Program Analysis, University of Stuttgart, Winter 2023/24

Exercise 1: Operational Semantics

Deadline for uploading solutions via Ilias:
November 2, 2023, 11:59pm Stuttgart time

Consider the following SIMP program P:

```
y := 2; while !x > 3 do if ¬ (!x = !y) then x := !x - 2 else x := !x - 1
```

and an initial store:

$$s = \{x \mapsto 4\}$$

Your task is to evaluate this program using the abstract machine, small-step operational semantics, and big-step operational semantics, as introduced in the lecture. As a reference, see the rules and axioms provided in the appendix (copied from Fernandez' book).

To ease the presentation of your solution, please use the following abbreviations when referring to parts of the program:

Abbrevation	Code
W	`while !x > 3 do if ¬ (!x = !y) then x := !x - 2 else x := !x - 1`
C_1	`!x > 3`
F	`if ¬ (!x = !y) then x := !x - 2 else x := !x - 1`
C_2	`¬ (!x = !y)`
T	`x := !x - 2`
E	`x := !x - 1`

For each task, we provide a template to fill in your solution. The correct solutions fit into the template and should align with those parts of the solution that we provide.

Hint: Sketch your solution on scratch paper before filling it into the template.

1 Abstract Machine [30 points]

Provide the semantics of the program as a sequence of transitions of the abstract machine for SIMP. Use the following template, where each line corresponds to a new configuration.

	Program	Stack	Store
$\rightarrow$	$P \circ nil$	nil	$\{x \mapsto 4\}$
$\rightarrow$			
$\rightarrow$			
$\rightarrow$			
$\rightarrow$			
$\rightarrow$	$C_1 \circ \texttt{while} \circ nil$	$C_1 \circ F \circ nil$	$\{x \mapsto 4, y \mapsto 2\}$
$\rightarrow$			
$\rightarrow$			
$\rightarrow$			
$\rightarrow$			
$\rightarrow$			
$\rightarrow$	$C_2 \circ \texttt{if} \circ W \circ nil$	$T \circ E \circ nil$	$\{x \mapsto 4, y \mapsto 2\}$
	Recall C_2: ¬ `(!x = !y)`		
$\rightarrow$			
$\rightarrow$			
$\rightarrow$			
$\rightarrow$			
$\rightarrow$			
$\rightarrow$			
$\rightarrow$	$T \circ W \circ nil$	nil	$\{x \mapsto 4, y \mapsto 2\}$
	Recall T: `x := !x - 2`		

$\rightarrow$			
$\rightarrow$			
$\rightarrow$			
$\rightarrow$			
$\rightarrow$			
$\rightarrow$			
$\rightarrow$	$C_1 \circ \texttt{while} \circ nil$	$C_1 \circ F \circ nil$	$\{x \mapsto 2, y \mapsto 2\}$
	Recall C_1: `!x > 3`		
$\rightarrow$			
$\rightarrow$			
$\rightarrow$			
$\rightarrow$			
$\rightarrow$	nil	nil	$\{x \mapsto 2, y \mapsto 2\}$

2 Small-Step Semantics [40 points]

Provide the semantics of the program as an evaluation sequence using the small-step semantics of SIMP. Use the following template to provide your solution. For each transition, indicate above the arrow why you can take the transition. Specifically:

- On top of each transition, indicate the rule or axiom that you use to take the transition. Note that you have to indicate the name of the reduction axiom or rule at the bottom of the proof tree. For example, if you resolve a variable in a statement, which is one of two statements in a sequence, you have to indicate the sequence rule, not the variable rule.
- We do not ask you to provide the proof trees for all transitions, but please provide the proof tree of the first transition enabled the *if* reduction rule. Use the template at the end of the question to provide this proof tree. We provide the proof tree for the first transition enabled by the *seq* reduction rule as an example. It is strongly recommended to at least sketch the proof trees of all transitions for yourself, because it helps understand what transition steps you can(not) take.

Hint 1: The transitions used in the correct solution should contain 3 uses of axioms and 16 uses of rules.

Hint 2: Each transition is enabled either by a rule or an axiom. In case of an axiom, the proof tree is flat, i.e., without nothing on top of the horizontal line. In case of a rule, the proof tree has at least one horizontal line with a hypothesis on top (see the example of the first usage of *seq* below).

$\langle P, \{x \mapsto 4\}\rangle$

$\longrightarrow$ ______________________________

$\longrightarrow$ ______________________________

$\longrightarrow$ ______________________________

$\longrightarrow$ ______________________________

$\longrightarrow$ ______________________________

$\xrightarrow{(if_T)}$ $\langle$ F; `while` C_1 `do` F, $\{x \mapsto 4, y \mapsto 2\}\rangle$

Recall F represents... `if` $\neg$ (!x = !y) `then` T `else` E

$\longrightarrow$ ______________________________

$\longrightarrow$ ______________________________

$\longrightarrow$ ______________________________

$\longrightarrow$ ______________________________

$\xrightarrow{(seq)}$ $\langle$ T; `while` C_1 `do` F, $\{x \mapsto 4, y \mapsto 2\}\rangle$

Recall T represents... `x := !x - 2`

$\longrightarrow$

$\longrightarrow$

$\longrightarrow$

$\longrightarrow$

$\xrightarrow{(while)}$ ⟨ `if` C_1 `then` (F; `while` C_1 `do` F) else *skip*, $\{x \mapsto 2, y \mapsto 2\}\rangle$
Recall C_1 represents... `!x > 3`

$\longrightarrow$

$\longrightarrow$

$\longrightarrow$

Proof trees (s stands for $\{x \mapsto 4\}$ and s' stands for $\{x \mapsto 4, y \mapsto 2\}$):

- Proof tree for first use of *seq* rule:

$$\dfrac{\dfrac{}{\langle y := 2, s\rangle \rightarrow \langle skip, s'\rangle}\ (:=)}{\langle y := 2; W, s\rangle \rightarrow \langle skip; W, s'\rangle}\ (seq)$$

- Proof tree for first use of *if* rule:

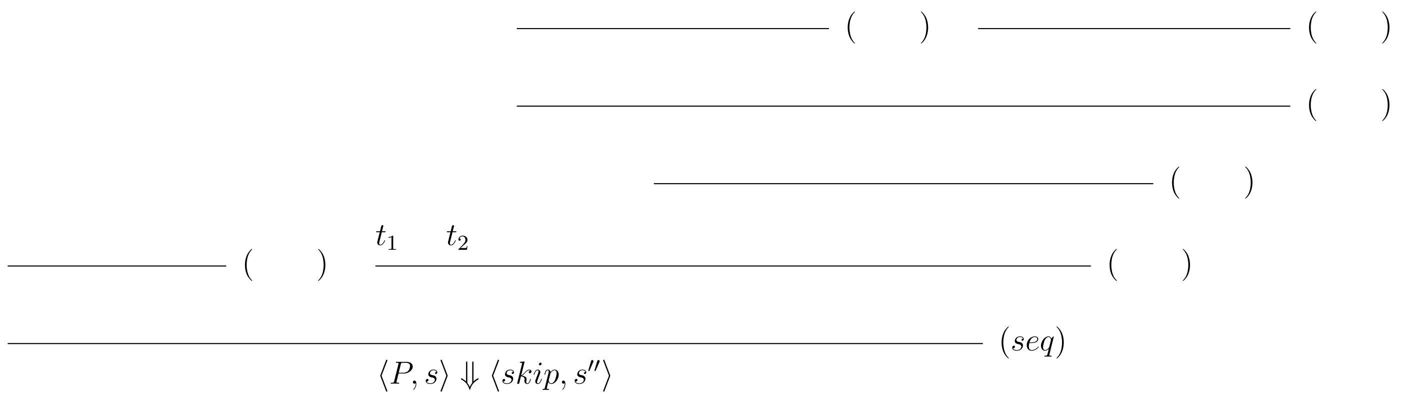
$$\dfrac{\dfrac{\dfrac{}{\qquad\qquad\qquad}\ (\qquad)}{\qquad\qquad\qquad}\ (\qquad)}{\qquad\qquad\qquad\qquad}\ (\qquad)$$

3 Big-Step Semantics [30 points]

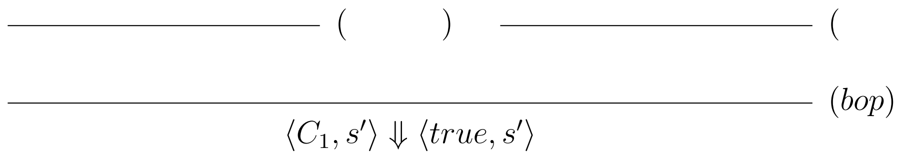
Give the semantics of the program as a proof tree based on big-step operational semantics. Use the template to provide your solution. For each rule or axiom, indicate the name of it, as given in the appendix. To save space, use the following table to abbreviate different stores.
Hint: You will need only three different stores.

Abbreviation	Store
s	$\{x \mapsto 4\}$
s'	
s"	

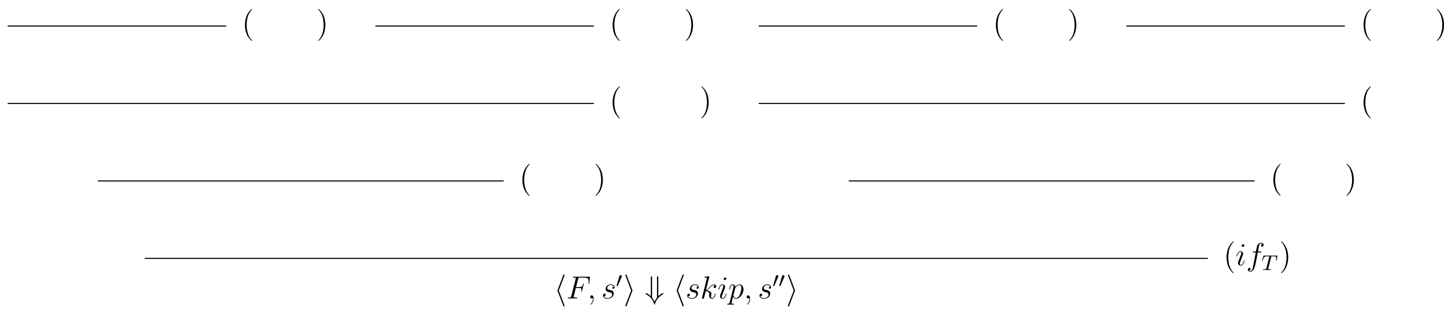
Main proof tree:



Subtree t_1:



Subtree t_2:



Appendix 1: Rules of SIMP Abstract Machine

1. Evaluation of Expressions:

$$
\begin{array}{rcll}
\langle n \cdot c, r, m\rangle & \to & \langle c, n \cdot r, m\rangle & \\
\langle b \cdot c, r, m\rangle & \to & \langle c, b \cdot r, m\rangle & \\
\\
\langle \neg B \cdot c, r, m\rangle & \to & \langle B \cdot \neg \cdot c, r, m\rangle & \\
\langle (B_1 \ \wedge \ B_2) \cdot c, r, m\rangle & \to & \langle B_1 \cdot B_2 \cdot \wedge \cdot c, r, m\rangle & \\
\langle \neg \cdot c, b \cdot r, m\rangle & \to & \langle c, b' \cdot r, m\rangle & \text{if } b' = not\ b \\
\langle \wedge \cdot c, b_2 \cdot b_1 \cdot r, m\rangle & \to & \langle c, b \cdot r, m\rangle & \text{if } b_1\ and\ b_2 = b \\
\\
\langle (E_1\ op\ E_2) \cdot c, r, m\rangle & \to & \langle E_1 \cdot E_2 \cdot op \cdot c, r, m\rangle & \\
\langle (E_1\ bop\ E_2) \cdot c, r, m\rangle & \to & \langle E_1 \cdot E_2 \cdot bop \cdot c, r, m\rangle & \\
\langle op \cdot c, n_2 \cdot n_1 \cdot r, m\rangle & \to & \langle c, n \cdot r, m\rangle & \text{if } n_1\ op\ n_2 = n \\
\langle bop \cdot c, n_2 \cdot n_1 \cdot r, m\rangle & \to & \langle c, b \cdot r, m\rangle & \text{if } n_1\ bop\ n_2 = b \\
\\
\langle !l \cdot c, r, m\rangle & \to & \langle c, n \cdot r, m\rangle & \text{if } m(l) = n
\end{array}
$$

2. Evaluation of Commands:

$$
\begin{array}{rcl}
\langle skip \cdot c, r, m\rangle & \to & \langle c, r, m\rangle \\
\\
\langle (l := E) \cdot c, r, m\rangle & \to & \langle E \cdot := \cdot c, l \cdot r, m\rangle \\
\langle := \cdot c, n \cdot l \cdot r, m\rangle & \to & \langle c, r, m[l \mapsto n]\rangle \\
\\
\langle (C_1; C_2) \cdot c, r, m\rangle & \to & \langle C_1 \cdot C_2 \cdot c, r, m\rangle \\
\\
\langle (if\ B\ then\ C_1\ else\ C_2) \cdot c, r, m\rangle & \to & \langle B \cdot if \cdot c, C_1 \cdot C_2 \cdot r, m\rangle \\
\langle if \cdot c, True \cdot C_1 \cdot C_2 \cdot r, m\rangle & \to & \langle C_1 \cdot c, r, m\rangle \\
\langle if \cdot c, False \cdot C_1 \cdot C_2 \cdot r, m\rangle & \to & \langle C_2 \cdot c, r, m\rangle \\
\\
\langle (while\ B\ do\ C) \cdot c, r, m\rangle & \to & \langle B \cdot while \cdot c, B \cdot C \cdot r, m\rangle \\
\langle while \cdot c, True \cdot B \cdot C \cdot r, m\rangle & \to & \langle C \cdot (while\ B\ do\ C) \cdot c, r, m\rangle \\
\langle while \cdot c, False \cdot B \cdot C \cdot r, m\rangle & \to & \langle c, r, m\rangle
\end{array}
$$

(Copied from *Programming Languages and Operational Semantics* by Maribel Fernandez.)

Appendix 2: Rules and Axioms of Small-Step Semantics

Reduction Semantics of Expressions:

$$\frac{}{\langle !l, s\rangle \to \langle n, s\rangle \quad \text{if } s(l) = n}\ (\mathsf{var})$$

$$\frac{}{\langle n_1\ op\ n_2, s\rangle \to \langle n, s\rangle \quad \text{if } n = (n_1\ op\ n_2)}\ (\mathsf{op})$$

$$\frac{}{\langle n_1\ bop\ n_2, s\rangle \to \langle b, s\rangle \quad \text{if } b = (n_1\ bop\ n_2)}\ (\mathsf{bop})$$

$$\frac{\langle E_1, s\rangle \to \langle E_1', s'\rangle}{\langle E_1 op E_2, s\rangle \to \langle E_1' op E_2, s'\rangle}\ (\mathsf{op_L}) \qquad \frac{\langle E_2, s\rangle \to \langle E_2', s'\rangle}{\langle n_1 op E_2, s\rangle \to \langle n_1 op E_2', s'\rangle}\ (\mathsf{op_R})$$

$$\frac{\langle E_1, s\rangle \to \langle E_1', s'\rangle}{\langle E_1 bop E_2, s\rangle \to \langle E_1' bop E_2, s'\rangle}\ (\mathsf{bop_L}) \qquad \frac{\langle E_2, s\rangle \to \langle E_2', s'\rangle}{\langle n_1 bop E_2, s\rangle \to \langle n_1 bop E_2', s'\rangle}\ (\mathsf{bop_R})$$

$$\frac{}{\langle b_1 \wedge b_2, s\rangle \to \langle b, s\rangle \quad \text{if } b = (b_1\ and\ b_2)}\ (\mathsf{and})$$

$$\frac{}{\langle \neg b, s\rangle \to \langle b', s\rangle \quad \text{if } b' = not\ b}\ (\mathsf{not}) \qquad \frac{\langle B_1, s\rangle \to \langle B_1', s'\rangle}{\langle \neg B_1, s\rangle \to \langle \neg B_1', s'\rangle}\ (\mathsf{notArg})$$

$$\frac{\langle B_1, s\rangle \to \langle B_1', s'\rangle}{\langle B_1 \wedge B_2, s\rangle \to \langle B_1' \wedge B_2, s'\rangle}\ (\mathsf{and_L}) \qquad \frac{\langle B_2, s\rangle \to \langle B_2', s'\rangle}{\langle b_1 \wedge B_2, s\rangle \to \langle b_1 \wedge B_2', s'\rangle}\ (\mathsf{and_R})$$

Reduction Semantics of Commands:

$$\frac{\langle E, s\rangle \to \langle E', s'\rangle}{\langle l := E, s\rangle \to \langle l := E', s'\rangle}\ (:=_{\mathsf{R}}) \qquad \frac{}{\langle l := n, s\rangle \to \langle skip, s[l \mapsto n]\rangle}\ (:=)$$

$$\frac{\langle C_1, s\rangle \to \langle C_1', s'\rangle}{\langle C_1; C_2, s\rangle \to \langle C_1'; C_2, s'\rangle}\ (\mathsf{seq}) \qquad \frac{}{\langle skip; C, s\rangle \to \langle C, s\rangle}\ (\mathsf{skip})$$

$$\frac{\langle B, s\rangle \to \langle B', s'\rangle}{\langle if\, B\ then\ C_1\ else\ C_2, s\rangle \to \langle if\, B'\ then\ C_1\ else\ C_2, s'\rangle}\ (\mathsf{if})$$

$$\frac{}{\langle if\, True\ then\ C_1\ else\ C_2, s\rangle \to \langle C_1, s\rangle}\ (\mathsf{if_T})$$

$$\frac{}{\langle if\, False\ then\ C_1\ else\ C_2, s\rangle \to \langle C_2, s\rangle}\ (\mathsf{if_F})$$

$$\frac{}{\langle while\ B\ do\ C, s\rangle \to \langle if\, B\ then\ (C; while\ B\ do\ C)\ else\ skip, s\rangle}\ (\mathsf{while})$$

(Copied from *Programming Languages and Operational Semantics* by Maribel Fernandez.)

Appendix 3: Rules and Axioms of Big-Step Semantics

$$\frac{}{\langle c, s\rangle \Downarrow \langle c, s\rangle \quad \text{if } c \in Z \cup \{True, False\}} \text{(const)}$$

$$\frac{}{\langle !l, s\rangle \Downarrow \langle n, s\rangle \quad \text{if } s(l) = n} \text{(var)}$$

$$\frac{\langle B_1, s\rangle \Downarrow \langle b_1, s'\rangle \quad \langle B_2, s'\rangle \Downarrow \langle b_2, s''\rangle}{\langle B_1 \,\wedge\, B_2, s\rangle \Downarrow \langle b, s''\rangle \quad \text{if } b = b_1 \, and \, b_2} \text{(and)}$$

$$\frac{\langle B_1, s\rangle \Downarrow \langle b_1, s'\rangle}{\langle \neg B_1, s\rangle \Downarrow \langle b, s'\rangle \quad \text{if } b = not\ b_1} \text{(not)}$$

$$\frac{\langle E_1, s\rangle \Downarrow \langle n_1, s'\rangle \quad \langle E_2, s'\rangle \Downarrow \langle n_2, s''\rangle}{\langle E_1 \, op \, E_2, s\rangle \Downarrow \langle n, s''\rangle \quad \text{if } n = n_1 \, op \, n_2} \text{(op)}$$

$$\frac{\langle E_1, s\rangle \Downarrow \langle n_1, s'\rangle \quad \langle E_2, s'\rangle \Downarrow \langle n_2, s''\rangle}{\langle E_1 \, bop \, E_2, s\rangle \Downarrow \langle b, s''\rangle \quad \text{if } b = n_1 \, bop \, n_2} \text{(bop)}$$

$$\frac{}{\langle skip, s\rangle \Downarrow \langle skip, s\rangle} \text{(skip)} \qquad \frac{\langle E, s\rangle \Downarrow \langle n, s'\rangle}{\langle l := E, s\rangle \Downarrow \langle skip, s'[l \mapsto n]\rangle} (:=)$$

$$\frac{\langle C_1, s\rangle \Downarrow \langle skip, s'\rangle \quad \langle C_2, s'\rangle \Downarrow \langle skip, s''\rangle}{\langle C_1; C_2, s\rangle \Downarrow \langle skip, s''\rangle} \text{(seq)}$$

$$\frac{\langle B, s\rangle \Downarrow \langle True, s'\rangle \quad \langle C_1, s'\rangle \Downarrow \langle skip, s''\rangle}{\langle if\, B \; then \; C_1 \; else \; C_2, s\rangle \Downarrow \langle skip, s''\rangle} (\text{if}_\text{T})$$

$$\frac{\langle B, s\rangle \Downarrow \langle False, s'\rangle \quad \langle C_2, s'\rangle \Downarrow \langle skip, s''\rangle}{\langle if\, B \; then \; C_1 \; else \; C_2, s\rangle \Downarrow \langle skip, s''\rangle} (\text{if}_\text{F})$$

$$\frac{\langle B, s\rangle \Downarrow \langle True, s_1\rangle \ \langle C, s_1\rangle \Downarrow \langle skip, s_2\rangle \ \langle while \; B \; do \; C, s_2\rangle \Downarrow \langle skip, s_3\rangle}{\langle while \; B \; do \; C, s\rangle \Downarrow \langle skip, s_3\rangle} (\text{while}_\text{T})$$

$$\frac{\langle B, s\rangle \Downarrow \langle False, s'\rangle}{\langle while \; B \; do \; C, s\rangle \Downarrow \langle skip, s'\rangle} (\text{while}_\text{F})$$

(Copied from *Programming Languages and Operational Semantics* by Maribel Fernandez.)