sigma\rangle \rightarrow \langle i_1 *_{nat} i_2,\sigma\rangle$$

Simple derivation

- Let us consider $\sigma(x) = 10$. Here is a derivation for sequent $\langle 2 + x, \sigma \rangle \to \langle 2 + 10, \sigma \rangle$:

$$\dfrac{\dfrac{.}{\langle x, \sigma \rangle \to \langle 10, \sigma \rangle}\ \textsc{SmallStep-Lookup}}{\langle 2 + x, \sigma \rangle \to \langle 2 + 10, \sigma \rangle}\ \textsc{SmallStep-Add2}$$

- What about $\langle 2 + x, \sigma \rangle \to \langle 12, \sigma \rangle$?

Sequences

- Reflexive and transitive closure $\rightarrow^*$:

$$\frac{\cdot}{a \rightarrow^* a}\ \textsc{Refl} \qquad \frac{a_1 \rightarrow a_2 \qquad a_2 \rightarrow^* a_3}{a_1 \rightarrow^* a_3}\ \textsc{Tran}$$

Derivation example

- We can now prove $\langle 2 + x, \sigma \rangle \rightarrow^* \langle 12, \sigma \rangle$:

$$
\dfrac{
  \dfrac{
    \dfrac{\cdot}{\langle n, \sigma \rangle \rightarrow \langle 10, \sigma \rangle}\ \textsc{Lookup}
  }{
    \langle 2 \ +' \ n, \sigma \rangle \rightarrow \langle 2 \ +' \ 10, \sigma \rangle
  }\ \textsc{Add2}
  \quad
  \dfrac{
    \dfrac{\cdot}{\langle 2 \ +' \ 10, \sigma \rangle \rightarrow \langle 12, \sigma \rangle}\ \textsc{Add}
    \quad
    \dfrac{\cdot}{\langle 12, \sigma \rangle \rightarrow^* \langle 12, \sigma \rangle}\ \textsc{Refl}
  }{
    \langle 2 \ +' \ 10, \sigma \rangle \rightarrow^* \langle 12, \sigma \rangle
  }\ \textsc{Tran}
}{
  \langle 2 \ +' \ n, \sigma \rangle \rightarrow^* \langle 12, \sigma \rangle
}\ \textsc{Tran}
$$

Outline

Small-step rules for IMP – 2: boolean expressions

The small-step SOS rules for boolean expressions are different from the big-step SOS rules:

- no rules for evaluating the boolean constants
- 3 rules for negation: one rule to evaluate of the negated boolean, two rules to evaluate the base cases;
- 3 rules for conjunction:
  - one which evaluates the first argument
  - one rule which evaluates the conjunction when the first argument is false (shortcircuited conjunction)
  - one rule which evaluate the conjunction when the first argument is true
- comparisons: each argument can be evaluated by a separate rule and there is one rule which compares two *concrete values*.

Small-step rules for IMP – 2: boolean expressions

Negations and conjunction:

SMALLSTEP-NOT: $$\frac{\langle b,\ \sigma\rangle \to \langle b',\sigma\rangle}{\langle\, !\, b,\ \sigma\rangle \to \langle\, !\, b',\sigma\rangle}$$

SMALLSTEP-NOTTRUE: $$\langle\, !\, \mathit{true},\ \sigma\rangle \to \langle \mathit{false},\sigma\rangle$$

SMALLSTEP-NOTFALSE: $$\langle\, !\, \mathit{false},\ \sigma\rangle \to \langle \mathit{true},\sigma\rangle$$

Small-step rules for IMP – 2: boolean expressions

Negations and conjunction:

SMALLSTEP-NOT: $$\frac{\langle b,\ \sigma\rangle \to \langle b', \sigma\rangle}{\langle\, !\, b,\ \sigma\rangle \to \langle\, !\, b', \sigma\rangle}$$

SMALLSTEP-NOTTRUE: $$\langle\, !\, \mathit{true},\ \sigma\rangle \to \langle \mathit{false}, \sigma\rangle$$

SMALLSTEP-NOTFALSE: $$\langle\, !\, \mathit{false},\ \sigma\rangle \to \langle \mathit{true}, \sigma\rangle$$

SMALLSTEP-AND1: $$\frac{\langle b_1, \sigma\rangle \to \langle b_1', \sigma\rangle}{\langle b_1 \ \texttt{and'}\ \ b_2, \sigma\rangle \to \langle b_1' \ \texttt{and'}\ \ b_2, \sigma\rangle}$$

SMALLSTEP-ANDFALSE: $$\langle \mathit{false}\ \ \texttt{and'}\ \ b_2,\ \sigma\rangle \to \langle \mathit{false}, \sigma\rangle$$

SMALLSTEP-ANDTRUE: $$\langle \mathit{true}\ \ \texttt{and'}\ \ b_2,\ \sigma\rangle \to \langle b_2, \sigma\rangle$$

Small-step rules for IMP – 2: boolean expressions

Comparisons:

SMALLSTEP-LT1: $$\frac{\langle a_1, \sigma\rangle \rightarrow \langle a_1', \sigma\rangle}{\langle a_1 \;<'\; a_2, \sigma\rangle \rightarrow \langle a_1' \;<'\; a_2, \sigma\rangle}$$

SMALLSTEP-LT2: $$\frac{\langle a_2, \sigma\rangle \rightarrow \langle a_2', \sigma\rangle}{\langle i_1 \;<'\; a_2, \sigma\rangle \rightarrow \langle i_1 \;<'\; a_2, \sigma\rangle}$$

SMALLSTEP-LT: $$\langle i_1 \;<'\; i_2,\, \sigma\rangle \rightarrow \langle i_1 <_{nat} i_2, \sigma\rangle$$

Small-step rules for IMP – 2: boolean expressions

Comparisons:

SMALLSTEP-LT1: $$\frac{\langle a_1, \sigma\rangle \rightarrow \langle a_1', \sigma\rangle}{\langle a_1 \ <' \ a_2, \sigma\rangle \rightarrow \langle a_1' \ <' \ a_2, \sigma\rangle}$$

SMALLSTEP-LT2: $$\frac{\langle a_2, \sigma\rangle \rightarrow \langle a_2', \sigma\rangle}{\langle i_1 \ <' \ a_2, \sigma\rangle \rightarrow \langle i_1 \ <' \ a_2, \sigma\rangle}$$

SMALLSTEP-LT: $$\langle i_1 \ <' \ i_2, \ \sigma\rangle \rightarrow \langle i_1 <_{nat} i_2, \sigma\rangle$$

SMALLSTEP-GT1: $$\frac{\langle a_1, \sigma\rangle \rightarrow \langle a_1', \sigma\rangle}{\langle a_1 \ >' \ a_2, \sigma\rangle \rightarrow \langle a_1' \ >' \ a_2, \sigma\rangle}$$

SMALLSTEP-GT2: $$\frac{\langle a_2, \sigma\rangle \rightarrow \langle a_2', \sigma\rangle}{\langle a_1 \ >' \ a_2, \sigma\rangle \rightarrow \langle a_1 \ >' \ a_2, \sigma\rangle}$$

SMALLSTEP-GT: $$\langle i_1 \ >' \ i_2, \ \sigma\rangle \rightarrow \langle i_1 >_{nat} i_2, \sigma\rangle$$

Derivation example

$$
\dfrac{
  \dfrac{
    \dfrac{\cdot}{\langle 1 \ +' \ 3, \sigma \rangle \rightarrow \langle 4, \sigma \rangle}\ \textsc{Add}
  }{
    \langle 1 \ +' \ 3 \ <' \ 5, \sigma \rangle \rightarrow \langle 4 \ <' \ 5, \sigma \rangle
  }\ \textsc{Lt1}
  \qquad
  \dfrac{
    \dfrac{\cdot}{\langle 4 \ <' \ 5, \sigma \rangle \rightarrow \langle \mathit{true}, \sigma \rangle}\ \textsc{Lt}
    \quad
    \dfrac{\cdot}{\langle \mathit{true}, \sigma \rangle \rightarrow^* \langle \mathit{true}, \sigma \rangle}\ \textsc{Refl}
  }{
    \langle 4 \ <' \ 5, \sigma \rangle \rightarrow^* \langle \mathit{true}, \sigma \rangle
  }\ \textsc{Tran}
}{
  \langle 1 \ +' \ 3 \ <' \ 5, \sigma \rangle \rightarrow^* \langle \mathit{true}, \sigma \rangle
}\ \textsc{Tran}
$$

Outline

Small-step rules for IMP – 3: statements

Assignments and Sequences:

SMALLSTEP-ASSIGN-2: $$\frac{\langle a, \sigma\rangle \rightarrow \langle a', \sigma\rangle}{\langle x \ \texttt{::=} \ a, \sigma\rangle \rightarrow \langle x \ \texttt{::=} \ a', \sigma\rangle}$$

SMALLSTEP-ASSIGN: $$\langle x \ \texttt{::=} \ i, \sigma\rangle \rightarrow \langle \texttt{skip}, \sigma[i/x]\rangle$$

SMALLSTEP-SEQ: $$\frac{\langle s_1, \sigma\rangle \rightarrow \langle s_1', \sigma'\rangle}{\langle s_1 \ \texttt{;;} \ s_2, \sigma\rangle \rightarrow \langle s_1' \ \texttt{;;} \ s_2, \sigma'\rangle}$$

SMALLSTEP-SKIP: $$\langle \texttt{skip} \ \texttt{;;} \ s_2, \sigma\rangle \rightarrow \langle s_2, \sigma\rangle$$

Small-step rules for IMP – 3: statements

Decisionals and loops:

SMALLSTEP-ITE: $$\frac{\langle b, \sigma\rangle \rightarrow \langle b', \sigma\rangle}{\langle \texttt{ite}\, b\, s_1\, s_2, \sigma\rangle \rightarrow \langle \texttt{ite}\, b'\, s_1\, s_2, \sigma\rangle}$$

SMALLSTEP-ITETRUE: $$\langle \texttt{ite btrue}\, s_1\, s_2, \sigma\rangle \rightarrow \langle s_1, \sigma\rangle$$

SMALLSTEP-ITEFALSE: $$\langle \texttt{ite bfalse}\, s_1\, s_2, \sigma\rangle \rightarrow \langle s_2, \sigma\rangle$$

SMALLSTEP-WHILE: $$\langle \texttt{while}\, b\, s, \sigma\rangle \rightarrow \langle \texttt{ite}\, b\, (s \,\texttt{;;}\, \texttt{while}\, b\, s)\, \texttt{skip}, \sigma\rangle$$

Derivation example

We show here a derivation example for the sequent

$$\langle x::=n, \sigma_1\rangle \to \langle \texttt{skip}, \sigma_2\rangle,$$

where $\sigma_1(n) = 10$ and $\sigma_2 = \sigma_1[10/x]$ (with rule labels):

$$\dfrac{\dfrac{\dfrac{\cdot}{\langle n, \sigma_1\rangle \to \langle 10, \sigma_1\rangle}\ \textsc{Lookup}}{\langle x::=n, \sigma_1\rangle \to \langle x::=10, \sigma_1\rangle}\ \textsc{Assign2} \qquad \dfrac{\dfrac{\cdot}{\langle x::=10, \sigma_1\rangle \to \langle \texttt{skip}, \sigma_2\rangle}\ \textsc{Assign} \qquad \dfrac{\cdot}{\langle \texttt{skip}, \sigma_2\rangle \to^* \langle \texttt{skip}, \sigma_2\rangle}\ \textsc{Refl}}{\langle x::=10, \sigma_1\rangle \to^* \langle \texttt{skip}, \sigma_2\rangle}\ \textsc{Tran}}{\langle x::=n, \sigma_1\rangle \to^* \langle \texttt{skip}, \sigma_2\rangle}\ \textsc{Tran}$$

Without labels:

$$\dfrac{\dfrac{\dfrac{\cdot}{\langle n, \sigma_1\rangle \to \langle 10, \sigma_1\rangle}}{\langle x::=n, \sigma_1\rangle \to \langle x::=10, \sigma_1\rangle} \qquad \dfrac{\dfrac{\cdot}{\langle x::=10, \sigma_1\rangle \to \langle \texttt{skip}, \sigma_2\rangle} \qquad \dfrac{\cdot}{\langle \texttt{skip}, \sigma_2\rangle \to^* \langle \texttt{skip}, \sigma_2\rangle}}{\langle x::=10, \sigma_1\rangle \to^* \langle \texttt{skip}, \sigma_2\rangle}}{\langle x::=n, \sigma_1\rangle \to^* \langle \texttt{skip}, \sigma_2\rangle}$$

Bibliography

1. Chapter Small-Step Operational Semantics in Software Foundations - Volume 2, 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/plf-current/Smallstep.html`
2. Small-step SOS:
   `http://fsl.cs.illinois.edu/images/6/65/CS422-Spring-2015-02c-SmallStep.pdf`