11 Appendix: Logic Soundness Proofs

In this section, we prove our logic is sound via the simulation in Definition 5. We prove Lemma 1 in Sec 11.1 and Lemma 2 in Sec 11.2, and get the final soundness theorem (Theorem 1) directly from them (in Sec 11.3).

11.1 Proofs for Lemma 1(Simulation Implies Contextual Refinement)

We define two auxiliary simulations:

1. A thread-local simulation between concrete and abstract client code, including method calls (Definition 7). We will show:
   (a) it is implied by the simulation for method in Defintion 5 when the two levels are method calls (the first in Figure 32);
   (b) it trivially holds for client commands (the second, third and fourth lines in Figure 32);
   (c) it is also compositional (other rules in Figure 32) and could ensure the following whole program simulation.
2. A whole-program simulation between concrete and abstract levels (Definition 6 below). We will show it implies a subset relation between observable behaviors of the two levels (Lemma 15), thus could ensure contextual refinement.

Intuitively, Lemma 1 (method simulations imply contextual refinement) can be proved by following steps:

1. Those method simulations can compose a global, whole program simulation. Which can be further devided into:
   - Method simulations can compose thread simulations (the composing rules are in Fig 32, except the last rule);
   - Thread simulations can compose a whole program simulations (The last rule Fig 32, proved by Lemma 13).

   In Fig 32, task simulations are composed by method simulations syntactically: the first rule is about how to compose method call (proved by Lemma 3), the second rule is about how to compose other primitive commands (Lemma 5), the third rule is about skip (Lemma 6), the fourth rule is about atomic (Lemma 7), the fifth rule is about sequential commands (Lemma 8), the sixth rule is about if statements (Lemma 11), the seventh rule is about while statements (Lemma 12).

   All statements appearing in thread programs (defined in Fig 3, all possible statements have been considered) can be composed of method simulations by recursively applying these rules.
2. The program simulation can imply contextual refinement.(Lemma 15)

$$\frac{(x, C) \preceq^{R;G;I}_{trp;trq} \mathbb{s} \quad \Pi(f) = (x, C) \quad \Gamma(f) = (\mathbb{s}, trp, trq) \quad I \rhd \{R, G\} \quad x \notin dom(R) \quad x \notin dom(G)}{(y := f(E), \Pi) \preceq^{t}_{R;G;I} (y := f(E), \Gamma)}$$

$$\frac{C \in \{x := E, x := [E], [E] := E, \mathsf{print}\ E\}}{(C, \Pi) \preceq^{t}_{R;G;I} (C, \Gamma)}$$

$$\frac{}{(\mathbf{skip}, \Pi) \preceq^{t}_{R;G;I} (\mathbf{skip}, \Gamma)}$$

$$\frac{}{(\langle C \rangle, \Pi) \preceq^{t}_{R;G;I} (\langle C \rangle, \Gamma)}$$

$$\frac{(C_1, \Pi) \preceq^{t}_{R;G;I} (\mathbb{C}_1, \Gamma) \quad (C_2, \Pi) \preceq^{t}_{R;G;I} (\mathbb{C}_2, \Gamma)}{(C_1; C_2, \Pi) \preceq^{t}_{R;G;I} (\mathbb{C}_1; \mathbb{C}_2, \Gamma)}$$

$$\frac{(C_1, \Pi) \preceq^{t}_{R;G;I} (\mathbb{C}_1, \Gamma) \quad (C_2, \Pi) \preceq^{t}_{R;G;I} (\mathbb{C}_2, \Gamma)}{(\mathbf{if}\ (B)\ C_1\ \mathbf{else}\ C_2, \Pi) \preceq^{t}_{R;G;I} (\mathbf{if}\ (B)\ \mathbb{C}_1\ \mathbf{else}\ \mathbb{C}_2, \Gamma)}$$

$$\frac{(C, \Pi) \preceq^{t}_{R;G;I} (\mathbb{C}, \Gamma)}{(\mathbf{while}\ (B)\ \{C\}, \Pi) \preceq^{t}_{R;G;I} (\mathbf{while}\ (B)\ \{\mathbb{C}\}, \Gamma)}$$

$$\frac{\begin{array}{c}(C_i, \Pi) \ \preceq^{i}_{R_i;G_i;I} \ (\mathbb{C}_i, \Gamma) \\ (\forall j.j \neq i \implies G_i \Rightarrow R_j) \\ I \ \rhd \ \{R_i, G_i\} \quad \varphi \ \Rightarrow \ I\end{array}}{\mathbf{let}\ \Pi\ \mathbf{in}\ C_1 \parallel \ldots \parallel C_n \preceq_{\varphi} \mathbf{with}\ \Gamma\ \mathbf{do}\ \mathbb{C}_1 \parallel \ldots \parallel \mathbb{C}_n}$$

Fig.32. Compositionality Rules for Simulation

$$\begin{array}{ll}
\lfloor \mu \rfloor_f & \stackrel{\mathsf{def}}{=} \mathbf{fexec}(\mathbb{s}, n), \mathsf{if}\ \mu = (\mathbb{s}, n) \\
\sigma_1 \perp \ldots \perp \sigma_n & \stackrel{\mathsf{def}}{=} \forall i, j.\ 0 < i < j \leq n \implies \sigma_i \perp \sigma_j \\
\mathsf{getJ}(ak) & \stackrel{\mathsf{def}}{=} \begin{cases} J & \mathsf{if}\ ak = ((_, J), _, _, _) \\ \emptyset & \mathsf{if}\ ak = \circ \end{cases} \\
\mathsf{wellFormedTokens}(J, \mathbb{K}) & \mathsf{iff}\ \exists J_0.J_0 = J \uplus (\biguplus_i \mathsf{getJ}(\mathbb{K}(i))) \\
x \notin dom(R) & \mathsf{iff}\ \forall \sigma, \sigma'. \\
 & \quad (((\sigma, _), _), ((\sigma', _), _)) \models R \implies \\
 & \quad x \notin dom(\sigma) \wedge x \notin dom(\sigma')
\end{array}$$

Fig.33. Auxiliary Definitions

The detailed proof is:

Proof. To show $\Pi \sqsubseteq_{\varphi} \Gamma$, we want to prove: for any $n, C_1, \ldots, C_n, \sigma_c, K_c, \Sigma$ and σ_o, if $(\sigma_o, \Sigma) \in \varphi$ and $\neg\mathsf{Abort}(\mathbf{with}\ \Gamma\ \mathbf{do}\ C_1 \parallel \ldots \parallel C_n, \mathsf{init}(\sigma_c, K_c, \Sigma))$, then

$$\mathcal{O}[\![\mathbf{let}\ \Pi\ \mathbf{in}\ C_1 \parallel \ldots \parallel C_n, \mathsf{init}(\sigma_c, K_c, \sigma_o)]\!] \subseteq$$

$$\mathcal{O}[\![\mathbf{with}\ \Gamma\ \mathbf{do}\ C_1 \parallel \ldots \parallel C_n, \mathsf{init}(\sigma_c, K_c, \Sigma)]\!]$$

By Lemma 15(the whole program simulation implies a behavior-subset relation), we only need to prove:

$$(\mathbf{let}\ \Pi\ \mathbf{in}\ C_1 \parallel \ldots \parallel C_n) \preceq_{\varphi} (\mathbf{with}\ \Gamma\ \mathbf{do}\ C_1 \parallel \ldots \parallel C_n)$$

Also, for any i, since premise 2, we know $(\forall j.j \neq i \implies G_i \Rightarrow R_j)$;

since premise 2, we know $I \rhd \{R_i, G_i\}$;

since premise 3, we know $\varphi \Rightarrow I$.

Thus we can apply the parrallel compositionality of the simulation for thread (the last rule in Fig 32), then we only need to show: for any i,

$$(C_i, \Pi) \preceq^i_{R_i;G_i;I} (C_i, \Gamma)$$

It is proved by induction over the structure of C_i. For the base case, we have the first to fourth rules in Figure 32; for the inductive step, we have other compositionality rules in Figure 32. □

11.1.1 Definitions of the Simulation for Thread and Program

We first define some notations in Figure 33.

Definition 6 (Simulation for Program)**.** $W \preceq_\varphi \mathbb{W}$ *iff*

$$\begin{array}{l}\forall \sigma, \sigma_o, \Sigma, \sigma_c, K_c. \\ \quad (\sigma_o, \Sigma) \in \varphi \implies \\ \quad \neg \mathsf{Abort}(\mathbb{W}, \mathsf{init}(\sigma_c, K_c, \Sigma)) \implies \\ \quad (W, \mathsf{init}(\sigma_c, K_c, \sigma_o)) \preceq (\mathbb{W}, \mathsf{init}(\sigma_c, K_c, \Sigma))\end{array}$$

Whenever $(W, S) \preceq (\mathbb{W}, \mathbb{S})$, then

1. if $(W, S) \overset{e}{\longmapsto} (W', S')$, then there exist $\mathbb{W}', \mathbb{S}'$ and ξ, such that $(\mathbb{W}, \mathbb{S}) \overset{\xi}{\circ\!\!\longrightarrow}{}^* (\mathbb{W}', \mathbb{S}')$, $\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi)$, and $(W', S') \preceq (\mathbb{W}', \mathbb{S}')$;
2. if $W = \textbf{skip}$, then $\mathbb{W} = \textbf{skip}$;
3. if $(W, S) \overset{e}{\longmapsto} \textbf{abort}$, then there exist ξ, such that $(\mathbb{W}, \mathbb{S}) \overset{\xi}{\circ\!\!\longrightarrow}{}^* \textbf{abort}$ and $\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi)$.

Definition 7 (Simulation for Thread)**.** $(C, \Pi) \preceq^t_{R;G;I} (\mathbb{C}, \Gamma)$ *iff*

$$\begin{array}{l}\forall \sigma_c, \sigma_{cl}, \sigma_o, \Sigma, J. \\ \quad (((\sigma_o, \emptyset), \Sigma) \models I) \wedge (\sigma_c \perp \sigma_{cl} \perp \sigma_o) \implies \\ \quad (C, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ), \Pi) \preceq^t_{R;G;I} (\mathbb{C}, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ), \Gamma)\end{array}$$

Whenever $(C, (\sigma, \kappa), \Pi) \preceq^t_{R;G;I} (\mathbb{C}, ((\sigma_c, J, \sigma_{cl}), \Sigma, ak), \Gamma)$, then there exists σ_o and σ_t, such that $\sigma = \sigma_c \uplus \sigma_{cl} \uplus \sigma_o \uplus \sigma_t$, $((\sigma_o, \emptyset), \Sigma) \models I$ and followings hold:

1. $(\kappa = (\_, x, \_)) \implies (ak = (\_, (\sigma_t, \_), x, \_))$ and
   $(\kappa = \circ) \implies (ak = \circ \wedge \mathbb{C} = C \wedge \sigma_t = \emptyset)$;
2. if $(C, (\sigma, \kappa)) \xrightarrow[t,\Pi]{e} (C', (\sigma', \kappa'))$, then

$$\exists \mathbb{C}', \sigma'_{cl}, \sigma'_c, \sigma'_o, \sigma'_t, J', \Sigma', ak', \xi.$$
$$(\mathbb{C}, ((\sigma_c, J, \sigma_{cl}), \Sigma, ak)) \circ\!\!\xrightarrow{\xi}{}^{*}_{t,\Gamma} (\mathbb{C}', ((\sigma'_c, J', \sigma'_{cl}), \Sigma', ak')) \wedge$$
$$\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi) \wedge$$
$$\sigma' = \sigma'_c \uplus \sigma'_{cl} \uplus \sigma'_o \uplus \sigma'_t \wedge$$
$$((\sigma'_o, \emptyset), \Sigma') \models I \wedge$$
$$(((\sigma_o \uplus \sigma_t, \mathsf{getJ}(ak)), \Sigma), ((\sigma'_o, \emptyset), \Sigma')) \models G * \mathsf{True} \wedge$$
$$(C', (\sigma', \kappa'), \Pi) \preceq^t_{R;G;I} (\mathbb{C}', ((\sigma'_c, J', \sigma'_{cl}), \Sigma', ak'), \Gamma);$$

or

$$(\mathbb{C}, ((\sigma_c, J, \sigma_{cl}), \Sigma, ak)) \overset{(t,\mathbf{abort})}{\circ\!\!\longrightarrow}_{t,\Gamma} \mathbf{abort};$$

3. if $(C, (\sigma, \kappa)) \xrightarrow[t,\Pi]{e} \mathbf{abort}$,
   then there exists ξ, such that
$$(\mathbb{C}, ((\sigma_c, J, \sigma_{cl}), \Sigma, ak)) \circ\!\!\xrightarrow{\xi}{}^{*}_{t,\Gamma} \mathbf{abort} \wedge$$
$$\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi);$$
4. if $C = \mathbf{skip}$, then $\kappa = \circ$ and $\mathbb{C} = \mathbf{skip}$;
5. for any σ'_c, σ'_o, J', and Σ',
   if $\mathsf{EnvA}((\sigma_o \uplus \sigma_t, \mathsf{getJ}(ak), \Sigma), R, I)((\sigma'_o \uplus \sigma_t, \Sigma')), ((\sigma'_o, \emptyset), \Sigma') \models I$ and $\sigma'_c \perp \sigma_{cl} \perp \sigma'_o \perp \sigma_t$,
   then $(C, (\sigma'_c \uplus \sigma_{cl} \uplus \sigma'_o \uplus \sigma_t, \kappa), \Pi) \preceq^t_{R;G;I} (\mathbb{C}, ((\sigma'_c, J', \sigma_{cl}), \Sigma', ak), \Gamma)$.

11.1.2 Simulation for Thread is Lifted from Simulation for Method, and is Compositional

Lemma 3 (Sim for Thread is Lifted from Sim for Method)**.** *For any $y, f, E, \Pi, t, R, G, I, \Gamma, C, \mathbb{s}, trp$ and trq, if*

1. $\Pi(f) = (x, C)$, $\Gamma(f) = (\mathbb{s}, trp, trq)$,
2. $I \rhd \{R, G\}$, $x \notin dom(R)$, $x \notin dom(G)$
3. $(x, C) \preceq^{R;G;I}_{trp;trq} \mathbb{s}$;

then $(y := f(E), \Pi) \preceq^t_{R;G;I} (y := f(E), \Gamma)$.

Proof. Below we prove: for any σ_c, σ_{cl}, σ_o, Σ and J, if $(\sigma_o, \Sigma) \models I$ and $\sigma_c \perp \sigma_{cl} \perp \sigma_o$, then

$$(y := f(E), (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ), \Pi) \preceq^t_{R;G;I}$$
$$(y := f(E), ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ), \Gamma).$$

Thus we have the following cases:

(a) If $(y := f(E), (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ)) \xrightarrow[t,\Pi]{e} \mathbf{abort}$.

Proof: We need to prove:

$$(y := f(E), ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ)) \overset{(t,\mathbf{abort})}{\circ\!\!\longrightarrow}_{t,\Gamma} \mathbf{abort}$$

From the low-level operational semantics, we know: $f \notin dom(\Pi)$, or $[\![E]\!]_{\sigma_c \uplus \sigma_{cl} \uplus \sigma_o}$ is *undefined*, or $x \notin dom(\sigma_c \uplus \sigma_{cl} \uplus \sigma_o)$.

Then we know: $f \notin dom(\Pi)$, or $[\![E]\!]_{\sigma_c \uplus \sigma_{cl}}$ is *undefined*, or $x \notin dom(\sigma_{cl})$.

Thus by the specification language operational semantics, we know:

$$(y := f(E), ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ)) \xrightarrow[t,\Gamma]{(t,\mathbf{abort})} \mathbf{abort}$$

(b) If $(y := f(E), (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ)) \xrightarrow[t,\Pi]{e} (C', (\sigma', \kappa'))$.

Proof: Let $[\![E]\!]_{\sigma_c \uplus \sigma_{cl} \uplus \sigma_o} = n$.

By the low-level operational semantics, we know $C' = C; \mathbf{noret}$, $\sigma' = \sigma_c \uplus \sigma_{cl} \uplus \sigma_o$ and $\kappa' = (\{x \rightsquigarrow n\}, y, \mathbf{skip})$.

By the definition, we only need to prove:

$$\begin{aligned}
&\exists \mathbb{C}', \sigma'_{cl}, \sigma'_c, \sigma'_o, \sigma'_a, J', \Sigma', ak', \xi. \\
&(y := f(E), ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ)) \xrightarrow[t,\Gamma]{\xi} \\
&(\mathbb{C}', ((\sigma'_c, J', \sigma'_{cl}), \Sigma', ak')) \wedge \\
&\mathsf{get_obsv}(e) = \mathsf{get_obsv}(\xi) \wedge \\
&\sigma' = \sigma'_c \uplus \sigma'_{cl} \uplus \sigma'_o \uplus \sigma'_a \wedge \\
&((\sigma'_o, \emptyset), \Sigma') \models I \wedge \\
&(((\sigma_o, \mathsf{getJ}(ak)), \Sigma), ((\sigma'_o, \emptyset), \Sigma')) \models G * \mathsf{True} \wedge \\
&(C', (\sigma', \kappa'), \Pi) \preceq^t_{R;G;I} (\mathbb{C}', ((\sigma'_c, J', \sigma'_{cl}), \Sigma', ak'), \Gamma)
\end{aligned}$$

or

$$(y := f(E), ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ)) \xrightarrow[t,\Gamma]{(t,\mathbf{abort})} \mathbf{abort}$$

Thus we have the following two cases:

(I) If $(y := f(E), ((\sigma_c, J, \sigma_{cl}), \Sigma, ak)) \xrightarrow[t,\Gamma]{(t,\mathbf{abort})} \mathbf{abort}$.

Proof: By the operational semantics, we know:

$$(y := f(E), (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ)) \xrightarrow[t,\Pi]{e} \mathbf{abort}$$

which leads to a contradiction.

(II) If

$(y := f(E), ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ)) \xrightarrow[t,\Gamma]{\xi}$

$(\mathbb{C}', ((\sigma'_c, J', \sigma'_{cl}), \Sigma', ak'))$.

Proof: By the high-level operational semantics, we know, there exist σ_t, J_t, J' and σ'_c, such that

$$\begin{aligned}
&\Gamma(f) = (\mathbb{s}, trp, trq), \text{and} \\
&\xi = (t, f, n), \text{and} \\
&\mathbb{C}' = \mathbf{fexec}(f, \mathbb{s}, n), \text{and}
\end{aligned}$$

$$[\![E]\!]_{\sigma_c \uplus \sigma_{cl}} = n, \text{and}$$
$$((\sigma_t, J_t), \Sigma) \models trp(n, \overrightarrow{v}), \text{and}$$
$$J = J' \uplus J_t, \text{and}$$
$$\sigma_c = \sigma_c' \uplus \sigma_t. \text{and}$$
$$\sigma_{cl}' = \sigma_{cl}, \text{and}$$
$$\Sigma' = \Sigma, \text{and}$$
$$ak' = ((\sigma_t, J_t), (n, \overrightarrow{v}), y, \mathbf{skip}), \text{and}$$
$$x \in dom(\sigma_{cl}).$$

Thus we get:

$$\begin{aligned}\sigma' &= \sigma_c \uplus \sigma_{cl} \uplus \sigma_o \\ &= (\sigma_c' \uplus \sigma_t) \uplus \sigma_{cl} \uplus \sigma_o \\ &= \sigma_c' \uplus \sigma_{cl} \uplus \sigma_o \uplus \sigma_t\end{aligned}$$

By $I \rhd G$, we know:

$$(((\sigma_o, \emptyset), \Sigma), ((\sigma_o, \emptyset), \Sigma)) \models G * \mathsf{True}$$

Thus we only need to prove:

$$\begin{aligned}&(C; \mathbf{noret}, \\ &(\sigma_c' \uplus \sigma_{cl} \uplus \sigma_o \uplus \sigma_t, (\{x \rightsquigarrow n\}, y, \mathbf{skip})), \Pi) \\ &\preceq^t_{R;G;I} (\mathbf{fexec}(f, \mathbb{s}, n), ((\sigma_c', J', \sigma_{cl}), \Sigma, ak'), \Gamma)\end{aligned}$$

From $(x, C) \preceq^{R;G;I}_{trp;trq} \mathbb{s}$, we know:

$$\begin{aligned}&q = (\exists n'. \langle \mathbf{skip}, n' \rangle * (x \mapsto \_) * \mathsf{lift}(trq(n, n', \overrightarrow{v})) * I), \\ &(C; \mathbf{noret}, \sigma_o \uplus \sigma_a \uplus \sigma_l) \preceq^{R;G;I}_q ((\mathbb{s}, n), (\sigma_t, J_t, \Sigma))\end{aligned}$$

By the following Lemma 4, we have

$$\begin{aligned}&(C; \mathbf{noret}, (\sigma_c' \uplus \sigma_{cl} \uplus \sigma_o \uplus \sigma_t, \\ &\qquad (\{x \rightsquigarrow n\}, y, \mathbf{skip})), \Pi) \\ &\preceq^t_{R;G;I} (\mathbf{fexec}(f, \mathbb{s}, n), ((\sigma_c', J', \sigma_{cl}), \Sigma, ak'), \Gamma)\end{aligned}$$

Thus we are done. □

Lemma 4. *For any C, σ_o, σ_a, σ_l, R, G, I, $trq, n, \overrightarrow{v}$, μ, σ_t, J_t and Σ, if*

1. $(C; \mathbf{noret}, \sigma_o \uplus \sigma_a \uplus \sigma_l) \preceq^{R;G;I}_q (\mu, (\sigma_t, J_t, \Sigma))$,

2. $\sigma_l = \{x \rightsquigarrow \_\}$,

3. $I \rhd \{R, G\}$, $x \notin dom(R)$, $x \notin dom(G)$,

4. $\kappa = (\sigma_l, y, \mathbf{skip})$, $ak = (f, (\sigma_t, J_t, (n, \overrightarrow{v}), y, \mathbf{skip}))$,

5. $((\sigma_o, \emptyset), \Sigma) \models I$,

6. $q = (\exists n'. \langle \mathbf{skip}, n' \rangle * (x \mapsto \_) * \mathsf{lift}(trq(n, n', \overrightarrow{v})) * I)$,

then for any σ_c *and* J, *if* $\sigma_c \perp \sigma_{cl} \perp \sigma_o \perp \sigma_a$, *then*

$$(C; \mathbf{noret}, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o \uplus \sigma_a, \kappa), \Pi)$$
$$\preceq^t_{R;G;I}$$
$$(\lfloor \mu \rfloor_f, ((\sigma_c, J, \sigma_{cl}), \Sigma, ak), \Gamma)$$

Proof. By co-induction, we only need to prove following (1)(2)(3)(4)(5).

(1) $(\kappa = (\_, x, \_)) \implies (ak = (\_, \_, x, \_))$ and
$(\kappa = \circ) \implies (ak = \circ \wedge \mathbb{C} = C \wedge \sigma_t = \emptyset)$.
Proof: By $\kappa = (\sigma_l, y, \mathbf{skip})$ and $ak = (f, (\sigma_t, J_t, (n, \overrightarrow{v}), y, \mathbf{skip}))$.

(2) if $C; \mathbf{noret} = \mathbf{skip}$, then $\kappa = \circ$ and $\mathbb{C} = \mathbf{skip}$.
Proof: It is vacuously true.

(3) for any σ'_c, σ'_o, J', and Σ',
if $\mathsf{EnvA}((\sigma_o \uplus \sigma_a, \mathsf{getJ}(ak), \Sigma), R, I)((\sigma'_o \uplus \sigma_a, \Sigma'))$, $((\sigma'_o, \emptyset), \Sigma') \models I$ and $\sigma'_c \perp \sigma_{cl} \perp \sigma'_o \perp \sigma_a$,
then $(C, (\sigma'_c \uplus \sigma_{cl} \uplus \sigma'_o \uplus \sigma_a, \kappa), \Pi) \preceq^t_{R;G;I} (\lfloor \mu \rfloor_f, ((\sigma'_c, J', \sigma_{cl}), \Sigma', ak), \Gamma)$.
Proof: By $x \notin dom(R)$, we know:

$$\mathsf{EnvA}((\sigma_o \uplus \sigma_a \uplus \sigma_l, J_t, \Sigma), R, I)((\sigma'_o \uplus \sigma_a \uplus \sigma_l, \Sigma')).$$

Thus by $(C, \sigma_o \uplus \sigma_a \uplus \sigma_l) \preceq^{R;G;I}_q (\mu, (\sigma_t, J_t, \Sigma))$, we get:

$$(C, \sigma'_o \uplus \sigma_a \uplus \sigma_l) \preceq^{R;G;I}_q (\mu, (\sigma_t, J_t, \Sigma')).$$

By the co-induction hypothesis, we get:

$$(C, (\sigma'_c \uplus \sigma_{cl} \uplus \sigma'_o \uplus \sigma_a, \kappa), \Pi)$$
$$\preceq^t_{R;G;I}$$
$$(\lfloor \mu \rfloor_f, ((\sigma'_c, J', \sigma_{cl}), \Sigma', ak), \Gamma)$$

(4) if $(C, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o \uplus \sigma_a, \kappa)) \xrightarrow[t,\Pi]{e} \mathbf{abort}$, then we need to prove:
there exists ξ, such that

$$(\mathbb{C}, ((\sigma_c, J, \sigma_{cl}), \Sigma, ak)) \circ \xrightarrow[t,\Gamma]{\xi}{}^{*} \mathbf{abort} \wedge$$
$$\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi);$$

Proof: $(C, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o \uplus \sigma_a, \kappa)) \xrightarrow[t,\Pi]{e} \mathbf{abort}$ contradicts
$(C, \sigma_o \uplus \sigma_a \uplus \sigma_l) \preceq^{R;G;I}_q (\mu, (\sigma_t, J_t, \Sigma))$.

(5) if $(C, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o \uplus \sigma_a, \kappa)) \xrightarrow[t,\Pi]{e} (C', (\sigma', \kappa'))$.

This is the last case, thus we prove it outside.

The last case:

If $(C, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o \uplus \sigma_a, \kappa)) \xrightarrow[t,\Pi]{e} (C', (\sigma', \kappa'))$, then we need to prove:

$$
\begin{aligned}
&\exists \mathbb{C}', \sigma'_{cl}, \sigma'_c, \sigma'_o, \sigma'_a, J', \Sigma', ak', \xi. \\
&(\lfloor \mu \rfloor_f, ((\sigma_c, J, \sigma_{cl}), \Sigma, ak)) \\
&\circ\!\!\xrightarrow[t,\Gamma]{\xi}{}^{*} (\mathbb{C}', ((\sigma'_c, J', \sigma'_{cl}), \Sigma', ak')) \wedge \\
&\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi) \wedge \\
&\sigma' = \sigma'_c \uplus \sigma'_{cl} \uplus \sigma'_o \uplus \sigma'_a \wedge \\
&((\sigma'_o, \emptyset), \Sigma') \models I \wedge \\
&(((\sigma_o \uplus \sigma_a, \mathsf{getJ}(ak)), \Sigma), ((\sigma'_o, \emptyset), \Sigma')) \models G * \mathsf{True} \wedge \\
&(C', (\sigma', \kappa'), \Pi) \preceq^t_{R;G;I} (\mathbb{C}', ((\sigma'_c, J', \sigma'_{cl}), \Sigma', ak'), \Gamma);
\end{aligned}
$$

or

$$
(\lfloor \mu \rfloor_f, ((\sigma_c, J, \sigma_{cl}), \Sigma, ak)) \overset{(t,\mathbf{abort})}{\circ\!\!\longrightarrow_{t,\Gamma}} \mathbf{abort}.
$$

Thus we need to prove following cases (A)(B)(C):

(A) If $C \neq \mathbf{E}[\mathbf{return}\ E]$ and

$(C, \sigma_o \uplus \sigma_a \uplus \sigma_l) \bullet\!\!\longrightarrow (C', \sigma'')$:

Proof: By $(C, \sigma_o \uplus \sigma_a \uplus \sigma_l) \preceq^{R;G;I}_q (\mu, (\sigma_t, J_t, \Sigma))$, we know: there exist μ', σ'_t, J'_t and Σ', such that
$(\mu, (\sigma_t, J_t, \Sigma)) \rightrightarrows^* (\mu', (\sigma'_t, J'_t, \Sigma'))$,
$(((\sigma_o \uplus \sigma_a \uplus \sigma_l, J_t), \Sigma), ((\sigma'', \emptyset), \Sigma')) \models G * \mathsf{True}$ and
$(C', \sigma'') \preceq^{R;G;I}_q (\mu', (\sigma'_t, J'_t, \Sigma'))$.
By $(\mu, (\sigma_t, J_t, \Sigma)) \rightrightarrows^* (\mu', (\sigma'_t, J'_t, \Sigma'))$, let $ak' = (f, (\sigma'_t, J'_t), (n, \overrightarrow{v}), y, \mathbf{skip})$, we know:

$$
\begin{aligned}
&(\lfloor \mu \rfloor_f, ((\sigma_c, J, \sigma_{cl}), \Sigma, ak)) \circ\!\!\xrightarrow[t,\Gamma]{}{}^{*} \\
&(\lfloor \mu' \rfloor_f, ((\sigma_c, J, \sigma_{cl}), \Sigma', ak'))
\end{aligned}
$$

By $(((\sigma_o \uplus \sigma_a \uplus \sigma_l, J_t), \Sigma), ((\sigma'', \emptyset), \Sigma')) \models G * \mathsf{True}$ and $I \rhd G$, we know: there exist $\sigma'_o, \sigma'_a, \sigma'_l$, such that

$$
\begin{aligned}
\sigma'' &= \sigma'_o \uplus \sigma'_a \uplus \sigma'_l \\
(\sigma'_o, \Sigma') &\models I \\
dom(\sigma'_l) &= \{x\}
\end{aligned}
$$

Then from $(\sigma'_o, \Sigma') \models I$ and $I \rhd G$, we know:

$$
(((\sigma_o \uplus \sigma_a \uplus \sigma_l, J_t), \Sigma), ((\sigma'_o, \emptyset), \Sigma')) \models G * \mathsf{True}
$$

By $x \notin dom(G)$, we know:

$$(((\sigma_o \uplus \sigma_a, J_t), \Sigma), ((\sigma'_o, \emptyset), \Sigma')) \models G * \mathsf{True}$$

By the co-inductive hypothesis, we get:

$$(C', (\sigma', \kappa'), \Pi) \preceq^t_{R;G;I} (\lfloor \mu' \rfloor_f, ((\sigma_c, J, \sigma_{cl}), \Sigma', ak'), \Gamma)$$

(B) If $C \neq \mathbf{E}[\mathbf{return}\ E]$ and

$(C, \sigma_o \uplus \sigma_a \uplus \sigma_l) \bullet\!\!\longrightarrow \mathbf{abort}$.

Proof: it contradicts $(C, \sigma_o \uplus \sigma_a \uplus \sigma_l) \preceq^{R;G;I}_q (\mu, (\sigma_t, J_t, \Sigma))$.

(C) If $C = \mathbf{E}[\mathbf{return}\ E]$.

By $(C, \sigma_o \uplus \sigma_a \uplus \sigma_l) \preceq^{R;G;I}_q (\mu, (\sigma_t, J_t, \Sigma))$, we know: there exists n', such that

$$\begin{aligned}
&[\![E]\!]_{\sigma_o \uplus \sigma_a \uplus \sigma_l} = n' \\
&(\sigma_o \uplus \sigma_a \uplus \sigma_l), (\sigma_t, J_t, \Sigma, \mu) \models \\
&\qquad \langle \mathbf{skip}, n' \rangle * (x \mapsto _) * \mathsf{lift}(trq(n, n', \overrightarrow{v})) * I
\end{aligned}$$

From $(\sigma_o, \Sigma) \models I$, we know $\sigma_a = \sigma_t$, $((\sigma_t, J_t), \Sigma) \models trq(n, n', \overrightarrow{v})$ and $\mu = (\mathbf{skip}, n')$.

Thus we get $\sigma' = (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o \uplus \sigma_a)\{y \rightsquigarrow n'\}$ and $\kappa' = \circ$ and $C' = \mathbf{skip}$.

We need to prove following cases:

(a) If

$$(\mathbf{fexec}(f, \mathbf{skip}, n'), ((\sigma_c, J, \sigma_{cl}), \Sigma, ak)) \xrightarrow[t,\Gamma]{(t, \mathbf{abort})} \mathbf{abort}$$

By the definition of thread simulations, it is trivial.

(b) If

$$\begin{aligned}
&(\mathbf{fexec}(f, \mathbf{skip}, n'), ((\sigma_c, J, \sigma_{cl}), \Sigma, ak)) \xrightarrow[t,\Gamma]{(t, \mathbf{ok}, n')} \\
&(\mathbf{skip}, ((\sigma'_c, J', \sigma'_{cl}), \Sigma, \circ)).
\end{aligned}$$

Proof: By the abstract operational semantics, we know $\sigma'_c = \sigma_c \uplus \sigma_t$, $J' = J \uplus J_t$ and $\sigma'_{cl} = \sigma_{cl}\{y \rightsquigarrow n\}$.

Thus we know:

$$\begin{aligned}
&\sigma' = \sigma'_c \uplus \sigma'_{cl} \uplus \sigma_o \uplus \emptyset(\text{let } \sigma'_a = \emptyset) \\
&(((\sigma_o \uplus \sigma_a, \mathsf{getJ}(ak))), I), ((\sigma_o, \emptyset), \Sigma) \models G * \mathsf{True}
\end{aligned}$$

From the definition of the thread simulation, we get:

$$(\mathbf{skip}, (\sigma', \circ), \Pi) \preceq^t_{R;G;I} (\mathbf{skip}, ((\sigma'_c, J', \sigma'_{cl}), \Sigma, \circ))$$

Thus we are done. □

Lemma 5 (Simulation for Thread Holds on Simple Commands)**.** *If*

$C \in \{x := E, x := [E], [E] := E, \mathsf{print}\ E\}$,

then $(C, \Pi) \preceq^t_{R;G;I} (C, \Gamma)$.

Proof. Proofs for those four cases are similar. Thus we only show the proof for $(x := E)$.

We need to prove:

$$(x := E, \Pi) \preceq^t_{R;G;I} (x := E, \Gamma)$$

By definition, we need to prove: for any $\sigma_c, \sigma_{cl}, \sigma_o, \Sigma$ and J,

if $(((\sigma_o, \emptyset), \Sigma) \models I)$ and $\sigma_c \perp \sigma_{cl} \perp \sigma_o$, then

$(x := E, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \perp), \Pi)$

$\preceq^t_{R;G;I} (x := E, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ), \Gamma)$.

By co-induction, we need to prove following cases:

1. for any σ'_c, σ'_o, J' and Σ',

   if $\mathsf{EnvA}((\sigma_o, \mathsf{getJ}(\circ), \Sigma), R, I)((\sigma'_o, \Sigma'))$,

   $((\sigma'_o, \emptyset), \Sigma') \models I$ and $\sigma'_c \perp \sigma_{cl} \perp \sigma'_o$, then

   $(x := E, (\sigma'_c \uplus \sigma_{cl} \uplus \sigma'_o, \circ), \Pi)$

   $\preceq^t_{R;G;I} (x := E, ((\sigma'_c, J', \sigma_{cl}), \Sigma', \circ), \Pi)$.

   Proof: By co-induction hypothesis, we prove it.

2. if $(x := E, (\sigma_c \uplus \sigma_l \uplus \sigma_o, \circ)) \xrightarrow[t,\Pi]{e} (C', (\sigma', \circ))$, then there exist $\mathbb{C}', \sigma'_{cl}, \sigma'_c$ and ξ, such that

   $$\begin{aligned}
   &(x := E, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ)) \circ\!\!\xrightarrow[t,\Gamma]{\xi}{}^{*} \\
   &(\mathbb{C}', ((\sigma'_c, J, \sigma'_{cl}), \Sigma, \circ)) \wedge \\
   &\mathsf{get_obsv}(e) = \mathsf{get_obsv}(\xi) \wedge \\
   &\sigma' = \sigma'_c \uplus \sigma'_{cl} \uplus \sigma_o \wedge \\
   &(C', (\sigma', \circ), \Pi) \preceq^t_{R;G;I} (\mathbb{C}', ((\sigma'_c, J, \sigma'_{cl}), \Sigma, \circ), \Gamma).
   \end{aligned}$$

   Proof: By operational semantics, we know:

   $$\begin{aligned}
   &C' = \mathbf{skip} \\
   &[\![E]\!]_{\sigma_c \uplus \sigma_l \uplus \sigma_o} = n \\
   &\sigma'_c \uplus \sigma'_{cl} = \sigma_c \uplus \sigma_{cl}\{x \rightsquigarrow n\}
   \end{aligned}$$

   Thus we only need to prove:

   $$\begin{aligned}
   &(\mathbf{skip}, (\sigma'_c \uplus \sigma'_{cl} \uplus \sigma_o, \circ), \Pi) \preceq^t_{R;G;I} \\
   &(\mathbf{skip}, ((\sigma'_c, J, \sigma'_{cl}), \Sigma, \circ), \Gamma)
   \end{aligned}$$

   which is trivial.

3. if $(x := E, (\sigma_c \uplus \sigma_l \uplus \sigma_o, \circ)) \xrightarrow[t,\Pi]{e} \mathbf{abort}$,

   then we know $(x := E, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ)) \multimap \xrightarrow[t,\Gamma]{\xi}{}^{*} \mathbf{abort}$.

   Thus we prove it immediately.

□

Lemma 6 (Simulation for Thread Holds on Skip)**.** $(\mathbf{skip}, \Pi) \preceq^t_{R;G;I} (\mathbf{skip}, \Gamma)$.

Proof. By definition, it trivially holds. □

Lemma 7 (Simulation for Thread Holds on Atomic Block)**.** $(\langle C\rangle, \Pi) \preceq^t_{R;G;I} (\langle C\rangle, \Gamma)$

Proof. The proof is similar to the proof of Lemma 5. We can view $\langle C\rangle$ as a single-step instruction from clients, which does not access object states. □

Lemma 8 (Sequential Compositionality of Simulation)**.** *If*
$(C_1, \Pi) \preceq^t_{R;G;I} (\mathbb{C}_1, \Gamma)$ *and* $(C_2, \Pi) \preceq^t_{R;G;I} (\mathbb{C}_2, \Gamma)$,
then $(C_1; C_2, \Pi) \preceq^t_{R;G;I} (\mathbb{C}_1; \mathbb{C}_2, \Gamma)$.

Proof. By definition, we need to prove: for any $\sigma_c, \sigma_{cl}, \sigma_o, \Sigma$ and J,
if $(((\sigma_o, \emptyset), \Sigma)) \models I$ and $\sigma_c \perp \sigma_{cl} \perp \sigma_o$,
then $(C_1; C_2, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ), \Pi)$
$\preceq^t_{R;G;I} (\mathbb{C}_1; \mathbb{C}_2, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ), \Gamma)$.

By the premise, we know
$(C_1, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ), \Pi)$
$\preceq^t_{R;G;I} (\mathbb{C}_1, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ), \Gamma)$.

By the following Lemma 9, we prove it. □

Lemma 9. *If*

1. $(C_1, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ), \Pi)$
   $\preceq^t_{R;G;I} (\mathbb{C}_1, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ), \Gamma)$ *and*
2. $(C_2, \Pi) \preceq^t_{R;G;I} (\mathbb{C}_2, \Gamma)$ *and*
3. $((\sigma_o, \emptyset), \Sigma) \models I$,

then
$(C_1; C_2, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ), \Pi)$
$\preceq^t_{R;G;I} (\mathbb{C}_1; \mathbb{C}_2, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ), \Gamma)$.

Proof. By co-induction, we need to prove following cases:

1. for any σ'_c, σ'_o, J' and Σ',
   if $\mathsf{EnvA}((\sigma_o, \emptyset, \Sigma), R, I)((\sigma'_o, \Sigma'))$,
   $((\sigma'_o, \emptyset), \Sigma') \models I$ and $\sigma'_c \perp \sigma_{cl} \perp \sigma'_o$,
   then $(C'_1 C_2, (\sigma'_c \uplus \sigma_{cl} \uplus \sigma'_o, \kappa), \Pi)$
   $\preceq^t_{R;G;I} (\mathbb{C}_1; \mathbb{C}_2, ((\sigma'_c, J', \sigma_{cl}), \Sigma', \circ), \Gamma)$.

proof : By co-induction hypothesis, we can prove it.

2. if $(C_1; C_2, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ)) \xrightarrow[t,\Gamma]{e} \mathbf{abort}$,
   then there exists ξ, such that
   $(\mathbb{C}_1; \mathbb{C}_2, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ)) \circ\!\!\xrightarrow[t,\Gamma]{\xi}{}^{*} \mathbf{abort}$
   and $\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi)$.
   proof : By the premise $(C_1, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o), \Pi) \preceq^t_{R;G;I} (\mathbb{C}_1, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ), \Gamma)$, we know, there exists ξ, such that
   $(\mathbb{C}_1, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ)) \circ\!\!\xrightarrow[t,\Gamma]{\xi}{}^{*} \mathbf{abort}$
   and $\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi)$.
   Thus we prove it.
3. if $C_1 \neq \mathbf{E}[y := f(E)]$ and
   $(C_1; C_2, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ)) \xrightarrow[t,\Pi]{e} (C', (\sigma', \circ))$,
   then there exist $\mathbb{C}', \sigma'_{cl}, J'$ and ξ, such that

$$
\begin{aligned}
&(\mathbb{C}_1; \mathbb{C}_2, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ)) \circ\!\!\xrightarrow[t,\Gamma]{\xi} (\mathbb{C}', ((\sigma'_c, J', \sigma'_{cl}), \Sigma, \circ)) \wedge \\
&\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi) \wedge \\
&\sigma' = \sigma'_c \uplus \sigma'_{cl} \uplus \sigma_o \wedge \\
&(C', (\sigma', \circ), \Pi) \preceq^t_{R;G;I} (\mathbb{C}', ((\sigma'_c, J', \sigma'_{cl}), \Sigma, \circ), \Gamma).
\end{aligned}
$$

   Proof: We have follwing cases:
   (a) if $C_1 = \mathbf{skip}$, then we have: $C' = C_2$, $\sigma' = \sigma_c \uplus \sigma_{cl} \uplus \sigma_o$ and $\mathbb{C}_1 = \mathbf{skip}$.
   Thus we have:
   $(\mathbf{skip}; \mathbb{C}_2, ((\sigma_t, J, \sigma_{cl}), \Sigma, \circ)) \circ\!\!\xrightarrow[t,\Gamma]{\xi}{}^{*} (\mathbb{C}_2, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ))$
   Thus we only need to prove:
   $(C_2, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ), \Pi)$
   $\preceq^t_{R;G;I} (\mathbb{C}_2, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ), \Gamma)$
   By premise 2, we prove it.
   (b) if $C_1 \neq \mathbf{skip}$, then we know:

$$
\begin{aligned}
&C' = C'_1; C_2 \\
&(C_1, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ)) \xrightarrow[t,\Pi]{e} (C'_1, (\sigma'_c \uplus \sigma'_{cl} \uplus \sigma_o, \circ)) \\
&\sigma' = \sigma'_c \uplus \sigma'_{cl} \uplus \sigma_o
\end{aligned}
$$

   By the premise
   $(C_1, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ), \Pi) \preceq^t_{R;G;I}$
   $(\mathbb{C}_1, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ), \Gamma)$, we know:

$$
(\mathbb{C}_1, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ)) \circ\!\!\xrightarrow[t,\Gamma]{\xi}{}^{*} (\mathbb{C}'_1, ((\sigma'_c, J', \sigma'_{cl}), \Sigma, \circ)) \wedge
$$

$$\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi) \wedge$$
$$(C_1', (\sigma_c' \uplus \sigma_{cl}' \uplus \sigma_o, \circ), \Pi) \preceq^t_{R;G;I}$$
$$(\mathbb{C}_1', ((\sigma_c', J', \sigma_{cl}'), \Sigma, \circ), \Gamma)$$

Thus we have:
$(\mathbb{C}_1; \mathbb{C}_2, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ)) \circ\!\xrightarrow[t,\Gamma]{\xi}{}^{*}$
$(\mathbb{C}_1'; \mathbb{C}_2, ((\sigma_c', J', \sigma_{cl}'), \Sigma, \circ))$.

By co-induction hypothesis, we prove the case.

4. if $C_1 = \mathbf{E}[y := f(E)]$, let $C_1 = (y := f(E)); C_1'$,
   then we have: $\Pi(f) = (x, C)$,
   $(C_1; C_2, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ)) \xrightarrow[t,\Pi]{e}$
   $(C; \mathbf{noret}, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \kappa))$,
   $\kappa = (\{x \rightsquigarrow n\}, y, C_1'; C_2)$ and
   $[\![E]\!]_{\sigma_c \uplus \sigma_{cl} \uplus \sigma_o} = n$.
   Let $\kappa' = (\{x \rightsquigarrow n\}, y, C_1')$, we have
   $(C_1, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ)) \xrightarrow[t,\Pi]{e}$
   $(C; \mathbf{noret}, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \kappa'))$.

   By the premise $(C_1, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ), \Pi) \preceq^t_{R;G;I} (\mathbb{C}_1, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ), \Gamma)$,
   we know there exist $\mathbb{C}', \sigma_c', \sigma_o', \Sigma', J', \sigma_t', ak'$ and ξ, such that

$$(\mathbb{C}_1, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ)) \circ\!\xrightarrow[t,\Gamma]{\xi}{}^{*} (\mathbb{C}', ((\sigma_c', J', \sigma_{cl}'), \Sigma', ak')) \quad (*C.1)$$
$$\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi)$$
$$\sigma_c \uplus \sigma_{cl} \uplus \sigma_o = \sigma_c' \uplus \sigma_{cl}' \uplus \sigma_o' \uplus \sigma_t'$$
$$((\sigma_o', \emptyset), \Sigma') \models I$$
$$(((\sigma_o, \emptyset), \Sigma), ((\sigma_o', \emptyset), \Sigma')) \models G * \mathsf{True}$$

   and

$$(C; \mathbf{noret}, (\sigma_c' \uplus \sigma_{cl}' \uplus \sigma_o' \uplus \sigma_t', \kappa'), \Pi) \preceq^t_{R;G;I}$$
$$(\mathbb{C}', ((\sigma_c', J', \sigma_{cl}'), \Sigma', ak'), \Gamma) \quad (*C.2)$$

   By (*C.2), we know there exist $\mathbb{C}_1', f', n', \overrightarrow{v}'$ and J_t', such that
   $ak' = (f', (\sigma_t', J_t'), (n', \overrightarrow{v}'), y, \mathbb{C}_1')$.
   Let $ak = (f', (\sigma_t', J_t'), (n', \overrightarrow{v}'), y, \mathbb{C}_1'; \mathbb{C}_2)$.
   By (*C.1), we know

$$\sigma_c = \sigma_c' \uplus \sigma_t'$$

$$J = J' \uplus J'_t$$

$$\sigma'_o = \sigma_o$$

$$\Sigma' = \Sigma$$

$$\sigma'_{cl} = \sigma_{cl}$$

$$\mathbb{C}' = \mathbf{fexec}(f', n')$$

Thus we only need to prove:

$$(C; \mathbf{noret}, (\sigma'_c \uplus \sigma_{cl} \uplus \sigma_o \uplus \sigma'_t, \kappa), \Pi) \preceq^t_{R;G;I}$$
$$(\mathbb{C}', ((\sigma'_c, J', \sigma_{cl}), \Sigma, ak), \Gamma)$$

By the following Lemma 10, we prove it.

□

Lemma 10. *If*

1. $(C, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o \uplus \sigma_t, \kappa_1), \Pi) \preceq^t_{R;G;I}$
   $(\mathbb{C}, ((\sigma_c, J, \sigma_{cl}), \Sigma, ak_1), \Gamma)$,
2. $(C_1, \Pi) \preceq^t_{R;G;I} (\mathbb{C}_2, \Gamma)$,
3. $ak_1 = (f, (\sigma_t, J_t), (n, \overrightarrow{v}), y, \mathbb{C}'_1)$,
   $ak = (f, (\sigma_t, J_t), (n, \overrightarrow{v}), y, \mathbb{C}'_1\mathbb{C}_2)$,
   $\kappa_1 = (\sigma_l, y, C'_1)$,
   $\kappa = (\sigma_l, y, C'_1; C_2)$,
4. $\exists C'.C = C'; \mathbf{noret}$,

then

$(C, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o \uplus \sigma_t, \kappa), \Pi) \preceq^t_{R;G;I}$
$(\mathbb{C}, ((\sigma_c, J, \sigma_{cl}), \Sigma, ak), \Gamma)$.

Proof. By co-induction, we have following cases:

1. for any σ'_c, σ'_o, J' and Σ',
   if $\mathsf{EnvA}((\sigma_o \uplus \sigma_t, \mathsf{getJ}(ak), \Sigma), R, I)((\sigma'_o \uplus \sigma_t, \Sigma'))$,
   $((\sigma'_o, \emptyset), \Sigma') \models I$ and $\sigma'_c \perp \sigma_{cl} \perp \sigma'_o \perp \sigma_t$, then
   $(C, (\sigma'_c \uplus \sigma_{cl} \uplus \sigma'_o \uplus \sigma_t, \kappa), \Pi) \preceq^t_{R;G;I}$
   $(\mathbb{C}, ((\sigma'_c, J', \sigma_{cl}), \Sigma', ak), \Gamma)$.
   Proof: By co-induction hypothesis, we can prove it easily.
2. if $(C, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o \uplus \sigma_t, \kappa)) \xrightarrow[t,\Pi]{e} \mathbf{abort}$,
   then there exists ξ, such that
   $(\mathbb{C}, ((\sigma_c, J, \sigma_{cl}), \Sigma, ak)) \xrightarrow[t,\Gamma]{\xi}{}^{*} \mathbf{abort}$
   and $\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi)$.
   Proof: By the task simulation in premise 1, we can prove it easily.

3. If $C = \textbf{skip}$, then ...

   Proof: This is vacuously true.

4. if $(C, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o \uplus \sigma_t, \kappa)) \xrightarrow[t,\Pi]{e} (C', (\sigma', \kappa'))$

   and $\kappa' \neq \circ$, then there exist $\mathbb{C}', \sigma'_{cl}, \sigma'_c, \sigma'_o, \sigma'_t, J', \Sigma', ak'$ and ξ, such that

$$(\mathbb{C}, ((\sigma_c, J, \sigma_{cl}), \Sigma, ak)) \circ\xrightarrow[t,\Pi]{\xi}{}^{*} (\mathbb{C}', ((\sigma'_c, J', \sigma'_{cl}), \Sigma', ak'))$$

$$\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi)$$

$$\sigma' = \sigma'_c \uplus \sigma'_{cl} \uplus \sigma'_o \uplus \sigma'_t$$

$$((\sigma'_o, \emptyset), \Sigma') \models I$$

$$(((\sigma_o \uplus \sigma_t, \mathsf{getJ}(ak)), \Sigma), ((\sigma'_o, \emptyset), \Sigma')) \models G * \mathsf{True}$$

$$(C', (\sigma', \kappa'), \Pi) \preceq^t_{R;G;I} (\mathbb{C}', ((\sigma'_c, J', \sigma'_{cl}), \Sigma', ak'), \Gamma).$$

   proof : Construct κ'_1 by replacing $(C'_1; C_2)$ to C'_1 in κ'.

   Then we have

   $(C, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o \uplus \sigma_t, \kappa_1)) \xrightarrow[t,\Pi]{e} (C', (\sigma', \kappa'_1))$.

   By premise 1, we know there exist $\mathbb{C}', \sigma'_{cl}, \sigma'_c, \sigma'_o, \sigma'_t, J', \Sigma', ak'_1$ and ξ, such that

$$(\mathbb{C}, ((\sigma_c, J, \sigma_{cl}), \Sigma, ak_1)) \circ\xrightarrow[t,\Pi]{\xi}{}^{*} (\mathbb{C}', ((\sigma'_c, J', \sigma'_{cl}), \Sigma', ak'_1))$$

$$\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi)$$

$$\sigma' = \sigma'_c \uplus \sigma'_{cl} \uplus \sigma'_o \uplus \sigma'_t$$

$$((\sigma'_o, \emptyset), \Sigma') \models I$$

$$(((\sigma_o \uplus \sigma_t, \mathsf{getJ}(ak)), \Sigma), ((\sigma'_o, \emptyset), \Sigma')) \models G * \mathsf{True}$$

$$(C', (\sigma', \kappa'_1), \Pi) \preceq^t_{R;G;I} (\mathbb{C}', ((\sigma'_c, J', \sigma'_{cl}), \Sigma', ak'_1), \Gamma).$$

   Construct ak' by replacing $\mathbb{C}'_1$ to $\mathbb{C}'_1; \mathbb{C}_2$ in ak'_1.

   Thus we have:

   $(\mathbb{C}, ((\sigma_c, J, \sigma_{cl}), \Sigma, ak)) \circ\xrightarrow[t,\Pi]{\xi}{}^{*} (\mathbb{C}', ((\sigma'_c, J', \sigma'_{cl}), \Sigma', ak'))$

   Thus we only need to prove:

   $(C', (\sigma', \kappa'), \Pi) \preceq^t_{R;G;I} (\mathbb{C}', ((\sigma'_c, J', \sigma'_{cl}), \Sigma', ak'), \Gamma)$

   By co-induction hypothesis, we prove it.

5. if $(C, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o \uplus \sigma_t, \kappa)) \xrightarrow[t,\Pi]{e} (C', (\sigma', \circ))$.

   proof : We know $C' = C'_1; C_2$ and $\sigma' = \sigma'_c \uplus \sigma'_{cl} \uplus \sigma_o$.

   Thus we have:

   $(C, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o \uplus \sigma_t, \kappa_1)) \xrightarrow[t,\Pi]{e} (C', (\sigma', \circ))$

   By premise 1, we know:

$$(\mathbb{C}, ((\sigma_c, J, \sigma_{cl}), \Sigma, ak_1)) \circ\xrightarrow[t,\Pi]{\xi}{}^{*} (\mathbb{C}'_1, ((\sigma'_c, J', \sigma'_{cl}), \Sigma, \circ))$$

$$\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi)$$

$$(C_1', (\sigma', \circ), \Pi) \preceq^t_{R;G;I} (\mathbb{C}_1', ((\sigma_c', J', \sigma_{cl}'), \Sigma, \circ), \Gamma)$$

Thus we know:

$(\mathbb{C}, ((\sigma_c, J, \sigma_{cl}), \Sigma, ak)) \circ\xrightarrow[t,\Pi]{\xi}{}^{*} (\mathbb{C}_1', ((\sigma_c', J', \sigma_{cl}'), \Sigma, \circ))$

Thus we only need to prove:

$(C_1'; C_2, (\sigma', \circ), \Pi) \preceq^t_{R;G;I} (\mathbb{C}_1'; \mathbb{C}_2, ((\sigma_c', J', \sigma_{cl}'), \Sigma, \circ), \Gamma)$

By Lemma 9, we prove it.

□

Lemma 11 (If-then-else Compositionality of Simulation)**.** *If*

1. $(C_1, \Pi) \preceq^t_{R;G;I} (\mathbb{C}_1, \Gamma)$*, and*
2. $(C_2, \Pi) \preceq^t_{R;G;I} (\mathbb{C}_2, \Gamma)$*,*

then (**if** (B) C_1 **else** $C_2, \Pi) \preceq^t_{R;G;I}$ (**if** (B) $\mathbb{C}_1$ **else** $\mathbb{C}_2, \Gamma$).

Proof. Let $C =$ (**if** (B) C_1 **else** C_2) and $\mathbb{C} =$ (**if** (B) $\mathbb{C}_1$ **else** $\mathbb{C}_2$).

By definition, we need to prove, for any $\sigma_c, \sigma_{cl}, \sigma_o, \Sigma$ and J, if $(((\sigma_o, \emptyset), \Sigma) \models I)$ and $(\sigma_c \perp \sigma_{cl} \perp \sigma_o)$, then $(C, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ), \Pi) \preceq^t_{R;G;I} (\mathbb{C}, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ), \Gamma)$.

By co-induction, we need to prove following cases:

(1) for any σ_c', σ_o', J', and Σ',

if $\mathsf{EnvA}((\sigma_o, \mathsf{getJ}(ak), \Sigma), R, I)((\sigma_o', \Sigma'))$,

$((\sigma_o', \emptyset), \Sigma') \models I$ and $\sigma_c' \perp \sigma_{cl} \perp \sigma_o'$, then

$(C, (\sigma_c' \uplus \sigma_{cl} \uplus \sigma_o', \circ), \Pi) \preceq^t_{R;G;I} (\mathbb{C}, ((\sigma_c', J', \sigma_{cl}), \Sigma', \circ), \Gamma)$.

Proof: By co-induction hypothesis, we can prove it easily.

(2) if $(C, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ)) \xrightarrow[t,\Pi]{e}$ **abort**,

then there exists ξ, such that

$(\mathbb{C}, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ)) \circ\xrightarrow[t,\Gamma]{\xi}{}^{*}$ **abort** and

$\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi)$.

Proof: By operational semantics, we can prove it immediately.

(3) if $(C, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ)) \xrightarrow[t,\Pi]{e} (C_1, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ))$,

where $[\![B]\!]_{\sigma_c \uplus \sigma_{cl} \uplus \sigma_o} = \mathsf{true}$,

then there eixsts ξ, such that

$(\mathbb{C}, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ)) \circ\xrightarrow[t,\Gamma]{\xi}{}^{*} (\mathbb{C}_1, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ))$,

$\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi)$ and

$(C_1, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ), \Pi) \preceq^t_{R;G;I} (\mathbb{C}_1, ((\sigma_c, J, \sigma_{cl}), \Sigma, \circ), \Gamma)$.

Proof: By operational semantics and premise 1, we can finish this case.

(4) The case is similar when

$(C, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ)) \xrightarrow[t,\Pi]{e} (C_2, (\sigma_c \uplus \sigma_{cl} \uplus \sigma_o, \circ))$,

where $[\![B]\!]_{\sigma_c \uplus \sigma_{cl} \uplus \sigma_o} = \mathsf{false}$.

□

Lemma 12 (While-loop Compositionality of Simulation)**.** *If* $(C, \Pi) \preceq^{t}_{R;G;I} (\mathbb{C}, \Gamma)$*,*
then $(\textbf{while}\ (B)\ \{C\}, \Pi) \preceq^{t}_{R;G;I} (\textbf{while}\ (B)\ \{\mathbb{C}\}, \Gamma)$*.*

Proof. The proof is similar to the structure proving Lemma 11, with the help of Lemma 8 and Lemma 6. □

Lemma 13 (Parallel Compositionality of Simulation)**.** *If for any* $i \in \{1, \ldots, n\}$*, we have* $(C_i, \Pi) \preceq^{i}_{R_i;G_i;I} (\mathbb{C}_i, \Gamma)$*,*
$(\forall j.j \neq i \implies G_i \Rightarrow R_j)$*,* $I \rhd \{R_i, G_i\}$*,* $\varphi \Rightarrow I$*,*
then $(\textbf{let}\ \Pi\ \textbf{in}\ C_1 \parallel \ldots \parallel C_n) \preceq_{\varphi} (\textbf{with}\ \Gamma\ \textbf{do}\ \mathbb{C}_1 \parallel \ldots \parallel \mathbb{C}_n)$*.*

Proof. By the following Lemma 14. □

Lemma 14. *For any* $i \in \{1, \ldots, n\}$*, if there exist* σ_c*,* σ_s*,* Π*,* Γ*,* I*,* J*,* Σ*,* $\mathcal{K}$*,* $\mathbb{K}$*,* K_c*,* σ_w*,* W*,* S*,* $\mathbb{W}$*,* $\mathbb{S}$*,* ak_i*,* C_i*,* $\mathbb{C}_i$*,* G_i*,* R_i*,* κ_i *and* σ_{cl_i}*, such that*

1. $(C_i, (\sigma_c \uplus \sigma_{cl_i} \uplus \sigma_s \uplus \sigma_i, \kappa_i), \Pi) \preceq^{i}_{R_i;G_i;I}$
   $(\mathbb{C}_i, ((\sigma_c, J, \sigma_{cl_i}), \Sigma, ak_i), \Gamma)$,
2. $((\sigma_s, \emptyset), \Sigma) \models I$,
3. $(\forall i. \forall j.j \neq i \implies G_i \Rightarrow R_j)$,
   $I \rhd \{R_i, G_i\}$,
4. $\mathcal{K} = \{i \rightsquigarrow \kappa_i \mid i \in [1..n]\}$,
   $\mathbb{K} = \{i \rightsquigarrow ak_i \mid i \in [1..n]\}$,
   $K_c = \{i \rightsquigarrow \sigma_{cl_i} \mid i \in [1..n]\}$,
5. $\sigma_w = \sigma_c \uplus (\biguplus_i K_c(i)) \uplus \sigma_s \uplus (\biguplus_i \sigma_i)$,
6. $\mathsf{wellFormedTokens}(J, \mathbb{K})$,
7. $W = \textbf{let}\ \Pi\ \textbf{in}\ C_1 \parallel \ldots \parallel C_n$,
   $S = (\sigma_w, \mathcal{K})$,
8. $\mathbb{W} = \textbf{with}\ \Gamma\ \textbf{do}\ \mathbb{C}_1 \parallel \ldots \parallel \mathbb{C}_n$,
   $\mathbb{S} = ((\sigma_c, J, K_c), \Sigma, \mathbb{K})$,
9. $\neg\mathsf{Abort}(\mathbb{W}, \mathbb{S})$,

then $(W, S) \preceq (\mathbb{W}, \mathbb{S})$*.*

Proof. By co-induction, we only need to prove following (1)(2):

(1) If $(W, S) \xrightarrow{e} \textbf{abort}$, then we need to prove there exist ξ, such that

(A) $(\mathbb{W}, \mathbb{S}) \xrightarrow{\xi}{}^{*} \textbf{abort}$,

(B) $\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi)$.

Proof: By low-level operational semantics, we know there exist i, such that:

$$(C_i, (\sigma_c \uplus \sigma_{cl_i} \uplus \sigma_s \uplus \sigma_i, \kappa_i)) \xrightarrow[i,\Pi]{e} \textbf{abort}$$

where $(C_i, (\sigma_c \uplus \sigma_{cl_i} \uplus \sigma_s \uplus \sigma_i, \kappa_i))$ is the local view of concrete resource from thread i.

From $(C_i, (\sigma_c \uplus \sigma_{cl_i} \uplus \sigma_s \uplus \sigma_i, \kappa_i), \Pi) \preceq^i_{R_i;G_i;I} (\mathbb{C}_i, ((\sigma_c, J, \sigma_{cl_i}), \Sigma, ak_i), \Gamma)$, we know there exists ξ, such that:

$$(\mathbb{C}_i, ((\sigma_c, J, \sigma_{cl_i}), \Sigma, ak_i)) \xrightarrow{\xi}{}^*_{i,\Gamma} \mathbf{abort} \wedge$$
$$\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi).$$

By high-level operational semantics, we get: $\exists \xi'$.

$$(\mathbf{with}\ \Gamma\ \mathbf{do}\ \mathbb{C}_1 \parallel \ldots \parallel \mathbb{C}_n, ((\sigma_c, J, K_c), \Sigma, \mathbb{K}))$$
$$\xrightarrow{\xi'}{}^* \mathbf{abort}$$

Thus we have: $(\mathbb{W}, \mathbb{S}) \xrightarrow{\xi'}{}^* \mathbf{abort}$, which contradict with the premise:

$$\neg \mathsf{Abort}(\mathbb{W}, \mathbb{S})$$

(2) If $(W, S) \longmapsto (W', S')$, we need to prove: there exist $\mathbb{W}', \mathbb{S}'$ and ξ, such that

(I) $(\mathbb{W}, \mathbb{S}) \xrightarrow{\xi}{}^* (\mathbb{W}', \mathbb{S}')$, and

(II) $\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi)$, and

(III) $(W', S') \preceq (\mathbb{W}', \mathbb{S}')$.

we have the following cases:

(a) If $C_1 = \ldots = C_n = \mathbf{skip}$.

Proof: Then $e = \tau$ (we use τ to denote an empty (silent) event), $W' = \mathbf{skip}$ and $S' = (\sigma_o, \kappa)$.

Thus we need to prove: there exist $\mathbb{W}', \mathbb{S}'$ and ξ, such that

(I) $(\mathbb{W}, \mathbb{S}) \xrightarrow{\xi}{}^* (\mathbb{W}', \mathbb{S}')$, and

(II) $\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi)$, and

(III) $(W', S') \preceq (\mathbb{W}', \mathbb{S}')$.

From thread simulations in premise, we know: for any i, $\mathbb{C}_i = \mathbf{skip}$ and $\kappa_i = \circ$ and $ak_i = \circ$.

Thus we get:

$$(\mathbb{W}, \mathbb{S}) \longrightarrow (\mathbf{skip}, \mathbb{S})$$

Let $\mathbb{S}' = \mathbb{S}$ and $\mathbb{W}' = \mathbf{skip}$.

Thus by definition, we know $(W', S') \preceq (\mathbb{W}', \mathbb{S}')$.

(b) If $(C_i, (\sigma_c \uplus \sigma_{cl_i} \uplus \sigma_s \uplus \sigma_i, \kappa_i)) \xrightarrow[i,\Pi]{e} (C'_i, (\sigma'_w, \kappa'_i))$.

Then we know one of the following holds:

i. If $(C_i, (\sigma_c \uplus \sigma_{cl_i} \uplus \sigma_s \uplus \sigma_i, \kappa_i)) \xrightarrow[i,\Pi]{e} \mathbf{abort}$, then by the thread simulation in the premise, we know

$$(\mathbb{C}_i, ((\sigma_c, J, \sigma_{cl_i}), \Sigma, ak_i)) \xrightarrow[i,\Gamma]{} \mathbf{abort}$$

which leads to a contradiction.

ii. If

$$(C_i, (\sigma_c \uplus \sigma_{cl_i} \uplus \sigma_s \uplus \sigma_i, \kappa_i))$$

$$\xrightarrow{e}_{i,\Pi}(C_i', (\sigma', \kappa_i')).$$

For this is the last case of the proof, we prove it outside.

The last case: If

$(C_i, (\sigma_c \uplus \sigma_{cl_i} \uplus \sigma_s \uplus \sigma_i, \kappa_i)) \xrightarrow{e}_{i,\Pi} (C_i', (\sigma', \kappa_i'))$,

then by the thread simulation of i, we know: there exist $\mathbb{C}_i'$, σ_{cl_i}', σ_c', σ_s', σ_i', J', Σ', ak_i' and ξ, such that:

(a.) $(\mathbb{C}_i, ((\sigma_c, J, \sigma_{cl_i}), \Sigma, ak_i))$
$\circ\xrightarrow{\xi}{}^*_{i,\Gamma} (\mathbb{C}_i', ((\sigma_c', J', \sigma_{cl_i}'), \Sigma', ak_i'))$, and

(b.) $\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi)$, and

(c.) $\sigma' = \sigma_c' \uplus \sigma_{cl_i}' \uplus \sigma_s' \uplus \sigma_i'$, and

(d.) $((\sigma_s', \emptyset), \Sigma') \models I$, and

(e.) $(((\sigma_s \uplus \sigma_i, \mathsf{getJ}(ak_i)), \Sigma), ((\sigma_s', \emptyset), \Sigma')) \models G_i * \mathsf{True}$, and

(f.) $(C_i', (\sigma_c' \uplus \sigma_{cl_i}' \uplus \sigma_s' \uplus \sigma_i', \kappa_i'), \Pi)$
$\preceq^i_{R_i;G_i;I} (\mathbb{C}_i', ((\sigma_c', J', \sigma_{cl_i}'), \Sigma', ak_i'), \Gamma)$.

(We omit the abort case here, for the case contradicts with the premise soon.)

By low-level language operational semantics, we know there exist $\mathcal{K}'$ and σ_w' such that

$$\begin{aligned}
&W' = \textbf{let } \Pi \textbf{ in } C_1 \parallel \ldots C_i' \ldots \parallel C_n \\
&\mathcal{K}' = \mathcal{K}\{i \rightsquigarrow \kappa_i'\} \\
&\sigma_w' = \sigma_c' \uplus (\biguplus_{j \neq i} \sigma_{cl_j}) \uplus \sigma_s' \uplus (\biguplus_{j \neq i} \sigma_j) \uplus \sigma_{cl_i}' \uplus \sigma_i' \\
&S' = (\sigma_w', \mathcal{K}')
\end{aligned}$$

From the high-level operational semantics, we know there exist K_c', $\mathbb{K}'$, $\mathbb{W}'$ and $\mathbb{S}'$, such that

$$\begin{aligned}
&K_c' = K_c\{i \rightsquigarrow \sigma_{cl_i}'\}, \\
&\mathbb{K}' = \mathbb{K}\{i \rightsquigarrow ak_i'\}, \\
&\mathbb{W}' = \textbf{with } \Gamma \textbf{ do } \mathbb{C}_1 \parallel \ldots \mathbb{C}_i' \ldots \parallel \mathbb{C}_n, \\
&\mathbb{S}' = ((\sigma_c', J', K_c'), \Sigma', \mathbb{K}'), \\
&(\mathbb{W}, \mathbb{S}) \circ\xrightarrow{\xi}{}^* (\mathbb{W}', \mathbb{S}').
\end{aligned}$$

Thus we only need to prove:

$$(W', S') \preceq (\mathbb{W}', \mathbb{S}')$$

The goal equals to:

$$\begin{aligned}
&(\textbf{let } \Pi \textbf{ in } C_1 \parallel \ldots C_i' \ldots \parallel C_n, \\
&\quad (\sigma_c' \uplus (\biguplus_{j \neq i} \sigma_{cl_j}) \uplus \sigma_s' \uplus (\biguplus_{j \neq i} \sigma_j) \uplus \sigma_{cl_i}' \uplus \sigma_i', \mathcal{K}'))
\end{aligned}$$

$$\preceq$$

$$(\textbf{with } \Gamma \textbf{ do } \mathbb{C}_1 \parallel \ldots \mathbb{C}'_i \ldots \parallel \mathbb{C}_n, ((\sigma'_c, J', K'_c), \Sigma', \mathbb{K}'))$$

By the co-induction hypothesis, we only need to prove the following cases:

(A) $\forall j \neq i.(C_j, (\sigma'_c \uplus \sigma_{cl_j} \uplus \sigma'_s \uplus \sigma_j, \kappa_j), \Pi) \preceq^j_{R_j;G_j;I} (\mathbb{C}_j, ((\sigma'_c, J', \sigma_{cl_j}), \Sigma', ak_j), \Gamma)$.

Proof: From those thread simulations, we know for any σ'_c, σ'_s, J' and Σ', if $\mathsf{EnvA}((\sigma_s \uplus \sigma_j, \mathsf{getJ}(ak_j), \Sigma), R_j, I)(\sigma'_s \uplus \sigma_j, \Sigma')$ and $(\sigma'_s, \Sigma') \models I$, then

$$((\sigma'_c \uplus \sigma_{cl_j} \uplus \sigma'_s \uplus \sigma_j, \kappa), \Pi)$$
$$\preceq^j_{R_j;G_j;I}$$
$$((\mathbb{C}_j, ((\sigma'_c, J', \sigma_{cl_j}), \Sigma', ak_j), \Gamma))$$

Thus we only need to prove:

$$\mathsf{EnvA}((\sigma_s \uplus \sigma_j, \mathsf{getJ}(ak_j), \Sigma), R_j, I)(\sigma'_s \uplus \sigma_j, \Sigma')$$

By the definition, we only need to prove: $\exists \sigma_a, J_a$.

$$\sigma_j \uplus \sigma_s = \sigma_j \uplus \sigma_s, \text{ and}$$
$$\sigma_j \uplus \sigma'_s = \sigma_j \uplus \sigma'_s, \text{ and}$$
$$(\sigma_s, \Sigma) \models I, \text{ and}$$
$$(\sigma'_s, \Sigma') \models I, \text{ and}$$
$$\sigma_j \perp \sigma_a, \text{ and}$$
$$\mathsf{getJ}(ak_j) \perp J_a, \text{ and}$$
$$(((\sigma_s \uplus \sigma_a, J_a), \Sigma), ((\sigma'_s, \emptyset), \Sigma')) \models R_j.$$

By (e.): $(((\sigma_s \uplus \sigma_i, \mathsf{getJ}(ak_i)), \Sigma), ((\sigma'_s, \emptyset), \Sigma')) \models G_i * \mathsf{True}$ and $(G_i \Rightarrow R_j)$, we know: $\exists \sigma_a, \sigma''_i, J_a, J''_i$,

$$\sigma_i = \sigma_a \uplus \sigma''_i,$$
$$\mathsf{getJ}(ak_i) = J_a \uplus J''_i,$$
$$(((\sigma_s \uplus \sigma_a, J_a)), \Sigma), ((\sigma'_s, \emptyset), \Sigma') \models R_j.$$

Thus we know we only need to prove:

$$\mathsf{getJ}(ak_j) \perp \mathsf{getJ}(ak_i)$$

By $\mathsf{WellFormedTaskPool}(J, \mathbb{K})$, we prove it.

(B) $(C'_i, (\sigma'_c \uplus \sigma'_{cl_i} \uplus \sigma'_s \uplus \sigma'_i, \kappa'_i), \Pi) \preceq^i_{R_i;G_i;I} (\mathbb{C}'_i, ((\sigma'_c, J', \sigma'_{cl_i}), \Sigma', ak'_i), \Gamma)$.

Proof: By (f.).

(C) $((\sigma'_s, \emptyset), \Sigma') \models I$.

Proof: By (d.).

(D) $(\forall i.\forall j.j \neq i \implies G_i \Rightarrow R_j)$, $I \rhd \{R_i, G_i\}$.

Proof: By the premise.

(E) $\mathcal{K}' = \{j \rightsquigarrow \kappa_j \mid j \in [1..n] \wedge j \neq i\} \uplus \{i \rightsquigarrow \kappa_i'\}$,
$\mathbb{K}' = \{j \rightsquigarrow ak_j \mid j \in [1..n] \wedge j \neq i\} \uplus \{i \rightsquigarrow ak_i'\}$, and
$K_c' = \{j \rightsquigarrow \sigma_{cl_j} \mid j \in [1..n] \wedge j \neq i\} \uplus \{i \rightsquigarrow \sigma'_{cl_i}\}$.

Proof: Trivial.

(F) $\sigma_w' = \sigma_c' \uplus ((\biguplus_{j \neq i} \sigma_{cl_j}) \uplus \sigma'_{cl_i}) \uplus \sigma_s' \uplus ((\biguplus_{j \neq i} \sigma_j) \uplus \sigma_i')$.

Proof: Trivial.

(G) $\mathsf{wellFormedTokens}(J', \mathbb{K}')$.

Proof: By the definition, We only need to prove:

$$\exists J_0'.J_0' = J' \uplus (\biguplus_{j \neq i} \mathsf{getJ}(ak_j)) \uplus \mathsf{getJ}(ak_i')$$

By premise 6, we know:

$$\exists J_0.J_0 = J \uplus (\biguplus_{j \neq i} \mathsf{getJ}(ak_j)) \uplus \mathsf{getJ}(ak_i)$$

By (a.), we know:

$$(\mathbb{C}_i, ((\sigma_c, J, \sigma_{cl_i}, \Sigma, ak_i))) \xrightarrow{\xi}{}^{*}_{i,\Gamma} (\mathbb{C}_i', ((\sigma_c', J', \sigma'_{cl_i}), \Sigma', ak_i'))$$

By the specification language operational semantics, we can prove it.

(H) $W' = \mathbf{let}\ \Pi\ \mathbf{in}\ C_1 \parallel \ldots C_i' \ldots \parallel C_n$ and
$S' = (\sigma_w', \mathcal{K}')$.

Proof: Trivial.

(I) $\mathbb{W}' = \mathbf{with}\ \Gamma\ \mathbf{do}\ \mathbb{C}_1 \parallel \ldots \mathbb{C}_i' \ldots \parallel \mathbb{C}_n$ and
$\mathbb{S}' = ((\sigma_c', J', K_c'), \Sigma', \mathbb{K}')$.

Proof: Trivial.

(J) $\neg\mathsf{Abort}(\mathbb{W}', \mathbb{S}')$.

Proof: Trivial.

Thus we are done. □

11.1.3 Simulation for Program Implies Refinement

Lemma 15 (Simulation for Program Implies Refinement)**.** *For any $W, \mathbb{W}$ and φ, if $W \preceq_\varphi \mathbb{W}$, then*

$$\begin{array}{l} \forall \sigma_o, \Sigma, K_c, \sigma_c. \\ (\sigma_o, \Sigma) \in \varphi \implies \\ \mathcal{O}[\![W, \mathsf{init}(\sigma_c, K_c, \sigma_o)]\!] \subseteq \mathcal{O}[\![\mathbb{W}, \mathsf{init}(\sigma_c, K_c, \Sigma)]\!]. \end{array}$$

Proof. Immediate by applying the following Lemma 16. □ □

Lemma 16. *If $(W, S) \preceq (\mathbb{W}, \mathbb{S})$, then $\mathcal{O}[\![W, S]\!] \subseteq \mathcal{O}[\![\mathbb{W}, \mathbb{S}]\!]$.*

Proof. For any ξ such that $\xi \in \mathcal{O}[\![W, S]\!]$, we know one of the following holds:

1. There exists W', S' and ξ', such that $(W, S) \overset{\xi'}{\longmapsto}{}^{*} (W', S')$ and $\mathsf{get\_obsv}(\xi) = \mathsf{get\_obsv}(\xi')$.

   By the following Lemma 17, we know, there exist $\mathbb{W}', \mathbb{S}'$ and ξ'', such that:

   $$(\mathbb{W}, \mathbb{S}) \overset{\xi''}{\circ\!\!\longrightarrow}{}^{*} (\mathbb{W}', \mathbb{S}'),$$
   $$\mathsf{get\_obsv}(\xi') = \mathsf{get\_obsv}(\xi''),$$
   $$(W', S') \preceq (\mathbb{W}', \mathbb{S}').$$

   Thus we know $\mathsf{get\_obsv}(\xi) \in \mathcal{O}[\![\mathbb{W}, \mathbb{S}]\!]$.

2. There exist W', S' and ξ', such that $(W, S) \overset{\xi'}{\longmapsto}{}^{*} (W', S')$, $(W', S') \overset{e}{\longmapsto} \mathbf{abort}$, and $\mathsf{get\_obsv}(\xi) = \mathsf{get\_obsv}(\xi' :: e)$

   By the following Lemma 17, we know, there exist $\mathbb{W}', \mathbb{S}'$ and ξ'', such that:

   $$(\mathbb{W}, \mathbb{S}) \overset{\xi''}{\circ\!\!\longrightarrow}{}^{*} (\mathbb{W}', \mathbb{S}')$$
   $$\mathsf{get\_obsv}(\xi') = \mathsf{get\_obsv}(\xi''),$$
   $$(W', S') \preceq (\mathbb{W}', \mathbb{S}').$$

   Also we know, there exists ξ''' such that $(\mathbb{W}', \mathbb{S}') \overset{\xi'''}{\circ\!\!\longrightarrow} \mathbf{abort}$ and $\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi''')$.

   Thus we have:

   $$(\mathbb{W}, \mathbb{S}) \overset{\xi'' :: \xi'''}{\circ\!\!\longrightarrow}{}^{*} \mathbf{abort},$$
   $$\mathsf{get\_obsv}(\xi'' :: \xi''') = \mathsf{get\_obsv}(\xi).$$

   Thus $\xi \in \mathcal{O}[\![\mathbb{W}, \mathbb{S}]\!]$.

Thus we get the conclusion. □

Lemma 17. *For any n, if* $(W, S) \overset{\xi}{\longmapsto}{}^{n} (W', S')$ *and* $(W, S) \preceq (\mathbb{W}, \mathbb{S})$*, then there exist* $\mathbb{W}', \mathbb{S}'$ *and* ξ'*, such that* $(\mathbb{W}, \mathbb{S}) \overset{\xi'}{\circ\!\!\longrightarrow}{}^{*} (\mathbb{W}', \mathbb{S}')$*,* $\mathsf{get\_obsv}(\xi) = \mathsf{get\_obsv}(\xi')$ *and* $(W', S') \preceq (\mathbb{W}', \mathbb{S}')$.

Proof. By induction over n.

Base case: $n = 0$. Thus $W' = W$, $S' = S$ and $\mathsf{get\_obsv}(\epsilon) = \epsilon$. We take $\mathbb{W}' = \mathbb{W}$ and $\mathbb{S}' = \mathbb{S}$.

Inductive step: $n = k + 1$.

Thus exist W'', S'', ξ_1 and e such that:

$$(W, S) \overset{e}{\longmapsto} (W'', S''),$$
$$(W'', S'') \overset{\xi_1}{\longmapsto}{}^{k} (W', S').$$

And $\xi = e :: \xi_1$.

By $(W, S) \preceq (\mathbb{W}, \mathbb{S})$, we know there exist $\mathbb{W}'', \mathbb{S}''$ and ξ'' such that

$$(\mathbb{W}, \mathbb{S}) \overset{\xi''}{\circ\!\!\longrightarrow}{}^{*} (\mathbb{W}'', \mathbb{S}''),$$

$$\mathsf{get\_obsv}(e) = \mathsf{get\_obsv}(\xi''),$$
$$(W'', S'') \preceq (\mathbb{W}'', \mathbb{S}'').$$

By the induction hypothesis, we know there exist $\mathbb{W}', \mathbb{S}'$ and ξ_1'', such that:

$$(W', S') \preceq (\mathbb{W}', \mathbb{S}'),$$
$$(\mathbb{W}'', \mathbb{S}'') \circ\!\!\xrightarrow{\xi_1'}{}^{*} (\mathbb{W}', \mathbb{S}'),$$
$$\mathsf{get\_obsv}(\xi_1') = \mathsf{get\_obsv}(\xi_1).$$

Thus we know $(\mathbb{W}, \mathbb{S}) \circ\!\!\xrightarrow{\xi'' :: \xi_1'*} (\mathbb{W}', \mathbb{S}')$, $\mathsf{get\_obsv}((\xi'' :: \xi_1')) = \mathsf{get\_obsv}(e :: \xi_1)$ and $(W', S') \preceq (\mathbb{W}', \mathbb{S}')$. □

11.2 Proofs for Lemma 2(Logic Judgment Implies Simulation for Method)

Below we first define the rely-guarantee-style judgment semantics. We derive the simulation for method (Definition 5) from the judgment semantics, and then prove that all inference rules in Figure 17 are sound *w.r.t.* this semantics. Some auxiliary definition is defined in Fig 34.

$$\begin{array}{ll}
(\Theta, \Theta') \models R & \mathsf{iff}\ \exists \sigma, \sigma_t, J, \Sigma, \mu, \sigma', \sigma_t', J', \Sigma', \mu'. \\
& \quad \Theta = (\sigma, (\sigma_t, J, \Sigma, \mu)) \wedge \\
& \quad \Theta' = (\sigma', (\sigma_t', J', \Sigma', \mu')) \wedge \\
& \quad (((\sigma, J), \Sigma), ((\sigma', \emptyset), \Sigma)) \models R \\
\mathsf{EnvA}(\Theta, R, I)(\Theta') & \mathsf{iff}\ \exists \sigma, \sigma_t, J, \Sigma, \mu, \sigma', \Sigma'. \\
& \quad \Theta = (\sigma, (\sigma_t, J, \Sigma, \mu)) \wedge \\
& \quad \Theta' = (\sigma', (\sigma_t, J, \Sigma', \mu)) \wedge \\
& \quad \mathsf{Env}_I^R(\sigma, J, \Sigma)(\sigma', \Sigma')
\end{array}$$

Fig.34. Auxiliary Definitions for Logic Rule Sound

11.2.1 Derive Simulation from Semantics of Judgments

Definition 8 (Semantics of Sequential Judgment)**.** $\models \{p\}C\{q\}$ *iff,*
for any $\sigma, \sigma_t, J, \Sigma$ *and* μ*, if* $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models p$*, followings are true:*

1. *for any* σ'*, if* $(C, \sigma) \bullet\!\!\longrightarrow^{*} (\mathbf{skip}, \sigma')$ *, then there exist* σ_t', J', Σ' *and* μ'*, such that*
   $(\mu, (\sigma_t, J, \Sigma)) \rightrightarrows^{*} (\mu', (\sigma_t', J', \Sigma'))$ *and*
   $(\sigma', (\sigma_t', J', \Sigma', \mu')) \models q$*;*
2. $\neg(\ (C, \sigma) \bullet\!\!\longrightarrow^{*} \mathbf{abort}\ )$.

Definition 9 (Semantics of Rely-Guarantee-Style Judgment)**.** $R, G, I \models \{p\}\ C\ \{q\}$ *iff,*
for any $\sigma, \sigma_t, J, \Sigma$ *and* μ*, if* $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models p$*,*
then $(C, \sigma) \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$.

Our logic is sound *w.r.t.* this semantics, as shown in the following theorem. We will prove this theorem in Appendix 11.2.2

Theorem 2 (Logic Soundness as Rely-Guarantee Reasoning)**.** *If* $R, G, I \vdash \{p\}\ C\ \{q\}$*, then* $R, G, I \models \{p\}\ C\ \{q\}$*.*

The proof for above theorem is shown in Sec 11.2.2. To some extend, this theorem is an another form of Lemma 2. We show the fact in the following proof.

We prove Lemma 2 as following:

Proof. By Theorem 2(logic soundness as rely-guarantee reasoning), we know:

$$\begin{aligned} R, G, I \models & \{\langle \mathbb{s}, n\rangle * (x \mapsto \_) * \mathsf{lift}(trq(n, n', \overrightarrow{v})) * I\} \\ & C \\ & \{\exists n'.\langle \mathbf{skip}, n'\rangle * (x \mapsto \_) * \mathsf{lift}(trq(n, n', \overrightarrow{v})) * I\} \end{aligned}$$

Then by the following Lemma 18(from judgment semantics to simulation), we know $(x, C) \preceq^{R;G;I}_{trp;trq} \mathbb{s}$. □

Lemma 18 (From Judgment Semantics to Simulation)**.** *For any* $C, R, G, I, trp, trq, x, \mathbb{s}$*,*

if followings are all true:

1. *for any* n *and* $\overrightarrow{v}$ *:*

$$\begin{aligned} R, G, I \models & \{\langle \mathbb{s}, n\rangle * x \mapsto n * \mathsf{lift}(trp(n, \overrightarrow{v})) * I\} \\ & C \\ & \{\exists n'.\langle \mathbf{skip}, n'\rangle * (x \mapsto \_) * \mathsf{lift}(trq(n, n', \overrightarrow{v})) * I\}, \end{aligned}$$

2. $x \notin dom(I)$ *and* $I \rhd \{R, G\}$*,*

then $(x, C) \preceq^{R;G;I}_{trp;trq} \mathbb{s}$*.*

Proof. This proof is trivial, for the only thing we need to do is to unfold definitions of judgment semantics and $(x, C) \preceq^{R;G;I}_{trp;trq} \mathbb{s}$. The detailed proof is shown below:

We need to prove: for any $\sigma, \sigma_t, J, \Sigma, \mu, n, \overrightarrow{v}$ and q, if $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models \langle \mathbb{s}, n\rangle * (x \mapsto n) * \mathsf{lift}(trp(n, \overrightarrow{v})) * I$ and $q = \exists n'.\langle \mathbf{skip}, n'\rangle * (x \mapsto \_) * \mathsf{lift}(trq(n, n', \overrightarrow{v}))$, then $(C, \sigma) \preceq^{R;G;I}_{q} (\mu, (\sigma_t, J, \Sigma))$.

By definition of judgment semantics, we know for any $\sigma, \sigma_t, J, \Sigma$ and μ, if $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models p$, then $(C, \sigma) \preceq^{R;G;I}_{q} (\mu, (\sigma_t, J, \Sigma))$, where

$q = \exists n'.\langle \mathbf{skip}, n'\rangle * (x \mapsto \_) * \mathsf{lift}(trq(n, n', \overrightarrow{v})) * I$ and

$p = \langle \mathbb{s}, n\rangle * x \mapsto n * \mathsf{lift}(trp(n, \overrightarrow{v})) * I$.

Thus we get $(x, C) \preceq^{R;G;I}_{trp;trq} \mathbb{s}$. □

11.2.2 Soundness of Inference Rules

Theorem 2 (logic soundness *w.r.t.* the rely-guarantee-style semantics) is proved by induction over the derivation of the judgment $R, G, I \vdash \{p\}\ C\ \{q\}$. The whole proof consists of the soundness proof for each individual rules. Here we show main lemmas used to prove the soundness of ATOM-R(Lemma 19), ATOM(Lemma 20), P-CONSEQ(Lemma 22), P-IF(Lemma 24), P-WHILE(Lemma 27), P-SEQ(Lemma 25) and RET(Lemma 29), including all rules in Fig 17.

The ATOM-R Rule.

Lemma 19 (ATOM-R-Sound)**.** *For any R, G, I, p, C and q, if followings are all true:*

1. $[I], G, I \models \{p\}\ \langle C\rangle\ \{q\}$
2. $\mathsf{Sta}_I(\{p, q\}, R)$,
3. $I \rhd \{R, G\}$,

then $R, G, I \models \{p\}\ \langle C\rangle\ \{q\}$.

Proof. Below we prove: for any $\sigma, \sigma_t, J, \Sigma$ and μ, if $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models p$, then

$$(C, \sigma) \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$$

By co-induction, we only need to prove following (1)(2)(3)(4)(5) (by definition of method simulation).

(1) For any C' and σ',

if $(\langle C\rangle, \sigma) \bullet\!\!\longrightarrow (C', \sigma')$,

then there exist μ', σ_t', J' and Σ', such that

(a) $(\mu, (\sigma_t, J, \Sigma)) \rightrightarrows^* (\mu', (\sigma_t', J', \Sigma'))$, and

(b) $(((\sigma, J), \Sigma), ((\sigma', \emptyset), \Sigma')) \models G * \mathsf{True}$, and

(c) $(C', \sigma') \preceq_q^{R;G;I} (\mu', (\sigma_t', J', \Sigma'))$.

Proof: By operational semantics, we know C' must be **skip** and

$$(C, \sigma) \bullet\!\!\longrightarrow (\mathbf{skip}, \sigma')$$

From $[I], G, I \models \{p\}\ \langle C\rangle\ \{q\}$, we know: there exist μ', σ_t', J' and Σ' such that

(A) $(\mu, (\sigma_t, J, \Sigma)) \rightrightarrows^* (\mu', (\sigma_t', J', \Sigma'))$, and

(B) $((\sigma, J), \Sigma), ((\sigma', \emptyset), \Sigma') \models G * \mathsf{True}$, and

(C) $(\mathbf{skip}, \sigma') \preceq_q^{[I],G,I} (\mu', \sigma_t', J', \Sigma')$.

Thus we know we only need to prove:

$$(\mathbf{skip}, \sigma') \preceq_q^{R;G;I} (\mu', (\sigma_t', J', \Sigma'))$$

By $(\mathbf{skip}, \sigma') \preceq_q^{[I],G,I} (\mu', \sigma_t', J', \Sigma')$, we prove it.

(2) If $\langle C\rangle \neq \mathbf{E}[\mathbf{return}\ E]$,

then $\neg(\ (\langle C\rangle, \sigma) \bullet\!\!\longrightarrow \mathbf{abort}\ )$.

Proof: By $[I], G, I \models \{p\}\ \langle C\rangle\ \{q\}$ in premise, we know $(\langle C\rangle, \sigma) \preceq_q^{[I];G;I} (\mu, (\sigma_t, J, \Sigma))$. Then by the fact $(\langle C\rangle \neq \mathbf{E}[\mathbf{return}\ E])$, we get $\neg(\ (\langle C\rangle, \sigma) \bullet\!\!\longrightarrow \mathbf{abort}\ )$.

(3) For any σ' and Σ', if $\mathsf{Env}_I^R(((\sigma, J), \Sigma))(((\sigma', J), \Sigma'))$, then $(\langle C\rangle, \sigma') \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma'))$.

Proof: From $\mathsf{Sta}_I(p, R)$ and

$\mathsf{Env}_I^R(((\sigma, J), \Sigma))(((\sigma', J), \Sigma'))$, we know:

$$(\sigma', (\sigma_t, J, \Sigma', \mu)) \models p$$

By the co-induction hypothesis, we need to prove:

(a) $[I], G, I \models \{p\}\ \langle C\rangle\ \{q\}$

Proof: By the identical premise.

(b) $\mathsf{Sta}_I(\{p, q\}, R)$

Proof: By the identical premise.

(c) $I \rhd \{R, G\}$

Proof: By the identical premise.

(d) $(\sigma', (\sigma_t, J, \Sigma', \mu)) \models p$

Proof: By the definition of stability, from $\mathsf{Sta}_I(p, R)$ and $\mathsf{Env}_I^R(((\sigma, J), \Sigma))(((\sigma', J), \Sigma'))$, we know:

$$(\sigma', (\sigma_t, J, \Sigma', \mu)) \models p.$$

(4) If $\langle C\rangle = \mathbf{E}[\mathbf{return}\ E]$, then ...

Proof: It is vacuously true.

(5) If $\langle C\rangle = \mathbf{skip}$, then ...

Proof: It is vacuously true.

Thus we are done. □

The ATOM Rule.

Lemma 20 (ATOM-Rule-Sound)**.** *For any I, G, p, C and q, if followings are true:*

1. $\vdash \{p\}\ C\ \{q\}$,
2. $p \ltimes q \Rightarrow G * \mathsf{True}$,
3. $p \vee q \Rightarrow I * \mathsf{true}$,
4. $I \rhd G$,

then $[I], G, I \models \{p\}\ \langle C\rangle\ \{q\}$.

Proof. By Lemma 21(from sequential jugments to semantics), we know:

$$\models \{p\}C\{q\}$$

Below we prove: for any $\sigma, \sigma_t, J, \Sigma$ and μ, if $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models p$, then

$$(\langle C\rangle, \sigma) \preceq_q^{[I],G,I} (\mu, (\sigma_t, J, \Sigma))$$

By co-induction, we only need to prove following (1)(2)(3)(4)(5).

(1) For any C' and σ',

if $(\langle C\rangle, \sigma) \bullet\!\!\longrightarrow (C', \sigma')$,

then there exist μ', σ_t', J' and Σ', such that

(a) $(\mu, (\sigma_t, J, \Sigma)) \rightrightarrows^* (\mu', (\sigma_t', J', \Sigma'))$, and

(b) $(((\sigma, J), \Sigma), ((\sigma', \emptyset), \Sigma')) \models G * \mathsf{True}$, and

(c) $(C', \sigma') \preceq_q^{[I],G,I} (\mu', (\sigma_t', J', \Sigma'))$.

Proof: By operational semantics, we know C' must be **skip** and

$$(C, \sigma) \bullet\!\!\longrightarrow (\mathbf{skip}, \sigma')$$

From $\models \{p\}C\{q\}$, we know: there exist μ', σ'_t, J' and Σ' such that

(A) $(\mu, (\sigma_t, J, \Sigma)) \rightrightarrows^* (\mu', (\sigma'_t, J', \Sigma'))$, and

(B) $(\sigma', (\sigma'_t, J', \Sigma', \mu')) \models q$.

By the relation between actions $p \ltimes q \Rightarrow G * \mathsf{True}$, we know:

$$(((\sigma, J), \Sigma), ((\sigma', \emptyset), \Sigma')) \models G * \mathsf{True}$$

Thus we know we only need to prove:

$$(C', \sigma') \preceq_q^{[I],G,I} (\mu', (\sigma'_t, J', \Sigma'))$$

From $C' = \mathbf{skip}$ and $(\sigma', (\sigma'_t, J', \Sigma', \mu')) \models q$, we know:

$$(C', \sigma') \preceq_q^{[I],G,I} (\mu', (\sigma'_t, J', \Sigma'))$$

(2) If $\langle C\rangle \neq \mathbf{E}[\mathbf{return}\ E]$,

then $\neg(\ (\langle C\rangle, \sigma) \bullet\!\!\longrightarrow \mathbf{abort}\ )$.

Proof: By $\models \{p\}C\{q\}$.

(3) For any σ' and Σ', if $\mathsf{Env}_I^{[I]}(((\sigma, J), \Sigma))(((\sigma', J), \Sigma'))$, then $(\langle C\rangle, \sigma') \preceq_q^{[I],G,I} (\mu, (\sigma_t, J, \Sigma'))$.

Proof: For the rely condition $[I]$ is the identical action, we know $\sigma' = \sigma$ and $\Sigma' = \Sigma$.

By the co-induction hypothesis, we get

$$(\langle C\rangle, \sigma) \preceq_q^{[I],G,I} (\mu, (\sigma_t, J, \Sigma))$$

Using co-induction hypothesis in this way is correct for the example in [23] proving ATOM rule at page 30.

(4) If $\langle C\rangle = \mathbf{E}[\mathbf{return}\ E]$, then ...

Proof: It is vacuously true.

(5) If $\langle C\rangle = \mathbf{skip}$, then ...

Proof: It is vacuously true.

Thus we are done. □

Lemma 21 (From Sequential Jugments to Semantics)**.** *For any p, C and q, if $\vdash \{p\}\ C\ \{q\}$, then we have $\models \{p\}C\{q\}$.*

Proof. By induction over the derivation of the judgment $\vdash \{p\}\ C\ \{q\}$. □

The Conseq Rule.

Lemma 22 (P-Conseq-rule)**.** *For any p, C, q, p', I, R and q', if*

1. $R, G, I \models \{p\}\ C\ \{q\}$,
2. $p' \Rightarrow p$,
3. $q \Rightarrow q'$,

then $R, G, I \models \{p'\}\ C\ \{q'\}$.

Proof. Below we prove: for any $\sigma, \sigma_t, J, \Sigma$ and μ, if $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models p'$, then

$$(C, \sigma) \preceq_{q'}^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$$

By $p' \Rightarrow p$, we get:

$$(\sigma, (\sigma_t, J, \Sigma, \mu)) \models p$$

Then by $R, G, I \models \{p\}\ C\ \{q\}$ and $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models p$, we know:

$$(C, \sigma) \preceq_{q}^{R;G;I} (\mu, (\sigma_t, J, \Sigma)) \tag{*14.1}$$

By (*14.1), $q \Rightarrow q'$, and Lemma 23, we have:

$$(C, \sigma) \preceq_{q'}^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$$

□

Lemma 23. *For any* $C, \sigma, \sigma, \mu, \sigma_t, J, \Sigma, q$ *and* q', *if*

1. $(C, \sigma) \preceq_{q}^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$,
2. $q \Rightarrow q'$,

then $(C, \sigma) \preceq_{q'}^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$.

Proof. By co-induction, we only need to prove following (1)(2)(3)(4)(5).

(1) For any C' and σ',

if $(C, \sigma) \bullet\!\!\longrightarrow (C', \sigma')$,

then there exist μ', σ_t', J' and Σ', such that

(a) $(\mu, (\sigma_t, J, \Sigma)) \rightrightarrows^* (\mu', (\sigma_t', J', \Sigma'))$, and

(b) $(((\sigma, J), \Sigma), ((\sigma', \emptyset), \Sigma')) \models G * \mathsf{True}$, and

(c) $(C', \sigma') \preceq_{q'}^{R;G;I} (\mu', (\sigma_t', J', \Sigma'))$.

Proof: From $(C, \sigma) \preceq_{q}^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$, we know: there exist μ', σ_t', J' and Σ' such that

(A) $(\mu, (\sigma_t, J, \Sigma)) \rightrightarrows^* (\mu', (\sigma_t', J', \Sigma'))$, and

(B) $((\sigma, J), \Sigma), ((\sigma', \emptyset), \Sigma') \models G * \mathsf{True}$, and

(C) $(C', \sigma') \preceq_{q}^{R;G;I} (\mu', \sigma_t', J', \Sigma')$.

Thus we know we only need to prove:

$$(C', \sigma') \preceq_{q'}^{R;G;I} (\mu', (\sigma_t', J', \Sigma'))$$

By the co-induction hypothesis, we get it.

(2) If $C \neq \mathbf{E}[\textbf{return } E]$,

then $\neg(\ (C,\sigma) \bullet\!\!\longrightarrow \textbf{abort}\ )$.

Proof: By $(C,\sigma) \preceq_q^{R;G;I} (\mu,(\sigma_t,J,\Sigma))$.

(3) For any σ' and Σ', if $\mathsf{Env}_I^R(((\sigma,J),\Sigma))(((\sigma',J),\Sigma'))$, then $(C,\sigma') \preceq_{q'}^{R;G;I} (\mu,(\sigma_t,J,\Sigma'))$.

Proof: By $(C,\sigma) \preceq_q^{R;G;I} (\mu,(\sigma_t,J,\Sigma))$, we get:

$$(C,\sigma') \preceq_q^{R;G;I} (\mu,(\sigma_t,J,\Sigma'))$$

By the co-induction hypothesis, we get:

$$(C,\sigma') \preceq_{q'}^{R;G;I} (\mu,(\sigma_t,J,\Sigma'))$$

(4) If $C = \mathbf{E}[\textbf{return } E]$, then there exist n', such that

(a) $[\![E]\!]_\sigma = n'$, and

(b) $\mu = (\textbf{skip}, n')$, and

(c) $(\sigma,(\sigma_t,J,\Sigma,\mu)) \models q'$.

Proof: By $(C,\sigma) \preceq_q^{R;G;I} (\mu,(\sigma_t,J,\Sigma))$, we know:

(a) $[\![E]\!]_\sigma = n'$, and

(b) $\mu = (\textbf{skip}, n')$, and

(c) $(\sigma,(\sigma_t,J,\Sigma,\mu)) \models q$.

By $q \Rightarrow q'$, we get:

$$(\sigma,(\sigma_t,J,\Sigma,\mu)) \models q'$$

(5) If $C = \textbf{skip}$, then $(\sigma,(\sigma_t,J,\Sigma,\mu)) \models q'$.

Proof: By $(C,\sigma) \preceq_q^{R;G;I} (\mu,(\sigma_t,J,\Sigma))$, we know:

$$(\sigma,(\sigma_t,J,\Sigma,\mu)) \models q$$

By $q \Rightarrow q'$, we get:

$$(\sigma,(\sigma_t,J,\Sigma,\mu)) \models q'$$

Thus we are done. □

The P-IF Rule.

Lemma 24 (P-IF-Rule)**.** *For any R, G, I, p, B, q, C_1 and C_2, if*

1. $R,G,I \models \{p * B\}\ C_1\ \{q\}$,
2. $R,G,I \models \{p * \neg B\}\ C_2\ \{q\}$,
3. $p \Rightarrow I$,

4. $\mathsf{Sta}_I(p, R)$,

then $R, G, I \models \{p * (B = B)\}$ (**if** B C_1 **else** C_2) $\{q\}$.

Proof. Below we prove: for any $\sigma, \sigma_t, J, \Sigma$ and μ, if $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models p$, then

$$(\textbf{if } B \ C_1 \textbf{ else } C_2, \sigma) \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$$

By co-induction, we only need to prove following (1)(2)(3)(4)(5).

(1) For any C' and σ',

if $(\textbf{if } B \ C_1 \textbf{ else } C_2, \sigma) \bullet\!\!\longrightarrow (C', \sigma')$,

then there exist μ', σ'_t, J' and Σ', such that

(a) $(\mu, (\sigma_t, J, \Sigma)) \rightrightarrows^* (\mu', (\sigma'_t, J', \Sigma'))$, and

(b) $(((\sigma, J), \Sigma), ((\sigma', \emptyset), \Sigma')) \models G * \mathsf{True}$, and

(c) $(C', \sigma') \preceq_q^{R;G;I} (\mu', (\sigma'_t, J', \Sigma'))$.

Proof: By the operational semantics, we have following cases:

(A) If $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models B$.

Proof: By operational semantics, we know $C' = C_1$ and $\sigma' = \sigma$.

Let $\mu' = \mu$, $\sigma'_t = \sigma_t$, $J' = J$ and $\Sigma' = \Sigma$.

By $R, G, I \models \{p\}\ C_1\ \{q\}$, we get $(C_1, \sigma) \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$.

(B) If $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models \neg B$.

Proof: By operational semantics, we know $C' = C_2$ and $\sigma' = \sigma$.

Let $\mu' = \mu$, $\sigma'_t = \sigma_t$, $J' = J$ and $\Sigma' = \Sigma$.

By $R, G, I \models \{p\}\ C_2\ \{q\}$, we get $(C_2, \sigma) \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$.

(2) If **if** B C_1 **else** $C_2 \neq \mathbf{E}[\textbf{return } E]$,

then $\neg($ (**if** B C_1 **else** C_2, σ) $\bullet\!\!\longrightarrow$ **abort**).

Proof: By $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models p * (B = B)$, we get it.

(3) For any σ' and Σ', if $\mathsf{Env}_I^R((\sigma, J), \Sigma)((\sigma', J), \Sigma')$, then (**if** B C_1 **else** $C_2, \sigma') \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma'))$.

Proof: From $\mathsf{Sta}_I(p, R)$ and

$\mathsf{Env}_I^R((\sigma, J), \Sigma)((\sigma', J), \Sigma')$, we know:

$$(\sigma', (\sigma_t, J, \Sigma', \mu)) \models p$$

By the co-induction hypothesis, we get:

$$(\textbf{if } B \ C_1 \textbf{ else } C_2, \sigma') \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma'))$$

(4) If **if** B C_1 **else** $C_2 = \mathbf{E}[\textbf{return } E]$, then ...

Proof: It is vacuously true.

(5) If **if** B C_1 **else** $C_2 =$ **skip**, then ...

Proof: It is vacuously true.

Thus we are done. □

The P-SEQ Rule.

Lemma 25 (P-SEQ-Rule)**.** *For any R, G, I, C_1, C_2, p, r and q, if*

1. $R, G, I \models \{p\}\ C_1\ \{r\}$,
2. $R, G, I \models \{r\}\ C_2\ \{q\}$,

then $R, G, I \models \{p\}\ C_1; C_2\ \{q\}$.

Proof. Below we prove: for any $\sigma, \sigma_t, J, \Sigma$ and μ, if $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models p$, then

$$(C_1; C_2, \sigma) \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$$

By $R, G, I \models \{p\}\ C_1\ \{r\}$, we know:

$$(C_1, \sigma) \preceq_r^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$$

From Lemma 26, we get:

$$(C_1; C_2, \sigma) \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$$

□

Lemma 26. *For any $C_1, C_2, \sigma, \mu, \sigma_t, J, \Sigma, R, G, I, q$ and r, if*

1. $(C_1, \sigma) \preceq_r^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$,
2. $R, G, I \models \{r\}\ C_2\ \{q\}$,

then $(C_1; C_2, \sigma) \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$.

Proof. By co-induction, we only need to prove following (1)(2)(3)(4)(5).

(1) For any C' and σ',

if $(C_1; C_2, \sigma) \bullet\!\!\longrightarrow (C', \sigma')$,

then there exist μ', σ_t', J' and Σ', such that

(a) $(\mu, (\sigma_t, J, \Sigma)) \rightrightarrows^* (\mu', (\sigma_t', J', \Sigma'))$, and

(b) $(((\sigma, J), \Sigma), ((\sigma', \emptyset), \Sigma')) \models G * \mathsf{True}$, and

(c) $(C', \sigma') \preceq_q^{R;G;I} (\mu', (\sigma_t', J', \Sigma'))$.

Proof: By the operational semantics, we have following cases:

(A) $C_1 \neq \mathbf{skip}$.

Proof: Then we know $C' = C_1'; C_2$ and

$$(C_1, \sigma) \bullet\!\!\longrightarrow (C_1', \sigma')$$

By $(C_1, \sigma) \preceq_r^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$, we know: there exist μ', σ_t', J' and Σ',

- $(\mu, (\sigma_t, J, \Sigma)) \rightrightarrows^* (\mu', (\sigma_t', J', \Sigma'))$, and
- $((\sigma, J), \Sigma), ((\sigma', \emptyset), \Sigma') \models G * \mathsf{True}$, and
- $(C_1', \sigma') \preceq_r^{R;G;I} (\mu', \sigma_t', J', \Sigma')$.

By the co-induction hypothesis, we get

$$(C_1'; C_2, \sigma') \preceq_q^{R;G;I} (\mu', (\sigma_t', J', \Sigma'))$$

(B) $C_1 = \textbf{skip}$.

Proof: Then we know $C' = C_2$, $\sigma' = \sigma$, $\mu' = \mu$, $\sigma_t' = \sigma_t$ and $\Sigma' = \Sigma$, and we need to prove:

$$(C_2, \sigma) \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$$

By $(\textbf{skip}, \sigma) \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$, we know:

$$(\sigma, (\sigma_t, J, \Sigma, \mu)) \models r$$

By $R, G, I \models \{r\}\ C_2\ \{q\}$, we get:

$$(C_2, \sigma) \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$$

(2) If $C_1; C_2 \neq \mathbf{E}[\textbf{return}\ E]$,
then $\neg(\ (C_1; C_2 c, \sigma) \bullet\!\!\longrightarrow \textbf{abort}\ )$.
Proof: By $(C_1, \sigma) \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$.

(3) For any σ' and Σ', if $\mathsf{Env}_I^R((\sigma, J), \Sigma)((\sigma', J), \Sigma')$, then $(C_1; C_2, \sigma') \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma'))$.
Proof: From $(C_1, \sigma) \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$ and $\mathsf{Env}_I^R((\sigma, J), \Sigma)((\sigma', J), \Sigma')$, we get:

$$(C_1, \sigma') \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma'))$$

By the co-induction hypothesis, we get:

$$(C_1; C_2, \sigma') \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma'))$$

(4) If $C_1; C_2 = \mathbf{E}[\textbf{return}\ E]$, then ...
Proof: It is vacuously true.

(5) If $C_1; C_2 = \textbf{skip}$, then ...
Proof: It is vacuously true.

Thus we are done. □

The P-WHILE Rule.

Lemma 27 (P-WHILE-Rule)**.** *For any R, G, I, p, B and C_0,*

1. $R, G, I \models \{p * B\}\ C_0\ \{p * (B = B)\}$,
2. $p \Rightarrow I$,
3. $\mathsf{Sta}_I(p, R)$,

then $R, G, I \models \{p * (B = B)\}\ \textbf{while}\ (B)\ \{C_0\}\ \{p * \neg B\}$.

Proof. Below we prove: for any $\sigma, \sigma_t, J, \Sigma$ and μ, if $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models p * (B = B)$, then

$$(\textbf{while } (B) \ \{C_0\}, \sigma) \preceq^{R;G;I}_{p*\neg B} (\mu, (\sigma_t, J, \Sigma))$$

By co-induction, we only need to prove following (1)(2)(3)(4)(5).

(1) For any C' and σ',

if $(\textbf{while } (B) \ \{C_0\}, \sigma) \bullet\!\!\longrightarrow (C', \sigma')$,

then there exist μ', σ'_t, J' and Σ', such that

(a) $(\mu, (\sigma_t, J, \Sigma)) \rightrightarrows^* (\mu', (\sigma'_t, J', \Sigma'))$, and

(b) $(((\sigma, J), \Sigma), ((\sigma', \emptyset), \Sigma')) \models G * \mathsf{True}$, and

(c) $(C', \sigma') \preceq^{R;G;I}_{p*\neg B} (\mu', (\sigma'_t, J', \Sigma'))$.

Proof: By the operational semantics, we have following cases:

(A) If $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models B$, then by operational semantics, we know $C' = C_0; \textbf{while } (B) \ \{C_0\}$ and $\sigma' = \sigma$.

Let $\mu' = \mu$, $\sigma'_t = \sigma_t$, $J' = J$ and $\Sigma' = \Sigma$.

From $R, G, I \models \{p * B\} \ C_0 \ \{p * (B = B)\}$, and $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models p$, we know:

$$(C_0, \sigma) \preceq^{R;G;I}_{p*(B=B)} (\mu, (\sigma_t, J, \Sigma))$$

By the following Lemma 28, we get:

$$(C_0; \textbf{while } (B) \ \{C_0\}, \sigma) \preceq^{R;G;I}_{p*\neg B} (\mu, (\sigma_t, J, \Sigma))$$

(B) If $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models \neg B$, then by operational semantics, we know $C' = \textbf{skip}$ and $\sigma' = \sigma$.

Thus we know we only need to prove:

$$(\textbf{skip}, \sigma) \preceq^{R;G;I}_{p*\neg B} (\mu, (\sigma_t, J, \Sigma))$$

By $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models p * \neg B$, we get it.

(2) If $\textbf{while } (B) \ \{C_0\} \neq \mathbf{E}[\textbf{return } E]$,

then $\neg(\ (\textbf{while } (B) \ \{C_0\}, \sigma) \bullet\!\!\longrightarrow \textbf{abort}\ )$.

Proof: By $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models p * (B = B)$, we get it.

(3) For any σ' and Σ', if $\mathsf{Env}^R_I(((\sigma, J), \Sigma))(((\sigma', J), \Sigma'))$, then $(\textbf{while } (B) \ \{C_0\}, \sigma') \preceq^{R;G;I}_{p*\neg B} (\mu, (\sigma_t, J, \Sigma'))$.

Proof: From $\mathsf{Sta}_I(p, R)$ and

$\mathsf{Env}^R_I((\sigma, J), \Sigma)((\sigma', J), \Sigma')$, we know:

$$(\sigma', (\sigma_t, J, \Sigma', \mu)) \models p * (B = B)$$

By the co-induction hypothesis, we get:

$$(\textbf{while } (B) \ \{C_0\}, \sigma') \preceq^{R;G;I}_{p*\neg B} (\mu, (\sigma_t, J, \Sigma'))$$

(4) If $\textbf{while } (B) \ \{C_0\} = \mathbf{E}[\textbf{return } E]$, then ...

Proof: It is vacuously true.

(5) If **while** (B) $\{C_0\} = \mathbf{skip}$, then ...

Proof: It is vacuously true.

Thus we are done. □

Lemma 28. *For any* $C, C_0, B, \sigma, p, R, G, I, \mu, \sigma_t, J$ *and* Σ, *if*

1. $R, G, I \models \{p * B\}\ C_0\ \{p * (B = B)\}$,
2. $(C, \sigma) \preceq^{R;G;I}_{p*(B=B)} (\mu, (\sigma_t, J, \Sigma))$,
3. $\mathsf{Sta}_I(p, R)$,
4. $p \Rightarrow I$,

then $(C; \mathbf{while}\ (B)\ \{C_0\}, \sigma) \preceq^{R;G;I}_{p*\neg B} (\mu, (\sigma_t, J, \Sigma))$.

Proof. By co-induction, we only need to prove following (1)(2)(3)(4)(5)

(1) For any C' and σ',

if $(C; \mathbf{while}\ (B)\ \{C_0\}, \sigma) \bullet\!\!\longrightarrow (C', \sigma')$,

then there exist μ', σ'_t, J' and Σ', such that

(a) $(\mu, (\sigma_t, J, \Sigma)) \rightrightarrows^* (\mu', (\sigma'_t, J', \Sigma'))$, and

(b) $(((\sigma, J), \Sigma), ((\sigma', \emptyset), \Sigma')) \models G * \mathsf{True}$, and

(c) $(C', \sigma') \preceq^{R;G;I}_{p*\neg B} (\mu', (\sigma'_t, J', \Sigma'))$.

Proof: We have two cases depending on whether C is **skip** or not.

(I) If $C \neq \mathbf{skip}$, then we know $C' = C''; \mathbf{while}\ (B)\ \{C_0\}$ and

$$(C, \sigma) \bullet\!\!\longrightarrow (C'', \sigma')$$

By $(C, \sigma) \preceq^{R;G;I}_{p*(B=B)} (\mu, (\sigma_t, J, \Sigma))$, we know ther exist μ', σ'_t, J' and Σ', such that

(A) $(\mu, (\sigma_t, J, \Sigma)) \rightrightarrows^* (\mu', (\sigma'_t, J', \Sigma'))$, and

(B) $(((\sigma, J), \Sigma), ((\sigma', \emptyset), \Sigma')) \models G * \mathsf{True}$, and

(C) $(C'', \sigma') \preceq^{R;G;I}_{p*(B=B)} (\mu', (\sigma'_t, J', \Sigma'))$.

By the co-induction hypothesis, we get:

$$(C''; \mathbf{while}\ (B)\ \{C_0\}, \sigma') \preceq^{R;G;I}_{p*\neg B} (\mu', (\sigma'_t, J', \Sigma'))$$

(II) If $C = \mathbf{skip}$, then we know $C' = \mathbf{while}\ (B)\ \{C_0\}$ and $\sigma' = \sigma$.

Let $\mu' = \mu$, $J' = J$, $\sigma'_t = \sigma_t$ and $\Sigma' = \Sigma$, then we know only need to prove:

$$(\mathbf{while}\ (B)\ \{C_0\}, \sigma) \preceq^{R;G;I}_{p*\neg B} (\mu, (\sigma_t, J, \Sigma)) \qquad (*\text{G1})$$

By co-induction, we only need to prove the following (a)(b)(c)(d)(e).

(a) For any C' and σ',

if $(\mathbf{while}\ (B)\ \{C_0\}, \sigma) \bullet\!\!\longrightarrow (C', \sigma')$,

then there exist μ', σ'_t, J' and Σ', such that

(A) $(\mu, (\sigma_t, J, \Sigma)) \rightrightarrows^* (\mu', (\sigma'_t, J', \Sigma'))$, and

(B) $(((\sigma, J), \Sigma), ((\sigma', \emptyset), \Sigma')) \models G * \mathsf{True}$, and

(C) $(C', \sigma') \preceq^{R;G;I}_{p*\neg B} (\mu', (\sigma'_t, J', \Sigma'))$.

Proof: We have two cases depending on whether $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models B$ is true or false.

• If $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models B$, then we know we only need to prove:

$$(C_0; \textbf{while } (B)\ \{C_0\}, \sigma) \preceq^{R;G;I}_{p*\neg B}$$
$$(\mu, (\sigma_t, J, \Sigma))$$

By the co-induction hypothesis, we get it.

• If $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models \neg B$, then we know we only need to prove:

$$(\textbf{skip}, \sigma) \preceq^{R;G;I}_{(p*\neg B)} (\mu, (\sigma_t, J, \Sigma))$$

By $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models p * \neg B$, we get:

$$(\textbf{skip}, \sigma) \preceq^{R;G;I}_{(p*\neg B)} (\mu, (\sigma_t, J, \Sigma))$$

(b) If **while** $(B)\ \{C_0\} \neq \mathbf{E}[\textbf{return } E]$, then

$\neg(\ (\textbf{while } (B)\ \{C_0\}, \sigma) \bullet\!\!\longrightarrow \textbf{abort}\ )$.

Proof: By $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models p * (B = B)$, we get it.

(c) For any σ' and Σ',

if $\mathsf{Env}^R_I((\sigma, J), \Sigma)((\sigma', J), \Sigma')$, then $(\textbf{while } (B)\ \{C_0\}, \sigma') \preceq^{R;G;I}_{p*\neg B} (\mu, (\sigma_t, J, \Sigma'))$.

Proof: From $\mathsf{Sta}_I(p, R)$ and

$\mathsf{Env}^R_I((\sigma, J), \Sigma)((\sigma', J), \Sigma')$, we know:

$$(\sigma', (\sigma_t, J, \Sigma', \mu)) \models p * (B = B)$$

Then we know:

$$(\textbf{skip}, \sigma') \preceq^{R;G;I}_{(p*(B=B))} (\mu, (\sigma_t, J, \Sigma'))$$

By the co-induction hypothesis, we get:

$$(\textbf{while } (B)\ \{C_0\}, \sigma') \preceq^{R;G;I}_{p*\neg B} (\mu, (\sigma_t, J, \Sigma'))$$

(d) If **while** $(B)\ \{C_0\} = \mathbf{E}[\textbf{return } E]$, then ...

Proof: It is vacuously true.

(e) If **while** $(B)\ \{C_0\} = \textbf{skip}$, then ...

Proof: It is vacuously true.

Thus we have proved (*G1).

(2) If $C; \textbf{while } (B)\ \{C_0\} \neq \mathbf{E}[\textbf{return } E]$,

then $\neg(\ (C; \textbf{while } (B)\ \{C_0\}, \sigma) \bullet\!\!\longrightarrow \textbf{abort}\ )$.

Proof: By $(C, \sigma) \preceq^{R;G;I}_{p*B=B} (\mu, (\sigma_t, J, \Sigma))$, we get it.

(3) For any σ' and Σ', if $\mathsf{Env}_I^R(((\sigma, J), \Sigma))(((\sigma', J), \Sigma'))$, then $(C; \textbf{while}\ (B)\ \{C_0\}, \sigma') \preceq_{p*\neg B}^{R;G;I} (\mu, (\sigma_t, J, \Sigma'))$.

Proof: By $(C, \sigma) \preceq_{p*B=B}^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$, we know:

$$(C, \sigma') \preceq_{p*B=B}^{R;G;I} (\mu, (\sigma_t, J, \Sigma'))$$

By the co-induction hypothesis, we get:

$$(C; \textbf{while}\ (B)\ \{C_0\}, \sigma') \preceq_{p*\neg B}^{R;G;I} (\mu, (\sigma_t, J, \Sigma'))$$

(4) If $C; \textbf{while}\ (B)\ \{C_0\} = \mathbf{E}[\textbf{return}\ E]$, then ...

Proof: It is vacuously true.

(5) If $C; \textbf{while}\ (B)\ \{C_0\} = \textbf{skip}$, then ...

Proof: It is vacuously true.

Thus we are done. □

The RET Rule.

Lemma 29 (RET-rule)**.** *For any P, E', R, G, I and E, if*

1. $p \Rightarrow (E' = E)$,
2. $\mathsf{Sta}_I(p, R)$,

then $R, G, I \models \{\langle \textbf{skip}, E' \rangle * p\}\ \textbf{return}\ E\ \{\langle \textbf{skip}, E' \rangle * p\}$.

Proof. Below we prove: for any $\sigma, \sigma_t, J, \Sigma$ and μ, if $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models \langle \textbf{skip}, E' \rangle * p$, then

$$(\textbf{return}\ E', \sigma) \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma))$$

By co-induction, we only need to prove following (1)(2)(3)(4).

(1) If $\textbf{return}\ E' \neq \mathbf{E}[\textbf{return}\ \_]$, then ...

Proof: It is vacuously true.

(2) For any σ' and Σ', if $\mathsf{Env}_I^R((\sigma, J), \Sigma)((\sigma', J), \Sigma')$, then $(\textbf{return}\ E', \sigma') \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma'))$.

Proof: From $\mathsf{Sta}_I(p, R)$ and $\mathsf{Env}_I^R((\sigma, J), \Sigma)((\sigma', J), \Sigma')$, we know:

$$(\sigma', (\sigma_t, J, \Sigma', \mu)) \models \langle \textbf{skip}, E' \rangle * p$$

By the co-induction hypothesis, we get:

$$(\textbf{return}\ E', \sigma') \preceq_q^{R;G;I} (\mu, (\sigma_t, J, \Sigma')).$$

(3) We need to prove: there exist n', such that

(A) $[\![E]\!]_\sigma = n'$, and

(B) $\mu = (\textbf{skip}, n')$, and

(C) $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models \langle \textbf{skip}, E' \rangle * p$.

By $p \Rightarrow (E' = E)$, we know there exist n, such that

$$[\![E]\!]_\sigma = [\![E']\!]_\sigma = n$$

From $(\sigma, (\sigma_t, J, \Sigma, \mu)) \models \langle \mathbf{skip}, E' \rangle * p$, we know

$$\mu = (\mathbf{skip}, n)$$

(4) If **return** $E' =$ **skip**, then ...

Proof: It is vacuously true.

Thus we are done. □

11.3 Proof of Theorem 1 (Logic Soundness *w.r.t.* Contextual Refinement)

Theorem 1 is obtained immediately from Lemma 1 and 2.

12 Appendix: Example: Lock

A typical lock object has the methods $\mathsf{lock}(x)$ and $\mathsf{unlock}(x)$. It also has mechanisms for constructing and disposing locks, such as $\mathsf{makeLock}()$ and $\mathsf{disposeLock}(x)$. We will first define the lock object's specification and then demonstrate a test-and-set lock implementation satisfying the specification.

12.1 The Specification

We show the specification of the lock object in Fig. 36. According to the abstract object Γ, the specification of the lock method is $(\gamma_{\mathsf{lock}}; \mathbf{skip}, trp_{\mathsf{lock}}, trq_{\mathsf{lock}})$, the unlock is $(\gamma_{\mathsf{unlock}}; \mathbf{skip}, trp_{\mathsf{unlock}}, trq_{\mathsf{unlock}})$, the makeLock is $(\gamma_{\mathsf{make}}; \mathbf{skip}, trp_{\mathsf{make}}, trq_{\mathsf{make}})$, and the disposeLock is $(\gamma_{\mathsf{dispose}}; \mathbf{skip}, trp_{\mathsf{dispose}}, trq_{\mathsf{dispose}})$.

$trp_{\mathsf{lock}}(x, \overrightarrow{v})$ specifies that calling $\mathsf{lock}(x)$ requires the ownership of the lock x. We use $\mathsf{isLock}(x, \pi)$ to indicates that the client actually have the ownership of the lock x, with the permission π ($\pi > 0$). $\mathsf{isLock}(x, \pi)$ is defined as a token $[\mathsf{IsLock}(v)]^{\pi}$ and the assertion $(\mathsf{LS} \Mapsto \mathsf{locks} * v \in dom(\mathsf{locks}))$, which specifies that v is actually a lock in the abstract state. We show the structure of the abstract state Σ in Fig. 35. Σ consists of a map containing all locks, and we call it locks. locks has the form $\{v_1 \rightsquigarrow B_1, \ldots, v_n \rightsquigarrow B_n\}$, where v_i is the ID of the lock and B_i is the state of the lock. B is either 0(idle) or 1(locked).

$trq_{\mathsf{lock}}(x, \overrightarrow{v})$ means that $\mathsf{lock}(x)$ will return the ownership of the lock x ($\mathsf{isLock}(x, \pi)$), as well as the ownership of the locked lock x ($\mathsf{Locked}(x)$) to the client. $\mathsf{Locked}(x)$ is defined in a similar way to $\mathsf{isLock}(x, \pi)$. Note that the $\overrightarrow{v}$ in trp_{lock} and the $\overrightarrow{v}$ in trq_{lock} are used to specify that $\mathsf{lock}(x)$ transfers to the client the same permission of the lock x as the client gives it. γ_{lock} checks the state of the lock x, if x is idle, then update the state of x to locked. Otherwise, the thread's execution will get stuck and wait for the lock x to be idle.

The specification of the unlock method is very similar to the specification of the lock method. Thus we omit the explanation here.

trp_{make} means that $\mathsf{makeLock}()$ does not need any resources from the client. trq_{make} means that $\mathsf{makeLock}()$ will return the full ownership of a newly allocated lock to the client. And γ_{make} will add a new idel lock to the abstract lock map locks.

trp_{dispose} indicates that $\mathsf{disposeLock}(x)$ requires full ownership of the lock x, and also needs the ownership of the locked lock x. We use this requirement to indicate that, after executing $\mathsf{disposeLock}(x)$, no thread can use the lock x. trq_{dispose} means that $\mathsf{disposeLock}(x)$ will not return any resources to the client. And γ_{dispos} will remove the lock x from locks.

$$
\begin{aligned}
\Sigma &\stackrel{\mathsf{def}}{=} \{\mathsf{LS} \rightsquigarrow \mathsf{locks}\} \\
\mathsf{locks} &\stackrel{\mathsf{def}}{=} \{v_1 \rightsquigarrow B_1, \ldots, v_n \rightsquigarrow B_n\} \\
v &\in \mathit{Val} \\
B &\in \{0, 1\}
\end{aligned}
$$

Fig.35. Structure of Σ

$$
\begin{array}{lcl}
\mathsf{isLock}(v,\pi) & \stackrel{\mathsf{def}}{=} & [\mathsf{IsLock}(v)]^{\pi} * \mathsf{LS} \Rightarrow \mathsf{locks} * v \in dom(\mathsf{locks}) \\
\mathsf{Locked}(v) & \stackrel{\mathsf{def}}{=} & [\mathsf{Locked}(v)]^{1} * \mathsf{LS} \Rightarrow \mathsf{locks} * \mathsf{locks}(v) = 1 \\
\mathsf{UpdLocks}(\mathsf{locks}, v, s) & \stackrel{\mathsf{def}}{=} & \mathsf{locks}\{v \rightsquigarrow s\} \\
\mathsf{SetLock}(\Sigma, v, s) & \stackrel{\mathsf{def}}{=} & \{\mathsf{LS} \rightsquigarrow (\mathsf{UpdLocks}(\Sigma(\mathsf{LS}), v, s))\} \\
trp_{\mathsf{lock}}(x, \overrightarrow{v}) & \stackrel{\mathsf{def}}{=} & \begin{cases} \mathsf{isLock}(x,\pi) & \text{if } \overrightarrow{v} = \pi :: \mathsf{nil} \wedge 0 < \pi \leq 1 \\ \mathit{undefined} & \text{otherwise} \end{cases} \\
trq_{\mathsf{lock}}(x, \_, \overrightarrow{v}) & \stackrel{\mathsf{def}}{=} & \begin{cases} \mathsf{isLock}(x,\pi) * \mathsf{Locked}(x) & \text{if } \overrightarrow{v} = \pi :: \mathsf{nil} \wedge 0 < \pi \leq 1 \\ \mathit{undefined} & \text{otherwise} \end{cases} \\
\gamma_{\mathsf{lock}}((v, \emptyset, J), \Sigma) & \stackrel{\mathsf{def}}{=} & \begin{cases} ((\_, \emptyset, \{(\mathsf{isLock}, v) \rightsquigarrow \pi, (\mathsf{Locked}, v) \rightsquigarrow 1\}), \mathsf{SetLock}(\Sigma, v, 1)) & \text{if } \Sigma(\mathsf{LS}) = \mathsf{locks} \wedge \mathsf{locks}(v) = 0 \wedge \\ & J = \{(\mathsf{isLock}, v) \rightsquigarrow \pi\} \\ \mathit{undefined} & \text{otherwise} \end{cases} \\
trp_{\mathsf{unlock}}(x, \_, \overrightarrow{v}) & \stackrel{\mathsf{def}}{=} & \begin{cases} \mathsf{isLock}(x,\pi) * \mathsf{Locked}(x) & \text{if } \overrightarrow{v} = \pi :: \mathsf{nil} \wedge 0 < \pi \leq 1 \\ \mathit{undefined} & \text{otherwise} \end{cases} \\
trq_{\mathsf{unlock}}(x, \overrightarrow{v}) & \stackrel{\mathsf{def}}{=} & \begin{cases} \mathsf{isLock}(x,\pi) & \text{if } \overrightarrow{v} = \pi :: \mathsf{nil} \wedge 0 < \pi \leq 1 \\ \mathit{undefined} & \text{otherwise} \end{cases} \\
\gamma_{\mathsf{unlock}}((v, \emptyset, J), \Sigma) & \stackrel{\mathsf{def}}{=} & \begin{cases} ((\_, \emptyset, \{(\mathsf{isLock}, v) \rightsquigarrow \pi\}), \mathsf{SetLock}(\Sigma, v, 0)) & \text{if } \Sigma(\mathsf{LS}) = \mathsf{locks} \wedge \mathsf{locks}(v) = 1 \wedge \\ & J = \{(\mathsf{isLock}, v) \rightsquigarrow \pi, (\mathsf{Locked}, v) \rightsquigarrow 1\} \\ \mathit{undefined} & \text{otherwise} \end{cases} \\
trp_{\mathsf{make}}(\_, \_) & \stackrel{\mathsf{def}}{=} & \mathsf{emp} \\
trq_{\mathsf{make}}(\_, v, \_) & \stackrel{\mathsf{def}}{=} & \mathsf{isLock}(v, 1) \\
\gamma_{\mathsf{make}}((\_, \emptyset, \emptyset), \Sigma) & \stackrel{\mathsf{def}}{=} & \begin{cases} ((\_, \emptyset, \{(\mathsf{isLock}, v) \rightsquigarrow 1\}), \Sigma\{\mathsf{LS} \rightsquigarrow (\mathsf{locks} \uplus \{v \rightsquigarrow 0\})\}) & \text{if } \exists v. \Sigma(\mathsf{LS}) = \mathsf{locks} \wedge v \notin dom(\mathsf{locks}) \\ \mathit{undefined} & \text{otherwise} \end{cases} \\
trp_{\mathsf{dispose}}(v, \_) & \stackrel{\mathsf{def}}{=} & \mathsf{isLock}(v, 1) * \mathsf{Locked}(v) \\
trq_{\mathsf{dispose}}(v, \_, \_) & \stackrel{\mathsf{def}}{=} & \mathsf{emp} \\
\gamma_{\mathsf{dispose}}((\_, \emptyset, J), \Sigma) & \stackrel{\mathsf{def}}{=} & \begin{cases} ((\_, \emptyset, \emptyset), \Sigma\{\mathsf{LS} \rightsquigarrow \mathsf{locks}'\}) & \text{if } \exists \mathsf{locks}'.\ \Sigma(\mathsf{LS}) = \mathsf{locks} \wedge \mathsf{locks}(v) = 1 \wedge \\ & J = \{(\mathsf{isLock}, v) \rightsquigarrow 1, (\mathsf{Locked}, v) \rightsquigarrow 1\} \wedge \\ & \mathsf{locks} = \mathsf{locks}' \uplus \{v \rightsquigarrow 1\} \\ \mathit{undefined} & \text{otherwise} \end{cases} \\
\Gamma & \stackrel{\mathsf{def}}{=} & \{\mathsf{lock} \rightsquigarrow (\gamma_{\mathsf{lock}}; \mathbf{skip}, trp_{\mathsf{lock}}, trq_{\mathsf{lock}}), \mathsf{unlock} \rightsquigarrow (\gamma_{\mathsf{unlock}}; \mathbf{skip}, trp_{\mathsf{unlock}}, trq_{\mathsf{unlock}}), \\
& & \mathsf{makeLock} \rightsquigarrow (\gamma_{\mathsf{make}}; \mathbf{skip}, trp_{\mathsf{make}}, trq_{\mathsf{make}}), \mathsf{disposeLock} \rightsquigarrow (\gamma_{\mathsf{dispose}}; \mathbf{skip}, trp_{\mathsf{dispose}}, trq_{\mathsf{dispose}})\}
\end{array}
$$

Fig.36. Lock Specifications

12.2 An Test-and-Set Lock Implementation

We show a test-and-set lock implementation in Fig. 37. They are much standard, and we omit the explanations here.

```
lock(x){
 1   local b := false;
 2   while (!b){
 3     <b := cas(&x, 0, 1)>;
 4   }
 }

unlock(x){
 5   <[x] := 0>;
 }

makelock(){
 6   local x;
 7   <x := cons(0)>;
 8   return x;
 }

disposelock(x){
 9    <dispose(x)> ;
 }
```

Fig.37. Test-and-Set Lock

12.3 Judgments, Rely/Guarantee and the Invariant

We show the final judgments of the lock methods in Fig. 39. According to the OBJ rule, the judgments can be easily built by the lock specification. We also show I, R and G in Fig. 38. I is defined as $\mathsf{ConsI}(\mathsf{locks})$, where locks records all locks. $\mathsf{ConsI}(\mathsf{locks})$ consists of locks on the concrete side $\mathsf{Concrete}(\mathsf{locks})$, locks on the abstract side $\mathsf{LS} \Rrightarrow \mathsf{locks}$, and the property of locks. $\mathsf{Concrete}(\mathsf{locks})$ specifies each lock l in $locks$ with the assertion $(l \mapsto \mathsf{locks}(l))$, meaning that the abstract lock l is implemented by the memory slot at address l. $\mathsf{wellFL}(\mathsf{locks})$ constraints that for all lock l in the locks, the state of l must be 0 or 1. R and G are equal and defined as the union of five actions: lock action, unlock action, $\mathsf{makelock}$ action, $\mathsf{disposelock}$ action and the identical action $[I]$. The first four actions can be constructed easily with the corresponding abstract operation γ.

12.4 Proof Sketch

The proof of the four lock's methods are much straightforward, thus we omit here.

$$
\begin{array}{lll}
\mathsf{Concrete}(\mathsf{locks}) & \stackrel{\mathsf{def}}{=} & \circledast_{l \in dom(\mathsf{locks})}(l \mapsto \mathsf{locks}(l)) \\
\mathsf{wellFL}(\mathsf{locks}) & \stackrel{\mathsf{def}}{=} & \forall l.\ l \in dom(\mathsf{locks}) \implies \mathsf{locks}(l) \in \{0,1\} \\
\mathsf{ConsI}(\mathsf{locks}) & \stackrel{\mathsf{def}}{=} & \mathsf{LS} \Rrightarrow \mathsf{locks} * \mathsf{wellFL}(\mathsf{locks}) * \mathsf{Concrete}(\mathsf{locks}) \\
I & \stackrel{\mathsf{def}}{=} & \exists \mathsf{locks}.\mathsf{ConsI}(\mathsf{locks}) \\
 & & \\
\mathsf{lock} & \stackrel{\mathsf{def}}{=} & \exists l, \pi, \mathsf{locks}, \mathsf{locks}'. \\
 & & ((\mathsf{isLock}(l,\pi)) * \mathsf{ConsI}(\mathsf{locks}) * \mathsf{locks}(l) = 0) \ltimes \\
 & & (\mathsf{ConsI}(\mathsf{locks}') * \mathsf{locks}' = \mathsf{locks}\{l \rightsquigarrow 1\}) \\
\mathsf{unlock} & \stackrel{\mathsf{def}}{=} & \exists l, \pi, \mathsf{locks}, \mathsf{locks}'. \\
 & & ((\mathsf{isLock}(l,\pi) * \mathsf{Locked}(l)) * \mathsf{ConsI}(\mathsf{locks})) \ltimes \\
 & & (\mathsf{ConsI}(\mathsf{locks}') * \mathsf{locks}' = \mathsf{locks}\{l \rightsquigarrow 0\}) \\
\mathsf{makelock} & \stackrel{\mathsf{def}}{=} & \exists \mathsf{locks}, l, \mathsf{locks}'. \\
 & & (\mathsf{ConsI}(\mathsf{locks}) * l \notin dom(\mathsf{locks})) \ltimes \\
 & & (\mathsf{ConsI}(\mathsf{locks}') * \mathsf{locks}' = \mathsf{locks} \uplus \{l \rightsquigarrow 0\}) \\
\mathsf{disposelock} & \stackrel{\mathsf{def}}{=} & \exists \mathsf{locks}, l, \mathsf{locks}'. \\
 & & ((\mathsf{isLock}(l,1) * \mathsf{Locked}(l)) * \mathsf{ConsI}(\mathsf{locks})) \ltimes \\
 & & (\mathsf{ConsI}(\mathsf{locks}') * \mathsf{locks} = \mathsf{locks}' \uplus \{l \rightsquigarrow 1\}) \\
 & & \\
R = G & \stackrel{\mathsf{def}}{=} & \mathsf{lock} \vee \mathsf{unlock} \vee \mathsf{makelock} \vee \mathsf{disposelock} \vee [I]
\end{array}
$$

Fig.38. Definitions of I, R and G

$$
\begin{array}{ll}
R,G,I \vdash & \{\langle \gamma_{\mathsf{lock}}; \mathbf{skip}, l\rangle * \mathsf{isLock}(l,\pi) * x \mapsto l * I\} \\
 & \mathsf{lock}(x) \\
 & \{\langle \mathbf{skip}, \_\rangle * \mathsf{isLock}(l,\pi) * \mathsf{Locked}(l) * x \mapsto \_ * I\} \\
 & \\
R,G,I \vdash & \{\langle \gamma_{\mathsf{unlock}}; \mathbf{skip}, l\rangle * \mathsf{isLock}(l,\pi) * \mathsf{Locked}(l) * x \mapsto l * I\} \\
 & \mathsf{unlock}(x) \\
 & \{\langle \mathbf{skip}, \_\rangle * \mathsf{isLock}(l,\pi) * x \mapsto \_ * I\} \\
 & \\
R,G,I \vdash & \{\langle \gamma_{\mathsf{make}}; \mathbf{skip}, \_\rangle * I\} \\
 & \mathsf{makeLock}() \\
 & \{\exists l.\langle \mathbf{skip}, l\rangle * \mathsf{isLock}(l,1) * I\} \\
 & \\
R,G,I \vdash & \{\langle \gamma_{\mathsf{dispose}}; \mathbf{skip}, l\rangle * \mathsf{isLock}(l,1) * \mathsf{Locked}(l) * x \mapsto l * I\} \\
 & \mathsf{disposeLock}(x) \\
 & \{\langle \mathbf{skip}, \_\rangle * x \mapsto \_ * I\}
\end{array}
$$

Fig.39. Final Judgments for lock methods

13 Appendix: Full Rules

In this section, we show full rules for concrete operational semantics and sequential logic rules.

$$\frac{(C,\sigma) \bullet\!\!\longrightarrow^{*} (\mathbf{skip},\sigma')}{(\mathbf{E}[\langle C\rangle],\sigma) \bullet\!\!\longrightarrow (\mathbf{E}[\mathbf{skip}],\sigma')} \qquad \frac{(C,\sigma) \bullet\!\!\longrightarrow^{*} \mathbf{abort}}{(\mathbf{E}[\langle C\rangle],\sigma) \bullet\!\!\longrightarrow \mathbf{abort}}$$

$$\frac{[\![E]\!]_\sigma = n \quad x \in dom(\sigma) \quad \sigma' = \sigma\{x \rightsquigarrow n\}}{(\mathbf{E}[x := E],\sigma) \bullet\!\!\longrightarrow (\mathbf{E}[\mathbf{skip}],\sigma')} \qquad \frac{[\![E]\!]_\sigma \ undefined \quad or \quad x \notin dom(\sigma)}{(\mathbf{E}[x := E],\sigma) \bullet\!\!\longrightarrow \mathbf{abort}}$$

$$\frac{[\![E]\!]_\sigma = l \quad l \in dom(\sigma) \quad \sigma(l) = n \quad x \in dom(\sigma) \quad \sigma' = \sigma\{x \rightsquigarrow n\}}{(\mathbf{E}[x := [E]],\sigma) \bullet\!\!\longrightarrow (\mathbf{E}[\mathbf{skip}],\sigma')} \qquad \frac{[\![E]\!]_\sigma \ undefined \quad or \quad x \notin dom(\sigma) \quad or \quad l \notin dom(\sigma)}{(\mathbf{E}[x := [E]],\sigma) \bullet\!\!\longrightarrow \mathbf{abort}}$$

$$\frac{[\![E_1]\!]_\sigma = l \quad l \in dom(\sigma) \quad [\![E_2]\!]_\sigma = n \quad \sigma' = \sigma\{l \rightsquigarrow n\}}{(\mathbf{E}[[E_1] := E_2],\sigma) \bullet\!\!\longrightarrow (\mathbf{E}[\mathbf{skip}],\sigma')} \qquad \frac{[\![E_1]\!]_\sigma \ undefined \quad or \quad l \notin dom(\sigma) \quad or \quad [\![E_2]\!]_\sigma \ undefined}{(\mathbf{E}[[E_1] := E_2],\sigma) \bullet\!\!\longrightarrow \mathbf{abort}}$$

$$\frac{[\![B]\!]_\sigma = \mathsf{true}}{(\mathbf{E}[\mathbf{if}\ B\ C_1\ \mathbf{else}\ C_2],\sigma) \bullet\!\!\longrightarrow (\mathbf{E}[C_1],\sigma)} \qquad \frac{[\![B]\!]_\sigma = \mathsf{false}}{(\mathbf{E}[\mathbf{if}\ B\ C_1\ \mathbf{else}\ C_2],\sigma) \bullet\!\!\longrightarrow (\mathbf{E}[C_2],\sigma)}$$

$$\frac{[\![B]\!]_\sigma \ undefined}{(\mathbf{E}[\mathbf{if}\ B\ C_1\ \mathbf{else}\ C_2],\sigma) \bullet\!\!\longrightarrow \mathbf{abort}} \qquad \frac{[\![B]\!]_\sigma \ undefined}{(\mathbf{E}[\mathbf{while}\ (B)\ \{C\}],\sigma) \bullet\!\!\longrightarrow \mathbf{abort}}$$

$$\frac{[\![B]\!]_\sigma = \mathsf{true}}{(\mathbf{E}[\mathbf{while}\ (B)\ \{C\}],\sigma) \bullet\!\!\longrightarrow (\mathbf{E}[C; \mathbf{while}\ (B)\ \{C\}],\sigma)} \qquad \frac{[\![B]\!]_\sigma = \mathsf{false}}{(\mathbf{E}[\mathbf{while}\ (B)\ \{C\}],\sigma) \bullet\!\!\longrightarrow (\mathbf{E}[\mathbf{skip}],\sigma)}$$

(c) Thread Transitions Inside Method Calls

Fig.40. Operational Semantics Rules of Concrete Language

$$\frac{}{\vdash \{x \mapsto \_ * X = E\}\ x := E\ \{x \mapsto X\}}\ (\text{ASSN})$$

$$\frac{}{\vdash \{x \mapsto X * E \mapsto Y\}\ x := [E]\ \{x \mapsto Y * [X/x]E \mapsto Y\}}\ (\text{LD})$$

$$\frac{}{\vdash \{p\}\ \mathbf{skip}\ \{p\}}\ (\text{SKIP}) \qquad \frac{p \Rightarrow p' \quad \vdash \{p'\}\ C\ \{q'\} \quad q' \Rightarrow q}{\vdash \{p\}\ C\ \{q\}}\ (\text{CONSEQ})$$

$$\frac{\vdash \{p\}\ C\ \{q\}}{\vdash \{p * r\}\ C\ \{q * r\}}\ (\text{FRM}) \qquad \frac{\vdash \{p\}\ C\ \{q\}}{\vdash \{\exists X.p\}\ C\ \{\exists X.q\}}\ (\text{EXISTS})$$

$$\frac{\vdash \{p\}\ C_1\ \{p'\} \quad \vdash \{p'\}\ C_2\ \{q\}}{\vdash \{p\}\ C_1; C_2\ \{q\}}\ (\text{SEQ})$$

$$\frac{p \Rightarrow B = B \quad \vdash \{p * B\}\ C_1\ \{q\} \quad \vdash \{p * \neg B\}\ C_2\ \{q\}}{\vdash \{p\}\ \mathbf{if}\ B\ C_1\ \mathbf{else}\ C_2\ \{q\}}\ (\text{IF})$$

$$\frac{p \Rightarrow B = B \quad \vdash \{p * B\}\ C\ \{p\}}{\vdash \{p\}\ \mathbf{while}\ (B)\ \{C\}\ \{ * \neg B\}}\ (\text{WHILE})$$

$$\frac{\vdash \{p\}\ C\ \{q\} \quad \vdash \{p'\}\ C\ \{q'\}}{\vdash \{p \wedge p'\}\ C\ \{q \wedge q'\}}\ (\text{CONJ}) \qquad \frac{\vdash \{p\}\ C\ \{q\} \quad or \quad \vdash \{p'\}\ C\ \{q'\}}{\vdash \{p \vee p'\}\ C\ \{q \vee q'\}}\ (\text{DISJ})$$

Fig.41. Full Sequential Logic Rules