Concepts in Programming Languages

Recitation 3: Structural 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**

Natural Operational Semantics: Formally Defining $\rightarrow$

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

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

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

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

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

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

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

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

Semantic Equivalence in NOS

- 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 ?

Example: Loop Unrolling

Prove: while b do S $\approx$ if b then (S ; while b do S) else skip

W

1. Assume <W, s> $\rightarrow$ s'' , show < if b then (S ; W) else skip, s> $\rightarrow$ s''
   Two cases:

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

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

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

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

s = s''

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

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

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

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

$$\frac{}{\langle \text{skip}, s\rangle \to s}\ \textbf{skip}_{ns}$$

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

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

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

Case $\textbf{while}_{ns}^{tt}$

$$\frac{T_1 \qquad T_2}{\dfrac{\langle S, s\rangle \to s' \qquad \langle \text{while b do S}, s'\rangle \to s''}{\langle \text{while b do S}, s\rangle \to s''}\ \textbf{while}_{ns}^{tt}}$$

$\mathcal{B}[\![b]\!]\, s = \textbf{tt}$

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

Direction 2

2. Assume < if b then (S ; W) else skip, s> $\rightarrow$ s'', show <W, s> $\rightarrow$ s''
   Two cases:

$$\dfrac{\dfrac{\dfrac{T_3}{\langle S, s\rangle \rightarrow s'} \quad \dfrac{T_4}{\langle W, s'\rangle \rightarrow s''}}{\langle S;W, s\rangle \rightarrow s''}\ \mathbf{comp}_{ns}}{\langle \text{if b then (S;W) else skip}, s\rangle \rightarrow s''}\ \mathbf{if}_{ns}^{tt}$$

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

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

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

s = s''

Case $\mathbf{if}_{ns}^{ff}$

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

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

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

s = s''

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

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

$$\dfrac{\dfrac{\dfrac{T_3}{\langle S, s\rangle \to s'} \quad \dfrac{T_4}{\langle W, s'\rangle \to s''}}{\langle S;W, s\rangle \to s''}\ \mathbf{comp}_{ns}}{\langle \text{if } b \text{ then } (S;W) \text{ else skip}, s\rangle \to s''}\ \mathbf{if}_{ns}^{tt}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Adding Semantics of 3-Way Case

Add semantics of:

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

case_{ss}^{1} $\quad \langle \text{case } (b_1: S_1),(b_2: S_2),(\text{else}: S_3) , s\rangle \Rightarrow \langle S_1, s\rangle$ if $\mathbf{B}[\![b_1]\!]s = \mathbf{tt}$

case_{ss}^{2} $\quad \langle \text{case } (b_1: S_1),(b_2: S_2),(\text{else}: S_3) , s\rangle \Rightarrow \langle S_2, s\rangle$ if $\mathbf{B}[\![b_1]\!]s = \mathbf{ff}$ and $\mathbf{B}[\![b_2]\!]s = \mathbf{tt}$

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

Adding Semantics of Conditions with Side Effects

Add semantics of:

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

$$\langle \text{if } (x:=e) \text{ then } S_1 \text{ else } S_2, s\rangle \Rightarrow$$

$$\langle x:=e; \text{ if } x=0 \text{ then } S_1 \text{ else } S_2, s\rangle$$

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>

Example: Non-Termination

<while x ≠ y do (y := y+1), s_0[y ↦ 1]> ⇒

<if x ≠ y then (y := y + 1; while x ≠ y do (y := y+1)) else skip, s_0[y ↦ 1]> ⇒

<y := y + 1; while x ≠ y do (y := y+1), s_0[y ↦ 1]> ⇒

<while x ≠ y do (y := y+1), s_0[y ↦ 2]> ⇒

<if x ≠ y then (y := y + 1; while x ≠ y do (y := y+1)) else skip, s_0[y ↦ 2]> ⇒

<y := y + 1; while x ≠ y do (y := y+1), s_0[y ↦ 2]> ⇒

<while x ≠ y do (y := y+1), s_0[y ↦ 2]> ⇒

…

Semantic Equivalence in SOS

- Two statements S_1 and S_2 are **semantically equivalent in SOS** ($S_1 \approx S_2$) if for any states s and configuration γ we have

$$\langle S_1, s\rangle \Rightarrow \gamma \quad \textbf{iff} \quad \langle S_2, s\rangle \Rightarrow \gamma$$

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 $\langle S_2, s\rangle$

Example: x:=a; x:=3 ≈ x:=3

- $\langle x := 3, s\rangle \Rightarrow s[x \mapsto 3]\rangle$

- $\langle x := d; x := 3, s\rangle$

  $\Rightarrow \langle x := 3, s[x \mapsto \mathbf{A}[\![d]\!]s]\rangle$

  $\Rightarrow s[x \mapsto \mathbf{A}[\![d]\!]s][x \mapsto (\mathbf{A}[\![3]\!](s[x \mapsto \mathbf{A}[\![d]\!]s]))] = s[x \mapsto 3]$

Example: S; skip ≈ S

From rhs to lhs:

- Assume <S, s> $\Rightarrow^*$ γ where γ is final or stuck.
  Show that <S; skip, s> $\Rightarrow^*$ γ.
  - In the basic While language there are no stuck configurations, so we can consider just γ=s'.
  - ?

- (Also nonterminating computations)

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

Sequential Composition – Property 1

- Claim: $\langle S_1, s\rangle \Rightarrow^k s'$ implies $\langle S_1; S_2, s\rangle \Rightarrow^k \langle S_2, s'\rangle$
- Proof: By induction over k.
  - Base case, k=0: holds vacuously

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

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

Inductive Step

- **Assume for all $j \leq k$**: if $\langle S_1, s\rangle \Rightarrow^j s'$ then $\langle S_1; S_2, s\rangle \Rightarrow^j \langle S_2, s'\rangle$. Prove for k+1.
- Assume $\langle S_1, s\rangle \Rightarrow^{k+1} s''$, then, by definition of deriv. seq. :
  - $\langle S_1, s\rangle \Rightarrow \gamma \Rightarrow^k s''$
  - $\gamma = \langle S'_1, s'\rangle$ is not a final or stuck configuration.
- By the induction hypothesis, $\langle S'_1; S_2, s'\rangle \Rightarrow^k \langle S_2, s''\rangle$
- $\langle S_1, s\rangle \Rightarrow \langle S'_1, s'\rangle$, so by [comp^1] we get $\langle S_1; S_2, s\rangle \Rightarrow \langle S'_1; S_2, s'\rangle$
- Together we get $\langle S_1; S_2, s\rangle \Rightarrow^{k+1} \langle S_2, s''\rangle$

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

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

S; skip ≈ S

From rhs to lhs:

- **Final:** Assume <S, s> $\Rightarrow^*$ s'.
  From the previous claim, <S; skip, s> $\Rightarrow^*$ <skip, s'>.
  <skip, s'> $\Rightarrow$ s', so overall <S; skip, s> $\Rightarrow^*$s'.
- **Non-terminating:**

  <S, s> $\Rightarrow$ <S', s'> $\Rightarrow$ <S'', s''> $\Rightarrow$ …

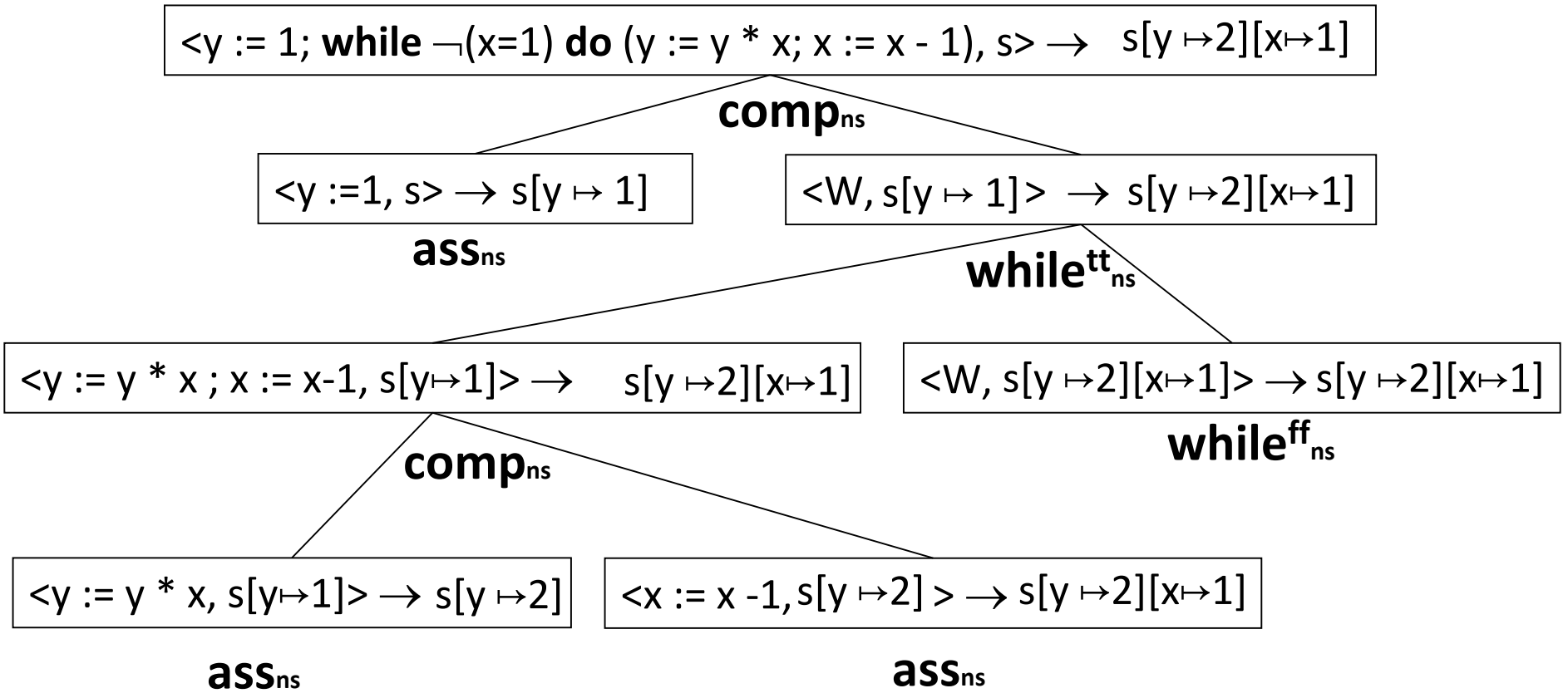
  similarly to the proof of the previous claim,
  <S; skip, s> $\Rightarrow$ <S'; skip, s'> $\Rightarrow$ <S''; skip, s''> $\Rightarrow$ …
  (by induction on the length of the prefix)

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



From NOS to an interpreter

- Implement semantics of arithmetic and Boolean expressions
- Implement NOS as a function:
  - input: S, s
  - output: s' such that <S, s> $\rightarrow$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

CODE DEMO

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