Semantics with Applications

3. More on Operational Semantics

Hanne Riis Nielson, Flemming Nielson

(thanks to Henrik Pilegaard)

[SwA] Hanne Riis Nielson, Flemming Nielson
Semantics with Applications: An Appetizer
Springer, 2007
Recap

Natural Semantics

Structural Operational Semantics
Language While: Syntactic Categories

- **Numerals** $n \in \textbf{Num}$

  
  
  
  $n ::= 0 \mid 1 \mid n0 \mid n1$

- **Variables** $x \in \textbf{Var}$ – e.g. $x, y, z, \ldots$

- **Arithmetic expressions** $a \in \textbf{Aexp}$

  
  
  
  $a ::= n \mid x \mid a_1 + a_2 \mid a_1 \times a_2 \mid a_1 - a_2$

- **Boolean expressions** $b \in \textbf{Bexp}$

  
  
  
  $b ::= \text{true} \mid \text{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \land b_2$

- **Statements** $S \in \textbf{Stm}$

  
  
  
  $S ::= x := a \mid \text{skip} \mid S_1; S_2 \mid \text{if } b \text{ then } S_1 \text{ else } S_2 \mid \text{while } b \text{ do } S$
Natural semantics for While

\[ \text{[ass}_{ns}] \quad \langle x := a, s \rangle \rightarrow s[x \rightarrow A[a]]s \]

\[ \text{[skip}_{ns}] \quad \langle \text{skip}, s \rangle \rightarrow s \]

\[ \langle S_1, s \rangle \rightarrow s', \langle S_2, s' \rangle \rightarrow s'' \]

\[ \frac{}{\langle S_1; S_2, s \rangle \rightarrow s''} \]

\[ \langle S_1, s \rangle \rightarrow s' \]

\[ \frac{}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s \rangle \rightarrow s'} \]

if \( B[b]s = \text{tt} \)

\[ \langle S_2, s \rangle \rightarrow s' \]

\[ \frac{}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s \rangle \rightarrow s'} \]

if \( B[b]s = \text{ff} \)

\[ \langle S, s \rangle \rightarrow s', \langle \text{while } b \text{ do } S, s' \rangle \rightarrow s'' \]

\[ \frac{}{\langle \text{while } b \text{ do } S, s \rangle \rightarrow s''} \]

if \( B[b]s = \text{tt} \)

\[ \langle \text{while } b \text{ do } S, s \rangle \rightarrow s \text{ if } B[b]s = \text{ff} \]
The meaning of statements can be summarised as a partial function from `State` to `State`:

\[ S_{ns} : \text{Stmt} \rightarrow (\text{State} \leftrightarrow \text{State}) \]

- Partial function: not necessarily defined for all elements of the domain
- Definition:

\[ S_{ns}[S]s = \begin{cases} 
  s' & \text{if } \langle S, s \rangle \rightarrow s' \\
  \text{undef} & \text{otherwise}
\end{cases} \]

- Need for partiality: non-terminating statements such as `while true do skip`
Induction on the Shape of Derivation Trees

- In the proof of the previous lemma we were inspecting the structure of the derivation tree for certain transitions

- This can be generalized to the following proof technique:
  - Prove that the property holds for all the axioms
  - Prove that the property holds for all other rules:
    - Assume that the property holds for its premises (this is called the induction hypothesis, sometimes abbreviated $IH$)
    - Prove that it holds for the conclusion (provided the side conditions are satisfied)
Structural Operational Semantics for While

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

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

\[ \text{[comp}^1_{\text{sos}}] \quad \langle S_1, s \rangle \Rightarrow \langle S'_1, s' \rangle \]
\[ \frac{}{\langle S_1; S_2, s \rangle \Rightarrow \langle S'_1; S_2, s' \rangle} \]

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

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

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

\[ \text{[while}_{\text{sos}}] \quad \langle \text{while } b \text{ do } S, s \rangle \Rightarrow \langle \text{if } b \text{ then } (S; \text{while } b \text{ do } S) \text{ else skip}, s \rangle \]
A derivation sequence of a statement $S$ starting in state $s$ is either

- a finite sequence $\gamma_0, \gamma_1, \ldots, \gamma_k$

  \[
  \gamma_0 \Rightarrow \gamma_1 \Rightarrow \ldots \Rightarrow \gamma_k
  \]

  and $\gamma_k$ is final (of the form $s$)

- an infinite sequence $\gamma_0, \gamma_1, \gamma_2, \ldots$

  \[
  \gamma_0 \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \ldots
  \]

  where $\gamma_0 = \langle S, s \rangle$ and $\gamma_i \Rightarrow \gamma_{i+1}$ for $0 \leq i (< k)$

We write $\gamma_0 \Rightarrow^i \gamma_i$ to indicate $i$ derivation steps

We write $\gamma_0 \Rightarrow^* \gamma_i$ to indicate finitely many derivation steps
The Semantic Function for Statements

- As we did in the case for Natural Semantics, the meaning of statements can be summarised as a partial function from \textbf{State} to \textbf{State}

\[ S_{sos} : \textbf{Stm} \rightarrow (\textbf{State} \leftrightarrow \textbf{State}) \]

- Definition:

\[ S_{sos}[S]s = \begin{cases} s' & \text{if } \langle S, s \rangle \Rightarrow^* s' \\ \text{undef} & \text{otherwise} \end{cases} \]
Concepts for NS and SOS

- The following concepts can be defined both for Natural Semantics and Structural Operational Semantics
  - Termination
  - Looping
  - Semantic equivalence
  - Determinism

- The formalisations differ however – we will compare them in the following
Termination

- Natural Semantics:

The execution of $S$ from state $s$ terminates if and only if there is a state $s'$ such that $\langle S, s \rangle \rightarrow s'$.

- Structural Operational Semantics:

The execution of $S$ from state $s$ terminates if and only if there is a finite derivation sequence starting with $\langle S, s \rangle$, i.e.

$$\langle S, s \rangle \Rightarrow \gamma_1 \Rightarrow \ldots \Rightarrow \gamma_k$$

and $\gamma_k$ is final.
Natural Semantics:

We say that the execution of $S$ from state $s$ *loops* if and only if there is no state $s'$ such that $\langle S, s \rangle \rightarrow s'$.

Structural Operational Semantics:

We say that the execution of $S$ from state $s$ *loops* if and only if there is an infinite derivation sequence starting with $\langle S, s \rangle$, i.e.

$$\langle S, s \rangle \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \ldots$$
Semantic Equivalence

▶ Natural Semantics:

Two statements $S_1$ and $S_2$ are *semantically equivalent* if for all states $s$ and $s'$

$$\langle S_1, s \rangle \rightarrow s' \text{ if and only if } \langle S_2, s \rangle \rightarrow s'$$

▶ Structural Operational Semantics:

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

- $\langle S_1, s \rangle \Rightarrow^* \gamma$ if and only if $\langle S_2, s \rangle \Rightarrow^* \gamma$, whenever $\gamma$ is final
- there is an infinite derivation sequence starting with $\langle S_1, s \rangle$ if and only if there is one starting in $\langle S_2, s \rangle$
Determinism

- Natural Semantics:

  The semantics is *deterministic* if for all statements $S$ and states $s$, $s'$, and $s''$ we have that

  $$\langle S, s \rangle \rightarrow s' \text{ and } \langle S, s \rangle \rightarrow s'' \text{ imply } s' = s''$$

- Structural Operational Semantics:

  The semantics is *deterministic* if for all $S$ and $s$, $\gamma'$, and $\gamma''$ we have that

  $$\langle S, s \rangle \Rightarrow \gamma' \text{ and } \langle S, s \rangle \Rightarrow \gamma'' \text{ imply } \gamma' = \gamma''$$
Induction on the Length of Derivation Sequences

- For Structural Operational Semantics it is often useful to conduct proofs by the length of derivation sequences.

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

- 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*).
  - Prove that it holds for derivation sequences of length \(k+1\).
Three Proof Principles

- Structural Induction
- Induction on the Shape of Derivation Trees
  - Especially important for Natural Semantics
- Induction on the Length of Derivation Sequences
  - Especially important for Structural Operational Semantics
Non-sequential Language Extensions

Reading material: Section 3.1 of SwA
Abortion

- We extend the syntax of While with the statement `abort`

\[
S ::= x := a \mid \text{skip} \mid S_1; S_2 \mid \text{if } b \text{ then } S_1 \text{ else } S_2 \mid \text{while } b \text{ do } S \mid \text{abort}
\]

- **Intuition:** `abort` stops the execution of the complete program

- We model this by ensuring that configurations \( \langle \text{abort}, s \rangle \) are stuck

- Neither for NS nor for SOS we change the set of transition rules! Then there is no rule to apply when the `abort`-statement is encountered, ensuring that the respective configuration is stuck.
Abortion: Three examples

- abort
  - NS: no derivation tree
  - SOS: finite derivation sequence ⟨abort, s⟩

- skip
  - NS: derivation tree ⟨skip, s⟩ → s
  - SOS: finite derivation sequence ⟨skip, s⟩ ⇒ s

Both NS and SOS can establish that abort and skip are not semantically equivalent.
Abortion: Three examples

- **abort**
  - NS: no derivation tree
  - SOS: finite derivation sequence $\langle \text{abort}, s \rangle$

- **skip**
  - NS: derivation tree $\langle \text{skip}, s \rangle \rightarrow s$
  - SOS: finite derivation sequence $\langle \text{skip}, s \rangle \Rightarrow s$

Both NS and SOS can establish that abort and skip are not semantically equivalent.

- **while true do skip**
  - NS: no derivation tree
  - SOS: infinite derivation sequence
Abortion: Three examples

- **abort**
  - NS: no derivation tree
  - SOS: finite derivation sequence $\langle \text{abort}, s \rangle$

- **skip**
  - NS: derivation tree $\langle \text{skip}, s \rangle \rightarrow s$
  - SOS: finite derivation sequence $\langle \text{skip}, s \rangle \Rightarrow s$

Both NS and SOS can establish that **abort** and **skip** are not semantically equivalent.

- **while true do skip**
  - NS: no derivation tree
  - SOS: infinite derivation sequence

**abort and while true do skip** are semantically equivalent in NS! SOS however can distinguish between abnormal termination and looping.
Natural semantics: we cannot distinguish between looping and abnormal termination

SOS: looping is reflected by infinite derivation sequence and abnormal termination by finite derivation sequence ending in a stuck configuration
Non-determinism

- We extend the syntax of While with the statement `or`

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

- **Intuition:** For \(S_1\) or \(S_2\) we can non-deterministically choose to execute either \(S_1\) or \(S_2\)

- For both NS and SOS, new rules have to be introduced to handle the statement
Rules for non-determinism in Natural Semantics

\[ \text{[or}_{ns}^1 \] \]

\[
\frac{\langle S_1, s \rangle \rightarrow s'}{
\langle S_1 \text{ or } S_2, s \rangle \rightarrow s'
}
\]

\[ \text{[or}_{ns}^2 \] \]

\[
\frac{\langle S_2, s \rangle \rightarrow s'}{
\langle S_1 \text{ or } S_2, s \rangle \rightarrow s'
}
\]
Rules for non-determinism in SOS

\[ \text{[or}^1_{\text{sos}} \]  \quad \langle S_1 \text{ or } S_2, s \rangle \Rightarrow \langle S_1, s \rangle \\
\[ \text{[or}^2_{\text{sos}} \]  \quad \langle S_1 \text{ or } S_2, s \rangle \Rightarrow \langle S_2, s \rangle \]
Non-determinism: Two examples

- $x := 1$ or $(x := 2; x := x + 2)$
  - NS: $\langle x := 1 \text{ or } (x := 2; x := x + 2), s \rangle \rightarrow$

- (while true do skip) or $(x := 2; x := x + 2)$

NS suppresses looping! (Never chooses the “wrong” branch.)

SOS: Looping is not suppressed.
Non-determinism: Two examples

- \( x := 1 \) or \( (x := 2; x := x + 2) \)
  - NS: \( \langle x := 1 \) or \( (x := 2; x := x + 2), s \rangle \rightarrow s[x \mapsto 1] \)

- (while true do skip) or \( (x := 2; x := x + 2) \)
Non-determinism: Two examples

▸ \( x := 1 \) or \( (x := 2; x := x + 2) \)
  ▸ NS: \( \langle x := 1 \) or \( (x := 2; x := x + 2), s \rangle \rightarrow s[x \mapsto 1] \)
  ▸ NS: \( \langle x := 1 \) or \( (x := 2; x := x + 2), s \rangle \rightarrow s[x \mapsto 4] \)
  ▸ Similar in SOS

▸ \((\text{while true do skip}) \) or \((x := 2; x := x + 2)\)
Non-determinism: Two examples

- $x := 1 \text{ or } (x := 2; x := x + 2)$
  - NS: $\langle x := 1 \text{ or } (x := 2; x := x + 2), s \rangle \rightarrow s[x \mapsto 1]$
  - NS: $\langle x := 1 \text{ or } (x := 2; x := x + 2), s \rangle \rightarrow s[x \mapsto 4]$
  - Similar in SOS

- $(\text{while true do skip}) \text{ or } (x := 2; x := x + 2)$
  - NS: $\langle(\text{while true do skip}) \text{ or } (x := 2; x := x + 2), s \rangle \rightarrow s[x \mapsto 4]$
  - is the only derivation tree!
Non-determinism: Two examples

- \( x := 1 \) or \( (x := 2; x := x + 2) \)
  - NS: \( \langle x := 1 \text{ or } (x := 2; x := x + 2), s \rangle \rightarrow s[x \mapsto 1] \)
  - NS: \( \langle x := 1 \text{ or } (x := 2; x := x + 2), s \rangle \rightarrow s[x \mapsto 4] \)
  - Similar in SOS

- \( (\text{while true do skip}) \text{ or } (x := 2; x := x + 2) \)
  - NS: \( \langle (\text{while true do skip}) \text{ or } (x := 2; x := x + 2), s \rangle \rightarrow s[x \mapsto 4] \)
    is the only derivation tree!
  - SOS: Infinite derivation sequence, if the left branch is chosen; finite derivation sequence ending in \( s[x \mapsto 4] \) if the right branch is chosen
Non-determinism: Two examples

- $x := 1$ or $(x := 2; x := x + 2)$
  - NS: $\langle x := 1 \text{ or } (x := 2; x := x + 2), s \rangle \rightarrow s[x \mapsto 1]$
  - NS: $\langle x := 1 \text{ or } (x := 2; x := x + 2), s \rangle \rightarrow s[x \mapsto 4]$
  - Similar in SOS

- $(\text{while true do skip}) \text{ or } (x := 2; x := x + 2)$
  - NS: $\langle (\text{while true do skip}) \text{ or } (x := 2; x := x + 2), s \rangle \rightarrow s[x \mapsto 4]$
    is the only derivation tree!
  - SOS: Infinite derivation sequence, if the left branch is chosen; finite derivation sequence ending in $s[x \mapsto 4]$ if the right branch is chosen

NS suppresses looping! (Never chooses the “wrong” branch.)
SOS: Looping is not suppressed.
Parallelism

- We extend the syntax of While with the statement \( \text{par} \)

\[
S ::= \ldots
\mid S_1 \text{ par } S_2
\]

- **Intuition:** For \( S_1 \text{ par } S_2 \), both branches have to be executed, but their execution can be *interleaved*

- For example: \( x := 1 \text{ par } x := 2; x := x + 2 \)
Parallelism

- We extend the syntax of While with the statement `par`

\[
S ::= \ldots \\
| \quad S_1 \text{ par } S_2
\]

- **Intuition:** For \( S_1 \text{ par } S_2 \), both branches have to be executed, but their execution can be *interleaved*

- For example: \( x := 1 \text{ par } x := 2; x := x + 2 \) can be executed as either:
  
  \[
  x := 1; x := 2; x := x + 2 \\
  x := 2; x := 1; x := x + 2 \\
  x := 2; x := x + 2; x := 1
  \]

- For SOS new rules can be introduced to handle the statement

- For NS?
Rules for parallelism in SOS

\[ [\text{par}^1_{\text{sos}}] \] \[ \langle S_1, s \rangle \Rightarrow \langle S'_1, s' \rangle \]
\[ \langle S_1 \text{ par } S_2, s \rangle \Rightarrow \langle S'_1 \text{ par } S_2, s' \rangle \]

\[ [\text{par}^2_{\text{sos}}] \] \[ \langle S'_1, s \rangle \Rightarrow s' \]
\[ \langle S_1 \text{ par } S_2, s \rangle \Rightarrow \langle S_2, s' \rangle \]

\[ [\text{par}^3_{\text{sos}}] \] \[ \langle S_2, s \rangle \Rightarrow \langle S'_2, s' \rangle \]
\[ \langle S_1 \text{ par } S_2, s \rangle \Rightarrow \langle S_1 \text{ par } S'_2, s' \rangle \]

\[ [\text{par}^4_{\text{sos}}] \] \[ \langle S_2, s \rangle \Rightarrow s' \]
\[ \langle S_1 \text{ par } S_2, s \rangle \Rightarrow \langle S_1, s' \rangle \]
An example of parallelism in SOS (1/3)

\[
\langle x := 1 \text{ par } (x := 2; x := x+2), \ s \rangle \Rightarrow \langle x := 2; x := x+2, \ s[x\mapsto 1] \rangle \\
\Rightarrow \langle x := x+2, \ s[x\mapsto 2] \rangle \\
\Rightarrow s[x\mapsto 4]
\]
An example of parallelism in SOS (2/3)

\[
\langle x := 1 \par (x := 2; x := x+2), s \rangle \Rightarrow \langle x := 1 \par x := x+2, s[x\mapsto 2] \rangle \\
\Rightarrow \langle x := 1, s[x\mapsto 4] \rangle \\
\Rightarrow s[x\mapsto 1]
\]
An example of parallelism in SOS (3/3)

\[
\langle x := 1 \par (x := 2; x := x+2), s \rangle \Rightarrow \langle x := 1 \par x := x + 2, s[x\mapsto 2] \rangle \\
\Rightarrow \langle x := x + 2, s[x\mapsto 1] \rangle \\
\Rightarrow s[x\mapsto 3]
\]
Attempt at rules for parallelism in Natural Semantics

\[
\begin{align*}
\langle S_1, s \rangle &\rightarrow s', \langle S_2, s' \rangle \rightarrow s'' \\
\hline
\langle S_1 \text{ par } S_2, s \rangle &\rightarrow s'' \\
\langle S_2, s \rangle &\rightarrow s', \langle S_1, s' \rangle \rightarrow s'' \\
\hline
\langle S_1 \text{ par } S_2, s \rangle &\rightarrow s''
\end{align*}
\]

Unfortunately these are erroneous as they do not express interleaving – so Natural Semantics cannot express parallelism
## Summary: Non-sequential Language Constructs

<table>
<thead>
<tr>
<th></th>
<th>Natural semantics</th>
<th>Structural Operational Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-determinism</td>
<td>suppresses looping</td>
<td>does not suppress looping</td>
</tr>
<tr>
<td>Parallelism</td>
<td>cannot express interleaving of computations</td>
<td>can express interleaving of computations</td>
</tr>
</tbody>
</table>
Summary

- Recap
- Extensions of operational semantics
- non-determinism, parallelism

Exercise Class
- Exercises 3.1, 3.3, 3.4, 3.5, 3.6 from SwA.