Compositional Verification of Termination-Preserving Refinement of Concurrent Programs (Technical Report)

Hongjin Liang[1], Xinyu Feng[1], and Zhong Shao[2]

[1]University of Science and Technology of China
[2]Yale University

June 9, 2014

NOTES: This TR is a supplement to our CSL-LICS'14 paper. It includes full formulations of the technical settings (Section 1), our RGSim-T definitions (Section 2), the full program logic (Section 3), all the examples we have verified (Section 4) and the full formal soundness proofs (Section 5).

Moreover, we introduce a new interesting assertion $p \owedge q$ which allows local reasoning about the number of tokens that is conditional upon the shared state in runtime. See Section 2 for its semantics, Section 3 for the related local reasoning rule and Section 4 for its use in practical examples.

We also provide a transitivity rule on the binary judgments. We introduce new assertions to specify the compositions of two relational assertions and of two actions (see Section 2).

For more informal explanations and the high-level picture, please see our CSL-LICS'14 paper. Both the paper and this companion TR can be found at the following url:

`http://kyhcs.ustcsz.edu.cn/relconcur/rgsimt`

1 Basic Technical Settings and Termination-Preserving Refinement

1.1 The Language

We show the language in Figure 1. We assume the program variables used in the target code are different from the ones used in the source (e.g., we use x and X for target and source level variables respectively).

$$
\begin{array}{rcll}
(\mathit{Event}) & e & ::= & \ldots \qquad\qquad (\mathit{Label}) \quad \iota \ ::= \ e \mid \tau \\
(\mathit{Store}) & s, \mathbb{s} & \in & \mathit{PVar} \rightharpoonup \mathit{Val} \qquad (\mathit{Heap}) \quad h, \mathbb{h} \ \in \ \mathit{Addr} \rightharpoonup \mathit{Val} \\
(\mathit{State}) & \sigma, \Sigma & ::= & (s, h) \\
(\mathit{Instr}) & c, \mathbb{c} & \in & \mathit{State} \rightharpoonup \mathscr{P}((\mathit{Label} \times \mathit{State}) \cup \{\mathbf{abort}\}) \\
(\mathit{Expr}) & E, \mathbb{E} & ::= & x \mid n \mid E + E \mid \ldots \\
(\mathit{BExp}) & B, \mathbb{B} & ::= & \mathbf{true} \mid \mathbf{false} \mid E = E \mid !B \mid \ldots \\
(\mathit{Stmt}) & C, \mathbb{C} & ::= & \mathbf{skip} \mid c \mid \langle C \rangle \mid C_1; C_2 \mid \mathbf{if}\ (B)\ C_1\ \mathbf{else}\ C_2 \\
& & \mid & \mathbf{while}\ (B)\ C \mid C_1 \,\|\, C_2
\end{array}
$$

Figure 1: Generic language at target and source levels.

We show the operational semantics in Figure 2. The semantics of E and B are defined by $[\![E]\!]$ and $[\![B]\!]$ respectively. $[\![E]\!]$ is a partial function of type $\mathit{Store} \rightharpoonup \mathit{Val}$. $[\![B]\!]$ is a partial function of type $\mathit{Store} \rightharpoonup \{\mathbf{true}, \mathbf{false}\}$. They are undefined if variables in E and B are not assigned values in the store s. Their definitions are omitted here.

Conventions. We usually write blackboard bold or capital letters ($\mathbb{s}$, $\mathbb{h}$, Σ, $\mathbb{c}$, $\mathbb{E}$, $\mathbb{B}$ and $\mathbb{C}$) for the notations at the source level to distinguish from the target-level ones (s, h, σ, c, E, B and C). When we discuss the transitivity, we use θ and $\mathrm{C_M}$ for the state and the code at the middle level.

Below we use $\_ \longrightarrow^* \_$ for zero or multiple-step transitions with no events generated, $\_ \longrightarrow^+ \_$ for multiple-step transitions without events, $\_ \xrightarrow{e}{}^+ \_$ for multiple-step transitions with *only one* event e generated, and $\_ \longrightarrow^\omega \cdot$ for an infinite execution without events.

$$\frac{(\iota, \sigma') \in c\ \sigma}{(c, \sigma) \xrightarrow{\iota} (\mathbf{skip}, \sigma')} \qquad \frac{\mathbf{abort} \in c\ \sigma}{(c, \sigma) \longrightarrow \mathbf{abort}} \qquad \frac{\sigma \notin \mathit{dom}(c)}{(c, \sigma) \longrightarrow (c, \sigma)}$$

$$\frac{(C, \sigma) \longrightarrow^* (\mathbf{skip}, \sigma')}{(\langle C \rangle, \sigma) \longrightarrow (\mathbf{skip}, \sigma')} \qquad \frac{(C, \sigma) \longrightarrow^* \mathbf{abort}}{(\langle C \rangle, \sigma) \longrightarrow \mathbf{abort}} \qquad \frac{(C, \sigma) \longrightarrow^\omega \cdot}{(\langle C \rangle, \sigma) \longrightarrow (\langle C \rangle, \sigma)}$$

$$\frac{(C, \sigma) \longrightarrow (C', \sigma')}{(C; C'', \sigma) \longrightarrow (C'; C'', \sigma')} \qquad \frac{(C, \sigma) \xrightarrow{e} (C', \sigma')}{(C; C'', \sigma) \xrightarrow{e} (C'; C'', \sigma')}$$

$$\frac{}{(\mathbf{skip}; C', \sigma) \longrightarrow (C', \sigma)} \qquad \frac{(C, \sigma) \longrightarrow \mathbf{abort}}{(C; C', \sigma) \longrightarrow \mathbf{abort}}$$

$$\frac{[\![B]\!]_s = \mathbf{true}}{(\mathbf{while}\ (B)\ C, (s, h)) \longrightarrow (C; \mathbf{while}\ (B)\ C, (s, h))}$$

$$\frac{[\![B]\!]_s = \mathbf{false}}{(\mathbf{while}\ (B)\ C, (s, h)) \longrightarrow (\mathbf{skip}, (s, h))} \qquad \frac{[\![B]\!]_s\ \mathit{undefined}}{(\mathbf{while}\ (B)\ C, (s, h)) \longrightarrow \mathbf{abort}}$$

$$\frac{[\![B]\!]_s = \mathbf{true}}{(\mathbf{if}\ (B)\ C_1\ \mathbf{else}\ C_2, (s, h)) \longrightarrow (C_1, (s, h))} \qquad \frac{[\![B]\!]_s = \mathbf{false}}{(\mathbf{if}\ (B)\ C_1\ \mathbf{else}\ C_2, (s, h)) \longrightarrow (C_2, (s, h))}$$

$$\frac{[\![B]\!]_s\ \mathit{undefined}}{(\mathbf{if}\ (B)\ C_1\ \mathbf{else}\ C_2, (s, h)) \longrightarrow \mathbf{abort}}$$

$$\frac{(C_1, \sigma) \xrightarrow{\iota} (C_1', \sigma')}{(C_1 \,\|\, C_2, \sigma) \xrightarrow{\iota} (C_1' \,\|\, C_2, \sigma')} \qquad \frac{(C_2, \sigma) \xrightarrow{\iota} (C_2', \sigma')}{(C_1 \,\|\, C_2, \sigma) \xrightarrow{\iota} (C_1 \,\|\, C_2', \sigma')}$$

$$\frac{}{(\mathbf{skip} \,\|\, \mathbf{skip}, \sigma) \longrightarrow (\mathbf{skip}, \sigma)} \qquad \frac{(C_1, \sigma) \longrightarrow \mathbf{abort} \quad \text{or} \quad (C_2, \sigma) \longrightarrow \mathbf{abort}}{(C_1 \,\|\, C_2, \sigma) \longrightarrow \mathbf{abort}}$$

Figure 2: Operational semantics.

1.2 Termination-Preserving Event Trace Refinement

$$(EvtTrace) \quad \mathcal{E} \quad ::= \quad \Downarrow \mid \lightning \mid \epsilon \mid e :: \mathcal{E} \quad \text{(co-inductive interpretation)}$$

We define $ETr(C, \sigma, \mathcal{E})$ in Figure 3.

$$\frac{(C, \sigma) \longrightarrow^* (\mathbf{skip}, \sigma')}{ETr(C, \sigma, \Downarrow)} \qquad \frac{(C, \sigma) \longrightarrow^+ \mathbf{abort}}{ETr(C, \sigma, \lightning)}$$

$$\frac{(C, \sigma) \longrightarrow^+ (C', \sigma') \qquad ETr(C', \sigma', \mathcal{E})}{ETr(C, \sigma, \mathcal{E})} \qquad \frac{(C, \sigma) \stackrel{e}{\longrightarrow}^+ (C', \sigma') \qquad ETr(C', \sigma', \mathcal{E})}{ETr(C, \sigma, e :: \mathcal{E})}$$

Figure 3: Co-inductive definition of $ETr(C, \sigma, \mathcal{E})$.

Definition 1 (Termination-Preserving Refinement).
$(C, \sigma) \sqsubseteq (\mathbb{C}, \Sigma)$ iff $\forall \mathcal{E}.\ ETr(C, \sigma, \mathcal{E}) \implies ETr(\mathbb{C}, \Sigma, \mathcal{E})$.

2 RGSim-T

2.1 Assertion Language

We first define the assertions used in our simulation RGSim-T and our program logic. Their syntax is shown in Figure 4, and their semantics is shown in Figures 5 and 6.

$$
\begin{array}{rrcl}
(\mathit{RelAssn}) & P, Q, I & ::= & B \mid \mathsf{own}(x) \mid emp \mid \boldsymbol{emp} \mid E \mapsto E \mid E \Mapsto E \\
 & & \mid & [\![p]\!] \mid P * Q \mid P \vee Q \mid P \wedge Q \mid P \mathbin{⨾} Q \mid \ldots \\
(\mathit{FullAssn}) & p, q & ::= & P \mid \mathsf{arem}(\mathbb{C}) \mid \mathsf{wf}(E) \mid \lfloor p \rfloor_{\mathsf{a}} \mid \lfloor p \rfloor_{\mathsf{w}} \\
 & & \mid & p * q \mid p \vee q \mid p \wedge q \mid p \oslash q \mid \ldots \\
(\mathit{RelAct}) & R, G & ::= & P \propto Q \mid P \ltimes Q \mid [P] \mid R * R \mid R^{+} \\
 & & \mid & R \vee R \mid R \wedge R \mid R \mathbin{\hat{⨾}} R \mid R \mathbin{\check{⨾}} R \mid \ldots
\end{array}
$$

Figure 4: Assertion language.

The above assertion language extends the one in our CSL-LICS paper with the following new assertions.

1. $p \oslash q$, which is like a conjunction over the concrete and the abstract states and like a separating conjunction over the number of tokens and the abstract code. It would be useful to simplify the verification of some specific examples (see Section 4).

2. $P \mathbin{⨾} Q$, $R \mathbin{\hat{⨾}} R$ and $R \mathbin{\check{⨾}} R$, which are compositions of two relational assertions and of two actions. They are used in the transitivity of the binary judgments (the TRANS rule in Figure 7). We use θ and $\mathrm{C_M}$ to represent the middle-level state and the middle-level code respectively. We also define a predicate $\mathsf{MPrecise}(P, Q)$ in Figure 5, which specifies the precise property about the middle-level states. Here P and Q are relational assertions between low-level and middle-level states and between middle-level and high-level states respectively.

Note that our logic is already very useful without the above extensions. All the examples that we mentioned in our CSL-LICS'14 paper can be verified without these extensions.

$$f_1 \bot f_2 \;\text{ iff }\; (dom(f_1) \cap dom(f_2) = \emptyset)$$
$$(s_1, h_1) \bot (s_2, h_2) \;\text{ iff }\; (s_1 \bot s_2) \wedge (h_1 \bot h_2)$$
$$(s_1, h_1) \uplus (s_2, h_2) \stackrel{\text{def}}{=} \begin{cases} (s_1 \cup s_2, h_1 \cup h_2) & \text{if } (s_1, h_1) \bot (s_2, h_2) \\ \textit{undefined} & \text{otherwise} \end{cases}$$

$$
\begin{array}{lll}
((s,h),(\mathbb{s},\mathbb{h})) \models B & \text{iff} & [\![B]\!]_{s \uplus \mathbb{s}} = \mathbf{true} \\
((s,h),(\mathbb{s},\mathbb{h})) \models \mathsf{own}(x) & \text{iff} & dom(s \uplus \mathbb{s}) = \{x\} \\
((s,h),(\mathbb{s},\mathbb{h})) \models emp & \text{iff} & (dom(s) = \emptyset) \wedge (dom(h) = \emptyset) \\
((s,h),(\mathbb{s},\mathbb{h})) \models \boldsymbol{emp} & \text{iff} & (dom(\mathbb{s}) = \emptyset) \wedge (dom(\mathbb{h}) = \emptyset) \\
((s,h),(\mathbb{s},\mathbb{h})) \models E_1 \mapsto E_2 & \text{iff} & \exists l, n.\ [\![E_1]\!]_{s \uplus \mathbb{s}} = l \wedge [\![E_2]\!]_{s \uplus \mathbb{s}} = n \wedge dom(h) = \{l\} \wedge h(l) = n \\
((s,h),(\mathbb{s},\mathbb{h})) \models E_1 \Mapsto E_2 & \text{iff} & \exists l, n.\ [\![E_1]\!]_{s \uplus \mathbb{s}} = l \wedge [\![E_2]\!]_{s \uplus \mathbb{s}} = n \wedge dom(\mathbb{h}) = \{l\} \wedge \mathbb{h}(l) = n
\end{array}
$$

$$\mathsf{emp} \stackrel{\text{def}}{=} emp \wedge \boldsymbol{emp}$$

$$(\sigma, \Sigma) \models P \,\mathbin{;}\, Q \quad \text{iff} \quad \exists \theta.\ (\sigma, \theta) \models P \wedge (\theta, \Sigma) \models Q$$

$$((\sigma, \Sigma), (\sigma', \Sigma'), b) \models P \propto Q \;\text{ iff }\; (\sigma, \Sigma) \models P \wedge (\sigma', \Sigma') \models Q \wedge (b = \mathbf{true})$$
$$((\sigma, \Sigma), (\sigma', \Sigma'), b) \models P \ltimes Q \;\text{ iff }\; (\sigma, \Sigma) \models P \wedge (\sigma', \Sigma') \models Q$$
$$((\sigma, \Sigma), (\sigma', \Sigma'), b) \models [P] \;\text{ iff }\; (\sigma, \Sigma) \models P \wedge (\sigma = \sigma') \wedge (\Sigma = \Sigma')$$
$$
\begin{array}{l}
((\sigma, \Sigma), (\sigma', \Sigma'), b) \models R_1 * R_2 \;\text{ iff} \\
\quad \exists \sigma_1, \Sigma_1, \sigma_2, \Sigma_2, \sigma_1', \Sigma_1', \sigma_2', \Sigma_2'.\ ((\sigma_1, \Sigma_1), (\sigma_1', \Sigma_1'), b) \models R_1 \wedge ((\sigma_2, \Sigma_2), (\sigma_2', \Sigma_2'), b) \models R_2 \\
\qquad \wedge (\sigma = \sigma_1 \uplus \sigma_2) \wedge (\sigma' = \sigma_1' \uplus \sigma_2') \wedge (\Sigma = \Sigma_1 \uplus \Sigma_2) \wedge (\Sigma' = \Sigma_1' \uplus \Sigma_2')
\end{array}
$$
$$
\begin{array}{l}
((\sigma, \Sigma), (\sigma', \Sigma'), b) \models R^+ \;\text{ iff} \\
\quad (((\sigma, \Sigma), (\sigma', \Sigma'), b) \models R) \\
\quad \vee (\exists \sigma'', \Sigma'', b', b''.\ (((\sigma, \Sigma), (\sigma'', \Sigma''), b') \models R) \wedge (((\sigma'', \Sigma''), (\sigma', \Sigma'), b'') \models R^+) \wedge (b = b' \vee b''))
\end{array}
$$

$$\mathsf{Id} \stackrel{\text{def}}{=} [\mathbf{true}] \qquad \mathsf{Emp} \stackrel{\text{def}}{=} \mathsf{emp} \ltimes \mathsf{emp} \qquad \mathsf{True} \stackrel{\text{def}}{=} \mathbf{true} \ltimes \mathbf{true}$$

$$
\begin{array}{l}
((\sigma, \Sigma), (\sigma', \Sigma'), b) \models R_1 \,\hat{;}\, R_2 \;\text{ iff} \\
\quad \exists \theta, \theta', b_1, b_2.\ ((\sigma, \theta), (\sigma', \theta'), b_1) \models R_1 \wedge ((\theta, \Sigma), (\theta', \Sigma'), b_2) \models R_2 \wedge (b = b_1 \wedge b_2)
\end{array}
$$
$$
\begin{array}{l}
((\sigma, \Sigma), (\sigma', \Sigma'), b) \models R_1 \,\check{;}\, R_2 \;\text{ iff} \\
\quad \exists \theta, \theta', b_1, b_2.\ ((\sigma, \theta), (\sigma', \theta'), b_1) \models R_1 \wedge ((\theta, \Sigma), (\theta', \Sigma'), b_2) \models R_2 \wedge (b = b_1 \vee b_2)
\end{array}
$$

$$\mathsf{Sta}(P, R) \;\text{ iff }\; \forall \sigma, \Sigma, \sigma', \Sigma', b.\ ((\sigma, \Sigma) \models P) \wedge (((\sigma, \Sigma), (\sigma', \Sigma'), b) \models R) \implies ((\sigma', \Sigma') \models P)$$
$$
\begin{array}{l}
\mathsf{Precise}(P) \;\text{ iff }\; \forall \sigma_1, \Sigma_1, \sigma_2, \Sigma_2, \sigma_1', \Sigma_1', \sigma_2', \Sigma_2'. \\
\qquad ((\sigma_1 \uplus \sigma_2 = \sigma_1' \uplus \sigma_2') \wedge ((\sigma_1, \_) \models P) \wedge ((\sigma_1', \_) \models P) \implies (\sigma_1 = \sigma_1')) \\
\qquad \wedge ((\Sigma_1 \uplus \Sigma_2 = \Sigma_1' \uplus \Sigma_2') \wedge ((\_, \Sigma_1) \models P) \wedge ((\_, \Sigma_1') \models P) \implies (\Sigma_1 = \Sigma_1'))
\end{array}
$$
$$I \rhd R \;\text{ iff }\; ([I] \Rightarrow R) \wedge (R \Rightarrow I \ltimes I) \wedge \mathsf{Precise}(I)$$

$$
\begin{array}{l}
\mathsf{MPrecise}(P, Q) \;\text{ iff} \\
\quad \forall \theta_1, \theta_1', \theta_2, \theta_2'.\ (\theta_1 \uplus \theta_2 = \theta_1' \uplus \theta_2') \wedge ((\_, \theta_1) \models P) \wedge ((\theta_1', \_) \models Q) \implies (\theta_1 = \theta_1')
\end{array}
$$

Figure 5: Semantics of assertions (part I).

$$(\mathit{HCState}) \quad \mathbb{D} \quad ::= \quad \mathbb{C} \mid \bullet$$
$$(\mathit{FullState}) \quad \mathcal{S} \quad ::= \quad (\sigma, w, \mathbb{D}, \Sigma) \qquad \text{where } w \in \mathit{Nat}$$

$$\begin{array}{lll}
(\sigma, w, \mathbb{D}, \Sigma) \models P & \text{iff} & (\sigma, \Sigma) \models P \\
(\sigma, w, \mathbb{D}, \Sigma) \models \mathsf{arem}(\mathbb{C}') & \text{iff} & \mathbb{D} = \mathbb{C}' \\
((s, h), w, \mathbb{D}, \Sigma) \models \mathsf{wf}(E) & \text{iff} & \exists n.\ (\llbracket E \rrbracket_s = n) \wedge (n \leq w) \\
(\sigma, w, \mathbb{D}, \Sigma) \models \lfloor p \rfloor_{\mathsf{a}} & \text{iff} & \exists \mathbb{D}'.\ (\sigma, w, \mathbb{D}', \Sigma) \models p \\
(\sigma, w, \mathbb{D}, \Sigma) \models \lfloor p \rfloor_{\mathsf{w}} & \text{iff} & \exists w'.\ (\sigma, w', \mathbb{D}, \Sigma) \models p \\
(\sigma, w, \mathbb{D}, \Sigma) \models p \circledast q & \text{iff} & \exists w_1, w_2, \mathbb{D}_1, \mathbb{D}_2.\ (\sigma, w_1, \mathbb{D}_1, \Sigma) \models p \wedge (\sigma, w_2, \mathbb{D}_2, \Sigma) \models q \\
 & & \wedge (w = w_1 + w_2) \wedge (\mathbb{D} = \mathbb{D}_1 \uplus \mathbb{D}_2) \\
 & & \\
(\sigma, \Sigma) \models \llfloor p \rrfloor & \text{iff} & \exists w, \mathbb{D}.\ (\sigma, w, \mathbb{D}, \Sigma) \models p
\end{array}$$

$$\mathbb{D}_1 \bot \mathbb{D}_2 \quad \text{iff} \quad (\mathbb{D}_1 = \bullet) \vee (\mathbb{D}_2 = \bullet)$$
$$\mathbb{D}_1 \uplus \mathbb{D}_2 \stackrel{\text{def}}{=} \begin{cases} \mathbb{D}_2 & \text{if } \mathbb{D}_1 = \bullet \\ \mathbb{D}_1 & \text{if } \mathbb{D}_2 = \bullet \\ \mathit{undefined} & \text{otherwise} \end{cases}$$
$$(\sigma_1, w_1, \mathbb{D}_1, \Sigma_1) \uplus (\sigma_2, w_2, \mathbb{D}_2, \Sigma_2)$$
$$\stackrel{\text{def}}{=} \begin{cases} (\sigma_1 \uplus \sigma_2, w_1 + w_2, \mathbb{D}_1 \uplus \mathbb{D}_2, \Sigma_1 \uplus \Sigma_1) & \text{if } \sigma_1 \bot \sigma_2, \mathbb{D}_1 \bot \mathbb{D}_2 \text{ and } \Sigma_1 \bot \Sigma_2 \\ \mathit{undefined} & \text{otherwise} \end{cases}$$
$$\mathcal{S} \models p * q \quad \text{iff} \quad \exists \mathcal{S}_1, \mathcal{S}_2.\ (\mathcal{S} = \mathcal{S}_1 \uplus \mathcal{S}_2) \wedge (\mathcal{S}_1 \models p) \wedge (\mathcal{S}_2 \models q)$$

$\mathsf{Sta}(p, R)$ iff

$$\forall \sigma, w, \mathbb{D}, \Sigma, \sigma', \Sigma', b.\ ((\sigma, w, \mathbb{D}, \Sigma) \models p) \wedge (((\sigma, \Sigma), (\sigma', \Sigma'), b) \models R)$$
$$\Longrightarrow \exists w'.\ (\sigma', w', \mathbb{D}, \Sigma') \models p \ \wedge \ (b = \mathbf{false} \Longrightarrow w' = w)$$

Figure 6: Semantics of assertions (part II).

2.2 Definition of RGSim-T

Definition 2 (RGSim-T).
$R, G, I \models \{P\}C \preceq \mathbb{C}\{Q\}$ iff
for all σ and Σ, if $(\sigma, \Sigma) \models P$, then there exists M such that $R, G, I \models (C, \sigma, M) \preceq_Q (\mathbb{C}, \Sigma)$.

Whenever $R, G, I \models (C, \sigma, M) \preceq_Q (\mathbb{C}, \Sigma)$, then $(\sigma, \Sigma) \models I * \mathbf{true}$ and the following are true:

1. for any σ_F, Σ_F, C' and σ'', if $(C, \sigma \uplus \sigma_F) \longrightarrow (C', \sigma'')$ and $\Sigma \bot \Sigma_F$, then there exists σ' such that $\sigma'' = \sigma' \uplus \sigma_F$ and one of the following holds:
   (a) either, there exist M', $\mathbb{C}'$ and Σ' such that $(\mathbb{C}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', M') \preceq_Q (\mathbb{C}', \Sigma')$;
   (b) or, there exists M' such that $M' < M$, $((\sigma, \Sigma), (\sigma', \Sigma), \mathbf{false}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', M') \preceq_Q (\mathbb{C}, \Sigma)$;
2. for any σ_F, Σ_F, e, C' and σ'', if $(C, \sigma \uplus \sigma_F) \xrightarrow{e} (C', \sigma'')$ and $\Sigma \bot \Sigma_F$, then there exist σ', M', $\mathbb{C}'$ and Σ' such that $\sigma'' = \sigma' \uplus \sigma_F$, $(\mathbb{C}, \Sigma \uplus \Sigma_F) \xrightarrow{e}{}^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', M') \preceq_Q (\mathbb{C}', \Sigma')$;
3. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models R^+ * \mathsf{Id}$, then there exists M' such that $R, G, I \models (C, \sigma', M') \preceq_Q (\mathbb{C}, \Sigma')$;
4. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{false}) \models R^+ * \mathsf{Id}$, then $R, G, I \models (C, \sigma', M) \preceq_Q (\mathbb{C}, \Sigma')$;
5. if $C = \mathbf{skip}$, then for any Σ_F, if $\Sigma \bot \Sigma_F$, one of the following holds:
   (a) either, there exists Σ' such that $(\mathbb{C}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbf{skip}, \Sigma' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma, \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma, \Sigma') \models Q$;
   (b) or, $\mathbb{C} = \mathbf{skip}$ and $(\sigma, \Sigma) \models Q$;
6. for any σ_F and Σ_F, if $(C, \sigma \uplus \sigma_F) \longrightarrow \mathbf{abort}$ and $\Sigma \bot \Sigma_F$, then $(\mathbb{C}, \Sigma \uplus \Sigma_F) \longrightarrow^+ \mathbf{abort}$.

Inspired by Vafeiadis [13], we directly embed the framing aspect of separation logic in Def. 2. At each condition, we introduce the frame states σ_F and Σ_F at the target and source levels to represent the remaining parts of the states owned by other threads in the system. The commands C and $\mathbb{C}$ must not change the frame states during their executions.

Technically, we introduce theses σ_F and Σ_F quantifications to admit the frame rules (e.g., the B-FRAME rule in Fig. 7) and the parallel compositionality. Suppose we remove the frame states in Definition 2. Then consider the following example. We can prove

$$\mathsf{Emp}, \mathsf{Emp}, \mathsf{emp} \models \{\mathsf{emp}\}\ ([100] := 1) \preceq ([100] := 2)\ \{\mathsf{emp}\} \tag{2.1}$$

since both programs would abort at empty states. If the frame rule holds, we would get the following by framing $[100] \mapsto 0 \land [100] \Mapsto 0$ to (2.1):

$$\mathsf{Emp}, \mathsf{Emp}, \mathsf{emp} \models \{[100] \mapsto 0 \land [100] \Mapsto 0\}\ ([100] := 1) \preceq ([100] := 2)\ \{[100] \mapsto 0 \land [100] \Mapsto 0\}$$

which obviously does not hold! (In our previous work RGSim [7], the frame rule we provided is more like an invariance rule in Hoare logic. We do not have a real frame rule due to the above reason.) Similar issue also shows up in admitting the parallel compositionality (the B-PAR rule in Fig. 7). The thread t would abort if it accesses the local state of another thread t', while the whole program may not abort with t and t' running in parallel. So we can construct a similar counterexample as (2.1) where the simulation holds for each single thread but fails for the whole program.

Here we address the above issue by embedding the framing aspect directly in the simulation definition, inspired by Vafeiadis [13]. For the simulation in Definition 2 with the σ_F and Σ_F quantifications, the above example (2.1) is no longer satisfied.

3 Logic

Inference rules are shown in Figures 7 and 8.

$$\frac{R, G, I \vdash \{P\}C_1 \preceq \mathbb{C}_1\{P'\} \qquad R, G, I \vdash \{P'\}C_2 \preceq \mathbb{C}_2\{Q\}}{R, G, I \vdash \{P\}C_1; C_2 \preceq \mathbb{C}_1; \mathbb{C}_2\{Q\}} \ (\textsc{b-seq})$$

$$\frac{P \Rightarrow (B \Leftrightarrow \mathbb{B}) * I \qquad R, G, I \vdash \{P \wedge B\}C_1 \preceq \mathbb{C}_1\{Q\} \qquad R, G, I \vdash \{P \wedge \neg B\}C_2 \preceq \mathbb{C}_2\{Q\}}{R, G, I \vdash \{P\}\mathbf{if}\ (B)\ C_1\ \mathbf{else}\ C_2 \preceq \mathbf{if}\ (\mathbb{B})\ \mathbb{C}_1\ \mathbf{else}\ \mathbb{C}_2\{Q\}} \ (\textsc{b-if})$$

$$\frac{P \Rightarrow (B \Leftrightarrow \mathbb{B}) * I \qquad R, G, I \vdash \{P \wedge B\}C \preceq \mathbb{C}\{P\}}{R, G, I \vdash \{P\}\mathbf{while}\ (B)\ C \preceq \mathbf{while}\ (\mathbb{B})\ \mathbb{C}\{P \wedge \neg B\}} \ (\textsc{b-while})$$

$$\frac{\begin{array}{c} R \vee G_2, G_1, I \vdash \{P_1 * P\}C_1 \preceq \mathbb{C}_1\{Q_1 * Q_1'\} \qquad R \vee G_1, G_2, I \vdash \{P_2 * P\}C_2 \preceq \mathbb{C}_2\{Q_2 * Q_2'\} \\ P \vee Q_1' \vee Q_2' \Rightarrow I \qquad I \rhd R \end{array}}{R, G_1 \vee G_2, I \vdash \{P_1 * P_2 * P\}C_1 \parallel C_2 \preceq \mathbb{C}_1 \parallel \mathbb{C}_2\{Q_1 * Q_2 * (Q_1' \wedge Q_2')\}} \ (\textsc{b-par})$$

$$\frac{}{\mathsf{Emp}, \mathsf{Emp}, \mathsf{emp} \vdash \{P\}\mathbf{skip} \preceq \mathbf{skip}\{P\}} \ (\textsc{b-skip}) \qquad \frac{P \Rightarrow (E = \mathbb{E})}{\mathsf{Emp}, \mathsf{Emp}, \mathsf{emp} \vdash \{P\}\mathbf{print}(E) \preceq \mathbf{print}(\mathbb{E})\{P\}} \ (\textsc{b-prt})$$

$$\frac{R, G, I \vdash \{P\}C \preceq \mathbb{C}\{Q\} \qquad G^+ \Rightarrow G \qquad \mathsf{Sta}(P', (R')^+ * \mathsf{Id}) \qquad I' \rhd \{R', G'\} \qquad P' \Rightarrow I' * \mathbf{true}}{R * R', G * G', I * I' \vdash \{P * P'\}C \preceq \mathbb{C}\{Q * P'\}} \ (\textsc{b-frame})$$

$$\frac{\begin{array}{c} R_1, G_1, I_1 \vdash \{P_1\}C \preceq \mathrm{C_M}\{Q_1\} \qquad R_2, G_2, I_2 \vdash \{P_2\}\mathrm{C_M} \preceq \mathbb{C}\{Q_2\} \\ \mathsf{MPrecise}(I_1, I_2) \qquad ((G_1)^+ \mathbin{\hat{\fatsemi}} (G_2)^+) \Rightarrow (G_1 \mathbin{\hat{\fatsemi}} G_2)^+ \qquad (R_1 \mathbin{\check{\fatsemi}} R_2)^+ \Rightarrow ((R_1)^+ \mathbin{\check{\fatsemi}} (R_2)^+) \end{array}}{(R_1 \mathbin{\check{\fatsemi}} R_2), (G_1 \mathbin{\hat{\fatsemi}} G_2), (I_1 \fatsemi I_2) \vdash \{P_1 \fatsemi P_2\}C \preceq \mathbb{C}\{Q_1 \fatsemi Q_2\}} \ (\textsc{trans})$$

$$\frac{R, G, I \vdash \{P \wedge \mathsf{arem}(\mathbb{C})\}C\{Q \wedge \mathsf{arem}(\mathbf{skip})\}}{R, G, I \vdash \{P\}C \preceq \mathbb{C}\{Q\}} \ (\textsc{u2b})$$

Figure 7: Selected binary inference rules.

Definition 3 (Abstract Step "Implication").
$p \stackrel{G}{\Rrightarrow}{}^{+} q$ iff,
for any σ, w, $\mathbb{D}$, Σ and Σ_F, if $(\sigma, w, \mathbb{D}, \Sigma) \models p$ and $\Sigma \perp \Sigma_F$, then
there exist w', $\mathbb{C}'$ and Σ' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$,
$((\sigma, \Sigma), (\sigma, \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma, w', \mathbb{C}', \Sigma') \models q$.

We also define the following syntactic sugars:

$$p \Rrightarrow^+ q \ \text{ iff } \ p \stackrel{\mathsf{Emp}}{\Rrightarrow}{}^{+} q \qquad p \stackrel{G}{\Rrightarrow}{}^{0} q \ \text{ iff } \ p \Rightarrow q \qquad p \Rrightarrow^0 q \ \text{ iff } \ p \Rightarrow q$$
$$p \stackrel{G}{\Rrightarrow}{}^{*} q \ \text{ iff } \ p \stackrel{G}{\Rrightarrow}{}^{+} q \vee p \stackrel{G}{\Rrightarrow}{}^{0} q \qquad p \Rrightarrow^* q \ \text{ iff } \ p \Rrightarrow^+ q \vee p \Rrightarrow^0 q$$

Note that here we introduce the Σ_F quantification similar to Definition 2 for RGSim-T. In our CSL-LICS'14 paper, we simplified the above definition and only defined $p \Rrightarrow^+ q$ to save space. The more general case $p \stackrel{G}{\Rrightarrow}{}^{+} q$ defined here is useful in the A-CONSEQ rule, which is omitted in our CSL-LICS'14 paper.

We prove a few properties of $p \stackrel{G}{\Rrightarrow}{}^{+} q$, as shown in Figure 9. For instance, the first rule says, we can derive $(P \wedge \mathsf{arem}(\mathbb{C})) \Rrightarrow^+ (Q \wedge \mathsf{arem}(\mathbf{skip}) \wedge \mathsf{wf}(E))$ by executing the source code $\mathbb{C}$. And since the source

$$\frac{}{\mathsf{Emp}, \mathsf{Emp}, \mathsf{emp} \vdash \{p\}\mathbf{skip}\{p\}} \ (\textsc{skip}) \qquad \frac{\vdash_{\text{SL}} [p]c[q] \qquad c \text{ is silent}}{\mathsf{Emp}, \mathsf{Emp}, \mathsf{emp} \vdash \{p\}c\{q\}} \ (\textsc{env})$$

$$\frac{\vdash_{\text{SL}} [p]C[q] \qquad (\lfloor\!\lfloor p \rfloor\!\rfloor \ltimes \lfloor\!\lfloor q \rfloor\!\rfloor) \Rightarrow G * \mathsf{True} \qquad I \triangleright G \qquad p \vee q \Rightarrow I * \mathbf{true}}{[I], G, I \vdash \{p\}\langle C \rangle\{q\}} \ (\textsc{atom})$$

$$\frac{\begin{array}{c} p \Rrightarrow^{a} p' \qquad \vdash_{\text{SL}} [p']C[q'] \qquad q' \Rrightarrow^{b} q \qquad + \in \{a, b\} \\ (\lfloor\!\lfloor p \rfloor\!\rfloor \propto \lfloor\!\lfloor q \rfloor\!\rfloor) \Rightarrow G * \mathsf{True} \qquad I \triangleright G \qquad p \vee q \Rightarrow I * \mathbf{true} \end{array}}{[I], G, I \vdash \{p\}\langle C \rangle\{q\}} \ (\textsc{atom}^{+})$$

$$\frac{[I], G, I \vdash \{p\}\langle C \rangle\{q\} \qquad \mathsf{Sta}(\{p, q\}, R * \mathsf{Id}) \qquad I \triangleright R}{R, G, I \vdash \{p\}\langle C \rangle\{q\}} \ (\textsc{atom-r})$$

$$\frac{R, G, I \vdash \{p\}C_1\{p'\} \qquad R, G, I \vdash \{p'\}C_2\{q\}}{R, G, I \vdash \{p\}C_1; C_2\{q\}} \ (\textsc{seq})$$

$$\frac{p \Rightarrow (B = B) * I \qquad p \wedge B \Rightarrow p' * (\mathsf{wf}(1) \wedge \mathsf{emp}) \qquad R, G, I \vdash \{p'\}C\{p\}}{R, G, I \vdash \{p\}\mathbf{while}\ (B)\ C\{p \wedge \neg B\}} \ (\textsc{while})$$

$$\frac{R, G, I \vdash \{p\}C\{q\}}{R, G, I \vdash \{\lfloor p \rfloor_{\mathsf{w}}\}C\{\lfloor q \rfloor_{\mathsf{w}}\}} \ (\textsc{hide-w})$$

$$\frac{R, G, I \vdash \{p\}C\{q\} \qquad \mathsf{Sta}(p', (R')^{+} * \mathsf{Id}) \qquad I' \triangleright \{R', G'\} \qquad p' \Rightarrow I' * \mathbf{true} \qquad G^{+} \Rightarrow G}{R * R', G * G', I * I' \vdash \{p * p'\}C\{q * p'\}} \ (\textsc{frame})$$

$$\frac{R, G, I \vdash \{p\}C\{q\} \qquad \mathsf{Sta}(p', \{R^{+} * \mathsf{Id}, G * \mathsf{True}\})}{R, G, I \vdash \{p \oslash p'\}C\{q \oslash p'\}} \ (\textsc{fr-conj})$$

$$\frac{R, G, I \vdash \{\lfloor p \rfloor_{\mathsf{a}} \wedge \mathsf{arem}(\mathbb{C}_1)\}C\{\lfloor q \rfloor_{\mathsf{a}} \wedge \mathsf{arem}(\mathbb{C}_2)\}}{R, G, I \vdash \{\lfloor p \rfloor_{\mathsf{a}} \wedge \mathsf{arem}(\mathbb{C}_1; \mathbb{C}_3)\}C\{\lfloor q \rfloor_{\mathsf{a}} \wedge \mathsf{arem}(\mathbb{C}_2; \mathbb{C}_3)\}} \ (\textsc{arem})$$

$$\frac{R, G, I \vdash \{p_1\}C\{q_1\} \qquad R, G, I \vdash \{p_2\}C\{q_2\}}{R, G, I \vdash \{p_1 \vee p_2\}C\{q_1 \vee q_2\}} \ (\textsc{disj})$$

$$\frac{p \overset{G}{\Rrightarrow}{}^{*}\ p' \qquad R, G, I \vdash \{p'\}C\{q'\} \qquad q' \overset{G}{\Rrightarrow}{}^{*}\ q \qquad \mathsf{Sta}(\{p, q\}, R * \mathsf{Id}) \qquad p \vee q \Rightarrow I * \mathbf{true}}{R, G, I \vdash \{p\}C\{q\}} \ (\textsc{a-conseq})$$

Figure 8: Selected unary inference rules.

code makes multiple steps, we are allowed to increase the number of tokens ($\mathsf{wf}(E)$). We can also execute the source code in trivial cases, for example, when the source code is $\mathbf{skip};\mathbb{C}$, or it is a while loop but we know for sure the value of the loop condition. In those cases, the step of the source code is an identity transition. Moreover, $p \overset{G}{\Rightarrow}{}^{+} q$ is transitive and we can also have "frame rule" (i.e., local reasoning) over it.

$$\frac{\mathbb{C} \neq \mathbf{skip} \qquad \vdash_{\text{SL}} [P]\mathbb{C}[Q]}{(P \wedge \mathsf{arem}(\mathbb{C})) \Rightarrow^{+} (Q \wedge \mathsf{arem}(\mathbf{skip}) \wedge \mathsf{wf}(E))}$$

$$\frac{P \Rightarrow I * \mathbf{true}}{(P \wedge \mathsf{arem}(\mathbf{skip};\mathbb{C})) \overset{[I]}{\Longrightarrow}{}^{+} (P \wedge \mathsf{arem}(\mathbb{C}) \wedge \mathsf{wf}(E))}$$

$$\frac{P \Rightarrow \mathbb{B} * I}{(P \wedge \mathsf{arem}(\mathbf{if}\ (\mathbb{B})\ \mathbb{C}_1\ \mathbf{else}\ \mathbb{C}_2)) \overset{[I]}{\Longrightarrow}{}^{+} (P \wedge \mathsf{arem}(\mathbb{C}_1) \wedge \mathsf{wf}(E))}$$

$$\frac{P \Rightarrow (\neg\mathbb{B}) * I}{(P \wedge \mathsf{arem}(\mathbf{if}\ (\mathbb{B})\ \mathbb{C}_1\ \mathbf{else}\ \mathbb{C}_2)) \overset{[I]}{\Longrightarrow}{}^{+} (P \wedge \mathsf{arem}(\mathbb{C}_2) \wedge \mathsf{wf}(E))}$$

$$\frac{P \Rightarrow \mathbb{B} * I}{(P \wedge \mathsf{arem}(\mathbf{while}\ (\mathbb{B})\ \mathbb{C})) \overset{[I]}{\Longrightarrow}{}^{+} (P \wedge \mathsf{arem}(\mathbb{C};\mathbf{while}\ (\mathbb{B})\ \mathbb{C}) \wedge \mathsf{wf}(E))}$$

$$\frac{P \Rightarrow (\neg\mathbb{B}) * I}{(P \wedge \mathsf{arem}(\mathbf{while}\ (\mathbb{B})\ \mathbb{C})) \overset{[I]}{\Longrightarrow}{}^{+} (P \wedge \mathsf{arem}(\mathbf{skip}) \wedge \mathsf{wf}(E))}$$

$$\frac{(P \wedge \mathsf{arem}(\mathbb{C}_1)) \overset{G}{\Rightarrow}{}^{+} (Q \wedge \mathsf{arem}(\mathbb{C}_2) \wedge \mathsf{wf}(E))}{(P \wedge \mathsf{arem}(\mathbb{C}_1;\mathbb{C}_3)) \overset{G}{\Rightarrow}{}^{+} (Q \wedge \mathsf{arem}(\mathbb{C}_2;\mathbb{C}_3) \wedge \mathsf{wf}(E))}$$

$$\frac{p \overset{G}{\Rightarrow}{}^{+} p' \qquad p' \overset{G}{\Rightarrow}{}^{+} q \qquad I \rhd G}{p \overset{G}{\Rightarrow}{}^{+} q} \qquad \frac{p \Rightarrow p' \qquad p' \overset{G'}{\Longrightarrow}{}^{+} q' \qquad q' \Rightarrow q \qquad G' \Rightarrow G}{p \overset{G}{\Rightarrow}{}^{+} q}$$

$$\frac{p_1 \overset{G}{\Rightarrow}{}^{+} q_1 \qquad p_2 \overset{G}{\Rightarrow}{}^{+} q_2}{(p_1 \vee p_2) \overset{G}{\Rightarrow}{}^{+} (q_1 \vee q_2)} \qquad \frac{p \overset{G}{\Rightarrow}{}^{+} q}{(p * p') \overset{G}{\Rightarrow}{}^{+} (q * p')}$$

Figure 9: Properties of $p \overset{G}{\Rightarrow} q$.

Below we discuss some interesting rules which are not shown in our CSL-LICS'14 paper due to the space limit. The binary rules are very similar to those in our previous work RGSim [7]. The TRANS rule shows the transitivity of our RGSim-T relation.

For the unary rules in Figure 8, in addition to rules for atomic blocks, we have SKIP and ENV rules to reason about **skip** and primitive instructions. Here we assume the unary logic handles only programs which do not produce external events (e.g., the ENV rule has a side condition saying that "c is silent"). For commands producing events, such as the print command, we require *lockstep* at the target and source levels and prove such refinement using the binary inference rules (e.g., the B-PRT rule in Figure 7). It is also possible to extend the current unary logic with assertions for event traces and provide unary rules to reason about commands with events. Note that although the shared resource is empty in the SKIP and ENV rules, we can derive rules allowing resource sharing from them and the FRAME rule in Figure 8.

In addition to the rules for while loops as in the CSL-LICS'14 paper, we also have unary rules for sequential composition (the SEQ rule in Figure 8) and for if-then-else composition (omitted here), both of which are in the same forms as in LRG [2]. The unary FRAME rule is similar to the binary one in Figure 7. It is also in the same form as in LRG [2].

The FR-CONJ rule is like the frame rule in RGSep [12]. The frame p' may specify the number of tokens used by the context of the code C, i.e., the code C does not consume these tokens in p'. The frame p' may also specify the shared concrete and abstract states (and the case usually occurs when the number of tokens depends on the concrete and abstract states). So we use the new operator $\circledwedge$ to ensure that the concrete and abstract states specified in p and p' coincide.

The AREM rule is like a frame rule over source code. It allows us to reason about refinement using "local" source code, i.e., source code which is really refined by the target.

The A-CONSEQ rule allows us to execute the source code outside of an atomic block. It requires that the transitions of the source code over the shared states satisfy G^+, but it is usually used when the steps are simply identity transitions. For instance, we can use the rule to unfold a while loop at the source at any time in a refinement proof (we do not have to be in an atomic block of the target code). When $p \overset{G}{\Rrightarrow}{}^* p'$ and $q' \overset{G}{\Rrightarrow}{}^* q$ are $p \Rrightarrow p'$ and $q' \Rrightarrow q$ respectively, this rule becomes the normal CONSEQ rule (see RGSep [12] and LRG [2]).

We can also *derive* the following WHILE-TERM rule from the WHILE rule. The derivation is shown in Section 5.

$$\frac{\begin{array}{c} R, G, I \vdash \{p \wedge B \wedge (E = \alpha)\}C\{p \wedge (E < \alpha)\} \qquad p \wedge B \Rightarrow E > 0 \\ p \Rightarrow ((B = B) \wedge (E = E)) * I \qquad G^+ \Rightarrow G \qquad \alpha \text{ is a fresh logical variable} \end{array}}{R, G, I \vdash \{\lfloor p \rfloor_{\mathsf{w}}\}\textbf{while } (B)\ C\{\lfloor p \rfloor_{\mathsf{w}} \wedge \neg B\}} \ (\text{WHILE-TERM})$$

The WHILE-TERM rule is similar to a total correctness while rule (e.g., see [10]). In every round of the loop, the loop variant E decreases (but should always be positive). We can verify refinement for such a locally-terminating loop (a loop that always terminates regardless of environment steps) without specifying tokens. To derive this rule, we actually need to introduce the number of tokens as an auxiliary state for the loop iterations and relate it to the loop variant E in the real state.

Soundness of the logic is proved in Section 5 (where we also define the unary judgment semantics).

4 Examples

In this section, we verify the examples claimed in our CSL-LICS'14 paper (see Figure 10). To simplify the presentation of the proofs, assume we always have the ownerships of program variables.

Linearizability & Lock-Freedom	Counter and its variants
	Treiber stack
	Michael-Scott lock-free queue [8]
	DGLM lock-free queue [1]
Non-Atomic Object Correctness	Synchronous queue [9]
Correctness of Optimized Algo (Equivalence)	Counter vs. its variants
	TAS lock vs. TTAS lock [3]

Figure 10: Verified examples using our logic.

4.1 Counter and Its Variants

In Figure 11, we show four possible implementations of the counter. Though they are quite simple, they illustrate different choices that programmers may make to implement a concurrent object. The abstract atomic `INC` operation is shown below:

```
INC() {   X := X + 1;  }
```

```
inc() {
 local t, b;
 b := false;
 while (!b) {
   < t := x; >
   b := cas(&x, t, t+1);
 }
}
```

```
inc'() {
 local t, b;
 b := false;
 < t := x; >
 while (!b) {
   b := cas(&x, t, t+1);
   < t := x; >
 }
}
```

```
incOpt() {
 local t, b, b';
 b := false;
 while (!b) {
   b' := false;
   while (!b') {
     < t := x; >
     < b' := (t = x); >
   }
   b := cas(&x, t, t+1);
 }
}
```

```
incOpt'() {
 local t, b, b';
 b := false;
 while (!b) {
   < t := x; >
   < b' := (t = x); >
   while (!b') {
     < t := x; >
     < b' := (t = x); >
   }
   b := cas(&x, t, t+1);
 }
}
```

Figure 11: Various implementations of counter.

Below we first verify that each implementation C of the counter is correct w.r.t. to `INC`. Here correctness refer to linearizability and lock-freedom together. As explained in the submitted paper, we only need to prove the following in our logic:

$$R, G, I \vdash \{I\}\ C \preceq \texttt{INC}\ \{I\}$$

where R and G specify the possible actions (i.e., increments) on the well-formed shared data structure (i.e., counter) fenced by I. In all these examples, they share the same R, G and I as follows:

$$I \stackrel{\text{def}}{=} (\mathtt{x} = \mathtt{X}) \qquad\qquad R = G \stackrel{\text{def}}{=} (I \propto I) \vee [I]$$

By the U2B rule, the above is reduced to proving the following unary judgment:

$$R, G, I \vdash \{I \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1)\} C \{I \wedge \mathsf{arem}(\mathbf{skip})\}$$

The proofs are shown in Figures 12, 13, 14 and 15.

We can also prove the equivalence between `incOpt` and `inc`. That is, we prove:

$$R, G, I \vdash \{I\}\ \mathtt{incOpt} \preceq \mathtt{inc}\ \{I\} \qquad \text{and} \qquad R, G, I \vdash \{I\}\ \mathtt{inc} \preceq \mathtt{incOpt}\ \{I\}$$

Here we use the same R, G and I as above (always use `x` at the left side and `X` at the right side). The proofs are shown in Figures 17 and 18. The equivalence between `incOpt'` and `inc` is similar.

`inc() {`
` local t, b;`
$\{I \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1)\}$
` b := false;`
$\{(\neg\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1)) \ \vee\ (\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathbf{skip}))\}$ //Applying the WHILE rule and the HIDE-W rule
` while (!b) {`
$\{\neg\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1) \wedge \mathsf{wf}(0)\}$
$\{\mathtt{x} = \mathtt{X}\} * (\mathsf{emp} \wedge \neg\mathtt{b} \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1) \wedge \mathsf{wf}(0))$ //Applying the FRAME rule
`   < t := x; >`
$\{(\mathtt{x} = \mathtt{X} = \mathtt{t}) \vee ((\mathtt{x} = \mathtt{X} \neq \mathtt{t}) \wedge \mathsf{wf}(1))\} * (\mathsf{emp} \wedge \neg\mathtt{b} \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1) \wedge \mathsf{wf}(0))$
$\left\{\begin{array}{l}(\neg\mathtt{b} \wedge (\mathtt{x} = \mathtt{X} = \mathtt{t}) \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1) \wedge \mathsf{wf}(0)) \\ \quad \vee (\neg\mathtt{b} \wedge (\mathtt{x} = \mathtt{X} \neq \mathtt{t}) \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1) \wedge \mathsf{wf}(1))\end{array}\right\}$
`   b := cas(&x, t, t+1);`
$\{(\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathbf{skip}) \wedge \mathsf{wf}(1)) \ \vee\ (\neg\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1) \wedge \mathsf{wf}(1))\}$
` }`
$\{I \wedge \mathsf{arem}(\mathbf{skip})\}$
`}`

Figure 12: Proving `inc` refines `INC`.

`inc'() {`
` local t, b;`
$\{I \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1)\}$
` b := false;`
$\{\neg\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1)\}$
` < t := x; >`
$\left\{\begin{array}{l}(\neg\mathtt{b} \wedge (\mathtt{x} = \mathtt{X} = \mathtt{t}) \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1)) \\ \quad \vee (\neg\mathtt{b} \wedge (\mathtt{x} = \mathtt{X} \neq \mathtt{t}) \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1)) \\ \quad \vee (\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathbf{skip}))\end{array}\right\}$ //Applying the WHILE rule and the HIDE-W rule
` while (!b) {`
$\{(\neg\mathtt{b} \wedge (\mathtt{x} = \mathtt{X} = \mathtt{t}) \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1) \wedge \mathsf{wf}(0)) \vee (\neg\mathtt{b} \wedge (\mathtt{x} = \mathtt{X} \neq \mathtt{t}) \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1) \wedge \mathsf{wf}(1))\}$
`   b := cas(&x, t, t+1);`
$\{(\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathbf{skip}) \wedge \mathsf{wf}(1)) \ \vee\ (\neg\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1) \wedge \mathsf{wf}(1))\}$
`   < t := x; >`
$\left\{\begin{array}{l}(\neg\mathtt{b} \wedge (\mathtt{x} = \mathtt{X} = \mathtt{t}) \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1) \wedge \mathsf{wf}(1)) \\ \quad \vee (\neg\mathtt{b} \wedge (\mathtt{x} = \mathtt{X} \neq \mathtt{t}) \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1) \wedge \mathsf{wf}(2)) \\ \quad \vee (\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathbf{skip}) \wedge \mathsf{wf}(1))\end{array}\right\}$
` }`
$\{I \wedge \mathsf{arem}(\mathbf{skip})\}$
`}`

Figure 13: Proving `inc'` refines `INC`.

```
incOpt() {
 local t, b, b';
 {I ∧ arem(X := X + 1)}
 b := false;
 {(¬b ∧ I ∧ arem(X := X + 1)) ∨ (b ∧ I ∧ arem(skip))}     //Applying the WHILE rule and the HIDE-W rule
 while (!b) {
   {¬b ∧ I ∧ arem(X := X + 1) ∧ wf(1)}
   {(x = X) ∧ wf(1)}  ∗ (emp ∧ ¬b ∧ arem(X := X + 1))      //Applying the FRAME rule
   b' := false;
   {(¬b' ∧ (x = X) ∧ wf(1)) ∨ (b' ∧ (x = X = t)) ∨ (b' ∧ (x = X ≠ t) ∧ wf(2))}     //Applying the WHILE rule
   while (!b') {
     {(x = X) ∧ wf(0)}
     < t := x; >
     {(x = X = t) ∨ ((x = X ≠ t) ∧ wf(1))}
     < b' := (t = x); >
     {(b' ∧ (x = X = t)) ∨ (b' ∧ (x = X ≠ t) ∧ wf(2)) ∨ (¬b' ∧ (x = X ≠ t) ∧ wf(1))}
   }
   {(x = X = t) ∨ ((x = X ≠ t) ∧ wf(2))}  ∗ (emp ∧ ¬b ∧ arem(X := X + 1))
   {(¬b ∧ (x = X = t) ∧ arem(X := X + 1) ∧ wf(0))
     ∨ (¬b ∧ (x = X ≠ t) ∧ arem(X := X + 1) ∧ wf(2))}
   b := cas(&x, t, t+1);
   {(b ∧ I ∧ arem(skip) ∧ wf(1)) ∨ (¬b ∧ I ∧ arem(X := X + 1) ∧ wf(2))}
 }
 {I ∧ arem(skip)}
}
```

Figure 14: Proving `incOpt` refines `INC`.

```
incOpt'() {
 local t, b, b';
 {I ∧ arem(X := X + 1)}
 b := false;
 {(¬b ∧ I ∧ arem(X := X + 1)) ∨ (b ∧ I ∧ arem(skip))}     //Applying the WHILE rule and the HIDE-W rule
 while (!b) {
   {¬b ∧ I ∧ arem(X := X + 1) ∧ wf(0)}
   {x = X}  ∗ (emp ∧ ¬b ∧ arem(X := X + 1) ∧ wf(0))      //Applying the FRAME rule
   < t := x; >
   {(x = X = t) ∨ ((x = X ≠ t) ∧ wf(1))}
   < b' := (t = x); >
   {(b' ∧ (x = X = t)) ∨ ((x = X ≠ t) ∧ wf(1))}     //Applying the WHILE rule
   while (!b') {
     {(x = X) ∧ wf(0)}
     < t := x; >
     {(x = X = t) ∨ ((x = X ≠ t) ∧ wf(1))}
     < b' := (t = x); >
     {(b' ∧ (x = X = t)) ∨ ((x = X ≠ t) ∧ wf(1))}
   }
   {(x = X = t) ∨ ((x = X ≠ t) ∧ wf(1))}  ∗ (emp ∧ ¬b ∧ arem(X := X + 1) ∧ wf(0))
   {(¬b ∧ (x = X = t) ∧ arem(X := X + 1) ∧ wf(0))
     ∨ (¬b ∧ (x = X ≠ t) ∧ arem(X := X + 1) ∧ wf(1))}
   b := cas(&x, t, t+1);
   {(b ∧ I ∧ arem(skip) ∧ wf(1)) ∨ (¬b ∧ I ∧ arem(X := X + 1) ∧ wf(1))}
 }
 {I ∧ arem(skip)}
}
```

Figure 15: Proving `incOpt'` refines `INC`.

$I \stackrel{\text{def}}{=} (\mathtt{x} = \mathtt{X})$

$R = G \stackrel{\text{def}}{=} (\exists n.\ (\mathtt{x} = \mathtt{X} = n) \propto (\mathtt{x} = \mathtt{X} > n)) \vee [I]$

```
incOpt'() {
 local t, b;
```
$\{I \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X} + 1)\}$
```
 b := false;
```
$\{(\neg\mathsf{b} \wedge I \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1)) \ \vee \ (\mathsf{b} \wedge I \wedge \mathsf{arem}(\mathbf{skip}))\}$ //Applying the WHILE rule and the HIDE-W rule
```
 while (!b) {
```
$\{\neg\mathsf{b} \wedge I \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1) \wedge \mathsf{wf}(0)\}$

$\{\mathtt{x} = \mathtt{X}\} \ * (\mathsf{emp} \wedge \neg\mathsf{b} \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1) \wedge \mathsf{wf}(0))$ //Applying the FRAME rule
```
  < t := x; >
```
$\{(\mathtt{x} = \mathtt{X} = \mathtt{t} = \alpha) \vee ((\mathtt{x} = \mathtt{X} > \alpha) \wedge (\mathtt{t} = \alpha) \wedge \mathsf{wf}(1))\}$
```
  < b' := (t = x); >
```
$\{(\mathtt{b'} \wedge (\mathtt{x} = \mathtt{X} = \mathtt{t} = \alpha)) \vee ((\mathtt{x} = \mathtt{X} > \alpha) \wedge (\mathtt{t} = \alpha) \wedge \mathsf{wf}(1))\}$

$\{(\mathtt{b'} \wedge (\mathtt{x} = \mathtt{X} = \mathtt{t} = \alpha)) \vee (\mathtt{x} = \mathtt{X} > \alpha)\} \ \oslash ((\mathtt{x} = \mathtt{X} = \alpha) \vee (\mathtt{x} = \mathtt{X} > \alpha) \wedge \mathsf{wf}(1))$

//Applying the FR-CONJ rule //Applying the WHILE rule and the HIDE-W rule
```
  while (!b') {
```
$\{(\mathtt{x} = \mathtt{X} > \alpha) \wedge \mathsf{wf}(0)\}$
```
   < t := x; >
```
$\{(\mathtt{x} = \mathtt{X} = \mathtt{t} > \alpha) \vee ((\mathtt{x} = \mathtt{X} > \mathtt{t} > \alpha) \wedge \mathsf{wf}(1))\}$
```
   < b' := (t = x); >
```
$\{(\mathtt{b'} \wedge (\mathtt{x} = \mathtt{X} = \mathtt{t} > \alpha)) \vee ((\mathtt{x} = \mathtt{X} > \mathtt{t} > \alpha) \wedge \mathsf{wf}(1))\}$

$\{(\mathtt{b'} \wedge (\mathtt{x} = \mathtt{X} = \mathtt{t} \geq \alpha)) \vee ((\mathtt{x} = \mathtt{X} > \alpha) \wedge \mathsf{wf}(1))\}$
```
  }
```
$\{(\mathtt{x} = \mathtt{X} = \mathtt{t} = \alpha) \vee (\mathtt{x} = \mathtt{X} > \alpha)\} \ \oslash ((\mathtt{x} = \mathtt{X} = \alpha) \vee (\mathtt{x} = \mathtt{X} > \alpha) \wedge \mathsf{wf}(1))$

$\{(\mathtt{x} = \mathtt{X} = \mathtt{t} = \alpha) \vee ((\mathtt{x} = \mathtt{X} > \alpha) \wedge \mathsf{wf}(1))\}$

$\{(\mathtt{x} = \mathtt{X} = \mathtt{t}) \vee ((\mathtt{x} = \mathtt{X} \neq \mathtt{t}) \wedge \mathsf{wf}(1))\} \ * (\mathsf{emp} \wedge \neg\mathsf{b} \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1) \wedge \mathsf{wf}(0))$

$$\left\{\begin{array}{l}(\neg\mathsf{b} \wedge (\mathtt{x} = \mathtt{X} = \mathtt{t}) \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1) \wedge \mathsf{wf}(0)) \\ \ \vee (\neg\mathsf{b} \wedge (\mathtt{x} = \mathtt{X} \neq \mathtt{t}) \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1) \wedge \mathsf{wf}(1))\end{array}\right\}$$
```
  b := cas(&x, t, t+1);
```
$\{(\mathsf{b} \wedge I \wedge \mathsf{arem}(\mathbf{skip}) \wedge \mathsf{wf}(1)) \ \vee \ (\neg\mathsf{b} \wedge I \wedge \mathsf{arem}(\mathtt{X} := \mathtt{X}+1) \wedge \mathsf{wf}(1))\}$
```
 }
```
$\{I \wedge \mathsf{arem}(\mathbf{skip})\}$
```
}
```

Figure 16: Proving `incOpt'` refines `INC` (an alternative approach by using the FR-CONJ rule). α is a logical variable.

`inc` $\stackrel{\text{def}}{=}$ (`B := false; incLoop;`)

`incLoop` $\stackrel{\text{def}}{=}$ (`while(!B) { <T:=X>; incCas; }`)

`incCas` $\stackrel{\text{def}}{=}$ (`B := cas(&X, T, T+1);`)

```
incOpt() {
 local t, b, b';
```
$\{I \wedge \mathsf{arem}(\mathtt{inc})\}$
```
 b := false;
```
$\{(\neg\mathtt{b} \wedge \neg\mathtt{B} \wedge I \wedge \mathsf{arem}(\mathtt{incLoop})) \ \vee \ (\mathtt{b} \wedge \mathtt{B} \wedge I \wedge \mathsf{arem}(\mathbf{skip}))\}$

//Applying the WHILE rule and the HIDE-W rule
```
 while (!b) {
```
$\{\neg\mathtt{b} \wedge \neg\mathtt{B} \wedge I \wedge \mathsf{arem}(\mathtt{incLoop}) \wedge \mathsf{wf}(0)\}$
```
   b' := false;
```
$$\left\{\begin{array}{l}(\neg\mathtt{b'} \wedge \neg\mathtt{b} \wedge \neg\mathtt{B} \wedge (\mathtt{x} = \mathtt{X}) \wedge \mathsf{arem}(\mathtt{incLoop}) \wedge \mathsf{wf}(0)) \\ \ \vee (\mathtt{b'} \wedge \neg\mathtt{b} \wedge \neg\mathtt{B} \wedge (\mathtt{x} = \mathtt{X}) \wedge (\mathtt{t} = \mathtt{T}) \wedge \mathsf{arem}(\mathtt{incCas;incLoop}) \wedge \mathsf{wf}(0))\end{array}\right\}$$

//Applying the WHILE rule and the HIDE-W rule
```
   while (!b') {
```
$\{\neg\mathtt{b'} \wedge \neg\mathtt{b} \wedge \neg\mathtt{B} \wedge (\mathtt{x} = \mathtt{X}) \wedge \mathsf{arem}(\mathtt{incLoop}) \wedge \mathsf{wf}(0)\}$

$\{\neg\mathtt{b} \wedge \neg\mathtt{B} \wedge (\mathtt{x} = \mathtt{X}) \wedge \mathsf{arem}(\mathtt{<T:=X>}; \mathtt{incCas}; \mathtt{incLoop}) \wedge \mathsf{wf}(1)\}$
```
     < t := x; >
```
$\{\neg\mathtt{b} \wedge \neg\mathtt{B} \wedge (\mathtt{x} = \mathtt{X}) \wedge (\mathtt{t} = \mathtt{T}) \wedge \mathsf{arem}(\mathtt{incCas}; \mathtt{incLoop}) \wedge \mathsf{wf}(1)\}$
```
     < b' := (t = x); >
```
$$\left\{\begin{array}{l}(\neg\mathtt{b'} \wedge \neg\mathtt{b} \wedge \neg\mathtt{B} \wedge (\mathtt{x} = \mathtt{X}) \wedge \mathsf{arem}(\mathtt{incLoop}) \wedge \mathsf{wf}(1)) \\ \ \vee (\mathtt{b'} \wedge \neg\mathtt{b} \wedge \neg\mathtt{B} \wedge (\mathtt{x} = \mathtt{X}) \wedge (\mathtt{t} = \mathtt{T}) \wedge \mathsf{arem}(\mathtt{incCas;incLoop}) \wedge \mathsf{wf}(1))\end{array}\right\}$$
```
   }
```
$\{\mathtt{b'} \wedge (\mathtt{x} = \mathtt{X}) \wedge (\mathtt{t} = \mathtt{T}) \wedge \mathsf{arem}(\mathtt{incCas;incLoop}) \wedge \mathsf{wf}(0)\}$
```
   b := cas(&x, t, t+1);
```
$\{(\mathtt{b} = \mathtt{B}) \wedge I \wedge \mathsf{arem}(\mathtt{incLoop}) \wedge \mathsf{wf}(1)\}$

$\{(\mathtt{b} \wedge \mathtt{B} \wedge I \wedge \mathsf{arem}(\mathbf{skip})) \ \vee \ (\neg\mathtt{b} \wedge \neg\mathtt{B} \wedge I \wedge \mathsf{arem}(\mathtt{incLoop}) \wedge \mathsf{wf}(1))\}$
```
 }
```
$\{I \wedge \mathsf{arem}(\mathbf{skip})\}$
```
}
```

Figure 17: Proving `incOpt` refines `inc`.

$\texttt{incOpt} \stackrel{\text{def}}{=} (\texttt{B := false; incOptLoop;})$

$\texttt{incOptLoop} \stackrel{\text{def}}{=} (\texttt{while(!B) \{ incOptInner; incCas; \}})$

$\texttt{incOptInner} \stackrel{\text{def}}{=} (\texttt{B':=false; while(!B') \{ <T:=X>; <B':=(T=X)>; \}})$

$\texttt{incCas} \stackrel{\text{def}}{=} (\texttt{B := cas(\&X, T, T+1);})$

```
inc() {
 local t, b;
```

$\{ I \land \mathsf{arem}(\texttt{incOpt}) \}$

```
 b := false;
```

$\{ (\neg \mathsf{b} \land \neg \mathsf{B} \land I \land \mathsf{arem}(\texttt{incOptLoop})) \;\lor\; (\mathsf{b} \land \mathsf{B} \land I \land \mathsf{arem}(\mathbf{skip})) \}$

//Applying the WHILE rule and the HIDE-W rule

```
 while (!b) {
```

$\{ \neg \mathsf{b} \land \neg \mathsf{B} \land I \land \mathsf{arem}(\texttt{incOptLoop}) \land \mathsf{wf}(0) \}$

```
   < t := x; >
```

$\{ \neg \mathsf{b} \land \neg \mathsf{B} \land (\mathtt{x} = \mathtt{X}) \land (\mathtt{t} = \mathtt{T}) \land \mathsf{arem}(\texttt{incCas}; \texttt{incOptLoop}) \land \mathsf{wf}(1) \}$

```
   b := cas(&x, t, t+1);
```

$\{ (\mathsf{b} = \mathsf{B}) \land I \land \mathsf{arem}(\texttt{incOptLoop}) \land \mathsf{wf}(1) \}$

```
 }
```

$\{ I \land \mathsf{arem}(\mathbf{skip}) \}$

```
}
```

Figure 18: Proving `inc` refines `incOpt`.

4.2 TAS Lock and TTAS Lock

```
lock() {
 local b, b’;
 b := true;
 while (b) {
   < b’ := l; >
   while (b’) {
     < b’ := l; >
   }
   b := getAndSet(&l, true);
 }
}

unlock() {
 < l := false; >
}
```

```
LOCK() {
 local B;
 B := getAndSet(&L, true);
 while (B) {
   B := getAndSet(&L, true);
 }
}

UNLOCK() {
 < L := false; >
}
```

Figure 19: `TTASLock` (the left) and `TASLock` (the right).

In Figure 19, we show the implementations of TTAS lock and TAS lock [3]. We can prove the equivalence between these two implementations. That is, we prove:

$$R, G, I \vdash \{I\}\ \texttt{lock} \preceq \texttt{LOCK}\ \{I\} \quad \text{and} \quad R, G, I \vdash \{I\}\ \texttt{LOCK} \preceq \texttt{lock}\ \{I\}$$
$$R, G, I \vdash \{I\}\ \texttt{unlock} \preceq \texttt{UNLOCK}\ \{I\} \quad \text{and} \quad R, G, I \vdash \{I\}\ \texttt{UNLOCK} \preceq \texttt{unlock}\ \{I\}$$

As in the example of counters, R and G specify the possible actions on the well-formed shared data structure fenced by I. Here R, G and I can be defined as follows:

$$I \stackrel{\text{def}}{=} (\texttt{l} = \texttt{L}) \qquad\qquad R = G \stackrel{\text{def}}{=} (I \propto I) \vee [I]$$

The proofs for the refinements between `unlock` and `UNLOCK` are straightforward since their code is the same. We show the proofs for the refinements between `lock` and `LOCK` in Figures 20 and 21.

$\texttt{GAS} \stackrel{\text{def}}{=} (\texttt{B := getAndSet(\&L, true)})$

$\texttt{LoopGAS} \stackrel{\text{def}}{=} (\texttt{while(B) GAS;})$

```
lock() {
 local b, b';
 {I ∧ arem(LOCK)}
 b := true;
 {(b ∧ I ∧ arem(GAS; LoopGAS)) ∨ (¬b ∧ I ∧ arem(skip))}     //Applying the WHILE rule and the HIDE-W rule
 while (b) {
   {b ∧ I ∧ arem(GAS; LoopGAS) ∧ wf(0)}
   < b' := l; >
   {(b ∧ b' ∧ B ∧ I ∧ arem(LoopGAS) ∧ wf(1)) ∨ (b ∧ ¬b' ∧ I ∧ arem(GAS; LoopGAS) ∧ wf(0))}
   {b ∧ I ∧ arem(GAS; LoopGAS)}     //Applying the WHILE rule and the HIDE-W rule
   while (b') {
     {b ∧ I ∧ arem(GAS; LoopGAS) ∧ wf(0)}
     < b' := l; >
     {(b ∧ b' ∧ B ∧ I ∧ arem(LoopGAS) ∧ wf(1)) ∨ (b ∧ ¬b' ∧ I ∧ arem(GAS; LoopGAS) ∧ wf(0))}
     {(b ∧ b' ∧ I ∧ arem(GAS; LoopGAS) ∧ wf(1)) ∨ (b ∧ ¬b' ∧ I ∧ arem(GAS; LoopGAS) ∧ wf(0))}
   }
   {b ∧ I ∧ arem(GAS; LoopGAS) ∧ wf(0)}
   b := getAndSet(&l, true);
   {(b = B) ∧ I ∧ arem(LoopGAS) ∧ wf(1)}
   {(¬b ∧ I ∧ arem(skip) ∧ wf(1)) ∨ (b ∧ I ∧ arem(GAS; LoopGAS) ∧ wf(1))}
 }
 {I ∧ arem(skip)}
}
```

Figure 20: Proving TTASLock refines TASLock.

$\texttt{loopTTAS} \stackrel{\text{def}}{=} (\texttt{while(b) \{...\}})$

```
LOCK() {
 local B;
 {I ∧ arem(lock)}
 B := getAndSet(&L, true);
 {(b = B) ∧ I ∧ arem(loopTTAS) ∧ wf(1)}
 {(b = B) ∧ I ∧ arem(loopTTAS)}     //Applying the WHILE rule and the HIDE-W rule
 while (B) {
   {b ∧ B ∧ I ∧ arem(loopTTAS) ∧ wf(0)}
   B := getAndSet(&L, true);
   {(b = B) ∧ I ∧ arem(loopTTAS) ∧ wf(1)}
 }
 {¬b ∧ ¬B ∧ I ∧ arem(loopTTAS)}
 {I ∧ arem(skip)}
}
```

Figure 21: Proving TASLock refines TTASLock.

4.3 Treiber Stack

```
push(v) {
 local x, t, b;
 b := false;
 x := cons(v, null);
 while (!b) {
   < t := S; >
   x.next := t;
   b := cas(&S, t, x);
 }
}
```

```
pop() {
 local v, x, t, b;
 b := false;
 while (!b) {
   < t := S; >
   if (t = null) {
     v := EMPTY;
     b := true;
   } else {
     v := t.data;
     x := t.next;
     b := cas(&S, t, x);
   }
 }
 return v;
}
```

```
PUSH(V) {
 < Stk := V :: Stk; >
}

POP() {
 local V;
 < if (Stk = ϵ) {
     V := EMPTY;
   } else {
     V := head(Stk);
     Stk := tail(Stk);
   }
 >
 return V;
}
```

Figure 22: Treiber stack.

In Figure 22, we show the implementation of Treiber stack (at the left of the figure), and the abstract atomic operations (at the right). The abstract PUSH and POP operations manipulate an abstract mathematical list Stk, and when popping from an empty stack, POP returns EMPTY.

Below we use our logic to prove the linearizability and lock-freedom together of Treiber stack. As explained in the submitted paper, we only need to prove the following in our logic:

$$R, G, I \vdash \{I \wedge (\mathtt{v} = \mathtt{V})\}\ \mathtt{push(v)} \preceq \mathtt{PUSH(V)}\ \{I\} \qquad \text{and} \qquad R, G, I \vdash \{I\}\ \mathtt{pop} \preceq \mathtt{POP}\ \{I \wedge (\mathtt{v} = \mathtt{V})\}$$

By the U2B rule, the above is reduced to proving the following unary judgment:

$$R, G, I \vdash \{I \wedge \mathsf{arem}(\mathtt{PUSH(V)}) \wedge (\mathtt{v} = \mathtt{V})\}\ \mathtt{push(v)}\ \{I \wedge \mathsf{arem}(\boldsymbol{skip})\}$$
$$\text{and} \qquad R, G, I \vdash \{I \wedge \mathsf{arem}(\mathtt{POP})\}\ \mathtt{pop}\ \{I \wedge \mathsf{arem}(\boldsymbol{skip}) \wedge (\mathtt{v} = \mathtt{V})\}$$

We define the precise invariant I, the rely R and the guarantee G in Figure 23. The invariant I in Figure 23 maps the value sequence A of the concrete list pointed to by S (denoted by $(\mathtt{S} = x) * \mathsf{ls}(x, A, \mathtt{null})$) to the abstract stack Stk. To ensure there is no "ABA" problem [3], we follow Turon and Wand [11] and introduce a write-only auxiliary variable GN to remember the nodes which used to be on the stack but no longer are. The precise invariant for shared states should include those garbage nodes (garb). GN does not affect the behaviors of the implementation and is introduced for verification only.

$$
\begin{array}{l}
I \stackrel{\text{def}}{=} \exists x, A.\ (\mathtt{Stk} = A) \wedge (\mathtt{S} = x) * \mathsf{ls}(x, A, \mathtt{null}) * \mathsf{garb} \\
\mathsf{node}(x, v, y) \stackrel{\text{def}}{=} x \mapsto (v, y) \qquad\qquad \mathsf{node}(x) \stackrel{\text{def}}{=} \mathsf{node}(x, \_, \_) \\
\mathsf{ls}(x, A, y) \stackrel{\text{def}}{=} (x = y \wedge A = \epsilon \wedge emp) \vee (x \neq y \wedge \exists z, v, A'.\ A = v :: A' \wedge \mathsf{node}(x, v, z) * \mathsf{ls}(z, A', y)) \\
\mathsf{ls}(x, y) \stackrel{\text{def}}{=} \exists A.\ \mathsf{ls}(x, A, y) \\
\mathsf{garb} \stackrel{\text{def}}{=} \exists S_g.\ (\mathtt{GN} = S_g) * (\circledast_{x \in S_g}.\mathsf{node}(x)) \\
\\
R = G \stackrel{\text{def}}{=} (\mathit{Push} \vee \mathit{Pop} \vee \mathsf{Id}) * \mathsf{Id} \wedge (I \ltimes I) \\
\mathit{Push} \stackrel{\text{def}}{=} \exists x, y, v, A.\ ((\mathtt{Stk} = A) \wedge (\mathtt{S} = y)) \propto ((\mathtt{Stk} = v :: A) \wedge (\mathtt{S} = x) * \mathsf{node}(x, v, y)) \\
\mathit{Pop} \stackrel{\text{def}}{=} \exists x, y, v, A, S_g.\ ((\mathtt{Stk} = v :: A) \wedge (\mathtt{S} = x) * \mathsf{node}(x, v, y) * (\mathtt{GN} = S_g)) \\
\qquad\quad \propto\ ((\mathtt{Stk} = A) \wedge (\mathtt{S} = y) * \mathsf{node}(x, v, y) * (\mathtt{GN} = S_g \cup \{x\}))
\end{array}
$$

Figure 23: Precise invariant, rely and guarantee of Treiber stack.

The guarantee includes the push and the pop actions. At the concrete side, the steps at line `8` for `push` and line `12` for `pop` in Figure 22 are the linearization points, i.e., they correspond to the abstract atomic `PUSH` and `POP` operations (thus the effect bits of the actions are **true**!). Note that when popping a node, we also add the node to `GN`. The rely of a thread is the same as its guarantee.

We show the proof in Figure 24. For linearizability, we let the abstract operations be executed simultaneously with the concrete code at linearization points. Note that when popping from an empty stack, the linearization point is at line `5` (see `pop` in Figure 22), where the thread reads the stack pointer.

On lock-freedom, we know the failure of the `cas`es at line `8` for `push` and line `12` for `pop` must be caused by the successful progress of other threads. In the proof, we can increase the number of tokens when the environment updates the `S` pointer (i.e., the environment does *Push* or *Pop*), thus are allowed to do more loop iterations.

```
push(v) {
 local x, t, b;
```
$$\{I \wedge \mathsf{arem}(\mathtt{PUSH(V)}) \wedge \mathtt{v} = \mathtt{V}\}$$
```
 b := false;
 x := cons(v, null);
```
$$\left\{\begin{array}{l}(\neg\mathtt{b} \wedge I * \mathsf{node}(\mathtt{x},\mathtt{v},\_) \wedge \mathsf{arem}(\mathtt{PUSH(V)}) \wedge (\mathtt{v} = \mathtt{V}))\\ \;\vee (\mathtt{b} \wedge I \wedge \mathsf{arem}(\boldsymbol{skip}))\end{array}\right\}$$

//Applying the WHILE rule and the HIDE-W rule

```
 while (!b) {
```
$$\{\neg\mathtt{b} \wedge I * \mathsf{node}(\mathtt{x},\mathtt{v},\_) \wedge \mathsf{arem}(\mathtt{PUSH(V)}) \wedge (\mathtt{v} = \mathtt{V}) \wedge \mathsf{wf}(0)\}$$
```
   < t := S; >
   x.next := t;
```
$$\left\{\begin{array}{l}\neg\mathtt{b} \wedge I * \mathsf{node}(\mathtt{x},\mathtt{v},\mathtt{t}) \wedge \mathsf{arem}(\mathtt{PUSH(V)}) \wedge (\mathtt{v} = \mathtt{V})\\ \;\wedge \exists a.\ (\mathtt{S} = a) * \mathbf{true} \wedge (\mathtt{t} = a \wedge \mathsf{wf}(0) \vee \mathtt{t} \neq a \wedge \mathsf{wf}(1))\end{array}\right\}$$
```
   b := cas(&S, t, x);
```
$$\left\{\begin{array}{l}(\mathtt{b} \wedge I \wedge \mathsf{arem}(\boldsymbol{skip}) \wedge \mathsf{wf}(1))\\ \;\vee (\neg\mathtt{b} \wedge I * \mathsf{node}(\mathtt{x},\mathtt{v},\_) \wedge \mathsf{arem}(\mathtt{PUSH(V)}) \wedge (\mathtt{v} = \mathtt{V}) \wedge \mathsf{wf}(1))\end{array}\right\}$$
```
 }
```
$$\{I \wedge \mathsf{arem}(\boldsymbol{skip})\}$$
```
}
```

IntSet GN;
//Auxiliary global variable for verification: popped garbage nodes

```
pop() {
 local v, x, t, b;
```
$$\{I \wedge \mathsf{arem}(\mathtt{POP})\}$$
```
 b := false;
```
$$\{(\neg\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathtt{POP}))\ \vee\ (\mathtt{b} \wedge I \wedge \mathsf{arem}(\boldsymbol{skip}) \wedge (\mathtt{v} = \mathtt{V}))\}$$

//Applying the WHILE rule and the HIDE-W rule

```
 while (!b) {
```
$$\{\neg\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathtt{POP}) \wedge \mathsf{wf}(0)\}$$
```
   < t := S; >
```
$$\left\{\begin{array}{l}(\mathtt{t} = \mathtt{null} \wedge \neg\mathtt{b} \wedge I \wedge \mathsf{arem}(\boldsymbol{skip}) \wedge (\mathtt{V} = \mathtt{EMPTY}) \wedge \mathsf{wf}(1))\\ \;\vee (\neg\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathtt{POP}) \wedge \exists a.\ (\mathtt{S} = a) * \mathsf{node}(\mathtt{t}) * \mathbf{true} \wedge (\mathtt{t} = a \wedge \mathsf{wf}(0) \vee \mathtt{t} \neq a \wedge \mathsf{wf}(1)))\end{array}\right\}$$
```
   if (t = null) {
```
$$\{\mathtt{t} = \mathtt{null} \wedge \neg\mathtt{b} \wedge I \wedge \mathsf{arem}(\boldsymbol{skip}) \wedge (\mathtt{V} = \mathtt{EMPTY})\}$$
```
     v := EMPTY;
     b := true;
```
$$\{\mathtt{b} \wedge I \wedge \mathsf{arem}(\boldsymbol{skip}) \wedge (\mathtt{v} = \mathtt{V} = \mathtt{EMPTY})\}$$
```
   } else {
```
$$\{\neg\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathtt{POP}) \wedge \exists a.\ (\mathtt{S} = a) * \mathsf{node}(\mathtt{t}) * \mathbf{true} \wedge (\mathtt{t} = a \wedge \mathsf{wf}(0) \vee \mathtt{t} \neq a \wedge \mathsf{wf}(1))\}$$
```
     v := t.data;
     x := t.next;
```
$$\{\neg\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathtt{POP}) \wedge \exists a.\ (\mathtt{S} = a) * \mathsf{node}(\mathtt{t},\mathtt{v},\mathtt{x}) * \mathbf{true} \wedge (\mathtt{t} = a \wedge \mathsf{wf}(0) \vee \mathtt{t} \neq a \wedge \mathsf{wf}(1))\}$$
```
     < b := cas(&S, t, x); GN := GN ∪ {t}; >
```
$$\{(\mathtt{b} \wedge I \wedge \mathsf{arem}(\boldsymbol{skip}) \wedge (\mathtt{v} = \mathtt{V}) \wedge \mathsf{wf}(1))\ \vee\ (\neg\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathtt{POP}) \wedge \mathsf{wf}(1))\}$$
```
   }
 }
```
$$\{I \wedge \mathsf{arem}(\boldsymbol{skip}) \wedge (\mathtt{v} = \mathtt{V})\}$$
```
 return v;
}
```

Figure 24: Proof outline for Treiber stack.

4.4 MS Lock-Free Queue

```
enq(v) {
 local x, t, s, b;
 b := false;
 x := cons(v, null);
 while (!b) {
   < t := Tail; >
   s := t.next;
   if (t = Tail) {
     if (s = null) {
       b := cas(&(t.next), s, x);
       if (b) {
         cas(&Tail, t, x);
       }
     } else {
       cas(&Tail, t, s);
     }
   }
 }
}
```

```
deq() {
 local v, s, h, t, b;
 b := false;
 while (!b) {
   < h := Head; >
   < t := Tail; >
   s := h.next;
   if (h = t) {
     if (s = null) {
       v := EMPTY;
       b := true;
     } else {
       cas(&Tail, t, s);
     }
   } else {
     v := s.val;
     b := cas(&Head, h, s);
   }
 }
 return v;
}
```

```
ENQ(V) {
 < Q := Q :: V; >
}

DEQ() {
 local V;
 < if (Q = ϵ) {
     V := EMPTY;
   } else {
     V := head(Q);
     Q := tail(Q);
   }
 >
 return V;
}
```

Figure 25: Variant of MS lock-free queue.

In Figure 25, we show a variant[1] of Michael-Scott lock-free queue [8] (at the left of the figure) and the abstract atomic operations (at the right). We use our logic to prove the linearizability and lock-freedom together of the MS queue. By similar arguments as for Treiber stack in Section 4.3, here we only need to prove the following:

$$R, G, I \vdash \{I \wedge \mathsf{arem}(\mathtt{ENQ(V)}) \wedge (\mathtt{v} = \mathtt{V})\}\ \mathtt{enq(v)}\ \{I \wedge \mathsf{arem}(\boldsymbol{skip})\}$$
$$\text{and} \qquad R, G, I \vdash \{I \wedge \mathsf{arem}(\mathtt{DEQ})\}\ \mathtt{deq}\ \{I \wedge \mathsf{arem}(\boldsymbol{skip}) \wedge (\mathtt{v} = \mathtt{V})\}$$

We define the precise invariant I, the rely R and the guarantee G in Figure 26, and show the proof in Figures 27 and 28, The invariant I for the well-formed shared data structure is defined in the same way as in linearizability proofs (e.g., [6]). Here we introduce an auxiliary variable `GH` to collect those nodes which were dequeued from the list. Initially it is set to `Head`, and would not change any more. Then the list segment from `GH` to `Head` includes all the dequeued nodes.

The rely R and the guarantee G contain three actions in addition to identity transitions: *Enq*, *Deq* and *Swing*. The actions *Enq* and *Deq* insert and remove a node from the queue, and correspond to abstract steps (the effect bits are **true**). The action *Swing* moves the `Tail` pointer, which does not correspond to any abstract steps.

The proofs in Figures 27 and 28 are based on the linearizability proofs (e.g., [6]) but also take into account the lock-freedom property.[2] We need to specify in the loop invariants (in both Figures 27 and 28)

[1] We removed in `deq` the double check on the read of the `Head` pointer. As explained in our previous work [6], this double check introduces a non-fixed linearization point in this queue algorithm, but removing it would not affect the correctness of the algorithm. Currently we use a simplified setting and do not support non-fixed linearization points (since they are orthogonal to our main focus in this paper on *termination preservation*). We can further extend the logic in this paper with the techniques for verifying linearizability with non-fixed linearization points [6], then we would be able to verify the original MS queue implementation. Due to the same reason, we remove the double check in DGLM queue implementation as well.

[2] We actually found that the lock-freedom proofs in Hoffmann et al's work [5] has bugs on computing the number of tokens. The authors confirmed our finding in our private communications.

$$I \stackrel{\text{def}}{=} \exists h, t, A.\ (\mathtt{Q} = A) \wedge (\mathtt{Head} = h) * (\mathtt{Tail} = t) * \mathsf{lsq}(h, t, A) * \mathsf{garb}(h)$$

$$\mathsf{node}(x, v, y) \stackrel{\text{def}}{=} x \mapsto (v, y) \qquad \mathsf{node}(x, y) \stackrel{\text{def}}{=} \mathsf{node}(x, \_, y) \qquad \mathsf{garb}(h) \stackrel{\text{def}}{=} \exists g.\ (\mathtt{GH} = g) * \mathsf{ls}(g, h)$$

$$\mathsf{lsq}(h, t, A) \stackrel{\text{def}}{=} \exists v, A', A''.\ (v :: A = A' :: A'') \wedge \mathsf{ls}(h, A', t) * \mathsf{tls}(t, \_, A'')$$

$$\mathsf{ls}(x, A, y) \stackrel{\text{def}}{=} (x = y \wedge A = \epsilon \wedge emp) \vee (x \neq y \wedge \exists z, v, A'.\ A = v :: A' \wedge \mathsf{node}(x, v, z) * \mathsf{ls}(z, A', y))$$

$$\mathsf{ls}(x, y) \stackrel{\text{def}}{=} \exists A.\ \mathsf{ls}(x, A, y)$$

$$\mathsf{last2}(t, v, x, v') \stackrel{\text{def}}{=} \mathsf{node}(t, v, x) * \mathsf{node}(x, v', \mathtt{null}) \qquad \mathsf{last2}(t, x) \stackrel{\text{def}}{=} \mathsf{last2}(t, \_, x, \_) \qquad \mathsf{last2}(t) \stackrel{\text{def}}{=} \mathsf{last2}(t, \_)$$

$$\mathsf{tls}(t, x, A) \stackrel{\text{def}}{=} \exists v, v'.\ (A = v \wedge \mathsf{node}(t, v, x) \wedge x = \mathtt{null}) \vee (A = v :: v' \wedge \mathsf{last2}(t, v, x, v')) \qquad \mathsf{tls}(t, x) \stackrel{\text{def}}{=} \exists A.\ \mathsf{tls}(t, x, A)$$

$$R = G \stackrel{\text{def}}{=} (\mathit{Enq} \vee \mathit{Deq} \vee \mathit{Swing} \vee \mathsf{Id}) * \mathsf{Id} \wedge (I \ltimes I)$$

$$\mathit{Enq} \stackrel{\text{def}}{=} \exists v, v', A, t, x.\ ((\mathtt{Q} = A) \wedge (\mathtt{Tail} = t) * \mathsf{node}(t, v, \mathtt{null})) \propto ((\mathtt{Q} = A :: v') \wedge (\mathtt{Tail} = t) * \mathsf{last2}(t, v, x, v'))$$

$$\begin{aligned}\mathit{Deq} \stackrel{\text{def}}{=} \exists v, A, h, t, x, y.\ & ((\mathtt{Q} = v :: A) \wedge (\mathtt{Head} = h) * \mathsf{node}(h, x) * \mathsf{node}(x, v, y) * (\mathtt{Tail} = t) \wedge h \neq t) \\ & \propto ((\mathtt{Q} = A) \wedge (\mathtt{Head} = x) * \mathsf{node}(h, x) * \mathsf{node}(x, v, y) * (\mathtt{Tail} = t))\end{aligned}$$

$$\mathit{Swing} \stackrel{\text{def}}{=} \exists v, v', t, x.\ (\boldsymbol{emp} \wedge (\mathtt{Tail} = t) * \mathsf{last2}(t, v, x, v')) \ltimes (\boldsymbol{emp} \wedge (\mathtt{Tail} = x) * \mathsf{last2}(t, v, x, v'))$$

Figure 26: Precise invariant, rely and guarantee of MS lock-free queue. The auxiliary global variable `GH` is set to `Head` in the initialization method.

the least number n of tokens to execute the loops (i.e., the thread can only run the loop for no more than n rounds before it or its environment fulfills some source steps). For instance, in the proof for `enq` (Figure 27), when the `Tail` pointer lags behind the last node, we need to have at least two tokens to first advance the `Tail` pointer in one iteration and then enqueue a node in another iteration. Thus we define `tw` (in Figure 27) saying that we have at least two tokens if `Tail` lags behind and one token otherwise. It is part of our loop invariants in both the proofs for `enq` and `deq`. Moreover, to maintain this loop invariant, we should get *two* more tokens whenever the environment enqueues a node (such that the `Tail` pointer lags behind the last node) and makes the `cas` of the current thread fail.

$$\mathsf{tw}(t) \stackrel{\text{def}}{=} (\mathtt{Tail} = t) * ((\mathsf{last2}(t) \wedge \mathsf{wf}(2)) \vee (\mathsf{node}(t, \mathtt{null}) \wedge \mathsf{wf}(1))) \qquad \mathsf{tw} \stackrel{\text{def}}{=} \exists t.\ \mathsf{tw}(t)$$

$$\mathsf{tw'}(t, n) \stackrel{\text{def}}{=} (\mathtt{Tail} = t) * ((\mathsf{last2}(t, n) \wedge \mathsf{wf}(1)) \vee (\mathsf{node}(t, n) \wedge n = \mathtt{null} \wedge \mathsf{wf}(0)))$$

$$\mathsf{tw'}(t) \stackrel{\text{def}}{=} \mathsf{tw'}(t, \_) \qquad \mathsf{tw'} \stackrel{\text{def}}{=} \exists t.\ \mathsf{tw'}(t)$$

$$\begin{aligned}\mathsf{newTail}(n) \stackrel{\text{def}}{=}\ & (\mathsf{node}(n, \mathtt{null}) * (\mathtt{Tail} = n) \wedge \mathsf{wf}(1)) \vee (\mathsf{last2}(n) * (\mathtt{Tail} = n) \wedge \mathsf{wf}(2)) \\ & \vee (\exists x, y.\ \mathsf{node}(n, x) * \mathsf{ls}(x, y) * \mathsf{tw}(y) \wedge \mathsf{wf}(2))\end{aligned}$$

$$\mathsf{readTailEnvAdv}(t, n) \stackrel{\text{def}}{=} \mathsf{node}(t, n) * \mathsf{newTail}(n) \qquad \mathsf{readTailEnvAdv}(t) \stackrel{\text{def}}{=} \mathsf{readTailEnvAdv}(t, \_)$$

$$\mathsf{readTail}(t) \stackrel{\text{def}}{=} \mathsf{tw'}(t) \vee \mathsf{readTailEnvAdv}(t)$$

$$\mathsf{readTailNextNullEnv}(t, n) \stackrel{\text{def}}{=} (n = \mathtt{null}) \wedge ((\mathtt{Tail} = t) * \mathsf{last2}(t) \wedge \mathsf{wf}(2)) \vee \mathsf{readTailEnvAdv}(t))$$

$$\mathsf{readTailNext}(t, n) \stackrel{\text{def}}{=} \mathsf{tw'}(t, n) \vee \mathsf{readTailEnvAdv}(t, n) \vee \mathsf{readTailNextNullEnv}(t, n)$$

$$\mathsf{readTailNextNull}(t, n) \stackrel{\text{def}}{=} ((\mathtt{Tail} = t) * \mathsf{node}(t, n) \wedge n = \mathtt{null} \wedge \mathsf{wf}(0)) \vee \mathsf{readTailNextNullEnv}(t, n)$$

$$\mathsf{readTailNextNonnull}(t, n) \stackrel{\text{def}}{=} ((\mathtt{Tail} = t) * \mathsf{last2}(t, n) \wedge \mathsf{wf}(1)) \vee \mathsf{readTailEnvAdv}(t, n)$$

`enq(v) {`

`local x, t, s, b;`

$\{I \wedge \mathsf{arem}(\mathtt{ENQ(V)}) \wedge \mathtt{v} = \mathtt{V}\}$

`b := false;`

`x := cons(v, null);`

$\left\{\begin{array}{l}(\neg\mathsf{b} \wedge I * \mathsf{node}(\mathtt{x}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\mathtt{ENQ(V)}) \wedge (\mathtt{v} = \mathtt{V})) \\ \ \vee (\mathsf{b} \wedge I \wedge \mathsf{arem}(\textbf{\textit{skip}}))\end{array}\right\}$ //Applying the WHILE rule and the HIDE-W rule

`while (!b) {`

$\{\neg\mathsf{b} \wedge (I \wedge \mathsf{tw'} * \mathbf{true}) * \mathsf{node}(\mathtt{x}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\mathtt{ENQ(V)}) \wedge (\mathtt{v} = \mathtt{V})\}$

`< t := Tail; >`

$\{\neg\mathsf{b} \wedge (I \wedge \mathsf{readTail}(\mathtt{t}) * \mathbf{true}) * \mathsf{node}(\mathtt{x}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\mathtt{ENQ(V)}) \wedge (\mathtt{v} = \mathtt{V})\}$

`s := t.next;`

$\{\neg\mathsf{b} \wedge (I \wedge \mathsf{readTailNext}(\mathtt{t}, \mathtt{s}) * \mathbf{true}) * \mathsf{node}(\mathtt{x}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\mathtt{ENQ(V)}) \wedge (\mathtt{v} = \mathtt{V})\}$

`if (t = Tail) {`

$\{\neg\mathsf{b} \wedge (I \wedge \mathsf{readTailNext}(\mathtt{t}, \mathtt{s}) * \mathbf{true}) * \mathsf{node}(\mathtt{x}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\mathtt{ENQ(V)}) \wedge (\mathtt{v} = \mathtt{V})\}$

`if (s = null) {`

$\{\neg\mathsf{b} \wedge (I \wedge \mathsf{readTailNextNull}(\mathtt{t}, \mathtt{s}) * \mathbf{true}) * \mathsf{node}(\mathtt{x}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\mathtt{ENQ(V)}) \wedge (\mathtt{v} = \mathtt{V})\}$

`b := cas(&(t.next), s, x);`

$\left\{\begin{array}{l}(\mathsf{b} \wedge I \wedge \mathsf{readTailNextNonnull}(\mathtt{t}, \mathtt{x}) * \mathbf{true} \wedge \mathsf{arem}(\textbf{\textit{skip}})) \\ \ \vee (\neg\mathsf{b} \wedge (I \wedge \mathsf{readTailNextNullEnv}(\mathtt{t}, \mathtt{s}) * \mathbf{true}) * \mathsf{node}(\mathtt{x}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\mathtt{ENQ(V)}) \wedge (\mathtt{v} = \mathtt{V}))\end{array}\right\}$

`if (b) {`

$\{\mathsf{b} \wedge I \wedge \mathsf{readTailNextNonnull}(\mathtt{t}, \mathtt{x}) * \mathbf{true} \wedge \mathsf{arem}(\textbf{\textit{skip}})\}$

`cas(&Tail, t, x);`

$\{\mathsf{b} \wedge I \wedge \mathsf{arem}(\textbf{\textit{skip}})\}$

`}`

$\left\{\begin{array}{l}(\mathsf{b} \wedge I \wedge \mathsf{arem}(\textbf{\textit{skip}})) \\ \ \vee (\neg\mathsf{b} \wedge (I \wedge \mathsf{tw} * \mathbf{true}) * \mathsf{node}(\mathtt{x}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\mathtt{ENQ(V)}) \wedge (\mathtt{v} = \mathtt{V}))\end{array}\right\}$

`} else {`

$\{\neg\mathsf{b} \wedge (I \wedge \mathsf{readTailNextNonnull}(\mathtt{t}, \mathtt{s}) * \mathbf{true}) * \mathsf{node}(\mathtt{x}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\mathtt{ENQ(V)}) \wedge (\mathtt{v} = \mathtt{V})\}$

`cas(&Tail, t, s);`

$\{\neg\mathsf{b} \wedge (I \wedge \mathsf{tw} * \mathbf{true}) * \mathsf{node}(\mathtt{x}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\mathtt{ENQ(V)}) \wedge (\mathtt{v} = \mathtt{V})\}$

`}`

`}`

$\left\{\begin{array}{l}(\neg\mathsf{b} \wedge (I \wedge \mathsf{tw} * \mathbf{true}) * \mathsf{node}(\mathtt{x}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\mathtt{ENQ(V)}) \wedge (\mathtt{v} = \mathtt{V})) \\ \ \vee (\mathsf{b} \wedge I \wedge \mathsf{arem}(\textbf{\textit{skip}}))\end{array}\right\}$

`}`

$\{I \wedge \mathsf{arem}(\textbf{\textit{skip}})\}$

`}`

Figure 27: Proof outline for enq of MS lock-free queue.

$$\mathsf{readHeadEnv}(h,n,x) \stackrel{\text{def}}{=} (h \neq x) \wedge \mathsf{node}(h,n) * \mathsf{ls}(n,x) * (\mathtt{Head} = x)$$

$$\mathsf{readHead}(h,x) \stackrel{\text{def}}{=} ((h = x) \wedge (\mathtt{Head} = x)) \vee (\mathsf{readHeadEnv}(h,\_,x) * \mathsf{wf}(1)) \qquad \mathsf{readHead}(h) \stackrel{\text{def}}{=} \mathsf{readHead}(h,\_)$$

$$\mathsf{readTailAfterHead}(h,t) \stackrel{\text{def}}{=} \exists x.\ \mathsf{readHead}(h,x) * \mathsf{ls}(x,t) * \mathsf{readTail}(t)$$

$$\begin{aligned}\mathsf{readHeadNextAfterTail}(h,n,t) \stackrel{\text{def}}{=}\ & (((\mathtt{Head} = h) \wedge (h = t)) * \mathsf{readTailNext}(t,n)) \\ & \vee ((\mathtt{Head} = h) * \mathsf{node}(h,n) * \mathsf{ls}(n,t) * \mathsf{readTail}(t)) \\ & \vee (\exists x.\ \mathsf{readHeadEnv}(h,n,x) * \mathsf{wf}(1) * \mathsf{ls}(x,t) * \mathsf{readTail}(t))\end{aligned}$$

$$\begin{aligned}\mathsf{readHeadNextVal}(h,n,v) \stackrel{\text{def}}{=}\ & ((\mathtt{Head} = h) * \mathsf{node}(h,n) * \mathsf{node}(n,v,\_) * (\mathtt{Tail} = n)) \\ & \vee (\exists x, t.\ (\mathtt{Head} = h) * \mathsf{node}(h,n) * \mathsf{node}(n,v,x) * \mathsf{ls}(x,t) * (\mathtt{Tail} = t)) \\ & \vee (\mathsf{readHeadEnv}(h,n,\_) * \mathsf{tw})\end{aligned}$$

```
deq() {
 local v, s, h, t, b;
 {I ∧ arem(DEQ)}
 b := false;
 {(¬b ∧ I ∧ arem(DEQ)) ∨ (b ∧ I ∧ arem(skip) ∧ (v = V))}
          //Applying the WHILE rule and the HIDE-W rule
 while (!b) {
   {¬b ∧ I ∧ tw' ∗ true ∧ arem(DEQ)}
   < h := Head; >
   {¬b ∧ I ∧ tw' ∗ readHead(h) ∗ true ∧ arem(DEQ)}
   < t := Tail; >
   {¬b ∧ I ∧ readTailAfterHead(h, t) ∗ true ∧ arem(DEQ)}
   s := h.next;
   {¬b ∧ I ∧ readHeadNextAfterTail(h, s, t) ∗ true
     ∧ ((h = t ∧ s = null ∧ arem(skip) ∧ V = EMPTY) ∨ ((h ≠ t ∨ s ≠ null) ∧ arem(DEQ)))}
   if (h = t) {
     if (s = null) {
       {¬b ∧ I ∧ h = t ∧ s = null ∧ arem(skip) ∧ V = EMPTY}
       v := EMPTY;
       b := true;
       {b ∧ I ∧ arem(skip) ∧ (v = V = EMPTY)}
     } else {
       {¬b ∧ I ∧ readHeadNextAfterTail(h, s, t) ∗ true ∧ h = t ∧ s ≠ null ∧ arem(DEQ)}
       {¬b ∧ I ∧ readTailNextNonnull(t, s) ∗ true ∧ arem(DEQ)}
       cas(&Tail, t, s);
       {¬b ∧ I ∧ tw ∗ true ∧ arem(DEQ)}
     }
   } else {
     {¬b ∧ I ∧ readHeadNextAfterTail(h, s, t) ∗ true ∧ h ≠ t ∧ arem(DEQ)}
     v := s.val;
     {¬b ∧ I ∧ readHeadNextAfterTail(h, s, t) ∗ true ∧ node(s, v, _) ∗ true ∧ h ≠ t ∧ arem(DEQ)}
     {¬b ∧ I ∧ readHeadNextVal(h, s, v) ∗ true ∧ arem(DEQ)}
     < b := cas(&Head, h, s); >
     {(¬b ∧ I ∧ tw ∗ true ∧ arem(DEQ)) ∨ (b ∧ I ∧ arem(skip) ∧ (v = V))}
   }
   {(¬b ∧ I ∧ tw ∗ true ∧ arem(DEQ)) ∨ (b ∧ I ∧ arem(skip) ∧ (v = V))}
 }
 {I ∧ arem(skip) ∧ (v = V)}
 return v;
}
```

Figure 28: Proof outline for a variant of deq in MS lock-free queue.

4.5 DGLM Lock-Free Queue

```
1 enq(v) {
2  local x, t, s, b;
3  b := false;
4  x := cons(v, null);
5  while (!b) {
6    < t := Tail; >
7    s := t.next;
8    if (t = Tail) {
9      if (s = null) {
10       b := cas(&(t.next), s, x);
11       if (b) {
12         cas(&Tail, t, x);
13       }
14     } else {
15       cas(&Tail, t, s);
16     }
17   }
18 }
19 }
```

```
1 deq() {
2  local v, s, h, t, b;
3  b := false;
4  while (!b) {
5    < h := Head; >
6    s := h.next;
7    if (s = null) {
8      v := EMPTY;
9      b := true;
10   } else {
11     v := s.val;
12     b := cas(&Head, h, s);
13     if (b) {
14       < t := Tail; >
15       if (h = t) {
16         cas(&Tail, t, s);
17       }
18     }
19   }
20 }
21 return v;
22 }
```

Figure 29: Variant of DGLM lock-free queue.

Doherty et al. [1] present an optimized version of the `deq` method in MS lock-free queue, and verify linearizability of the algorithm by constructing a forward and a backward simulations. Here we prove its linearizability and lock-freedom together. We show a variant[3] of the code in Figure 29. Its `enq` method is the same as the MS lock-free queue. For `deq`, it tests whether `Tail` points to the sentinel node (line `15` in Figure 29) only after `Head` has been updated (line `12`), while in Michael and Scott's version, the test (line `8` in the `deq` of Figure 25) is performed before knowing the queue is not empty.

The precise invariant I and the rely/guarantee conditions R and G are almost the same as MS lock-free queue, as shown in Figure 30. The proof for `enq` is the same as that of MS lock-free queue. In Figure 31, we show the proof of the `deq` method for the DGLM queue using our logic. Different from the `deq` method of MS queue, here we would not first use one iteration to advance the `Tail` pointer before dequeuing nodes (instead, only after we have dequeued nodes, we may advance the `Tail` pointer, as shown at line `16` of the `deq` method in Figure 29). Thus in the loop invariant, we no longer need to have at least two tokens when `Tail` lags behind the last node. We can just use $\mathsf{wf}(1)$ as the loop invariant on the number of tokens, for all cases.

$$
\begin{array}{l}
I \stackrel{\text{def}}{=} \exists h,t,A.\ (\&\mathtt{Q} \Rrightarrow A) \wedge (\&\mathtt{Head} \mapsto h) * (\&\mathtt{Tail} \mapsto t) * (\mathsf{lsq}(h,t,A) \vee \mathsf{cross}(h,t,A)) * \mathsf{garb}(h) \\
\mathsf{cross}(h,t,A) \stackrel{\text{def}}{=} (A = \epsilon) \wedge \mathsf{node}(t,h) * \mathsf{node}(h,\mathtt{null})
\end{array}
$$

$$
\begin{array}{l}
R = G \stackrel{\text{def}}{=} (\mathit{Enq} \vee \mathit{Deq} \vee \mathit{Swing} \vee \mathsf{Id}) * \mathsf{Id} \wedge (I \ltimes I) \\
\mathit{Deq} \stackrel{\text{def}}{=} \exists v,A,h,x,y.\ ((\&\mathtt{Q} \Rrightarrow v :: A) \wedge (\&\mathtt{Head} \mapsto h) * \mathsf{node}(h,x) * \mathsf{node}(x,v,y)) \\
\qquad\quad \propto\ ((\&\mathtt{Q} \Rrightarrow A) \wedge (\&\mathtt{Head} \mapsto x) * \mathsf{node}(h,x) * \mathsf{node}(x,v,y))
\end{array}
$$

Figure 30: Precise invariant, rely and guarantee of DGLM lock-free queue. Here lsq, garb, Enq and Swing are the same as those for MS queue.

[3] As for MS lock-free queue, we also remove the double check on the read of `Head` in the `deq` method of DGLM queue.

$$\mathsf{readHeadNextNullEnv}(h,n) \stackrel{\text{def}}{=} (n = \mathtt{null}) \wedge \exists x, y.\ \mathsf{node}(h,x) * ((\mathsf{node}(x,y) * (\&\mathtt{Head} \mapsto h)) \vee (\mathsf{ls}(x,y) * (\&\mathtt{Head} \mapsto y)))$$

$$\mathsf{readHeadNext}(h,n) \stackrel{\text{def}}{=} (\mathsf{node}(h,n) * (\&\mathtt{Head} \mapsto h)) \vee (\mathsf{readHeadEnv}(h,n,x) * \mathsf{wf}(1)) \vee \mathsf{readHeadNextNullEnv}(h,n)$$

$$\mathsf{readHeadNextVal}(h,n,v) \stackrel{\text{def}}{=} ((\&\mathtt{Head} \mapsto h) * \mathsf{node}(h,n) * \mathsf{node}(n,v,\_)) \vee (\mathsf{readHeadEnv}(h,n,x) * \mathsf{wf}(1))$$

$$\mathsf{readTailEnvAdv}(t,n) \stackrel{\text{def}}{=} \exists x.\ (x \neq t) \wedge \mathsf{node}(t,n) * \mathsf{ls}(n,x) * (\&\mathtt{Tail} \mapsto x)$$

$$\mathsf{readTail}(t) \stackrel{\text{def}}{=} ((\&\mathtt{Tail} \mapsto t) * \mathsf{tls}(t,\_)) \vee \mathsf{readTailEnvAdv}(t,\_)$$

$$\mathsf{readLagTail}(t,n) \stackrel{\text{def}}{=} ((\&\mathtt{Tail} \mapsto t) * \mathsf{last2}(t,n)) \vee \mathsf{readTailEnvAdv}(t,n)$$

```
deq() {
 local v, s, h, t, b;
```

$\{I \wedge \mathsf{arem}(\mathtt{DEQ})\}$

```
 b := false;
```

$\{(\neg\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathtt{DEQ})) \vee (\mathtt{b} \wedge I \wedge \mathsf{arem}(\textbf{\textit{skip}}) \wedge (\mathtt{v} = \mathtt{V}))\}$

//Applying the WHILE rule and the HIDE-W rule

```
 while (!b) {
```

$\{\neg\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathtt{DEQ}) \wedge \mathsf{wf}(0)\}$

```
  < h := Head; >
```

$\{\neg\mathtt{b} \wedge I \wedge \mathsf{readHead}(\mathtt{h}) * \mathbf{true} \wedge \mathsf{arem}(\mathtt{DEQ})\}$

```
  s := h.next;
```

$$\left\{\begin{array}{l}\neg\mathtt{b} \wedge I \wedge \mathsf{readHeadNext}(\mathtt{h},\mathtt{s}) * \mathbf{true} \\ \quad \wedge ((\mathtt{s} = \mathtt{null} \wedge \mathsf{arem}(\textbf{\textit{skip}}) \wedge \mathtt{V} = \mathtt{EMPTY}) \vee (\mathtt{s} \neq \mathtt{null} \wedge \mathsf{arem}(\mathtt{DEQ})))\end{array}\right\}$$

```
  if (s = null) {
```

$\{\neg\mathtt{b} \wedge I \wedge \mathtt{s} = \mathtt{null} \wedge \mathsf{arem}(\textbf{\textit{skip}}) \wedge \mathtt{V} = \mathtt{EMPTY}\}$

```
   v := EMPTY;
   b := true;
```

$\{\mathtt{b} \wedge I \wedge \mathsf{arem}(\textbf{\textit{skip}}) \wedge (\mathtt{v} = \mathtt{V} = \mathtt{EMPTY})\}$

```
  } else {
```

$\{\neg\mathtt{b} \wedge I \wedge \mathsf{readHeadNext}(\mathtt{h},\mathtt{s}) * \mathbf{true} \wedge (\mathtt{s} \neq \mathtt{null}) \wedge \mathsf{arem}(\mathtt{DEQ}))\}$

```
   v := s.val;
```

$\{\neg\mathtt{b} \wedge I \wedge \mathsf{readHeadNextVal}(\mathtt{h},\mathtt{s},\mathtt{v}) * \mathbf{true} \wedge \mathsf{arem}(\mathtt{DEQ}))\}$

```
   b := cas(&Head, h, s);
```

$\{(\mathtt{b} \wedge I \wedge \mathsf{node}(\mathtt{h},\mathtt{s}) * \mathsf{node}(\mathtt{s},\_) * \mathbf{true} \wedge \mathsf{arem}(\textbf{\textit{skip}}) \wedge (\mathtt{v} = \mathtt{V})) \vee (\neg\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathtt{DEQ}) \wedge \mathsf{wf}(1))\}$

```
   if (b) {
```

$\{\mathtt{b} \wedge I \wedge \mathsf{node}(\mathtt{h},\mathtt{s}) * \mathsf{node}(\mathtt{s},\_) * \mathbf{true} \wedge \mathsf{arem}(\textbf{\textit{skip}}) \wedge (\mathtt{v} = \mathtt{V})\}$

```
    < t := Tail; >
```

$\{\mathtt{b} \wedge I \wedge \mathsf{node}(\mathtt{h},\mathtt{s}) * \mathsf{node}(\mathtt{s},\_) * \mathbf{true} \wedge \mathsf{readTail}(\mathtt{t}) * \mathbf{true} \wedge \mathsf{arem}(\textbf{\textit{skip}}) \wedge (\mathtt{v} = \mathtt{V})\}$

```
    if (h = t) {
```

$\{\mathtt{b} \wedge I \wedge \mathsf{readLagTail}(\mathtt{t},\mathtt{s}) * \mathbf{true} \wedge \mathsf{arem}(\textbf{\textit{skip}}) \wedge (\mathtt{v} = \mathtt{V})\}$

```
     cas(&Tail, t, s);
    }
```

$\{\mathtt{b} \wedge I \wedge \mathsf{arem}(\textbf{\textit{skip}}) \wedge (\mathtt{v} = \mathtt{V})\}$

```
   }
  }
```

$\{(\neg\mathtt{b} \wedge I \wedge \mathsf{arem}(\mathtt{DEQ}) \wedge \mathsf{wf}(1)) \vee (\mathtt{b} \wedge I \wedge \mathsf{arem}(\textbf{\textit{skip}}) \wedge (\mathtt{v} = \mathtt{V}))\}$

```
 }
```

$\{I \wedge \mathsf{arem}(\textbf{\textit{skip}}) \wedge (\mathtt{v} = \mathtt{V})\}$

```
 return v;
}
```

Figure 31: Proof outline for a variant of deq in DGLM lock-free queue. Here readHead and readHeadEnv are the same as those for MS queue.

4.6 Synchronous Queue

```
1 initialize() {
2   local sentinel;
3   sentinel := new Node(null, DATA, null);
4   GH := Head := Tail := sentinel;
5 }
```

```
1 enq(v) {
2  local t, h, n, offer, b, v';
3  b := false;
4  offer := new Node(v, DATA, null);
5  while (!b) {
6    t := Tail;
7    h := Head;
8    if (h = t || t.type = DATA) {
9      n := t.next;
10     if (t = Tail) {
11       if (n != null) {
12         cas(&Tail, t, n);
13       } else if (cas(&(t.next), n, offer)){
14         cas(&Tail, t, offer);
15         v' := offer.data;
16         while (v' = v) { v' := offer.data; }
17         h := Head;
18         if (offer = h.next)
19           cas(&Head, h, offer);
20         b := true;
21       }
22     }
23   } else {
24     n := h.next;
25     if (t = Tail && h = Head && n != null) {
26       b := cas(&(n.data), null, v);
27       cas(Head, h, n);
28       if (b) free(offer);
29     }
30   }
31  }
32 }
```

```
1 deq() {
2  local t, h, n, req, b, v;
3  b := false;
4  req := new Node(null, REQ, null);
5  while (!b) {
6    t := Tail;
7    h := Head;
8    if (h = t || t.type = REQ) {
9      n := t.next;
10     if (t = Tail) {
11       if (n != null) {
12         cas(&Tail, t, n);
13       } else if (cas(&(t.next), n, req)){
14         cas(&Tail, t, req);
15         v := req.data;
16         while (v = null) { v := req.data; }
17         h := Head;
18         if (req = h.next)
19           cas(&Head, h, req);
20         b := true;
21       }
22     }
23   } else {
24     n := h.next;
25     if (t = Tail && h = Head && n != null) {
26       v := n.data;
27       if (v != null) {
28         b := cas(&(n.data), v, null);
29       }
30       cas(Head, h, n);
31       if (b) free(offer);
32     }
33   }
34  }
35  return v;
36 }
```

Figure 32: Synchronous dual queue. Here `GH` is an auxiliary variable.

A synchronous queue is a concurrent transfer channel in which each producer presenting an item must wait for a consumer to take this item, and vice versa. We show the implementation of synchronous queue (used in Java 6 [9]) in Figure 32. It is based on the Michael-Scott queue. At any time, the queue contains either `enq` reservations (nodes whose `type` fields are `DATA`), `deq` reservations (nodes whose `type` fields are `REQ`), or it is empty. In the `enq` method (also known as `put`), a thread first checks if the queue is empty or contains `DATA`-type reservations (line `8` in `enq` in Figure 32). If so, it enqueues (puts in) its new `DATA`-type reservation (lines `13` and `14` in `enq`), and waits at the item for a `deq` thread to take it (lines `15` and `16` in `enq`). When a `deq` thread finds this reservation, it will take away the data contained in the item (line `26` in `deq`), set the `data` field to `null` (line `28` in `deq`) and remove this item (line `30` in `deq`). Also when the waiting `enq` thread finds that the item has been taken, it can try to remove the item as well (lines `18` and `19` in `enq`). Symmetrically, a `deq` thread first checks if the queue is empty or contains

REQ-type reservations (line 8 in deq), and if so, it enqueues (puts in) its new REQ-type reservation (lines 13 and 14 in deq), and waits for a enq thread to fulfill it (lines 15 and 16 in deq).

The synchronous queue does not satisfy the traditional linearizability definition [4]. But we can see that the steps for a thread to put in its reservation (which are actually like the enq method in MS queue in Figure 25) are "linearizable" and "lock-free" (in that the multiple steps can be abstracted as an atomic operation), and the steps for taking away the data or fulfilling the reservation (which are like the deq method in MS queue) are also "linearizable" and "lock-free". The waiting steps are certainly not "lock-free" which require interactions from other threads to progress. We can define non-atomic abstract code and prove that the synchronous queue implementation refines it.

```
 1 ENQ(V) {
 2  local nd, mustWait, va;
 3  < nd := dequeue(D);
 4    mustWait := (nd = null);
 5    if (mustWait) { nd := enqueue(E, V); }
 6  >
 7  if (mustWait) {
 8    va := nd.data;
 9    while(va = V) { va := nd.data; }
10  }
11  else {
12    nd.data := V;
13  }
14 }
```

```
 1 DEQ() {
 2  local nd, mustWait, V;
 3  < nd := dequeue(E);
 4    mustWait := (nd = null);
 5    if (mustWait) { nd := enqueue(D, null); }
 6  >
 7  if (mustWait) {
 8    V := nd.data;
 9    while(V = null) { V := nd.data; }
10  }
11  else {
12    V := nd.data;
13    nd.data := null;
14  }
15  return V;
16 }
```

Figure 33: Abstract synchronous queue.

As shown in Figure 33, the abstract code follows Java SE 5.0 SynchronousQueue class [9]. We maintain two abstract queues: D for waiting dequeuers and E for waiting enqueuers. Each queue is a mathematical list of node *addresses* (as an abstraction/simplification of a linked list). The command enqueue(E, v) allocates a new abstract node with data v and inserts its address at the tail of the queue E, and returns the address. The command dequeue(E) removes the first item (a node address) from the queue E and returns it if E is not empty ($\mathtt{E} \neq \epsilon$), and returns null otherwise.

In the ENQ method, a thread first checks if D is empty (line 4 of ENQ in Figure 33), and if so, it atomically puts in its reservation to E (line 5). Then it waits for a deq thread to take away the data in the reservation (lines 8 and 9). If D is not empty, then it dequeues a reservation from D and writes its enqueued value V to the data field of the reservation (line 12). The DEQ method is symmetric.

To simplify the proof, we assume the abstract state always contain a dummy node whose data is null. The node is never accessed by the code. It is used to correspond to the initial sentinel node of the concrete list.

To prove the concrete implementation in Figure 32 refines the abstract operations in Figure 33 using our logic, we first define the invariant I and the rely and guarantee conditions R and G in Figure 34.

The invariant I says, the shared memory contains the queue Q and some garbage nodes Garb which were removed from the queue by either enq or deq. As usual we introduce an auxiliary variable GH to collect those nodes which were removed from the list. Initially it is set to Head, and would not change any more. Then the list segment (Gls) from GH to Head includes all the removed nodes. Also these removed nodes must have been sentinel nodes (stnl), i.e., those DATA-type nodes whose data has been taken and those REQ-type nodes whose data has been fulfilled. The queue Q is either a DATA-type queue (and the abstract D must be empty) or a REQ-type queue (and the abstract E must be empty). And it always contains one or two sentinel nodes (the two-sentinel case occurs since the Head pointer may lag behind the new sentinel node). Also as in MS queue, the `Tail` pointer may lag behind the last node. But if `Head` lags behind the new sentinel node, `Tail` would not be equal to `Head`, as indicated by the implementation in Figure 32.

The rely and guarantee conditions contain six possible actions in addition to the identity transitions. *AdvHead* and *AdvTail* are to swing the `Head` and `Tail` pointers when they lag behind. These two actions do not correspond to any abstract step. *ResvE* and *ResvD* each inserts a new node at the tail of the queue. *Put* fulfills the `data` field of a `REQ`-type node at the head of the queue, and *Take* takes away the `data` of a `DATA`-type node. They both make a normal node into a sentinel node. The four actions *ResvE*, *ResvD*, *Put* and *Take* correspond to abstract steps and thus their effect bits must be **true**.

We show the proofs of `enq` in Figures 37 and 38, with some auxiliary predicates defined in Figures 35 and 36. Proofs for `deq` is symmetric and omitted here. Similar to the proofs for MS queue, we need to specify in the loop invariants the least number n of tokens to execute the loops (i.e., the thread can only run the loop for no more than n rounds before it or its environment fulfills some source steps). In the proof for `enq` (Figure 37), when either the `Head` or the `Tail` pointer lags behind, we need to have at least two tokens (as defined by $\mathsf{loopInv}$ in Figure 35). To maintain this loop invariant, we should get *two* more tokens whenever the environment inserts a node at the tail (such that the `Tail` pointer lags behind the last node), and whenever the environment makes a normal node becomes a sentinel node (such that the `Head` pointer lags behind the new sentinel).

$$I \stackrel{\text{def}}{=} \exists h,t.\ (\texttt{Head} \doteq h) * (\texttt{Tail} \doteq t) * \mathsf{Q}(h,t) * \mathsf{Garb}(h)$$

$$\mathsf{Q}(h,t) \stackrel{\text{def}}{=} \exists b.\ \mathsf{Q}_b(h,t)$$

$$\mathsf{Q}_b(h,t) \stackrel{\text{def}}{=} \exists L.\ \mathsf{Q}_b(h,t,L) * ((b = \texttt{DATA} \wedge (\texttt{E} \doteq L) * (\texttt{D} \doteq \epsilon)) \vee (b = \texttt{REQ} \wedge (\texttt{D} \doteq L) * (\texttt{E} \doteq \epsilon)))$$

$$\begin{aligned}\mathsf{Q}_b(h,t,L) \stackrel{\text{def}}{=}\ & \mathsf{Ss}_b(h,t,\texttt{null}) \wedge L = \epsilon \\ & \vee \exists x, X.\ \mathsf{Ss}_b(h,t,x) * \mathsf{Qn}_b(x,\texttt{null},X) \wedge L = X :: \epsilon \\ & \vee \exists x, L', L''.\ \mathsf{Ss}_b(h,\_,x) * \mathsf{Qls}_b(x,t,L') * \mathsf{Qtl}_b(t,\_,L'') \wedge L = L' :: L''\end{aligned}$$

$$\mathsf{Garb}(h) \stackrel{\text{def}}{=} \exists g.\ (\texttt{GH} \doteq g) * \mathsf{Gls}(g,h)$$

$$\mathsf{Ss}(x,y,z) \stackrel{\text{def}}{=} \exists b.\ \mathsf{Ss}_b(x,y,z) \qquad \mathsf{Ss}_b(x,y,z) \stackrel{\text{def}}{=} (\mathsf{Stnl}(x,z) \wedge (x = y)) \vee (\mathsf{Stnl}(x,y) * \mathsf{Stnl}_b(y,z))$$

$$\mathsf{Gls}(x,y) \stackrel{\text{def}}{=} (x = y) \vee (x \neq y \wedge \exists z.\ \mathsf{Stnl}(x,z) * \mathsf{Gls}(z,y))$$

$$\mathsf{Qls}_b(x,y,L) \stackrel{\text{def}}{=} (x = y \wedge L = \epsilon) \vee (x \neq y \wedge \exists z, X, L'.\ L = X :: L' \wedge \mathsf{Qn}_b(x,z,X) * \mathsf{Qls}_b(z,y,L'))$$

$$\begin{aligned}\mathsf{Qtl}_b(x,y,L) \stackrel{\text{def}}{=}\ & (\exists X.\ \mathsf{Qn}_b(x,y,X) \wedge y = \texttt{null} \wedge L = X :: \epsilon) \\ & \vee (\exists X, Y.\ \mathsf{Qn}_b(x,y,X) * \mathsf{Qn}_b(y,\texttt{null},Y) \wedge L = X :: Y :: \epsilon)\end{aligned}$$

$$\mathsf{Stnl}(x,y) \stackrel{\text{def}}{=} \exists b.\ \mathsf{Stnl}_b(x,\_,y,\_) \qquad \mathsf{Stnl}_b(x,v,y,X) \stackrel{\text{def}}{=} \mathsf{stnl}_b(x,v,y) \wedge \mathsf{NODE}(X,v)$$

$$\mathsf{Qn}_b(x,y,X) \stackrel{\text{def}}{=} \mathsf{Qn}_b(x,\_,y,X) \qquad \mathsf{Qn}_b(x,v,y,X) \stackrel{\text{def}}{=} \mathsf{qn}_b(x,v,y) \wedge \mathsf{NODE}(X,v)$$

$$\mathsf{stnl}_b(x,v,y) \stackrel{\text{def}}{=} \mathsf{node}_b(x,v,y) \wedge ((b = \texttt{DATA} \wedge v = \texttt{null}) \vee (b = \texttt{REQ} \wedge v \neq \texttt{null}))$$

$$\mathsf{qn}_b(x,v,y) \stackrel{\text{def}}{=} \mathsf{node}_b(x,v,y) \wedge ((b = \texttt{DATA} \wedge v \neq \texttt{null}) \vee (b = \texttt{REQ} \wedge v = \texttt{null}))$$

$$\mathsf{node}_b(x,v,y) \stackrel{\text{def}}{=} x \mapsto (v,b,y) \qquad \mathsf{NODE}(X,V) \stackrel{\text{def}}{=} X \Mapsto (V)$$

$$\mathsf{stnl}(x,y) \stackrel{\text{def}}{=} \exists b.\ \mathsf{stnl}_b(x,\_,y) \qquad \mathsf{qn}(x,y) \stackrel{\text{def}}{=} \exists b.\ \mathsf{qn}_b(x,\_,y) \qquad \mathsf{node}(x,y) \stackrel{\text{def}}{=} \exists b.\ \mathsf{node}_b(x,\_,y)$$

$$\mathsf{stnl}_b(x,y) \stackrel{\text{def}}{=} \mathsf{stnl}_b(x,\_,y) \qquad \mathsf{node}_b(x,y) \stackrel{\text{def}}{=} \mathsf{node}_b(x,\_,y) \qquad \mathsf{node}(x,v,y) \stackrel{\text{def}}{=} \exists b.\ \mathsf{node}_b(x,v,y)$$

$$R = G \stackrel{\text{def}}{=} (\mathit{AdvHead} \vee \mathit{AdvTail} \vee \mathit{ResvE} \vee \mathit{ResvD} \vee \mathit{Put} \vee \mathit{Take} \vee \mathsf{Id}) * \mathsf{Id} \wedge (I \ltimes I)$$

$$\mathit{AdvHead} \stackrel{\text{def}}{=} \exists x,y,z,s.\ [\mathsf{stnl}(x,y) * \mathsf{stnl}(y,z) \wedge \boldsymbol{emp}] * ((\texttt{Head} \doteq x) \ltimes (\texttt{Head} \doteq y))$$

$$\mathit{AdvTail} \stackrel{\text{def}}{=} \exists x,y.\ [\mathsf{node}(x,y) * \mathsf{node}(y,\texttt{null}) \wedge \boldsymbol{emp}] * ((\texttt{Tail} \doteq x) \ltimes (\texttt{Tail} \doteq y))$$

$$\begin{aligned}\mathit{ResvE} \stackrel{\text{def}}{=}\ & \exists v, v', b, t, x, L, X.\ ((\texttt{Tail} = t) * \mathsf{node}_b(t,v,\texttt{null}) \wedge (\texttt{E} = L) * (\texttt{D} = \epsilon)) \\ & \propto ((\texttt{Tail} = t) * \mathsf{node}_b(t,v,x) * \mathsf{qn}_{\texttt{DATA}}(x,v',\texttt{null}) \wedge (\mathsf{NODE}(X,v') * (\texttt{E} = L :: X) * (\texttt{D} = \epsilon)))\end{aligned}$$

$$\begin{aligned}\mathit{ResvD} \stackrel{\text{def}}{=}\ & \exists v, v', b, t, x, L, X.\ ((\texttt{Tail} = t) * \mathsf{node}_b(t,v,\texttt{null}) \wedge (\texttt{E} = \epsilon) * (\texttt{D} = L)) \\ & \propto ((\texttt{Tail} = t) * \mathsf{node}_b(t,v,x) * \mathsf{qn}_{\texttt{REQ}}(x,v',\texttt{null}) \wedge (\mathsf{NODE}(X,v') * (\texttt{E} = \epsilon) * (\texttt{D} = L :: X)))\end{aligned}$$

$$\begin{aligned}\mathit{Put} \stackrel{\text{def}}{=}\ & \exists h,t,x,y,X,L.\ [(\texttt{Head} \doteq h) * (\texttt{Tail} \doteq t) * \mathsf{Stnl}(h,x) * (\texttt{E} \doteq \epsilon) \wedge (h \neq t)] \\ & * ((\mathsf{Qn}_{\texttt{REQ}}(x,y,X) * (\texttt{D} \doteq X :: L)) \propto (\mathsf{Stnl}_{\texttt{REQ}}(x,y,X) * (\texttt{D} \doteq L))\end{aligned}$$

$$\begin{aligned}\mathit{Take} \stackrel{\text{def}}{=}\ & \exists h,t,x,y,X,L.\ [(\texttt{Head} \doteq h) * (\texttt{Tail} \doteq t) * \mathsf{Stnl}(h,x) * (\texttt{D} \doteq \epsilon) \wedge (h \neq t)] \\ & * ((\mathsf{Qn}_{\texttt{DATA}}(x,y,X) * (\texttt{E} \doteq X :: L)) \propto (\mathsf{Stnl}_{\texttt{DATA}}(x,y,X) * (\texttt{E} \doteq L))\end{aligned}$$

Figure 34: Precise invariant, rely and guarantee of synchronous queue. Here we use $E_1 \doteq E_2$ and $\mathbb{E}_1 \doteq \mathbb{E}_2$ short for $(E_1 = E_2) \wedge \boldsymbol{emp}$ and $(\mathbb{E}_1 = \mathbb{E}_2) \wedge emp$ respectively.

$$\mathsf{node2}_p(t,n,x) \stackrel{\text{def}}{=} \mathsf{node}_p(t,n) * \mathsf{node}(n,x) \qquad \mathsf{node2}(t,n,x) \stackrel{\text{def}}{=} \exists p.\ \mathsf{node2}_p(t,n,x)$$

$$\mathsf{stnl2}_p(h,n,v) \stackrel{\text{def}}{=} \mathsf{stnl}(h,n) * \mathsf{stnl}_p(n,v,\_) \qquad \mathsf{stnl2}_p(h) \stackrel{\text{def}}{=} \mathsf{stnl2}_p(h,\_,\_) \qquad \mathsf{stnl2}(h) \stackrel{\text{def}}{=} \exists p.\ \mathsf{stnl2}_p(h)$$

$$\mathsf{stnl1}_p(h,n,v) \stackrel{\text{def}}{=} \mathsf{stnl}(h,n) * \mathsf{qn}_p(n,v,\_) \qquad \mathsf{stnl1}_p(h) \stackrel{\text{def}}{=} \mathsf{stnl1}_p(h,\_,\_) \qquad \mathsf{stnl1}(h) \stackrel{\text{def}}{=} \exists p.\ \mathsf{stnl1}_p(h)$$

$$\mathsf{gls}(x,y) \stackrel{\text{def}}{=} (x=y) \vee (x \neq y \wedge \exists z.\ \mathsf{stnl}(x,z) * \mathsf{gls}(z,y))$$

$$\mathsf{ls}(x,y) \stackrel{\text{def}}{=} (x=y) \vee (x \neq y \wedge \exists z.\ \mathsf{node}(x,z) * \mathsf{ls}(z,y))$$

$$\mathsf{lagTail} \stackrel{\text{def}}{=} \mathsf{node2}(\texttt{Tail},\_,\texttt{null}) \qquad \mathsf{nonlagTail} \stackrel{\text{def}}{=} \mathsf{node}(\texttt{Tail},\texttt{null}) \qquad \mathsf{tail} \stackrel{\text{def}}{=} \mathsf{lagTail} \vee \mathsf{nonlagTail}$$

$$\mathsf{lagHead} \stackrel{\text{def}}{=} \mathsf{stnl2}(\texttt{Head}) \qquad \mathsf{nonlagHead} \stackrel{\text{def}}{=} \mathsf{stnl}(\texttt{Head},\texttt{null}) \vee \mathsf{stnl1}(\texttt{Head}) \qquad \mathsf{head} \stackrel{\text{def}}{=} \mathsf{lagHead} \vee \mathsf{nonlagHead}$$

$$\mathsf{loopInv} \stackrel{\text{def}}{=} ((\mathsf{lagTail} \vee \mathsf{lagHead}) \wedge \mathsf{wf}(2)) \ \vee\ (\mathsf{nonlagTail} \wedge \mathsf{nonlagHead} \wedge \mathsf{wf}(1))$$

$$\mathsf{loopBody} \stackrel{\text{def}}{=} ((\mathsf{lagTail} \vee \mathsf{lagHead}) \wedge \mathsf{wf}(1)) \ \vee\ (\mathsf{nonlagTail} \wedge \mathsf{nonlagHead} \wedge \mathsf{wf}(0))$$

$$\begin{aligned}\mathsf{newTail}_p(n,v) \stackrel{\text{def}}{=}\ & (\mathsf{node}_p(n,v,\texttt{null}) \wedge (n=\texttt{Tail}) \wedge \mathsf{wf}(1)) \\ & \vee (\exists x.\ \mathsf{node}_p(n,v,x) * \mathsf{node}(x,\texttt{null}) \wedge (n=\texttt{Tail}) \wedge \mathsf{wf}(2)) \\ & \vee (\exists x.\ \mathsf{node}_p(n,v,x) * \mathsf{ls}(x,\texttt{Tail}) * \mathsf{tail} \wedge \mathsf{wf}(2))\end{aligned}$$

$$\mathsf{newTail}(n) \stackrel{\text{def}}{=} \exists p,v.\ \mathsf{newTail}_p(n,v) \qquad \mathsf{NewTail}_p(n,v,N) \stackrel{\text{def}}{=} \mathsf{newTail}_p(n,v) * \mathsf{NODE}(N,v)$$

$$\mathsf{readTailEnvAdv}_{p,q}(t,n,v) \stackrel{\text{def}}{=} \mathsf{node}_p(t,n) * \mathsf{newTail}_q(n,v)$$

$$\mathsf{readTailEnvAdv}_p(t) \stackrel{\text{def}}{=} \exists q.\ \mathsf{readTailEnvAdv}_{p,q}(t,\_,\_) \qquad \mathsf{readTailEnvAdv}_p(t,n) \stackrel{\text{def}}{=} \exists q.\ \mathsf{readTailEnvAdv}_{p,q}(t,n,\_)$$

$$\mathsf{readTail}_p(t) \stackrel{\text{def}}{=} (t=\texttt{Tail} \wedge (\mathsf{node2}_p(t,\_,\texttt{null}) \vee \mathsf{node}_p(t,\texttt{null}))) \ \vee\ \mathsf{readTailEnvAdv}_p(t)$$

$$\mathsf{readTailNextNullEnv}_p(t,n) \stackrel{\text{def}}{=} (n=\texttt{null}) \wedge ((t=\texttt{Tail} \wedge \mathsf{node2}_p(t,\_,\texttt{null}) \wedge \mathsf{wf}(2)) \vee \mathsf{readTailEnvAdv}_p(t))$$

$$\begin{aligned}\mathsf{readTailNext}_p(t,n) \stackrel{\text{def}}{=}\ & (t=\texttt{Tail} \wedge (\mathsf{node2}_p(t,n,\texttt{null}) \vee (\mathsf{node}_p(t,n) \wedge n=\texttt{null}))) \\ & \vee \mathsf{readTailEnvAdv}_p(t,n) \ \vee\ \mathsf{readTailNextNullEnv}_p(t,n)\end{aligned}$$

$$\mathsf{readTailNextNonnull}_p(t,n) \stackrel{\text{def}}{=} (t=\texttt{Tail} \wedge \mathsf{node2}_p(t,n,\texttt{null}) \wedge \mathsf{wf}(1)) \ \vee\ \mathsf{readTailEnvAdv}_p(t,n)$$

$$\mathsf{readTailNextNull}_p(t,n) \stackrel{\text{def}}{=} (t=\texttt{Tail} \wedge \mathsf{node}_p(t,n) \wedge n=\texttt{null} \wedge \mathsf{wf}(0)) \ \vee\ \mathsf{readTailNextNullEnv}_p(t,n)$$

$$\mathsf{EnvXchg}_q(n,v,N) \stackrel{\text{def}}{=} \exists x.\ \mathsf{Stnl}_q(n,v,x,N) * \mathsf{ls}(x,\texttt{Tail}) * \mathsf{tail} \wedge (\mathsf{stnl}(\texttt{Head},n) \vee \mathsf{gls}(n,\texttt{Head}))$$

$$\mathsf{EnvXchgReadHead}_q(n,v,N,h) \stackrel{\text{def}}{=} \exists x.\ \mathsf{Stnl}_q(n,v,x,N) * \mathsf{ls}(x,\texttt{Tail}) * \mathsf{tail} \wedge (\mathsf{stnl}(h,n) \vee \mathsf{gls}(n,h)) \wedge \mathsf{gls}(h,\texttt{Head})$$

$$\mathsf{EnvXchgLagHead}_q(n,v,N,h) \stackrel{\text{def}}{=} \exists x.\ \mathsf{Stnl}_q(n,v,x,N) * \mathsf{ls}(x,\texttt{Tail}) * \mathsf{tail} \wedge \mathsf{stnl}(h,n) \wedge \mathsf{gls}(h,\texttt{Head})$$

$$\mathsf{EnvXchgNonlagHead}_q(n,v,N) \stackrel{\text{def}}{=} \exists x.\ \mathsf{Stnl}_q(n,v,x,N) * \mathsf{ls}(x,\texttt{Tail}) * \mathsf{tail} \wedge \mathsf{gls}(n,\texttt{Head})$$

$$\begin{aligned}\mathsf{Resv}_q(t,n,v,v',N) \stackrel{\text{def}}{=}\ & (t=\texttt{Tail} \wedge \mathsf{node}(t,n) * \mathsf{Qn}_q(n,v,\texttt{null},N)) \\ & \vee \mathsf{node}(t,n) * \mathsf{NewTail}_q(n,v,N) \ \vee\ \mathsf{node}(t,n) * \mathsf{EnvXchg}_q(n,v',N)\end{aligned}$$

$$\mathsf{ResvAdv}_q(n,v,v',N) \stackrel{\text{def}}{=} \mathsf{NewTail}_q(n,v,N) \ \vee\ \mathsf{EnvXchg}_q(n,v',N)$$

$$\mathsf{ResvAdvReadData}_q(n,v,v',v_r,N) \stackrel{\text{def}}{=} \mathsf{NewTail}_q(n,v,N) \wedge (v_r=v) \ \vee\ \mathsf{EnvXchg}_q(n,v',N) \wedge (v_r=v' \vee v_r=v)$$

$$\texttt{ENQWait} \stackrel{\text{def}}{=} (\texttt{va := nd.data; ENQWhile})$$

$$\texttt{ENQWhile} \stackrel{\text{def}}{=} (\texttt{while(va=V)\{ va := nd.data; \}})$$

Figure 35: Auxiliary definition - I.

$$\begin{aligned}
\mathsf{newHead}_p(n,v) \stackrel{\text{def}}{=}\ & (\mathsf{stnl}_p(n,v,\texttt{null}) \wedge (n = \texttt{Head}) \wedge \mathsf{wf}(1)) \\
& \vee (\exists x.\ \mathsf{stnl}_p(n,v,x) * \mathsf{qn}(x,_) \wedge (n = \texttt{Head}) \wedge \mathsf{wf}(1)) \\
& \vee (\exists x.\ \mathsf{stnl}_p(n,v,x) * \mathsf{stnl}(x,_) \wedge (n = \texttt{Head}) \wedge \mathsf{wf}(2)) \\
& \vee (\exists x.\ \mathsf{stnl}_p(n,v,x) * \mathsf{gls}(x,\texttt{Head}) * \mathsf{head} \wedge \mathsf{wf}(2))
\end{aligned}$$

$$\mathsf{newHead}(n) \stackrel{\text{def}}{=} \exists p, v.\ \mathsf{newHead}_p(n,v)$$

$$\mathsf{readHeadEnvAdv}_p(h,n,v) \stackrel{\text{def}}{=} \mathsf{stnl}(h,n) * \mathsf{newHead}_p(n,v)$$

$$\mathsf{readHeadEnvAdv}_p(h) \stackrel{\text{def}}{=} \mathsf{readHeadEnvAdv}_p(h,\_,\_) \qquad \mathsf{readHeadEnvAdv}_p(h,n) \stackrel{\text{def}}{=} \mathsf{readHeadEnvAdv}_p(h,n,\_)$$

$$\mathsf{readHead}_p(h) \stackrel{\text{def}}{=} (h = \texttt{Head} \wedge (\mathsf{stnl}_p(h,\texttt{null}) \vee \mathsf{stnl1}_p(h) \vee \mathsf{stnl2}_p(h))) \vee \mathsf{readHeadEnvAdv}_p(h)$$

$$\mathsf{readHeadNextNullEnv}_p(h,n) \stackrel{\text{def}}{=} (n = \texttt{null}) \wedge ((h = \texttt{Head} \wedge \mathsf{stnl}_p(h,x) * \mathsf{node}(x,\_) \wedge \mathsf{wf}(2)) \vee \mathsf{readHeadEnvAdv}_p(h))$$

$$\begin{aligned}
\mathsf{readHeadNext}_p(h,n) \stackrel{\text{def}}{=}\ & (h = \texttt{Head} \wedge ((\mathsf{stnl}_p(h,n) \wedge n = \texttt{null}) \vee \mathsf{stnl1}_p(h,n,_) \vee \mathsf{stnl2}_p(h,n,_))) \\
& \vee \mathsf{readHeadEnvAdv}_p(h,n) \vee \mathsf{readHeadNextNullEnv}_p(h,n)
\end{aligned}$$

$$\mathsf{readHeadNextNonnull}_p(h,n) \stackrel{\text{def}}{=} (h = \texttt{Head} \wedge (\mathsf{stnl1}_p(h,n,\_) \vee \mathsf{stnl2}_p(h,n,\_))) \vee \mathsf{readHeadEnvAdv}_p(h,n)$$

$$\mathsf{readHeadNextNull}_p(h,n) \stackrel{\text{def}}{=} (h = \texttt{Head} \wedge \mathsf{stnl}_p(h,n) \wedge n = \texttt{null}) \vee \mathsf{readHeadNextNullEnv}_p(h,n)$$

$$\mathsf{Xchg}_p(h,n,v) \stackrel{\text{def}}{=} (h = \texttt{Head} \wedge \mathsf{stnl2}_p(h,n,v)) \vee \mathsf{readHeadEnvAdv}_p(h,n,v)$$

$$\mathsf{Xchg}_p(h,n) \stackrel{\text{def}}{=} \mathsf{Xchg}_p(h,n,\_)$$

Figure 36: Auxiliary definition - II.

```
enq(v) {
 local t, h, n, offer, b, v';
 {I ∧ loopInv ∗ true ∧ arem(ENQ)}
 b := false;
 offer := new Node(v, DATA, null);
 {(¬b ∧ (I ∧ loopInv ∗ true) ∗ node_DATA(offer, v, null) ∧ arem(ENQ)) ∨ (b ∧ I ∧ arem(skip))}
 while (!b) {
   {(I ∧ loopBody ∗ true) ∗ node_DATA(offer, v, null) ∧ arem(ENQ) ∧ ¬b}
   t := tail;
   {∃p. (Q_p ∗ Garb ∧ loopBody ∗ true ∧ readTail_p(t) ∗ true) ∗ node_DATA(offer, v, null) ∧ arem(ENQ) ∧ ¬b}
   h := head;
   {∃p. (Q_p ∗ Garb ∧ loopBody ∗ true ∧ readTail_p(t) ∗ true ∧ readHead_p(h) ∗ true)
     ∗ node_DATA(offer, v, null) ∧ arem(ENQ) ∧ ¬b}
   if (h = t || t.type = DATA) {
     {∃p. (I ∧ loopBody ∗ true ∧ readTail_p(t) ∗ true ∧ gls(h, Head) ∗ true)
       ∗ node_DATA(offer, v, null) ∧ arem(ENQ) ∧ (h = t ∨ p = DATA) ∧ ¬b}
     n := t.next;
     {∃p. (I ∧ loopBody ∗ true ∧ readTailNext_p(t, n) ∗ true ∧ gls(h, Head) ∗ true)
       ∗ node_DATA(offer, v, null) ∧ arem(ENQ) ∧ (h = t ∨ p = DATA) ∧ ¬b}
     if (t = tail) {
       {∃p. (I ∧ loopBody ∗ true ∧ readTailNext_p(t, n) ∗ true ∧ gls(h, Head) ∗ true)
         ∗ node_DATA(offer, v, null) ∧ arem(ENQ) ∧ (h = t ∨ p = DATA) ∧ ¬b}
       if (n != null) {
         {∃p. (I ∧ loopBody ∗ true ∧ readTailNextNonnull_p(t, n) ∗ true)
           ∗ node_DATA(offer, v, null) ∧ arem(ENQ) ∧ ¬b}
         cas(&tail, t, n);
         {(I ∧ loopInv ∗ true) ∗ node_DATA(offer, v, null) ∧ arem(ENQ) ∧ ¬b}
       } else {
         {∃p. (I ∧ loopBody ∗ true ∧ readTailNextNull_p(t, n) ∗ true ∧ gls(h, Head) ∗ true)
           ∗ node_DATA(offer, v, null) ∧ arem(ENQ) ∧ (h = t ∨ p = DATA) ∧ ¬b}
         if (cas(&(t.next), n, offer)){
           {(I ∧ Resv_DATA(t, offer, v, null, nd) ∗ true) ∧ arem(ENQWait) ∧ ¬b}
           cas(&tail, t, offer);
           {(I ∧ ResvAdv_DATA(offer, v, null, nd) ∗ true) ∧ arem(ENQWait) ∧ ¬b}
           v' := offer.data;
           {(I ∧ ResvAdvReadData_DATA(offer, v, null, v', nd) ∗ true) ∧ (v' = va) ∧ arem(ENQWhile) ∧ ¬b}
           while (v' = v) { v' := offer.data; }
           {(I ∧ EnvXchg_DATA(offer, null, nd) ∗ true) ∧ (v' = va = null) ∧ arem(skip) ∧ ¬b}
           h := head;
           {(I ∧ EnvXchgReadHead_DATA(offer, null, nd, h) ∗ true) ∧ arem(skip) ∧ ¬b}
           if (offer = h.next)
             {(I ∧ EnvXchgLagHead_DATA(offer, null, nd, h) ∗ true) ∧ arem(skip) ∧ ¬b}
             cas(&head, h, offer);
           {(I ∧ EnvXchgNonlagHead_DATA(offer, null, nd) ∗ true) ∧ arem(skip) ∧ ¬b}
           b := true;
           {b ∧ I ∧ arem(skip)}
         }
       }
     }
   }
```

Figure 37: Proof outline - I.

```
    else {
```

$$\left\{\begin{array}{l}\exists p.\ (I \wedge \mathsf{loopBody} * \mathbf{true} \wedge \mathsf{readTail}_p(\mathtt{t}) * \mathbf{true} \wedge \mathsf{readHead}_p(\mathtt{h}) * \mathbf{true}) \\ \ * \mathsf{node}_{\mathtt{DATA}}(\mathtt{offer}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\mathtt{ENQ}) \wedge (\mathtt{h} \neq \mathtt{t} \wedge p = \mathtt{REQ}) \wedge \neg \mathsf{b}\end{array}\right\}$$

```
      n := h.next;
```

$$\left\{\begin{array}{l}\exists p.\ (I \wedge \mathsf{loopBody} * \mathbf{true} \wedge \mathsf{readTail}_p(\mathtt{t}) * \mathbf{true} \wedge \mathsf{readHeadNext}_p(\mathtt{h}, \mathtt{n}) * \mathbf{true}) \\ \ * \mathsf{node}_{\mathtt{DATA}}(\mathtt{offer}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\mathtt{ENQ}) \wedge (\mathtt{h} \neq \mathtt{t} \wedge p = \mathtt{REQ}) \wedge \neg \mathsf{b}\end{array}\right\}$$

```
      if (t = tail && h = head && n != null) {
```

$$\left\{\begin{array}{l}(I \wedge \mathsf{loopBody} * \mathbf{true} \wedge \mathsf{readHeadNextNonnull}_{\mathtt{REQ}}(\mathtt{h}, \mathtt{n}) * \mathbf{true}) \\ \ * \mathsf{node}_{\mathtt{DATA}}(\mathtt{offer}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\mathtt{ENQ}) \wedge \neg \mathsf{b}\end{array}\right\}$$

```
        b := cas(&(n.data), null, v);
```

$$\left\{\begin{array}{l}\mathsf{b} \wedge (I \wedge \mathsf{loopBody} * \mathbf{true} \wedge \mathsf{Xchg}_{\mathtt{REQ}}(\mathtt{h}, \mathtt{n}, \mathtt{v}) * \mathbf{true}) * \mathsf{node}_{\mathtt{DATA}}(\mathtt{offer}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\boldsymbol{skip}) \\ \ \vee \neg \mathsf{b} \wedge (I \wedge \mathsf{loopBody} * \mathbf{true} \wedge \mathsf{Xchg}_{\mathtt{REQ}}(\mathtt{h}, \mathtt{n}) * \mathbf{true}) * \mathsf{node}_{\mathtt{DATA}}(\mathtt{offer}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\mathtt{ENQ})\end{array}\right\}$$

```
        cas(head, h, n);
```

$$\left\{\begin{array}{l}(\mathsf{b} \wedge I * \mathsf{node}_{\mathtt{DATA}}(\mathtt{offer}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\boldsymbol{skip})) \\ \ \vee (\neg \mathsf{b} \wedge (I \wedge \mathsf{loopInv} * \mathbf{true}) * \mathsf{node}_{\mathtt{DATA}}(\mathtt{offer}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\mathtt{ENQ}))\end{array}\right\}$$

```
        if (b) free(offer);
```

$$\left\{(\neg \mathsf{b} \wedge (I \wedge \mathsf{loopInv} * \mathbf{true}) * \mathsf{node}_{\mathtt{DATA}}(\mathtt{offer}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\mathtt{ENQ})) \ \vee \ (\mathsf{b} \wedge I \wedge \mathsf{arem}(\boldsymbol{skip}))\right\}$$

```
      } else {
```

$$\left\{\begin{array}{l}(I \wedge \mathsf{loopBody} * \mathbf{true} \wedge (\mathsf{readTailEnvAdv}_{\mathtt{REQ}}(\mathtt{t}) \vee \mathsf{readHeadEnvAdv}_{\mathtt{REQ}}(\mathtt{h}) \vee \mathsf{readHeadNextNullEnv}_{\mathtt{REQ}}(\mathtt{h}, \mathtt{n})) * \mathbf{true}) \\ \ * \mathsf{node}_{\mathtt{DATA}}(\mathtt{offer}, \mathtt{v}, \mathtt{null}) \wedge (\mathtt{h} \neq \mathtt{t}) \wedge \mathsf{arem}(\mathtt{ENQ}) \wedge \neg \mathsf{b}\end{array}\right\}$$

$$\left\{\neg \mathsf{b} \wedge (I \wedge \mathsf{loopInv} * \mathbf{true}) * \mathsf{node}_{\mathtt{DATA}}(\mathtt{offer}, \mathtt{v}, \mathtt{null}) \wedge \mathsf{arem}(\mathtt{ENQ})\right\}$$

```
      }
    }
  }
```

$$\left\{I \wedge \mathsf{arem}(\boldsymbol{skip})\right\}$$

```
}
```

Figure 38: Proof outline - II.

5 Soundness Proofs

Below we first prove the adequacy of RGSim-T w.r.t. the termination-sensitive refinement (Section 5.1). Then we define the unary judgment semantics (Section 5.2), and we prove the soundness of the binary inference rules of Figure 7 (Section 5.3), where the binary judgment semantics is just RGSim-T in Definition 2, and also prove the soundness of the unary rules of Figure 8 (Section 5.4). Finally we show the derivation of the WHILE-TERM rule (Section 5.5).

5.1 Adequacy of RGSim-T

RGSim-T in Definition 2 (which is also the binary judgment semantics) implies the termination-sensitive refinement in Definition 1.

Theorem 4 (Adequacy of RGSim-T). If there exist R, G, I, Q and a metric M such that $R, G, I \models (C, \sigma, M) \preceq_Q (\mathbb{C}, \Sigma)$, then $(C, \sigma) \sqsubseteq (\mathbb{C}, \Sigma)$.

Proof: We want to prove the following: for any R, G, I, Q,

$$\begin{array}{l} \forall \mathbb{C}, \Sigma, \mathcal{E}. \\ (\,\exists C, \sigma, M.\ R, G, I \models (C, \sigma, M) \preceq_Q (\mathbb{C}, \Sigma) \ \wedge\ ETr(C, \sigma, \mathcal{E})\,) \implies ETr(\mathbb{C}, \Sigma, \mathcal{E}) \end{array}$$

By co-induction.

$$\textsf{Co-induction Principle:}\ \ \forall x.\ \ (\exists S.\ S \subseteq F(S) \wedge x \in S) \implies x \in \text{gfp } F$$

Figure 3 defines F and gfp F (i.e., ETr). Let

$$S \stackrel{\text{def}}{=} \{(\mathbb{C}, \Sigma, \mathcal{E}) \mid \exists C, \sigma, M.\ R, G, I \models (C, \sigma, M) \preceq_Q (\mathbb{C}, \Sigma) \ \wedge\ ETr(C, \sigma, \mathcal{E})\}.$$

So from the co-induction principle, we only need to prove:

$$S \subseteq F(S), \ \text{ i.e., } \ \ \forall \mathbb{C}, \Sigma, \mathcal{E}.\ (\mathbb{C}, \Sigma, \mathcal{E}) \in S \implies (\mathbb{C}, \Sigma, \mathcal{E}) \in F(S)\,.$$

After unfolding S, we only need to prove:

$$\forall M, \mathbb{C}, \Sigma, \mathcal{E}, C, \sigma.\ R, G, I \models (C, \sigma, M) \preceq_Q (\mathbb{C}, \Sigma) \ \wedge\ ETr(C, \sigma, \mathcal{E}) \implies (\mathbb{C}, \Sigma, \mathcal{E}) \in F(S)\,. \tag{5.1}$$

By transfinite induction over M.

$$\textsf{Transfinite Induction Principle:}\ (\forall M.\ (\forall M'.\ M' < M \Longrightarrow P(M')) \Longrightarrow P(M)\,) \implies \forall M. P(M)$$

We view (5.1) as $\forall M. P(M)$. So we only need to prove:

$$\begin{array}{l} \forall M. \\ \quad (\forall M'.\ M' < M \\ \quad\ \implies (\forall \mathbb{C}', \Sigma', \mathcal{E}', C', \sigma'.\ R, G, I \models (C', \sigma', M') \preceq_Q (\mathbb{C}', \Sigma') \ \wedge\ ETr(C', \sigma', \mathcal{E}') \\ \qquad\qquad \implies (\mathbb{C}', \Sigma', \mathcal{E}') \in F(S))\,) \\ \implies \\ \quad (\forall \mathbb{C}, \Sigma, \mathcal{E}, C, \sigma.\ R, G, I \models (C, \sigma, M) \preceq_Q (\mathbb{C}, \Sigma) \ \wedge\ ETr(C, \sigma, \mathcal{E}) \\ \quad\ \implies (\mathbb{C}, \Sigma, \mathcal{E}) \in F(S)) \end{array}$$

By inversion over $ETr(C, \sigma, \mathcal{E})$,

1. $(C, \sigma) \longrightarrow^* (\mathbf{skip}, \sigma')$ and $\mathcal{E} = \Downarrow$:
   From $R, G, I \models (C, \sigma, M) \preceq_Q (\mathbb{C}, \Sigma)$, we know there exists Σ' such that $(\mathbb{C}, \Sigma) \longrightarrow^* (\boldsymbol{skip}, \Sigma')$.
   Thus from the definition of F (Figure 3), we know $(\mathbb{C}, \Sigma, \mathcal{E}) \in F(S)$.

2. $(C, \sigma) \longrightarrow^+ \textbf{abort}$ and $\mathcal{E} = \lightning$:
   From $R, G, I \models (C, \sigma, M) \preceq_Q (\mathbb{C}, \Sigma)$, we know $(\mathbb{C}, \Sigma) \longrightarrow^+ \textbf{abort}$.
   Thus from the definition of F (Figure 3), we know $(\mathbb{C}, \Sigma, \mathcal{E}) \in F(S)$.

3. $(C, \sigma) \longrightarrow^+ (C', \sigma')$ and $ETr(C', \sigma', \mathcal{E})$:
   From $R, G, I \models (C, \sigma, M) \preceq_Q (\mathbb{C}, \Sigma)$, we know one of the following two cases holds:

   (a) there exist M', $\mathbb{C}'$ and Σ' such that $(\mathbb{C}, \Sigma) \longrightarrow^+ (\mathbb{C}', \Sigma')$ and $R, G, I \models (C', \sigma', M') \preceq_Q (\mathbb{C}', \Sigma')$.
   Thus $(\mathbb{C}', \Sigma', \mathcal{E}) \in S$. Then from the definition of F (Figure 3), we know $(\mathbb{C}, \Sigma, \mathcal{E}) \in F(S)$.

   (b) there exists M' such that $M' < M$ and $R, G, I \models (C', \sigma', M') \preceq_Q (\mathbb{C}, \Sigma)$.
   Then from the induction hypothesis, we know $ETr(\mathbb{C}, \Sigma, \mathcal{E})$.

4. $(C, \sigma) \stackrel{e}{\longrightarrow}^+ (C', \sigma')$, $ETr(C', \sigma', \mathcal{E}')$ and $\mathcal{E} = e :: \mathcal{E}'$:
   From $R, G, I \models (C, \sigma, M) \preceq_Q (\mathbb{C}, \Sigma)$, we know:
   there exist $\mathbb{C}'$, Σ' and M' such that $(\mathbb{C}, \Sigma) \stackrel{e}{\longrightarrow}^+ (\mathbb{C}', \Sigma')$ and $R, G, I \models (C', \sigma', M') \preceq_Q (\mathbb{C}', \Sigma')$.
   Thus $(\mathbb{C}', \Sigma', \mathcal{E}') \in S$. Then from the definition of F (Figure 3), we know $(\mathbb{C}, \Sigma, \mathcal{E}) \in F(S)$.

Then we are done. $\Box$

5.2 Unary Judgment Semantics

The unary judgment semantics $R, G, I \models \{p\}C\{q\}$ follows RGSim-T (Definition 2). The initial abstract code in the simulation comes from the precondition p, and the postcondition q specifies the final abstract code that corresponds to the concrete final code **skip**. The assertions p and q also specify the while-specific metric w (the numbers of tokens), which must be related to the metric M used in the simulation RGSim-T.

Below we first show how we instantiate the abstract metric M in RGSim-T based on w.

5.2.1 Instantiation of the Abstract Metric M

For each single thread, its metric ws (defined below) is a list of (w, n) pairs, where w is the while-specific metric and n is "code size" which will be explained later. We let the threaded metric ws be a list (a stack actually) to allow different while-specific metrics for nested loops. That is, when entering a loop, we can push a (w, n) pair to the ws stack; and when exiting the loop, we pop the pair out of ws.

The threaded metric ws uses the dictionary order. However, the usual dictionary order over lists is not well-founded (consider $B > AB > AAB > AAAB > \ldots$ in a dictionary). To address this issue, we introduce a bound of the list length (stack height), $\mathcal{H}$, and define the well-founded order $<_{\mathcal{H}}$ by requiring the lists should be not longer than $\mathcal{H}$. Intuitively, the stack height $\mathcal{H}$ represents the maximal depth of nested loops, so it can be determined for any given program.

To get the whole-program metric, we compose threaded metrics by pairing them. Thus the abstract metric M in RGSim-T is instantiated as follows:

$$M \ ::= \ (ws, \mathcal{H}) \mid (M, M)$$

and we define the well-founded oder $<$ and the composition operation $+$ (see Lemma 16) as follows:

$$\frac{ws' <_{\mathcal{H}} ws \qquad \mathcal{H}' = \mathcal{H}}{(ws', \mathcal{H}') < (ws, \mathcal{H})} \qquad \frac{M_1' < M_1 \qquad M_2' = M_2}{(M_1', M_2') < (M_1, M_2)} \qquad \frac{M_1' = M_1 \qquad M_2' < M_2}{(M_1', M_2') < (M_1, M_2)}$$

$$M_1 + M_2 \ \stackrel{\text{def}}{=} \ (M_1, M_2)$$

The threaded metric ws and the well-founded order $<_{\mathcal{H}}$ are defined below. Note that we allow "$A < AB < B$" in a dictionary.

$$\begin{array}{rrcl}
(\mathit{WfStack}) & ws & ::= & (w, n) \mid (w, n) :: ws \\
(\mathit{StkHeight}) & \mathcal{H} & \in & \mathit{Nat}
\end{array}$$

$$ws' <_{\mathcal{H}} ws \quad \text{iff} \quad (ws' \ll ws) \wedge (|ws'| \leq \mathcal{H}) \wedge (|ws| \leq \mathcal{H})$$

$$\frac{(w', n') < (w, n)}{(w', n') \ll (w, n)} \qquad \frac{(w', n') \leq (w, n)}{(w', n') \ll (w, n) :: ws_1} \qquad \frac{(w', n') < (w, n)}{(w', n') :: ws_1' \ll (w, n)}$$

$$\frac{(w', n') < (w, n)}{(w', n') :: ws_1' \ll (w, n) :: ws_1} \qquad \frac{(w', n') = (w, n) \qquad ws_1' \ll ws_1}{(w', n') :: ws_1' \ll (w, n) :: ws_1}$$

Here $|ws|$ is the length of ws, which is defined as follows:

$$\begin{array}{rcl}
|(w, n)| & = & 1 \\
|(w, n) :: ws| & = & 1 + |ws|
\end{array}$$

The well-founded order over the (w, n) pairs is a usual dictionary order:

$$\begin{array}{rcl}
(w', n') < (w, n) & \text{iff} & (w' < w) \vee (w' = w \wedge n' < n) \\
(w', n') = (w, n) & \text{iff} & (w' = w) \wedge (n' = n) \\
(w', n') \leq (w, n) & \text{iff} & (w', n') < (w, n) \vee (w', n') = (w, n)
\end{array}$$

Lemma 5 (Well-foundedness). The relation $M' < M$ defined above is a well-founded relation.

Proof: Easy to prove from Lemma 6. □

Lemma 6. The relation $ws' <_{\mathcal{H}} ws$ defined above is a well-founded relation.

Proof: Suppose there is an infinite descending chain:

$$ws_0 > ws_1 > ws_2 > \dots \tag{5.2}$$

Thus we know

$$ws_0 \gg ws_1 \gg ws_2 \gg \dots \tag{5.3}$$

and

$$\forall k.\ |ws_k| \leq \mathcal{H} \tag{5.4}$$

We prove the following property which generalizes (5.4) over the maximum size $\mathcal{H}$:

$$\forall ws_0, ws_1, ws_2, \dots .\ (\forall k.\ ws_k \gg ws_{k+1}) \implies (\forall m \geq 1.\ \exists j.\ |ws_j| > m) \tag{5.5}$$

By induction over m.

- **Base Case:** $m = 1$. Suppose $\forall k.\ |ws_k| = 1$. Thus we have an infinite descending chain:

$$(w_0, n_0) > (w_1, n_1) > (w_2, n_2) > \dots \tag{5.6}$$

  It violates the definition of $(w', n') < (w, n)$ (which is a well-founded relation).

- **Inductive Step:** $m = m' + 1$. Since $(w', n') < (w, n)$ is a well-founded relation, we know there must exists k such that

$$\forall j \geq k.\ \mathsf{root}(ws_j) = \mathsf{root}(ws_{j+1}) \tag{5.7}$$

  and there exist ws'_k, ws'_{k+1}, ws'_{k+2}, … such that $\forall j \geq k.\ ws_j = \mathsf{root}(ws_j) :: ws'_j$ and

$$\forall j \geq k.\ ws'_j \gg ws'_{j+1} \tag{5.8}$$

  Here $\mathsf{root}(ws)$ takes the first element of ws if ws has the first element and undefined otherwise. From the induction hypothesis, we know there exists $j \geq k$ such that

$$|ws'_j| > m'. \tag{5.9}$$

  Thus $|ws_j| > m' + 1$.

So we are done. □

5.2.2 Intuitions of $\mathcal{H}$ and the Second Dimension of ws

Below we give more informal explanations (and examples) about the stack height $\mathcal{H}$ and the second dimension ("code size" n in each pair) of the threaded metric ws.

As we said, the stack height $\mathcal{H}$ represents the maximal depth of nested loops. For any given program C, we can determine the stack height using a function height defined in Figure 39.

The threaded metric ws as a stack requires us to distinguish the executions of the loop body from the executions of the code out of the loop. When entering a loop (for the first time), we can push a (w, n) pair onto the ws stack. But when we repeatedly execute the loop body (not for the first time), we do not want to push a new pair onto the stack.

Thus we introduce the runtime command $\mathsf{while}\ (B)\{C\}$ to represent the while-loop continuation when we have unfolded the loop **while** (B) C. And we revised the low-level operational semantics as follows:

$$
\begin{array}{rcl}
\mathsf{height}(\mathbf{skip}) & = & 1 \\
\mathsf{height}(c) & = & 1 \\
\mathsf{height}(\langle C\rangle) & = & 1 \\
\mathsf{height}(C_1; C_2) & = & max\{\mathsf{height}(C_1), \mathsf{height}(C_2)\} \\
\mathsf{height}(\mathbf{if}\ (B)\ C_1\ \mathbf{else}\ C_2) & = & max\{\mathsf{height}(C_1), \mathsf{height}(C_2)\} \\
\mathsf{height}(\mathbf{while}\ (B)\ C) & = & \mathsf{height}(C) + 1
\end{array}
$$

Figure 39: Definition of height.

$$
\frac{[\![B]\!]_s = \mathbf{true}}{(\mathbf{while}\ (B)\ C, (s,h)) \longrightarrow (C; \mathsf{while}\ (B)\{C\}, (s,h))}
\qquad
\frac{[\![B]\!]_s = \mathbf{false}}{(\mathbf{while}\ (B)\ C, (s,h)) \longrightarrow (\mathbf{skip}, (s,h))}
$$

$$
\frac{[\![B]\!]_s = \mathbf{true}}{(\mathsf{while}\ (B)\{C\}, (s,h)) \longrightarrow (C; \mathsf{while}\ (B)\{C\}, (s,h))}
\qquad
\frac{[\![B]\!]_s = \mathbf{false}}{(\mathsf{while}\ (B)\{C\}, (s,h)) \longrightarrow (\mathbf{skip}, (s,h))}
$$

We can see that the new operational semantics for while loops is equivalent to the original one (see Figure 2). Below we will assume the new semantics and use it to prove the logic soundness. However, we want the readers to note that without the new operational semantics, we can still define the unary judgment semantics and prove the soundness of *all* the inference rules, based on the original operational semantics. The new operational semantics for while loops just makes the proofs (and the intuition) clearer, in particular, for the HIDE-W rule, the rule for "locally" reasoning about nested while loops.

With the runtime $\mathsf{while}\ (B)\{C\}$, we can calculate the code size n in each (w, n) pair of ws. We first label the code such that different layers of a nested while loop are assigned different labels.

Labeling the Code The syntax of the labeled code is defined below. Its operational semantics is straightforward, as shown in Figure 40.

$$
\begin{array}{rrcl}
(\mathit{Label}) & l & \in & \mathit{Nat} \\
(\mathit{LabStmt}) & \widehat{C} & ::= & \mathbf{skip}^l \mid c^l \mid \langle C\rangle^l \mid \widehat{C}_1; \widehat{C}_2 \mid \mathbf{if}^l(B)\ \widehat{C}_1\ \mathbf{else}\ \widehat{C}_2 \\
 & & \mid & \mathbf{while}^l(B)\ \widehat{C} \mid \mathsf{while}^l(B)\ \widehat{C}
\end{array}
$$

We label the low-level code in the following way. Note that we do not need to label the runtime command $\mathsf{while}\ (B)\{C\}$, whose label is known during the runtime execution.

$$
\begin{array}{rcl}
\mathsf{labeling}(\mathbf{skip}, l) & = & \mathbf{skip}^l \\
\mathsf{labeling}(c, l) & = & c^l \\
\mathsf{labeling}(\langle C\rangle, l) & = & \langle C\rangle^l \\
\mathsf{labeling}(C_1; C_2, l) & = & \mathsf{labeling}(C_1, l); \mathsf{labeling}(C_2, l) \\
\mathsf{labeling}(\mathbf{if}\ (B)\ C_1\ \mathbf{else}\ C_2, l) & = & \mathbf{if}^l(B)\ \mathsf{labeling}(C_1, l)\ \mathbf{else}\ \mathsf{labeling}(C_2, l) \\
\mathsf{labeling}(\mathbf{while}\ (B)\ C, l) & = & \mathbf{while}^l(B)\ \mathsf{labeling}(C, l+1)
\end{array}
$$

We define the functions label, $\mathsf{toplabel}$, $\mathsf{minlabel}$ and $\mathsf{maxlabel}$ in Figure 41. Then the stack height $\mathcal{H}$ of C is actually the maximum label of $\widehat{C}$, which is obtained by labeling C with 1. That is, the following holds:

$$
\mathsf{height}(C) \ = \ \mathsf{maxlabel}(\mathsf{labeling}(C, 1))
$$

We can prove the following property.

Lemma 7. For any C, $\widehat{C}$, $\widehat{C}'$, σ, σ' and R, if $\mathsf{labeling}(C, 1) = \widehat{C}$ and $(\widehat{C}, \sigma) \overset{R}{\longmapsto}{}^* (\widehat{C}', \sigma')$, then there exist l, $\widehat{C}_1$, …, $\widehat{C}_l$ such that $\widehat{C}' = (\widehat{C}_l; \ldots; \widehat{C}_1)$ and $\forall i \in [1..l]$. $\mathsf{label}(\widehat{C}_i) = i$.

$$\frac{[\![B]\!]_s = \mathbf{true}}{(\mathbf{while}^l(B)\ \widehat{C}, (s,h)) \longrightarrow (\widehat{C}; \mathsf{while}^l(B)\ \widehat{C}, (s,h))} \qquad \frac{[\![B]\!]_s = \mathbf{false}}{(\mathbf{while}^l(B)\ \widehat{C}, (s,h)) \longrightarrow (\mathbf{skip}^l, (s,h))}$$

$$\frac{[\![B]\!]_s = \mathbf{true}}{(\mathsf{while}^l(B)\ \widehat{C}, (s,h)) \longrightarrow (\widehat{C}; \mathsf{while}^l(B)\ \widehat{C}, (s,h))} \qquad \frac{[\![B]\!]_s = \mathbf{false}}{(\mathsf{while}^l(B)\ \widehat{C}, (s,h)) \longrightarrow (\mathbf{skip}^l, (s,h))}$$

$$\frac{(\widehat{C}, \sigma) \longrightarrow (\widehat{C}', \sigma')}{(\widehat{C}; \widehat{C}'', \sigma) \longrightarrow (\widehat{C}'; \widehat{C}'', \sigma')} \qquad \frac{}{(\mathbf{skip}^l; \widehat{C}', \sigma) \longrightarrow (\widehat{C}', \sigma)}$$

$$\frac{(\widehat{C}, \sigma) \longrightarrow (\widehat{C}', \sigma')}{(\widehat{C}, \sigma) \overset{R}{\longmapsto} (\widehat{C}', \sigma')} \qquad \frac{((\sigma, \Sigma), (\sigma', \Sigma'), b) \models R}{(\widehat{C}, \sigma) \overset{R}{\longmapsto} (\widehat{C}, \sigma')}$$

Figure 40: Selected operational semantics rules of the labeled language.

$$\begin{array}{rcl}
\mathsf{label}(\mathbf{skip}^l) & = & l \\
\mathsf{label}(c^l) & = & l \\
\mathsf{label}(\langle C \rangle^l) & = & l \\
\mathsf{label}(\widehat{C}_1; \widehat{C}_2) & = & \begin{cases} \mathsf{label}(\widehat{C}_1) & \text{if } \mathsf{label}(\widehat{C}_1) = \mathsf{label}(\widehat{C}_2) \\ \textit{undefined} & \text{otherwise} \end{cases} \\
\mathsf{label}(\mathbf{if}^l(B)\ \widehat{C}_1\ \mathbf{else}\ \widehat{C}_2) & = & l \\
\mathsf{label}(\mathbf{while}^l(B)\ \widehat{C}) & = & l \\
\mathsf{label}(\mathsf{while}^l(B)\ \widehat{C}) & = & l
\end{array}$$

$$\begin{array}{rcl}
\mathsf{minlabel}(\mathbf{skip}^l) & = & l \\
\mathsf{minlabel}(c^l) & = & l \\
\mathsf{minlabel}(\langle C \rangle^l) & = & l \\
\mathsf{minlabel}(\widehat{C}_1; \widehat{C}_2) & = & \mathsf{minlabel}(\widehat{C}_2) \\
\mathsf{minlabel}(\mathbf{if}^l(B)\ \widehat{C}_1\ \mathbf{else}\ \widehat{C}_2) & = & l \\
\mathsf{minlabel}(\mathbf{while}^l(B)\ \widehat{C}) & = & l \\
\mathsf{minlabel}(\mathsf{while}^l(B)\ \widehat{C}) & = & l
\end{array}
\qquad
\begin{array}{rcl}
\mathsf{maxlabel}(\mathbf{skip}^l) & = & l \\
\mathsf{maxlabel}(c^l) & = & l \\
\mathsf{maxlabel}(\langle C \rangle^l) & = & l \\
\mathsf{maxlabel}(\widehat{C}_1; \widehat{C}_2) & = & max\{\mathsf{maxlabel}(\widehat{C}_1), \mathsf{maxlabel}(\widehat{C}_2)\} \\
\mathsf{maxlabel}(\mathbf{if}^l(B)\ \widehat{C}_1\ \mathbf{else}\ \widehat{C}_2) & = & max\{\mathsf{maxlabel}(\widehat{C}_1), \mathsf{maxlabel}(\widehat{C}_2)\} \\
\mathsf{maxlabel}(\mathbf{while}^l(B)\ \widehat{C}) & = & \mathsf{maxlabel}(\widehat{C})
\end{array}$$

Figure 41: Functions on labeled code.

It says, at any time in the execution of $\widehat{C}$, the runtime code must be in the form of $\widehat{C}_l; \widehat{C}_{l-1} \ldots ; \widehat{C}_1$, where each $\widehat{C}_i$ has a fixed label i.

Code Sizes for Labeled Code For each pair (w, n) in any ws, n can be statically determined by the code. We use $\mathsf{proj}_2(ws)$ to project each pair (w, n) in ws to n. $\mathsf{proj}_1(ws)$ is defined similarly.

$$ns \quad ::= \quad n \mid n :: ns$$

$$\begin{array}{rcl} \mathsf{proj}_2(w, n) & = & n \\ \mathsf{proj}_2((w, n) :: ws) & = & n :: \mathsf{proj}_2(ws) \end{array}$$

We use $[\![\widehat{C}]\!]$ to compute a list of code sizes for $\widehat{C}$. Then

$\mathsf{proj}_2(ws) = [\![\widehat{C}]\!]$, where $\widehat{C}$ is some run-time labeled code and ws is the metric for $\widehat{C}$.

We define $[\![\widehat{C}]\!]$ as follows.

$$\begin{array}{rcl} [\![\mathbf{skip}^l]\!] & = & 0 \\ [\![c^l]\!] & = & 1 \\ [\![\langle C\rangle^l]\!] & = & 1 \\ [\![\widehat{C}_1; \widehat{C}_2]\!] & = & \begin{cases} [\![\widehat{C}_1]\!] \oplus |\widehat{C}_2| \oplus 1 & \text{if } \mathsf{minlabel}(\widehat{C}_1) = \mathsf{label}(\widehat{C}_2) \\ |\widehat{C}_2| :: ([\![\widehat{C}_1]\!] \oplus 1) & \text{if } \mathsf{minlabel}(\widehat{C}_1) > \mathsf{label}(\widehat{C}_2) \end{cases} \\ [\![\mathbf{if}^l(B)\ \widehat{C}_1 \ \mathbf{else}\ \widehat{C}_2]\!] & = & max\{|\widehat{C}_1|, |\widehat{C}_2|\} + 1 \\ [\![\mathbf{while}^l(B)\ \widehat{C}]\!] & = & 1 \\ [\![\mathsf{while}^l(B)\ \widehat{C}]\!] & = & 0 :: 0 \end{array}$$

Here the static size of commands $|\widehat{C}|$ is defined as follows.

$$\begin{array}{rcl} |\mathbf{skip}^l| & = & 0 \\ |c^l| & = & 1 \\ |\langle C\rangle^l| & = & 1 \\ |\widehat{C}_1; \widehat{C}_2| & = & |\widehat{C}_1| + |\widehat{C}_2| + 1 \\ |\mathbf{if}^l(B)\ \widehat{C}_1 \ \mathbf{else}\ \widehat{C}_2| & = & max\{|\widehat{C}_1|, |\widehat{C}_2|\} + 1 \\ |\mathbf{while}^l(B)\ \widehat{C}| & = & 1 \\ |\mathsf{while}^l(B)\ \widehat{C}| & = & 0 \end{array}$$

And $ns \oplus n$ is defined as follows:

$$ns \oplus n \ \stackrel{\text{def}}{=} \ \begin{cases} n_1 + n & \text{if } ns = n_1 \\ (n_1 + n) :: ns' & \text{if } ns = n_1 :: ns' \\ \mathit{undefined} & \text{otherwise} \end{cases}$$

Examples of ws Below we use a few simple examples to show how ws changes during an execution. The second dimension of the ws for the runtime labeled code $\widehat{C}$ coincides with the above definition $[\![\widehat{C}]\!]$.

		C	σ	ws
1		`while`^{1}`(i > 0) i--`^{2}`;`	`i` $= 2$	$(0, 1)$
2	$\to$	`i--`^{2}`;` while^1`(i > 0) i--`^{2}`;`	`i` $= 2$	$(0, 0) :: (1, 2)$
3	$\to$	`skip`^{2}`;` while^1`(i > 0) i--`^{2}`;`	`i` $= 1$	$(0, 0) :: (1, 1)$
4	$\to$	while^1`(i > 0) i--`^{2}`;`	`i` $= 1$	$(0, 0) :: (1, 0)$
5	$\to$	`i--`^{2}`;` while^1`(i > 0) i--`^{2}`;`	`i` $= 1$	$(0, 0) :: (0, 2)$
6	$\to$	`skip`^{2}`;` while^1`(i > 0) i--`^{2}`;`	`i` $= 0$	$(0, 0) :: (0, 1)$
7	$\to$	while^1`(i > 0) i--`^{2}`;`	`i` $= 0$	$(0, 0) :: (0, 0)$
8	$\to$	`skip`^{1}`;`	`i` $= 0$	$(0, 0)$

		C	σ	ws
1		i:=2^{1}; while^{1}(i>0){ j:=1^{2}; while^{2}(j>0){j--^{3};}; i--^{2}; }	$\mathtt{i}=0, \mathtt{j}=0$	$(0,3)$
2	$\to$	skip^{1}; while^{1}(i>0){ j:=1^{2}; while^{2}(j>0){j--^{3};}; i--^{2}; }	$\mathtt{i}=2, \mathtt{j}=0$	$(0,2)$
3	$\to$	while^{1}(i>0){ j:=1^{2}; while^{2}(j>0){j--^{3};}; i--^{2}; }	$\mathtt{i}=2, \mathtt{j}=0$	$(0,1)$
4	$\to$	j:=1^{2}; while^{2}(j>0){j--^{3};}; i--^{2}; while^{1}(i>0){...}	$\mathtt{i}=2, \mathtt{j}=0$	$(0,0)::(1,6)$
5	$\to$	skip^{2}; while^{2}(j>0){j--^{3};}; i--^{2}; while^{1}(i>0){...}	$\mathtt{i}=2, \mathtt{j}=1$	$(0,0)::(1,5)$
6	$\to$	while^{2}(j>0){j--^{3};}; i--^{2}; while^{1}(i>0){...}	$\mathtt{i}=2, \mathtt{j}=1$	$(0,0)::(1,4)$
7	$\to$	j--^{3}; while^{2}(j>0){j--^{3};}; i--^{2}; while^{1}(i>0){...}	$\mathtt{i}=2, \mathtt{j}=1$	$(0,0)::(1,3)::(0,2)$
8	$\to$	skip^{3}; while^{2}(j>0){j--^{3};}; i--^{2}; while^{1}(i>0){...}	$\mathtt{i}=2, \mathtt{j}=0$	$(0,0)::(1,3)::(0,1)$
9	$\to$	while^{2}(j>0){j--^{3};}; i--^{2}; while^{1}(i>0){...}	$\mathtt{i}=2, \mathtt{j}=0$	$(0,0)::(1,3)::(0,0)$
10	$\to$	skip^{2}; i--^{2}; while^{1}(i>0){...}	$\mathtt{i}=2, \mathtt{j}=0$	$(0,0)::(1,3)$
11	$\to$	i--^{2}; while^{1}(i>0){...}	$\mathtt{i}=2, \mathtt{j}=0$	$(0,0)::(1,2)$
12	$\to$	skip^{2}; while^{1}(i>0){...}	$\mathtt{i}=1, \mathtt{j}=0$	$(0,0)::(1,1)$
13	$\to$	while^{1}(i>0){ j:=1^{2}; while^{2}(j>0){j--^{3};}; i--^{2}; }	$\mathtt{i}=1, \mathtt{j}=0$	$(0,0)::(1,0)$
14	$\to$	j:=1^{2}; while^{2}(j>0){j--^{3};}; i--^{2}; while^{1}(i>0){...}	$\mathtt{i}=1, \mathtt{j}=0$	$(0,0)::(0,6)$
15	$\to$	skip^{2}; while^{2}(j>0){j--^{3};}; i--^{2}; while^{1}(i>0){...}	$\mathtt{i}=1, \mathtt{j}=1$	$(0,0)::(0,5)$
16	$\to$	while^{2}(j>0){j--^{3};}; i--^{2}; while^{1}(i>0){...}	$\mathtt{i}=1, \mathtt{j}=1$	$(0,0)::(0,4)$
17	$\to$	j--^{3}; while^{2}(j>0){j--^{3};}; i--^{2}; while^{1}(i>0){...}	$\mathtt{i}=1, \mathtt{j}=1$	$(0,0)::(0,3)::(0,2)$
18	$\to$	skip^{3}; while^{2}(j>0){j--^{3};}; i--^{2}; while^{1}(i>0){...}	$\mathtt{i}=1, \mathtt{j}=0$	$(0,0)::(0,3)::(0,1)$
19	$\to$	while^{2}(j>0){j--^{3};}; i--^{2}; while^{1}(i>0){...}	$\mathtt{i}=1, \mathtt{j}=0$	$(0,0)::(0,3)::(0,0)$
20	$\to$	skip^{2}; i--^{2}; while^{1}(i>0){...}	$\mathtt{i}=1, \mathtt{j}=0$	$(0,0)::(0,3)$
21	$\to$	i--^{2}; while^{1}(i>0){...}	$\mathtt{i}=1, \mathtt{j}=0$	$(0,0)::(0,2)$
22	$\to$	skip^{2}; while^{1}(i>0){...}	$\mathtt{i}=0, \mathtt{j}=0$	$(0,0)::(0,1)$
23	$\to$	while^{1}(i>0){ j:=1^{2}; while^{2}(j>0){j--^{3};}; i--^{2}; }	$\mathtt{i}=0, \mathtt{j}=0$	$(0,0)::(0,0)$
24	$\to$	skip^{1}	$\mathtt{i}=0, \mathtt{j}=0$	$(0,0)$

The next example is a loop that uses the counter. It involves environment steps, denoted by R, and defined in Section 4.1. When the environment updates `x` (see line 7), we increase the number of tokens by 1, i.e., w at the outermost pair of the stack ws is increased from 0 to 1.

		C	σ	ws

1		`while`^{1}`(i > 0){`
		`b:=false`^{2}`;`
		`while`^{2}`(!b){ t:=x`^{3}`;b:=cas(&x,t,t+1)`^{3}`;if`^{3}`(b) i--`^{3}`; };`
		`}`
			$\mathtt{x}=5$
			$\mathtt{i}=1$
			$\mathtt{b}=\mathbf{false}$
			$\mathtt{t}=0$
				$(0,1)$

2	$\rightarrow$	`b:=false`^{2}`; while`^{2}`(!b){...}; while`^{1}`(i > 0){...}`	...	$(0,0)::(0,4)$
3	$\rightarrow$	`skip`^{2}`; while`^{2}`(!b){...}; while`^{1}`(i > 0){...}`	...	$(0,0)::(0,3)$
4	$\rightarrow$	`while`^{2}`(!b){...}; while`^{1}`(i > 0){...}`	...	$(0,0)::(0,2)$
5	$\rightarrow$	`t:=x`^{3}`;b:=cas(&x,t,t+1)`^{3}`;if`^{3}`(b) i--`^{3}`;` `while`^{2}`(!b){...}; while`^{1}`(i > 0){...}`	...	$(0,0)::(0,1)::(0,7)$
6	$\rightarrow$	`skip`^{3}`;b:=cas(&x,t,t+1)`^{3}`;if`^{3}`(b) i--`^{3}`;` `while`^{2}`(!b){...}; while`^{1}`(i > 0){...}`	$\mathtt{x}=5$ ... $\mathtt{t}=5$	$(0,0)::(0,1)::(0,6)$
7	R		$\mathtt{x}=8,\ldots$	$(0,0)::(0,1)::(1,6)$
8	$\rightarrow^*$	`while`^{2}`(!b){...}; while`^{1}`(i > 0){...}`	$\mathtt{x}=8$ $\mathtt{i}=1$ $\mathtt{b}=\mathbf{false}$ $\mathtt{t}=5$	$(0,0)::(0,1)::(1,0)$
9	$\rightarrow$	`t:=x`^{3}`;b:=cas(&x,t,t+1)`^{3}`;if`^{3}`(b) i--`^{3}`;` `while`^{2}`(!b){...}; while`^{1}`(i > 0){...}`	...	$(0,0)::(0,1)::(0,7)$
10	$\rightarrow^*$	`while`^{2}`(!b){...}; while`^{1}`(i > 0){...}`	$\mathtt{x}=8$ $\mathtt{i}=0$ $\mathtt{b}=\mathbf{true}$ $\mathtt{t}=8$	$(0,0)::(0,1)::(0,0)$
11	$\rightarrow$	`skip`^{2}`; while`^{1}`(i > 0){...}`	...	$(0,0)::(0,1)$
12	$\rightarrow$	`while`^{1}`(i > 0){...}`	...	$(0,0)::(0,0)$
13	$\rightarrow$	`skip`^{1}`;`	...	$(0,0)$

Note that in this section we assume that the outer loop and the inner loop each uses a "local" while-specific metric w. The intuition explained here actually shows how we prove the soundness of the WHILE-L rule. For the WHILE rule, we use a "global" while-specific metric, and hence the depth of ws could be just 1 and we do not need to push a new (w, n) pair whenever entering a loop. In this case, the second dimension of ws, i.e., the size of the code, will count in the runtime while command $\mathsf{while}\ (B)\{C\}$ too. We show a simple example below, where the stack ws is always of depth 1.

		C	σ	ws
1		`while`^{1}`(i > 0) i--`^{2}`;`	$\mathtt{i}=2$	$(2,1)$
2	$\rightarrow$	`i--`^{2}`; while`^{1}`(i > 0) i--`^{2}`;`	$\mathtt{i}=2$	$(1,3)$
3	$\rightarrow$	`skip`^{2}`; while`^{1}`(i > 0) i--`^{2}`;`	$\mathtt{i}=1$	$(1,2)$
4	$\rightarrow$	`while`^{1}`(i > 0) i--`^{2}`;`	$\mathtt{i}=1$	$(1,0)$
5	$\rightarrow$	`i--`^{2}`; while`^{1}`(i > 0) i--`^{2}`;`	$\mathtt{i}=1$	$(0,3)$
6	$\rightarrow$	`skip`^{2}`; while`^{1}`(i > 0) i--`^{2}`;`	$\mathtt{i}=0$	$(0,2)$
7	$\rightarrow$	`while`^{1}`(i > 0) i--`^{2}`;`	$\mathtt{i}=0$	$(0,1)$
8	$\rightarrow$	`skip`^{1}`;`	$\mathtt{i}=0$	$(0,0)$

5.2.3 Unary Judgment Semantics

Definition 8. $R, G, I \models \{p\}C\{q\}$ iff
for all σ, w, $\mathbb{D}$ and Σ, if $(\sigma, w, \mathbb{D}, \Sigma) \models p$, then $R, G, I \models (C, \sigma, (0, |C|)) \preceq_{\mathsf{height}(C);w;q} (\mathbb{D}, \Sigma)$.

Whenever $R, G, I \models (C, \sigma, ws) \preceq_{\mathcal{H};w;q} (\mathbb{D}, \Sigma)$, then $(\sigma, \Sigma) \models I * \mathbf{true}$ and the following are true:

1. for any σ_F, Σ_F, C' and σ'', if $(C, \sigma \uplus \sigma_F) \longrightarrow (C', \sigma'')$ and $\Sigma \bot \Sigma_F$, then there exists σ' such that $\sigma'' = \sigma' \uplus \sigma_F$ and one of the following holds:
   (a) either, there exist ws', w', $\mathbb{C}'$ and Σ' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', ws') \preceq_{\mathcal{H};w';q} (\mathbb{C}', \Sigma')$;
   (b) or, there exists ws' such that $ws' <_{\mathcal{H}} ws$, $((\sigma, \Sigma), (\sigma', \Sigma), \mathbf{false}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', ws') \preceq_{\mathcal{H};w;q} (\mathbb{D}, \Sigma)$;
2. for any σ_F, Σ_F, e, C' and σ'', if $(C, \sigma \uplus \sigma_F) \xrightarrow{e} (C', \sigma'')$ and $\Sigma \bot \Sigma_F$, then there exist σ', ws', w', $\mathbb{C}'$ and Σ' such that $\sigma'' = \sigma' \uplus \sigma_F$, $(\mathbb{D}, \Sigma \uplus \Sigma_F) \xrightarrow{e}{}^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', ws') \preceq_{\mathcal{H};w';q} (\mathbb{C}', \Sigma')$;
3. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models R^+ * \mathsf{Id}$, then there exist ws' and w' such that $R, G, I \models (C, \sigma', ws') \preceq_{\mathcal{H};w';q} (\mathbb{D}, \Sigma')$;
4. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{false}) \models R^+ * \mathsf{Id}$, then $R, G, I \models (C, \sigma', ws) \preceq_{\mathcal{H};w;q} (\mathbb{D}, \Sigma')$;
5. if $C = \mathbf{skip}$, then for any Σ_F, if $\Sigma \bot \Sigma_F$, one of the following holds:
   (a) either, there exist w', $\mathbb{C}'$ and Σ' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma, \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma, w', \mathbb{C}', \Sigma') \models q$;
   (b) or, there exists w' such that $ws = (w', 0)$ and $(\sigma, w + w', \mathbb{D}, \Sigma) \models q$;
6. for any σ_F and Σ_F, if $(C, \sigma \uplus \sigma_F) \longrightarrow \mathbf{abort}$ and $\Sigma \bot \Sigma_F$, then $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ \mathbf{abort}$.

Definition 9 (SL Judgment Semantics).
$\models_{\text{SL}} [p]C[q]$ iff, for all σ, w, $\mathbb{D}$ and Σ, if $(\sigma, w, \mathbb{D}, \Sigma) \models p$, the following are true:

1. for any σ', if $(C, \sigma) \longrightarrow^* (\mathbf{skip}, \sigma')$, then $(\sigma', w, \mathbb{D}, \Sigma) \models q$;
2. $(C, \sigma) \not\longrightarrow^* \mathbf{abort}$;
3. $(C, \sigma) \not\longrightarrow^\omega \cdot$.

$\models_{\text{SL}} [P]\mathbb{C}[Q]$ iff, for any σ and Σ, if $(\sigma, \Sigma) \models P$, the following are true:

1. for any Σ', if $(\mathbb{C}, \Sigma) \longrightarrow^* (\boldsymbol{skip}, \Sigma')$, then $(\sigma, \Sigma') \models Q$;
2. $(\mathbb{C}, \Sigma) \not\longrightarrow^* \mathbf{abort}$;
3. $(\mathbb{C}, \Sigma) \not\longrightarrow^\omega \cdot$.

Definition 10 (Locality).
$\mathsf{Locality}(C)$ iff, for any σ_1 and σ_2, let $\sigma = \sigma_1 \uplus \sigma_2$, then the following hold:

1. (Safety monotonicity) If $(C, \sigma_1) \not\longrightarrow^* \mathbf{abort}$, then $(C, \sigma) \not\longrightarrow^* \mathbf{abort}$.
2. (Termination monotonicity) If $(C, \sigma_1) \not\longrightarrow^* \mathbf{abort}$ and $(C, \sigma_1) \not\longrightarrow^\omega \cdot$, then $(C, \sigma) \not\longrightarrow^\omega \cdot$.
3. (Frame property) For any n and σ', if $(C, \sigma_1) \not\longrightarrow^* \mathbf{abort}$ and $(C, \sigma) \longrightarrow^n (C', \sigma')$, then there exists σ_1' such that $\sigma' = \sigma_1' \uplus \sigma_2$ and $(C, \sigma_1) \longrightarrow^n (C', \sigma_1')$.

$\mathsf{Locality}(\mathbb{C})$ is defined similarly.

5.3 Soundness of Binary Rules

Lemma 11. If $R, G, I \vdash \{P\}C \preceq \mathbb{C}\{Q\}$, then $I \triangleright \{R, G\}$, $P \vee Q \Rightarrow I * \mathbf{true}$ and $\mathsf{Sta}(\{P, Q\}, R * \mathsf{Id})$.

Proof: By induction over the derivation of $R, G, I \vdash \{P\}C \preceq \mathbb{C}\{Q\}$, and by Lemma 27. For the stability, we need Lemmas 12, 13 and 14. □

Lemma 12. If $\mathsf{Sta}(p \wedge B, R * \mathsf{Id})$, $\mathsf{Sta}(p \wedge \neg B, R * \mathsf{Id})$ and $p \Rightarrow (B = B)$, then $\mathsf{Sta}(p, R * \mathsf{Id})$.

Lemma 13. If $\mathsf{Sta}(p, R * \mathsf{Id})$, $p \Rightarrow (B = B) * I$ and $I \triangleright R$, then $\mathsf{Sta}(p \wedge B, R * \mathsf{Id})$.

Lemma 14. If $\mathsf{Sta}(p_1, R_1 * \mathsf{Id})$, $\mathsf{Sta}(p_2, R_2 * \mathsf{Id})$, $I_1 \triangleright R_1$, $I_2 \triangleright R_2$, $p_1 \Rightarrow I_1 * \mathbf{true}$, $p_2 \Rightarrow I_2 * \mathbf{true}$, then $\mathsf{Sta}(p_1 * p_2, R_1 * R_2 * \mathsf{Id})$.

The B-PAR rule. We define $M_1 + M_2$ as a pair (M_1, M_2). The corresponding well-founded order satisfies the following:

$$(M_1 < M_2) \implies (M_1 + M_3 < M_2 + M_3) \tag{5.10}$$

$$(M_1 < M_2) \implies (M_3 + M_1 < M_3 + M_2) \tag{5.11}$$

Lemma 15 (Parallel Compositioinality). If

1. $R \vee G_2, G_1, I \models \{P_1 * P\}C_1 \preceq \mathbb{C}_1\{Q_1 * Q_1'\}$;
2. $R \vee G_1, G_2, I \models \{P_2 * P\}C_2 \preceq \mathbb{C}_2\{Q_2 * Q_2'\}$;
3. $P \vee Q_1' \vee Q_2' \Rightarrow I$; $I \triangleright \{R, G_1, G_2\}$; $\mathsf{Sta}(Q_1 * Q_1', (R \vee G_2) * \mathsf{Id})$; $\mathsf{Sta}(Q_2 * Q_2', (R \vee G_1) * \mathsf{Id})$;

then $R, G_1 \vee G_2, I \models \{P_1 * P_2 * P\}C_1 \,\|\, C_2 \preceq \mathbb{C}_1 \,|||\, \mathbb{C}_2\{Q_1 * Q_2 * (Q_1' \wedge Q_2')\}$.

Proof: We need to prove: for all σ and Σ, if $(\sigma, \Sigma) \models P_1 * P_2 * P$, then there exists M such that $R, G_1 \vee G_2, I \models (C_1 \,\|\, C_2, \sigma, M) \preceq_{Q_1 * Q_2 * (Q_1' \wedge Q_2')} (\mathbb{C}_1 \,|||\, \mathbb{C}_2, \Sigma)$.

From $(\sigma, \Sigma) \models P_1 * P_2 * P$, we know there exist σ_1, σ_2, σ_r Σ_1, Σ_2 and Σ_r such that

$$(\sigma_1, \Sigma_1) \models P_1,\ \ (\sigma_2, \Sigma_2) \models P_2,\ \ (\sigma_r, \Sigma_r) \models P,\ \ \sigma = \sigma_1 \uplus \sigma_2 \uplus \sigma_r,\ \ \Sigma = \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r$$

From the premises, we know there exist M_1 and M_2 such that

$$\begin{array}{l} R \vee G_2, G_1, I \models (C_1, \sigma_1 \uplus \sigma_r, M_1) \preceq_{Q_1 * Q_1'} (\mathbb{C}_1, \Sigma_1 \uplus \Sigma_r) \\ R \vee G_1, G_2, I \models (C_2, \sigma_2 \uplus \sigma_r, M_2) \preceq_{Q_2 * Q_2'} (\mathbb{C}_2, \Sigma_2 \uplus \Sigma_r) \end{array}$$

By Lemma 16, we are done. □

Lemma 16. If

1. $R \vee G_2, G_1, I \models (C_1, \sigma_1 \uplus \sigma_r, M_1) \preceq_{Q_1 * Q_1'} (\mathbb{C}_1, \Sigma_1 \uplus \Sigma_r)$;
2. $R \vee G_1, G_2, I \models (C_2, \sigma_2 \uplus \sigma_r, M_2) \preceq_{Q_2 * Q_2'} (\mathbb{C}_2, \Sigma_2 \uplus \Sigma_r)$;
3. $(\sigma_r, \Sigma_r) \models I$; $Q_1' \vee Q_2' \Rightarrow I$; $I \triangleright \{R, G_1, G_2\}$; $\mathsf{Sta}(Q_1 * Q_1', (R \vee G_2) * \mathsf{Id})$; $\mathsf{Sta}(Q_2 * Q_2', (R \vee G_1) * \mathsf{Id})$;

then $R, G_1 \vee G_2, I \models (C_1 \,\|\, C_2, \sigma_1 \uplus \sigma_2 \uplus \sigma_r, M_1 + M_2) \preceq_{Q_1 * Q_2 * (Q_1' \wedge Q_2')} (\mathbb{C}_1 \,|||\, \mathbb{C}_2, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r)$.

Proof: By co-induction. We know $(\sigma_1 \uplus \sigma_2 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r) \models I * \mathbf{true}$.

1. for any σ_F, Σ_F, C' and σ'', if $(C_1 \,\|\, C_2, \sigma_1 \uplus \sigma_2 \uplus \sigma_r \uplus \sigma_F) \longrightarrow (C', \sigma'')$, then one of the following three cases holds:

(a) $C' = C'_1 \,\|\, C_2$ and $(C_1, \sigma_1 \uplus \sigma_2 \uplus \sigma_r \uplus \sigma_F) \longrightarrow (C'_1, \sigma'')$:
from the premise 1, we know: there exists σ' such that

$$\sigma'' = \sigma' \uplus \sigma_2 \uplus \sigma_F \tag{5.12}$$

and one of the following holds:

i. there exist M'_1, $\mathbb{C}'_1$ and Σ' such that

$$(\mathbb{C}_1, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}'_1, \Sigma' \uplus \Sigma_2 \uplus \Sigma_F) \tag{5.13}$$

$$((\sigma_1 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_r), (\sigma', \Sigma'), \mathbf{true}) \models {G_1}^+ * \mathsf{True} \tag{5.14}$$

$$R \vee G_2, G_1, I \models (C'_1, \sigma', M'_1) \preceq_{Q_1 * Q'_1} (\mathbb{C}'_1, \Sigma') \tag{5.15}$$

Below we prove 1(a) of Definition 2 holds.
From $I \triangleright G_1$, $(\sigma_r, \Sigma_r) \models I$ and (5.14), we know: there exist σ'_1, Σ'_1, σ'_r and Σ'_r such that

$$\sigma' = \sigma'_1 \uplus \sigma'_r\,, \quad \Sigma' = \Sigma'_1 \uplus \Sigma'_r\,, \quad (\sigma'_r, \Sigma'_r) \models I \tag{5.16}$$

$$((\sigma_r, \Sigma_r), (\sigma'_r, \Sigma'_r), \mathbf{true}) \models {G_1}^+ \tag{5.17}$$

From (5.12) and (5.16), we know

$$\sigma'' = \sigma'_1 \uplus \sigma_2 \uplus \sigma'_r \uplus \sigma_F \tag{5.18}$$

From (5.13) and (5.16), we know

$$(\mathbb{C}_1 \,|||\, \mathbb{C}_2, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}'_1 \,|||\, \mathbb{C}_2, \Sigma'_1 \uplus \Sigma_2 \uplus \Sigma'_r \uplus \Sigma_F) \tag{5.19}$$

From (5.17), we know:

$$((\sigma_1 \uplus \sigma_2 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r), (\sigma'_1 \uplus \sigma_2 \uplus \sigma'_r, \Sigma'_1 \uplus \Sigma_2 \uplus \Sigma'_r), \mathbf{true}) \models (G_1 \vee G_2)^+ * \mathsf{True} \tag{5.20}$$

and $((\sigma_2 \uplus \sigma_r, \Sigma_2 \uplus \Sigma_r), (\sigma_2 \uplus \sigma'_r, \Sigma_2 \uplus \Sigma'_r), \mathbf{true}) \models (G_1 \vee R)^+ * \mathsf{Id}$.
Then from the premise 2, we know: there exists M'_2 such that

$$R \vee G_1, G_2, I \models (C_2, \sigma_2 \uplus \sigma'_r, M'_2) \preceq_{Q_2 * Q'_2} (\mathbb{C}_2, \Sigma_2 \uplus \Sigma'_r) \tag{5.21}$$

From (5.15), (5.16), (5.21) and the co-induction hypothesis, we know:

$$R, G_1 \vee G_2, I \models (C'_1 \,\|\, C_2, \sigma'_1 \uplus \sigma_2 \uplus \sigma'_r, M'_1 + M'_2) \preceq_{Q_1 * Q_2 * (Q'_1 \wedge Q'_2)} (\mathbb{C}'_1 \,|||\, \mathbb{C}_2, \Sigma'_1 \uplus \Sigma_2 \uplus \Sigma'_r) \tag{5.22}$$

From (5.18), (5.19), (5.20) and (5.22), we are done.

ii. there exists M'_1 such that

$$M'_1 < M_1 \tag{5.23}$$

$$((\sigma_1 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_r), (\sigma', \Sigma_1 \uplus \Sigma_r), \mathbf{false}) \models {G_1}^+ * \mathsf{True} \tag{5.24}$$

$$R \vee G_2, G_1, I \models (C'_1, \sigma', M'_1) \preceq_{Q_1 * Q'_1} (\mathbb{C}_1, \Sigma_1 \uplus \Sigma_r) \tag{5.25}$$

Below we prove 1(b) of Definition 2 holds.
From $I \triangleright G_1$, $(\sigma_r, \Sigma_r) \models I$ and (5.24), we know: there exist σ'_1 and σ'_r such that

$$\sigma' = \sigma'_1 \uplus \sigma'_r\,, \quad (\sigma'_r, \Sigma_r) \models I \tag{5.26}$$

$$((\sigma_r, \Sigma_r), (\sigma'_r, \Sigma_r), \mathbf{false}) \models {G_1}^+ \tag{5.27}$$

From (5.12) and (5.26), we know

$$\sigma'' = \sigma'_1 \uplus \sigma_2 \uplus \sigma'_r \uplus \sigma_F \tag{5.28}$$

From (5.27), we know:

$$((\sigma_1 \uplus \sigma_2 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r), (\sigma_1' \uplus \sigma_2 \uplus \sigma_r', \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r), \mathbf{false}) \models (G_1 \vee G_2)^+ * \mathsf{True} \quad (5.29)$$

and $((\sigma_2 \uplus \sigma_r, \Sigma_2 \uplus \Sigma_r), (\sigma_2 \uplus \sigma_r', \Sigma_2 \uplus \Sigma_r), \mathbf{false}) \models (G_1 \vee R)^+ * \mathsf{Id}$.
Then from the premise 2, we know:

$$R \vee G_1, G_2, I \models (C_2, \sigma_2 \uplus \sigma_r', M_2) \preceq_{Q_2 * Q_2'} (\mathbb{C}_2, \Sigma_2 \uplus \Sigma_r) \quad (5.30)$$

From (5.25), (5.26), (5.30) and the co-induction hypothesis, we know:

$$R, G_1 \vee G_2, I \models (C_1' \,\|\, C_2, \sigma_1' \uplus \sigma_2 \uplus \sigma_r', M_1' + M_2) \preceq_{Q_1 * Q_2 * (Q_1' \wedge Q_2')} (\mathbb{C}_1 \,|\!|\!|\, \mathbb{C}_2, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r) \quad (5.31)$$

From (5.23), we get:

$$M_1' + M_2 < M_1 + M_2 \quad (5.32)$$

From (5.28), (5.29), (5.31) and (5.32), we are done.

(b) $C' = C_1 \,\|\, C_2'$ and $(C_2, \sigma_1 \uplus \sigma_2 \uplus \sigma_r \uplus \sigma_F) \longrightarrow (C_2', \sigma'')$: similar to the first case.

(c) $C' = \mathbf{skip}$, $C_1 = \mathbf{skip}$ and $C_2 = \mathbf{skip}$, thus we know

$$\sigma'' = \sigma_1 \uplus \sigma_2 \uplus \sigma_r \uplus \sigma_F \quad (5.33)$$

Below we prove 1(a) of Definition 2 holds.
From the premise 1, we know one of the following holds:

i. there exists Σ' such that

$$(\mathbb{C}_1, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r \uplus \Sigma_F) \longrightarrow^+ (\boldsymbol{skip}, \Sigma' \uplus \Sigma_2 \uplus \Sigma_F) \quad (5.34)$$

$$((\sigma_1 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_r), (\sigma_1 \uplus \sigma_r, \Sigma'), \mathbf{true}) \models {G_1}^+ * \mathsf{True} \quad (5.35)$$

$$(\sigma_1 \uplus \sigma_r, \Sigma') \models Q_1 * Q_1' \quad (5.36)$$

From $I \triangleright G_1$, $(\sigma_r, \Sigma_r) \models I$ and (5.35), we know: there exist Σ_1' and Σ_r' such that

$$\Sigma' = \Sigma_1' \uplus \Sigma_r' \,, \quad (\sigma_r, \Sigma_r') \models I \quad (5.37)$$

$$((\sigma_r, \Sigma_r), (\sigma_r, \Sigma_r'), \mathbf{true}) \models {G_1}^+ \quad (5.38)$$

Since $Q_1' \Rightarrow I$ and (5.36), we get:

$$(\sigma_1, \Sigma_1') \models Q_1 \,, \quad (\sigma_r, \Sigma_r') \models Q_1' \quad (5.39)$$

From (5.34) and (5.37), we know

$$(\mathbb{C}_1 \,|\!|\!|\, \mathbb{C}_2, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r \uplus \Sigma_F) \longrightarrow^+ (\boldsymbol{skip} \,|\!|\!|\, \mathbb{C}_2, \Sigma_1' \uplus \Sigma_2 \uplus \Sigma_r' \uplus \Sigma_F) \quad (5.40)$$

From (5.38), we know: $((\sigma_2 \uplus \sigma_r, \Sigma_2 \uplus \Sigma_r), (\sigma_2 \uplus \sigma_r, \Sigma_2 \uplus \Sigma_r'), \mathbf{true}) \models (G_1 \vee R)^+ * \mathsf{Id}$.
Then from the premise 2, we know: there exists M_2' such that

$$R \vee G_1, G_2, I \models (C_2, \sigma_2 \uplus \sigma_r, M_2') \preceq_{Q_2 * Q_2'} (\mathbb{C}_2, \Sigma_2 \uplus \Sigma_r') \quad (5.41)$$

Since $C_2 = \mathbf{skip}$, we know one of the following holds:

A. there exists Σ'' such that

$$(\mathbb{C}_2, \Sigma_1' \uplus \Sigma_2 \uplus \Sigma_r' \uplus \Sigma_F) \longrightarrow^+ (\textit{\textbf{skip}}, \Sigma'' \uplus \Sigma_1' \uplus \Sigma_F) \tag{5.42}$$

$$((\sigma_2 \uplus \sigma_r, \Sigma_2 \uplus \Sigma_r'), (\sigma_2 \uplus \sigma_r, \Sigma''), \mathbf{true}) \models {G_2}^+ * \mathsf{True} \tag{5.43}$$

$$(\sigma_2 \uplus \sigma_r, \Sigma'') \models Q_2 * Q_2' \tag{5.44}$$

From $I \triangleright G_2$, $(\sigma_r, \Sigma_r') \models I$ and (5.43), we know: there exist Σ_2' and Σ_r'' such that

$$\Sigma'' = \Sigma_2' \uplus \Sigma_r'', \quad (\sigma_r, \Sigma_r'') \models I \tag{5.45}$$

$$((\sigma_r, \Sigma_r'), (\sigma_r, \Sigma_r''), \mathbf{true}) \models {G_2}^+ \tag{5.46}$$

Since $Q_2' \Rightarrow I$ and (5.44), we get:

$$(\sigma_2, \Sigma_2') \models Q_2, \quad (\sigma_r, \Sigma_r'') \models Q_2' \tag{5.47}$$

From (5.40) and (5.42), we know

$$(\mathbb{C}_1 \,|||\, \mathbb{C}_2, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r \uplus \Sigma_F) \longrightarrow^+ (\textit{\textbf{skip}}, \Sigma_1' \uplus \Sigma_2' \uplus \Sigma_r'' \uplus \Sigma_F) \tag{5.48}$$

From (5.38) and (5.46), we know:

$$((\sigma_r, \Sigma_r), (\sigma_r, \Sigma_r''), \mathbf{true}) \models (G_1 \vee G_2)^+ \tag{5.49}$$

Thus we get:

$$((\sigma_1 \uplus \sigma_2 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r), (\sigma_1 \uplus \sigma_2 \uplus \sigma_r, \Sigma_1' \uplus \Sigma_2' \uplus \Sigma_r''), \mathbf{true}) \models (G_1 \vee G_2)^+ * \mathsf{True} \tag{5.50}$$

From (5.46), we get: $((\sigma_r, \Sigma_r'), (\sigma_r, \Sigma_r''), \mathbf{true}) \models (R \vee G_2)^+$. Since $(\sigma_1, \Sigma_1') \models Q_1$, $(\sigma_r, \Sigma_r') \models Q_1'$, $\mathsf{Sta}(Q_1 * Q_1', (R \vee G_2) * \mathsf{Id})$, $I \triangleright (R \vee G_2)$ and $Q_1' \Rightarrow I$, we know:

$$(\sigma_r, \Sigma_r'') \models Q_1' \tag{5.51}$$

From $(\sigma_1, \Sigma_1') \models Q_1$ and (5.47), we get:

$$(\sigma_1 \uplus \sigma_2 \uplus \sigma_r, \Sigma_1' \uplus \Sigma_2' \uplus \Sigma_r'') \models Q_1 * Q_2 * (Q_1' \wedge Q_2') \tag{5.52}$$

By the B-SKIP and B-FRAME rules, we get: there exists M' such that

$$R, G_1 \vee G_2, I \models (\mathbf{skip}, \sigma_1 \uplus \sigma_2 \uplus \sigma_r, M') \preceq_{Q_1 * Q_2 * (Q_1' \wedge Q_2')} (\textit{\textbf{skip}}, \Sigma_1' \uplus \Sigma_2' \uplus \Sigma_r'') \tag{5.53}$$

From (5.48), (5.50) and (5.53), we are done.

B. $\mathbb{C}_2 = \textit{\textbf{skip}}$ and $(\sigma_2 \uplus \sigma_r, \Sigma_2 \uplus \Sigma_r') \models Q_2 * Q_2'$.
From $Q_2' \Rightarrow I$ and $(\sigma_r, \Sigma_r') \models I$, we know:

$$(\sigma_2, \Sigma_2) \models Q_2, \quad (\sigma_r, \Sigma_r') \models Q_2' \tag{5.54}$$

From (5.40), we know

$$(\mathbb{C}_1 \,|||\, \mathbb{C}_2, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r \uplus \Sigma_F) \longrightarrow^+ (\textit{\textbf{skip}}, \Sigma_1' \uplus \Sigma_2 \uplus \Sigma_r' \uplus \Sigma_F) \tag{5.55}$$

From (5.38), we know:

$$((\sigma_r, \Sigma_r), (\sigma_r, \Sigma_r'), \mathbf{true}) \models (G_1 \vee G_2)^+ \tag{5.56}$$

Thus we get:

$$((\sigma_1 \uplus \sigma_2 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r), (\sigma_1 \uplus \sigma_2 \uplus \sigma_r, \Sigma_1' \uplus \Sigma_2 \uplus \Sigma_r'), \mathbf{true}) \models (G_1 \vee G_2)^+ * \mathsf{True} \tag{5.57}$$

From (5.39) and (5.54), we get:

$$(\sigma_1 \uplus \sigma_2 \uplus \sigma_r, \Sigma_1' \uplus \Sigma_2 \uplus \Sigma_r') \models Q_1 * Q_2 * (Q_1' \wedge Q_2') \tag{5.58}$$

By the B-SKIP and B-FRAME rules, we get: there exists M' such that

$$R, G_1 \vee G_2, I \models (\mathbf{skip}, \sigma_1 \uplus \sigma_2 \uplus \sigma_r, M') \preceq_{Q_1 * Q_2 * (Q_1' \wedge Q_2')} (\boldsymbol{skip}, \Sigma_1' \uplus \Sigma_2 \uplus \Sigma_r') \tag{5.59}$$

From (5.55), (5.57) and (5.59), we are done.

ii. $\mathbb{C}_1 = \boldsymbol{skip}$ and $(\sigma_1 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_r) \models Q_1 * Q_1'$.
From $Q_1' \Rightarrow I$ and $(\sigma_r, \Sigma_r) \models I$, we know:

$$(\sigma_1, \Sigma_1) \models Q_1\,, \quad (\sigma_r, \Sigma_r) \models Q_1' \tag{5.60}$$

From the premise 2, we know one of the following holds:

A. there exists Σ' such that

$$(\mathbb{C}_2, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r \uplus \Sigma_F) \longrightarrow^+ (\boldsymbol{skip}, \Sigma' \uplus \Sigma_1 \uplus \Sigma_F) \tag{5.61}$$

$$((\sigma_2 \uplus \sigma_r, \Sigma_2 \uplus \Sigma_r), (\sigma_2 \uplus \sigma_r, \Sigma'), \mathbf{true}) \models {G_2}^+ * \mathsf{True} \tag{5.62}$$

$$(\sigma_2 \uplus \sigma_r, \Sigma') \models Q_2 * Q_2' \tag{5.63}$$

From $I \triangleright G_2$, $(\sigma_r, \Sigma_r) \models I$ and (5.62), we know: there exist Σ_2' and Σ_r' such that

$$\Sigma' = \Sigma_2' \uplus \Sigma_r'\,, \quad (\sigma_r, \Sigma_r') \models I \tag{5.64}$$

$$((\sigma_r, \Sigma_r), (\sigma_r, \Sigma_r'), \mathbf{true}) \models {G_2}^+ \tag{5.65}$$

Since $Q_2' \Rightarrow I$ and (5.63), we get:

$$(\sigma_2, \Sigma_2') \models Q_2\,, \quad (\sigma_r, \Sigma_r') \models Q_2' \tag{5.66}$$

From (5.61), we know

$$(\mathbb{C}_1 \,|||\, \mathbb{C}_2, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r \uplus \Sigma_F) \longrightarrow^+ (\boldsymbol{skip}, \Sigma_1 \uplus \Sigma_2' \uplus \Sigma_r' \uplus \Sigma_F) \tag{5.67}$$

From (5.65), we know:

$$((\sigma_r, \Sigma_r), (\sigma_r, \Sigma_r'), \mathbf{true}) \models (G_1 \vee G_2)^+ \tag{5.68}$$

Thus we get:

$$((\sigma_1 \uplus \sigma_2 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r), (\sigma_1 \uplus \sigma_2 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_2' \uplus \Sigma_r'), \mathbf{true}) \models (G_1 \vee G_2)^+ * \mathsf{True} \tag{5.69}$$

From (5.65), we get: $((\sigma_r, \Sigma_r), (\sigma_r, \Sigma_r'), \mathbf{true}) \models (R \vee G_2)^+$. Since $(\sigma_1, \Sigma_1) \models Q_1$, $(\sigma_r, \Sigma_r) \models Q_1'$, $\mathsf{Sta}(Q_1 * Q_1', (R \vee G_2) * \mathsf{Id})$, $I \triangleright (R \vee G_2)$ and $Q_1' \Rightarrow I$, we know:

$$(\sigma_r, \Sigma_r') \models Q_1' \tag{5.70}$$

From $(\sigma_1, \Sigma_1) \models Q_1$ and (5.66), we get:

$$(\sigma_1 \uplus \sigma_2 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_2' \uplus \Sigma_r') \models Q_1 * Q_2 * (Q_1' \wedge Q_2') \tag{5.71}$$

By the B-SKIP and B-FRAME rules, we get: there exists M' such that

$$R, G_1 \vee G_2, I \models (\mathbf{skip}, \sigma_1 \uplus \sigma_2 \uplus \sigma_r, M') \preceq_{Q_1 * Q_2 * (Q_1' \wedge Q_2')} (\boldsymbol{skip}, \Sigma_1 \uplus \Sigma_2' \uplus \Sigma_r') \tag{5.72}$$

From (5.67), (5.69) and (5.72), we are done.

B. $\mathbb{C}_2 = \boldsymbol{skip}$ and $(\sigma_2 \uplus \sigma_r, \Sigma_2 \uplus \Sigma_r) \models Q_2 * Q_2'$.
From $Q_2' \Rightarrow I$ and $(\sigma_r, \Sigma_r) \models I$, we know:

$$(\sigma_2, \Sigma_2) \models Q_2\,, \quad (\sigma_r, \Sigma_r) \models Q_2' \tag{5.73}$$

We know

$$(\mathbb{C}_1 \,|||\, \mathbb{C}_2, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r \uplus \Sigma_F) \longrightarrow^{+} (\boldsymbol{skip}, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r \uplus \Sigma_F) \tag{5.74}$$

Also we have:

$$((\sigma_1 \uplus \sigma_2 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r), (\sigma_1 \uplus \sigma_2 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r), \mathbf{true}) \models (G_1 \vee G_2)^{+} * \mathsf{True} \tag{5.75}$$

From (5.60) and (5.73), we get:

$$(\sigma_1 \uplus \sigma_2 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r) \models Q_1 * Q_2 * (Q_1' \wedge Q_2') \tag{5.76}$$

By the B-SKIP and B-FRAME rules, we get: there exists M' such that

$$R, G_1 \vee G_2, I \models (\mathbf{skip}, \sigma_1 \uplus \sigma_2 \uplus \sigma_r, M') \preceq_{Q_1 * Q_2 * (Q_1' \wedge Q_2')} (\boldsymbol{skip}, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r) \tag{5.77}$$

From (5.74), (5.75) and (5.77), we are done.

2. for any σ_F, Σ_F, e, C' and σ'', if $(C_1 \,\|\, C_2, \sigma_1 \uplus \sigma_2 \uplus \sigma_r \uplus \sigma_F) \xrightarrow{e} (C', \sigma'')$, the proof is similar to the first case.

3. for any σ' and Σ', if $((\sigma_1 \uplus \sigma_2 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r), (\sigma', \Sigma'), \mathbf{true}) \models R^{+} * \mathsf{Id}$,
from $I \triangleright R$ and $(\sigma_r, \Sigma_r) \models I$, we know: there exist σ_r' and Σ_r' such that

$$\sigma' = \sigma_1 \uplus \sigma_2 \uplus \sigma_r'\,, \quad \Sigma' = \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r'\,, \quad (\sigma_r', \Sigma_r') \models I \tag{5.78}$$

$$((\sigma_r, \Sigma_r), (\sigma_r', \Sigma_r'), \mathbf{true}) \models R^{+} \tag{5.79}$$

Thus we get:

$$((\sigma_1 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_r), (\sigma_1 \uplus \sigma_r', \Sigma_1 \uplus \Sigma_r'), \mathbf{true}) \models (R \vee G_2)^{+} * \mathsf{Id} \tag{5.80}$$

$$((\sigma_2 \uplus \sigma_r, \Sigma_2 \uplus \Sigma_r), (\sigma_2 \uplus \sigma_r', \Sigma_2 \uplus \Sigma_r'), \mathbf{true}) \models (R \vee G_1)^{+} * \mathsf{Id} \tag{5.81}$$

From the premises, we know: there exist M_1' and M_2' such that

$$R \vee G_2, G_1, I \models (C_1, \sigma_1 \uplus \sigma_r', M_1') \preceq_{Q_1 * Q_1'} (\mathbb{C}_1, \Sigma_1 \uplus \Sigma_r') \tag{5.82}$$

$$R \vee G_1, G_2, I \models (C_2, \sigma_2 \uplus \sigma_r', M_2') \preceq_{Q_2 * Q_2'} (\mathbb{C}_2, \Sigma_2 \uplus \Sigma_r') \tag{5.83}$$

By the co-induction hypothesis, we get:

$$R, G_1 \vee G_2, I \models (C_1 \,\|\, C_2, \sigma_1 \uplus \sigma_2 \uplus \sigma_r', M_1' + M_2') \preceq_{Q_1 * Q_2 * (Q_1' \wedge Q_2')} (\mathbb{C}_1 \,|||\, \mathbb{C}_2, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r') \tag{5.84}$$

4. for any σ' and Σ', if $((\sigma_1 \uplus \sigma_2 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r), (\sigma', \Sigma'), \mathbf{false}) \models R^{+} * \mathsf{Id}$,
from $I \triangleright R$ and $(\sigma_r, \Sigma_r) \models I$, we know: there exist σ_r' and Σ_r' such that

$$\sigma' = \sigma_1 \uplus \sigma_2 \uplus \sigma_r'\,, \quad \Sigma' = \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r'\,, \quad (\sigma_r', \Sigma_r') \models I \tag{5.85}$$

$$((\sigma_r, \Sigma_r), (\sigma_r', \Sigma_r'), \mathbf{false}) \models R^{+} \tag{5.86}$$

Thus we get:

$$((\sigma_1 \uplus \sigma_r, \Sigma_1 \uplus \Sigma_r), (\sigma_1 \uplus \sigma_r', \Sigma_1 \uplus \Sigma_r'), \mathbf{false}) \models (R \vee G_2)^{+} * \mathsf{Id} \tag{5.87}$$

$$((\sigma_2 \uplus \sigma_r, \Sigma_2 \uplus \Sigma_r), (\sigma_2 \uplus \sigma_r', \Sigma_2 \uplus \Sigma_r'), \mathbf{false}) \models (R \vee G_1)^{+} * \mathsf{Id} \tag{5.88}$$

From the premises, we know:

$$R \vee G_2, G_1, I \models (C_1, \sigma_1 \uplus \sigma'_r, M_1) \preceq_{Q_1 * Q'_1} (\mathbb{C}_1, \Sigma_1 \uplus \Sigma'_r) \tag{5.89}$$

$$R \vee G_1, G_2, I \models (C_2, \sigma_2 \uplus \sigma'_r, M_2) \preceq_{Q_2 * Q'_2} (\mathbb{C}_2, \Sigma_2 \uplus \Sigma'_r) \tag{5.90}$$

By the co-induction hypothesis, we get:

$$R, G_1 \vee G_2, I \models (C_1 \,\|\, C_2, \sigma_1 \uplus \sigma_2 \uplus \sigma'_r, M_1 + M_2) \preceq_{Q_1 * Q_2 * (Q'_1 \wedge Q'_2)} (\mathbb{C}_1 \,|||\, \mathbb{C}_2, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma'_r) \tag{5.91}$$

5. for any σ_F and Σ_F, if $(C_1 \,\|\, C_2, \sigma_1 \uplus \sigma_2 \uplus \sigma_r \uplus \sigma_F) \longrightarrow$ **abort**, by the operational semantics and the premises, we know $(\mathbb{C}_1 \,|||\, \mathbb{C}_2, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_r \uplus \Sigma_F) \longrightarrow^+$ **abort**.

Thus we are done. $\square$

The U2B rule.

Lemma 17 (U2B). If $R, G, I \models \{P \wedge \mathsf{arem}(\mathbb{C})\}C\{Q \wedge \mathsf{arem}(\boldsymbol{skip})\}$, then $R, G, I \models \{P\}C \preceq \mathbb{C}\{Q\}$.

Proof: We need to prove: for all σ and Σ, if $(\sigma, \Sigma) \models P$, then there exists M such that $R, G, I \models (C, \sigma, M) \preceq_Q (\mathbb{C}, \Sigma)$.

From $(\sigma, \Sigma) \models P$, we know: $(\sigma, 0, \mathbb{C}, \Sigma) \models P \wedge \mathsf{arem}(\mathbb{C})$.

From the premise, we know: $R, G, I \models (C, \sigma, (0, |C|)) \preceq_{\mathsf{height}(C);0;Q \wedge \mathsf{arem}(\boldsymbol{skip})} (\mathbb{C}, \Sigma)$.

By Lemma 18, we are done. $\square$

Lemma 18. If $R, G, I \models (C, \sigma, ws) \preceq_{\mathcal{H};w;Q \wedge \mathsf{arem}(\boldsymbol{skip})} (\mathbb{C}, \Sigma)$, then $R, G, I \models (C, \sigma, (ws, \mathcal{H})) \preceq_Q (\mathbb{C}, \Sigma)$.

Proof: By co-induction. From the premise, we know $(\sigma, \Sigma) \models I * \mathbf{true}$.

1. for any σ_F, Σ_F, C' and σ'', if $(C, \sigma \uplus \sigma_F) \longrightarrow (C', \sigma'')$ and $\Sigma \bot \Sigma_F$,

   from the premise, we know: there exists σ' such that $\sigma'' = \sigma' \uplus \sigma_F$ and one of the following holds:

   (a) there exist ws', w', $\mathbb{C}'$ and Σ' such that $(\mathbb{C}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', ws') \preceq_{\mathcal{H};w';Q \wedge \mathsf{arem}(\boldsymbol{skip})} (\mathbb{C}', \Sigma')$.
   By the co-induction hypothesis, we know: $R, G, I \models (C', \sigma', (ws', \mathcal{H})) \preceq_Q (\mathbb{C}', \Sigma')$.

   (b) there exists ws' such that $ws' <_{\mathcal{H}} ws$, $((\sigma, \Sigma), (\sigma', \Sigma), \mathbf{false}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', ws') \preceq_{\mathcal{H};w;Q \wedge \mathsf{arem}(\boldsymbol{skip})} (\mathbb{C}, \Sigma)$.
   By the co-induction hypothesis, we know: $R, G, I \models (C', \sigma', (ws', \mathcal{H})) \preceq_Q (\mathbb{C}, \Sigma)$.
   By the instantiation of the abstract metric, we know: $(ws', \mathcal{H}) < (ws, \mathcal{H})$.

2. for any σ_F, Σ_F, e, C' and σ'', if $(C, \sigma \uplus \sigma_F) \xrightarrow{e} (C', \sigma'')$, the proof is similar to the previous case.

3. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models R^+ * \mathsf{Id}$,

   from the premise, we know: there exist ws' and w' such that $R, G, I \models (C, \sigma', ws') \preceq_{\mathcal{H};w';Q \wedge \mathsf{arem}(\boldsymbol{skip})} (\mathbb{C}, \Sigma')$.

   By the co-induction hypothesis, we know: $R, G, I \models (C, \sigma', (ws', \mathcal{H})) \preceq_Q (\mathbb{C}, \Sigma')$.

4. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{false}) \models R^+ * \mathsf{Id}$,

   from the premise, we know: $R, G, I \models (C, \sigma', ws) \preceq_{\mathcal{H};w;Q \wedge \mathsf{arem}(\boldsymbol{skip})} (\mathbb{C}, \Sigma')$.

   By the co-induction hypothesis, we know: $R, G, I \models (C, \sigma', (ws, \mathcal{H})) \preceq_Q (\mathbb{C}, \Sigma')$.

5. if $C = \mathbf{skip}$, then for any Σ_F, from the premise, we know one of the following holds:

   (a) there exist w', $\mathbb{C}'$ and Σ' such that $(\mathbb{C}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma, \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma, w', \mathbb{C}', \Sigma') \models Q \wedge \mathsf{arem}(\boldsymbol{skip})$.
   Thus we know $\mathbb{C}' = \boldsymbol{skip}$ and $(\sigma, \Sigma') \models Q$.

   (b) there exists w' such that $ws = (w', 0)$ and $(\sigma, w + w', \mathbb{C}, \Sigma) \models Q \wedge \mathsf{arem}(\boldsymbol{skip})$.
   Thus we know $\mathbb{C} = \boldsymbol{skip}$ and $(\sigma, \Sigma) \models Q$.

6. for any σ_F and Σ_F, if $(C, \sigma \uplus \sigma_F) \longrightarrow \mathbf{abort}$, from the premise, we know $(\mathbb{C}, \Sigma \uplus \Sigma_F) \longrightarrow^+ \mathbf{abort}$.

Thus we are done. $\square$

The TRANS rule. We define $M_2 \circ M_1$ as a pair (M_2, M_1) and the corresponding well-founded order as the lexical order. That is, the following hold:

$$(M_2 < M_2') \implies (M_2 \circ M_1 < M_2' \circ M_1') \tag{5.92}$$

$$(M_1 < M_1') \implies (M_2 \circ M_1 < M_2 \circ M_1') \tag{5.93}$$

Lemma 19 (TRANS). If

1. $R_1, G_1, I_1 \vdash \{P_1\} C \preceq \mathrm{C_M} \{Q_1\}$;
2. $R_2, G_2, I_2 \vdash \{P_2\} \mathrm{C_M} \preceq \mathbb{C} \{Q_2\}$;
3. $\mathsf{MPrecise}(I_1, I_2)$; $I_1 \triangleright \{R_1, G_1\}$; $I_2 \triangleright \{R_2, G_2\}$;
4. $((G_1)^{I_1} \mathbin{\hat{;}} (G_2)^{I_2}) \Rightarrow (G_1 \mathbin{\hat{;}} G_2)^{I_1 ; I_2}$; $(R_1 \mathbin{\check{;}} R_2)^{I_1 ; I_2} \Rightarrow ((R_1)^{I_1} \mathbin{\check{;}} (R_2)^{I_2})$;

then $(R_1 \mathbin{\check{;}} R_2), (G_1 \mathbin{\hat{;}} G_2), (I_1 ; I_2) \vdash \{P_1 ; P_2\} C \preceq \mathbb{C} \{Q_1 ; Q_2\}$.

Proof: For all σ and Σ, if $(\sigma, \Sigma) \models P_1 ; P_2$, we know there exists θ such that $(\sigma, \theta) \models P_1$ and $(\theta, \Sigma) \models P_2$.
From the premise, we know:

1. there exists M_1 such that $R_1, G_1, I_1 \models (C, \sigma, M_1) \preceq_{Q_1} (\mathrm{C_M}, \theta)$.
2. there exists M_2 such that $R_2, G_2, I_2 \models (\mathrm{C_M}, \theta, M_2) \preceq_{Q_2} (\mathbb{C}, \Sigma)$.

By Lemma 20, we know $(R_1 \mathbin{\check{;}} R_2), (G_1 \mathbin{\hat{;}} G_2), (I_1 ; I_2) \models (C, \sigma, (M_2 \circ M_1)) \preceq_{Q_1 ; Q_2} (\mathbb{C}, \Sigma)$. Thus we are done. □

Lemma 20. If

1. $R_1, G_1, I_1 \models (C, \sigma, M_1) \preceq_{Q_1} (\mathrm{C_M}, \theta)$;
2. $R_2, G_2, I_2 \models (\mathrm{C_M}, \theta, M_2) \preceq_{Q_2} (\mathbb{C}, \Sigma)$;
3. $\mathsf{MPrecise}(I_1, I_2)$; $I_1 \triangleright \{R_1, G_1\}$; $I_2 \triangleright \{R_2, G_2\}$;
4. $((G_1)^+ \mathbin{\hat{;}} (G_2)^+) \Rightarrow (G_1 \mathbin{\hat{;}} G_2)^+$; $(R_1 \mathbin{\check{;}} R_2)^+ \Rightarrow ((R_1)^+ \mathbin{\check{;}} (R_2)^+)$;

then $(R_1 \mathbin{\check{;}} R_2), (G_1 \mathbin{\hat{;}} G_2), (I_1 ; I_2) \models (C, \sigma, (M_2 \circ M_1)) \preceq_{Q_1 ; Q_2} (\mathbb{C}, \Sigma)$.

Proof: By co-induction. By the premises, we know $(\sigma, \theta) \models I_1 * \mathbf{true}$ and $(\theta, \Sigma) \models I_2 * \mathbf{true}$. Since $\mathsf{MPrecise}(I_1, I_2)$, we know $(\sigma, \Sigma) \models (I_1 ; I_2) * \mathbf{true}$.

1. for any σ_F, Σ_F, C' and σ'', if $(C, \sigma \uplus \sigma_F) \longrightarrow (C', \sigma'')$, then by the premise 1, we know:
   there exists σ' such that $\sigma'' = \sigma' \uplus \sigma_F$ and for any θ_F, one of the following holds:
   (a) either, there exist M_1', $\mathrm{C_M'}$ and θ' such that $(\mathrm{C_M}, \theta \uplus \theta_F) \longrightarrow^+ (\mathrm{C_M'}, \theta' \uplus \theta_F)$, $((\sigma, \theta), (\sigma', \theta'), \mathbf{true}) \models (G_1)^+ * \mathsf{True}$ and $R_1, G_1, I_1 \models (C', \sigma', M_1') \preceq_{Q_1} (\mathrm{C_M'}, \theta')$.
   By the premise 2 and Lemma 21, we know: one of the following holds:
      i. either, there exist M_2', $\mathbb{C}'$ and Σ' such that $(\mathbb{C}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\theta, \Sigma), (\theta', \Sigma'), \mathbf{true}) \models (G_2)^+ * \mathsf{True}$ and $R_2, G_2, I_2 \models (\mathrm{C_M'}, \theta', M_2') \preceq_{Q_2} (\mathbb{C}', \Sigma')$.
      Thus we know

$$((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models ((G_1)^+ * \mathsf{True}) \mathbin{\hat{;}} ((G_2)^+ * \mathsf{True}) \tag{5.94}$$

      Since $I_1 \triangleright G_1$ and $I_2 \triangleright G_2$, we know $I_1 \triangleright (G_1)^+$ and $I_2 \triangleright (G_2)^+$. Since $\mathsf{MPrecise}(I_1, I_2)$, by Lemma 25, we know

$$((G_1)^+ * \mathsf{True}) \mathbin{\hat{;}} ((G_2)^+ * \mathsf{True}) \Rightarrow ((G_1)^+ \mathbin{\hat{;}} (G_2)^+) * \mathsf{True} \tag{5.95}$$

Thus we get:

$$((\sigma,\Sigma),(\sigma',\Sigma'),\mathbf{true}) \models (G_1 \mathbin{\hat{;}} G_2)^+ * \mathsf{True} \tag{5.96}$$

Besides, by the co-induction hypothesis, we get:

$$(R_1 \mathbin{\check{;}} R_2),(G_1 \mathbin{\hat{;}} G_2),(I_1 \mathbin{;} I_2) \models (C',\sigma',(M_2' \circ M_1')) \preceq_{Q_1 ; Q_2} (\mathbb{C}',\Sigma') \tag{5.97}$$

ii. or, there exists M_2' such that $M_2' < M_2$,
$((\theta,\Sigma),(\theta',\Sigma),\mathbf{false}) \models (G_2)^+ * \mathsf{True}$ and $R_2, G_2, I_2 \models (\mathrm{C}_\mathrm{M}',\theta',M_2') \preceq_{Q_2} (\mathbb{C},\Sigma)$.
Thus we know

$$((\sigma,\Sigma),(\sigma',\Sigma),\mathbf{false}) \models ((G_1)^+ * \mathsf{True}) \mathbin{\hat{;}} ((G_2)^+ * \mathsf{True}) \tag{5.98}$$

Thus we get:

$$((\sigma,\Sigma),(\sigma',\Sigma),\mathbf{false}) \models (G_1 \mathbin{\hat{;}} G_2)^+ * \mathsf{True} \tag{5.99}$$

Besides, by the co-induction hypothesis, we get:

$$(R_1 \mathbin{\check{;}} R_2),(G_1 \mathbin{\hat{;}} G_2),(I_1 \mathbin{;} I_2) \models (C',\sigma',(M_2' \circ M_1')) \preceq_{Q_1 ; Q_2} (\mathbb{C},\Sigma) \tag{5.100}$$

Moreover, we know

$$(M_2' \circ M_1') < (M_2 \circ M_1) \tag{5.101}$$

(b) or, there exists M_1' such that $M_1' < M_1$,
$((\sigma,\theta),(\sigma',\theta),\mathbf{false}) \models (G_1)^+ * \mathsf{True}$ and $R_1, G_1, I_1 \models (C',\sigma',M_1') \preceq_{Q_1} (\mathrm{C}_\mathrm{M},\theta)$.
Since $(\theta,\Sigma) \models I_2 * \mathbf{true}$, we know $((\theta,\Sigma),(\theta,\Sigma),\mathbf{false}) \models (G_2)^+ * \mathsf{True}$. Thus

$$((\sigma,\Sigma),(\sigma',\Sigma),\mathbf{false}) \models ((G_1)^+ * \mathsf{True}) \mathbin{\hat{;}} ((G_2)^+ * \mathsf{True}) \tag{5.102}$$

Thus we get:

$$((\sigma,\Sigma),(\sigma',\Sigma),\mathbf{false}) \models (G_1 \mathbin{\hat{;}} G_2)^+ * \mathsf{True} \tag{5.103}$$

Besides, by the co-induction hypothesis, we get:

$$(R_1 \mathbin{\check{;}} R_2),(G_1 \mathbin{\hat{;}} G_2),(I_1 \mathbin{;} I_2) \models (C',\sigma',(M_2 \circ M_1')) \preceq_{Q_1 ; Q_2} (\mathbb{C},\Sigma) \tag{5.104}$$

Moreover, we know

$$(M_2 \circ M_1') < (M_2 \circ M_1) \tag{5.105}$$

2. for any σ_F, Σ_F, e, C' and σ'', if $(C,\sigma \uplus \sigma_F) \xrightarrow{e} (C',\sigma'')$, then by the premise 1, we know: for any θ_F, there exist σ', M_1', C_M' and θ' such that $\sigma'' = \sigma' \uplus \sigma_F$, $(\mathrm{C}_\mathrm{M},\theta \uplus \theta_F) \xrightarrow{e}{}^+ (\mathrm{C}_\mathrm{M}',\theta' \uplus \theta_F)$, $((\sigma,\theta),(\sigma',\theta'),\mathbf{true}) \models (G_1)^+ * \mathsf{True}$ and $R_1, G_1, I_1 \models (C',\sigma',M_1') \preceq_{Q_1} (\mathrm{C}_\mathrm{M}',\theta')$.

   By the premise 2 and Lemma 22, we know:

   there exist M_2', $\mathbb{C}'$ and Σ' such that $(\mathbb{C},\Sigma \uplus \Sigma_F) \xrightarrow{e}{}^+ (\mathbb{C}',\Sigma' \uplus \Sigma_F)$, $((\theta,\Sigma),(\theta',\Sigma'),\mathbf{true}) \models (G_2)^+ * \mathsf{True}$ and $R_2, G_2, I_2 \models (\mathrm{C}_\mathrm{M}',\theta',M_2') \preceq_{Q_2} (\mathbb{C}',\Sigma')$.

   Thus we know

$$((\sigma,\Sigma),(\sigma',\Sigma'),\mathbf{true}) \models ((G_1)^+ * \mathsf{True}) \mathbin{\hat{;}} ((G_2)^+ * \mathsf{True}) \tag{5.106}$$

   Thus we get:

$$((\sigma,\Sigma),(\sigma',\Sigma'),\mathbf{true}) \models (G_1 \mathbin{\hat{;}} G_2)^+ * \mathsf{True} \tag{5.107}$$

   Besides, by the co-induction hypothesis, we get:

$$(R_1 \mathbin{\check{;}} R_2),(G_1 \mathbin{\hat{;}} G_2),(I_1 \mathbin{;} I_2) \models (C',\sigma',(M_2' \circ M_1')) \preceq_{Q_1 ; Q_2} (\mathbb{C}',\Sigma') \tag{5.108}$$

3. for any σ' and Σ', if $((\sigma,\Sigma),(\sigma',\Sigma'),\mathbf{true}) \models (R_1 \check{;} R_2)^+ * \mathsf{Id}$, then we know

$$((\sigma,\Sigma),(\sigma',\Sigma'),\mathbf{true}) \models ((R_1)^+ \check{;} (R_2)^+) * \mathsf{Id} \tag{5.109}$$

By Lemma 26, we know

$$((R_1)^+ \check{;} (R_2)^+) * \mathsf{Id} \;\Rightarrow\; ((R_1)^+ * \mathsf{Id}) \check{;} ((R_2)^+ * \mathsf{Id}) \tag{5.110}$$

Thus we get: there exist θ, θ', b_1 and b_2 such that $b = b_1 \vee b_2$,

$$((\sigma,\theta),(\sigma',\theta'),b_1) \models (R_1)^+ * \mathsf{Id} \;\text{ and }\; ((\theta,\Sigma),(\theta',\Sigma'),b_2) \models (R_2)^+ * \mathsf{Id} \tag{5.111}$$

From the premises, we know: there exist M_1' and M_2' such that

(a) $R_1, G_1, I_1 \models (C,\sigma',M_1') \preceq_{Q_1} (\mathrm{C_M},\theta')$;
(b) $R_2, G_2, I_2 \models (\mathrm{C_M},\theta',M_2') \preceq_{Q_2} (\mathbb{C},\Sigma')$.

By the co-induction hypothesis, we get:

$$(R_1 \check{;} R_2),(G_1 \hat{;} G_2),(I_1 \mathbin{;} I_2) \models (C,\sigma',(M_2' \circ M_1')) \preceq_{Q_1 ; Q_2} (\mathbb{C},\Sigma') \tag{5.112}$$

4. for any σ' and Σ', if $((\sigma,\Sigma),(\sigma',\Sigma'),\mathbf{false}) \models (R_1 \check{;} R_2)^+ * \mathsf{Id}$, then we know

$$((\sigma,\Sigma),(\sigma',\Sigma'),\mathbf{false}) \models ((R_1)^+ * \mathsf{Id}) \check{;} ((R_2)^+ * \mathsf{Id}) \tag{5.113}$$

Thus we get: there exist θ and θ' such that

$$((\sigma,\theta),(\sigma',\theta'),\mathbf{false}) \models (R_1)^+ * \mathsf{Id} \;\text{ and }\; ((\theta,\Sigma),(\theta',\Sigma'),\mathbf{false}) \models (R_2)^+ * \mathsf{Id} \tag{5.114}$$

From the premises, we know:

(a) $R_1, G_1, I_1 \models (C,\sigma',M_1) \preceq_{Q_1} (\mathrm{C_M},\theta')$;
(b) $R_2, G_2, I_2 \models (\mathrm{C_M},\theta',M_2) \preceq_{Q_2} (\mathbb{C},\Sigma')$.

By the co-induction hypothesis, we get:

$$(R_1 \check{;} R_2),(G_1 \hat{;} G_2),(I_1 \mathbin{;} I_2) \models (C,\sigma',(M_2 \circ M_1)) \preceq_{Q_1 ; Q_2} (\mathbb{C},\Sigma') \tag{5.115}$$

5. if $C = \mathbf{skip}$, then by the premise 1, we know: for any θ_F, one of the following holds:

   (a) either, there exists θ' such that $(\mathrm{C_M}, \theta \uplus \theta_F) \longrightarrow^+ (\mathbf{skip}, \theta' \uplus \theta_F)$, $((\sigma,\theta),(\sigma,\theta'),\mathbf{true}) \models (G_1)^+ * \mathsf{True}$ and $(\sigma,\theta') \models Q_1$.
   By the premise 2 and Lemma 23, we know: for any Σ_F, one of the following holds:

      i. there exists Σ' such that $(\mathbb{C}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbf{skip}, \Sigma' \uplus \Sigma_F)$, $((\theta,\Sigma),(\theta',\Sigma'),\mathbf{true}) \models (G_2)^+ * \mathsf{True}$ and $(\theta',\Sigma') \models Q_2$.
      Thus we know

$$((\sigma,\Sigma),(\sigma,\Sigma'),\mathbf{true}) \models ((G_1)^+ * \mathsf{True}) \hat{;} ((G_2)^+ * \mathsf{True}) \tag{5.116}$$

      Thus we get:

$$((\sigma,\Sigma),(\sigma,\Sigma'),\mathbf{true}) \models (G_1 \hat{;} G_2)^+ * \mathsf{True} \tag{5.117}$$

      Besides, we get:

$$(\sigma,\Sigma') \models (Q_1 \mathbin{;} Q_2) \tag{5.118}$$

ii. or, $\mathbb{C} = \mathbf{skip}$, $((\theta, \Sigma), (\theta', \Sigma), \mathbf{false}) \models (G_2)^+ * \mathsf{True}$ and $(\theta', \Sigma) \models Q_2$.
We get:
$$(\sigma, \Sigma) \models (Q_1 \mathbin{⨟} Q_2) \tag{5.119}$$

(b) or, $\mathrm{C_M} = \mathbf{skip}$ and $(\sigma, \theta) \models Q_1$.
By the premise 2, we know one of the following holds:

i. there exists Σ' such that $(\mathbb{C}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbf{skip}, \Sigma' \uplus \Sigma_F)$,
$((\theta, \Sigma), (\theta, \Sigma'), \mathbf{true}) \models (G_2)^+ * \mathsf{True}$ and $(\theta, \Sigma') \models Q_2$.
Since $(\sigma, \theta) \models I_1 * \mathbf{true}$, we know: $((\sigma, \theta), (\sigma, \theta), \mathbf{true}) \models (G_1)^+ * \mathsf{True}$.
Thus we know
$$((\sigma, \Sigma), (\sigma, \Sigma'), \mathbf{true}) \models ((G_1)^+ * \mathsf{True}) \mathbin{⨟} ((G_2)^+ * \mathsf{True}) \tag{5.120}$$
Thus we get:
$$((\sigma, \Sigma), (\sigma, \Sigma'), \mathbf{true}) \models (G_1 \mathbin{⨟} G_2)^+ * \mathsf{True} \tag{5.121}$$
Besides, we get:
$$(\sigma, \Sigma') \models (Q_1 \mathbin{⨟} Q_2) \tag{5.122}$$

ii. or, $\mathbb{C} = \mathbf{skip}$ and $(\theta, \Sigma) \models Q_2$.
We get:
$$(\sigma, \Sigma) \models (Q_1 \mathbin{⨟} Q_2) \tag{5.123}$$

6. for any σ_F and Σ_F, if $(C, \sigma \uplus \sigma_F) \longrightarrow \mathbf{abort}$, then by the premise 1, we know: for any θ_F, $(\mathrm{C_M}, \theta \uplus \theta_F) \longrightarrow^+ \mathbf{abort}$. By the premise 2 and Lemma 24, we know: $(\mathbb{C}, \Sigma \uplus \Sigma_F) \longrightarrow^+ \mathbf{abort}$.

Thus we are done. □

Lemma 21. If $I \triangleright G$, $R, G, I \models (C, \sigma, M) \preceq_Q (\mathbb{C}, \Sigma)$, $(C, \sigma \uplus \sigma_F) \longrightarrow^{n+1} (C', \sigma'')$ and $\Sigma \bot \Sigma_F$, then there exists σ' such that $\sigma'' = \sigma' \uplus \sigma_F$ and one of the following holds:

(1) either, there exist M', $\mathbb{C}'$ and Σ' such that $(\mathbb{C}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$,
$((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', M') \preceq_Q (\mathbb{C}', \Sigma')$;

(2) or, there exists M' such that $M' < M$,
$((\sigma, \Sigma), (\sigma', \Sigma), \mathbf{false}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', M') \preceq_Q (\mathbb{C}, \Sigma)$.

Proof: By induction over n.
Base Case: $n = 0$. By Definition 2.
Inductive Step: $n = k + 1$. Thus there exist C_1 and σ_1' such that
$$(C, \sigma \uplus \sigma_F) \longrightarrow^1 (C_1, \sigma_1') \quad \text{and} \quad (C_1, \sigma_1') \longrightarrow^n (C', \sigma'')$$

By Definition 2, we know there exists σ_1 such that $\sigma_1' = \sigma_1 \uplus \sigma_F$ and one of the following holds:

(i) either, there exist M_1, $\mathbb{C}_1$ and Σ_1 such that $(\mathbb{C}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}_1, \Sigma_1 \uplus \Sigma_F)$,
$((\sigma, \Sigma), (\sigma_1, \Sigma_1), \mathbf{true}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C_1, \sigma_1, M_1) \preceq_Q (\mathbb{C}_1, \Sigma_1)$.

By the induction hypothesis, we know: there exists σ' such that $\sigma'' = \sigma' \uplus \sigma_F$ and one of the following holds:

(a) either, there exist M', $\mathbb{C}'$ and Σ' such that $(\mathbb{C}_1, \Sigma_1 \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$,
$((\sigma_1, \Sigma_1), (\sigma', \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', M') \preceq_Q (\mathbb{C}', \Sigma')$.
Then
$$(\mathbb{C}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F).$$
Since $I \triangleright G$, we know

$$((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}.$$

(b) or, there exists M' such that $M' < M_1$,
$((\sigma_1, \Sigma_1), (\sigma', \Sigma_1), \mathbf{false}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', M') \preceq_Q (\mathbb{C}_1, \Sigma_1)$.
Since $I \triangleright G$, we know

$$((\sigma, \Sigma), (\sigma', \Sigma_1), \mathbf{true}) \models G^+ * \mathsf{True}.$$

(ii) or, there exists M_1 such that $M_1 < M$,
$((\sigma, \Sigma), (\sigma_1, \Sigma), \mathbf{false}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C_1, \sigma_1, M_1) \preceq_Q (\mathbb{C}, \Sigma)$.

The case is similar.

Thus we are done. □

Lemma 22. If $I \triangleright G$, $R, G, I \models (C, \sigma, M) \preceq_Q (\mathbb{C}, \Sigma)$, $(C, \sigma \uplus \sigma_F) \xrightarrow{e}{}^{n+1} (C', \sigma'')$ and $\Sigma \bot \Sigma_F$, then there exist σ', M', $\mathbb{C}'$ and Σ' such that $\sigma'' = \sigma' \uplus \sigma_F$, $(\mathbb{C}, \Sigma \uplus \Sigma_F) \xrightarrow{e}{}^{+} (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', M') \preceq_Q (\mathbb{C}', \Sigma')$.

Proof: By induction over n. Similar to Lemma 21. □

Lemma 23. If $I \triangleright G$, $R, G, I \models (C, \sigma, M) \preceq_Q (\mathbb{C}, \Sigma)$, $(C, \sigma \uplus \sigma_F) \longrightarrow^n (\mathbf{skip}, \sigma'')$ and $\Sigma \bot \Sigma_F$, then there exists σ' such that $\sigma'' = \sigma' \uplus \sigma_F$ and one of the following holds:

(1) either, there exists Σ' such that $(\mathbb{C}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbf{skip}, \Sigma' \uplus \Sigma_F)$,
$((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma', \Sigma') \models Q$;

(2) or, $\mathbb{C} = \mathbf{skip}$, $((\sigma, \Sigma), (\sigma', \Sigma), \mathbf{false}) \models G^+ * \mathsf{True}$ and $(\sigma', \Sigma) \models Q$.

Proof: By induction over n. Similar to Lemma 21. □

Lemma 24. If $R, G, I \models (C, \sigma, M) \preceq_Q (\mathbb{C}, \Sigma)$ and $(C, \sigma \uplus \sigma_F) \longrightarrow^{n+1} \mathbf{abort}$ and $\Sigma \bot \Sigma_F$, then $(\mathbb{C}, \Sigma \uplus \Sigma_F) \longrightarrow^+ \mathbf{abort}$.

Proof: By induction over n. Similar to Lemma 21. □

Lemma 25. If $I_1 \triangleright G_1$, $I_2 \triangleright G_2$ and $\mathsf{MPrecise}(I_1, I_2)$, then $(G_1 * \mathsf{True}) \hat{\fatsemi} (G_2 * \mathsf{True}) \Rightarrow (G_1 \hat{\fatsemi} G_2) * \mathsf{True}$.

Proof: For any σ, Σ, σ', Σ' and b, if $((\sigma, \Sigma), (\sigma', \Sigma'), b) \models (G_1 * \mathsf{True}) \hat{\fatsemi} (G_2 * \mathsf{True})$, we know there exist θ, θ', b_1 and b_2 such that

$$((\sigma, \theta), (\sigma', \theta'), b_1) \models (G_1 * \mathsf{True}), \quad ((\theta, \Sigma), (\theta', \Sigma'), b_2) \models (G_2 * \mathsf{True}), \quad b = b_1 \wedge b_2.$$

Then we know there exist σ_1, θ_1, σ'_1, θ'_1, θ_2, Σ_2, θ'_2 and Σ'_2 such that

$$((\sigma_1, \theta_1), (\sigma'_1, \theta'_1), b_1) \models G_1, \quad ((\theta_2, \Sigma_2), (\theta'_2, \Sigma'_2), b_2) \models G_2,$$
$$\sigma_1 \subseteq \sigma, \quad \theta_1 \subseteq \theta, \quad \sigma'_1 \subseteq \sigma', \quad \theta'_1 \subseteq \theta', \quad \theta_2 \subseteq \theta, \quad \Sigma_2 \subseteq \Sigma, \quad \theta'_2 \subseteq \theta', \quad \Sigma'_2 \subseteq \Sigma'$$

Since $I_1 \triangleright G_1$ and $I_2 \triangleright G_2$, we know

$$(\sigma_1, \theta_1) \models I_1, \quad (\sigma'_1, \theta'_1) \models I_1, \quad (\theta_2, \Sigma_2) \models I_2, \quad (\theta'_2, \Sigma'_2) \models I_2.$$

Since $\mathsf{MPrecise}(I_1, I_2)$, we know

$$\theta_1 = \theta_2, \quad \theta'_1 = \theta'_2.$$

Thus we know

$$((\sigma_1, \Sigma_2), (\sigma'_1, \Sigma'_2), b) \models G_1 \hat{\fatsemi} G_2$$

Thus

$$((\sigma, \Sigma), (\sigma', \Sigma'), b) \models (G_1 \mathbin{\hat{;}} G_2) * \mathsf{True}.$$

Then we are done. $\square$

Lemma 26. $(R_1 \mathbin{\check{;}} R_2) * \mathsf{Id} \;\Rightarrow\; (R_1 * \mathsf{Id}) \mathbin{\check{;}} (R_2 * \mathsf{Id})$.

Proof: For any σ, Σ, σ', Σ' and b, if $((\sigma, \Sigma), (\sigma', \Sigma'), b) \models (R_1 \mathbin{\check{;}} R_2) * \mathsf{Id}$, we know there exist σ_1, Σ_1, σ'_1, Σ'_1, σ_2 and Σ_2 such that

$$\begin{array}{c} ((\sigma_1, \Sigma_1), (\sigma'_1, \Sigma'_1), b) \models R_1 \mathbin{\check{;}} R_2, \\ \sigma = \sigma_1 \uplus \sigma_2, \quad \Sigma = \Sigma_1 \uplus \Sigma_2, \quad \sigma' = \sigma'_1 \uplus \sigma_2, \quad \Sigma' = \Sigma'_1 \uplus \Sigma_2 \end{array}$$

Then we know there exist θ, θ', b_1 and b_2 such that

$$((\sigma_1, \theta), (\sigma'_1, \theta'), b_1) \models R_1, \quad ((\theta, \Sigma_1), (\theta', \Sigma'_1), b_2) \models R_2, \quad b = b_1 \vee b_2.$$

Thus we know

$$((\sigma, \theta), (\sigma', \theta'), b_1) \models R_1 * \mathsf{Id}, \quad ((\theta, \Sigma), (\theta', \Sigma'), b_2) \models R_2 * \mathsf{Id}.$$

Thus

$$((\sigma, \Sigma), (\sigma', \Sigma'), b) \models (R_1 * \mathsf{Id}) \mathbin{\check{;}} (R_2 * \mathsf{Id}).$$

Then we are done. $\square$

5.4 Soundness of Unary Rules

Lemma 27. If $R, G, I \vdash \{p\}C\{q\}$, then $I \triangleright \{R, G\}$, $p \vee q \Rightarrow I * \mathbf{true}$ and $\mathsf{Sta}(\{p, q\}, R * \mathsf{Id})$.

Proof: By induction over the derivation of $R, G, I \vdash \{p\}C\{q\}$. For the stability, we need Lemma 28. $\square$

Lemma 28. If $\mathsf{Sta}(p, R * \mathsf{Id})$, then $\mathsf{Sta}(\lfloor p \rfloor_{\mathsf{w}}, R * \mathsf{Id})$.

Lemma 29. If $R, G, I \models (C, \sigma, ws) \preceq_{\mathcal{H};w;q} (\mathbb{D}, \Sigma)$ and $\mathcal{H} \leq \mathcal{H}'$, then $R, G, I \models (C, \sigma, ws) \preceq_{\mathcal{H}';w;q} (\mathbb{D}, \Sigma)$.

Proof: We know: if $ws' <_{\mathcal{H}} ws$ and $\mathcal{H} \leq \mathcal{H}'$, then $ws' <_{\mathcal{H}'} ws$. $\square$

We define:

$$\mathsf{inchead}(ws, (k_1, k_2)) \stackrel{\text{def}}{=} \begin{cases} (w + k_1, n + k_2) & \text{if } ws = (w, n) \\ (w + k_1, n + k_2) :: ws' & \text{if } ws = (w, n) :: ws' \end{cases}$$

Lemma 30. If $R, G, I \models (C, \sigma, ws) \preceq_{\mathcal{H};w;q} (\mathbb{D}, \Sigma)$, $w_1 \leq w$ and $ws_1 = \mathsf{inchead}(ws, (w_1, 0))$, then $R, G, I \models (C, \sigma, ws_1) \preceq_{\mathcal{H};w-w_1;q} (\mathbb{D}, \Sigma)$.

Proof: By co-induction. From the premise, we know: $(\sigma, \Sigma) \models I * \mathbf{true}$.

1. For any σ_F, Σ_F, C' and σ'', if $(C, \sigma \uplus \sigma_F) \longrightarrow (C', \sigma'')$ and $\Sigma \bot \Sigma_F$, from the premise, we know there exists σ' such that $\sigma'' = \sigma' \uplus \sigma_F$ and one of the following holds:
   (a) there exist ws', w', $\mathbb{C}'$ and Σ' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', ws') \preceq_{\mathcal{H};w';q} (\mathbb{C}', \Sigma')$.
   By the co-induction hypothesis, let $ws_1' = \mathsf{inchead}(ws', (w_1, 0))$, we know $R, G, I \models (C', \sigma', ws_1') \preceq_{\mathcal{H};w'-w_1;q} (\mathbb{C}', \Sigma')$.
   (b) there exists ws' such that $ws' <_{\mathcal{H}} ws$, $((\sigma, \Sigma), (\sigma', \Sigma), \mathbf{false}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', ws') \preceq_{\mathcal{H};w;q} (\mathbb{D}, \Sigma)$.
   By the co-induction hypothesis, let $ws_1' = \mathsf{inchead}(ws', (w_1, 0))$, we know $R, G, I \models (C', \sigma', ws_1') \preceq_{\mathcal{H};w-w_1;q} (\mathbb{D}, \Sigma)$.
   Since $ws' <_{\mathcal{H}} ws$, we know $ws_1' <_{\mathcal{H}} ws_1$.

2. For any σ_F, Σ_F, e, C' and σ'', if $(C, \sigma \uplus \sigma_F) \xrightarrow{e} (C', \sigma'')$ and $\Sigma \bot \Sigma_F$, the proof is similar to the previous case.

3. For any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models R^+ * \mathsf{Id}$, from the premise, we know: there exist ws' and w' such that $R, G, I \models (C, \sigma', ws') \preceq_{\mathcal{H};w';q} (\mathbb{D}, \Sigma')$.

   By the co-induction hypothesis, let $ws_1' = \mathsf{inchead}(ws', (w_1, 0))$, we know $R, G, I \models (C, \sigma', ws_1') \preceq_{\mathcal{H};w'-w_1;q} (\mathbb{D}, \Sigma')$.

4. For any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{false}) \models R^+ * \mathsf{Id}$, from the premise, we know: $R, G, I \models (C, \sigma', ws) \preceq_{\mathcal{H};w;q} (\mathbb{D}, \Sigma')$.

   By the co-induction hypothesis, we know $R, G, I \models (C, \sigma', ws_1) \preceq_{\mathcal{H};w-w_1;q} (\mathbb{D}, \Sigma')$.

5. If $C = \mathbf{skip}$, then for any Σ_F, if $\Sigma \bot \Sigma_F$, from the premise we know one of the following holds:
   (a) there exist w', $\mathbb{C}'$ and Σ' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma, \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma, w', \mathbb{C}', \Sigma') \models q$.
   (b) there exists w' such that $ws = (w', 0)$ and $(\sigma, w + w', \mathbb{D}, \Sigma) \models q$.
   Thus $ws_1 = (w' + w_1, 0)$ and $(\sigma, (w - w_1) + (w' + w_1), \mathbb{D}, \Sigma) \models q$.

6. For any σ_F and Σ_F, if $(C, \sigma \uplus \sigma_F) \longrightarrow \mathbf{abort}$ and $\Sigma \bot \Sigma_F$, from the premise we know: $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ \mathbf{abort}$.

Thus we are done. $\square$

The HIDE-w rule.

Lemma 31 (HIDE-w). If $R, G, I \models \{p\}C\{q\}$, then $R, G, I \models \{\lfloor p \rfloor_{\mathsf{w}}\}C\{\lfloor q \rfloor_{\mathsf{w}}\}$.

Proof: We want to prove: for all σ, w_1, $\mathbb{D}$ and Σ, if $(\sigma, w_1, \mathbb{D}, \Sigma) \models \lfloor p \rfloor_{\mathsf{w}}$, then

$$R, G, I \models (C, \sigma, (0, |C|)) \preceq_{\mathsf{height}(C); w_1; \lfloor q \rfloor_{\mathsf{w}}} (\mathbb{D}, \Sigma).$$

We know there exists w such that

$$(\sigma, w, \mathbb{D}, \Sigma) \models p$$

From the premise, we know:

$$R, G, I \models (C, \sigma, (0, |C|)) \preceq_{\mathsf{height}(C); w; q} (\mathbb{D}, \Sigma).$$

By Lemma 32, we are done. $\square$

Lemma 32. If $R, G, I \models (C, \sigma, ws) \preceq_{\mathcal{H}; w; q} (\mathbb{D}, \Sigma)$, then $R, G, I \models (C, \sigma, ws) \preceq_{\mathcal{H}; w_1; \lfloor q \rfloor_{\mathsf{w}}} (\mathbb{D}, \Sigma)$.

Proof: By co-induction. From the premise, we know: $(\sigma, \Sigma) \models I * \mathbf{true}$.

1. For any σ_F, Σ_F, C' and σ'', if $(C, \sigma \uplus \sigma_F) \longrightarrow (C', \sigma'')$ and $\Sigma \bot \Sigma_F$, from the premise, we know there exists σ' such that $\sigma'' = \sigma' \uplus \sigma_F$ and one of the following holds:
   (a) there exist ws', w', $\mathbb{C}'$ and Σ' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', ws') \preceq_{\mathcal{H}; w'; q} (\mathbb{C}', \Sigma')$.
   By the co-induction hypothesis, we know $R, G, I \models (C', \sigma', ws') \preceq_{\mathcal{H}; w_1; \lfloor q \rfloor_{\mathsf{w}}} (\mathbb{C}', \Sigma')$.
   (b) there exists ws' such that $ws' <_{\mathcal{H}} ws$, $((\sigma, \Sigma), (\sigma', \Sigma), \mathbf{false}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', ws') \preceq_{\mathcal{H}; w; q} (\mathbb{D}, \Sigma)$.
   By the co-induction hypothesis, we know $R, G, I \models (C', \sigma', ws') \preceq_{\mathcal{H}; w_1; \lfloor q \rfloor_{\mathsf{w}}} (\mathbb{D}, \Sigma)$.
2. For any σ_F, Σ_F, e, C' and σ'', if $(C, \sigma \uplus \sigma_F) \xrightarrow{e} (C', \sigma'')$ and $\Sigma \bot \Sigma_F$, the proof is similar to the previous case.
3. For any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models R^+ * \mathsf{Id}$, from the premise, we know: there exist ws' and w' such that $R, G, I \models (C, \sigma', ws') \preceq_{\mathcal{H}; w'; q} (\mathbb{D}, \Sigma')$.
   By the co-induction hypothesis, we know $R, G, I \models (C, \sigma', ws') \preceq_{\mathcal{H}; w_1; \lfloor q \rfloor_{\mathsf{w}}} (\mathbb{D}, \Sigma')$.
4. For any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{false}) \models R^+ * \mathsf{Id}$, from the premise, we know: $R, G, I \models (C, \sigma', ws) \preceq_{\mathcal{H}; w; q} (\mathbb{D}, \Sigma')$.
   By the co-induction hypothesis, we know $R, G, I \models (C, \sigma', ws) \preceq_{\mathcal{H}; w_1; \lfloor q \rfloor_{\mathsf{w}}} (\mathbb{D}, \Sigma')$.
5. If $C = \mathbf{skip}$, then for any Σ_F, if $\Sigma \bot \Sigma_F$, from the premise we know one of the following holds:
   (a) there exist w', $\mathbb{C}'$ and Σ' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma, \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma, w', \mathbb{C}', \Sigma') \models q$.
   Thus $(\sigma, w', \mathbb{C}', \Sigma') \models \lfloor q \rfloor_{\mathsf{w}}$.
   (b) there exists w' such that $ws = (w', 0)$ and $(\sigma, w + w', \mathbb{D}, \Sigma) \models q$.
   Thus $(\sigma, w_1 + w', \mathbb{D}, \Sigma) \models \lfloor q \rfloor_{\mathsf{w}}$.
6. For any σ_F and Σ_F, if $(C, \sigma \uplus \sigma_F) \longrightarrow \mathbf{abort}$ and $\Sigma \bot \Sigma_F$, from the premise we know: $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ \mathbf{abort}$.

Thus we are done. $\square$

The WHILE rule.

Lemma 33 (WHILE). If

1. $R, G, I \models \{p'\}C\{p\}$;
2. $p \wedge B \Rightarrow p' * (\mathsf{wf}(1) \wedge \mathsf{emp})$;
3. $\mathsf{Sta}(p, R * \mathsf{Id})$; $I \triangleright \{R, G\}$; $p \Rightarrow (B = B) * I$;

then $R, G, I \models \{p\}\textbf{while}\ (B)\ C\{p \wedge \neg B\}$.

Proof: We want to prove: for all σ, w, $\mathbb{D}$ and Σ, if $(\sigma, w, \mathbb{D}, \Sigma) \models p$, then

$$R, G, I \models (\textbf{while}\ (B)\ C, \sigma, (0, |\textbf{while}\ (B)\ C|)) \preceq_{\mathsf{height}(\textbf{while}\ (B)\ C);w;p\wedge\neg B} (\mathbb{D}, \Sigma).$$

We know $|\textbf{while}\ (B)\ C| = 1$ and can prove $\mathsf{height}(\textbf{while}\ (B)\ C) = \mathsf{height}(C) + 1$.

By co-induction. From $(\sigma, w, \mathbb{D}, \Sigma) \models p$, since $p \Rightarrow I * (B = B)$, we know:

$$(\sigma, \Sigma) \models I * \textbf{true} \tag{5.124}$$

1. For any σ_F and Σ_F, if $(\textbf{while}\ (B)\ C, \sigma \uplus \sigma_F) \longrightarrow (C; \mathsf{while}\ (B)\{C\}, \sigma \uplus \sigma_F)$ and $[\![B]\!]_{\sigma\uplus\sigma_F} = \textbf{true}$, below we prove 1(b) of Definition 8 holds.

   Since $(\sigma, \Sigma) \models (B = B)$, we know $[\![B]\!]_\sigma = \textbf{true}$. Then we know

   $$(\sigma, w, \mathbb{D}, \Sigma) \models p \wedge B \tag{5.125}$$

   Since $p \wedge B \Rightarrow p' * (\mathsf{wf}(1) \wedge \mathsf{emp})$, we know there exists w' such that $w' < w$ and

   $$(\sigma, w', \mathbb{D}, \Sigma) \models p' \tag{5.126}$$

   From the premise 1, we know $R, G, I \models (C, \sigma, (0, |C|)) \preceq_{\mathsf{height}(C);w';p} (\mathbb{D}, \Sigma)$.

   By Lemma 34, we know: let

   $$ws' = (0, 0) :: (w', |C| + 1) \tag{5.127}$$

   then

   $$R, G, I \models (C; \mathsf{while}\ (B)\{C\}, \sigma, ws') \preceq_{\mathsf{height}(C)+1;w;p\wedge\neg B} (\mathbb{D}, \Sigma) \tag{5.128}$$

   We know $ws' <_{\mathsf{height}(C)+1} (0, 1)$.

   Also, since $I \triangleright G$ and $(\sigma, \Sigma) \models I * \textbf{true}$, we know $((\sigma, \Sigma), (\sigma, \Sigma), \textbf{false}) \models G^+ * \mathsf{True}$.

2. For any σ_F and Σ_F, if $(\textbf{while}\ (B)\ C, \sigma \uplus \sigma_F) \longrightarrow (\textbf{skip}, \sigma \uplus \sigma_F)$ and $[\![B]\!]_{\sigma\uplus\sigma_F} = \textbf{false}$, below we prove 1(b) of Definition 8 holds.

   since $(\sigma, \Sigma) \models (B = B)$, we know $[\![B]\!]_\sigma = \textbf{false}$. Then we know

   $$(\sigma, w, \mathbb{D}, \Sigma) \models p \wedge \neg B \tag{5.129}$$

   By the SKIP and FRAME rules, we know:

   $$R, G, I \models (\textbf{skip}, \sigma, (0, 0)) \preceq_{\mathsf{height}(C)+1;w;p\wedge\neg B} (\mathbb{D}, \Sigma) \tag{5.130}$$

   We know $(0, 0) <_{\mathsf{height}(C)+1} (0, 1)$ and $((\sigma, \Sigma), (\sigma, \Sigma), \textbf{false}) \models G^+ * \mathsf{True}$.

3. For any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \textbf{true}) \models R^+ * \mathsf{Id}$,

   since $\mathsf{Sta}(p, R * \mathsf{Id})$, we know $\mathsf{Sta}(p, R^+ * \mathsf{Id})$, thus there exists w' such that

   $$(\sigma', w', \mathbb{D}, \Sigma') \models p \tag{5.131}$$

   By the co-induction hypothesis, we get:

   $$R, G, I \models (\textbf{while}\ (B)\ C, \sigma', (0, 1)) \preceq_{\mathsf{height}(C)+1;w';p\wedge\neg B} (\mathbb{D}, \Sigma') \tag{5.132}$$

4. For any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{false}) \models R^{+} * \mathsf{Id}$,
since $\mathsf{Sta}(p, R * \mathsf{Id})$, we know $\mathsf{Sta}(p, R^{+} * \mathsf{Id})$, thus

$$(\sigma', w, \mathbb{D}, \Sigma') \models p \tag{5.133}$$

By the co-induction hypothesis, we get:

$$R, G, I \models (\mathbf{while}\ (B)\ C, \sigma', (0,1)) \preceq_{\mathsf{height}(C)+1; w; p \wedge \neg B} (\mathbb{D}, \Sigma') \tag{5.134}$$

Thus we are done. □

Lemma 34. If

1. $R, G, I \models (C_1, \sigma, ws_1) \preceq_{\mathcal{H}; w'_0; p} (\mathbb{D}, \Sigma)$;
2. for all σ, w, $\mathbb{D}$ and Σ, if $(\sigma, w, \mathbb{D}, \Sigma) \models p'$, then $R, G, I \models (C, \sigma, (0, |C|)) \preceq_{\mathcal{H}; w; p} (\mathbb{D}, \Sigma)$;
3. $p \wedge B \Rightarrow p' * (\mathsf{wf}(1) \wedge \mathsf{emp})$;
4. $\mathsf{Sta}(p, R * \mathsf{Id})$; $I \triangleright \{R, G\}$; $p \Rightarrow (B = B) * I$;
5. $ws = (0,0) :: \mathsf{inchead}(ws_1, (w'_0, 1))$;
6. $\mathsf{root}(ws_1) = (w_1, \_)$; $w'_0 + w_1 \le w_0$;

then $R, G, I \models (C_1; \mathsf{while}\ (B)\{C\}, \sigma, ws) \preceq_{\mathcal{H}+1; w_0; p \wedge \neg B} (\mathbb{D}, \Sigma)$.

Proof: By co-induction. From the first premise, we know $(\sigma, \Sigma) \models I * \mathbf{true}$.

1. For any σ_F, Σ_F, C'_1 and σ'', if $(C_1; \mathsf{while}\ (B)\{C\}, \sigma \uplus \sigma_F) \longrightarrow (C'_1; \mathsf{while}\ (B)\{C\}, \sigma'')$, i.e., $(C_1, \sigma \uplus \sigma_F) \longrightarrow (C'_1, \sigma'')$, from the premise 1, we know: there exists σ' such that $\sigma'' = \sigma' \uplus \sigma_F$ and one of the following holds:
   (a) there exist ws'_1, w''_0, $\mathbb{C}'$ and Σ' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^{+} (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models G^{+} * \mathsf{True}$ and $R, G, I \models (C'_1, \sigma', ws'_1) \preceq_{\mathcal{H}; w''_0; p} (\mathbb{C}', \Sigma')$.
   Suppose $\mathsf{root}(ws'_1) = (w'_1, \_)$.
   By the co-induction hypothesis, let $ws' = (0,0) :: \mathsf{inchead}(ws'_1, (w''_0, 1))$, we know:
   $R, G, I \models (C'_1; \mathsf{while}\ (B)\{C\}, \sigma', ws') \preceq_{\mathcal{H}+1; w''_0 + w'_1; p \wedge \neg B} (\mathbb{C}', \Sigma')$.
   (b) there exists ws'_1 such that $ws'_1 <_{\mathcal{H}} ws_1$,
   $((\sigma, \Sigma), (\sigma', \Sigma), \mathbf{false}) \models G^{+} * \mathsf{True}$ and $R, G, I \models (C'_1, \sigma', ws'_1) \preceq_{\mathcal{H}; w'_0; p} (\mathbb{D}, \Sigma)$.
   Suppose $\mathsf{root}(ws'_1) = (w'_1, \_)$. Since $ws'_1 <_{\mathcal{H}} ws_1$, we know $w'_1 \le w_1$. Thus $w'_0 + w'_1 \le w_0$.
   By the co-induction hypothesis, let $ws' = (0,0) :: \mathsf{inchead}(ws'_1, (w'_0, 1))$, we know:
   $R, G, I \models (C'_1; \mathsf{while}\ (B)\{C\}, \sigma', ws') \preceq_{\mathcal{H}+1; w_0; p \wedge \neg B} (\mathbb{D}, \Sigma)$.
   Since $ws'_1 <_{\mathcal{H}} ws_1$, we know: $ws' <_{\mathcal{H}+1} ws$.
2. For any σ_F, Σ_F, e, C'_1 and σ'', if $(C_1; \mathsf{while}\ (B)\{C\}, \sigma \uplus \sigma_F) \xrightarrow{e} (C'_1; \mathsf{while}\ (B)\{C\}, \sigma'')$, the proof is similar to the previous case.
3. For any σ_F and Σ_F, if $(C_1; \mathsf{while}\ (B)\{C\}, \sigma \uplus \sigma_F) \longrightarrow (\mathsf{while}\ (B)\{C\}, \sigma \uplus \sigma_F)$, i.e., $C_1 = \mathbf{skip}$, from the premise 1, we know one of the following holds:
   (a) there exists w_1 such that $ws_1 = (w_1, 0)$ and $(\sigma, w_1 + w'_0, \mathbb{D}, \Sigma) \models p$.
   Thus $ws = (0,0) :: (w_1 + w'_0, 1)$. We know $(0,0) :: (w_1 + w'_0, 0) <_{\mathcal{H}+1} ws$.
   Also we know $((\sigma, \Sigma), (\sigma, \Sigma), \mathbf{false}) \models G^{+} * \mathsf{True}$.
   Below we prove:

$$R, G, I \models (\mathsf{while}\ (B)\{C\}, \sigma, (0,0) :: (w_1 + w'_0, 0)) \preceq_{\mathcal{H}+1; w_0; p \wedge \neg B} (\mathbb{C}', \Sigma') \tag{5.135}$$

   By co-induction. Since $p \Rightarrow I * (B = B)$, we know $(\sigma, \Sigma') \models I * \mathbf{true}$.

i. For any σ_F and Σ_F, if $(\mathsf{while}\ (B)\{C\}, \sigma \uplus \sigma_F) \longrightarrow (C; \mathsf{while}\ (B)\{C\}, \sigma \uplus \sigma_F)$ and $[\![B]\!]_{\sigma \uplus \sigma_F} =$ **true**, below we prove 1(b) of Definition 8 holds.
Since $(\sigma, \Sigma') \models (B = B)$, we know $[\![B]\!]_\sigma = \mathbf{true}$. Then we know

$$(\sigma, w_1 + w_0', \mathbb{C}', \Sigma') \models p \wedge B \tag{5.136}$$

Since $p \wedge B \Rightarrow p' * (\mathsf{wf}(1) \wedge \mathsf{emp})$, we know there exists w_1' such that $w_1' < w_1 + w_0'$ and

$$(\sigma, w_1', \mathbb{C}', \Sigma') \models p' \tag{5.137}$$

From the premise 2, we know $R, G, I \models (C, \sigma, (0, |C|)) \preceq_{\mathcal{H}; w_1'; p} (\mathbb{C}', \Sigma')$.
By the co-induction hypothesis, we know:

$$R, G, I \models (C; \mathsf{while}\ (B)\{C\}, \sigma, (0,0) :: (w_1', |C| + 1)) \preceq_{\mathcal{H}+1; w_0; p \wedge \neg B} (\mathbb{C}', \Sigma') \tag{5.138}$$

We know $(0,0) :: (w_1', |C| + 1) <_{\mathcal{H}+1} (0,0) :: (w_1 + w_0', 0)$.
Also we know $((\sigma, \Sigma'), (\sigma, \Sigma'), \mathbf{false}) \models G^+ * \mathsf{True}$.

ii. For any σ_F and Σ_F, if $(\mathsf{while}\ (B)\{C\}, \sigma \uplus \sigma_F) \longrightarrow (\mathbf{skip}, \sigma \uplus \sigma_F)$ and $[\![B]\!]_{\sigma \uplus \sigma_F} = \mathbf{false}$, below we prove 1(b) of Definition 8 holds.
Since $(\sigma, \Sigma') \models (B = B)$, we know $[\![B]\!]_\sigma = \mathbf{false}$. Since $(\sigma, w_1 + w_0', \mathbb{C}', \Sigma') \models p$, we know:

$$(\sigma, w_1 + w_0', \mathbb{C}', \Sigma') \models p \wedge \neg B \tag{5.139}$$

Since $w_1 + w_0' \leq w_0$, we know:

$$(\sigma, w_0, \mathbb{C}', \Sigma') \models p \wedge \neg B \tag{5.140}$$

By the SKIP and FRAME rules, we know:

$$R, G, I \models (\mathbf{skip}, \sigma, (0,0)) \preceq_{\mathcal{H}+1; w_0; p \wedge \neg B} (\mathbb{C}', \Sigma') \tag{5.141}$$

We know $(0,0) <_{\mathcal{H}+1} (0,0) :: (w_1 + w_0', 0)$ and $((\sigma, \Sigma'), (\sigma, \Sigma'), \mathbf{false}) \models G^+ * \mathsf{True}$.

iii. For any σ' and Σ'', if $((\sigma, \Sigma'), (\sigma', \Sigma''), \mathbf{true}) \models R^+ * \mathsf{Id}$,
since $\mathsf{Sta}(p, R * \mathsf{Id})$, we know $\mathsf{Sta}(p, R^+ * \mathsf{Id})$, thus there exists w_1' such that

$$(\sigma', w_1' + w_0', \mathbb{C}', \Sigma'') \models p \tag{5.142}$$

By the co-induction hypothesis, we get:

$$R, G, I \models (\mathsf{while}\ (B)\{C\}, \sigma', (0,0) :: (w_1' + w_0', 0)) \preceq_{\mathcal{H}+1; w_1' + w_0'; p \wedge \neg B} (\mathbb{C}', \Sigma'') \tag{5.143}$$

iv. For any σ' and Σ'', if $((\sigma, \Sigma'), (\sigma', \Sigma''), \mathbf{false}) \models R^+ * \mathsf{Id}$,
since $\mathsf{Sta}(p, R * \mathsf{Id})$, we know $\mathsf{Sta}(p, R^+ * \mathsf{Id})$, thus

$$(\sigma', w_1 + w_0', \mathbb{C}', \Sigma'') \models p \tag{5.144}$$

By the co-induction hypothesis, we get:

$$R, G, I \models (\mathsf{while}\ (B)\{C\}, \sigma', (0,0) :: (w_1 + w_0', 0)) \preceq_{\mathcal{H}+1; w_0; p \wedge \neg B} (\mathbb{C}', \Sigma'') \tag{5.145}$$

Thus we have proved (5.135).

(b) there exist w_1', $\mathbb{C}'$ and Σ' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma, \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma, w_1', \mathbb{C}', \Sigma') \models p$.
We can prove:

$$R, G, I \models (\mathsf{while}\ (B)\{C\}, \sigma, (0,0) :: (w_1', 0)) \preceq_{\mathcal{H}+1; w_1'; p \wedge \neg B} (\mathbb{C}', \Sigma') \tag{5.146}$$

in the similar way as the previous case.

4. For any σ' and Σ', if $((\sigma,\Sigma),(\sigma',\Sigma'),\mathbf{true}) \models R^+ * \mathsf{Id}$,

   from the premise, we know there exist ws'_1 and w''_0 such that $R, G, I \models (C_1, \sigma', ws'_1) \preceq_{\mathcal{H};w''_0;p} (\mathbb{D}, \Sigma')$.

   Suppose $\mathsf{root}(ws'_1) = (w'_1, \_)$.

   By the co-induction hypothesis, we know: let $ws' = (0,0) :: \mathsf{inchead}(ws'_1, (w''_0, 1))$, then $R, G, I \models (C_1; \mathsf{while}\ (B)\{C\}, \sigma', ws') \preceq_{\mathcal{H}+1;w''_0+w'_1;p\wedge\neg B} (\mathbb{D}, \Sigma')$.

5. For any σ' and Σ', if $((\sigma,\Sigma),(\sigma',\Sigma'),\mathbf{false}) \models R^+ * \mathsf{Id}$,

   from the premise, we know: $R, G, I \models (C_1, \sigma', ws_1) \preceq_{\mathcal{H};w'_0;p} (\mathbb{D}, \Sigma')$.

   By the co-induction hypothesis, we know: $R, G, I \models (C_1; \mathsf{while}\ (B)\{C\}, \sigma', ws) \preceq_{\mathcal{H}+1;w_0;p\wedge\neg B} (\mathbb{D}, \Sigma')$.

6. For any σ_F and Σ_F, if $(C_1; \mathsf{while}\ (B)\{C\}, \sigma \uplus \sigma_F) \longrightarrow \mathbf{abort}$, we know $(C_1, \sigma \uplus \sigma_F) \longrightarrow \mathbf{abort}$. By the premise 1, we know: $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ \mathbf{abort}$.

Thus we are done. $\square$

The SEQ rule.

Lemma 35 (SEQ). If

1. $R, G, I \models \{p\}C_1\{p'\}$;
2. $R, G, I \models \{p'\}C_2\{q\}$;
3. $I \triangleright G$;

then $R, G, I \models \{p\}C_1; C_2\{q\}$.

Proof: We want to prove: for all σ, w, $\mathbb{D}$ and Σ, if $(\sigma, w, \mathbb{D}, \Sigma) \models p$, then

$$R, G, I \models (C_1; C_2, \sigma, (0, |C_1; C_2|)) \preceq_{\mathsf{height}(C_1;C_2);w;q} (\mathbb{D}, \Sigma).$$

We know $|C_1; C_2| = |C_1| + |C_2| + 1$ and can prove $\mathsf{height}(C_1; C_2) = max\{\mathsf{height}(C_1), \mathsf{height}(C_2)\}$.

Since $(\sigma, w, \mathbb{D}, \Sigma) \models p$, by the premise 1, we know:

$$R, G, I \models (C_1, \sigma, (0, |C_1|)) \preceq_{\mathsf{height}(C_1);w;p'} (\mathbb{D}, \Sigma).$$

By Lemma 29, we know: $R, G, I \models (C_1, \sigma, (0, |C_1|)) \preceq_{\mathsf{height}(C_1;C_2);w;p'} (\mathbb{D}, \Sigma)$.

From the premise 2, by Lemma 29, we know: for all σ, w, $\mathbb{D}$ and Σ, if $(\sigma, w, \mathbb{D}, \Sigma) \models p'$, then $R, G, I \models (C_2, \sigma, (0, |C_2|)) \preceq_{\mathsf{height}(C_1;C_2);w;q} (\mathbb{D}, \Sigma)$.

By Lemma 36, we are done. $\square$

Lemma 36. If

1. $R, G, I \models (C_1, \sigma, ws_1) \preceq_{\mathcal{H};w;p'} (\mathbb{D}, \Sigma)$;
2. for all σ, w, $\mathbb{D}$ and Σ, if $(\sigma, w, \mathbb{D}, \Sigma) \models p'$, then $R, G, I \models (C_2, \sigma, (0, |C_2|)) \preceq_{\mathcal{H};w;q} (\mathbb{D}, \Sigma)$;
3. $I \triangleright G$;
4. $ws = \mathsf{inchead}(ws_1, (0, |C_2| + 1))$;

then $R, G, I \models (C_1; C_2, \sigma, ws) \preceq_{\mathcal{H};w;q} (\mathbb{D}, \Sigma)$.

Proof: By co-induction. From the premise 1, we know: $(\sigma, \Sigma) \models I * \mathbf{true}$.

1. for any σ_F, Σ_F, C'_1 and σ'', if $(C_1; C_2, \sigma \uplus \sigma_F) \longrightarrow (C'_1; C_2, \sigma'')$, i.e., $(C_1, \sigma \uplus \sigma_F) \longrightarrow (C'_1, \sigma'')$,

   from the premise 1, we know: there exists σ' such that $\sigma'' = \sigma' \uplus \sigma_F$ and one of the following holds:

(a) there exist ws'_1, w', $\mathbb{C}'$ and Σ' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma', \Sigma'), \textbf{true}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C'_1, \sigma', ws'_1) \preceq_{\mathcal{H};w';p'} (\mathbb{C}', \Sigma')$.
By the co-induction hypothesis, we know: let $ws' = \mathsf{inchead}(ws'_1, (0, |C_2| + 1))$, then $R, G, I \models (C'_1; C_2, \sigma', ws') \preceq_{\mathcal{H};w';q} (\mathbb{C}', \Sigma')$.

(b) there exists ws'_1 such that $ws'_1 <_{\mathcal{H}} ws_1$, $((\sigma, \Sigma), (\sigma', \Sigma), \textbf{false}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C'_1, \sigma', ws'_1) \preceq_{\mathcal{H};w;p'} (\mathbb{D}, \Sigma)$.
By the co-induction hypothesis, we know: let $ws' = \mathsf{inchead}(ws'_1, (0, |C_2| + 1))$, $R, G, I \models (C'_1; C_2, \sigma', ws') \preceq_{\mathcal{H};w;q} (\mathbb{D}, \Sigma)$.
Since $ws'_1 <_{\mathcal{H}} ws_1$, we know: $ws' <_{\mathcal{H}} ws$.

2. for any σ_F, Σ_F, e, C'_1 and σ'', if $(C_1; C_2, \sigma \uplus \sigma_F) \xrightarrow{e} (C'_1; C_2, \sigma'')$, the proof is similar to the previous case.

3. for any σ_F and Σ_F, if $(C_1; C_2, \sigma \uplus \sigma_F) \longrightarrow (C_2, \sigma \uplus \sigma_F)$ and $C_1 = \textbf{skip}$,

   from the premise 1, we know one of the following holds:

   (a) there exist w', $\mathbb{C}'$ and Σ' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma, \Sigma'), \textbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma, w', \mathbb{C}', \Sigma') \models p'$.
   From the premise 2, we know: $R, G, I \models (C_2, \sigma, (0, |C_2|)) \preceq_{\mathcal{H};w';q} (\mathbb{C}', \Sigma')$.

   (b) there exists w_1 such that $ws_1 = (w_1, 0)$ and $(\sigma, w + w_1, \mathbb{D}, \Sigma) \models p'$.
   Thus we know $ws = (w_1, |C_2| + 1)$.
   We know $(w_1, |C_2|) <_{\mathcal{H}} ws$.
   Since $(\sigma, \Sigma) \models I * \textbf{true}$ and $I \triangleright G$, we know $((\sigma, \Sigma), (\sigma, \Sigma), \textbf{false}) \models G^+ * \mathsf{True}$.
   From the premise 2, we know: $R, G, I \models (C_2, \sigma, (0, |C_2|)) \preceq_{\mathcal{H};w+w_1;q} (\mathbb{D}, \Sigma)$.
   By Lemma 30, we get: $R, G, I \models (C_2, \sigma, (w_1, |C_2|)) \preceq_{\mathcal{H};w;q} (\mathbb{D}, \Sigma)$.

4. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \textbf{true}) \models R^+ * \mathsf{Id}$,

   from the premise, we know: there exists ws'_1 and w' such that $R, G, I \models (C_1, \sigma', ws'_1) \preceq_{\mathcal{H};w';p'} (\mathbb{D}, \Sigma')$.

   By the co-induction hypothesis, we know: let $ws' = \mathsf{inchead}(ws'_1, (0, |C_2| + 1))$, then $R, G, I \models (C_1; C_2, \sigma', ws') \preceq_{\mathcal{H};w';q} (\mathbb{D}, \Sigma')$.

5. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \textbf{false}) \models R^+ * \mathsf{Id}$,

   from the premise, we know: $R, G, I \models (C_1, \sigma', ws_1) \preceq_{\mathcal{H};w;p'} (\mathbb{D}, \Sigma')$.

   By the co-induction hypothesis, we know: $R, G, I \models (C_1; C_2, \sigma', ws) \preceq_{\mathcal{H};w;q} (\mathbb{D}, \Sigma')$.

6. for any σ_F and Σ_F, if $(C_1; C_2, \sigma \uplus \sigma_F) \longrightarrow \textbf{abort}$, we know: $(C_1, \sigma \uplus \sigma_F) \longrightarrow \textbf{abort}$. By the premise 1, we know: $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ \textbf{abort}$.

Thus we are done. $\square$

The ATOM rule.

Lemma 37 (ATOM). If

1. $\models_{\text{SL}} [p]C[q]$;
2. $(\llbracket p \rrbracket \ltimes \llbracket q \rrbracket) \Rightarrow G * \mathsf{True}$;
3. $p \vee q \Rightarrow I * \textbf{true}$;
4. $\mathsf{Locality}(C)$;

then $[I], G, I \models \{p\}\langle C\rangle\{q\}$.

Proof: We want to prove: for all σ, w, $\mathbb{D}$ and Σ, if $(\sigma, w, \mathbb{D}, \Sigma) \models p$, then

$$[I], G, I \models (\langle C\rangle, \sigma, (0, |\langle C\rangle|)) \preceq_{\mathsf{height}(\langle C\rangle); w; q} (\mathbb{D}, \Sigma).$$

We know $|\langle C\rangle| = 1$ and can prove $\mathsf{height}(\langle C\rangle) = 1$.

By co-induction. Since $p \Rightarrow I * \mathbf{true}$, we know $(\sigma, \Sigma) \models I * \mathbf{true}$. From the premises 1 and 2, we can prove:

$$(C, \sigma) \not\longrightarrow^* \mathbf{abort}\,, \quad (C, \sigma) \not\longrightarrow^\omega \cdot \tag{5.147}$$

By $\mathsf{Locality}(C)$, we know: for any σ_F,

$$(C, \sigma \uplus \sigma_F) \not\longrightarrow^* \mathbf{abort}\,, \quad (C, \sigma \uplus \sigma_F) \not\longrightarrow^\omega \cdot \tag{5.148}$$

1. for any σ_F, Σ_F, C' and σ'', if $(\langle C\rangle, \sigma \uplus \sigma_F) \longrightarrow (C', \sigma'')$,

   by the operational semantics, we know $C' = \mathbf{skip}$ and

   $$(C, \sigma \uplus \sigma_F) \longrightarrow^* (\mathbf{skip}, \sigma'') \tag{5.149}$$

   by $\mathsf{Locality}(C)$, we know: there exists σ' such that $\sigma'' = \sigma' \uplus \sigma_F$ and $(C, \sigma) \longrightarrow^* (\mathbf{skip}, \sigma')$.

   From $\models_{\text{SL}} [p]C[q]$ and $(C, \sigma) \longrightarrow^* (\mathbf{skip}, \sigma')$, we know:

   $$(\sigma', w, \mathbb{D}, \Sigma) \models q \tag{5.150}$$

   Thus we know:

   $$((\sigma, \Sigma), (\sigma', \Sigma), \mathbf{false}) \models \llbracket p \rrbracket \ltimes \llbracket q \rrbracket \tag{5.151}$$

   Since $(\llbracket p \rrbracket \ltimes \llbracket q \rrbracket) \Rightarrow G * \mathsf{True}$, we know $((\sigma, \Sigma), (\sigma', \Sigma), \mathbf{false}) \models G^+ * \mathsf{True}$.

   Since $q \Rightarrow I * \mathbf{true}$ and $\mathsf{Sta}(q, [I] * \mathsf{Id})$, by the SKIP and FRAME rules, we know:

   $$[I], G, I \models (\mathbf{skip}, \sigma', (0, 0)) \preceq_{1; w; q} (\mathbb{D}, \Sigma) \tag{5.152}$$

   Also, we know: $(0, 0) <_1 (0, 1)$.

2. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models ([I])^+ * \mathsf{Id}$, we know $\sigma' = \sigma$ and $\Sigma' = \Sigma$.

   By the co-induction hypothesis, we know: $[I], G, I \models (\langle C\rangle, \sigma, (0, 1)) \preceq_{1; w; q} (\mathbb{D}, \Sigma)$.

3. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{false}) \models ([I])^+ * \mathsf{Id}$, we know $\sigma' = \sigma$ and $\Sigma' = \Sigma$.

   By the co-induction hypothesis, we know: $[I], G, I \models (\langle C\rangle, \sigma, (0, 1)) \preceq_{1; w; q} (\mathbb{D}, \Sigma)$.

Thus we are done. □

The ATOM$^+$ rule.

Lemma 38 (ATOM$^+$). If

1. $\models_{\text{SL}} [p']C[q']$;
2. $p \Rrightarrow^a p'$; $q' \Rrightarrow^b q$; $+ \in \{a, b\}$;
3. $(\llbracket p \rrbracket \propto \llbracket q \rrbracket) \Rightarrow G * \mathsf{True}$;
4. $p \vee q \Rightarrow I * \mathbf{true}$;
5. $\mathsf{Locality}(C)$;

then $[I], G, I \models \{p\}\langle C\rangle\{q\}$.

Proof: We want to prove: for all σ, w, $\mathbb{D}$ and Σ, if $(\sigma, w, \mathbb{D}, \Sigma) \models p$, then

$$[I], G, I \models (\langle C\rangle, \sigma, (0, |\langle C\rangle|)) \preceq_{\mathsf{height}(\langle C\rangle);w;q} (\mathbb{D}, \Sigma).$$

We know $|\langle C\rangle| = 1$ and can prove $\mathsf{height}(\langle C\rangle) = 1$.

By co-induction. Since $p \Rightarrow I * \mathbf{true}$, we know $(\sigma, \Sigma) \models I * \mathbf{true}$. From the premises 1 and 2, we can prove:

$$(C, \sigma) \not\longrightarrow^* \mathbf{abort}\,, \quad (C, \sigma) \not\longrightarrow^\omega \cdot \tag{5.153}$$

By $\mathsf{Locality}(C)$, we know: for any σ_F,

$$(C, \sigma \uplus \sigma_F) \not\longrightarrow^* \mathbf{abort}\,, \quad (C, \sigma \uplus \sigma_F) \not\longrightarrow^\omega \cdot \tag{5.154}$$

1. for any σ_F, Σ_F, C' and σ'', if $(\langle C\rangle, \sigma \uplus \sigma_F) \longrightarrow (C', \sigma'')$,

   by the operational semantics, we know $C' = \mathbf{skip}$ and

   $$(C, \sigma \uplus \sigma_F) \longrightarrow^* (\mathbf{skip}, \sigma'') \tag{5.155}$$

   by $\mathsf{Locality}(C)$, we know: there exists σ' such that $\sigma'' = \sigma' \uplus \sigma_F$ and $(C, \sigma) \longrightarrow^* (\mathbf{skip}, \sigma')$.

   From $p \Rrightarrow^a p'$, we know one of the following holds:

   (a) either, a is $+$, and there exist w', $\mathbb{D}'$ and Σ' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{D}', \Sigma' \uplus \Sigma_F)$ and $(\sigma, w', \mathbb{D}', \Sigma') \models p'$;

   (b) or, a is 0, and there exist w', $\mathbb{D}'$ and Σ' such that $(\sigma, w', \mathbb{D}', \Sigma') \models p'$, $w' = w$, $\mathbb{D}' = \mathbb{D}$ and $\Sigma' = \Sigma$.

   For either case, from $\models_{\text{SL}} [p']C[q']$ and $(C, \sigma) \longrightarrow^* (\mathbf{skip}, \sigma')$, we know:

   $$(\sigma', w', \mathbb{D}', \Sigma') \models q' \tag{5.156}$$

   From $q' \Rrightarrow^b q$, we know one of the following holds:

   (a) either, b is $+$, and there exist w'', $\mathbb{D}''$ and Σ'' such that $(\mathbb{D}', \Sigma' \uplus \Sigma_F) \longrightarrow^+ (\mathbb{D}'', \Sigma'' \uplus \Sigma_F)$ and $(\sigma', w'', \mathbb{D}'', \Sigma'') \models q$;

   (b) or, b is 0, and there exist w'', $\mathbb{D}''$ and Σ'' such that $(\sigma', w'', \mathbb{D}'', \Sigma'') \models q$, $w'' = w'$, $\mathbb{D}'' = \mathbb{D}'$ and $\Sigma'' = \Sigma'$.

   Since $+ \in \{a, b\}$, we know the following must hold:

   there exist w'', $\mathbb{C}''$ and Σ'' such that $(\mathbb{C}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}'', \Sigma'' \uplus \Sigma_F)$ and $(\sigma', w'', \mathbb{C}'', \Sigma'') \models q$.

   We know:

   $$((\sigma, \Sigma), (\sigma', \Sigma''), \mathbf{true}) \models \llbracket p \rrbracket \propto \llbracket q \rrbracket \tag{5.157}$$

   Since $(\llbracket p \rrbracket \propto \llbracket q \rrbracket) \Rightarrow G * \mathsf{True}$, we know $((\sigma, \Sigma), (\sigma', \Sigma''), \mathbf{true}) \models G^+ * \mathsf{True}$.

   Since $q \Rightarrow I * \mathbf{true}$ and $\mathsf{Sta}(q, [I] * \mathsf{Id})$, by the SKIP and FRAME rules, we know:

   $$[I], G, I \models (\mathbf{skip}, \sigma', (0, 0)) \preceq_{1;w'';q} (\mathbb{C}'', \Sigma'') \tag{5.158}$$

2. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models ([I])^+ * \mathsf{Id}$, we know $\sigma' = \sigma$ and $\Sigma' = \Sigma$.

   By the co-induction hypothesis, we know: $[I], G, I \models (\langle C\rangle, \sigma, (0, 1)) \preceq_{1;w;q} (\mathbb{D}, \Sigma)$.

3. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{false}) \models ([I])^+ * \mathsf{Id}$, we know $\sigma' = \sigma$ and $\Sigma' = \Sigma$.

   By the co-induction hypothesis, we know: $[I], G, I \models (\langle C\rangle, \sigma, (0, 1)) \preceq_{1;w;q} (\mathbb{D}, \Sigma)$.

Thus we are done. $\square$

Lemma 39. If

1. $R, G, I \vdash \{p\}\langle C\rangle\{q\}$;
2. $\vdash_{\text{SL}}$ is sound w.r.t. $\models_{\text{SL}}$;
3. $\mathsf{Locality}(C)$;
4. $(\sigma, w, \mathbb{D}, \Sigma) \models p$,

then for any σ_F, $(C, \sigma \uplus \sigma_F) \not\longrightarrow^* \textbf{abort}$ and $(C, \sigma \uplus \sigma_F) \not\longrightarrow^\omega \cdot$.

Proof: By induction over the derivation of $R, G, I \vdash \{p\}\langle C\rangle\{q\}$. $\square$

The ATOM-R rule.

Lemma 40 (ATOM-R). If

1. $[I], G, I \models \{p\}\langle C\rangle\{q\}$;
2. $\mathsf{Sta}(\{p, q\}, R * \mathsf{Id})$; $I \triangleright \{R, G\}$; $p \vee q \Rightarrow I * \textbf{true}$;
3. for all σ and σ_F, if $(\sigma, \_, \_, \_, \_) \models p$, $(C, \sigma \uplus \sigma_F) \not\longrightarrow^* \textbf{abort}$ and $(C, \sigma \uplus \sigma_F) \not\longrightarrow^\omega \cdot$;

then $R, G, I \models \{p\}\langle C\rangle\{q\}$.

Proof: We want to prove: for all σ, w, $\mathbb{D}$ and Σ, if $(\sigma, w, \mathbb{D}, \Sigma) \models p$, then

$$R, G, I \models (\langle C\rangle, \sigma, (0, |\langle C\rangle|)) \preceq_{\mathsf{height}(\langle C\rangle);w;q} (\mathbb{D}, \Sigma).$$

We know $|\langle C\rangle| = 1$ and can prove $\mathsf{height}(\langle C\rangle) = 1$.

By co-induction. Since $p \Rightarrow I * \textbf{true}$, we know $(\sigma, \Sigma) \models I * \textbf{true}$.

1. for any σ_F, Σ_F, C' and σ'', if $(\langle C\rangle, \sigma \uplus \sigma_F) \longrightarrow (C', \sigma'')$, by the operational semantics, we know $C' = \textbf{skip}$ and

$$(C, \sigma \uplus \sigma_F) \longrightarrow^* (\textbf{skip}, \sigma'') \tag{5.159}$$

From the first premise, we know:

$$[I], G, I \models (\langle C\rangle, \sigma, (0, 1)) \preceq_{1;w;q} (\mathbb{D}, \Sigma).$$

Thus there exists σ' such that $\sigma'' = \sigma' \uplus \sigma_F$ and one of the following holds:

(a) there exist ws', w', $\mathbb{C}'$ and Σ' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma', \Sigma'), \textbf{true}) \models G^+ * \mathsf{True}$ and

$$[I], G, I \models (\textbf{skip}, \sigma', ws') \preceq_{1;w';q} (\mathbb{C}', \Sigma') \tag{5.160}$$

From (5.160), we know one of the following holds:

i. there exist w'', $\mathbb{C}''$ and Σ'' such that $(\mathbb{C}', \Sigma' \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}'', \Sigma'' \uplus \Sigma_F)$, $((\sigma', \Sigma'), (\sigma', \Sigma''), \textbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma', w'', \mathbb{C}'', \Sigma'') \models q$. Thus we know:

$$(\mathbb{C}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}'', \Sigma'' \uplus \Sigma_F) \tag{5.161}$$

$$((\sigma, \Sigma), (\sigma', \Sigma''), \textbf{true}) \models G^+ * \mathsf{True} \tag{5.162}$$

Since $q \Rightarrow I * \textbf{true}$ and $\mathsf{Sta}(q, R * \mathsf{Id})$, by the SKIP and FRAME rules, we know:

$$R, G, I \models (\textbf{skip}, \sigma', (0, 0)) \preceq_{1;w'';q} (\mathbb{C}'', \Sigma'') \tag{5.163}$$

ii. there exists w'' such that $ws' = (w'', 0)$ and $(\sigma', w' + w'', \mathbb{C}', \Sigma') \models q$.
Since $q \Rightarrow I * \mathbf{true}$ and $\mathsf{Sta}(q, R * \mathsf{Id})$, by the SKIP and FRAME rules, we know:

$$R, G, I \models (\mathbf{skip}, \sigma', (0,0)) \preceq_{1;w'+w'';q} (\mathbb{C}', \Sigma') \tag{5.164}$$

(b) there exists ws' such that $ws' <_1 (0,1)$, $((\sigma, \Sigma), (\sigma', \Sigma), \mathbf{false}) \models G^+ * \mathsf{True}$ and

$$[I], G, I \models (\mathbf{skip}, \sigma', ws') \preceq_{1;w;q} (\mathbb{D}, \Sigma) \tag{5.165}$$

From (5.165), we know one of the following holds:

i. there exist w', $\mathbb{C}'$ and Σ' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}', \Sigma' \uplus \Sigma_F)$, $((\sigma', \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma', w', \mathbb{C}', \Sigma') \models q$.
Thus we know:

$$((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True} \tag{5.166}$$

Since $q \Rightarrow I * \mathbf{true}$ and $\mathsf{Sta}(q, R * \mathsf{Id})$, by the SKIP and FRAME rules, we know:

$$R, G, I \models (\mathbf{skip}, \sigma', (0,0)) \preceq_{1;w';q} (\mathbb{C}', \Sigma') \tag{5.167}$$

ii. there exists w' such that $ws' = (w', 0)$ and $(\sigma', w + w', \mathbb{D}, \Sigma) \models q$.
Since $ws' <_1 (0,1)$, we know $w' = 0$.
Since $q \Rightarrow I * \mathbf{true}$ and $\mathsf{Sta}(q, R * \mathsf{Id})$, by the SKIP and FRAME rules, we know:

$$R, G, I \models (\mathbf{skip}, \sigma', (0,0)) \preceq_{1;w;q} (\mathbb{D}, \Sigma) \tag{5.168}$$

2. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models R^+ * \mathsf{Id}$,
Since $(\sigma, w, \mathbb{D}, \Sigma) \models p$ and $\mathsf{Sta}(p, R * \mathsf{Id})$, we know there exists w' such that $(\sigma', w', \mathbb{D}, \Sigma') \models p$.
By the co-induction hypothesis, we know: $R, G, I \models (\langle C \rangle, \sigma', (0,1)) \preceq_{1;w';q} (\mathbb{D}, \Sigma')$.

3. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{false}) \models R^+ * \mathsf{Id}$,
Since $(\sigma, w, \mathbb{D}, \Sigma) \models p$ and $\mathsf{Sta}(p, R * \mathsf{Id})$, we know $(\sigma', w, \mathbb{D}, \Sigma') \models p$.
By the co-induction hypothesis, we know: $R, G, I \models (\langle C \rangle, \sigma', (0,1)) \preceq_{1;w;q} (\mathbb{D}, \Sigma')$.

Thus we are done. □

The A-CONSEQ rule.

Lemma 41 (A-CONSEQ). If

1. $p \overset{G}{\Rightarrow} p'$;
2. $R, G, I \models \{p'\}C\{q'\}$;
3. $q' \overset{G}{\Rightarrow} q$;
4. $\mathsf{Sta}(\{p, q\}, R * \mathsf{Id})$; $I \triangleright \{R, G\}$; $p \vee q \vee p' \vee q' \Rightarrow I * \mathbf{true}$;

then $R, G, I \models \{p\}C\{q\}$.

Proof: We want to prove: for all σ, w, $\mathbb{D}$ and Σ, if $(\sigma, w, \mathbb{D}, \Sigma) \models p$, then

$$R, G, I \models (C, \sigma, (0, |C|)) \preceq_{\mathsf{height}(C);w;q} (\mathbb{D}, \Sigma).$$

Let $\mathcal{H} = \mathsf{height}(C)$.
By co-induction. Since $p \Rightarrow I * \mathbf{true}$, we know $(\sigma, \Sigma) \models I * \mathbf{true}$.

1. for any σ_F, Σ_F, C' and σ'', if $(C, \sigma \uplus \sigma_F) \longrightarrow (C', \sigma'')$,

   from $p \overset{G}{\Rightarrow} p'$, we know one of the following holds:

   (a) either, there exist w', $\mathbb{D}'$ and Σ' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{D}', \Sigma' \uplus \Sigma_F)$ $((\sigma, \Sigma), (\sigma, \Sigma'), \textbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma, w', \mathbb{D}', \Sigma') \models p'$;

   (b) or, there exist w', $\mathbb{D}'$ and Σ' such that $(\sigma, w', \mathbb{D}', \Sigma') \models p'$, $w' = w$, $\mathbb{D}' = \mathbb{D}$ and $\Sigma' = \Sigma$.

   For either case, from $R, G, I \models \{p'\}C\{q'\}$, we know:

   $$R, G, I \models (C, \sigma, (0, |C|)) \preceq_{\mathcal{H};w';q'} (\mathbb{D}', \Sigma') \tag{5.169}$$

   Thus there exists σ' such that $\sigma'' = \sigma' \uplus \sigma_F$ and one of the following holds:

   (a) either, there exist ws', w'', $\mathbb{C}''$ and Σ'' such that $(\mathbb{D}', \Sigma' \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}'', \Sigma'' \uplus \Sigma_F)$, $((\sigma, \Sigma'), (\sigma', \Sigma''), \textbf{true}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', ws') \preceq_{\mathcal{H};w'';q'} (\mathbb{C}'', \Sigma'')$;

   (b) or, there exists ws' such that $ws' <_{\mathcal{H}} (0, |C|)$, $((\sigma, \Sigma'), (\sigma', \Sigma'), \textbf{false}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', ws') \preceq_{\mathcal{H};w';q'} (\mathbb{D}', \Sigma')$.

   Then, we know one of the following holds:

   (a) there exist ws', w'', $\mathbb{C}''$ and Σ'' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}'', \Sigma'' \uplus \Sigma_F)$, $((\sigma, \Sigma), (\sigma', \Sigma''), \textbf{true}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', ws') \preceq_{\mathcal{H};w'';q'} (\mathbb{C}'', \Sigma'')$. By Lemma 42, we know:

   $$R, G, I \models (C', \sigma', ws') \preceq_{\mathcal{H};w'';q} (\mathbb{C}'', \Sigma'') \tag{5.170}$$

   (b) there exists ws' such that $ws' <_{\mathcal{H}} (0, |C|)$, $((\sigma, \Sigma), (\sigma', \Sigma), \textbf{false}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', ws') \preceq_{\mathcal{H};w;q'} (\mathbb{D}, \Sigma)$. By Lemma 42, we know:

   $$R, G, I \models (C', \sigma', ws') \preceq_{\mathcal{H};w;q} (\mathbb{D}, \Sigma) \tag{5.171}$$

2. for any σ_F, Σ_F, e, C' and σ'', if $(C, \sigma \uplus \sigma_F) \xrightarrow{e} (C', \sigma'')$, the proof is similar to the previous case.

3. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \textbf{true}) \models R^+ * \mathsf{Id}$,

   Since $(\sigma, w, \mathbb{D}, \Sigma) \models p$ and $\mathsf{Sta}(p, R * \mathsf{Id})$, we know there exists w' such that $(\sigma', w', \mathbb{D}, \Sigma') \models p$.

   By the co-induction hypothesis, we know: $R, G, I \models (C, \sigma', (0, |C|)) \preceq_{\mathcal{H};w';q} (\mathbb{D}, \Sigma')$.

4. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \textbf{false}) \models R^+ * \mathsf{Id}$,

   Since $(\sigma, w, \mathbb{D}, \Sigma) \models p$ and $\mathsf{Sta}(p, R * \mathsf{Id})$, we know $(\sigma', w, \mathbb{D}, \Sigma') \models p$.

   By the co-induction hypothesis, we know: $R, G, I \models (C, \sigma', (0, |C|)) \preceq_{\mathcal{H};w;q} (\mathbb{D}, \Sigma')$.

5. if $C = \textbf{skip}$, then for any Σ_F,

   from $p \overset{G}{\Rightarrow} p'$, we know one of the following holds:

   (a) either, there exist w', $\mathbb{D}'$ and Σ' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{D}', \Sigma' \uplus \Sigma_F)$ $((\sigma, \Sigma), (\sigma, \Sigma'), \textbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma, w', \mathbb{D}', \Sigma') \models p'$;

   (b) or, there exist w', $\mathbb{D}'$ and Σ' such that $(\sigma, w', \mathbb{D}', \Sigma') \models p'$, $w' = w$, $\mathbb{D}' = \mathbb{D}$ and $\Sigma' = \Sigma$.

   For either case, from $R, G, I \models \{p'\}C\{q'\}$, we know:

   $$R, G, I \models (\textbf{skip}, \sigma, (0, 0)) \preceq_{\mathcal{H};w';q'} (\mathbb{D}', \Sigma') \tag{5.172}$$

   Then one of the following holds:

(a) either, there exist w'', $\mathbb{D}''$ and Σ'' such that $(\mathbb{D}', \Sigma' \uplus \Sigma_F) \longrightarrow^+ (\mathbb{D}'', \Sigma'' \uplus \Sigma_F)$, $((\sigma, \Sigma'), (\sigma, \Sigma''), \mathbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma, w'', \mathbb{D}'', \Sigma'') \models q'$;

(b) or, there exist w'', $\mathbb{D}''$ and Σ'' such that $w'' = w'$, $\mathbb{D}'' = \mathbb{D}'$, $\Sigma'' = \Sigma'$ and $(\sigma, w'', \mathbb{D}'', \Sigma'') \models q'$.

From $q' \overset{G}{\Rightarrow} q$, we know one of the following holds:

(a) either, there exist w''', $\mathbb{D}'''$ and Σ''' such that $(\mathbb{D}'', \Sigma'' \uplus \Sigma_F) \longrightarrow^+ (\mathbb{D}''', \Sigma''' \uplus \Sigma_F)$ $((\sigma, \Sigma''), (\sigma, \Sigma'''), \mathbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma, w''', \mathbb{D}''', \Sigma''') \models q$;

(b) or, there exist w''', $\mathbb{D}'''$ and Σ''' such that $(\sigma, w''', \mathbb{D}''', \Sigma''') \models q$, $w''' = w''$, $\mathbb{D}''' = \mathbb{D}''$ and $\Sigma''' = \Sigma''$.

Thus we get one of the following holds:

(a) either, there exist w''', $\mathbb{C}'''$ and Σ''' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}''', \Sigma''' \uplus \Sigma_F)$ $((\sigma, \Sigma), (\sigma, \Sigma'''), \mathbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma, w''', \mathbb{C}''', \Sigma''') \models q$;

(b) or, $(\sigma, w, \mathbb{D}, \Sigma) \models q$.

6. for any σ_F and Σ_F, if $(C, \sigma \uplus \sigma_F) \longrightarrow \mathbf{abort}$,

   from $p \overset{G}{\Rightarrow} p'$, we know one of the following holds:

   (a) either, there exist w', $\mathbb{D}'$ and Σ' such that $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{D}', \Sigma' \uplus \Sigma_F)$ $((\sigma, \Sigma), (\sigma, \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma, w', \mathbb{D}', \Sigma') \models p'$;

   (b) or, there exist w', $\mathbb{D}'$ and Σ' such that $(\sigma, w', \mathbb{D}', \Sigma') \models p'$, $w' = w$, $\mathbb{D}' = \mathbb{D}$ and $\Sigma' = \Sigma$.

   For either case, from $R, G, I \models \{p'\}C\{q'\}$, we know:

$$R, G, I \models (C, \sigma, (0, |C|)) \preceq_{\mathcal{H}; w'; q'} (\mathbb{D}', \Sigma') \tag{5.173}$$

   Then we know: $(\mathbb{D}', \Sigma' \uplus \Sigma_F) \longrightarrow^+ \mathbf{abort}$. Thus $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ \mathbf{abort}$.

Thus we are done. □

Lemma 42. If

1. $R, G, I \models (C, \sigma, ws) \preceq_{\mathcal{H}; w; q'} (\mathbb{D}, \Sigma)$;
2. $q' \overset{G}{\Rightarrow} q$;
3. $\mathsf{Sta}(q, R * \mathsf{Id})$; $I \triangleright \{R, G\}$; $q \Rightarrow I * \mathbf{true}$;

then $R, G, I \models (C, \sigma, ws) \preceq_{\mathcal{H}; w; q} (\mathbb{D}, \Sigma)$.

Proof: By co-induction. □

The ENV rule.

Lemma 43 (ENV). If $\models_{\mathrm{SL}} [p]c[q]$, c is silent and $\mathsf{Locality}(c)$, then $\mathsf{Emp}, \mathsf{Emp}, \mathsf{emp} \models \{p\}c\{q\}$.

Proof: We want to prove: for all σ, w, $\mathbb{D}$ and Σ, if $(\sigma, w, \mathbb{D}, \Sigma) \models p$, then

$$\mathsf{Emp}, \mathsf{Emp}, \mathsf{emp} \models (c, \sigma, (0, |c|)) \preceq_{\mathsf{height}(c); w; q} (\mathbb{D}, \Sigma).$$

We know $|c| = 1$ and can prove $\mathsf{height}(c) = 1$.

By co-induction. We know $(\sigma, \Sigma) \models \mathsf{emp} * \mathbf{true}$. From $\models_{\text{SL}} [p]c[q]$, we know:

$$(c, \sigma) \not\longrightarrow^* \mathbf{abort}\,, \quad (c, \sigma) \not\longrightarrow^{\omega} \cdot \tag{5.174}$$

By $\mathsf{Locality}(c)$, we know: for any σ_F,

$$(c, \sigma \uplus \sigma_F) \not\longrightarrow^* \mathbf{abort}\,, \quad (c, \sigma \uplus \sigma_F) \not\longrightarrow^{\omega} \cdot \tag{5.175}$$

1. for any σ_F, Σ_F, C' and σ'', if $(c, \sigma \uplus \sigma_F) \longrightarrow (C', \sigma'')$,
   by the operational semantics, we know $C' = \mathbf{skip}$.
   By $\mathsf{Locality}(c)$, we know: there exists σ' such that $\sigma'' = \sigma' \uplus \sigma_F$ and $(c, \sigma) \longrightarrow (\mathbf{skip}, \sigma')$.
   From $\models_{\text{SL}} [p]c[q]$, we know:
   $$(\sigma', w, \mathbb{D}, \Sigma) \models q \tag{5.176}$$
   By the SKIP rule, we know:
   $$\mathsf{Emp}, \mathsf{Emp}, \mathsf{emp} \models (\mathbf{skip}, \sigma', (0,0)) \preceq_{1;w;q} (\mathbb{D}, \Sigma) \tag{5.177}$$
   We know $((\sigma, \Sigma), (\sigma', \Sigma), \mathbf{false}) \models \mathsf{Emp}^+ * \mathsf{True}$.
   Also, we know: $(0,0) <_1 (0,1)$.
2. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models \mathsf{Emp}^+ * \mathsf{Id}$, we know $\sigma' = \sigma$ and $\Sigma' = \Sigma$.
   By the co-induction hypothesis, we know: $\mathsf{Emp}, \mathsf{Emp}, \mathsf{emp} \models (c, \sigma', (0,1)) \preceq_{1;w;q} (\mathbb{D}, \Sigma')$.
3. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{false}) \models \mathsf{Emp}^+ * \mathsf{Id}$, we know $\sigma' = \sigma$ and $\Sigma' = \Sigma$.
   By the co-induction hypothesis, we know: $\mathsf{Emp}, \mathsf{Emp}, \mathsf{emp} \models (c, \sigma', (0,1)) \preceq_{1;w;q} (\mathbb{D}, \Sigma')$.

Thus we are done. $\square$

The FRAME rule.

Lemma 44 (FRAME). If

1. $R, G, I \models \{p\}C\{q\}$;
2. $\mathsf{Sta}(\{p, q\}, R * \mathsf{Id})$; $\mathsf{Sta}(p', (R')^+ * \mathsf{Id})$; $I \triangleright \{R, G\}$; $I' \triangleright \{R', G'\}$; $p \vee q \Rightarrow I * \mathbf{true}$; $p' \Rightarrow I' * \mathbf{true}$; $G^+ \Rightarrow G$;

then $R * R', G * G', I * I' \models \{p * p'\}C\{q * p'\}$.

Proof: We want to prove: for all σ, w, $\mathbb{D}$ and Σ, if $(\sigma, w, \mathbb{D}, \Sigma) \models p * p'$, then

$$R * R', G * G', I * I' \models (C, \sigma, (0, |C|)) \preceq_{\mathsf{height}(C);w;q*p'} (\mathbb{D}, \Sigma).$$

Since $(\sigma, w, \mathbb{D}, \Sigma) \models p * p'$, we know: there exist σ_1, σ_2, w_1, w_2, $\mathbb{D}_1$, $\mathbb{D}_2$, Σ_1 and Σ_2 such that

$$(\sigma_1, w_1, \mathbb{D}_1, \Sigma_1) \models p, \;\; (\sigma_2, w_2, \mathbb{D}_2, \Sigma_2) \models p', \;\; \sigma = \sigma_1 \uplus \sigma_2, \;\; w = w_1 + w_2, \;\; \mathbb{D} = \mathbb{D}_1 \uplus \mathbb{D}_2, \;\; \Sigma = \Sigma_1 \uplus \Sigma_2$$

From the premise, we know: $R, G, I \models (C, \sigma_1, (0, |C|)) \preceq_{\mathsf{height}(C);w_1;q} (\mathbb{D}_1, \Sigma_1)$.
By Lemma 45, we are done. $\square$

Lemma 45. If

1. $R, G, I \models (C, \sigma_1, ws) \preceq_{\mathcal{H};w_1;q} (\mathbb{D}_1, \Sigma_1)$;

2. $\mathsf{Sta}(q, R * \mathsf{Id})$; $\mathsf{Sta}(p', (R')^+ * \mathsf{Id})$; $I \triangleright \{R, G\}$; $I' \triangleright \{R', G'\}$; $q \Rightarrow I * \mathbf{true}$; $p' \Rightarrow I' * \mathbf{true}$; $G^+ \Rightarrow G$;

3. $(\sigma_2, w_2, \mathbb{D}_2, \Sigma_2) \models p'$; $\sigma = \sigma_1 \uplus \sigma_2$; $\mathbb{D} = \mathbb{D}_1 \uplus \mathbb{D}_2$; $\Sigma = \Sigma_1 \uplus \Sigma_2$;

then $R * R', G * G', I * I' \models (C, \sigma, ws) \preceq_{\mathcal{H};w_1+w_2;q*p'} (\mathbb{D}, \Sigma)$.

Proof: By co-induction. From the premises, we know: $(\sigma_1, \Sigma_1) \models I * \mathbf{true}$ and $(\sigma_2, \Sigma_2) \models I' * \mathbf{true}$. Thus we know: $(\sigma, \Sigma) \models I * I' * \mathbf{true}$.

1. for any σ_F, Σ_F, C' and σ'', if $(C, \sigma \uplus \sigma_F) \longrightarrow (C', \sigma'')$,

   from the first premise, we know: there exists σ_1' such that $\sigma'' = \sigma_1' \uplus \sigma_2 \uplus \sigma_F$, and one of the following holds:

   (a) there exist ws', w_1', $\mathbb{C}_1'$ and Σ_1' such that $(\mathbb{D}_1, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}_1', \Sigma_1' \uplus \Sigma_2 \uplus \Sigma_F)$, $((\sigma_1, \Sigma_1), (\sigma_1', \Sigma_1'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma_1', ws') \preceq_{\mathcal{H};w_1';q} (\mathbb{C}_1', \Sigma_1')$.
   Since $(\sigma_2, \Sigma_2) \models I' * \mathbf{true}$ and $I' \triangleright G'$, we know:

   $$((\sigma_2, \Sigma_2), (\sigma_2, \Sigma_2), \mathbf{true}) \models G' * \mathsf{True}.$$

   Since $G^+ \Rightarrow G$, we know:

   $$((\sigma_1 \uplus \sigma_2, \Sigma_1 \uplus \Sigma_2), (\sigma_1' \uplus \sigma_2, \Sigma_1' \uplus \Sigma_2), \mathbf{true}) \models (G * G')^+ * \mathsf{True}.$$

   Since $\mathbb{D} = \mathbb{D}_1 \uplus \mathbb{D}_2$, we know $\mathbb{D}_2 = \bullet$ and $\mathbb{D} = \mathbb{D}_1$. Let $\mathbb{D}' = \mathbb{C}_1' \uplus \mathbb{D}_2 = \mathbb{C}_1'$.
   By the co-induction hypothesis, we know

   $$R * R', G * G', I * I' \models (C', \sigma_1' \uplus \sigma_2, ws') \preceq_{\mathcal{H};w_1'+w_2;q*p'} (\mathbb{D}', \Sigma_1' \uplus \Sigma_2).$$

   (b) there exists ws' such that $ws' <_{\mathcal{H}} ws$,
   $((\sigma_1, \Sigma_1), (\sigma_1', \Sigma_1), \mathbf{false}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma_1', ws') \preceq_{\mathcal{H};w_1;q} (\mathbb{D}_1, \Sigma_1)$.
   Since $(\sigma_2, \Sigma_2) \models I' * \mathbf{true}$ and $I' \triangleright G'$, we know:

   $$((\sigma_2, \Sigma_2), (\sigma_2, \Sigma_2), \mathbf{false}) \models G' * \mathsf{True}.$$

   Since $G^+ \Rightarrow G$, we know:

   $$((\sigma_1 \uplus \sigma_2, \Sigma_1 \uplus \Sigma_2), (\sigma_1' \uplus \sigma_2, \Sigma_1 \uplus \Sigma_2), \mathbf{false}) \models (G * G')^+ * \mathsf{True}.$$

   By the co-induction hypothesis, we know

   $$R * R', G * G', I * I' \models (C', \sigma_1' \uplus \sigma_2, ws') \preceq_{\mathcal{H};w_1+w_2;q*p'} (\mathbb{D}, \Sigma_1 \uplus \Sigma_2).$$

2. for any σ_F, Σ_F, e, C' and σ'', if $(C, \sigma \uplus \sigma_F) \xrightarrow{e} (C', \sigma'')$, the proof is similar to the previous case.

3. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models (R * R')^+ * \mathsf{Id}$,

   since $I \triangleright R$, $I' \triangleright R'$, $(\sigma_1, \Sigma_1) \models I * \mathbf{true}$ and $(\sigma_2, \Sigma_2) \models I' * \mathbf{true}$, we know: there exist σ_1', σ_2', Σ_1' and Σ_2' such that $\sigma' = \sigma_1' \uplus \sigma_2'$, $\Sigma' = \Sigma_1' \uplus \Sigma_2'$,

   $$((\sigma_1, \Sigma_1), (\sigma_1', \Sigma_1'), \mathbf{true}) \models R^+ * \mathsf{Id}, \quad ((\sigma_2, \Sigma_2), (\sigma_2', \Sigma_2'), \mathbf{true}) \models (R')^+ * \mathsf{Id}$$

   From the first premise, we know there exist ws' and w_1' such that

   $$R, G, I \models (C, \sigma_1', ws') \preceq_{\mathcal{H};w_1';q} (\mathbb{D}_1, \Sigma_1').$$

   Since $(\sigma_2, w_2, \mathbb{D}_2, \Sigma_2) \models p'$ and $\mathsf{Sta}(p', (R')^+ * \mathsf{Id})$, we know: there exists w_2' such that

   $$(\sigma_2', w_2', \mathbb{D}_2, \Sigma_2') \models p'.$$

   By the co-induction hypothesis, we know:

$$R * R', G * G', I * I' \models (C, \sigma', ws') \preceq_{\mathcal{H};w_1'+w_2';q*p'} (\mathbb{D}, \Sigma').$$

4. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \textbf{false}) \models (R * R')^+ * \mathsf{Id}$,

   since $I \triangleright R$, $I' \triangleright R'$, $(\sigma_1, \Sigma_1) \models I * \textbf{true}$ and $(\sigma_2, \Sigma_2) \models I' * \textbf{true}$, we know: there exist σ_1', σ_2', Σ_1' and Σ_2' such that $\sigma' = \sigma_1' \uplus \sigma_2'$, $\Sigma' = \Sigma_1' \uplus \Sigma_2'$,

   $$((\sigma_1, \Sigma_1), (\sigma_1', \Sigma_1'), \textbf{false}) \models R^+ * \mathsf{Id}, \quad ((\sigma_2, \Sigma_2), (\sigma_2', \Sigma_2'), \textbf{false}) \models (R')^+ * \mathsf{Id}$$

   From the first premise, we know

   $$R, G, I \models (C, \sigma_1', ws) \preceq_{\mathcal{H};w_1;q} (\mathbb{D}_1, \Sigma_1').$$

   Since $(\sigma_2, w_2, \mathbb{D}_2, \Sigma_2) \models p'$ and $\mathsf{Sta}(p', (R')^+ * \mathsf{Id})$, we know:

   $$(\sigma_2', w_2, \mathbb{D}_2, \Sigma_2') \models p'.$$

   By the co-induction hypothesis, we know:

   $$R * R', G * G', I * I' \models (C, \sigma', ws) \preceq_{\mathcal{H};w_1+w_2;q*p'} (\mathbb{D}, \Sigma').$$

5. if $C = \textbf{skip}$, then for any Σ_F, from the first premise we know one of the following holds:

   (a) there exist w_1', $\mathbb{C}_1'$ and Σ_1' such that $(\mathbb{D}_1, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}_1', \Sigma_1' \uplus \Sigma_2 \uplus \Sigma_F)$, $((\sigma_1, \Sigma_1), (\sigma_1, \Sigma_1'), \textbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma_1, w_1', \mathbb{C}_1', \Sigma_1') \models q$.
   Since $(\sigma_2, \Sigma_2) \models I' * \textbf{true}$ and $I' \triangleright G'$, we know:

   $$((\sigma_2, \Sigma_2), (\sigma_2, \Sigma_2), \textbf{true}) \models G' * \mathsf{True}.$$

   Since $G^+ \Rightarrow G$, we know:

   $$((\sigma_1 \uplus \sigma_2, \Sigma_1 \uplus \Sigma_2), (\sigma_1 \uplus \sigma_2, \Sigma_1' \uplus \Sigma_2), \textbf{true}) \models (G * G')^+ * \mathsf{True}.$$

   Since $\mathbb{D} = \mathbb{D}_1 \uplus \mathbb{D}_2$, we know $\mathbb{D}_2 = \bullet$ and $\mathbb{D} = \mathbb{D}_1$. Thus $\mathbb{C}_1' \uplus \mathbb{D}_2 = \mathbb{C}_1'$.
   Since $(\sigma_1, w_1', \mathbb{C}_1', \Sigma_1') \models q$, we get:

   $$(\sigma, w_1' + w_2, \mathbb{C}_1' \uplus \mathbb{D}_2, \Sigma_1' \uplus \Sigma_2) \models q * p'.$$

   (b) there exists w_1' such that $ws = (w_1', 0)$ and $(\sigma_1, w_1 + w_1', \mathbb{D}_1, \Sigma_1) \models q$.
   Since $(\sigma_2, w_2, \mathbb{D}_2, \Sigma_2) \models p'$, we have

   $$(\sigma, w_1 + w_2 + w_1', \mathbb{D}, \Sigma) \models q * p'.$$

6. for any σ_F and Σ_F, if $(C, \sigma \uplus \sigma_F) \longrightarrow \textbf{abort}$,

   from the first premise, we know: $(\mathbb{D}_1, \Sigma_1 \uplus \Sigma_2 \uplus \Sigma_F) \longrightarrow^+ \textbf{abort}$. Thus $\mathbb{D}_2 = \bullet$ and $\mathbb{D} = \mathbb{D}_1$. Thus $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ \textbf{abort}$.

Thus we are done. $\square$

The FR-CONJ rule.

Lemma 46 (FR-CONJ). If

1. $R, G, I \models \{p\}C\{q\}$;
2. $\mathsf{Sta}(\{p, q\}, R * \mathsf{Id})$; $\mathsf{Sta}(p', R^+ * \mathsf{Id})$; $\mathsf{Sta}(p', G * \mathsf{True})$; $I \triangleright \{R, G\}$; $p \vee q \Rightarrow I * \mathbf{true}$;

then $R, G, I \models \{p \oslash p'\}C\{q \oslash p'\}$.

Proof: We want to prove: for all σ, w, $\mathbb{D}$ and Σ, if $(\sigma, w, \mathbb{D}, \Sigma) \models p \oslash p'$, then

$$R, G, I \models (C, \sigma, (0, |C|)) \preceq_{\mathsf{height}(C); w; q \oslash p'} (\mathbb{D}, \Sigma).$$

Since $(\sigma, w, \mathbb{D}, \Sigma) \models p \oslash p'$, we know: there exist w_1, w_2, $\mathbb{D}_1$ and $\mathbb{D}_2$ such that

$$(\sigma, w_1, \mathbb{D}_1, \Sigma) \models p, \;\; (\sigma, w_2, \mathbb{D}_2, \Sigma) \models p', \;\; w = w_1 + w_2, \;\; \mathbb{D} = \mathbb{D}_1 \uplus \mathbb{D}_2$$

From the premise, we know: $R, G, I \models (C, \sigma, (0, |C|)) \preceq_{\mathsf{height}(C); w_1; q} (\mathbb{D}_1, \Sigma)$.
By Lemma 47, we are done. □

Lemma 47. If

1. $R, G, I \models (C, \sigma, ws_1) \preceq_{\mathcal{H}; w_1; q} (\mathbb{D}_1, \Sigma)$;
2. $\mathsf{Sta}(q, R * \mathsf{Id})$; $\mathsf{Sta}(p', R^+ * \mathsf{Id})$; $\mathsf{Sta}(p', G * \mathsf{True})$; $I \triangleright \{R, G\}$; $q \Rightarrow I * \mathbf{true}$;
3. $(\sigma, w_2, \mathbb{D}_2, \Sigma) \models p'$; $w = w_1 + w_2$; $\mathbb{D} = \mathbb{D}_1 \uplus \mathbb{D}_2$;

then $R, G, I \models (C, \sigma, ws_1) \preceq_{\mathcal{H}; w; q \oslash p'} (\mathbb{D}, \Sigma)$.

Proof: By co-induction. From the premises, we know: $(\sigma, \Sigma) \models I * \mathbf{true}$.

1. for any σ_F, Σ_F, C' and σ'', if $(C, \sigma \uplus \sigma_F) \longrightarrow (C', \sigma'')$,
   from the first premise, we know: there exists σ' such that $\sigma'' = \sigma' \uplus \sigma_F$, and one of the following holds:

   (a) there exist ws_1', $\mathbb{C}_1'$ and Σ' such that $(\mathbb{D}_1, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}_1', \Sigma' \uplus \Sigma_F)$,
   $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', ws_1') \preceq_{\mathcal{H}; w_1; q} (\mathbb{C}_1', \Sigma')$.
   Since $\mathsf{Sta}(p', G * \mathsf{True})$, we know

   $$\mathsf{Sta}(p', G^+ * \mathsf{True})$$

   Since $(\sigma, w_2, \mathbb{D}_2, \Sigma) \models p'$, we know there exists w_2' such that

   $$(\sigma', w_2', \mathbb{D}_2, \Sigma') \models p'$$

   Since $\mathbb{D} = \mathbb{D}_1 \uplus \mathbb{D}_2$, we know $\mathbb{D}_2 = \bullet$ and $\mathbb{D} = \mathbb{D}_1$. Let $\mathbb{D}' = \mathbb{C}_1' \uplus \mathbb{D}_2 = \mathbb{C}_1'$ and $w' = w_1 + w_2'$.
   By the co-induction hypothesis, we know

   $$R, G, I \models (C', \sigma', ws_1') \preceq_{\mathcal{H}; w'; q \oslash p'} (\mathbb{D}', \Sigma').$$

   (b) there exists ws_1' such that $ws_1' <_{\mathcal{H}} ws_1$,
   $((\sigma, \Sigma), (\sigma', \Sigma), \mathbf{false}) \models G^+ * \mathsf{True}$ and $R, G, I \models (C', \sigma', ws_1') \preceq_{\mathcal{H}; w_1; q} (\mathbb{D}_1, \Sigma)$.
   Since $(\sigma, w_2, \mathbb{D}_2, \Sigma) \models p'$ and $\mathsf{Sta}(p', G * \mathsf{True})$, we know

   $$(\sigma', w_2, \mathbb{D}_2, \Sigma) \models p'$$

   By the co-induction hypothesis, we know

   $$R, G, I \models (C', \sigma', ws_1') \preceq_{\mathcal{H}; w; q \oslash p'} (\mathbb{D}, \Sigma).$$

2. for any σ_F, Σ_F, e, C' and σ'', if $(C, \sigma \uplus \sigma_F) \xrightarrow{e} (C', \sigma'')$, the proof is similar to the previous case.

3. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{true}) \models R^+ * \mathsf{Id}$,
   from the first premise, we know there exists ws_1' such that

$$R, G, I \models (C, \sigma', ws_1') \preceq_{\mathcal{H};w_1;q} (\mathbb{D}_1, \Sigma').$$

   Since $(\sigma, w_2, \mathbb{D}_2, \Sigma) \models p'$ and $\mathsf{Sta}(p', R^+ * \mathsf{Id})$, we know: there exists w_2' such that

$$(\sigma', w_2', \mathbb{D}_2, \Sigma') \models p'.$$

   By the co-induction hypothesis, we know: let $w' = w_1 + w_2'$,

$$R, G, I \models (C, \sigma', ws_1') \preceq_{\mathcal{H};w';q \oslash p'} (\mathbb{D}, \Sigma').$$

4. for any σ' and Σ', if $((\sigma, \Sigma), (\sigma', \Sigma'), \mathbf{false}) \models R^+ * \mathsf{Id}$,
   from the first premise, we know

$$R, G, I \models (C, \sigma', ws_1) \preceq_{\mathcal{H};w_1;q} (\mathbb{D}_1, \Sigma').$$

   Since $(\sigma, w_2, \mathbb{D}_2, \Sigma) \models p'$ and $\mathsf{Sta}(p', R^+ * \mathsf{Id})$, we know:

$$(\sigma', w_2, \mathbb{D}_2, \Sigma') \models p'.$$

   By the co-induction hypothesis, we know:

$$R, G, I \models (C, \sigma', ws_1) \preceq_{\mathcal{H};w;q \oslash p'} (\mathbb{D}, \Sigma').$$

5. if $C = \mathbf{skip}$, then for any Σ_F, from the first premise we know one of the following holds:

   (a) there exist w_1', $\mathbb{C}_1'$ and Σ' such that $(\mathbb{D}_1, \Sigma \uplus \Sigma_F) \longrightarrow^+ (\mathbb{C}_1', \Sigma' \uplus \Sigma_F)$,
   $((\sigma, \Sigma), (\sigma, \Sigma'), \mathbf{true}) \models G^+ * \mathsf{True}$ and $(\sigma, w_1', \mathbb{C}_1', \Sigma') \models q$.
   Since $(\sigma, w_2, \mathbb{D}_2, \Sigma) \models p'$ and $\mathsf{Sta}(p', G * \mathsf{True})$, we know there exists w_2' such that

$$(\sigma, w_2', \mathbb{D}_2, \Sigma') \models p'$$

   Since $\mathbb{D} = \mathbb{D}_1 \uplus \mathbb{D}_2$, we know $\mathbb{D}_2 = \bullet$ and $\mathbb{D} = \mathbb{D}_1$. Thus $\mathbb{C}_1' \uplus \mathbb{D}_2 = \mathbb{C}_1'$. Thus we get:

$$(\sigma, w_1' + w_2', \mathbb{C}_1' \uplus \mathbb{D}_2, \Sigma') \models q \oslash p'.$$

   (b) there exists w_1' such that $ws_1 = (w_1', 0)$ and $(\sigma, w_1 + w_1', \mathbb{D}_1, \Sigma) \models q$.
   Since $(\sigma, w_2, \mathbb{D}_2, \Sigma) \models p'$, we have

$$(\sigma, w_1 + w_2 + w_1', \mathbb{D}, \Sigma) \models q \oslash p'.$$

6. for any σ_F and Σ_F, if $(C, \sigma \uplus \sigma_F) \longrightarrow \mathbf{abort}$,
   from the first premise, we know: $(\mathbb{D}_1, \Sigma \uplus \Sigma_F) \longrightarrow^+ \mathbf{abort}$. Thus $\mathbb{D}_2 = \bullet$ and $\mathbb{D} = \mathbb{D}_1$. Thus $(\mathbb{D}, \Sigma \uplus \Sigma_F) \longrightarrow^+ \mathbf{abort}$.

Thus we are done. $\square$

5.5 Derivation of WHILE-TERM Rule

Lemma 48 (WHILE-TERM Derivable). If

1. $R, G, I \vdash \{p \wedge B \wedge (E = \alpha)\}C\{p \wedge (E < \alpha)\}$;
2. $p \wedge B \Rightarrow E > 0$;
3. $p \Rightarrow ((B = B) \wedge (E = E)) * I$;
4. $G^+ \Rightarrow G$;
5. α is a fresh logical variable;

then $R, G, I \vdash \{\lfloor p \rfloor_{\mathsf{w}}\}\mathbf{while}\ (B)\ C\{\lfloor p \rfloor_{\mathsf{w}} \wedge \neg B\}$.

Proof: Take a fresh logical variable β and by applying the CONSEQ rule to the premise 1, we get:

$$R, G, I \vdash \{\exists \beta.\ p \wedge (E = \beta) \wedge B \wedge (E = \alpha)\}C\{\exists \beta.\ p \wedge (E = \beta) \wedge (E < \alpha)\} \tag{5.178}$$

From $p \wedge B \Rightarrow E > 0$, we know

$$p \wedge B \wedge (E = \alpha) \ \Rightarrow\ \alpha > 0 \tag{5.179}$$

Since $G^+ \Rightarrow G$, $\mathsf{Sta}(\mathsf{wf}(\alpha) \wedge \mathsf{emp}, \mathsf{Emp} * \mathsf{Id})$, $\mathsf{emp} \rhd \mathsf{Emp}$ and $(\mathsf{wf}(\alpha) \wedge \mathsf{emp}) \Rightarrow \mathsf{emp} * \mathbf{true}$, we can apply the FRAME rule to (5.178) and get

$$R, G, I \vdash \{(\exists \beta.\ p \wedge (E = \beta) \wedge B \wedge (E = \alpha)) * (\mathsf{wf}(\alpha) \wedge \mathsf{emp})\}C\{(\exists \beta.\ p \wedge (E = \beta) \wedge (E < \alpha)) * (\mathsf{wf}(\alpha) \wedge \mathsf{emp})\} \tag{5.180}$$

We reduce (5.180) as follows:

$$R, G, I \vdash \{\exists \beta.\ (p \wedge (E = \beta)) * (\mathsf{wf}(\alpha) \wedge \mathsf{emp}) \wedge B \wedge (E = \alpha)\}C\{\exists \beta.\ (p \wedge (E = \beta)) * (\mathsf{wf}(\alpha) \wedge \mathsf{emp}) \wedge (E < \alpha)\} \tag{5.181}$$

$$R, G, I \vdash \{\exists \beta.\ (p \wedge (E = \beta)) * (\mathsf{wf}(\beta) \wedge \mathsf{emp})) \wedge B \wedge (E = \alpha)\}C\{\exists \beta.\ (p \wedge (E = \beta)) * (\mathsf{wf}(\beta + 1) \wedge \mathsf{emp})) \wedge (E < \alpha)\} \tag{5.182}$$

Since $(\mathsf{wf}(\beta + 1) \wedge \mathsf{emp}) \Rightarrow (\mathsf{wf}(\beta) \wedge \mathsf{emp}) * (\mathsf{wf}(1) \wedge \mathsf{emp})$, we let

$$p_0 \ \stackrel{\mathrm{def}}{=}\ (\exists \beta.\ (p \wedge (E = \beta)) * (\mathsf{wf}(\beta) \wedge \mathsf{emp})) \tag{5.183}$$

then (5.182) can be written as:

$$R, G, I \vdash \{p_0 \wedge B \wedge (E = \alpha)\}C\{(p_0 * (\mathsf{wf}(1) \wedge \mathsf{emp})) \wedge (E < \alpha)\} \tag{5.184}$$

By the EXISTS rule and α is not free in R, G and I, we get:

$$R, G, I \vdash \{\exists \alpha.\ p_0 \wedge B \wedge (E = \alpha)\}C\{\exists \alpha.\ (p_0 * (\mathsf{wf}(1) \wedge \mathsf{emp})) \wedge (E < \alpha)\} \tag{5.185}$$

Since α is not free in p, B and E, we know

$$(p_0 \wedge B) \ \Rightarrow\ (\exists \alpha.\ p_0 \wedge B \wedge (E = \alpha)) \tag{5.186}$$

and

$$(\exists \alpha.\ (p_0 * (\mathsf{wf}(1) \wedge \mathsf{emp})) \wedge (E < \alpha)) \ \Rightarrow\ (p_0 * (\mathsf{wf}(1) \wedge \mathsf{emp})) \tag{5.187}$$

Thus by applying CONSEQ rule to (5.185), we get:

$$R, G, I \vdash \{p_0 \wedge B\}C\{p_0 * (\mathsf{wf}(1) \wedge \mathsf{emp})\} \tag{5.188}$$

From $p \Rightarrow (B = B) * I$ and $p_0 * (\mathsf{wf}(1) \wedge \mathsf{emp}) \wedge B \Rightarrow (p_0 \wedge B) * (\mathsf{wf}(1) \wedge \mathsf{emp})$, by applying the WHILE rule and the HIDE-W rule, we get:

$$R, G, I \vdash \{\lfloor p_0 * (\mathsf{wf}(1) \wedge \mathsf{emp}) \rfloor_{\mathsf{w}}\}\mathbf{while}\ (B)\ C\{\lfloor p_0 * (\mathsf{wf}(1) \wedge \mathsf{emp}) \rfloor_{\mathsf{w}} \wedge \neg B\} \tag{5.189}$$

It can be reduced to:

$$R, G, I \vdash \{\exists\beta.\ \lfloor p \rfloor_{\mathsf{w}} \wedge (E = \beta)\}\textbf{while}\ (B)\ C\{\exists\beta.\ \lfloor p \rfloor_{\mathsf{w}} \wedge (E = \beta) \wedge \neg B\} \tag{5.190}$$

Since $p \Rightarrow (E = E) * I$, we know

$$R, G, I \vdash \{\lfloor p \rfloor_{\mathsf{w}}\}\textbf{while}\ (B)\ C\{\lfloor p \rfloor_{\mathsf{w}} \wedge \neg B\} \tag{5.191}$$

Thus we are done. $\square$

References

[1] Simon Doherty, Lindsay Groves, Victor Luchangco, and Mark Moir. Formal verification of a practical lock-free queue algorithm. In *FORTE'04*.

[2] Xinyu Feng. Local rely-guarantee reasoning. In *POPL'09*.

[3] Maurice Herlihy and Nir Shavit. *The Art of Multiprocessor Programming*.

[4] Maurice Herlihy and Jeannette Wing. Linearizability: a correctness condition for concurrent objects. *ACM Trans. Program. Lang. Syst.*, 12(3):463–492, 1990.

[5] Jan Hoffmann, Michael Marmar, and Zhong Shao. Quantitative reasoning for proving lock-freedom. In *LICS*, pages 124–133, 2013.

[6] Hongjin Liang and Xinyu Feng. Modular verification of linearizability with non-fixed linearization points. In *PLDI*, pages 459–470, 2013.

[7] Hongjin Liang, Xinyu Feng, and Ming Fu. A rely-guarantee-based simulation for verifying concurrent program transformations. In *POPL*, 2012.

[8] Maged M. Michael and Michael L. Scott. Simple, fast, and practical non-blocking and blocking concurrent queue algorithms. In *PODC'96*.

[9] William N. Scherer III, Doug Lea, and Michael L. Scott. Scalable synchronous queues. In *PPoPP*, pages 147–156, 2006.

[10] Ketil Stølen. A method for the development of totally correct shared-state parallel programs. In *CONCUR*, pages 510–525, 1991.

[11] Aaron Turon and Mitchell Wand. A separation logic for refining concurrent objects. In *POPL'11*.

[12] Viktor Vafeiadis. Modular fine-grained concurrency verification. Thesis.

[13] Viktor Vafeiadis. Concurrent separation logic and operational semantics. In *MFPS*, 2011.