Concepts of Programming Languages – Recitation 4:
Natural 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 | **skip** | S_1; S_2 | **if** b **then** S_1 **else** S_2 |
  while b do S
- Use parenthesizes 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
- A⟦ Aexp ⟧ : State $\rightarrow$ N
- Defined by structural induction on the syntax tree

$$
\begin{aligned}
\mathcal{A}[\![n]\!]s &= \mathcal{N}[\![n]\!] \\
\mathcal{A}[\![x]\!]s &= s\ x \\
\mathcal{A}[\![a_1 + a_2]\!]s &= \mathcal{A}[\![a_1]\!]s + \mathcal{A}[\![a_2]\!]s \\
\mathcal{A}[\![a_1 \star a_2]\!]s &= \mathcal{A}[\![a_1]\!]s \star \mathcal{A}[\![a_2]\!]s \\
\mathcal{A}[\![a_1 - a_2]\!]s &= \mathcal{A}[\![a_1]\!]s - \mathcal{A}[\![a_2]\!]s
\end{aligned}
$$

Semantics of Boolean expressions

- Assume that Boolean expressions are side-effect free
- B⟦ Bexp ⟧ : State $\rightarrow$ T
- Defined by induction on the syntax tree:

$$
\begin{aligned}
\mathcal{B}[\![\texttt{true}]\!]s &= \mathbf{tt} \\
\mathcal{B}[\![\texttt{false}]\!]s &= \mathbf{ff} \\
\mathcal{B}[\![a_1 = a_2]\!]s &= \begin{cases} \mathbf{tt} & \text{if } \mathcal{A}[\![a_1]\!]s = \mathcal{A}[\![a_2]\!]s \\ \mathbf{ff} & \text{if } \mathcal{A}[\![a_1]\!]s \neq \mathcal{A}[\![a_2]\!]s \end{cases} \\
\mathcal{B}[\![a_1 \leq a_2]\!]s &= \begin{cases} \mathbf{tt} & \text{if } \mathcal{A}[\![a_1]\!]s \leq \mathcal{A}[\![a_2]\!]s \\ \mathbf{ff} & \text{if } \mathcal{A}[\![a_1]\!]s > \mathcal{A}[\![a_2]\!]s \end{cases} \\
\mathcal{B}[\![\neg\ b]\!]s &= \begin{cases} \mathbf{tt} & \text{if } \mathcal{B}[\![b]\!]s = \mathbf{ff} \\ \mathbf{ff} & \text{if } \mathcal{B}[\![b]\!]s = \mathbf{tt} \end{cases} \\
\mathcal{B}[\![b_1 \wedge b_2]\!]s &= \begin{cases} \mathbf{tt} & \text{if } \mathcal{B}[\![b_1]\!]s = \mathbf{tt} \text{ and } \mathcal{B}[\![b_2]\!]s = \mathbf{tt} \\ \mathbf{ff} & \text{if } \mathcal{B}[\![b_1]\!]s = \mathbf{ff} \text{ or } \mathcal{B}[\![b_2]\!]s = \mathbf{ff} \end{cases}
\end{aligned}
$$

Natural Operational Semantics: $\rightarrow$

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

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

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

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

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

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

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

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

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:

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

Structural Operational Semantics

- Emphasizes the individual execution steps
- **<S, *s*>** $\Rightarrow \gamma$
  - the **"first"** step of executing **S** on state ***s*** leads to γ
- Two possibilities for γ
  - γ **= <S', s'>**
    The execution of S is not completed and, S' is the remaining computation to be performed on s'
  - γ **= 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

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

$$[\mathrm{skip}_{\mathrm{sos}}] \qquad \langle \mathtt{skip},\ s\rangle \Rightarrow s$$

$$[\mathrm{comp}^1_{\mathrm{sos}}] \qquad \frac{\langle S_1,\ s\rangle \Rightarrow \langle S_1',\ s'\rangle}{\langle S_1;S_2,\ s\rangle \Rightarrow \langle S_1';S_2,\ s'\rangle}$$

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

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

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

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

SOS Example

$\langle y := 1; \text{while } x \neq 1 \text{ do } (y := y*x; x := x-1), s_0[x \mapsto 3] \rangle \Rightarrow$

$\langle \text{while } x \neq 1 \text{ do } (y := y*x; x := x-1), s_0[x \mapsto 3, y \mapsto 1] \rangle \Rightarrow$

$\langle \text{if } x \neq 1 \text{ then } (y := y*x; x := x-1; \text{while } x \neq 1 \text{ do } (y := y*x; x := x-1)) \text{ else skip}, s_0[x \mapsto 3, y \mapsto 1] \rangle \Rightarrow$

$\langle y := y*x; x := x-1; \text{while } x \neq 1 \text{ do } (y := y*x; x := x-1), s_0[x \mapsto 3, y \mapsto 1] \rangle \Rightarrow$

$\langle x := x-1; \text{while } x \neq 1 \text{ do } (y := y*x; x := x-1), s_0[x \mapsto 3, y \mapsto 3] \rangle \Rightarrow$

$\langle \text{while } x \neq 1 \text{ do } (y := y*x; x := x-1), s_0[x \mapsto 2, y \mapsto 3] \rangle \Rightarrow$

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

$\langle y := y*x; x := x-1; \text{while } x \neq 1 \text{ do } (y := y*x; x := x-1), s_0[x \mapsto 2, y \mapsto 3] \rangle \Rightarrow$

$\langle x := x-1; \text{while } x \neq 1 \text{ do } (y := y*x; x := x-1), s_0[x \mapsto 2, y \mapsto 6] \rangle \Rightarrow$

$\langle \text{while } x \neq 1 \text{ do } (y := y*x; x := x-1), s_0[x \mapsto 1, y \mapsto 6] \rangle \Rightarrow$

$\langle \text{if } x \neq 1 \text{ then } (y := y*x; x := x-1; \text{while } x \neq 1 \text{ do } (y := y*x; x := x-1)) \text{ else skip}, s_0[x \mapsto 1, y \mapsto 6] \rangle \Rightarrow$

$\langle \text{skip}, s_0[x \mapsto 1, y \mapsto 6] \rangle \Rightarrow$

$s_0[x \mapsto 1, y \mapsto 6]$

Finite Derivation Sequences

- Computation is modeled by **derivation sequences**
- A finite derivation sequence starting at <S, s>:
  - $\gamma_0, \gamma_1, \gamma_2 \ldots, \gamma_k$
  - $\gamma_0 = <S, s>$
  - $\gamma_i \Rightarrow \gamma_{i+1}$
  - 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 <S, s>:
  - $\gamma_0, \gamma_1, \gamma_2 \ldots$ such that
  - γ_0=<S, s>
  - $\gamma_i \Rightarrow \gamma_{i+1}$
- Example
  - S = 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 <S, s>
  - S terminates successfully on s if there exists a finite derivation sequence starting at <S, s> leading to a final state
  - S loops on s if there exists an infinite derivation sequence starting at <S, s>

Semantic Equivalence in SOS

- S

No! This is not our definition of semantic equivalence!

We want to have e.g. S1=x:=1;x:=2 equivalent to S2=x:=2. Think about compiler optimization for instance…

Semantic Equivalence in SOS

- S_1 and S_2 are semantically equivalent if the following two conditions hold:
  - For all s and γ which is either final or stuck $\langle S_1, s\rangle \Rightarrow^* \gamma$ if and only if $\langle S_2, s\rangle \Rightarrow^* \gamma$
  - For all s, there is an infinite derivation sequence starting at $\langle S_1, s\rangle$ if and only if there is an infinite derivation sequence starting at $\langle S_2, s\rangle$

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:
  - $\langle S_1, s\rangle \Rightarrow^{k1} s'$
  - $\langle S_2, s'\rangle \Rightarrow^{k2} s''$
  - $k = k_1 + k_2$

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: $\langle S_1, s\rangle \Rightarrow^{k1} s'$, $\langle S_2, s'\rangle \Rightarrow^{k2} s''$ 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: $\langle S_1, s\rangle \Rightarrow^{j1} s'$, $\langle S_2, s'\rangle \Rightarrow^{j2} s''$ 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 $\langle S_{1;}\ S_2, s\rangle \Rightarrow \gamma$

$$[\mathrm{comp}^1_{\mathrm{sos}}] \qquad \frac{\langle S_1,\ s\rangle \Rightarrow \langle S_1',\ s'\rangle}{\langle S_1;S_2,\ s\rangle \Rightarrow \langle S_1';S_2,\ s'\rangle}$$

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

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

- $\gamma = \langle S_2, s' \rangle$
- $\langle S_{1;}\ S_2, s \rangle \Rightarrow \langle S_2, s' \rangle$
- $\langle S_2, s' \rangle \Rightarrow^k s''$

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

- Set $k_1 = 1$, $k_2 = k$ and get:
  - $k+1 = k_1 + k_2$
  - $\langle S_1, s \rangle \Rightarrow^{k1} s'$
  - $\langle S_2, s' \rangle \Rightarrow^{k2} s''$

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

- $\gamma = \langle S'_{1;}\, S_2,\, s' \rangle$
- $\langle S_{1;}\, S_2,\, s \rangle \Rightarrow \langle S'_{1;}\, S_2,\, s' \rangle$ $\quad [\mathrm{comp}^1_{\mathrm{sos}}] \quad \dfrac{\langle S_1,\, s\rangle \Rightarrow \langle S'_1,\, s'\rangle}{\langle S_1;S_2,\, s\rangle \Rightarrow \langle S'_1;S_2,\, s'\rangle}$
- $\langle S'_{1;}\, S_2,\, s' \rangle \Rightarrow^k s''$
- 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:
  - $k = j_1 + j_2$
  - $\langle S'_1,\, s' \rangle \Rightarrow^{j1} \hat{s}$
  - $\langle S_2,\, \hat{s} \rangle \Rightarrow^{j2} s''$
- Set $k_1 = j_1 + 1$, $k_2 = j_2$ and get:
  - $k+1 = k_1 + k_2$
  - $\langle S_1,\, s \rangle \Rightarrow^{k1} \hat{s}$ since $\langle S_1\, ,\, s \rangle \Rightarrow \langle S'_1\, ,\, s' \rangle \Rightarrow^{j1} \hat{s}$
  - $\langle S_2,\, \hat{s} \rangle \Rightarrow^{k2} s''$

Exam Question (2017-A-Q3)

נתונות התוכניות הבאות בשפת While, כאשר b הוא ביטוי בוליאני כלשהו ו S,Q פקודות כלשהן:

P_1 = (while b do S)

P_2 = (while b do S) ; (if b then Q else skip)

א. האם התוכניות שקולות ב Natural Operational Semantics ? הוכיחי או תני דוגמה נגדית.

ב. האם התוכניות שקולות ב Structural Operational Semantics ? הוכיחי או תני דוגמה נגדית.