# 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 2019 - 2020

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


Structural Operational Semantics (SOS)

Comparing the Natural and Structural Operational Semantics (NOS vs SOS) of **While**

Comparing NOS and SOS on some Extensions of Language **While**

Conclusion / Summary

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

Structural Operational Semantics (SOS)

Comparing the Natural and Structural Operational Semantics (NOS vs SOS) of **While**

Comparing NOS and SOS on some Extensions of Language **While**

Conclusion / Summary

# Structural Operational Semantics: intuition

Structural Operational Semantics (SOS) is aka "small-step semantics".

Emphasis on *individual steps* of the execution:

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

$\hookrightarrow$ consequences on the definition of computation associated with the other programming constructs.

## Intuition on SOS

- Transitions are of the form:

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

- $\Rightarrow$ is the transition relation between configurations.
- The result $\gamma$ of an execution step can be either:
  - $(S', \sigma')$: the execution is *not completed*, or
  - $\sigma'$: the execution *has terminated*.

# Transition system: natural vs structural semantics

An operational semantics is defined by a transition system.

## Transition system – general definition (reminder)

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

- $\Gamma$ is the set of configurations
- $T \subseteq \Gamma$ is the set of final configurations
- $\rightarrow \subseteq \Gamma \times \Gamma$ is the transition relation (between configurations)

## Transition system for natural operational semantics

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

*Non-final configurations are related (only) to final ones.*

## Transition system for structural operational semantics

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. $\Rightarrow$ defined by derivation sequences

*Non-final configurations are related to non-final and final ones.*

# Structural Operational Semantics: Inference System

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

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

## Axioms

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

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

## Example 1 (Application of axioms)

- $(\mathsf{skip}, [x \mapsto 42]) \Rightarrow [x \mapsto 42]$ because $\frac{}{(\mathsf{skip}, [x \mapsto 42]) \Rightarrow [x \mapsto 42]} \; [\mathtt{skip}_{\mathtt{sos}}]$
- $(y := 42 + x, [x \mapsto 1]) \Rightarrow \sigma[y \mapsto \mathcal{A}[42 + x][x \mapsto 1]]$

  because

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

# Structural Operational Semantics: Inference System

## Rules for sequential statements

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

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

## Example 2 (Application of the rules for sequential statements)

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

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

$$\cfrac{\cfrac{\overline{(\mathsf{skip}, [\,]) \Rightarrow [\,]} \; [\mathtt{skip}_{\mathtt{sos}}]}{(\mathsf{skip}; x := 42, [\,]) \Rightarrow (x := 42, [\,])} \; [\mathtt{comp}^1_{\mathtt{sos}}]}{((\mathsf{skip}; x := 42); y := x+1, [\,]) \Rightarrow (x := 42; y := x+1, [\,])} \; [\mathtt{comp}^2_{\mathtt{sos}}]$$

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

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

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

# Structural Operational Semantics: Inference System (ctd)

## 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

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

## Rule for iterative statements (unbounded)

$$\frac{}{(\text{while } b \text{ do } S \text{ od}, \sigma) \Rightarrow (\text{if } b \text{ then } (S; \text{while } b \text{ do } S \text{ od}) \text{ else skip fi}, \sigma)} \; [\mathtt{while}_{\mathtt{sos}}]$$

**Exercise 1 (Application of the rule for iterative statements)**

Give an example of derivation obtained using the above rule.

# Derivation Sequence and Execution

Using the axioms and rules, one can obtain two sorts of derivation sequences.

## Definition 1 (Finite derivation sequences)

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

where:

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

## Definition 2 (Infinite derivation sequences)

$$\gamma_1, \gamma_2, \ldots, \text{where } \gamma_i \Rightarrow \gamma_{i+1}, \text{ for } i \geq 1$$

## Definition 3 (Execution of a statement)

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

## 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)

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

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

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

## 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 Language **While** and some Extensions

Structural Operational Semantics (SOS)

Comparing the Natural and Structural Operational Semantics (NOS vs SOS) of **While**

Comparing NOS and SOS on some Extensions of Language **While**

Conclusion / Summary

# Program divergence

How do the two operational semantics model *program divergence*?

## Definition 5 (Program divergence)

Consider a statement $S$ and a state $\sigma$:

- Natural semantics:
  $S$ *diverges* in $\sigma$, if $(S, \sigma)$ *does not have a successor (configuration)*:

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

  (equivalently there exists an infinite derivation tree)

- Structural semantics:

  $S$ *diverges* in $\sigma$,
  if *there exists an infinite derivation sequence* starting from $(S, \sigma)$.

  (equivalently all configurations have a successor configuration)

**Remark** As was the case with NOS, there is always at least one derivation of a configuration. In other words, by examining the derivation rules, it is not possible to obtain a stuck 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 $\sigma$ and $\sigma'$:

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

## Semantic equivalence in structural semantics

$S_1$ and $S_2$ are semantically equivalent, if for all states $\sigma$

- for any final or blocking configuration $\gamma$:

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

- there exists an infinite derivation sequence starting from $(S_1, \sigma)$
  iff
  there exists an infinite derivation sequence starting from $(S_2, \sigma)$.

# Equivalence between NOS and SOS?

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

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

## Lemma 1 (NOS "simulates" SOS)

*For every statement $S$ in* **Stm**, *states $\sigma$ and $\sigma'$ in* **State**:

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

## Lemma 2 (SOS "simulates" NOS)

*For every statement $S$ in* **Stm**, *states $\sigma$ and $\sigma'$ 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**: $\mathcal{S}_{\mathrm{ns}}[S] = \mathcal{S}_{\mathrm{sos}}[S]$.

# 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.

# Intermediate Lemma: SOS "simulates" NOS

## Lemma 3 (Composing statements)

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

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

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

## Proof.

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

**Remark** The converse does not hold in general (see the exercises). □

## Lemma 4 (SOS "simulates" NOS)

*For every statement $S \in$ **Stm**, states $\sigma, \sigma' \in$ **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 $(S, \sigma) \to \sigma'$.

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

[skip$_{\text{nos}}$] Analogous to the previous case.

[comp$_{\text{nos}}$] $S$ is necessarily of the form $S_1; S_2$ and $\sigma'$ is obtained as follows:

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

(for some state $\sigma''$)
We apply the induction hypothesis to the two premisses $(S_1, \sigma) \to \sigma''$ and $(S_2, \sigma'') \to \sigma'$ to obtain $(S_1, \sigma) \Rightarrow^* \sigma''$ and $(S_2, \sigma'') \Rightarrow^* \sigma'$, respectively.
From the lemma (composing statements), we obtain $(S_1; S_2, \sigma) \Rightarrow^* (S_2, \sigma'')$. And, using $(S_2, \sigma'') \Rightarrow^* \sigma'$ again, we find $(S_1; S_2, \sigma) \Rightarrow^* \sigma'$.

□

# Intermediate Lemma: SOS "simulates" NOS

## Proof of SOS "simulates" NOS (ctd).

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

$[\texttt{if}^{\texttt{tt}}_{\texttt{nos}}]$ $S$ is necessarily of the form if $b$ then $S_1$ else $S_2$ fi, for some $S_1, S_2 \in \textbf{Stm}$ and $\sigma'$ has been obtained as described by the rule:

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

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

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

□

# Intermediate Lemma: SOS "simulates" NOS

## Proof of SOS "simulates" NOS (ctd).

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

$[\texttt{while}^{\texttt{tt}}_{\texttt{nos}}]$ 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:

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

$[\texttt{while}^{\texttt{tt}}_{\texttt{nos}}]$ Straightforward.

□

# Intermediate Lemma: NOS "simulates" SOS

We need an additional intermediate lemma.

## Lemma 5 (Decomposing computations in SOS)

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

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

## Proof.

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

## Lemma 6 (NOS "simulates" SOS)

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

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

## 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.

[ass$_{\text{sos}}$] Straightforward ($(S, \sigma) \Rightarrow^1 \sigma[x \mapsto \mathcal{A}[a]\sigma]$ and $k = 0$).

[skip$_{\text{sos}}$] Straightforward $(S, \sigma) \Rightarrow^1 \sigma$, $\sigma = \sigma'$ and $k = 0$).

[comp$^{\text{x}}_{\text{sos}}$] Cases [comp$^1_{\text{sos}}$] and [comp$^2_{\text{sos}}$]. 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 $\sigma' \in$ **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

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

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

□

# 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 $[\mathtt{if}^{\mathtt{tt}}_{\mathtt{nos}}]$.

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

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

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

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

Structural Operational Semantics (SOS)

Comparing the Natural and Structural Operational Semantics (NOS vs SOS) of **While**

Comparing NOS and SOS on some Extensions of Language **While**
- Command **abort** for abnormal termination
- Operator **or** for Non-Determinism
- The $\|$ Operator for Parallelism

Conclusion / Summary

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


Structural Operational Semantics (SOS)

Comparing the Natural and Structural Operational Semantics (NOS vs SOS) of **While**

Comparing NOS and SOS on some Extensions of Language **While**

- Command **abort** for abnormal termination
- Operator **or** for Non-Determinism
- The $\|$ Operator for Parallelism

Conclusion / Summary

# Extending **While** with abnormal termination

## Definition 6 (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 $(\text{abort}, \sigma)$ has no successors (blocking):

$$\begin{array}{ll} \text{for all } \sigma \in \textbf{State} : (\text{abort}, \sigma) \nrightarrow & (\text{in NOS}) \\ \text{ and} & \\ \text{for all } \sigma \in \textbf{State} : (\text{abort}, \sigma) \nRightarrow & (\text{in SOS}) \end{array}$$

$\hookrightarrow$ 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)

Using the abort construct, provide natural and structural operational semantics rules to the following construct:

$$\text{assert b before S}$$

The informal semantics is that one should check that `b` holds before executing `S`. If `b` does not hold, `S` should not be executed.

# 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

Structural Operational Semantics (SOS)

Comparing the Natural and Structural Operational Semantics (NOS vs SOS) of **While**

Comparing NOS and SOS on some Extensions of Language **While**

- Command **abort** for abnormal termination
- Operator **or** for Non-Determinism
- The $\|$ Operator for Parallelism

Conclusion / Summary

# Operator **or**

### Definition 7 (Or operator)

- Non-determinism
- Statement $S_1 \text{or}\, S_2$

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

### Example 6 (Using the or operator)

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** operator: extending the transition system

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

- Configurations: $(\textbf{While}^{\mathrm{or}} \times \textbf{State}) \cup \textbf{State}$
- Natural semantics:

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

- Structural semantics:

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

## Example 7 (Applying the rules of operator **or**)

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

- Natural semantics:

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

- Structural semantics:

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

# Discussion on **or** and non-termination

With natural operational semantics, operator 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
- $S_2 =$ while false do skip od

Comparing semantics:

- natural semantics: $(S_1 \text{ or } S_2, \sigma)$ has one derivation tree (the one corresponding to $S_2$);
- structural semantics: $(S_1 \text{ or } S_2)$ has an infinite derivation sequence for $S_1$ (in addition to the finite one for $S_2$).

Henceforth:

- in NOS: $S_1 \text{ or } S_2$ and $S_2$ *are* semantically equivalent;
- in SOS: $S_1 \text{ or } S_2$, $S_1$, and $S_2$ *are pairwise not* semantically equivalent.

## Natural/structural operational semantics and looping

- **In NOS, non-determinism "hides" looping, if possible.**
- **In SOS, non-determinism does not "hide" looping.**

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


Structural Operational Semantics (SOS)

Comparing the Natural and Structural Operational Semantics (NOS vs SOS) of **While**

Comparing NOS and SOS on some Extensions of Language **While**

- Command **abort** for abnormal termination
- Operator **or** for Non-Determinism
- The $\|$ Operator for Parallelism

Conclusion / Summary

# Parallel execution

We add a construct noted $\|$ (syntactic extension).

In $S_1 \| S_2$, we expect $S_1$ and $S_2$ to execute in parallel (informal semantics).

We refer to this new language as **While**$^{\|}$.

## Definition 9 (Extending the transition system for parallel execution)

- Configurations: $\left(\mathbf{While}^{\|} \times \mathbf{State}\right) \cup \mathbf{State}$.
- Additional transition rules to handle $\|$:
  - 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 immediate constituents ($S_1$ and $S_2$) are *atomic*.
  - Structural semantics:

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

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

    $\to$ The executions of immediate constituents ($S_1$ 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
- structural operational semantics: 3 possible ending states

## Exercise 5 (Applying the rules for parallel execution)

Apply the new semantic rules to the statement in the previous example to determine the possible resulting states.

## Natural vs Structural (operational) semantics and interleaving

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

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

Structural Operational Semantics (SOS)

Comparing the Natural and Structural Operational Semantics (NOS vs SOS) of **While**

Comparing NOS and SOS on some Extensions of Language **While**

Conclusion / Summary

# 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** and not equivalent for the studied extensions.