Program Analysis, University of Stuttgart, Winter 2020/21

Exercise 1: Operational Semantics
—**Solution**—

Deadline for uploading solutions via Ilias:
November 20, 2020, 11:59pm Stuttgart time

Consider the following SIMP program:

```
x := 7; while !y < !x do x := !x - !y
```

and an initial store:

$$s = \{y \mapsto 5\}$$

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
P	`x := 7; while !y < !x do x:= !x - !y`
W	`while !y < !x do x:= !x - !y`
B	`!y < !x`
C_1	`x := 7`
C_2	`x:= !x - !y`

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.

Solution:

$$
\begin{aligned}
& \langle P \circ nil, nil, \{y \mapsto 5\}\rangle \\
\to & \langle C_1 \circ W \circ nil, nil, \{y \mapsto 5\}\rangle \\
\to & \langle\ \texttt{7} \circ \texttt{:=} \circ W \circ nil,\ \texttt{x} \circ nil, \{y \mapsto 5\}\rangle \\
\to & \langle\ \texttt{:=} \circ W \circ\ nil, 7 \circ \texttt{x} \circ nil, \{y \mapsto 5\}\rangle \\
\to & \langle W \circ\ nil, nil, \{x \mapsto 7, y \mapsto 5\}\rangle \\
\to & \langle B \circ \texttt{while} \circ nil, B \circ C_2 \circ\ nil,\ \{x \mapsto 7, y \mapsto 5\}\rangle \\
\to & \langle\ \texttt{!y} \circ \texttt{!x} \circ \texttt{<} \circ \texttt{while} \circ nil, B \circ C_2 \circ\ nil,\ \{x \mapsto 7, y \mapsto 5\}\rangle \\
\to & \langle\ \texttt{!x} \circ \texttt{<} \circ \texttt{while} \circ nil, 5\ \circ B \circ C_2 \circ\ nil,\ \{x \mapsto 7, y \mapsto 5\}\rangle \\
\to & \langle\ \texttt{<} \circ \texttt{while} \circ nil, 7\ \circ\ 5\ \circ B \circ C_2 \circ\ nil,\ \{x \mapsto 7, y \mapsto 5\}\rangle \\
\to & \langle\ \texttt{while} \circ nil, True \circ B \circ C_2 \circ\ nil,\ \{x \mapsto 7, y \mapsto 5\}\rangle \\
\to & \langle C_2 \circ W \circ\ nil,\ nil,\ \{x \mapsto 7, y \mapsto 5\}\rangle \\
\to & \langle\ \texttt{!x - !y} \circ \texttt{:=} \circ W \circ\ nil,\ \texttt{x} \circ nil,\ \{x \mapsto 7, y \mapsto 5\}\rangle \\
\to & \langle\ \texttt{!x} \circ \texttt{!y} \circ \texttt{-} \circ \texttt{:=} \circ W \circ\ nil,\ \texttt{x} \circ nil,\ \{x \mapsto 7, y \mapsto 5\}\rangle \\
\to & \langle\ \texttt{!y} \circ \texttt{-} \circ \texttt{:=} \circ W \circ\ nil, 7 \circ \texttt{x} \circ nil,\ \{x \mapsto 7, y \mapsto 5\}\rangle \\
\to & \langle\ \texttt{-} \circ \texttt{:=} \circ W \circ\ nil, 5 \circ 7 \circ \texttt{x} \circ nil,\ \{x \mapsto 7, y \mapsto 5\}\rangle \\
\to & \langle\ \texttt{:=} \circ W \circ\ nil, 2 \circ \texttt{x} \circ nil,\ \{x \mapsto 7, y \mapsto 5\}\rangle \\
\to & \langle W \circ\ nil, nil,\ \{x \mapsto 2, y \mapsto 5\}\rangle \\
\to & \langle B \circ \texttt{while} \circ nil, B \circ C_2 \circ nil,\ \{x \mapsto 2, y \mapsto 5\}\rangle \\
\to & \langle\ \texttt{!y} \circ \texttt{!x} \circ \texttt{<} \circ \texttt{while} \circ nil, B \circ C_2 \circ\ nil,\ \{x \mapsto 2, y \mapsto 5\}\rangle \\
\to & \langle\ \texttt{!x} \circ \texttt{<} \circ \texttt{while} \circ nil, 5 \circ B \circ C_2 \circ\ nil,\ \{x \mapsto 2, y \mapsto 5\}\rangle \\
\to & \langle\ \texttt{<} \circ \texttt{while} \circ nil, 2 \circ 5 \circ B \circ C_2 \circ\ nil,\ \{x \mapsto 2, y \mapsto 5\}\rangle \\
\to & \langle\ \texttt{while} \circ nil, False \circ B \circ C_2 \circ\ nil,\ \{x \mapsto 2, y \mapsto 5\}\rangle \\
\to & \langle nil, nil,\ \{x \mapsto 2, y \mapsto 5\}\rangle
\end{aligned}
$$

2 Small-step Semantics [30 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:

- If the transition is enabled by an axiom, give the name of the axiom.
- If the transition is enabled by a rule, give the name of the rule. You do not need to provide the proof tree for using a rule. However, it is strongly recommended to at least sketch the proof trees for yourself, because it helps understand what transition steps you can(not) take.

Hint: The solution contains 6 uses of axioms and 11 uses of rules.

Solution:

$\langle P, \{y \mapsto 5\}\rangle$
$\xrightarrow{(seq)} \langle\, \texttt{skip;}\ W, \{x \mapsto 7, y \mapsto 5\}\rangle$
$\xrightarrow{(skip)} \langle W, \{x \mapsto 7, y \mapsto 5\}\rangle$
$\xrightarrow{(while)} \langle\, \texttt{if}\ B\ \texttt{then}\ C_2\ ;\ W\ \texttt{else skip}\ , \{x \mapsto 7, y \mapsto 5\}\rangle$
$\xrightarrow{(if)} \langle\, \texttt{if 5 < !x then}\ C_2\ ;\ W\ \texttt{else skip}\ , \{x \mapsto 7, y \mapsto 5\}\rangle$
$\xrightarrow{(if)} \langle\, \texttt{if 5 < 7 then}\ C_2\ ;\ W\ \texttt{else skip}\ , \{x \mapsto 7, y \mapsto 5\}\rangle$
$\xrightarrow{(if)} \langle\, \texttt{if}\ True\ \texttt{then}\ C_2\ ;\ W\ \texttt{else skip}\ , \{x \mapsto 7, y \mapsto 5\}\rangle$
$\xrightarrow{(if_T)} \langle\, C_2\ ;\ W, \{x \mapsto 7, y \mapsto 5\}\rangle$
$\xrightarrow{(seq)} \langle\, \texttt{x := 7 - !y;}\ W, \{x \mapsto 7, y \mapsto 5\}\rangle$
$\xrightarrow{(seq)} \langle\, \texttt{x := 7 - 5;}\ W, \{x \mapsto 7, y \mapsto 5\}\rangle$
$\xrightarrow{(seq)} \langle\, \texttt{x := 2;}\ W, \{x \mapsto 7, y \mapsto 5\}\rangle$
$\xrightarrow{(seq)} \langle\, \texttt{skip;}\ W, \{x \mapsto 2, y \mapsto 5\}\rangle$
$\xrightarrow{(skip)} \langle W, \{x \mapsto 2, y \mapsto 5\}\rangle$
$\xrightarrow{(while)} \langle\, \texttt{if}\ B\ \texttt{then}\ C_2\ ;\ W\ \texttt{else skip}\ , \{x \mapsto 2, y \mapsto 5\}\rangle$
$\xrightarrow{(if)} \langle\, \texttt{if 5 < !x then}\ C_2\ ;\ W\ \texttt{else skip}\ , \{x \mapsto 2, y \mapsto 5\}\rangle$
$\xrightarrow{(if)} \langle\, \texttt{if 5 < 2 then}\ C_2\ ;\ W\ \texttt{else skip}\ , \{x \mapsto 2, y \mapsto 5\}\rangle$
$\xrightarrow{(if)} \langle\, \texttt{if}\ False\ \texttt{then}\ C_2\ ;\ W\ \texttt{else skip}\ , \{x \mapsto 2, y \mapsto 5\}\rangle$
$\xrightarrow{(if_F)} \langle\, \texttt{skip}\ , \{x \mapsto 2, y \mapsto 5\}\rangle$

Even though not required for the solution, here are some proof trees that show that the used rules are applicable (s' stands for $\{x \mapsto 7, y \mapsto 5\}$ and s'' stands for $\{x \mapsto 2, y \mapsto 5\}$):

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

$$\dfrac{\dfrac{}{\langle C_1, s\rangle \rightarrow \langle\, \texttt{skip}\ , s'\rangle}\ (:=)}{\langle P, s\rangle \rightarrow \langle\, \texttt{skip;}\ W, s'\rangle}\ (seq)$$

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

$$\dfrac{\dfrac{\dfrac{}{\langle !y, s'\rangle \to \langle 5, s'\rangle}\;(var)}{\langle B, s'\rangle \to \langle 5 \texttt{ < !x }, s'\rangle}\;(bop_L)}{\langle \texttt{ if } B \texttt{ then } C_2 \texttt{ ; } W \texttt{ else skip }, s'\rangle \to \langle \texttt{ if } 5 \texttt{ < !x then } C_2 \texttt{ ; } W \texttt{ else skip }, s'\rangle}\;(if)$$

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

$$\dfrac{\dfrac{\dfrac{}{\langle !x, s'\rangle \to \langle 7, s'\rangle}\;(var)}{\langle 5 \texttt{ < !x }, s'\rangle \to \langle 5 \texttt{ < } 7, s'\rangle}\;(bop_R)}{\langle \texttt{ if } 5 \texttt{ < !x then } C_2 \texttt{ ; } W \texttt{ else skip }, s'\rangle \to \langle \texttt{ if } 5 \texttt{ < } 7 \texttt{ then } C_2 \texttt{ ; } W \texttt{ else skip }, s'\rangle}\;(if)$$

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

$$\dfrac{\dfrac{}{\langle 5 \texttt{ < } 7, s'\rangle \to \langle True, s'\rangle}\;(bop)}{\langle \texttt{ if } 5 \texttt{ < } 7 \texttt{ then } C_2 \texttt{ ; } W \texttt{ else skip }, s'\rangle \to \langle \texttt{ if } True \texttt{ then } C_2 \texttt{ ; } W \texttt{ else skip }, s'\rangle}\;(if)$$

3 Big-step Semantics [40 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	$\{y \mapsto 5\}$
s'	$\{x \mapsto 7, y \mapsto 5\}$
s"	$\{x \mapsto 2, y \mapsto 5\}$

Solution:

Main proof tree:

$$\dfrac{\dfrac{\dfrac{}{\langle 7, s\rangle \Downarrow \langle 7, s\rangle}\,(const)}{\langle C_1, s\rangle \Downarrow \langle skip, s'\rangle}\,(:=) \qquad \dfrac{t_1 \quad t_2 \quad t_3}{\langle W, s'\rangle \Downarrow \langle skip, s''\rangle}\,(while_T)}{\langle P, s\rangle \Downarrow \langle skip, s''\rangle}\,(seq)$$

Subtree t_1:

$$\dfrac{\dfrac{}{\langle !y, s'\rangle \Downarrow \langle 5, s'\rangle}\,(var) \qquad \dfrac{}{\langle !x, s'\rangle \Downarrow \langle 7, s'\rangle}\,(var)}{\langle B, s'\rangle \Downarrow \langle True, s'\rangle}\,(bop)$$

Subtree t_2:

$$\dfrac{\dfrac{\dfrac{}{\langle !x, s'\rangle \Downarrow \langle 7, s'\rangle}\,(var) \qquad \dfrac{}{\langle !y, s'\rangle \Downarrow \langle 5, s'\rangle}\,(var)}{\langle !x - !y, s'\rangle \Downarrow \langle 2, s'\rangle}\,(op)}{\langle C_2, s'\rangle \Downarrow \langle skip, s''\rangle}\,(:=)$$

Subtree t_3:

$$\dfrac{\dfrac{\dfrac{}{\langle !y, s''\rangle \Downarrow \langle 5, s''\rangle}\,(var) \qquad \dfrac{}{\langle !x, s''\rangle \Downarrow \langle 2, s''\rangle}\,(var)}{\langle B, s''\rangle \Downarrow \langle False, s''\rangle}\,(bop)}{\langle W, s''\rangle \Downarrow \langle skip, s''\rangle}\,(while_F)$$

Appendix 1: Rules of SIMP Abstract Machine

1. Evaluation of Expressions:

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

2. Evaluation of Commands:

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

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

Appendix 2: Rules and Axioms of Small-Step Semantics

Reduction Semantics of Expressions:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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