

Concepts of Programming Languages – Recitation 3: (More) Natural Operational Semantics and Structural Operational Semantics

Oded Padon

Reference:

Semantics with Applications by H. Nielson and F. Nielson – Ch. 2
http://www.daimi.au.dk/~bra8130/Wiley_book/wiley.html

Operational Semantics

- Formal definition of program semantics
- The meaning of the program is described “operationally”
- Natural Operational Semantics - **NOS**
- Structural Operational Semantics - **SOS**
- We can go **systematically** from operational semantics (NOS or SOS) to an **interpreter**

The **While** Programming Language

- Abstract syntax

$S ::= x := a \mid \mathbf{skip} \mid S_1; S_2 \mid \mathbf{if} \ b \ \mathbf{then} \ S_1 \ \mathbf{else} \ S_2 \mid$
 $\mathbf{while} \ b \ \mathbf{do} \ S$

- Use parentheses for precedence
- Informal Semantics
 - **skip** behaves like no-operation
 - Import meaning of arithmetic and Boolean operations

Semantics of arithmetic expressions

- Assume that arithmetic expressions are side-effect free
- $\mathcal{A}[\text{Aexp}] : \text{State} \rightarrow \mathbb{N}$
- Defined by **structural** induction on the syntax tree

$$\mathcal{A}[n]s = \mathcal{N}[n]$$

$$\mathcal{A}[x]s = s\ x$$

$$\mathcal{A}[a_1 + a_2]s = \mathcal{A}[a_1]s + \mathcal{A}[a_2]s$$

$$\mathcal{A}[a_1 \star a_2]s = \mathcal{A}[a_1]s \star \mathcal{A}[a_2]s$$

$$\mathcal{A}[a_1 - a_2]s = \mathcal{A}[a_1]s - \mathcal{A}[a_2]s$$

Semantics of Boolean expressions

- Assume that Boolean expressions are side-effect free
- $\mathcal{B}[\text{Bexp}] : \text{State} \rightarrow \mathbb{T}$
- Defined by induction on the syntax tree:

$$\mathcal{B}[\text{true}]_s = \mathbf{tt}$$

$$\mathcal{B}[\text{false}]_s = \mathbf{ff}$$

$$\mathcal{B}[a_1 = a_2]_s = \begin{cases} \mathbf{tt} & \text{if } \mathcal{A}[a_1]_s = \mathcal{A}[a_2]_s \\ \mathbf{ff} & \text{if } \mathcal{A}[a_1]_s \neq \mathcal{A}[a_2]_s \end{cases}$$

$$\mathcal{B}[a_1 \leq a_2]_s = \begin{cases} \mathbf{tt} & \text{if } \mathcal{A}[a_1]_s \leq \mathcal{A}[a_2]_s \\ \mathbf{ff} & \text{if } \mathcal{A}[a_1]_s > \mathcal{A}[a_2]_s \end{cases}$$

$$\mathcal{B}[\neg b]_s = \begin{cases} \mathbf{tt} & \text{if } \mathcal{B}[b]_s = \mathbf{ff} \\ \mathbf{ff} & \text{if } \mathcal{B}[b]_s = \mathbf{tt} \end{cases}$$

$$\mathcal{B}[b_1 \wedge b_2]_s = \begin{cases} \mathbf{tt} & \text{if } \mathcal{B}[b_1]_s = \mathbf{tt} \text{ and } \mathcal{B}[b_2]_s = \mathbf{tt} \\ \mathbf{ff} & \text{if } \mathcal{B}[b_1]_s = \mathbf{ff} \text{ or } \mathcal{B}[b_2]_s = \mathbf{ff} \end{cases}$$

Natural Operational Semantics

- Notations:
 - S – program construct (word in the While language)
 - s, s' – states (functions $\text{Var} \rightarrow \mathbb{N}$)
- $\langle S, s \rangle \rightarrow s'$ means:
If S is executed on state s , it terminates and the state after execution is s'
- Describe the “overall” effect of program constructs
- Ignores non terminating computations

Formally defining \rightarrow

- \rightarrow is defined **inductively** using **inference rules**, with both **syntactic** conditions on S and **semantic** conditions on s

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

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

$$[\text{comp}_{\text{ns}}] \quad \frac{\langle S_1, s \rangle \rightarrow s', \langle S_2, s' \rangle \rightarrow s''}{\langle S_1; S_2, s \rangle \rightarrow s''}$$

$$[\text{if}_{\text{ns}}^{\text{tt}}] \quad \frac{\langle S_1, s \rangle \rightarrow s'}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s \rangle \rightarrow s'} \quad \text{if } \mathcal{B}[[b]]s = \text{tt}$$

$$[\text{if}_{\text{ns}}^{\text{ff}}] \quad \frac{\langle S_2, s \rangle \rightarrow s'}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s \rangle \rightarrow s'} \quad \text{if } \mathcal{B}[[b]]s = \text{ff}$$

$$[\text{while}_{\text{ns}}^{\text{tt}}] \quad \frac{\langle S, s \rangle \rightarrow s', \langle \text{while } b \text{ do } S, s' \rangle \rightarrow s''}{\langle \text{while } b \text{ do } S, s \rangle \rightarrow s''} \quad \text{if } \mathcal{B}[[b]]s = \text{tt}$$

$$[\text{while}_{\text{ns}}^{\text{ff}}] \quad \langle \text{while } b \text{ do } S, s \rangle \rightarrow s \quad \text{if } \mathcal{B}[[b]]s = \text{ff}$$

Derivation Trees

- A derivation tree is a way to write applications of inference rules
- A derivation tree is a “proof” that $\langle S, s \rangle \rightarrow s'$
- The root of tree is $\langle S, s \rangle \rightarrow s'$
- Each node is a conclusion from its children using an inference rule
- Leaves are instances of axioms (rules with no premises)
- Non-leaves are instances of inference rules with premises
 - Immediate children match rule premises
 - The semantic condition is satisfied

Example Derivation Tree

$\langle x := x + 1 ; \text{if } x > 0 \text{ then } y := 2 \text{ else } y := 3, s_0 \rangle \rightarrow s_0[x \mapsto 1] [y \mapsto 2]$

comp_{ns}

$\langle x := x + 1, s_0 \rangle \rightarrow s_0[x \mapsto 1]$

ass_{ns}

$\langle \text{if } x > 0 \text{ then } y := 2 \text{ else } y := 3, s_0[x \mapsto 1] \rangle \rightarrow s_0[x \mapsto 1] [y \mapsto 2]$

$\text{if}_{\text{ns}}^{\text{tt}}$

$\langle y := 2, s_0[x \mapsto 1] \rangle \rightarrow s_0[x \mapsto 1] [y \mapsto 2]$

ass_{ns}

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

$[\text{comp}_{\text{ns}}] \quad \frac{\langle S_1, s \rangle \rightarrow s', \langle S_2, s' \rangle \rightarrow s''}{\langle S_1; S_2, s \rangle \rightarrow s''}$

$[\text{if}_{\text{ns}}^{\text{tt}}] \quad \frac{\langle S_1, s \rangle \rightarrow s'}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s \rangle \rightarrow s'} \quad \text{if } \mathcal{B}[[b]]s = \text{tt}$

$[\text{if}_{\text{ns}}^{\text{ff}}] \quad \frac{\langle S_2, s \rangle \rightarrow s'}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s \rangle \rightarrow s'} \quad \text{if } \mathcal{B}[[b]]s = \text{ff}$

Alternative Notation

$$\text{ass}_{\text{ns}} \frac{}{\langle y := 2, s_0[x \mapsto 1] \rangle \rightarrow s_0[x \mapsto 1][y \mapsto 2]}$$

$$\text{ass}_{\text{ns}} \frac{}{\langle x := x+1, s_0 \rangle \rightarrow s_0[x \mapsto 1]} \quad \text{if}_{\text{ns}}^{\text{tt}} \frac{}{\langle \text{if } x > 0 \text{ then } y := 2 \text{ else } y := 3, s_0[x \mapsto 1] \rangle \rightarrow s_0[x \mapsto 1][y \mapsto 2]}$$

$$\text{comp}_{\text{ns}} \frac{}{\langle x := x+1 ; \text{if } x > 0 \text{ then } y := 2 \text{ else } y := 3, s_0 \rangle \rightarrow s_0[x \mapsto 1][y \mapsto 2]}$$

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

$$[\text{comp}_{\text{ns}}] \quad \frac{\langle S_1, s \rangle \rightarrow s', \langle S_2, s' \rangle \rightarrow s''}{\langle S_1; S_2, s \rangle \rightarrow s''}$$

$$[\text{if}_{\text{ns}}^{\text{tt}}] \quad \frac{\langle S_1, s \rangle \rightarrow s'}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s \rangle \rightarrow s'} \quad \text{if } \mathcal{B}[b]s = \text{tt}$$

$$[\text{if}_{\text{ns}}^{\text{ff}}] \quad \frac{\langle S_2, s \rangle \rightarrow s'}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s \rangle \rightarrow s'} \quad \text{if } \mathcal{B}[b]s = \text{ff}$$

Semantic Equivalence

- Two statements S_1 and S_2 are **semantically equivalent** ($S_1 \approx S_2$) if for any two states s, s' we have:

$$\langle S_1, s \rangle \rightarrow s' \quad \text{iff} \quad \langle S_2, s \rangle \rightarrow s'$$

- Examples:

- $S ; \text{skip} \approx S$?
- $\text{skip} ; S \approx S$?
- $S1 ; S2 \approx S2 ; S1$?
- $(S1 ; S2) ; S3 \approx (S1 ; (S2 ; S3))$?
- $x := 2 ; y := x + 3 \approx y := 5 ; x := 2$?
- $\text{while } b \text{ do } S \approx \text{if } b \text{ then } (S ; \text{while } b \text{ do } S) \text{ else skip}$?
- $\text{while true do } x:=x+1 \approx \text{while true do } x:=x+2$?

Proving Semantic Equivalence

- Two statements S_1 and S_2 are **semantically equivalent** ($S_1 \approx S_2$) if for any two states s, s' we have:

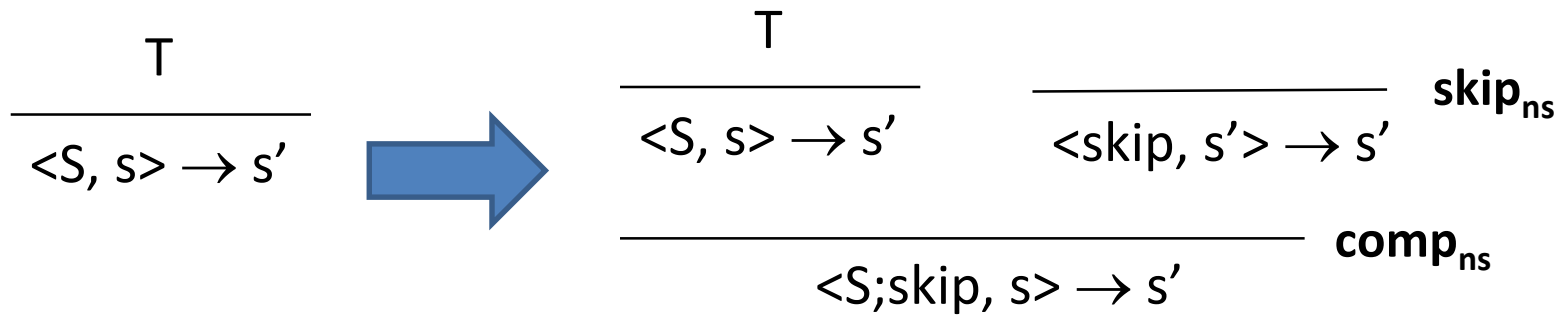
$$\langle S_1, s \rangle \rightarrow s' \quad \text{iff} \quad \langle S_2, s \rangle \rightarrow s'$$

- To prove semantic equivalence:
 1. Assume $\langle S_1, s \rangle \rightarrow s'$, show $\langle S_2, s \rangle \rightarrow s'$
 2. Assume $\langle S_2, s \rangle \rightarrow s'$, show $\langle S_1, s \rangle \rightarrow s'$
- The proof is usually by manipulating derivation trees

Example: $S ; \text{skip} \approx S$

- To prove semantic equivalence:
 1. Assume $\langle S, s \rangle \rightarrow s'$, show $\langle S; \text{skip}, s \rangle \rightarrow s'$
 2. Assume $\langle S; \text{skip}, s \rangle \rightarrow s'$, show $\langle S, s \rangle \rightarrow s'$

1.



Example: $S ; \text{skip} \approx S$

- To prove semantic equivalence:
 1. Assume $\langle S, s \rangle \rightarrow s'$, show $\langle S; \text{skip}, s \rangle \rightarrow s'$
 2. Assume $\langle S; \text{skip}, s \rangle \rightarrow s'$, show $\langle S, s \rangle \rightarrow s'$

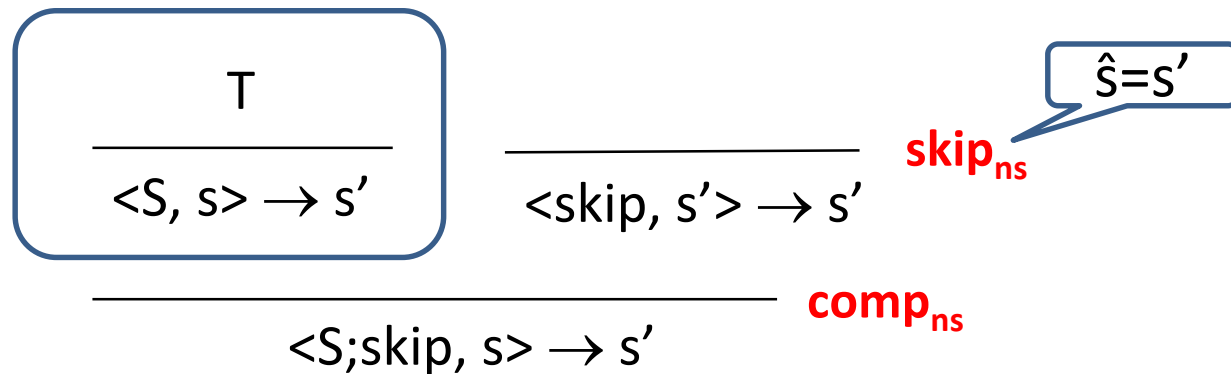
2.

$$\frac{\frac{\text{T}}{\langle S, s \rangle \rightarrow \hat{s}} \quad \frac{\text{skip}_{\text{ns}}}{\langle \text{skip}, \hat{s} \rangle \rightarrow s'}}{\text{comp}_{\text{ns}} \quad \langle S; \text{skip}, s \rangle \rightarrow s'}$$

Example: $S ; \text{skip} \approx S$

- To prove semantic equivalence:
 1. Assume $\langle S, s \rangle \rightarrow s'$, show $\langle S; \text{skip}, s \rangle \rightarrow s'$
 2. Assume $\langle S; \text{skip}, s \rangle \rightarrow s'$, show $\langle S, s \rangle \rightarrow s'$

2.



Example: while loop

Prove: $\underbrace{\text{while } b \text{ do } S}_W \approx \text{if } b \text{ then } (S ; \text{while } b \text{ do } S) \text{ else skip}$

W

1. Assume $\langle W, s \rangle \rightarrow s''$, show $\langle \text{if } b \text{ then } (S ; W) \text{ else skip}, s \rangle \rightarrow s''$

Two cases:

$$\frac{\frac{T_1}{\langle S, s \rangle \rightarrow s'} \quad \frac{T_2}{\langle \text{while } b \text{ do } S, s' \rangle \rightarrow s''}}{\langle \text{while } b \text{ do } S, s \rangle \rightarrow s''} \text{while}_{ns}^{tt}$$

$$\mathcal{B}[b] s = \mathbf{tt}$$

$$\frac{}{\langle \text{while } b \text{ do } S, s \rangle \rightarrow s} \text{while}_{ns}^{ff}$$

$$\mathcal{B}[b] s = \mathbf{ff}$$

$$s = s''$$

$$[\text{while}_{ns}^{tt}] \quad \frac{\langle S, s \rangle \rightarrow s', \langle \text{while } b \text{ do } S, s' \rangle \rightarrow s''}{\langle \text{while } b \text{ do } S, s \rangle \rightarrow s''} \text{ if } \mathcal{B}[b] s = \mathbf{tt}$$

$$[\text{while}_{ns}^{ff}] \quad \langle \text{while } b \text{ do } S, s \rangle \rightarrow s \text{ if } \mathcal{B}[b] s = \mathbf{ff}$$

Case $\text{while}_{\text{ns}}^{\text{ff}}$

$$\frac{}{\langle \text{while } b \text{ do } S, s \rangle \rightarrow s} \quad \text{while}_{\text{ns}}^{\text{ff}} \quad \mathcal{B}[b] s = \text{ff} \\ s = s''$$



$$\frac{}{\langle \text{skip}, s \rangle \rightarrow s} \quad \text{skip}_{\text{ns}}$$

$$\frac{}{\langle \text{if } b \text{ then } (S;W) \text{ else skip}, s \rangle \rightarrow s} \quad \text{if}_{\text{ns}}^{\text{ff}}$$

$$[\text{if}_{\text{ns}}^{\text{tt}}] \quad \frac{\langle S_1, s \rangle \rightarrow s'}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s \rangle \rightarrow s'} \quad \text{if } \mathcal{B}[b] s = \text{tt}$$

$$[\text{if}_{\text{ns}}^{\text{ff}}] \quad \frac{\langle S_2, s \rangle \rightarrow s'}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s \rangle \rightarrow s'} \quad \text{if } \mathcal{B}[b] s = \text{ff}$$

Case while_{ns}^{tt}

$$\frac{\frac{T_1}{\langle S, s \rangle \rightarrow s'} \quad \frac{T_2}{\langle \text{while } b \text{ do } S, s' \rangle \rightarrow s''}}{\langle \text{while } b \text{ do } S, s \rangle \rightarrow s''} \text{while}_{ns}^{tt} \quad \mathcal{B}[b] \text{ } s = tt$$



$$\frac{\frac{\frac{T_1}{\langle S, s \rangle \rightarrow s'} \quad \frac{T_2}{\langle W, s' \rangle \rightarrow s''}}{\langle S; W, s \rangle \rightarrow s''} \text{comp}_{ns}}{\langle \text{if } b \text{ then } (S; W) \text{ else skip, } s \rangle \rightarrow s''} \text{if}_{ns}^{tt}$$

Direction 2

2. Assume $\langle \text{if } b \text{ then } (S ; W) \text{ else skip}, s \rangle \rightarrow s''$, show $\langle W, s \rangle \rightarrow s''$
Two cases:

$$\begin{array}{c}
 \frac{\frac{T_3}{\langle S, s \rangle \rightarrow s'}}{\quad} \quad \frac{T_4}{\langle W, s' \rangle \rightarrow s''} \\
 \hline
 \langle S;W, s \rangle \rightarrow s'' \quad \text{comp}_{\text{ns}} \\
 \hline
 \langle \text{if } b \text{ then } (S;W) \text{ else skip}, s \rangle \rightarrow s'' \quad \text{if}_{\text{ns}}^{\text{tt}}
 \end{array}$$

$$\mathcal{B}[b] s = \text{tt}$$

$$\begin{array}{c}
 \frac{\quad}{\langle \text{skip}, s \rangle \rightarrow s} \quad \text{skip}_{\text{ns}} \\
 \hline
 \langle \text{if } b \text{ then } (S;W) \text{ else skip}, s \rangle \rightarrow s \quad \text{if}_{\text{ns}}^{\text{ff}}
 \end{array}$$

$$\begin{array}{l}
 \mathcal{B}[b] s = \text{ff} \\
 s = s''
 \end{array}$$

Case if_{ns}^{ff}

$\frac{}{\langle \text{skip}, s \rangle \rightarrow s}$ skip_{ns}

$\frac{}{\langle \text{if } b \text{ then } (S;W) \text{ else skip}, s \rangle \rightarrow s}$ if_{ns}^{ff}

$\mathcal{B}[b] \ s = \text{ff}$
 $s = s''$



$\frac{}{\langle \text{while } b \text{ do } S, s \rangle \rightarrow s}$ while_{ns}^{ff}

Case $\text{if}_{\text{ns}}^{\text{tt}}$

$$\frac{\frac{\frac{T_3}{\langle S, s \rangle \rightarrow s'} \quad \frac{T_4}{\langle W, s' \rangle \rightarrow s''}}{\langle S; W, s \rangle \rightarrow s''} \text{comp}_{\text{ns}}}{\langle \text{if } b \text{ then } (S; W) \text{ else skip, } s \rangle \rightarrow s''} \text{if}_{\text{ns}}^{\text{tt}} \quad \mathcal{B}[b] s = \text{tt}$$



$$\frac{\frac{T_3}{\langle S, s \rangle \rightarrow s'} \quad \frac{T_4}{\langle \text{while } b \text{ do } S, s' \rangle \rightarrow s''}}{\langle \text{while } b \text{ do } S, s \rangle \rightarrow s''} \text{while}_{\text{ns}}^{\text{tt}}$$

Adding Semantics of Conditions with Side Effects

Add semantics of:

if (x:=e) then S_1 else S_2

$$\text{if-ass}_{\text{ns}}^{\text{tt}} \frac{\langle x:=e, s \rangle \rightarrow s' \quad \langle S_1, s' \rangle \rightarrow s''}{\langle \text{if (x:=e) then } S_1 \text{ else } S_2, s \rangle \rightarrow s''} \quad \text{if } \mathbf{A}[[x]]s' \neq 0$$

$$\text{if-ass}_{\text{ns}}^{\text{ff}} \frac{\langle x:=e, s \rangle \rightarrow s' \quad \langle S_2, s' \rangle \rightarrow s''}{\langle \text{if (x:=e) then } S_1 \text{ else } S_2, s \rangle \rightarrow s''} \quad \text{if } \mathbf{A}[[x]]s' = 0$$

Adding Semantics of 3-Way Case

Add semantics of:

case (b₁: S₁), (b₂: S₂), (else: S₃)

$$\text{case}_{\text{ns}}^1 \frac{\langle S_1, s \rangle \rightarrow s'}{\langle \text{case } (b_1: S_1), (b_2: S_2), (\text{else}: S_3), s \rangle \rightarrow s'} \quad \text{if } \mathbf{B}[[b_1]]s = \mathbf{tt}$$

$$\text{case}_{\text{ns}}^2 \frac{\langle S_2, s \rangle \rightarrow s'}{\langle \text{case } (b_1: S_1), (b_2: S_2), (\text{else}: S_3), s \rangle \rightarrow s'} \quad \text{if } \mathbf{B}[[b_1]]s = \mathbf{ff} \text{ and } \mathbf{B}[[b_2]]s = \mathbf{tt}$$

$$\text{case}_{\text{ns}}^{\text{else}} \frac{\langle S_3, s \rangle \rightarrow s'}{\langle \text{case } (b_1: S_1), (b_2: S_2), (\text{else}: S_3), s \rangle \rightarrow s'} \quad \text{if } \mathbf{B}[[b_1]]s = \mathbf{ff} \text{ and } \mathbf{B}[[b_2]]s = \mathbf{ff}$$

Structural Operational Semantics

- Emphasizes the individual execution steps
- $\langle S, s \rangle \Rightarrow \gamma$
 - the **"first"** step of executing **S** on state **s** leads to γ
- Two possibilities for γ
 - $\gamma = \langle S', s' \rangle$

The execution of S is not completed and, S' is the remaining computation to be performed on s'
 - $\gamma = s'$

The execution of S has terminated with a final state s'
- γ is a **stuck** configuration when there are no transitions

Formally defining \Rightarrow

- \Rightarrow is defined **inductively** using **inference rules**, with both **syntactic** conditions on S and **semantic** conditions on s

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

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

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

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

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

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

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

Examples for \Rightarrow

s_0 : state which assigns zero to all variables

- $\langle y := 2, s_0[x \mapsto 1] \rangle \Rightarrow s_0[x \mapsto 1] [y \mapsto 2]$
- $\langle x := x+1, s_0 \rangle \Rightarrow s_0[x \mapsto 1]$
- $\langle x := x+2 ; x := x+3, s_0 \rangle \Rightarrow \langle x := x+3, s_0[x \mapsto 2] \rangle$
- $\langle \text{if } x > 0 \text{ then } y := 2 \text{ else } y := 3, s_0 \rangle \Rightarrow \langle y := 3, s_0 \rangle$
- $\langle \text{if } x > 0 \text{ then } y := 2 \text{ else } y := 3, s_0[x \mapsto 1] \rangle \Rightarrow \langle y := 2, s_0[x \mapsto 1] \rangle$
- $\langle \text{while } x < 5 \text{ do } x := x+2, s_0 \rangle \Rightarrow$
 $\langle \text{if } x < 5 \text{ then } (x := x+2 ; \text{while } x < 5 \text{ do } x := x+2) \text{ else skip}, s_0 \rangle$
- $\langle \text{while } x < 0 \text{ do } x := x+2, s_0 \rangle \Rightarrow$
 $\langle \text{if } x < 0 \text{ then } (x := x+2 ; \text{while } x < 0 \text{ do } x := x+2) \text{ else skip}, s_0 \rangle$

Finite Derivation Sequences

- Computation is modeled by **derivation sequences**
- A finite derivation sequence starting at $\langle S, s \rangle$:
 - $\gamma_0, \gamma_1, \gamma_2 \dots, \gamma_k$
 - $\gamma_0 = \langle S, s \rangle$
 - $\gamma_i \Rightarrow \gamma_{i+1}$
 - γ_k is either stuck configuration or a final state
- For each step there is a derivation tree
- $\gamma_0 \Rightarrow^k \gamma_k$ in k steps
- $\gamma_0 \Rightarrow^* \gamma$ in some finite number of steps
- Models terminating computation

Infinite Derivation Sequences

- An infinite derivation sequence starting at $\langle S, s \rangle$:
 - $\gamma_0, \gamma_1, \gamma_2 \dots$ such that
 - $\gamma_0 = \langle S, s \rangle$
 - $\gamma_i \Rightarrow \gamma_{i+1}$
- Example
 - $S = \text{while true do skip}$
 - $s = s_0$
- Models non-terminating computation

Program Termination

- Given a statement S and input s
 - S **terminates** on s if there exists a finite derivation sequence starting at $\langle S, s \rangle$
 - S **terminates successfully** on s if there exists a finite derivation sequence starting at $\langle S, s \rangle$ leading to a final state
 - S **loops** on s if there exists an infinite derivation sequence starting at $\langle S, s \rangle$

From SOS to an interpreter

- Implement semantics of arithmetic and Boolean expressions
- Implement SOS as a function:
 - input: S, s
 - output: γ such that $\langle S, s \rangle \Rightarrow \gamma$
 - either $\gamma = \langle S', s' \rangle$ or $\gamma = s'$
- The function is recursive
 - call tree matches derivation tree of each step
- Iteratively call the SOS function until a final or stuck configuration is reached
- The call sequence matches the derivation sequence
- In While: the derivation sequence is unique, and there are no stuck configurations

LIVE CODE DEMO

[sos.py](https://sossolutions.com)

Proving properties of SOS

- Many properties are proved by induction on the length of derivation sequence
- Prove that the property holds for all derivation sequences of length 0
- Prove that the property holds for all other derivation sequences:
 - Assume the property holds for all sequences of length k or less (induction hypothesis)
 - Show that the property holds for sequences of length $k+1$

Example Property: Sequential Composition

- If $\langle S_1; S_2, s \rangle \Rightarrow^k s''$ then there exists a state s' and numbers k_1 and k_2 such that:
 - $\langle S_1, s \rangle \Rightarrow^{k_1} s'$
 - $\langle S_2, s' \rangle \Rightarrow^{k_2} s''$
 - $k = k_1 + k_2$

Proof

- If $\langle S_1; S_2, s \rangle \Rightarrow^k s''$ then there exists a state s' and numbers k_1 and k_2 such that: $\langle S_1, s \rangle \Rightarrow^{k_1} s'$, $\langle S_2, s' \rangle \Rightarrow^{k_2} s''$ and $k = k_1 + k_2$
- Base case: $k=0$, holds vacuously

Inductive Step

- **Assume for all $j \leq k$:** if $\langle S_1; S_2, s \rangle \Rightarrow^j s''$ then there exists a state s' and numbers j_1 and j_2 such that: $\langle S_1, s \rangle \Rightarrow^{j_1} s'$, $\langle S_2, s' \rangle \Rightarrow^{j_2} s''$ and $j = j_1 + j_2$
- Assume $\langle S_1; S_2, s \rangle \Rightarrow^{k+1} s''$, then, by definition of \Rightarrow^n :
 - $\langle S_1; S_2, s \rangle \Rightarrow \gamma$
 - $\gamma \Rightarrow^k s''$
 - γ is not a final or stuck configuration
- Split according to two cases of $\langle S_1; S_2, s \rangle \Rightarrow \gamma$

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

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

Case $\text{comp}_{\text{sos}}^1$

- $\gamma = \langle S'_1; S_2, s' \rangle$
- $\langle S_1; S_2, s \rangle \Rightarrow \langle S'_1; S_2, s' \rangle \quad [\text{comp}_{\text{sos}}^1] \quad \frac{\langle S_1, s \rangle \Rightarrow \langle S'_1, s' \rangle}{\langle S_1; S_2, s \rangle \Rightarrow \langle S'_1; S_2, s' \rangle}$
- $\langle S'_1; S_2, s' \rangle \Rightarrow^k s''$
- By induction hypothesis on $\langle S'_1; S_2, s' \rangle \Rightarrow^k s''$ we get j_1, j_2 and state \hat{s} such that:
 - $k = j_1 + j_2$
 - $\langle S'_1, s' \rangle \Rightarrow^{j_1} \hat{s}$
 - $\langle S_2, \hat{s} \rangle \Rightarrow^{j_2} s''$
- Set $k_1 = j_1 + 1, k_2 = j_2$ and get:
 - $k+1 = k_1 + k_2$
 - $\langle S_1, s \rangle \Rightarrow^{k_1} \hat{s}$ since $\langle S_1, s \rangle \Rightarrow \langle S'_1, s' \rangle \Rightarrow^{j_1} \hat{s}$
 - $\langle S_2, \hat{s} \rangle \Rightarrow^{k_2} s''$

Case $\text{comp}_{\text{sos}}^2$

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

$[\text{comp}_{\text{sos}}^2]$

$$\frac{\langle S_1, s \rangle \Rightarrow s'}{\langle S_1; S_2, s \rangle \Rightarrow \langle S_2, s' \rangle}$$

- Set $k_1 = 1, k_2 = k$ and get:
 - $k+1 = k_1 + k_2$
 - $\langle S_1, s \rangle \Rightarrow^{k_1} s'$
 - $\langle S_2, s' \rangle \Rightarrow^{k_2} s''$