Programming Language Semantics and Compiler Design

(Sémantique des Langages de Programmation et Compilation)

Structural Operational Semantics of **While** and of extensions

Frédéric Lang & Laurent Mounier
firstname.lastname@univ-grenoble-alpes.fr
Univ. Grenoble Alpes, Inria,
Laboratoire d'Informatique de Grenoble & Verimag

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

Univ. Grenoble Alpes - UFR IM²AG
www.univ-grenoble-alpes.fr — im2ag.univ-grenoble-alpes.fr

Outline - Structural Operational Semantics of **While** and of extensions

Outline - Structural Operational Semantics of **While** and of extensions

Structural Operational Semantics (SOS)

Comparing NOS and SOS for **While**

Comparing NOS and SOS for extensions of **While**

Conclusion / Summary

Structural Operational Semantics (SOS): intuition

SOS is also known as "small-step semantics".

Emphasis on *individual steps* of the execution:

- In NOS: $(S, \sigma) \to \sigma'$ (big step)
- In SOS: $(S, \sigma) = (S_0, \sigma_0) \Rightarrow \ldots \Rightarrow (S_n, \sigma_n) \Rightarrow \sigma'$ (small steps).

SOS provides a finer definition of computation:

- necessary to define some language extensions (e.g., parallelism)
- useful to prove some properties (e.g., state invariants)
- better/more intuitive account of non-termination: infinite execution sequence rather than infinite derivation tree

Transition relation in SOS

- Written $\Rightarrow$ to be distinguished from NOS
- Transitions of the form $(S, \sigma) \Rightarrow \gamma$
- The result γ of an execution step can be either:
  - (S', σ'): the execution is *not completed*, or
  - σ': the execution *has terminated successfully*.

Transition system: NOS vs. SOS

Transition system – general definition (reminder)

A transition system is a 3-tuple $(\Gamma, T, \to)$, where:

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

Transition system for NOS (reminder)

1. $\Gamma = (\mathbf{Stm} \times \mathbf{State}) \cup \mathbf{State}$
2. $T = \mathbf{State}$
3. $\to\ \subseteq (\mathbf{Stm} \times \mathbf{State}) \times \mathbf{State}$
4. executions (one step) defined by derivation trees

Transition system for SOS

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

Structural Operational Semantics: Inference System

Goal: Define a single execution step.

We define $\Rightarrow$ by *induction* on **Stm**.

Axioms

Same as NOS.

$$\frac{}{(\mathsf{skip}, \sigma) \Rightarrow \sigma}\ [\mathtt{skip_{sos}}]$$

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

Example 1 (Application of the axioms)

- $(\mathsf{skip}, [x \mapsto 42]) \Rightarrow [x \mapsto 42]$ because $\frac{}{(\mathsf{skip}, [x \mapsto 42]) \Rightarrow [x \mapsto 42]}\ [\mathtt{skip_{sos}}]$
- $(y := 42 + x, [x \mapsto 1]) \Rightarrow [x \mapsto 1, y \mapsto 43]$ because

$$\frac{}{(y := 42 + x, [x \mapsto 1]) \Rightarrow [x \mapsto 1, y \mapsto \mathcal{A}[42 + x]([x \mapsto 1])]}\ [\mathtt{ass_{sos}}]$$

and $\mathcal{A}[42 + x]([x \mapsto 1]) = 43$.

Structural Operational Semantics: Inference System

Rules for sequential composition

Differ from NOS: 2 rules with 1 premise each instead of 1 rule with 2 premises.

$$\frac{(S_1,\sigma) \Rightarrow \sigma'}{(S_1;S_2,\sigma) \Rightarrow (S_2,\sigma')}\ [\mathtt{comp}^1_{\mathtt{sos}}] \qquad \frac{(S_1,\sigma) \Rightarrow (S_1',\sigma')}{(S_1;S_2,\sigma) \Rightarrow (S_1';S_2,\sigma')}\ [\mathtt{comp}^2_{\mathtt{sos}}]$$

"execution of S_1 *has* terminated" "execution of S_1 *has not* terminated"

Example 2 (Application of the rules for sequential composition)

$$((\text{skip}; x := 42); y := x+1, []) \overset{(1)}{\Rightarrow} (x := 42; y := x+1, [])$$
$$\overset{(2)}{\Rightarrow} (y := x+1, [x \mapsto 42]) \overset{(3)}{\Rightarrow} [x \mapsto 42, y \mapsto 43]$$

- First derivation ($\overset{(1)}{\Rightarrow}$):

$$\frac{\dfrac{\dfrac{}{(\text{skip}, []) \Rightarrow []}\ [\mathtt{skip_{sos}}]}{(\text{skip}; x := 42, []) \Rightarrow (x := 42, [])}\ [\mathtt{comp}^1_{\mathtt{sos}}]}{((\text{skip}; x := 42); y := x+1, []) \Rightarrow (x := 42; y := x+1, [])}\ [\mathtt{comp}^2_{\mathtt{sos}}]$$

- Second derivation ($\overset{(2)}{\Rightarrow}$):

$$\frac{\dfrac{}{(x := 42, []) \Rightarrow [x \mapsto 42]}\ [\mathtt{ass_{sos}}]}{(x := 42; y := x+1, []) \Rightarrow (y := x+1, [x \mapsto 42])}\ [\mathtt{comp}^1_{\mathtt{sos}}]$$

- Third derivation ($\overset{(3)}{\Rightarrow}$): $\dfrac{}{(y := x+1, [x \mapsto 42]) \Rightarrow [x \mapsto 42, y \mapsto 43]}\ [\mathtt{ass_{sos}}]$

Structural Operational Semantics: Inference System (continued)

Rules for conditional statements

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

$$\frac{}{(\text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi}, \sigma) \Rightarrow (S_1, \sigma)}\ [\mathtt{if}^{\mathbf{tt}}_{\mathtt{sos}}]$$

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

$$\frac{}{(\text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi}, \sigma) \Rightarrow (S_2, \sigma)}\ [\mathtt{if}^{\mathbf{ff}}_{\mathtt{sos}}]$$

Example 3 (Application of the rules for conditional statements)

$(\text{if } x > 0 \text{ then skip else } x := 42 \text{ fi}, [x \mapsto 0]) \Rightarrow (x := 42, [x \mapsto 0])$

because

$$\frac{}{(\text{if } x > 0 \text{ then skip else } x := 42 \text{ fi}, [x \mapsto 0]) \Rightarrow (x := 42, [x \mapsto 0])}\ [\mathtt{if}^{\mathbf{ff}}_{\mathtt{sos}}]$$

Structural Operational Semantics: Inference system (continued)

Rules for iterative statements (unbounded)

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

$$\frac{}{(\text{while } b \text{ do } S \text{ od}, \sigma) \Rightarrow (S; \text{while } b \text{ do } S \text{ od}, \sigma)} \; [\texttt{while}^{\mathbf{tt}}_{\texttt{sos}}]$$

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

$$\frac{}{(\text{while } b \text{ do } S \text{ od}, \sigma) \Rightarrow \sigma} \; [\texttt{while}^{\mathbf{ff}}_{\texttt{sos}}]$$

Exercise 1 (Application of the rule for iterative statements)

Give an example of derivation obtained using the above rule.

Derivation trees, derivation sequences, and execution

Remark Only sequential composition requires premises in the SOS of **While**. As sequences of statements are always finite, all derivation trees are finite. □

Derivation sequences can be either finite or infinite.

Definition 1 (Finite derivation sequence)

$$\gamma_1 \Rightarrow \ldots \Rightarrow \gamma_k \text{ for some } k > 0, \text{ where } \gamma_k \not\Rightarrow$$

(Read $\gamma_k \not\Rightarrow$ as: there is no configuration γ with $\gamma_k \Rightarrow \gamma$)
If $\gamma_k = (S, \sigma)$ is not a final configuration (e.g., avariable used by S is not defined in σ), it is called a blocking configuration.

Definition 2 (Infinite derivation sequence)

$$\gamma_1 \Rightarrow \gamma_2 \Rightarrow \ldots \text{ where for all } k > 0, \gamma_k \Rightarrow \gamma_{k+1}$$

Definition 3 (Execution of a statement)

The execution(s) of a statement S on a state σ is/are the maximal derivation sequence(s) starting with the initial configuration (S, σ).

Exercise 2 (Derivation sequences)

Give examples of finite and infinite derivation sequences.

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

Definition 4 (The $\mathcal{S}_{\mathrm{sos}}$ semantic function)

$$\mathcal{S}_{\mathrm{sos}}[S](\sigma) = \sigma' \text{ iff } (S, \sigma) \Rightarrow^* \sigma'$$

Example 4 (Applying function $\mathcal{S}_{\mathrm{sos}}$)

- $\mathcal{S}_{\mathrm{sos}}[\mathrm{skip}](\sigma) = \sigma$, for any $\sigma \in$ **State**
- $\mathcal{S}_{\mathrm{sos}}[x := 42 + y]([y \mapsto 2]) = [x \mapsto 44, y \mapsto 2]$
- $\mathcal{S}_{\mathrm{sos}}[\text{if } x + y > 0 \text{ then } x := 42 \text{ else } y := 42 \text{ fi}]([x \mapsto 1, y \mapsto 2]) = [x \mapsto 42, y \mapsto 2]$

Exercise 3 (Applying function $\mathcal{S}_{\mathrm{sos}}$)

Apply function $\mathcal{S}_{\mathrm{sos}}$ to some other statements of your choice in **While**.

Outline - Structural Operational Semantics of **While** and of extensions

Program divergence

How do the two operational semantics model *program divergence* (i.e., infinite execution)?

Definition 5 (Program divergence)

Consider a statement S and a state σ:

- Natural semantics:
  S *diverges* in σ iff the procedure to build a derivation tree does not terminate.
  This implies that (S, σ) *does not have a successor (configuration)*:

  $$(S, \sigma) \not\rightarrow, \text{ i.e., } \nexists \sigma' \in \textbf{State} : (S, \sigma) \rightarrow \sigma'$$

  but there may be other causes than divergence (e.g., use of a non-initialized variable).
- Structural semantics:

  S *diverges* in σ,
  iff *there exists an infinite derivation sequence* starting from (S, σ).

  (equivalently all configurations have a successor configuration)

Semantic equivalence

Consider two statements S_1 and S_2.

Semantic equivalence in natural semantics

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

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

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

Equivalence between NOS and SOS?

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

(that is, for all $S \in$ **Stm**, for all $\sigma \in$ **State**: $\mathcal{S}_{\text{ns}}[S](\sigma) = \mathcal{S}_{\text{sos}}[S](\sigma)$)

Lemma 1 (NOS "simulates" SOS)

For every statement S in **Stm**, *states σ and σ' in* **State**, *k in $\mathbb{N} \setminus \{0\}$:*

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

Lemma 2 (SOS "simulates" NOS)

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

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

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

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

Semantic styles and associated proof patterns (reminder)

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

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

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

Lemma: SOS "simulates" NOS

Lemma 3 (Composing statements)

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

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

(Executing a statement is not influenced by the sequentially composed statement – S_2 in the lemma)

Proof.

By induction on $k \in \mathbb{N}$ (left as an exercise). □

Lemma: SOS "simulates" NOS

Proof of Lemma 2: SOS "simulates" NOS.

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

$[\mathrm{ass_{nos}}]$ The conclusion of this rule has the form $(x := a, \sigma) \rightarrow \sigma[x \mapsto \mathcal{A}[a](\sigma)]$, i.e., S has the form "$x := a$" and $\sigma' = \sigma[x \mapsto \mathcal{A}[a](\sigma)]$.
According to $[\mathrm{ass_{sos}}]$, we also have $(x := a, \sigma) \Rightarrow \sigma[x \mapsto \mathcal{A}[a](\sigma)]$.

$[\mathrm{skip_{nos}}]$ S has the form "skip" and $\sigma' = \sigma$.
According to $[\mathrm{skip_{sos}}]$, we also have $(\mathrm{skip}, \sigma) \Rightarrow \sigma$.

$[\mathrm{comp_{nos}}]$ S has form "$S_1; S_2$" and σ' is obtained as follows:

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

By the induction hypothesis applied to both premisses $(S_1, \sigma) \rightarrow \sigma''$ and $(S_2, \sigma'') \rightarrow \sigma'$, we obtain $(S_1, \sigma) \Rightarrow^* \sigma''$ and $(S_2, \sigma'') \Rightarrow^* \sigma'$, respectively.
Since $(S_1, \sigma) \Rightarrow^* \sigma''$, we have $(S_1; S_2, \sigma) \Rightarrow^* (S_2, \sigma'')$ by the lemma (composing statements).
Then, since $(S_2, \sigma'') \Rightarrow^* \sigma'$, we have $(S_1; S_2, \sigma) \Rightarrow^* \sigma'$ by transitivity of the relation $\Rightarrow^*$.

□

Lemma: SOS "simulates" NOS

Proof of SOS "simulates" NOS (continued).

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

[$\mathtt{if}^{\mathtt{tt}}_{\mathtt{nos}}$] S has the form "if b then S_1 else S_2 fi" and σ' is obtained as follows:

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

Since, $\mathcal{B}[b](\sigma) = \mathbf{tt}$, rule [$\mathtt{if}^{\mathtt{tt}}_{\mathtt{sos}}$] implies (if b then S_1 else S_2 fi$, \sigma) \Rightarrow (S_1, \sigma)$. By the induction hypothesis applied to to the premise $(S_1, \sigma) \rightarrow \sigma'$, we obtain $(S_1, \sigma) \Rightarrow^* \sigma'$.
From $(S, \sigma) \Rightarrow (S_1, \sigma)$ and $(S_1, \sigma) \Rightarrow^* \sigma'$, we obtain $(S, \sigma) \Rightarrow^* \sigma'$

[$\mathtt{if}^{\mathtt{ff}}_{\mathtt{nos}}$] Analogous to [$\mathtt{if}^{\mathtt{tt}}_{\mathtt{nos}}$].

□

Lemma: SOS "simulates" NOS

Proof of SOS "simulates" NOS (continued).

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

[$\mathtt{while}^{\mathtt{tt}}_{\mathtt{nos}}$] S has the form "while b do S' od" and σ' is obtained as follows:

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

By the induction hypothesis applied to both premises $(S', \sigma) \rightarrow \sigma''$ and (while b do S' od$, \sigma'') \rightarrow \sigma'$, we obtain respectively

$$(S', \sigma) \Rightarrow^* \sigma'' \qquad (1)$$

$$(\text{while } b \text{ do } S' \text{ od}, \sigma'') \Rightarrow^* \sigma' \qquad (2)$$

From (1) and the lemma (composing statements), we obtain

$$(S'; \text{while } b \text{ do } S' \text{ od}, \sigma) \Rightarrow^* (\text{while } b \text{ do } S' \text{ od}, \sigma'') \qquad (3)$$

Then since $\mathcal{B}[b](\sigma) = \mathbf{tt}$, we have the following derivation:

$$\begin{array}{llll} (\text{while } b \text{ do } S' \text{ od}, \sigma) & \Rightarrow & (S'; \text{while } b \text{ do } S' \text{ od}, \sigma) & \text{by rule } [\mathtt{while}^{\mathtt{tt}}_{\mathtt{sos}}] \\ & \Rightarrow^* & (\text{while } b \text{ do } S' \text{ od}, \sigma'') & \text{from (3)} \\ & \Rightarrow^* & \sigma' & \text{from (2)} \end{array}$$

[$\mathtt{while}^{\mathtt{ff}}_{\mathtt{nos}}$] Straightforward.

□

Lemma: NOS "simulates" SOS

We need an additional intermediate lemma.

Lemma 4 (Decomposing computations in SOS)

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

$$(S_1; S_2, \sigma) \Rightarrow^k \sigma'' \text{ implies that there exist } \sigma' \text{ and } k_1, k_2 \in \mathbb{N} \setminus \{0\} \text{ such that } k = k_1 + k_2 \text{ and } (S_1, \sigma) \Rightarrow^{k_1} \sigma' \text{ and } (S_2, \sigma') \Rightarrow^{k_2} \sigma''.$$

Proof.

By induction on $k \in \mathbb{N}$ in $(S_1; S_2, \sigma) \Rightarrow^k \sigma''$ (left as an exercise). □

Lemma 5 (NOS "simulates" SOS)

*For every statement $S \in$ **Stm**, state $\sigma \in$ **State** and $\sigma' \in$ **State**, and $k \in \mathbb{N} \setminus \{0\}$:*

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

Proof.

By induction on $k \in \mathbb{N} \setminus \{0\}$ in $(S, \sigma) \Rightarrow^k \sigma'$, i.e., the length of the derivation sequence. □

Lemma: NOS "simulates" SOS

NOS "simulates" SOS.

Let us assume that there exists a number k such that for every derivation $(S, \sigma) \Rightarrow^i \sigma'$ of length $i \leq k$, then $(S, \sigma) \rightarrow^* \sigma'$. Note that this holds for $k = 0$, since there is no derivation of length 0. Now let us consider derivations of length $k + 1$ and show that if $(S, \sigma) \Rightarrow^{k+1} \sigma'$, then $(S, \sigma) \rightarrow^* \sigma'$. We proceed by case on the rule used to derive the first step of $(S, \sigma) \Rightarrow^{k+1} \sigma'$.

[ass_{sos}] Straightforward. In this case, $k = 0$ since the derivation has only one step.

[skip_{sos}] Straightforward. In this case, $k = 0$ (same as above).

[$\text{comp}^{x}_{\text{sos}}$] ($x = 1, 2$). S has the form "$S_1; S_2$" and we assume that $(S_1; S_2, \sigma) \Rightarrow^{k+1} \sigma''$. We can apply the lemma (decomposing computations) to get that there exist $\sigma' \in$ **State** and $k_1, k_2 \in \mathbb{N} \setminus \{0\}$ s.t.

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

where $k_1 + k_2 = k + 1$, i.e., $k_1 \leq k$ and $k_2 \leq k$. Thus, the induction hypothesis can be applied to both of these derivation sequences, giving:

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

Using [comp_{nos}], we get $(S_1; S_2, \sigma) \rightarrow \sigma''$.

□

Lemma: NOS "simulates" SOS

NOS "simulates" SOS.

[if^{tt}_{sos}] We have $\mathcal{B}[b](\sigma) = \mathbf{tt}$ and (if b then S_1 else S_2 fi$, \sigma) \Rightarrow (S_1, \sigma) \Rightarrow^k \sigma'$. The induction hypothesis can be applied to $(S_1, \sigma) \Rightarrow^k \sigma'$, giving $(S_1, \sigma) \rightarrow \sigma'$, which is the premise of rule [if^{tt}_{nos}]. Therefore, its conclusion (if b then S_1 else S_2 fi$, \sigma) \rightarrow \sigma'$ holds as expected.

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

[while^{tt}_{sos}] We have $\mathcal{B}[b](\sigma) = \mathbf{tt}$ and

$$(\text{while } b \text{ do } S \text{ od}, \sigma) \Rightarrow (S; \text{while } b \text{ do } S \text{ od}, \sigma) \Rightarrow^k \sigma''$$

From $(S; \text{while } b \text{ do } S \text{ od}, \sigma) \Rightarrow^k \sigma''$ and the lemma (decomposing computations), we get that there exists σ', $k_1 > 0$ and $k_2 > 0$ such that $k_1 + k_2 = k$, $(S, \sigma) \Rightarrow^{k_1} \sigma'$, and $(\text{while } b \text{ do } S \text{ od}, \sigma') \Rightarrow^{k_2} \sigma''$. Therefore, $k_1 < k$ and $k_2 < k$ and the induction hypothesis can be applied to those two last derivations, giving

$$(S, \sigma) \rightarrow \sigma' \qquad \text{and} \qquad (\text{while } b \text{ do } S \text{ od}, \sigma') \rightarrow \sigma''$$

which are the premises of rule [while^{tt}_{nos}]. Therefore, its conclusion $(\text{while } b \text{ do } S \text{ od}, \sigma) \rightarrow \sigma''$ holds as expected.

[while^{ff}_{sos}] Immediate.

□

Outline - Structural Operational Semantics of **While** and of extensions

Structural Operational Semantics (SOS)

Comparing NOS and SOS for **While**

Comparing NOS and SOS for extensions of **While**
- Statement **abort** for abnormal termination
- Statement **or** for Non-Determinism
- Statement $\|$ for parallel composition

Conclusion / Summary

Outline - Structural Operational Semantics of **While** and of extensions

Extending **While** with abnormal termination

Definition 6 (**abort**)

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

$$\text{for all } \sigma \in \textbf{State} : (\text{abort}, \sigma) \not\rightarrow \quad \text{(in NOS)}$$

and

$$\text{for all } \sigma \in \textbf{State} : (\text{abort}, \sigma) \not\Rightarrow \quad \text{(in SOS)}$$

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

Example 5 (Program with possible abnormal termination)

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

Examples with abort

Exercise 4 (Assertions)

Provide natural and structural operational semantic rules to the following construct:

assert b

The informal semantics is that one should check that b holds before executing the subsequent statements. If b does not hold, the program should stop.

$$\frac{}{(\text{assert b}, \sigma) \rightarrow \sigma}\ \mathcal{B}[b](\sigma) = \mathbf{tt}$$

$$\frac{}{(\text{assert b}, \sigma) \Rightarrow \sigma}\ \mathcal{B}[b](\sigma) = \mathbf{tt}$$

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

In *natural* operational semantics:

- abort and
- while true do skip od

are semantically equivalent.

In *structural* operational semantics:

- abort and
- while true do skip od

are not semantically equivalent.

Outline - Structural Operational Semantics of **While** and of extensions

Statement **or**

Definition 7 (Or statement)

$$S ::= S_1 \text{ or } S_2 \mid \ldots$$

Execute S_1 or S_2 nondeterministically

Example 6 (Using the or statement)

We expect that the execution of the statement

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

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

The **or** statement: extending the transition system

Definition 8 (NOS and SOS transition systems for statement **or**)

- Natural semantics:

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

- Structural semantics:

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

Remark Adding or makes it now impossible to define $\mathcal{S}_{\rm ns}$ and $\mathcal{S}_{\rm sos}$ as functions since the language becomes nondeterministic. □

Example 7 (Applying the rules of statement **or**)

Consider the statement $x := 1$ or $x := 2$.

- Natural semantics:

$$\frac{\overline{(x := 1, [\,]) \to [x \mapsto 1]}}{(x := 1 \textbf{ or } x := 2, [\,]) \to [x \mapsto 1]} \quad \frac{\overline{(x := 2, [\,]) \to [x \mapsto 2]}}{(x := 1 \text{ or } x := 2, [\,]) \to [x \mapsto 2]}$$

- Structural semantics:

$$(x := 1 \textbf{ or } x := 2, []) \Rightarrow (x := 1, []) \Rightarrow [x \mapsto 1]$$
$$(x := 1 \textbf{ or } x := 2, []) \Rightarrow (x := 2, []) \Rightarrow [x \mapsto 2]$$

Discussion on **or** and non-termination

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

Example 8 (**or** and non-termination in NOS and SOS)

Consider the two following statements:

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

Comparing semantics:

- natural semantics enables only one derivation: $(S_1 \text{ or } S_2, \sigma) \to \sigma$
- structural semantics enables two derivations:
  - a finite one $(S_1 \text{ or } S_2, \sigma) \Rightarrow (S_2, \sigma) \Rightarrow \sigma$ and
  - an infinite one $(S_1 \text{ or } S_2, \sigma) \Rightarrow (S_1, \sigma) \Rightarrow (S_1, \sigma) \Rightarrow \ldots$.

Henceforth:

- in NOS: "while true do skip od or skip" is semantically equivalent to skip.
- in SOS: "while true do skip od or skip" is not equivalent to skip;

Outline - Structural Operational Semantics of **While** and of extensions

Structural Operational Semantics (SOS)

Comparing NOS and SOS for **While**

Comparing NOS and SOS for extensions of **While**

- Statement **abort** for abnormal termination
- Statement **or** for Non-Determinism
- Statement $\|$ for parallel composition

Conclusion / Summary

Parallel composition

$$S ::= S_1 \parallel S_2 \mid \ldots$$

In $S_1 \parallel S_2$, we expect S_1 and S_2 to execute in parallel, i.e., both S_1 and S_2 will execute, not caring about the order.

Definition 9 (Attemps to define parallel composition semantics)

- Natural semantics:

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

$\to$ The executions of S_1 and S_2 are *atomic*.
$\to$ "$S_1 \parallel S_2$" is semantically equivalent to "$(S_1; S_2)$ or $(S_2; S_1)$".

- Structural semantics:

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

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

$\to$ The executions of S_1 and S_2 are *interleaved*.

Discussion about the parallelism and interleaving

Example 9 (Parallel execution)

Consider the following statement:

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

- natural operational semantics: 2 possible ending states
  1. $(x := 1 \parallel (x := 2; x := x + 2), [\,]) \to [x \mapsto 4]$ (left before right)
  2. $(x := 1 \parallel (x := 2; x := x + 2), [\,]) \to [x \mapsto 1]$ (right before left)
- structural operational semantics: 3 possible ending states
  1. $(x := 1 \parallel (x := 2; x := x + 2), [\,]) \Rightarrow (x := 2; x := x + 2, [x \mapsto 1])$
     $\Rightarrow (x := x + 2, [x \mapsto 2]) \Rightarrow [x \mapsto 4]$
  2. $(x := 1 \parallel (x := 2; x := x + 2), [\,]) \Rightarrow (x := 1 \parallel x := x + 2, |x \mapsto 2])$
     $\Rightarrow (x := x + 2, [x \mapsto 1]) \Rightarrow [x \mapsto 3]$
  3. $(x := 1 \parallel (x := 2; x := x + 2), [\,]) \Rightarrow (x := 1 \parallel x := x + 2, |x \mapsto 2])$
     $\Rightarrow (x := 1, [x \mapsto 4]) \Rightarrow [x \mapsto 1]$

Discussion about the parallelism and interleaving (continued)

Natural vs Structural (operational) semantics and interleaving

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

Remark There exist languages dedicated to the description of parallelism, whose semantics elegantly combine big steps (for atomic sequences of statements) and small steps (for non-atomic sequences of statements). □

Outline - Structural Operational Semantics of **While** and of extensions

Conclusion / Summary: Structural Operational Semantics of **While** and of extensions

Natural operational semantics (NOS):

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

Structural operational semantics (SOS):

- step-by-step view of computations
- distinguishes between blocking and non-termination
- non-determinism does not "hide" non-termination
- allows to express an interleaving semantics

As we have shown, NOS and SOS are equivalent for **While**.
They are not for the studied extensions.