

# Concepts of Programming Languages – Recitation 2: Natural 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](http://www.daimi.au.dk/~bra8130/Wiley_book/wiley.html)

# Formal Semantics

- Operational Semantics
  - The meaning of the program is described “operationally”
  - **Natural Operational Semantics ← today!**
  - Structural Operational Semantics
- Denotational Semantics
  - The meaning of the program is an input/output relation
  - Mathematically challenging but complicated
- Axiomatic Semantics
  - The meaning of the program are observed properties

# 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

# Example While Program

```
y := 1;  
while  $\neg(x=1)$  do (  
    y := y * x;  
    x := x - 1  
)
```

# General Notations

- Syntactic categories
  - Var the set of program variables
  - Aexp the set of arithmetic expressions
  - Bexp the set of Boolean expressions
  - Stm set of program statements
- Semantic categories
  - Natural values  $N = \{0, 1, 2, \dots\}$
  - Truth values  $T = \{ff, tt\}$
  - States  $\text{State} = \text{Var} \rightarrow N$
  - Lookup in a state  $s: s \ x$
  - Update of a state  $s: s \ [ \ x \mapsto 5 \ ]$

# Example State Manipulations

- $[x \mapsto 1, y \mapsto 7, z \mapsto 16] y =$
- $[x \mapsto 1, y \mapsto 7, z \mapsto 16] t =$
- $[x \mapsto 1, y \mapsto 7, z \mapsto 16][x \mapsto 5] =$
- $[x \mapsto 1, y \mapsto 7, z \mapsto 16][x \mapsto 5] x =$
- $[x \mapsto 1, y \mapsto 7, z \mapsto 16][x \mapsto 5] y =$

# 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

# Examples for $\rightarrow$

- $\langle y := 2, s_0[x \mapsto 1] \rangle \rightarrow s_0[x \mapsto 1] [y \mapsto 2]$
- $\langle x := x+1, s_0 \rangle \rightarrow s_0[x \mapsto 1]$
- $\langle x := x+1, s_0[x \mapsto 1] \rangle \rightarrow s_0[x \mapsto 2]$
- $\langle x := x+1 ; x := x+1, s_0 \rangle \rightarrow s_0[x \mapsto 2]$
- $\langle \text{if } x > 0 \text{ then } y := 2 \text{ else } y := 3, s_0 \rangle \rightarrow s_0[y \mapsto 3]$
- $\langle \text{if } x > 0 \text{ then } y := 2 \text{ else } y := 3, s_0[x \mapsto 1] \rangle \rightarrow s_0[x \mapsto 1] [y \mapsto 2]$
- $\langle x := x+1 ; \text{if } x > 0 \text{ then } y := 2 \text{ else } y := 3, s_0 \rangle \rightarrow s_0[x \mapsto 1] [y \mapsto 2]$
- $\langle \text{while } x < 5 \text{ do } (x := x+2 ; y := y+10), s_0 \rangle \rightarrow s_0[x \mapsto 6] [y \mapsto 30]$
- $\langle (\text{while } x < 5 \text{ do } (x := x+2 ; y := y+10)) ;$   
     $(\text{while } x > 0 \text{ do } x := x-5), s_0 \rangle \rightarrow s_0[x \mapsto -4] [y \mapsto 30]$
- $\langle \text{while } x \geq 0 \text{ do } x := x+1, s_0 \rangle \rightarrow ?$
- **NOT**  $\langle \text{while } x \geq 0 \text{ do } x := x+1, s_0 \rangle \rightarrow s'$  for any  $s'$

$s_0$ : state which assigns zero to all variables

# 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

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

$\text{comp}_{\text{ns}}$

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

$\text{ass}_{\text{ns}}$

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

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

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

$\text{ass}_{\text{ns}}$

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

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

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

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

# Bad Derivation Tree

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

$\text{comp}_{\text{ns}}$

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

$\text{ass}_{\text{ns}}$

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

semantic condition not satisfied:

$B[x > 0] (s_0[x \mapsto 1]) = \text{tt} \neq \text{ff}$

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

$\langle y := 3, s_0[x \mapsto 1] \rangle \rightarrow s_0[x \mapsto 1] [y \mapsto 3]$

$\text{ass}_{\text{ns}}$

$[\text{comp}_{\text{ns}}]$

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

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

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

$[\text{if}_{\text{ns}}^{\text{ff}}]$

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

# Example Derivation Tree

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

$\text{comp}_{\text{ns}}$

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

$\text{ass}_{\text{ns}}$

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

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

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

$\text{ass}_{\text{ns}}$

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

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

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

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

# Alternative Notation

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

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

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

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

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

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

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



# Top Down Evaluation of Derivation Trees

- Given a program  $S$  and an input state  $s$
- Find an output state  $s'$  such that  
 $\langle S, s \rangle \rightarrow s'$
- Start with the root and repeatedly apply rules until the axioms are reached
- Inspect different alternatives in order
- In While  $s'$  and the derivation tree is unique

# Example of Top Down Tree Construction

- Input state  $s$  such that  $s \ x = 2$
- Factorial program

$\langle y := 1; \text{while } \neg(x=1) \text{ do } (y := y * x; x := x - 1), s \rangle \rightarrow s[y \mapsto 2][x \mapsto 1]$

**comp<sub>ns</sub>**

$\langle y := 1, s \rangle \rightarrow s[y \mapsto 1]$

**ass<sub>ns</sub>**

$\langle W, s[y \mapsto 1] \rangle \rightarrow s[y \mapsto 2][x \mapsto 1]$

**while<sup>tt</sup><sub>ns</sub>**

$\langle y := y * x; x := x - 1, s[y \mapsto 1] \rangle \rightarrow s[y \mapsto 2][x \mapsto 1]$

**comp<sub>ns</sub>**

$\langle W, s[y \mapsto 2][x \mapsto 1] \rangle \rightarrow s[y \mapsto 2][x \mapsto 1]$

**while<sup>ff</sup><sub>ns</sub>**

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

**ass<sub>ns</sub>**

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

**ass<sub>ns</sub>**

# From NOS to an interpreter

- Implement semantics of arithmetic and Boolean expressions
- Implement NOS as a function:
  - input:  $S, s$
  - output:  $s'$  such that  $\langle S, s \rangle \rightarrow s'$
- The function is recursive
  - Call tree matches derivation tree
  - Axioms are the recursion base cases
- While is deterministic:  $s'$  and the derivation tree is unique

# LIVE CODE DEMO

1. `while_ast.py`
2. `expr.py`
3. `nos.py`
4. `nos_tree.py`