IMP: Big-step Semantics with Divergence
CPSC 509: Programming Language Principles

Ronald Garcia*

30 September 2019
(**Time Stamp**: 19:04, Friday 5th April, 2024)

Syntax

$$n \in \mathbb{Z}, \quad bv \in \textsc{Bool}, \quad X \in \textsc{Loc}, \quad a \in \textsc{AExp}, \quad b \in \textsc{BExp}, \quad c \in \textsc{Com},$$

$$\begin{array}{rcl} a & ::= & X \mid n \mid a + a \mid a - a \mid a * a \\ b & ::= & \mathsf{true} \mid \mathsf{false} \mid a = a \mid a \leq a \mid \neg b \mid b \wedge b \mid b \vee b \\ c & ::= & \mathsf{skip} \mid X := a \mid c; c \mid \mathsf{if}\ b\ \mathsf{then}\ c\ \mathsf{else}\ c \mid \mathsf{while}\ b\ \mathsf{do}\ c \\ bv & ::= & \mathsf{true} \mid \mathsf{false} \end{array}$$

Big-step Semantics

$$\sigma \in \textsc{Store} = \textsc{Loc} \to \mathbb{Z}$$
$$\textsc{ACfg} = \textsc{AExp} \times \textsc{Store}, \quad \textsc{BCfg} = \textsc{BExp} \times \textsc{Store}, \quad \textsc{CCfg} = \textsc{Com} \times \textsc{Store}$$

$$\sigma_z \in \textsc{Store}$$
$$\sigma_z(X) = 0$$

$$\cdot[\cdot \mapsto \cdot] : \textsc{Store} \times \textsc{Loc} \times \mathbb{Z} \to \textsc{Store}$$
$$\sigma[X_0 \mapsto n](X_0) = n$$
$$\sigma[X_0 \mapsto n](X_1) = \sigma(X_1) \quad \text{if } X_0 \neq X_1$$

$\boxed{\Downarrow_{\text{AExp}} \ \subseteq \text{ACfg} \times \mathbb{Z}}$

$$\frac{}{\langle n, \sigma\rangle \Downarrow_{\text{AExp}} n}\ \text{(enum)} \qquad \frac{}{\langle X, \sigma\rangle \Downarrow_{\text{AExp}} \sigma(X)}\ \text{(eloc)} \qquad \frac{\langle a_1, \sigma\rangle \Downarrow_{\text{AExp}} n_1 \quad \langle a_2, \sigma\rangle \Downarrow_{\text{AExp}} n_2}{\langle a_1 \texttt{+} a_2, \sigma\rangle \Downarrow_{\text{AExp}} n_1 + n_2}\ \text{(eplus)}$$

$$\frac{\langle a_1, \sigma\rangle \Downarrow_{\text{AExp}} n_1 \quad \langle a_2, \sigma\rangle \Downarrow_{\text{AExp}} n_2}{\langle a_1 \texttt{-} a_2, \sigma\rangle \Downarrow_{\text{AExp}} n_1 - n_2}\ \text{(eminus)} \qquad \frac{\langle a_1, \sigma\rangle \Downarrow_{\text{AExp}} n_1 \quad \langle a_2, \sigma\rangle \Downarrow_{\text{AExp}} n_2}{\langle a_1 \ast a_2, \sigma\rangle \Downarrow_{\text{AExp}} n_1 \ast n_2}\ \text{(etimes)}$$

$\boxed{\Downarrow_{\text{BExp}} \ \subseteq \text{BCfg} \times \text{Bool}}$

$$\frac{}{\langle \mathsf{true}, \sigma\rangle \Downarrow_{\text{BExp}} \mathsf{true}}\ \text{(etrue)} \qquad \frac{}{\langle \mathsf{false}, \sigma\rangle \Downarrow_{\text{BExp}} \mathsf{false}}\ \text{(efalse)}$$

$$\frac{\langle a_1, \sigma\rangle \Downarrow_{\text{AExp}} n_1 \quad \langle a_2, \sigma\rangle \Downarrow_{\text{AExp}} n_2}{\langle a_1 = a_2, \sigma\rangle \Downarrow_{\text{BExp}} bv}\ \text{(eeq)} \ \begin{cases} bv = \mathsf{true} \text{ if } n_1 = n_2 \\ bv = \mathsf{false} \text{ if } n_1 \neq n_2 \end{cases}$$

$$\frac{\langle a_1, \sigma\rangle \Downarrow_{\text{AExp}} n_1 \quad \langle a_2, \sigma\rangle \Downarrow_{\text{AExp}} n_2}{\langle a_1 \leq a_2, \sigma\rangle \Downarrow_{\text{BExp}} bv}\ \text{(eleq)} \ \begin{cases} bv = \mathsf{true} \text{ if } n_1 \leq n_2 \\ bv = \mathsf{false} \text{ if } n_1 > n_2 \end{cases}$$

$$\frac{\langle b, \sigma\rangle \Downarrow_{\text{BExp}} bv_1}{\langle \neg b, \sigma\rangle \Downarrow_{\text{BExp}} bv_2}\ \text{(enot)} \ \begin{cases} bv_2 = \mathsf{true} \text{ if } bv_1 = \mathsf{false} \\ bv_2 = \mathsf{false} \text{ if } bv_1 = \mathsf{true} \end{cases}$$

$$\frac{\langle b_1, \sigma\rangle \Downarrow_{\text{BExp}} bv_1 \quad \langle b_2, \sigma\rangle \Downarrow_{\text{BExp}} bv_2}{\langle b_1 \wedge b_2, \sigma\rangle \Downarrow_{\text{BExp}} bv_3}\ \text{(eand)} \ \begin{cases} bv_3 = \mathsf{true} \text{ if } bv_1 = bv_2 = \mathsf{true} \\ bv_3 = \mathsf{false} \text{ if otherwise} \end{cases}$$

$$\frac{\langle b_1, \sigma\rangle \Downarrow_{\text{BExp}} bv_1 \quad \langle b_2, \sigma\rangle \Downarrow_{\text{BExp}} bv_2}{\langle b_1 \vee b_2, \sigma\rangle \Downarrow_{\text{BExp}} bv_3}\ \text{(eor)} \ \begin{cases} bv_3 = \mathsf{true} \text{ if } bv_1 = \mathsf{true} \text{ or } bv_2 = \mathsf{true} \\ bv_3 = \mathsf{false} \text{ if otherwise} \end{cases}$$

$\boxed{\Downarrow_{\text{Com}} \ \subseteq \text{CCfg} \times \text{Store}}$

$$\frac{}{\langle \mathsf{skip}, \sigma\rangle \Downarrow_{\text{Com}} \sigma}\ \text{(eskip)} \qquad \frac{\langle a, \sigma\rangle \Downarrow_{\text{AExp}} n}{\langle X := a, \sigma\rangle \Downarrow_{\text{Com}} \sigma[X \mapsto n]}\ \text{(eassign)}$$

$$\frac{\langle c_1, \sigma\rangle \Downarrow_{\text{Com}} \sigma' \quad \langle c_2, \sigma'\rangle \Downarrow_{\text{Com}} \sigma''}{\langle c_1 ; c_2, \sigma\rangle \Downarrow_{\text{Com}} \sigma''}\ \text{(eseq)} \qquad \frac{\langle b, \sigma\rangle \Downarrow_{\text{BExp}} \mathsf{true} \quad \langle c_1, \sigma\rangle \Downarrow_{\text{Com}} \sigma'}{\langle \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2, \sigma\rangle \Downarrow_{\text{Com}} \sigma'}\ \text{(eif-t)}$$

$$\frac{\langle b, \sigma\rangle \Downarrow_{\text{BExp}} \mathsf{false} \quad \langle c_2, \sigma\rangle \Downarrow_{\text{Com}} \sigma'}{\langle \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2, \sigma\rangle \Downarrow_{\text{Com}} \sigma'}\ \text{(eif-f)} \qquad \frac{\langle b, \sigma\rangle \Downarrow_{\text{BExp}} \mathsf{false}}{\langle \mathsf{while}\ b\ \mathsf{do}\ c, \sigma\rangle \Downarrow_{\text{Com}} \sigma}\ \text{(ewhile-f)}$$

$$\frac{\langle b, \sigma\rangle \Downarrow_{\text{BExp}} \mathsf{true} \quad \langle c, \sigma\rangle \Downarrow_{\text{Com}} \sigma' \quad \langle \mathsf{while}\ b\ \mathsf{do}\ c, \sigma'\rangle \Downarrow_{\text{Com}} \sigma''}{\langle \mathsf{while}\ b\ \mathsf{do}\ c, \sigma\rangle \Downarrow_{\text{Com}} \sigma''}\ \text{(ewhile-t)}$$

$\boxed{\Uparrow \subseteq \textsc{CCfg}}$

$$\frac{\langle c_1, \sigma\rangle \Uparrow}{\langle c_1;c_2, \sigma\rangle \Uparrow}\ (\text{dseq1}) \qquad \frac{\langle c_1, \sigma\rangle \Downarrow_{\textsc{Com}} \sigma' \quad \langle c_2, \sigma'\rangle \Uparrow}{\langle c_1;c_2, \sigma\rangle \Uparrow}\ (\text{dseq2}) \qquad \frac{\langle b, \sigma\rangle \Downarrow_{\textsc{BExp}} \mathsf{true} \quad \langle c_1, \sigma\rangle \Uparrow}{\langle \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2, \sigma\rangle \Uparrow}\ (\text{dif-t})$$

$$\frac{\langle b, \sigma\rangle \Downarrow_{\textsc{BExp}} \mathsf{false} \quad \langle c_2, \sigma\rangle \Uparrow}{\langle \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2, \sigma\rangle \Uparrow}\ (\text{dif-f}) \qquad \frac{\langle b, \sigma\rangle \Downarrow_{\textsc{BExp}} \mathsf{true} \quad \langle c, \sigma\rangle \Uparrow}{\langle \mathsf{while}\ b\ \mathsf{do}\ c, \sigma\rangle \Uparrow}\ (\text{dwhile-c})$$

$$\frac{\langle b, \sigma\rangle \Downarrow_{\textsc{BExp}} \mathsf{true} \quad \langle c, \sigma\rangle \Downarrow_{\textsc{Com}} \sigma' \quad \langle \mathsf{while}\ b\ \mathsf{do}\ c, \sigma'\rangle \Uparrow}{\langle \mathsf{while}\ b\ \mathsf{do}\ c, \sigma\rangle \Uparrow}\ (\text{dwhile-w})$$

$$\textsc{Pgm} = \textsc{Com}, \quad \textsc{Obs} = \textsc{Store} \cup \{\, \infty \,\}$$

$eval_{IMP} : \textsc{Pgm} \overset{\text{dens}}{\to} \textsc{Obs}$
$eval_{IMP}(c) = \sigma$ if $\langle c, \sigma_z\rangle \Downarrow_{\textsc{Com}} \sigma$
$eval_{IMP}(c) = \infty$ if $\langle c, \sigma_z\rangle \Uparrow$

Reasoning Principles

Convergence

Proposition 1 (Principle of Induction on Derivations $\mathcal{D} :: \langle c, \sigma\rangle \Downarrow_{\text{COM}} \sigma'$)**.** *Let Φ be a predicate on derivations $\mathcal{D} :: \langle c, \sigma\rangle \Downarrow_{\text{COM}} \sigma'$. Then $\Phi(\mathcal{D})$ holds for all derivations $\mathcal{D} \in \text{DERIV}$ if:*

1. $\forall \sigma \in \text{STORE}.\, \Phi\left(\dfrac{}{\langle \textsf{skip}, \sigma\rangle \Downarrow_{\text{COM}} \sigma}\ \textit{(eskip)}\right)$;
2. $\forall a \in \text{AEXP}.\forall n \in \mathbb{Z}.\forall X \in \text{LOC}.\forall \sigma \in \text{STORE}.\ \langle a, \sigma\rangle \Downarrow_{\text{AEXP}} n \Rightarrow$ $\Phi\left(\dfrac{}{\langle X := a, \sigma\rangle \Downarrow_{\text{COM}} \sigma[X \mapsto n]}\ \textit{(eassign)}\right)$;
3. $\forall c_1, c_2 \in \text{COM}.\forall \sigma, \sigma', \sigma'' \in \text{STORE}.\forall \mathcal{D}_1, \mathcal{D}_2 \in \text{DERIV}.\ D_1 :: \langle c_1, \sigma\rangle \Downarrow_{\text{COM}} \sigma' \wedge D_2 :: \langle c_2, \sigma'\rangle \Downarrow_{\text{COM}} \sigma'' \wedge$ $\Phi(D_1) \wedge \Phi(D_2) \Rightarrow \Phi\left(\dfrac{\overset{\mathcal{D}_1}{\langle c_1, \sigma\rangle \Downarrow_{\text{COM}} \sigma'} \quad \overset{\mathcal{D}_2}{\langle c_2, \sigma'\rangle \Downarrow_{\text{COM}} \sigma''}}{\langle c_1; c_2, \sigma\rangle \Downarrow_{\text{COM}} \sigma''}\ \textit{(eseq)}\right)$;
4. $\forall b \in \text{BEXP}.\forall c_1, c_2 \in \text{COM}.\forall \sigma, \sigma' \in \text{STORE}.\forall \mathcal{D}_1 \in \text{DERIV}.\ \langle b, \sigma\rangle \Downarrow_{\text{BEXP}} \textsf{true} \wedge \mathcal{D}_1 :: \langle c_1, \sigma\rangle \Downarrow_{\text{COM}} \sigma' \wedge$ $\Phi(\mathcal{D}_1) \Rightarrow \Phi\left(\dfrac{\overset{\mathcal{D}_1}{\langle c_1, \sigma\rangle \Downarrow_{\text{COM}} \sigma'}}{\langle \textsf{if}\ b\ \textsf{then}\ c_1\ \textsf{else}\ c_2, \sigma\rangle \Downarrow_{\text{COM}} \sigma'}\ \textit{(eif-t)}\right)$;
5. $\forall b \in \text{BEXP}.\forall c_1, c_2 \in \text{COM}.\forall \sigma, \sigma' \in \text{STORE}.\forall \mathcal{D}_2 \in \text{DERIV}.\ \langle b, \sigma\rangle \Downarrow_{\text{BEXP}} \textsf{false} \wedge \mathcal{D}_2 :: \langle c_2, \sigma\rangle \Downarrow_{\text{COM}} \sigma' \wedge$ $\Phi(\mathcal{D}_2) \Rightarrow \Phi\left(\dfrac{\overset{\mathcal{D}_2}{\langle c_2, \sigma\rangle \Downarrow_{\text{COM}} \sigma'}}{\langle \textsf{if}\ b\ \textsf{then}\ c_1\ \textsf{else}\ c_2, \sigma\rangle \Downarrow_{\text{COM}} \sigma'}\ \textit{(eif-f)}\right)$;
6. $\forall b \in \text{BEXP}.\forall c \in \text{COM}.\forall \sigma \in \text{STORE}.\ \langle b, \sigma\rangle \Downarrow_{\text{BEXP}} \textsf{false} \Rightarrow \Phi\left(\dfrac{}{\langle \textsf{while}\ b\ \textsf{do}\ c, \sigma\rangle \Downarrow_{\text{COM}} \sigma}\ \textit{(ewhile-f)}\right)$;
7. $\forall b \in \text{BEXP}.\forall c \in \text{COM}.\forall \sigma, \sigma', \sigma'' \in \text{STORE}.\forall \mathcal{D}_1, \mathcal{D}_2 \in \text{DERIV}.$ $\langle b, \sigma\rangle \Downarrow_{\text{BEXP}} \textsf{true} \wedge \mathcal{D}_1 :: \langle c_1, \sigma\rangle \Downarrow_{\text{COM}} \sigma' \wedge \mathcal{D}_2 :: \langle \textsf{while}\ b\ \textsf{do}\ c, \sigma'\rangle \Downarrow_{\text{COM}} \sigma'' \wedge$ $\Phi(\mathcal{D}_1) \wedge \Phi(\mathcal{D}_2) \Rightarrow \Phi\left(\dfrac{\overset{\mathcal{D}_1}{\langle c, \sigma\rangle \Downarrow_{\text{COM}} \sigma'} \quad \overset{\mathcal{D}_2}{\langle \textsf{while}\ b\ \textsf{do}\ c, \sigma'\rangle \Downarrow_{\text{COM}} \sigma''}}{\langle \textsf{while}\ b\ \textsf{do}\ c, \sigma\rangle \Downarrow_{\text{COM}} \sigma''}\ \textit{(ewhile-t)}\right)$.

Proposition 2 (Principle of Rule Induction for $\langle c, \sigma\rangle \Downarrow_{\text{COM}} \sigma'$)**.** *Let Φ be a predicate on $\text{CCFG} \times \text{STORE}$. Then $\Phi(\langle c, \sigma\rangle, \sigma')$ holds for all $\langle c, \sigma\rangle \Downarrow_{\text{COM}} \sigma'$ if:*

1. $\forall \sigma \in \text{STORE}.\, \Phi(\langle \textsf{skip}, \sigma\rangle, \sigma)$;
2. $\forall a \in \text{AEXP}.\forall n \in \mathbb{Z}.\forall X \in \text{LOC}.\forall \sigma \in \text{STORE}.\ \langle a, \sigma\rangle \Downarrow_{\text{AEXP}} n \Rightarrow \Phi(\langle X := a, \sigma\rangle, \sigma[X \mapsto n])$;
3. $\forall c_1, c_2 \in \text{COM}.\forall \sigma, \sigma', \sigma'' \in \text{STORE}.\ \Phi(\langle c_1, \sigma\rangle, \sigma') \wedge \Phi(\langle c_2, \sigma'\rangle, \sigma'') \Rightarrow \Phi(\langle c_1; c_2, \sigma\rangle, \sigma'')$;
4. $\forall b \in \text{BEXP}.\forall c_1, c_2 \in \text{COM}.\forall \sigma, \sigma' \in \text{STORE}.\ \langle b, \sigma\rangle \Downarrow_{\text{BEXP}} \textsf{true} \wedge \Phi(\langle c_1, \sigma\rangle, \sigma') \Rightarrow$ $\Phi(\langle \textsf{if}\ b\ \textsf{then}\ c_1\ \textsf{else}\ c_2, \sigma\rangle, \sigma')$;
5. $\forall b \in \text{BEXP}.\forall c_1, c_2 \in \text{COM}.\forall \sigma, \sigma' \in \text{STORE}.\ \langle b, \sigma\rangle \Downarrow_{\text{BEXP}} \textsf{false} \wedge \Phi(\langle c_2, \sigma\rangle, \sigma') \Rightarrow$ $\Phi(\langle \textsf{if}\ b\ \textsf{then}\ c_1\ \textsf{else}\ c_2, \sigma\rangle, \sigma')$;
6. $\forall b \in \text{BEXP}.\forall c \in \text{COM}.\forall \sigma \in \text{STORE}.\ \langle b, \sigma\rangle \Downarrow_{\text{BEXP}} \textsf{false} \Rightarrow \Phi(\langle \textsf{while}\ b\ \textsf{do}\ c, \sigma\rangle, \sigma)$;
7. $\forall b \in \text{BEXP}.\forall c \in \text{COM}.\forall \sigma, \sigma', \sigma'' \in \text{STORE}.\ \langle b, \sigma\rangle \Downarrow_{\text{BEXP}} \textsf{true} \wedge \Phi(\langle c_1, \sigma\rangle, \sigma') \wedge \Phi(\langle \textsf{while}\ b\ \textsf{do}\ c, \sigma'\rangle, \sigma'') \Rightarrow$ $\Phi(\langle \textsf{while}\ b\ \textsf{do}\ c, \sigma\rangle, \sigma'')$.

Divergence

$$\mathcal{R}_{\Uparrow} = \text{(dwhile-w)} \cup \text{(dwhile-c)} \cup \text{(dif-f)} \cup \text{(dif-t)} \cup \text{(dseq2)} \cup \text{(dseq1)}$$

Monotone Function Induced by $\mathcal{R}_{\Uparrow}$

$$\mathcal{M} : \mathcal{P}(\text{CCFG}) \to \mathcal{P}(\text{CCFG})$$

$$\mathcal{M}(S) = \left\{ \langle c, \sigma \rangle \in \text{CCFG} \;\middle|\; \exists \frac{S'}{\langle c', \sigma' \rangle} \in \mathcal{R}_{\Uparrow}.\, c = c' \wedge \sigma = \sigma' \wedge S' \subseteq S \right\}$$

Split by Rules

$$\begin{aligned}
\mathcal{M}(S) = &\left\{ \langle c, \sigma \rangle \in \text{CCFG} \;\middle|\; \exists \frac{\langle c_1, \sigma' \rangle}{\langle c_1 ; c_2, \sigma' \rangle} \in \mathcal{R}_{\Uparrow}.\, c = c_1 ; c_2 \wedge \sigma = \sigma' \wedge \{ \langle c_1, \sigma' \rangle \} \subseteq S \right\} \\
\cup &\left\{ \langle c, \sigma \rangle \in \text{CCFG} \;\middle|\; \exists \frac{\langle c_2, \sigma'' \rangle}{\langle c_1 ; c_2, \sigma' \rangle} \in \mathcal{R}_{\Uparrow}.\, \langle c_1, \sigma' \rangle \Downarrow_{\text{Com}} \sigma'' \wedge c = c_1 ; c_2 \wedge \sigma = \sigma' \wedge \{ \langle c_2, \sigma'' \rangle \} \subseteq S \right\} \\
\cup &\left\{ \langle c, \sigma \rangle \in \text{CCFG} \;\middle|\; \begin{array}{l} \exists \dfrac{\langle c_1, \sigma' \rangle}{\langle \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2, \sigma' \rangle} \in \mathcal{R}_{\Uparrow}.\, \langle b, \sigma' \rangle \Downarrow_{\text{BExp}} \mathsf{true} \wedge \\ \quad c = \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2 \wedge \sigma = \sigma' \wedge \{ \langle c_1, \sigma' \rangle \} \subseteq S \end{array} \right\} \\
\cup &\left\{ \langle c, \sigma \rangle \in \text{CCFG} \;\middle|\; \begin{array}{l} \exists \dfrac{\langle c_2, \sigma' \rangle}{\langle \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2, \sigma' \rangle} \in \mathcal{R}_{\Uparrow}.\, \langle b, \sigma' \rangle \Downarrow_{\text{BExp}} \mathsf{false} \wedge \\ \quad c = \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2 \wedge \sigma = \sigma' \wedge \{ \langle c_2, \sigma' \rangle \} \subseteq S \end{array} \right\} \\
\cup &\left\{ \langle c, \sigma \rangle \in \text{CCFG} \;\middle|\; \begin{array}{l} \exists \dfrac{\langle c_1, \sigma' \rangle}{\langle \mathsf{while}\ b\ \mathsf{do}\ c_1, \sigma' \rangle} \in \mathcal{R}_{\Uparrow}.\, \langle b, \sigma' \rangle \Downarrow_{\text{BExp}} \mathsf{true} \wedge \\ \quad c = \mathsf{while}\ b\ \mathsf{do}\ c_1 \wedge \sigma = \sigma' \wedge \{ \langle c_1, \sigma' \rangle \} \subseteq S \end{array} \right\} \\
\cup &\left\{ \langle c, \sigma \rangle \in \text{CCFG} \;\middle|\; \begin{array}{l} \exists \dfrac{\langle \mathsf{while}\ b\ \mathsf{do}\ c_1, \sigma'' \rangle}{\langle \mathsf{while}\ b\ \mathsf{do}\ c_1, \sigma' \rangle} \in \mathcal{R}_{\Uparrow}.\, \langle b, \sigma' \rangle \Downarrow_{\text{BExp}} \mathsf{true} \wedge \langle c_1, \sigma' \rangle \Downarrow_{\text{Com}} \sigma'' \wedge \\ \quad c = \mathsf{while}\ b\ \mathsf{do}\ c_1 \wedge \sigma = \sigma' \wedge \{ \langle \mathsf{while}\ b\ \mathsf{do}\ c_1, \sigma'' \rangle \} \subseteq S \end{array} \right\}
\end{aligned}$$

Simplified

$$\begin{aligned}
\mathcal{M}(S) = &\{ \langle c, \sigma \rangle \in \text{CCFG} \mid \exists c_1, c_2 \in \text{Com}, \sigma' \in \text{Store}.\, c = c_1 ; c_2 \wedge \sigma = \sigma' \wedge \langle c_1, \sigma' \rangle \in S \} \\
\cup &\{ \langle c, \sigma \rangle \in \text{CCFG} \mid \exists c_1, c_2 \in \text{Com}, \sigma', \sigma'' \in \text{Store}.\, \langle c_1, \sigma' \rangle \Downarrow_{\text{Com}} \sigma'' \wedge c = c_1 ; c_2 \wedge \sigma = \sigma' \wedge \langle c_2, \sigma'' \rangle \in S \} \\
\cup &\left\{ \langle c, \sigma \rangle \in \text{CCFG} \;\middle|\; \begin{array}{l} \exists b \in \text{BExp}, c_1, c_2 \in \text{Com}, \sigma' \in \text{Store}. \\ \quad \langle b, \sigma' \rangle \Downarrow_{\text{BExp}} \mathsf{true} \wedge c = \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2 \wedge \sigma = \sigma' \wedge \langle c_1, \sigma' \rangle \in S \end{array} \right\} \\
\cup &\left\{ \langle c, \sigma \rangle \in \text{CCFG} \;\middle|\; \begin{array}{l} \exists b \in \text{BExp}, c_1, c_2 \in \text{Com}, \sigma' \in \text{Store}. \\ \quad \langle b, \sigma' \rangle \Downarrow_{\text{BExp}} \mathsf{false} \wedge c = \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2 \wedge \sigma = \sigma' \wedge \langle c_2, \sigma' \rangle \in S \end{array} \right\} \\
\cup &\left\{ \langle c, \sigma \rangle \in \text{CCFG} \;\middle|\; \begin{array}{l} \exists b \in \text{BExp}, c_1 \in \text{Com}, \sigma' \in \text{Store}. \\ \quad \langle b, \sigma' \rangle \Downarrow_{\text{BExp}} \mathsf{true} \wedge c = \mathsf{while}\ b\ \mathsf{do}\ c_1 \wedge \sigma = \sigma' \wedge \langle c_1, \sigma' \rangle \in S \end{array} \right\} \\
\cup &\left\{ \langle c, \sigma \rangle \in \text{CCFG} \;\middle|\; \begin{array}{l} \exists b \in \text{BExp}, c_1 \in \text{Com}, \sigma', \sigma'' \in \text{Store}.\, \langle b, \sigma' \rangle \Downarrow_{\text{BExp}} \mathsf{true} \wedge \\ \quad \langle c_1, \sigma' \rangle \Downarrow_{\text{Com}} \sigma'' \wedge c = \mathsf{while}\ b\ \mathsf{do}\ c_1 \wedge \sigma = \sigma' \wedge \langle \mathsf{while}\ b\ \mathsf{do}\ c_1, \sigma'' \rangle \in S \end{array} \right\}
\end{aligned}$$

More Simplified (resolve σ', rename σ'', reorder conjuncts)

$$
\begin{aligned}
\mathcal{M}(S) = {} & \{\, \langle c, \sigma\rangle \in \textsc{CCfg} \mid \exists c_1, c_2 \in \textsc{Com}.\, c = c_1;c_2 \wedge \langle c_1, \sigma\rangle \in S \,\} \\
& \cup \{\, \langle c, \sigma\rangle \in \textsc{CCfg} \mid \exists c_1, c_2 \in \textsc{Com}, \sigma' \in \textsc{Store}.\, c = c_1;c_2 \wedge \langle c_1, \sigma\rangle \Downarrow_{\textsc{Com}} \sigma' \wedge \langle c_2, \sigma'\rangle \in S \,\} \\
& \cup \left\{ \langle c, \sigma\rangle \in \textsc{CCfg} \;\middle|\; \begin{array}{l} \exists b \in \textsc{BExp}, c_1, c_2 \in \textsc{Com}. \\ \quad c = \textsf{if } b \textsf{ then } c_1 \textsf{ else } c_2 \wedge \langle b, \sigma\rangle \Downarrow_{\textsc{BExp}} \textsf{true} \wedge \langle c_1, \sigma\rangle \in S \end{array} \right\} \\
& \cup \left\{ \langle c, \sigma\rangle \in \textsc{CCfg} \;\middle|\; \begin{array}{l} \exists b \in \textsc{BExp}, c_1, c_2 \in \textsc{Com}. \\ \quad c = \textsf{if } b \textsf{ then } c_1 \textsf{ else } c_2 \wedge \langle b, \sigma\rangle \Downarrow_{\textsc{BExp}} \textsf{false} \wedge \langle c_2, \sigma\rangle \in S \end{array} \right\} \\
& \cup \left\{ \langle c, \sigma\rangle \in \textsc{CCfg} \;\middle|\; \begin{array}{l} \exists b \in \textsc{BExp}, c_1 \in \textsc{Com}. \\ \quad c = \textsf{while } b \textsf{ do } c_1 \wedge \langle b, \sigma\rangle \Downarrow_{\textsc{BExp}} \textsf{true} \wedge \langle c_1, \sigma\rangle \in S \end{array} \right\} \\
& \cup \left\{ \langle c, \sigma\rangle \in \textsc{CCfg} \;\middle|\; \begin{array}{r} \exists b \in \textsc{BExp}, c_1 \in \textsc{Com}, \sigma' \in \textsc{Store}.\, c = \textsf{while } b \textsf{ do } c_1 \wedge \langle b, \sigma\rangle \Downarrow_{\textsc{BExp}} \textsf{true} \wedge \\ \langle c_1, \sigma\rangle \Downarrow_{\textsc{Com}} \sigma' \wedge \sigma = \sigma \wedge \langle \textsf{while } b \textsf{ do } c_1, \sigma'\rangle \in S \end{array} \right\}
\end{aligned}
$$

Proposition 3 (Principle of Rule Coinduction for $\langle c, \sigma \rangle \Uparrow$). *Let $S \subseteq$ CCFG. Then $\langle c, \sigma \rangle \Uparrow$ holds for all $\langle c, \sigma \rangle \in S$ if for each $\langle c, \sigma \rangle \in S$ one of the following holds:*

1. $\exists c_1, c_2 \in \textsc{Com}.\, c = c_1;c_2 \wedge \langle c_1, \sigma \rangle \in S;$
2. $\exists c_1, c_2 \in \textsc{Com}, \sigma' \in \textsc{Store}.\, c = c_1;c_2 \wedge \langle c_1, \sigma \rangle \Downarrow_{\textsc{Com}} \sigma' \wedge \langle c_2, \sigma' \rangle \in S;$
3. $\exists b \in \textsc{BExp}, c_1, c_2 \in \textsc{Com}.$
   $c = \textit{if}\ b\ \textit{then}\ c_1\ \textit{else}\ c_2 \wedge \langle b, \sigma \rangle \Downarrow_{\textsc{BExp}} \textit{true} \wedge \langle c_1, \sigma \rangle \in S;$
4. $\exists b \in \textsc{BExp}, c_1, c_2 \in \textsc{Com}.$
   $c = \textit{if}\ b\ \textit{then}\ c_1\ \textit{else}\ c_2 \wedge \langle b, \sigma \rangle \Downarrow_{\textsc{BExp}} \textit{false} \wedge \langle c_2, \sigma \rangle \in S;$
5. $\exists b \in \textsc{BExp}, c_1 \in \textsc{Com}.$
   $c = \textit{while}\ b\ \textit{do}\ c_1 \wedge \langle b, \sigma \rangle \Downarrow_{\textsc{BExp}} \textit{true} \wedge \langle c_1, \sigma \rangle \in S;$
6. $\exists b \in \textsc{BExp}, c_1 \in \textsc{Com}, \sigma' \in \textsc{Store}.\, c = \textit{while}\ b\ \textit{do}\ c_1 \wedge \langle b, \sigma \rangle \Downarrow_{\textsc{BExp}} \textit{true} \wedge$
   $\langle c_1, \sigma \rangle \Downarrow_{\textsc{Com}} \sigma' \wedge \langle \textit{while}\ b\ \textit{do}\ c_1, \sigma' \rangle \in S.$