Semantics with Applications
Structural Operational Semantics

Pablo López

University of Málaga

December 1, 2021

Outline

Outline

Two Styles of Operational Semantics

Recall that there are two styles of operational semantics:

- **Natural Semantics** (*big-step* semantics) describe how the *overall* results are obtained from *initial* to *final* state
- **Structural Operational Semantics** (*small-step* semantics) describe how the individual steps change the states (initial, intermediate, and final)

Transition System

Recall that we model both operational semantics by **transition systems**.

A **transition system** is a tuple $(\Gamma, T, \triangleright)$ where:

- Γ is a set of **configurations**
- T a set of **terminal configurations** $T \subseteq \Gamma$
- $\triangleright$ is a **transition relation** $\triangleright \subseteq \Gamma \times \Gamma$

Configurations for `WHILE`

Recall that we define two types of **configurations**:

- $\langle S, s\rangle$ statement S is to be executed from the state s, and
- s terminal or final state

A configuration of the latter form is a **terminal configuration**.
The **natural** and **structural operational** semantics:

- use the same sets of configurations, Γ and T
- differ in the definition of the transition relation $\triangleright$.

Since `WHILE` is deterministic, we shall replace $\triangleright$ by $\rightarrow$ (natural semantics) or $\Rightarrow$ (structural operational semantics)

Transition System for Natural Semantics

Recall that the Natural Semantics of `WHILE` is defined by a transition system $(\Gamma, T, \rightarrow)$ where:

$$\begin{array}{rcl} \Gamma & = & \{\langle S, s\rangle \mid S \in \mathbf{Stm},\ s \in \mathbf{State}\} \cup \mathbf{State} \\ T & = & \mathbf{State} \\ \rightarrow & \subseteq & \{\langle S, s\rangle \mid S \in \mathbf{Stm},\ s \in \mathbf{State}\} \times \mathbf{State} \end{array}$$

Outline

Transition System for Structural Operational Semantics

The Structural Operational Semantics of `WHILE` is defined by a transition system $(\Gamma, T, \Rightarrow)$ where:

$$
\begin{array}{rcl}
\Gamma & = & \{\langle S, s\rangle \mid S \in \mathbf{Stm},\ s \in \mathbf{State}\} \cup \mathbf{State} \\
T & = & \mathbf{State} \\
\Rightarrow & \subseteq & \{\langle S, s\rangle \mid S \in \mathbf{Stm},\ s \in \mathbf{State}\} \times \Gamma
\end{array}
$$

Fundamentals of Structural Operational Semantics

- We are concerned with the **initial**, **intermediate**, and **final** configurations
- The transition relation $\langle S,\ s\rangle \Rightarrow \gamma$ expresses the **first step** of the execution of S from state s for each statement of `WHILE`
- A transition of the form

$$\langle S, s\rangle \Rightarrow \langle S', s'\rangle$$

  means that the execution of S from s is **not completed**; the intermediate configuration $\langle S', s'\rangle$ represents the remaining computation
- A transition of the form

$$\langle S, s\rangle \Rightarrow s'$$

  means that the execution of S from s has **terminated**, yielding the final state s'.
- A configuration $\langle S,\ s\rangle$ is **stuck** if there is no γ such that $\langle S,\ s\rangle \Rightarrow \gamma$

Structural Operational Semantics for WHILE

The transition relation $\Rightarrow$ is defined by a set of axioms and rules.

$[\text{ass}_{\text{sos}}]$	$\langle x := a,\ s\rangle \Rightarrow s[x \mapsto \mathcal{A}[\![a]\!]s]$
$[\text{skip}_{\text{sos}}]$	$\langle \texttt{skip},\ s\rangle \Rightarrow s$
$[\text{comp}^1_{\text{sos}}]$	$\dfrac{\langle S_1,\ s\rangle \Rightarrow \langle S_1',\ s'\rangle}{\langle S_1;S_2,\ s\rangle \Rightarrow \langle S_1';S_2,\ s'\rangle}$
$[\text{comp}^2_{\text{sos}}]$	$\dfrac{\langle S_1,\ s\rangle \Rightarrow s'}{\langle S_1;S_2,\ s\rangle \Rightarrow \langle S_2,\ s'\rangle}$
$[\text{if}^{\text{tt}}_{\text{sos}}]$	$\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2,\ s\rangle \Rightarrow \langle S_1,\ s\rangle$ if $\mathcal{B}[\![b]\!]s = \mathbf{tt}$
$[\text{if}^{\text{ff}}_{\text{sos}}]$	$\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2,\ s\rangle \Rightarrow \langle S_2,\ s\rangle$ if $\mathcal{B}[\![b]\!]s = \mathbf{ff}$
$[\text{while}_{\text{sos}}]$	$\langle \texttt{while}\ b\ \texttt{do}\ S,\ s\rangle \Rightarrow$ $\langle \texttt{if}\ b\ \texttt{then}\ (S;\ \texttt{while}\ b\ \texttt{do}\ S)\ \texttt{else}\ \texttt{skip},\ s\rangle$

Table 2.2: Structural operational semantics for **While**

The Assignment Statement $:=$

$$[\text{ass}_{\text{sos}}] \quad \langle x := a, s\rangle \Rightarrow s[x \mapsto \mathcal{A}[\![a]\!]s]$$

- executed by updating the value of x in s with the value of the arithmetic expression a in the state s
- same as in natural semantics because it is fully executed in one step

The `skip` Statement

$$[\text{skip}_{\text{sos}}] \quad \langle \texttt{skip}, s\rangle \Rightarrow s$$

- does not modify the state s
- same as in natural semantics because it is fully executed in one step

The ; Statement

Sequential composition; imposes sequential order: first *complete* S_1, then proceed with S_2

$$[\text{comp}^1_{\text{sos}}] \quad \frac{\langle S_1, s\rangle \Rightarrow \langle S_1', s'\rangle}{\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s'\rangle}$$

- S_1 has not been completed; proceed with S_1' before starting on S_2

$$[\text{comp}^2_{\text{sos}}] \quad \frac{\langle S_1, s\rangle \Rightarrow s'}{\langle S_1; S_2, s\rangle \Rightarrow \langle S_2, s'\rangle}$$

- S_1 has been completed, proceed with S_2

The `if then else` Statement

We need two axioms, discriminated by a condition on the guard b:

$$[\mathrm{if}^{\mathrm{tt}}_{\mathrm{sos}}] \quad \langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2, s\rangle \Rightarrow \langle S_1, s\rangle \quad \text{if } \mathcal{B}[\![b]\!]s = \texttt{tt}$$

$$[\mathrm{if}^{\mathrm{ff}}_{\mathrm{sos}}] \quad \langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2, s\rangle \Rightarrow \langle S_2, s\rangle \quad \text{if } \mathcal{B}[\![b]\!]s = \texttt{ff}$$

- the **first step** in executing a conditional is to evaluate the test and select the appropriate branch

The `while` Statement

We need one axiom:

$$[\text{while}_{\text{sos}}] \quad \langle \texttt{while}\ b\ \texttt{do}\ S, s\rangle \Rightarrow \langle \texttt{if}\ b\ \texttt{then}\ (S; \texttt{while}\ b\ \texttt{do}\ S)\ \texttt{else}\ \texttt{skip}, s\rangle$$

- the **first step** in executing a loop is to unfold it one level

Derivation Sequences

A **derivation sequence** of a statement S starting in state s is either:

1. a **finite sequence**

   $$\gamma_0 \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \cdots \Rightarrow \gamma_k$$

   where $\gamma_0 = \langle S,\ s\rangle$, $\gamma_i \Rightarrow \gamma_{i+1}$ for $0 \leq i < k$, $k \geq 0$ and γ_k is either a terminal or stuck configuration.

2. an **infinite sequence**

   $$\gamma_0 \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \cdots$$

   where $\gamma_0 = \langle S,\ s\rangle$, $\gamma_i \Rightarrow \gamma_{i+1}$ for $0 \leq i$.

Execution of a Finite Number of Steps

We shall write

- $\gamma \Rightarrow^{i} \gamma_i$ to indicate that there are i steps in the execution from γ_0 to γ_i, and
- $\gamma \Rightarrow^{\star} \gamma_i$ to indicate that there is a finite number of steps in the execution from γ_0 to γ_i

Note that $\gamma \Rightarrow^{i} \gamma_i$ and $\gamma \Rightarrow^{\star} \gamma_i$ are not necessarily derivation sequences: they will be if and only if γ_i is either a terminal or stuck configuration.

Derivation Sequence Example

For the statement:

```
(z:= x; x:= y); y:= z
```

and the state $s\ \mathrm{x} = 5$, $s\ \mathrm{y} = 7$, $s\ \_ = 0$ we have the derivation sequence:

$$
\begin{aligned}
&\langle(\mathrm{z} := \mathrm{x};\ \mathrm{x} := \mathrm{y});\ \mathrm{y} := \mathrm{z},\ s_0\rangle \\
&\Rightarrow \langle \mathrm{x} := \mathrm{y};\ \mathrm{y} := \mathrm{z},\ s_0[\mathrm{z}\mapsto\mathbf{5}]\rangle \\
&\Rightarrow \langle \mathrm{y} := \mathrm{z},\ (s_0[\mathrm{z}\mapsto\mathbf{5}])[\mathrm{x}\mapsto\mathbf{7}]\rangle \\
&\Rightarrow ((s_0[\mathrm{z}\mapsto\mathbf{5}])[\mathrm{x}\mapsto\mathbf{7}])[\mathrm{y}\mapsto\mathbf{5}]
\end{aligned}
$$

Derivation Tree Example

For each of the steps of the previous derivation sequence we have derivation trees like this one:

$$\dfrac{\dfrac{\langle \mathtt{z} := \mathtt{x},\ s_0\rangle \Rightarrow s_0[\mathtt{z}\mapsto\mathbf{5}]}{\langle \mathtt{z} := \mathtt{x};\ \mathtt{x} := \mathtt{y},\ s_0\rangle \Rightarrow \langle \mathtt{x} := \mathtt{y},\ s_0[\mathtt{z}\mapsto\mathbf{5}]\rangle}}{\langle (\mathtt{z} := \mathtt{x};\ \mathtt{x} := \mathtt{y});\ \mathtt{y} := \mathtt{z},\ s_0\rangle \Rightarrow \langle \mathtt{x} := \mathtt{y};\ \mathtt{y} := \mathtt{z},\ s_0[\mathtt{z}\mapsto\mathbf{5}]\rangle}$$

Exercises

Exercise. Assume that $s\ \mathrm{x} = 3$. Execute the statement:

```
y:= 1; while !(x = 1) do (y:= y*x; x:= x-1)
```

until you reach a configuration with state $s[\mathrm{y} \mapsto 3][\mathrm{x} \mapsto 2]$.
Exercise 2.16 Construct a derivation sequence for the statement:

```
z:= 0; while y <= x do (z:= z+1; x:= x-y)
```

when executed in a state $s\ \mathrm{x} = 17$, $s\ \mathrm{y} = 5$, $s\ \_ = 0$. Determine a state s such that the derivation sequence is infinite.

Termination, Successful Termination, and Looping

- The execution of a statement S on a state s
  - **terminates** if and only if there is a finite derivation sequence starting with $\langle S,\ s\rangle$
  - **terminates successfully** if $\langle S,\ s\rangle \Rightarrow^\star s'$ for some state s'
  - **loops** if and only if there is an infinite derivation sequence starting with $\langle S,\ s\rangle$
- Since in `WHILE` there are no stuck configurations, an execution terminates successfully if and only if it terminates
- The execution of a statement S
  - **always terminates** if it terminates for all choices of s
  - **always loops** if it loops for all choices of s

Exercises

Exercise 2.17 Extend the `WHILE` language with the statement

```
repeat S until b
```

and define the relation $\Rightarrow$ for it. You are not allowed to rely on the `while` construct.

Exercise 2.18 Extend the `WHILE` language with the statement

```
for x:= a1 to a2 do S
```

and define the relation $\Rightarrow$ for it. You are not allowed to rely on the `while` construct. *Hint*: Assume that you have an inverse to $\mathcal{N}$ to compute the numeral from a given number.

Outline

Yet Another Induction Principle

For Structural Operational Semantics it is often useful to conduct proofs by induction on the length of the **finite** derivation sequences.

Induction on the Length of Derivation Sequences

1: Prove that the property holds for all derivation sequences of length 0.

2: Prove that the property holds for all other derivation sequences: Assume that the property holds for all derivation sequences of length at most k (this is called the *induction hypothesis*) and show that it holds for derivation sequences of length k+1.

The **induction step** will often inspect:

- the structure of the syntactic element, or
- the derivation tree validating the first step of the derivation sequence

Splitting a Composition

Lemma 2.19 If $\langle S_1;\ S_2,\ s\rangle \Rightarrow^k s''$, then there exists a state s' and $k_1, k_2 \in \mathbb{N}$ such that $k = k_1 + k_2$, $\langle S_1, s\rangle \Rightarrow^{k_1} s'$, and $\langle S_2, s'\rangle \Rightarrow^{k_2} s''$.
Proof: By induction on the length of the derivation

$$\langle S_1;\ S_2,\ s\rangle \Rightarrow^k s''$$

.

Exercises

Exercise 2.20 Suppose that $\langle S_1;\ S_2,\ s\rangle \Rightarrow^\star \langle S_2,\ s'\rangle$. Show that it is *not* necessarily the case that $\langle S_1,\ s\rangle \Rightarrow^\star s'$.
Exercise 2.21 (Non-interference of statements) Prove that

$$\text{if} \quad \langle S_1,\ s\rangle \Rightarrow^k s' \quad \text{then} \quad \langle S_1;\ S_2,\ s\rangle \Rightarrow^k \langle S_2,\ s'\rangle$$

That is, the execution of S_1 is not influenced by the statement following it.

Deterministic Structural Operational Semantics

A Structural Operational Semantics is **deterministic** if for all choices of S, s, γ and γ' we have that

$$\langle S,\ s\rangle \Rightarrow \gamma \quad \text{and} \quad \langle S,\ s\rangle \Rightarrow \gamma' \quad \text{imply} \quad \gamma = \gamma'$$

Exercise 2.22 Show that the Structural Operational Semantics of `WHILE` is deterministic. Deduce that there is exactly one derivation sequence starting in a configuration $\langle S,\ s\rangle$. Argue that S cannot both terminate and loop from s and hence cannot both be always terminating and always looping.

Semantic Equivalence

Two statements S_1 and S_2 are **semantically equivalent** if for all states:

- $\langle S_1,\ s\rangle \Rightarrow^\star \gamma$ if and only if $\langle S_2,\ s\rangle \Rightarrow^\star \gamma$ where γ is either a terminal or stuck configuration
- there is an infinite derivation sequence starting in $\langle S_1, s\rangle$ if and only if there is one starting in $\langle S_2, s\rangle$

Note that the length of the two finite derivation sequences may be different.

Exercises

Exercise 2.23 Show that the following statements of `WHILE` are semantically equivalent:

- `S; skip` and `S`
- `while b do S` and `if b then (S; while b do S)else skip`
- `S1; (S2; S3)` and `(S1; S2); S3`

You may use the fact that `WHILE` is deterministic.

Exercise 2.24 Prove that `repeat S until b` is semantically equivalent to `S; while !b do S`

The Semantic Function $\mathcal{S}_{\text{sos}}$

The *meaning* of statements is given by the *partial* function:

$$\mathcal{S}_{\text{sos}} : \textbf{Stm} \rightarrow (\textbf{State} \hookrightarrow \textbf{State})$$

defined as:

$$\mathcal{S}_{\text{sos}}[\![S]\!]s = \begin{cases} s' & \text{if } \langle S,\ s\rangle \Rightarrow^{\star} s' \\ \textbf{undef} & \text{otherwise} \end{cases}$$

Exercise 2.25 Determine whether or not semantic equivalence of S_1 and S_2 amounts to $\mathcal{S}_{\text{sos}}[\![S_1]\!] = \mathcal{S}_{\text{sos}}[\![S_2]\!]$.