Concepts in Programming Languages

Recitation 2: Natural Operational Semantics

Yotam Feldman

Reference:

Semantics with Applications by H. Nielson and F. Nielson – Ch. 2

http://www.cs.kun.nl/~hubbers/courses/sc_1718/materiaal/wiley.pdf

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
  - Arithmetic and Boolean expressions imported from usual definitions

Example While Program

```
y := 1;
while ¬(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, …}
  - Truth values T={ff, tt}
  - States State = Var $\rightarrow$ N
  - Lookup in a state s: s x
  - Update of a state s: s [x $\mapsto$ 5]

Example State Manipulations

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

Semantics of arithmetic expressions

- Arithmetic expressions are side-effect free (unless otherwise stated)
- A⟦ Aexp ⟧ : State $\rightarrow$ N
- Defined by structural induction on the syntax tree
- Well-defined: defined over syntax trees
  (grammar should be unambiguous, expression $\mapsto$ unique 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:

$$\mathcal{B}[\![\texttt{true}]\!]s = \mathbf{tt}$$

$$\mathcal{B}[\![\texttt{false}]\!]s = \mathbf{ff}$$

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

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

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

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

Example: Semantics of Expressions

- $A[\![x+3]\!][x \mapsto 1, y \mapsto 7]$

  $= A[\![x]\!][x \mapsto 1, y \mapsto 7] + A[\![3]\!][x \mapsto 1, y \mapsto 7]$

  $= 1 + 3 = 4$

- $B[\![x+3 > y]\!][x \mapsto 1, y \mapsto 7] =$ **ff**, since

  $A[\![x+3]\!][x \mapsto 1, y \mapsto 7] = 4$

  $A[\![y]\!][x \mapsto 1, y \mapsto 7] = 7$

  and $4 > 7$ if false

Natural Operational Semantics

- Notations:
  - S – program in the While language
  - s, s’ – states
- <S, s> $\rightarrow$ 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 $\rightarrow$

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

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

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

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

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

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

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

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

Derivation Trees

- A derivation tree is a way to write applications of inference rules
- A derivation tree is a "proof" that <S, s> $\rightarrow$s'
- The root of tree is <S, s> $\rightarrow$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

$$\textbf{comp}_{ns}\ \dfrac{\dfrac{}{\langle x := x+1,\ s_0\rangle \rightarrow s_0[x \mapsto 1]}\ \textbf{ass}_{ns} \qquad \dfrac{}{\langle \textbf{if } x > 0 \textbf{ then } y := 2 \textbf{ else } y := 3,\ s_0[x \mapsto 1]\rangle \rightarrow s_0[x \mapsto 1][y \mapsto 2]}\ \textbf{if}_{ns}^{tt}}{\langle x := x+1 ;\ \textbf{if } x > 0 \textbf{ then } y := 2 \textbf{ else } y := 3,\ s_0\rangle \rightarrow s_0[x \mapsto 1][y \mapsto 2]}\ \textbf{ass}_{ns}$$

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

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

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

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

Bad Derivation Tree

$$\textbf{ass}_{ns}\ \frac{}{\langle y := 3,\ s_0[x \mapsto 1]\rangle \rightarrow s_0[x \mapsto 1][y \mapsto 3]}$$

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

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

semantic condition not satisfied:

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

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

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

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

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

Adding Semantics of 3-Way Case

Add semantics of:

$$\text{case } (b_1: S_1),(b_2: S_2),(\text{else}: S_3)$$

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

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

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

Adding Semantics of Conditions with Side Effects

Add semantics of:

$$\text{if } (x:=e) \text{ then } S_1 \text{ else } S_2$$

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

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

Proving Properties of NOS

- Theorem: NOS of While is deterministic:
  $\forall S,s.\ \langle S, s\rangle \rightarrow s' \wedge \langle S, s\rangle \rightarrow s'' \Rightarrow s' = s''$

- Proof:
  Induction over the derivation tree of $\langle S, s\rangle \rightarrow s'$
  (full proof: Nielson & Nielson p. 29)

Determinism of NOS: Induction Base

- $\forall S,s.\ \langle S, s\rangle \rightarrow s' \wedge \langle S, s\rangle \rightarrow s'' \Rightarrow s' = s''$
- <u>Base:</u>
  - Assignment:

$$\mathbf{ass_{ns}} \quad \frac{}{\langle x := a, s\rangle \rightarrow s[x \mapsto \mathbf{A}[\![x]\!]s]}$$

s'

This is the only rule that produces $\langle S, s\rangle \rightarrow s''$ is $\mathbf{ass_{ns}}$ so this is also s''.

  - Skip: similar in spirit.

Determinism of NOS: Induction Step

- $\forall S,s.\ \langle S, s\rangle \rightarrow s' \wedge \langle S, s\rangle \rightarrow s'' \Rightarrow s' = s''$
- <u>Step:</u>
  - …
  - **while**$_{ns}^{tt}$:

$$\frac{\langle S, s\rangle \rightarrow \hat{s}_0 \qquad \langle \text{while b do S}, \hat{s}_0\rangle \rightarrow s'}{\langle \text{while b do S}, s\rangle \rightarrow s'} \quad \mathcal{B}[b]\ s = \mathbf{tt}$$

Since $\mathcal{B}[b]\ s = \mathbf{tt}$, the only rule that can produce $\langle \text{while b do S}, s\rangle \rightarrow s''$ is **while**$_{ns}^{tt}$:

$$\frac{\langle S, s\rangle \rightarrow \hat{s}_1 \qquad \langle \text{while b do S}, \hat{s}_1\rangle \rightarrow s''}{\langle \text{while b do S}, s\rangle \rightarrow s''} \quad \mathcal{B}[b]\ s = \mathbf{tt}$$

From the induction hypothesis $s_0 = s_1$

and then again, on the second premise (from deriving s') we get s'=s''

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:

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

- Examples:
  - S ; skip ≈ S ?
  - skip ; S ≈ S ?
  - S1 ; S2 ≈ S2 ; S1 ?
  - ((S1 ; S2) ; S3) ≈ (S1 ;(S2 ; S3)) ?
  - x := 2 ; y := x + 3 ≈ y := 5; x := 2 ?
  - while b do S ≈ if b then (S ; while b do S) else skip ?
  - while true do x:=x+1 ≈ while true do x:=x+2 ?
  - while x > 1 do skip ≈ while x > 2 do skip ?

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:

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

- 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 ; skip ≈ S

- To prove semantic equivalence:
  1. Assume <S, s> → s' , show <S;skip, s> → s'
  2. Assume <S;skip, s> → s' , show <S, s> → s'

1.

$$\frac{T}{\langle S, s\rangle \to s'} \quad \Rightarrow \quad \frac{\dfrac{T}{\langle S, s\rangle \to s'} \qquad \dfrac{}{\langle skip, s'\rangle \to s'}\ \textbf{skip}_{ns}}{\langle S;skip, s\rangle \to s'}\ \textbf{comp}_{ns}$$

Example: S ; skip ≈ S

- To prove semantic equivalence:
  1. Assume <S, s> → s' , show <S;skip, s> → s'
  2. Assume <S;skip, s> → s' , show <S, s> → s'

2.

$$\dfrac{\dfrac{T}{\langle S, s\rangle \to \hat{s}} \qquad \dfrac{}{\langle skip, \hat{s} > \to s'}\ \textbf{skip}_{ns}}{\langle S;skip, s\rangle \to s'}\ \textbf{comp}_{ns}$$

Example: S ; skip ≈ S

- To prove semantic equivalence:
  1. Assume <S, s> → s' , show <S;skip, s> → s'
  2. Assume <S;skip, s> → s' , show <S, s> → s'

2.