Programming Language Semantics and Compiler Design / Sémantique des Langages de Programmation et Compilation

Structural Operational Semantics of Language **While** and some Extensions

Yliès Falcone

ylies.falcone@univ-grenoble-alpes.fr — www.ylies.fr

Univ. Grenoble Alpes, and LIG-Inria team CORSE

Master of Sciences in Informatics at Grenoble (MoSIG)
Master 1 info

Univ. Grenoble Alpes - UFR IM^{2}AG
www.univ-grenoble-alpes.fr - im2ag.univ-grenoble-alpes.fr

Academic Year 2017 - 2018

Outline - Structural Operational Semantics of Language **While** and some Extensions

Outline - Structural Operational Semantics of Language **While** and some Extensions

Structural Operational Semantics: intuition

Emphasis on *individual steps* of the execution:

- tests (of Boolean expression/condition),
- assignments.

Transitions: $(S, \sigma) \Rightarrow \gamma$

The result γ of an execution step can either be:

- (S', σ'): the execution is *not completed*, or
- σ': the execution *has terminated*.

Transition system: natural vs structural semantics

An operational semantics defines a transition system, i.e., a 3-tuple $(\Gamma, T, \rightarrow)$ where:

- Γ is the configuration set
- $T \subseteq \Gamma$ is the subset of final configurations
- $\rightarrow \subseteq \Gamma \times \Gamma$ is the transition relation

For the natural operational semantics:

1. $\Gamma = (\mathbf{Stm} \times \mathbf{State}) \cup \mathbf{State}$
2. $T = \mathbf{State}$
3. $\rightarrow$ defined by inference trees

For the structural operational semantics:

1. $\Gamma = (\mathbf{Stm} \times \mathbf{State}) \cup \mathbf{State}$
2. $T = \mathbf{State}$
3. $\Rightarrow$ defined by inference sequences

Structural Operation Semantics: Inference System

Goal: Describe how (i.e., step by step) the result of an execution is obtained.

Axioms

$$(x := a, \sigma) \Rightarrow \sigma[x \mapsto \mathcal{A}[a]\sigma]$$

$$(\mathrm{skip}, \sigma) \Rightarrow \sigma$$

Rules for sequential statements

$$\frac{(S_1, \sigma) \Rightarrow \sigma'}{(S_1; S_2, \sigma) \Rightarrow (S_2, \sigma')}$$

"execution of S_1 *has* terminated"

$$\frac{(S_1, \sigma) \Rightarrow (S_1', \sigma')}{(S_1; S_2, \sigma) \Rightarrow (S_1'; S_2, \sigma')}$$

"execution of S_1 *has not* terminated"

Structural Operational Semantics: Inference System (ctd)

Rules for conditional statements

- If $\mathcal{B}[b]\sigma = \mathbf{tt}$ then

$$(\text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi}, \sigma) \Rightarrow (S_1, \sigma)$$

- If $\mathcal{B}[b]\sigma = \mathbf{ff}$ then

$$(\text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi}, \sigma) \Rightarrow (S_2, \sigma)$$

Rule for iterative statements (unbounded)

$$(\text{while } b \text{ do } S \text{ od}, \sigma) \Rightarrow (\text{if } b \text{ then } (S; \text{while } b \text{ do } S \text{ od}) \text{ else skip fi}, \sigma)$$

Inference Sequence

- Finite sequence:

$$\gamma_1, \gamma_1, \ldots, \gamma_k$$

  where:
  - $\gamma_i \Rightarrow \gamma_{i+1}$, for $i \in [1, k-1]$, and
  - $\gamma_k \not\Rightarrow$
    i.e., there is no configuration γ with $\gamma_k \Rightarrow \gamma$
    if γ_k is not a final configuration, it is said to be a blocking configuration
- Infinite sequence:

$$\gamma_1, \gamma_2, \ldots \text{ where}$$

  $\gamma_i \Rightarrow \gamma_{i+1}$, for $i \geq 1$

Executions are the maximal inference sequences.

Outline - Structural Operational Semantics of Language **While** and some Extensions

Discussion about both operational semantics

How the two operational semantics model program divergence?

Consider a statement S and a state σ:

- Natural semantics:
  S *diverges* in σ, if (S, σ) *does not have a successor*:

  $$(S, \sigma) \not\to \text{ i.e., } \nexists \gamma \cdot (S, \sigma) \to \gamma$$

  (equivalently there exists an infinite derivation tree)
- Structural semantics:
  S diverges in σ if *there exists an infinite inference sequence* starting from (S, σ).

Semantic equivalence

Semantic equivalence in Natural Semantics

S_1 and S_2 are semantically equivalent, if for all states σ and σ':

$$(S_1, \sigma) \to \sigma' \text{ iff } (S_2, \sigma) \to \sigma'$$

Semantic equivalence in Structural semantics

S_1 and S_2 are semantically equivalent, if for all states σ

- for any final or blocking configuration γ:

$$(S_1, \sigma) \Rightarrow^* \gamma \text{ iff } (S_2, \sigma) \Rightarrow^* \gamma$$

- there exists an infinite inference sequence starting from (S_1, σ)
  iff
  there exists an infinite inference sequence starting from (S_2, σ).

The $\mathcal{S}_{\text{sos}}$ semantic function

Definition (The $\mathcal{S}_{\text{sos}}$ semantic function)

$$\mathcal{S}_{\text{sos}}[S]\sigma = \begin{cases} \sigma' & \text{if } (S,\sigma) \Rightarrow^* \sigma' \\ \text{undef} & \text{otherwise} \end{cases}$$

Question: Do we have $\mathcal{S}_{\text{ns}} = \mathcal{S}_{\text{sos}}$?

Lemma (NOS "simulates" SOS)

For every statement S in **Stm**, *states σ and σ' in* **State**:

$$(S,\sigma) \Rightarrow^k \sigma' \textit{ implies } (S,\sigma) \rightarrow \sigma'$$

Lemma (SOS "simulates" NOS)

For every statement S in **Stm**, *states σ and σ' in* **State**:

$$(S,\sigma) \rightarrow \sigma' \textit{ implies } (S,\sigma) \Rightarrow^* \sigma'$$

Theorem: equivalence of NOS and SOS for **While**

For every statement S in **Stm**: $\mathcal{S}_{\text{ns}}[S] = \mathcal{S}_{\text{sos}}[S]$.

Proof.

In the exercise sessions. □

Semantic styles and associated proof patterns

Inductive semantics: (e.g., $\mathcal{A}$)
Construction using composition rules
$\rightarrow$ Proofs by structural induction on the (syntax of) the arithmetic expressions

Natural operational semantics ("big steps/bird-eye view" of executions)
Transition relation defined by inference/derivation trees.
$\rightarrow$ Proofs by inductions on the structure of the inference trees.

Structural operational semantics ("small steps/fine-grain view" of executions)
Transition relation defined by inference/derivation sequences.
$\rightarrow$ Proofs by induction on the length of the inference sequences.

Intermediate Lemma: SOS "simulates" NOS

Lemma (Composing statements)

*For every statement $S_1, S_2 \in$ **Stm**, state $\sigma \in$ **State**, and $k \in \mathbb{N}$,*

$$(S_1, \sigma) \Rightarrow^k \sigma' \textit{ implies } (S_1; S_2, \sigma) \Rightarrow^k (S_2; \sigma')$$

(Executing a statement is not influenced by the sequentially-composed statement)

Proof.

By induction on $k \in \mathbb{N}$ (see the tutorial exercises). □

Remark The converse does not hold in general. □

Lemma (SOS "simulates" NOS)

*For every statement $S \in$ **Stm**, state $\sigma, \sigma' \in$ **State**,*

$$(S, \sigma) \to \sigma' \textit{ implies } (S, \sigma) \Rightarrow^* \sigma'.$$

Proof.

By induction on the structure of the derivation tree of $(S, \sigma) \to \sigma'$. □

Intermediate Lemma: SOS "simulates" NOS

Proof of SOS "simulates" NOS.

By induction on the structure of the derivation tree of $(S, \sigma) \to \sigma'$. That is, we distinguish cases according to the rule that has been applied to obtain $(S, \sigma) \to \sigma'$.

[$\texttt{ass}_{\texttt{ns}}$] We have $(x := a, \sigma) \to \sigma[x \mapsto \mathcal{A}[a]\sigma]$, that is S is necessarily a statement of the form $x := a$ (for some $x \in \textbf{Var}$ and $a \in \textbf{Aexp}$) and $\sigma' = \sigma[x \mapsto \mathcal{A}[a]\sigma$ (unique possibility for the rule to be applied). Moreover, according to [$\texttt{ass}_{\texttt{sos}}$], we have $(x := a, \sigma) \Rightarrow \sigma[x \mapsto \mathcal{A}[a]\sigma]$.

[$\texttt{skip}_{\texttt{ns}}$] Analogous to the previous case.

[$\texttt{comp}_{\texttt{ns}}$] We have:

$$\frac{(S_1, \sigma) \to \sigma'' \quad (S_2, \sigma'') \to \sigma'}{(S_1; S_2, \sigma) \to \sigma'}$$

That is, S is necessarily of the form $S_1; S_2$ and σ' is obtained as described by the rule. We apply the induction hypothesis to the two premisses $(S_1, \sigma) \to \sigma''$ and $(S_2, \sigma'') \to \sigma'$ to obtain $(S_1, \sigma) \Rightarrow^* \sigma''$ $(S_2, \sigma'') \Rightarrow^* \sigma'$, respectively.
From the lemma (composing statements), we obtain $(S_1; S_2, \sigma) \Rightarrow^* (S_2, \sigma'')$. And, now using $(S_2, \sigma'') \Rightarrow^* \sigma'$ again, we find $(S_1; S_2, \sigma) \Rightarrow^* \sigma'$.

$\square$

Intermediate Lemma: SOS "simulates" NOS

Proof of SOS "simulates" NOS (ctd).

By induction on the structure of the derivation tree of $(S, \sigma) \to \sigma'$.

$[\texttt{if}^{\texttt{tt}}_{\texttt{ns}}]$ We have

$$\frac{(S_1, \sigma) \to \sigma'}{(\text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi}, \sigma) \to \sigma'} \; \mathcal{B}[b]\sigma = \textbf{tt}$$

That is, S is necessarily of the form if b then S_1 else S_2 fi, for some $S_1, S_2 \in \textbf{Stm}$. Moreover, $(\text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi}, \sigma) \to \sigma'$ holds because $\mathcal{B}[b]\sigma = \textbf{tt}$ and $(S_1, \sigma) \to \sigma'$.

Since, $\mathcal{B}[b]\sigma = \textbf{tt}$, we get $(\text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi}, \sigma) \Rightarrow (S_1, \sigma) \Rightarrow^* \sigma'$ because $[\texttt{if}^{\texttt{tt}}_{\texttt{ns}}]$. Moreover, applying the induction hypothesis applied to the premise $(S_1, \sigma) \to \sigma'$, we get $(S_1, \sigma) \Rightarrow^* \sigma'$. From $(S, \sigma) \Rightarrow \sigma$ and $(S_1, \sigma) \Rightarrow^* \sigma'$, we obtain $(S, \sigma) \Rightarrow^* \sigma'$

$[\texttt{if}^{\texttt{ff}}_{\texttt{ns}}]$ Analogous.

$\square$

Intermediate Lemma: SOS "simulates" NOS

Proof of SOS "simulates" NOS (ctd).

By induction on the structure of the derivation tree of $(S, \sigma) \to \sigma'$.

$[\mathtt{while}_{\mathtt{ns}}^{\mathtt{tt}}]$ We have:

$$\frac{(S', \sigma) \to \sigma'' \quad (\text{while } b \text{ do } S' \text{ od}, \sigma'') \to \sigma'}{(\text{while } b \text{ do } S' \text{ od}, \sigma) \to \sigma'} \; \mathcal{B}[b]\sigma = \mathbf{tt}$$

That is, we assume that $(\text{while } b \text{ do } S' \text{ od}, \sigma) \to \sigma'$ holds because $\mathcal{B}[b]\sigma = \mathbf{tt}$, $(S, \sigma) \to \sigma''$ and $(\text{while } b \text{ do } S' \text{ od}, \sigma'') \to \sigma'$, for some $S' \in \mathbf{Stm}$, $\sigma'' \in \mathbf{State}$.
The induction hypothesis can be applied to both of the premises $(S', \sigma) \to \sigma''$ and $(\text{while } b \text{ do } S' \text{ od}, \sigma'') \to \sigma'$ and gives $(S', \sigma) \Rightarrow^* \sigma''$ and $(\text{while } b \text{ do } S' \text{ od}, \sigma'') \Rightarrow^* \sigma'$. Using the intermediate lemma (composing statements), we get $(S'; \text{while } b \text{ do } S' \text{ od}, \sigma) \Rightarrow^* \sigma'$. Then, we have the following derivation:

$$\begin{array}{lll}
(\text{while } b \text{ do } S' \text{ od}, \sigma) & \Rightarrow (\text{if } b \text{ then } S'; \text{while } b \text{ do } S' \text{ od else skip fi}, \sigma) & ([\mathtt{while}_{\mathtt{sos}}]) \\
& \Rightarrow (S'; \text{while } b \text{ do } S' \text{ od}, \sigma) & ([\mathtt{if}_{\mathtt{sos}}^{\mathtt{tt}}] \text{ and} \\
& & \mathcal{B}[b]\sigma = \mathbf{tt}) \\
& \Rightarrow^* \sigma' &
\end{array}$$

$[\mathtt{while}_{\mathtt{ns}}^{\mathtt{tt}}]$ Straightforward.

$\square$

Intermediate Lemma: NOS "simulates" SOS

We need an additional intermediate lemma.

Lemma (Decomposing computations in SOS)

$(S_1; S_2, \sigma) \Rightarrow^k \sigma''$ *implies*
there exist σ' *and* k_1 *s.t.* $(S_1, \sigma) \Rightarrow^{k_1} \sigma'$ *and* $(S_2, \sigma') \Rightarrow^{k-k_1} \sigma''$.

Proof.

By induction on $k \in \mathbb{N}$ in $(S_1; S_2, \sigma) \Rightarrow^k \sigma''$ (see the tutorial exercises). □

Lemma (NOS "simulates" SOS)

For all S, σ, and σ'

$$(S, \sigma) \Rightarrow^k \sigma' \textit{ implies } (S, \sigma) \rightarrow \sigma'.$$

Proof.

By induction on $k \in \mathbb{N}$ in $(S, \sigma) \Rightarrow^k \sigma'$, i.e., the length of the derivation sequence. □

Intermediate Lemma: NOS "simulates" SOS

NOS "simulates" SOS.

Let us assume that the result holds for all natural numbers lower than or equal to a natural number k and we shall prove the result for $k+1$. We distinguish how the first step of $(S,\sigma) \Rightarrow^{k+1} \sigma'$ is obtained, that is we inspect the first step of the derivation sequence in the computation in SOS.

$[\mathtt{ass}_{\mathtt{sos}}]$ Straightforward (and $k=0$).

$[\mathtt{skip}_{\mathtt{sos}}]$ Straightforward (and $k=0$).

$[\mathtt{comp}_{\mathtt{x}}]$ Cases $[\mathtt{comp}^1_{\mathtt{sos}}]$ and $[\mathtt{comp}^2_{\mathtt{sos}}]$. Necessarily, S is of the form $S_1; S_2$ and we assume that $(S_1; S_2, \sigma) \Rightarrow^{k+1} \sigma''$. We can apply the intermediate lemma to get that there exist $\sigma' \in \textbf{State}$ and $k_1, k_2 \in \mathbb{N}$ s.t.

$$(S_1, \sigma) \Rightarrow^{k_1} \sigma' \text{ and } (S_2, \sigma') \Rightarrow^{k_2} \sigma''$$

where $k_1 + k_2 = k+1$. The induction hypothesis can be applied to each of these derivation sequences because $k_1 \leq k$ and $k_2 \leq k$. Hence

$$(S_1, \sigma) \to \text{ and } (S_2, \sigma') \to \sigma''$$

Using $[\mathtt{comp}_{\mathtt{ns}}]$, we get $(S_1; S_2, \sigma) \to \sigma''$.

□

Intermediate Lemma: NOS "simulates" SOS

NOS "simulates" SOS.

$[\mathtt{if}^{\mathtt{tt}}_{\mathtt{sos}}]$ Assume that $\mathcal{B}[b]\sigma = \mathbf{tt}$ and that

$$(\text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi}, \sigma) \Rightarrow (S_1, \sigma) \Rightarrow^k \sigma'.$$

The induction hypothesis can be applied to the derivation sequence $(S_1, \sigma) \Rightarrow^k \sigma'$ and gives $(S_1, \sigma) \rightarrow \sigma'$. The result follows using $[\mathtt{if}^{\mathtt{tt}}_{\mathtt{ns}}]$.

$[\mathtt{if}^{\mathtt{ff}}_{\mathtt{sos}}]$ Analogous to the previous case.

$[\mathtt{while}^{\mathtt{tt}}_{\mathtt{sos}}]$ We have:

$$\begin{array}{ll}(\text{while } b \text{ do } S \text{ od}, \sigma) & \Rightarrow (\text{if } b \text{ then } S; \text{while } b \text{ do } S \text{ od else skip fi}, \sigma) \\ & \Rightarrow^k \sigma''\end{array}$$

The induction hypothesis can be applied to the k last steps of the derivation sequences and gives

$$(\text{if } b \text{ then } S; \text{while } b \text{ do } S \text{ od else skip fi}, \sigma) \rightarrow \sigma''$$

and we get the required $(\text{while } b \text{ do } S \text{ od}, \sigma) \rightarrow \sigma''$ (since while b do S od is semantically equivalent to if b then S; while b do S od else skip fi).

$\square$

Outline - Structural Operational Semantics of Language **While** and some Extensions

Outline - Structural Operational Semantics of Language **While** and some Extensions

Extending **While** with abnormal termination

Definition (Introducing statement **abort**)

- Statement abort: used to represent abnormal terminating computations
- "Behaves" differently than previous statements:
  $\hookrightarrow$ It stops the program
  - different from while true do skip od, and
  - different from skip.
- Configuration (abort, σ) has no successors (blocking):

$$\text{for all } \sigma \in \textbf{State}, (\text{abort}, \sigma) \not\Rightarrow \wedge (\text{abort}, \sigma) \not\to$$

  $\hookrightarrow$ we *do not* add any rule to the transitions systems

Examples with abort

Example (Program with possibly abnormal termination)

var *sensor* := some initial value
...
sensor := *read*(...)
if *sensor* $<$ 0 then abort else skip fi

Exercise: Assertions

Using abort, extend language **While** with

assert b before S

Comparison of **abort** in natural and structural semantics

In natural operational semantics:

- abort, and
- while true do skip od,

are semantically equivalent.

In natural operational semantics:

- abort, and
- skip,

are not semantically equivalent.

In structural operational semantics:

- while true do skip od,
- abort, and
- skip,

are pair-wise not semantically equivalent.

Outline - Structural Operational Semantics of Language **While** and some Extensions

Operator "**or**"

Definition (Or operator)

- Non-determinism
- Statement S_1or S_2

We note the new language obtained **While**^{or}

Example

We shall expect that the execution of the statement

$$x := 1 \quad \textbf{or} \quad x := 2$$

could result in a state where x has the value 1 or 2

The “or” operator: extending the transition system

Definition (Extending the transition system)

- Configurations:

$$\{(S, \sigma) \mid S \in \textbf{While}^{or} \wedge \sigma \in \textbf{State}\} \cup \textbf{State}$$

- Natural semantics:

$$\frac{(S_1, \sigma) \rightarrow \sigma'}{(S_1 \text{ or } S_2, \sigma) \rightarrow \sigma'}$$

$$\frac{(S_2, \sigma) \rightarrow \sigma'}{(S_1 \text{ or } S_2, \sigma) \rightarrow \sigma'}$$

- Structural semantics:

$$(S_1 \text{ or } S_2, \sigma) \Rightarrow (S_1, \sigma)$$

$$(S_1 \text{ or } S_2, \sigma) \Rightarrow (S_2, \sigma)$$

Discussion on **or** and non-termination

With natural operational semantics, "or" **hides non-termination.**

Example

- $S_1 =$ while true do skip od
- $S_2 =$ while false do skip od

Comparing semantics

- natural semantics:

$$(S_1 \text{ or } S_2, \sigma) \to \sigma$$

$\hookrightarrow$ one derivation tree

- structural semantics: $(S_1 \text{ or } S_2, \sigma)$ always has an infinite inference sequence.

Natural/structural operational semantics and looping

- **In NOS, non-determinism suppresses looping, if possible.**
- **In SOS, non-determinism does not suppress looping.**

Outline - Structural Operational Semantics of Language **While** and some Extensions

Parallel execution

We add a statement noted $\|$: $S_1 \parallel S_2$.

We denote this new language $\mathbf{While}^{\|}$.

Definition (Extending the transition system for parallel execution)

- Configuration: $\{(S,\sigma) \mid S \in \mathbf{While}^{\|}, \sigma \in \mathbf{State}\} \cup \mathbf{State}$.
- Transition rules for $\|$
  - Natural semantics:

$$\frac{(S_1,\sigma) \to \sigma' \quad (S_2,\sigma') \to \sigma''}{(S_1 \parallel S_2,\sigma) \to \sigma''} \qquad \frac{(S_2,\sigma) \to \sigma' \quad (S_1,\sigma') \to \sigma''}{(S_1 \parallel S_2,\sigma) \to \sigma''}$$

    $\to$ Executions of immediate constituents are atomic.
  - Structural semantics:

$$\frac{(S_1,\sigma) \Rightarrow (S_1',\sigma')}{(S_1 \parallel S_2,\sigma) \Rightarrow (S_1' \parallel S_2,\sigma')} \qquad \frac{(S_2,\sigma) \Rightarrow (S_2',\sigma')}{(S_1 \parallel S_2,\sigma) \Rightarrow (S_1 \parallel S_2',\sigma')}$$

$$\frac{(S_1,\sigma) \Rightarrow \sigma'}{(S_1 \parallel S_2,\sigma) \Rightarrow (S_2,\sigma')} \qquad \frac{(S_2,\sigma) \Rightarrow \sigma'}{(S_1 \parallel S_2,\sigma) \Rightarrow (S_1,\sigma')}$$

Discussion about the parallelism and interleaving

Example (Parallel execution)

Consider statement

$$x := 1 \parallel (x := 2; x := x + 2)$$

- natural operational semantics: 2 possible ending states
- structural operational semantics: 3 possible ending states

Natural vs Structural semantics and interleaving

- Natural semantics:
  - does not allow to express interleaving
  - executions of atomic constituents are atomic
- Structural semantics:
  - allows to express interleaving
  - we concentrate on the small steps of computations

Outline - Structural Operational Semantics of Language **While** and some Extensions

Conclusion / Summary: Structural Operational Semantics of Language **While** and some Extensions

Natural operational semantics (NOS):

- bird-eye view of computations
- does not distinguish between blocking and non-termination,
- non-determinism "hides" non-termination,
- does not allow to express an interleaving semantics.

Structural operational semantics (SOS):

- step-by-step view of execution (sequential composition, evaluation of conditions)
- distinguishes between blocking and non-termination,
- non-determinism does not "hide" non-termination,
- allows to express an interleaving semantics.

NOS and SOS are equivalent for **While**.