Formal Verification of Removing Unnecessary Synchronization Optimizations in Compilation

JUNPENG ZHA, Nanjing University, China

ACM Reference Format:
Junpeng Zha. 2018. Formal Verification of Removing Unnecessary Synchronization Optimizations in Compilation. 1, 1 (August 2018), 48 pages. https://doi.org/10.1145/1122445.1122456

There are some works, e.g. Jeff et al. [2] and Aldrich et al. [1], about removing unnecessary synchronizations, which is usually called lock elision and means that accessing thread local memory locations doesn't need to be protected by synchronizations in concurrent program. These works show that this optimization can reduce the synchronization overhead and improve the performance of concurrent program in practice. In this document, we consider how to verifying the correctness of removing unnecessary synchronization optimization formally.

As for verifying the concurrent program compilation correctness, Jiang et al. [3] propose a framework for verifying the compilation correctness for DRF concurrent programs and develop CASCompCert as an instantiation of their verification framework. However, CASCompCert work fails to support verifying this optimizations. The verification framework views ExtAtom as a switch pointer in non-preemptive semantics and requires that, when the source program does context switch, the target program needs to switch at the same time and to the same thread. This requirement provides some convenience for their verification work, because the source and target program can always execute the same threads. However, this requirement may not hold after applying lock elision optimization. As the following example shown in Fig. 1:

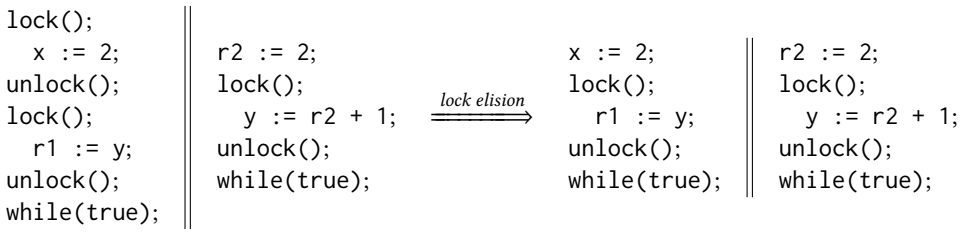


Fig. 1. Example of lock elision

In Fig. 1, we can find that the variable x is not a shared variable and doesn't need to be protected by synchronization. So, there is no problem that the compiler eliminates such synchronization. Verifying compiler with lock elision is not a trivial task, we need to solve the following problems:

(1) *How to say that a synchronization can be eliminated?* As the example shown in Fig. 1, we find that the synchronization protecting accessing variable x can be eliminated. However, in a language-independent

Author's address: Junpeng Zha, Nanjing University, Nanjin, China.

XXXX-XXXX/2018/8-ART $15.00
https://doi.org/10.1145/1122445.1122456

framework for verifying compiler correctness, there is no specific code and we need to consider how to describe that such synchronization can be eliminated.

(2) *How to establish the simulation relation between source and target program?* As we have explained, the CASCompCert fails to support verifying lock elision because it restricts that the switch points of source and target programs should match.

(3) *How to verify lock elision modularly?* The lock elision is just a compilation pass in a compiler. It needs to be composed with other compiler passes. And we also hope to achieve separate compilation.

In this document, we solve the problems shown above. We show our method to solve problem (1) in Sect. 1; we present our method to establish simulation relation between source and target programs in Sect. 2; and we show how to achieve modular verification in Sect. 3.

Definition 1 (Module-Local Downward Simulation)**.**
$(sl, ge, \gamma) \preccurlyeq_{\varphi, E} (tl, ge', \pi)$ iff

(1) $\lfloor\varphi\rfloor(ge) = ge'$;

(2) for all $\mathtt{f}, \Bbbk, \Sigma, \sigma, \mathbb{F}, F$, and $\mu = (\mathsf{dom}(\Sigma), \mathsf{dom}(\sigma), \varphi|_{\mathsf{dom}(\Sigma)})$, if $sl.\mathsf{InitCore}(\gamma, \mathtt{f}) = \Bbbk$, $\mathbb{F} \cap \mathsf{dom}(\Sigma) = F \cap \mathsf{dom}(\sigma) = \emptyset$, and $\mathsf{initM}(\varphi, ge, \Sigma, \sigma)$, then there exists $i \in \mathtt{index}$, κ such that:
$tl.\mathsf{InitCore}(\pi, \mathtt{f}) = \kappa$, and $(\mathbb{F}, (\Bbbk, \Sigma), \mathsf{emp}) \preccurlyeq_{\mu}^{i,E} (F, (\kappa, \sigma), \mathsf{emp})$,
where $(\mathbb{F}, (\Bbbk, \Sigma), \Delta) \preccurlyeq_{\mu}^{i,E} (F, (\kappa, \sigma), \delta)$ is defined in Def. 2.

Definition 2. $(\mathbb{F}, (\Bbbk, \Sigma), \Delta_0) \preccurlyeq_{\mu}^{i,E} (F, (\kappa, \sigma), \delta_0)$ is the largest relation such that, whenever $(\mathbb{F}, (\Bbbk, \Sigma), \Delta_0) \preccurlyeq_{\mu}^{i,E} (F, (\kappa, \sigma), \delta_0)$, then the following are true:

(1) for all $\Bbbk'$, Σ' and Δ, if $\mathbb{F} \vdash (\Bbbk, \Sigma) \overset{\tau}{\underset{\Delta}{\mapsto}} (\Bbbk', \Sigma')$, and $(\Delta_0 \cup \Delta) \subseteq (\mathbb{F} \cup \mu.\mathbb{S})$, then one of the following holds:

(a) $\exists j < i.\ (\mathbb{F}, (\Bbbk', \Sigma'), \Delta_0 \cup \Delta) \preccurlyeq_{\mu}^{j,E} (F, (\kappa, \sigma), \delta_0)$, or

(b) there exists κ', σ', δ and j such that:

(i) $F \vdash (\kappa, \sigma) \overset{\tau}{\underset{\delta}{\mapsto}}{}^{+} (\kappa', \sigma')$;

(ii) $(\delta_0 \cup \delta) \subseteq (F \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta, \delta_0 \cup \delta)$;

(iii) $(\mathbb{F}, (\Bbbk', \Sigma'), \Delta_0 \cup \Delta) \preccurlyeq_{\mu}^{j,E} (F, (\kappa', \sigma'), \delta_0 \cup \delta)$.

(2) for all $\Bbbk'$ and ι, if $\mathbb{F} \vdash (\Bbbk, \Sigma) \overset{\iota}{\underset{\mathsf{emp}}{\mapsto}} (\Bbbk', \Sigma)$, $\iota \neq \tau$, and $\mathsf{HG}(\Delta_0, \Sigma, \mathbb{F}, \mu.\mathbb{S})$, one of the following holds:

(a) there exists $\kappa', \kappa'', \sigma', \delta$, such that (* switch point match *)

(i) $F \vdash (\kappa, \sigma) \overset{\tau}{\underset{\delta}{\mapsto}}{}^{*} (\kappa', \sigma')$, $F \vdash (\kappa', \sigma') \overset{\iota}{\underset{\mathsf{emp}}{\mapsto}} (\kappa'', \sigma')$, and $\neg(\iota \in E)$;

(ii) $\mathsf{LG}(\mu, (\delta_0 \cup \delta, \sigma', F), (\Delta_0, \Sigma))$;

(iii) for all σ'' and Σ', if $\mathsf{Rely}(\mu, (\Sigma, \Sigma', \mathbb{F}), (\sigma', \sigma'', F))$, then there exists j such that
$(\mathbb{F}, (\Bbbk', \Sigma'), \mathsf{emp}) \preccurlyeq_{\mu}^{j,E} (F, (\kappa'', \sigma''), \mathsf{emp})$; or

(b) there exists $\kappa', \sigma', \delta, n$ and j, such that (* switch point not match *)

(i) $F \vdash (\kappa, \sigma) \overset{\tau}{\underset{\delta}{\mapsto}}{}^{n} (\kappa', \sigma')$, $\iota \in E$, and $(n = 0 \implies j < i)$;

(ii) $\mathsf{LG}(\mu, (\delta_0 \cup \delta, \sigma', F), (\Delta_0, \Sigma))$;

(iii) for all σ'' and Σ', if $\mathsf{Rely}(\mu, (\Sigma, \Sigma', \mathbb{F}), (\sigma', \sigma'', F))$, then
$(\mathbb{F}, (\Bbbk', \Sigma'), \mathsf{emp}) \preccurlyeq_{\mu}^{j,E} (F, (\kappa', \sigma''), \mathsf{emp})$.

Definition 3 (Whole-Program Downward Simulation). Let $\hat{\mathbb{P}} = \textbf{let}\ \Gamma\ \textbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$, $\hat{P} = \textbf{let}\ \Pi\ \textbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$, where $\Gamma = \{(sl_1, ge_1, \gamma_1), \ldots, (sl_m, ge_m, \gamma_m)\}$, $\Pi = \{(tl_1, ge'_1, \pi_1), \ldots, (tl_m, ge'_m, \pi_m)\}$. We say $\hat{\mathbb{P}} \preccurlyeq_E \hat{P}$ iff

- $\exists \varphi.\ (\forall 1 \le i \le m.\ \lfloor \varphi \rfloor ge_i = ge'_i)$, and
- for any $\widehat{\mathbb{W}}$, if $\hat{\mathbb{P}} :\overset{load}{\Longrightarrow} \widehat{\mathbb{W}}$, then there exists $\widehat{W}, i \in \mathtt{index}, \mu$, such that:
  $$\hat{P} :\overset{load}{\Longrightarrow} \widehat{W} \ \wedge\ (\widehat{\mathbb{W}}, \mathtt{emp}) \preccurlyeq^{i,\mu,E}_{(\widehat{W},\emptyset),0} (\widehat{W}, \mathtt{emp}, \mathtt{emp}).$$

Here we define $(\widehat{\mathbb{W}}, \Delta_0) \preccurlyeq^{i,\mu,E}_{p,n_0} (\widehat{W}, \delta_0, \delta_r)$ as the largest relation such that whenever $(\widehat{\mathbb{W}}, \Delta_0) \preccurlyeq^{i,\mu,E}_{p,n_0} (\widehat{W}, \delta_0, \delta_r)$ holds, then there exists $\mathtt{t}$, where $\widehat{W}.\mathtt{t} = \widehat{\mathbb{W}}.\mathtt{t} = \mathtt{t}$, such that the following are true:

(1) $\mathtt{t} \notin \mathsf{dom}(p.B)$, $\mathsf{dom}(p.B) \subseteq \mathsf{dom}(\mathbb{T})$, $\widehat{W}.\mathbb{d}(\mathtt{t}) \le \widehat{\mathbb{W}}.\mathbb{d}(\mathtt{t})$ and $p.\widehat{W} :\overset{\tau}{\underset{\delta_r \cup \delta_0}{\Longrightarrow}}{}^{n_0} \widehat{W}$;

(2) $\forall \widehat{\mathbb{W}}', \Delta$. if $\widehat{\mathbb{W}} \overset{\tau}{\underset{\Delta}{\Rightarrow}} \widehat{\mathbb{W}}'$, then one of the following holds:

(a) $\exists j < i.\ (\widehat{\mathbb{W}}', \Delta_0 \cup \Delta) \preccurlyeq^{j,\mu,E}_{p,n_0} (\widehat{W}, \delta_0, \delta_r)$; or

(b) $\exists \widehat{W}', \delta, j, n > 0$ such that:

(i) $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{n} \widehat{W}'$, and

(ii) $(\delta_0 \cup \delta) \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta, \delta_0 \cup \delta)$, and

(iii) $(\widehat{\mathbb{W}}', \Delta_0 \cup \Delta) \preccurlyeq^{j,\mu,E}_{p,n_0+n} (\widehat{W}', \delta_0 \cup \delta, \delta_r)$;

(3) $\forall \mathbb{T}', \mathtt{t}', \Sigma', o, \mathbb{d}'$, if $o \neq \tau$, and $\widehat{\mathbb{W}} :\overset{o}{\underset{\mathtt{emp}}{\Longrightarrow}} (\mathbb{T}', \mathtt{t}', \mathbb{d}', \Sigma')$, then one of the following holds:

(a) (* switch point match *) (* update the last switch point, and remove $\mathtt{t}''$ from reorder buffer *)
$\exists \widehat{W}', \delta, T', \sigma', \mathbb{d}'_1, j$, for any $\mathtt{t}'' \in \mathsf{dom}(\mathbb{T}')$, let $p' = ((T', \mathtt{t}'', \mathbb{d}'_1, \sigma'), B \backslash \{\mathtt{t}''\})$, such that

(i) $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{*} \widehat{W}'$, and $\widehat{W}' :\overset{o}{\underset{\mathtt{emp}}{\Longrightarrow}} (T', \mathtt{t}'', \mathbb{d}'_1, \sigma')$;

(ii) $(\delta_0 \cup \delta) \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0, \delta_0 \cup \delta)$;

(iii) $\exists T'', \sigma''.\ (T', \sigma') \Rrightarrow^{\mu}_{p.B} (T'', \sigma'') \wedge \mathsf{closed}(\mu.S, \sigma'')$;

(iv) if $p.B(\mathtt{t}'') = (n'', \delta'', \widehat{W}'')$, then:

- $\exists \widehat{W}''_1.\ (T', \mathtt{t}'', \mathbb{d}'_1, \sigma') :\overset{\tau}{\underset{\delta''}{\Longrightarrow}}{}^{n''} \widehat{W}''_1 \wedge \widehat{W}'' \overset{\delta''}{=\!=} \widehat{W}''_1$; (* redo executions of $\mathtt{t}''$ in buffer *)
- For any $\widehat{W}''_1$, if $(T', \mathtt{t}'', \mathbb{d}'_1, \sigma') :\overset{\tau}{\underset{\delta''}{\Longrightarrow}}{}^{n''} \widehat{W}''_1$ and $\widehat{W}'' \overset{\delta''}{=\!=} \widehat{W}''_1$, then
  $$((\mathbb{T}', \mathtt{t}'', \mathbb{d}', \Sigma'), \mathtt{emp}) \preccurlyeq^{j,\mu,E}_{p',n''} (\widehat{W}''_1, \mathtt{emp}, \delta'');$$

else (* no execution of $\mathtt{t}''$ in buffer *)
$((\mathbb{T}', \mathtt{t}'', \mathbb{d}', \Sigma'), \mathtt{emp}) \preccurlyeq^{j,\mu,E}_{p',0} ((T', \mathtt{t}'', \mathbb{d}'_1, \sigma'), \mathtt{emp}, \mathtt{emp})$; or

(b) (* switch point not match *)
$\exists \widehat{W}', \delta, n, j$, for any $\mathtt{t}'' \in \mathsf{dom}(\mathbb{T}')$, such that
(i) $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}^{n} \widehat{W}'$, $(n = 0 \implies j < i)$, $o \in E$; (* synchronization eliminated must in E *)
(ii) $(\delta_0 \cup \delta) \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0, \delta_0 \cup \delta)$;
(iii) (* undo the executions of current thread, and rollback to the last switch point *)
there exists $B = p.B\{\mathtt{t} \rightsquigarrow (n_0+n, \delta_r \cup \delta_0 \cup \delta, \widehat{W}')\}$, and $p' = (p.\widehat{W}^{\mathtt{t}''}, B\backslash\{\mathtt{t}''\})$ such that:
if $B(\mathtt{t}'') = (n'', \delta'', \widehat{W}'')$, then
- $\exists \widehat{W}_1''.\ p.\widehat{W}^{\mathtt{t}''} :\overset{\tau}{\underset{\delta''}{\Longrightarrow}}^{n''} \widehat{W}_1'' \wedge \widehat{W}'' \overset{\delta''}{=\!=} \widehat{W}_1''$; (* redo executions of $\mathtt{t}''$ in buffer *)
- For any $\widehat{W}_1''$, if $p.\widehat{W}^{\mathtt{t}''} :\overset{\tau}{\underset{\delta''}{\Longrightarrow}}^{n''} \widehat{W}_1''$ and $\widehat{W}'' \overset{\delta''}{=\!=} \widehat{W}_1''$, then
$$((\mathbb{T}', \mathtt{t}'', \mathbb{d}', \Sigma'), \mathsf{emp}) \preccurlyeq^{j,\mu,E}_{p',n''} (\widehat{W}_1'', \mathsf{emp}, \delta'');$$

else (* no execution of $\mathtt{t}''$ in buffer *)
$$((\mathbb{T}', \mathtt{t}'', \mathbb{d}', \Sigma'), \mathsf{emp}) \preccurlyeq^{j,\mu,E}_{p',0} (p.\widehat{W}^{\mathtt{t}''}, \mathsf{emp}, \mathsf{emp});$$

(4) if $\widehat{\mathbb{W}} :\overset{\tau}{\underset{\mathsf{emp}}{\Longrightarrow}}$ **done**, then $\exists \widehat{W}', \delta$.
(a) $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}^{*} \widehat{W}'$, $\widehat{W}' :\overset{\tau}{\underset{\mathsf{emp}}{\Longrightarrow}}$ **done**;
(b) $(\delta_0 \cup \delta) \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0, \delta_0 \cup \delta)$;

Definition 4 (Sequential Compiler Correctness). Correct(SeqComp, sl, tl, $\mathcal{R}$) holds iff
for any γ, π, ge, ge' and E, if $\mathsf{SeqComp.CodeT}(\gamma, E) = \pi$, $\lfloor \mathsf{SeqComp}.\varphi \rfloor(ge) = ge'$, then there exists $M_0, \ldots, M_n$ such that:
(1) $M_0 = \{(sl, ge, \gamma)\}$ and $M_n = \{(tl, ge', \pi)\}$; and
(2) $\forall i \in [0, n), \exists R \in \mathcal{R}, (M_i, M_{i+1}) \in R(E)$.

Theorem 5 (Final Theorem). For any $\mathsf{SeqComp}_1, \ldots, \mathsf{SeqComp}_m$, $sl_1, \ldots, sl_m$, $tl_1, \ldots, tl_m$ such that $\forall i \in \{1, \ldots, m\}$, we have Correct($\mathsf{SeqComp}_i$, sl_i, tl_i, $\mathcal{R}$), and for any $\mathtt{f}_1, \ldots, \mathtt{f}_n, \varphi, E = E_1 \cup \cdots \cup E_m, \Gamma = \{(sl_1, ge_1, \gamma_1), \ldots, (sl_m, ge_m, \gamma_m)\}$ and $\Pi = \{(tl_1, ge_1', \pi_1), \ldots, (tl_m, ge_m', \pi_m)\}$, if
(1) $\forall i \in \{1, \ldots, m\}.\ (\mathsf{SeqComp}_i.\mathsf{CodeT}(\gamma_i, E_i) = \pi_i) \wedge \mathsf{injective}(\varphi)$
$\wedge\ (\mathsf{SeqComp}_i.\varphi = \varphi) \wedge \lfloor \varphi \rfloor(ge_i) = ge_i' \wedge \{(sl_i, ge_i, \gamma_i), (tl_i, ge_i', \pi_i)\} \in \mathsf{Self}(\mathcal{R})$;
(2) Safe(**let** Γ **in** $\mathtt{f}_1 \parallel \ldots \parallel \mathtt{f}_n$), DRF(**let** Γ **in** $\mathtt{f}_1 \parallel \ldots \parallel \mathtt{f}_n$);
(3) AC(**let** Γ **in** $\mathtt{f}_1 \parallel \ldots \parallel \mathtt{f}_n, E$); (* the analysis result is correct *)
(4) $\forall i \in \{1, \ldots, m\}$. ReachClose($sl_i, ge_i, \gamma_i$);
then **let** Γ **in** $\mathtt{f}_1 \parallel \ldots \parallel \mathtt{f}_n \sqsupseteq$ **let** Π **in** $\mathtt{f}_1 \parallel \ldots \parallel \mathtt{f}_n$.

1 ABSTRACT PROGRAMMING LANGUAGE WITH IDENTIFIED SYNCHRONIZATION

In order to support verifying lock elision, we need to specify which synchronizations can be eliminated. we give a general framework for verifying lock elision optimization. Here, the compiler takes two arguments as the input: the first argument is the source code, and the second is the result of analysis declaring which synchronization operations can be eliminated. We use Figure. 2 to show the compilation process.

Here, we give each `lock()` and `unlock()` operation an unique identifier. The result of the analysis is a set of pairs of the identifiers of `lock()` and `unlock()` operations. The compiler does lock elision according to the result of the analysis. In this example, the synchronization operations identified id_1, id_2, id_5, id_6 will be eliminated by compiler. In the target code, we can find the synchronization operations identified id_1, id_2, id_5 and id_6 are eliminated. The code transformation about lock elision in compiler is trivial. It only need to eliminate

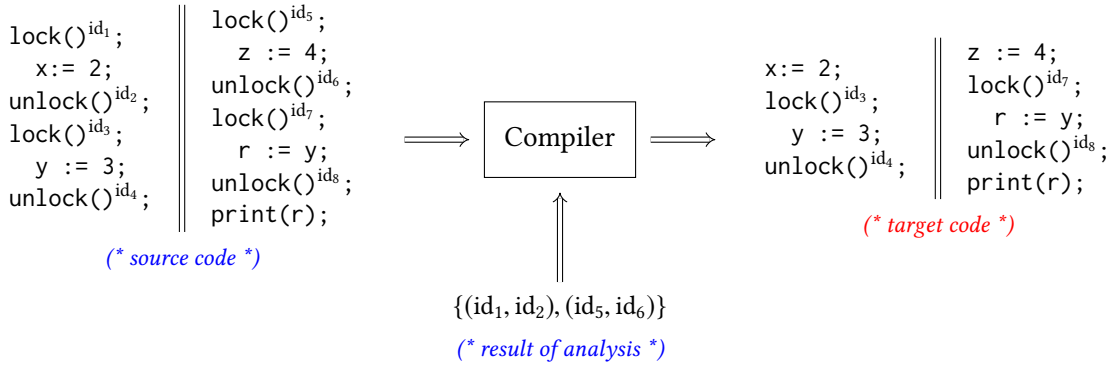


Fig. 2. The compilation process with lock elision

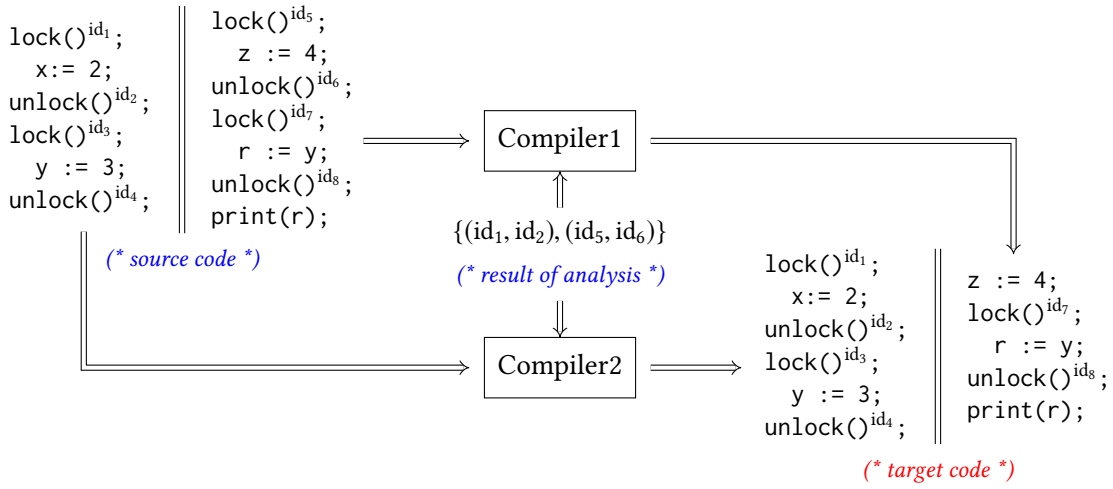


Fig. 3. The separate compilation process with lock elision

the synchronization operations according to the analysis result. Figure. 3 shows the lock elision in separate compilation. Here, only Compiler1 does lock elision.

1.1 Operational Semantics

We can find that the key to support the compilation process is given each synchronization operation an unique identifier. However, the abstract language defined in CASCompCert doesn't distinguish each synchronization operation in the program. So, in this subsection, we will make some modification to the language defined in CASCompCert. The first is the definition of *Msg*. Entering and exiting the atomic block will not only produce EntAtom and ExtAtom event, as well as their identifiers. The definitions of abstract concurrent language and its state are presented in Figure. 4 and 5.

$$
\begin{array}{rrcl}
(\mathit{Entry}) & \mathtt{f} & \in & \mathit{String} \\
(\mathit{Prog}) & P, \mathbb{P} & ::= & \textbf{let}\ \Pi\ \textbf{in}\ \mathtt{f}_1 \parallel \ldots \parallel \mathtt{f}_n \\
(\mathit{GEnv}) & \mathit{ge} & \in & \mathit{Addr} \rightharpoonup_{\mathsf{fin}} \mathit{Val} \\
(\mathit{MdSet}) & \Pi, \Gamma & ::= & \{(\mathit{tl}_1, \mathit{ge}_1, \pi_1), \ldots, (\mathit{tl}_m, \mathit{ge}_m, \pi_m)\} \\
(\mathit{Lang}) & \mathit{tl}, \mathit{sl} & ::= & (\mathit{Module},\ \mathit{Core},\ \mathsf{InitCore},\ \longmapsto) \\
(\mathit{Module}) & \pi, \gamma & ::= & \ldots \\
(\mathit{Core}) & \kappa, \Bbbk & ::= & \ldots \\
\mathsf{InitCore} & & \in & \mathit{Module} \to \mathit{Entry} \rightharpoonup \mathit{Core} \\
\longmapsto & & \in & \mathit{FList} \times (\mathit{Core} \times \mathit{State}) \to \\
 & & & \quad \mathcal{P}((\mathit{Msg} \times \mathit{FtPrt}) \times ((\mathit{Core} \times \mathit{State}) \cup \textbf{abort}))
\end{array}
$$

$$
\begin{array}{rrcl}
(\mathit{ThrdID}) & \mathtt{t} & \in & \mathbb{N} \\
(\mathit{Addr}) & l & ::= & \ldots \\
(\mathit{Val}) & v & ::= & l \mid \ldots \\
(\mathit{FList}) & F, \mathbb{F} & \in & \mathcal{P}^{\omega}(\mathit{Addr}) \\
(\mathit{State}) & \sigma, \Sigma & \in & \mathit{Addr} \rightharpoonup_{\mathsf{fin}} \mathit{Val} \\
(\mathit{FtPrt}) & \delta, \Delta & ::= & (\mathit{rs}, \mathit{ws}) \quad \text{where } \mathit{rs}, \mathit{ws} \in \mathcal{P}(\mathit{Addr}) \\
(\mathit{Msg}) & \iota & ::= & \tau \mid e \mid \mathtt{ret} \mid \mathrm{id.EntA} \mid \mathrm{id.ExtA} \quad (\text{where } \mathrm{id} \in \mathbb{N}\backslash\{0\}) \\
(\mathit{Event}) & e & ::= & \ldots \\
(\mathit{Config}) & \phi, \Phi & ::= & (\kappa, \sigma) \mid \textbf{abort}
\end{array}
$$

Fig. 4. The abstract concurrent language

$$
\begin{array}{rrcl}
(\mathit{World}) & W, \mathbb{W} & ::= & (T, \mathtt{t}, d, \sigma) \qquad (\mathit{AtomBit})\ d ::= \mathrm{id} \mid 0 \\
(\mathit{NPWorld}) & \widehat{W}, \widehat{\mathbb{W}} & ::= & (T, \mathtt{t}, \mathbb{d}, \sigma) \qquad (\mathit{AtomBits})\ \mathbb{d} ::= \{\mathtt{t}_1 \rightsquigarrow d_1, \ldots, \mathtt{t}_n \rightsquigarrow d_n\} \\
(\mathit{GMsg}) & o & ::= & \tau \mid e \mid \mathtt{sw} \mid \mathrm{id.EntA} \mid \mathrm{id.ExtA} \\
(\mathit{ThrdPool}) & T, \mathbb{T} & ::= & \{\mathtt{t}_1 \rightsquigarrow (\mathit{tl}_1, F_1, \kappa_1), \ldots, \mathtt{t}_n \rightsquigarrow (\mathit{tl}_1, F_n, \kappa_n)\}
\end{array}
$$

Fig. 5. The state of abstract concurrent language

We can introduce *identified synchronization* that is in the form of $\mathtt{lock()}^{\mathrm{id}}$ and $\mathtt{unlock()}^{\mathrm{id}}$. And their semantics can be defined as:

$$
\frac{}{F \vdash (\mathtt{lock()}^{\mathrm{id}}, \sigma) \xmapsto[\mathsf{emp}]{\mathrm{id.EntA}} (\textbf{skip}, \sigma)} \qquad \frac{}{F \vdash (\mathtt{unlock()}^{\mathrm{id}}, \sigma) \xmapsto[\mathsf{emp}]{\mathrm{id.ExtA}} (\textbf{skip}, \sigma)}
$$

$$\frac{T(\mathtt{t}) = (tl, F, \kappa) \quad F \vdash (\kappa, \sigma) \xmapsto[\delta]{\tau} (\kappa', \sigma')}{(T, \mathtt{t}, d, \sigma) \overset{\tau}{\underset{\delta}{\Rightarrow}} (T\{\mathtt{t} \rightsquigarrow (tl, F, \kappa')\}, \mathtt{t}, d, \sigma')}\ \tau\text{-step} \qquad \frac{T(\mathtt{t}) = (tl, F, \kappa) \quad F \vdash (\kappa, \sigma) \xmapsto[\mathsf{emp}]{e} (\kappa', \sigma) \quad T' = T\{\mathtt{t} \rightsquigarrow (tl, F, \kappa')\}}{(T, \mathtt{t}, 0, \sigma) \overset{e}{\underset{\mathsf{emp}}{\Longrightarrow}} (T', \mathtt{t}, 0, \sigma)}\ \text{Print}$$

$$\frac{T(\mathtt{t}) = (tl, F, \kappa) \quad F \vdash (\kappa, \sigma) \xmapsto[\mathsf{emp}]{\mathsf{id.EntA}} (\kappa', \sigma)}{(T, \mathtt{t}, 0, \sigma) \overset{\mathsf{id.EntA}}{\underset{\mathsf{emp}}{\Longrightarrow}} (T\{\mathtt{t} \rightsquigarrow (tl, F, \kappa')\}, \mathtt{t}, \mathsf{id}, \sigma)}\ \text{EntAt} \qquad \frac{T(\mathtt{t}) = (tl, F, \kappa) \quad F \vdash (\kappa, \sigma) \xmapsto[\mathsf{emp}]{\mathsf{id.ExtA}} (\kappa', \sigma)}{(T, \mathtt{t}, \mathsf{id}_0, \sigma) \overset{\mathsf{id.ExtA}}{\underset{\mathsf{emp}}{\Longrightarrow}} (T\{\mathtt{t} \rightsquigarrow (tl, F, \kappa')\}, \mathtt{t}, 0, \sigma)}\ \text{ExtAt}$$

$$\frac{\mathtt{t}' \in \mathrm{dom}(T)}{(T, \mathtt{t}, 0, \sigma) \overset{\mathsf{sw}}{\underset{\mathsf{emp}}{\Longrightarrow}} (T, \mathtt{t}', 0, \sigma)}\ \text{Switch} \qquad \frac{T(\mathtt{t}) = (tl, F, \kappa) \quad F \vdash (\kappa, \sigma) \longmapsto_{\delta} \mathbf{abort}}{(T, \mathtt{t}, 0, \sigma) \overset{\tau}{\underset{\delta}{\Rightarrow}} \mathbf{abort}}\ \text{Abort}$$

$$\frac{T(\mathtt{t}) = (tl, F, \kappa) \quad \mathtt{t}' \in \mathrm{dom}(T\backslash\mathtt{t}) \quad F \vdash (\kappa, \sigma) \xmapsto[\mathsf{emp}]{\mathsf{ret}} (\kappa', \sigma)}{(T, \mathtt{t}, 0, \sigma) \overset{\mathsf{sw}}{\underset{\mathsf{emp}}{\Longrightarrow}} (T\backslash\mathtt{t}, \mathtt{t}', 0, \sigma)}\ \text{Term} \qquad \frac{T(\mathtt{t}) = (tl, F, \kappa) \quad \mathrm{dom}(T) = \{\mathtt{t}\} \quad F \vdash (\kappa, \sigma) \xmapsto[\mathsf{emp}]{\mathsf{ret}} (\kappa', \sigma)}{(T, \mathtt{t}, 0, \sigma) \overset{\tau}{\underset{\mathsf{emp}}{\Longrightarrow}} \mathbf{done}}\ \text{Done}$$

Fig. 6. Modified preemptive global semantics

$$\frac{T(\mathtt{t}) = (tl, F, \kappa) \quad F \vdash (\kappa, \sigma) \overset{\tau}{\underset{\delta}{\mapsto}} (\kappa', \sigma') \quad T' = T\{\mathtt{t} \rightsquigarrow (tl, F, \kappa')\}}{(T, \mathtt{t}, \mathbb{d}, \sigma) :\overset{\tau}{\underset{\delta}{\Rightarrow}} (T', \mathtt{t}, \mathbb{d}, \sigma')}\ \tau\text{-step}_{\mathsf{np}} \qquad \frac{T(\mathtt{t}) = (tl, F, \kappa) \quad \mathbb{d}(\mathtt{t}) = 0 \quad \mathtt{t}' \in \mathrm{dom}(T) \quad F \vdash (\kappa, \sigma) \overset{e}{\underset{\mathsf{emp}}{\mapsto}} (\kappa', \sigma) \quad T' = T\{\mathtt{t} \rightsquigarrow (tl, F, \kappa')\}}{(T, \mathtt{t}, \mathbb{d}, \sigma) :\overset{e}{\underset{\mathsf{emp}}{\Longrightarrow}} (T', \mathtt{t}', \mathbb{d}, \sigma)}\ \text{Print}_{\mathsf{np}}$$

$$\frac{T(\mathtt{t}) = (tl, F, \kappa) \quad \mathbb{d}(\mathtt{t}) = 0 \quad \mathtt{t}' \in \mathrm{dom}(T) \quad F \vdash (\kappa, \sigma) \xmapsto[\mathsf{emp}]{\mathsf{id.EntA}} (\kappa', \sigma) \quad T' = T\{\mathtt{t} \rightsquigarrow (tl, F, \kappa')\}}{(T, \mathtt{t}, \mathbb{d}, \sigma) :\overset{\mathsf{id.EntA}}{\underset{\mathsf{emp}}{\Longrightarrow}} (T', \mathtt{t}', \mathbb{d}\{\mathtt{t} \rightsquigarrow \mathrm{id}\}, \sigma)}\ \text{EntAt}_{\mathsf{np}} \qquad \frac{T(\mathtt{t}) = (tl, F, \kappa) \quad \mathbb{d}(\mathtt{t}) = \mathrm{id}_0 \quad \mathtt{t}' \in \mathrm{dom}(T) \quad F \vdash (\kappa, \sigma) \xmapsto[\mathsf{emp}]{\mathsf{id.ExtA}} (\kappa', \sigma) \quad T' = T\{\mathtt{t} \rightsquigarrow (tl, F, \kappa')\}}{(T, \mathtt{t}, \mathbb{d}, \sigma) :\overset{\mathsf{id.ExtA}}{\underset{\mathsf{emp}}{\Longrightarrow}} (T', \mathtt{t}', \mathbb{d}\{\mathtt{t} \rightsquigarrow 0\}, \sigma)}\ \text{ExtAt}_{\mathsf{np}}$$

$$\frac{T(\mathtt{t}) = (tl, F, \kappa) \quad F \vdash (\kappa, \sigma) \longmapsto_{\delta} \mathbf{abort}}{(T, \mathtt{t}, \mathbb{d}, \sigma) :\overset{\tau}{\underset{\delta}{\Rightarrow}} \mathbf{abort}}\ \text{Abort}_{\mathsf{np}}$$

$$\frac{T(\mathtt{t}) = (tl, F, \kappa) \quad \mathbb{d}(\mathtt{t}) = 0 \quad F \vdash (\kappa, \sigma) \xmapsto[\mathsf{emp}]{\mathsf{ret}} (\kappa', \sigma) \quad \mathtt{t}' \in \mathrm{dom}(T\backslash\mathtt{t})}{(T, \mathtt{t}, \mathbb{d}, \sigma) :\overset{\mathsf{sw}}{\underset{\mathsf{emp}}{\Longrightarrow}} (T\backslash\mathtt{t}, \mathtt{t}', \mathbb{d}\backslash\mathtt{t}, \sigma)}\ \text{Term}_{\mathsf{np}} \qquad \frac{T(\mathtt{t}) = \{\mathtt{t} \rightsquigarrow (tl, F, \kappa)\} \quad \mathbb{d} = \{\mathtt{t} \rightsquigarrow 0\} \quad F \vdash (\kappa, \sigma) \xmapsto[\mathsf{emp}]{\mathsf{ret}} (\kappa', \sigma)}{(T, \mathtt{t}, \mathbb{d}, \sigma) :\overset{\tau}{\underset{\mathsf{emp}}{\Longrightarrow}} \mathbf{done}}\ \text{Done}_{\mathsf{np}}$$

Fig. 7. Modified non-preemptive global semantics

We present the operation semantics in Figure. 6 and 7. Here, we only modify the rules for entering and exiting atomic block. We mark the rule modified in blue.

$$\frac{P \xRightarrow{load} W \quad W \Rightarrow^* W' \quad (W', E) \Longmapsto \mathsf{WA}}{(P, E) \Longmapsto \mathsf{WA}} \qquad \frac{(W, E) \Longmapsto \mathsf{WA}_0 \vee (W, E) \Longmapsto \mathsf{WA}_1}{(W, E) \Longmapsto \mathsf{WA}}$$

$$\frac{\begin{array}{c}\mathtt{t}_1 \neq \mathtt{t}_2 \quad \delta_1 \frown \delta_2 \quad (\mathrm{id}_1, \_) \in E \\ \mathtt{predict}(W, \mathtt{t}_1, (\delta_1, \mathrm{id}_1)) \quad \mathtt{predict}(W, \mathtt{t}_2, (\delta_2, d_2))\end{array}}{(W, E) \Longmapsto \mathsf{WA}_0}$$

$$\frac{\begin{array}{c}W \Rightarrow^* W_1 \quad W_1 \xRightarrow[\mathsf{emp}]{\mathrm{id}'.\mathsf{ExtA}} W' \quad W_1.d = \mathrm{id} \\ ((\mathrm{id}, \mathrm{id}_1) \in E \wedge \mathrm{id}_1 \neq \mathrm{id}') \vee ((\mathrm{id}_2, \mathrm{id}') \in E \wedge \mathrm{id}_2 \neq \mathrm{id})\end{array}}{(W, E) \Longmapsto \mathsf{WA}_1}$$

$$\frac{\hat{P} :\xRightarrow{load} \widehat{W} \quad \widehat{W} :\Rightarrow^* \widehat{W}' \quad (\widehat{W}', E) :\Longmapsto \mathsf{WA}}{(\hat{P}, E) :\Longmapsto \mathsf{WA}} \qquad \frac{(\widehat{W}, E) :\Longmapsto \mathsf{WA}_0 \vee (\widehat{W}, E) :\Longmapsto \mathsf{WA}_1}{(\widehat{W}, E) :\Longmapsto \mathsf{WA}}$$

$$\frac{\begin{array}{c}\widehat{W} :\xRightarrow[\mathsf{emp}]{o} \widehat{W}' \quad o \neq \tau \quad \mathtt{t}_1 \neq \mathtt{t}_2 \quad \delta_1 \frown \delta_2 \quad (\mathrm{id}_1, \_) \in E \\ \mathtt{NPpredict}(\widehat{W}', \mathtt{t}_1, (\delta_1, \mathrm{id}_1)) \quad \mathtt{NPpredict}(\widehat{W}', \mathtt{t}_2, (\delta_2, d_2))\end{array}}{(\widehat{W}, E) :\Longmapsto \mathsf{WA}_0}$$

$$\frac{\begin{array}{c}\widehat{W} :\Rightarrow^* \widehat{W}_1^{\mathtt{t}} \quad \widehat{W}_1^{\mathtt{t}} :\xRightarrow[\mathsf{emp}]{\mathrm{id}'.\mathsf{ExtA}} \widehat{W}' \quad \widehat{W}_1.\mathbb{d}(\mathtt{t}) = \mathrm{id} \\ ((\mathrm{id}, \mathrm{id}_1) \in E \wedge \mathrm{id}_1 \neq \mathrm{id}') \vee ((\mathrm{id}_2, \mathrm{id}') \in E \wedge \mathrm{id}_2 \neq \mathrm{id})\end{array}}{(\widehat{W}, E) :\Longmapsto \mathsf{WA}_1}$$

Fig. 8. Wrong Analysis Result

1.2 Properties of Program Ensured by Analysis Result

In order to describe the properties of program ensured by analysis result. We first need to define the result of analysis below formally. *SyncEA* means synchronization elision analysis.

$$(\mathit{SyncEA}) \quad E ::= \{(\mathrm{id}_1, \mathrm{id}_1'), \ldots, (\mathrm{id}_n, \mathrm{id}_n')\}$$

Then, we define the properties of program ensured by analysis result. The analysis result ensures two properties:

(1) The code execution in synchronization that can be eliminated will not conflict with other threads execution;
(2) If an `lock()` operation is eliminated, the corresponding `unlock()` operation will also be eliminated.

Similar with the data race free property. We define "$(P, E) \Longmapsto \mathsf{WA}$" to represent the analysis result is wrong. In assumption, we require the analysis result is correct. Full definitions can be found in Figure. 8.

$$(\mathit{Analysis\ Correct}) \quad \mathsf{AC}(P, E) ::= \neg((P, E) \Longmapsto \mathsf{WA})$$

2 IDEA TO ESTABLISH SIMULATION RELATION BETWEEN SOURCE AND TARGET WHEN SWITCH POINT DOES NOT MATCH

We use the example shown in Fig. 1 to illustrate our idea to establish simulation relation between source and target programs. We can construct the target program's execution from the given source execution as below:

(1) The The source program first executes the code `x := 2` in `T1`, and the target program also execute the same code in step (1).

Source: `T1: lock(); x := 2; unlock();`

Target: `T1: x := 2;`

Buffer:

(2) In step (2), the source switches to thread `T2` to execute, but the target can't switch to `T2`, because there is no switch point in target at this moment. So, we undo the execution of `T1`, and let the `T2` executes.

Source: `T1: lock(); x := 2; unlock(); T2: r2 := 2; lock(); y := r2 + 1; unlock();`

Target: `T2: r2 := 2; lock(); y := r2 + 1; unlock();`

Buffer: `T1: x := 2;`

(3) In step (3), the source switches to thread `T1` again and executes `r1 := y`. The target can also switch to `T1` and redoes `x := 2` in buffer again.

Source: `T1: lock(); x := 2; unlock(); T2: r2 := 2; lock(); y := r2 + 1; unlock();`
`T1: lock(); r1 := y; unlock();`

Target: `T2: r2 := 2; lock(); y := r2 + 1; unlock(); T1: x := 2; lock(); r1 := y; unlock();`

Buffer:

(4) Finally, the source and target both switch to `T2` and diverge silently.

Source: `T1: lock(); x := 2; unlock(); T2: r2 := 2; lock(); y := r2 + 1; unlock();`
`T1: lock(); r1 := y; unlock(); T2: silent diverge;`

Target: `T2: r2 := 2; lock(); y := r2 + 1; unlock(); T1: x := 2; lock(); r1 := y; unlock();`
`T2: T2: silent diverge;`

Buffer:

Note that our undo and redo mechainism to construct target program executions is to simulate the reordering of thread executions. So, our method requires that the source program should be data-race-free, otherwise the execution of program before and after reordering will not be equal. We give the following counterexample:

```
x := 3;          ||   r2 := x;          lock elision   x := 3;          ||   r2 := x;
lock();          ||   print(r2);        ⟹⟹⟹⟹⟹⟹⟹⟹   y := 2;          ||   print(r2);
y := 2;          ||                                    while(true);     ||
unlock();        ||
while(true);     ||
```

We give a source program's execution below:

`T1: x := 3; T1: y := 2; T2: r2 := x; T2: print(r2)` (* output 3 *); `T1: silent diverge`

By using our method, we construct a target program's execution.

`T2: r2 := x; T2: print(r2)` (* output 0 *); `T1: x := 3; T1: y := 2; T1: silent diverge`

Supposing the variables x and y are 0 in the inital state. We can find that the source program will output 3 and the target program will output 0. Although our method is correct under the assumption that the source program is data-race-free, we can find that the source and target program's behaviors will not be equivalent, such as the example shown above. The target program can't output 3, but source program can.

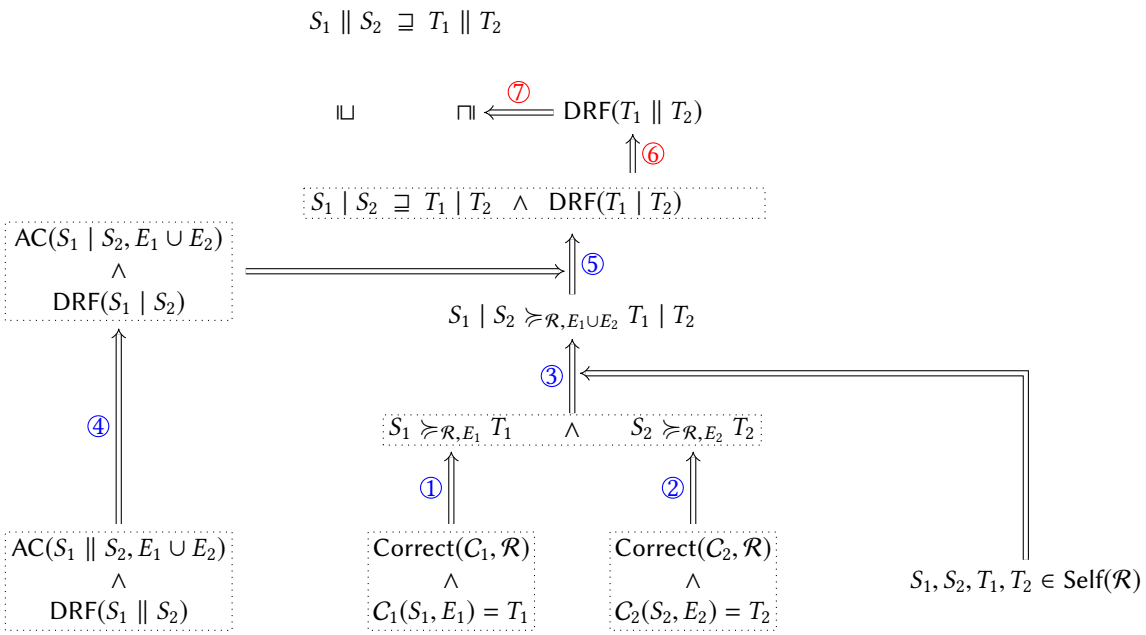


Fig. 9. Proof Sketch

3 ESTABLISHING COMPILER CORRECTNESS USING RUSC

We use the theory of RUSC proposed in CompCertM [4] to establish the proof of the compiler correctness.

Proof Sketch. We use Figure. 9 to show the structure of establishing compiler correctness using RUSC. The compilation and the compiler correctness ensure the refinement under self-related contexts between source and target, shown as ① and ②. Here, the compiler takes the source code and analysis result as input. The RUSC relation is compositional, so we get the whole source and target program have refinement under self-related contexts, shown as ③. The target code produced by compiler not only preserves the behaviors of the source code, but also the data-race-free and analysis correctness property, shown as ⑤.

Details about achieving ①, ②, and ③ can be found in Figure. 10. Here, the relations between the input and output of each pass is defined from R_1 to R_5, which parameters with an analysis result as argument.

Details. The module Module in RUSC is instantiated as high- and low-level module set Γ and Π. And the linking operation $\oplus$ in RUSC is instantiated as the set union operation $\cup$.

Here, because we have an additional input *analysis result* for compiler. The module set relation R is defined as a mapping from *analysis result* to a set of pairs of module set (*MdSet*).

$$\textit{(MdSetRel)} \quad R \in \textit{SyncEA} \to \mathcal{P}(\textit{MdSet} \times \textit{MdSet})$$

And a module set relation is well-defined, if it satisfies *VerComp* and *Adequency* properties. In Sec 4, we define a module local simulation to instantiate the module set relation. The module set relation will be instantiated as that each source module has a module local simulation with its corresponding target module. The module local simulation takes an *analysis result* as parameter. The instantiation can be found in Sec. 4.

Definition 6 (Well-defined Module Set Relation)**.** wd(R) holds, iff the following holds:

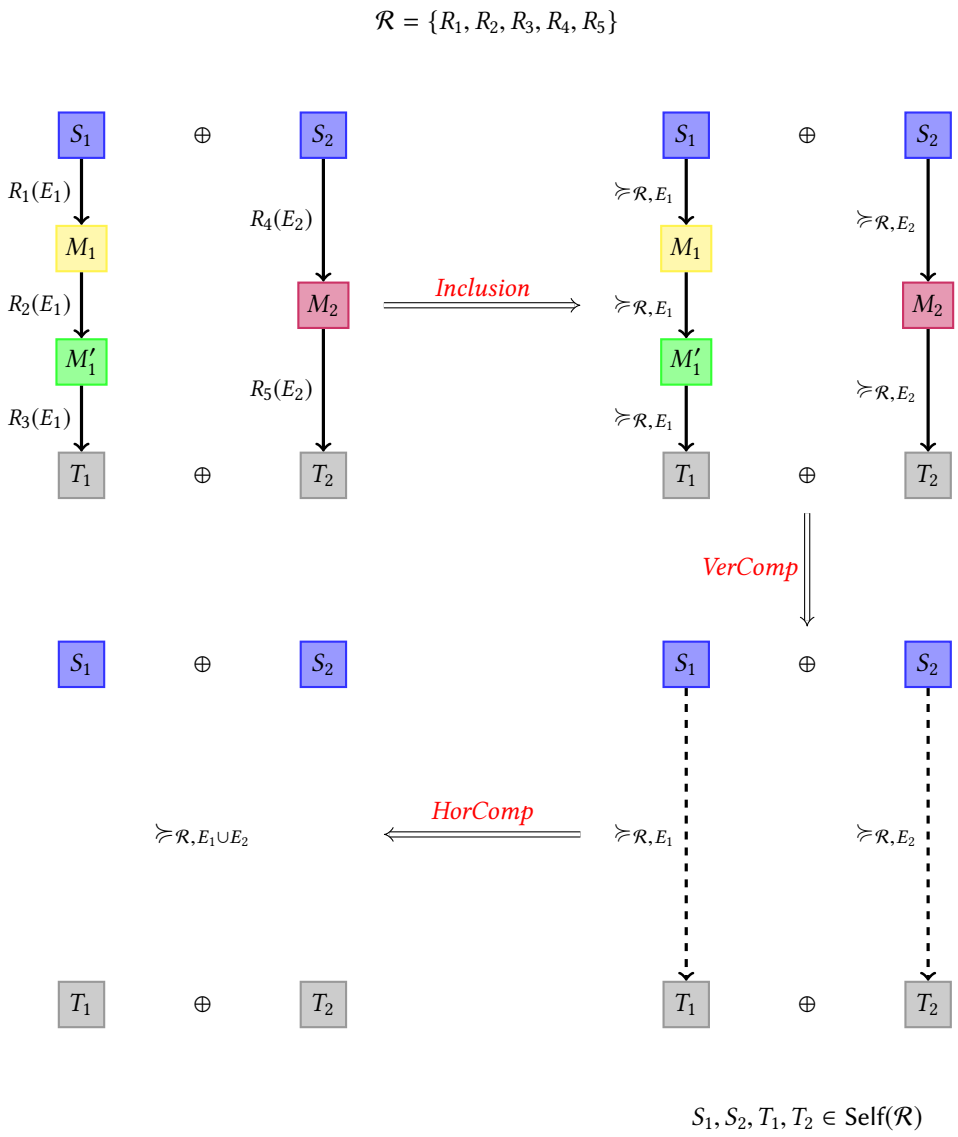


Fig. 10. Compositionality achievement

(1) For any $\Gamma, \Gamma', \Pi, \Pi' \in \mathit{MdSet}$, $E, E' \in \mathit{SyncEA}$,
if $(\Gamma, \Pi) \in R(E)$ and $(\Gamma', \Pi') \in R(E')$, then $(\Gamma \cup \Gamma, \Pi \cup \Pi') \in R(E \cup E')$;
(2) For any $\Gamma, \Pi \in \mathit{MdSet}$, $E \in \mathit{SyncEA}$,
if $(\Gamma, \Pi) \in R(E)$, then for any $\mathtt{f}_1, \ldots, \mathtt{f}_n$, **let** Γ **in** $\mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n \supseteq_E$ **let** Π **in** $\mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$.

Here, $\hat{\mathbb{P}} \supseteq_E \hat{P}$ (defined in Fig. 11) is an auxiliary refinement relation, which means the target program not only preserves the event trace of source program, but also its safety, data-race-free, analysis correctness and reach

$$\mathsf{RC\_Trans}(\widehat{\mathbb{W}},\mathbb{S}) \stackrel{\text{def}}{=} (\forall \widehat{\mathbb{W}}'.(\widehat{\mathbb{W}} \overset{\tau}{\underset{\Delta}{\Rightarrow}} \widehat{\mathbb{W}}') \implies \Delta \subseteq (\mathsf{curF}(\widehat{\mathbb{W}}) \cup \mathbb{S})) \vee (\forall \widehat{\mathbb{W}}'.(\widehat{\mathbb{W}} \overset{o}{\underset{\mathsf{emp}}{\Longrightarrow}} \widehat{\mathbb{W}}' \wedge o \neq \tau) \implies \mathsf{closed}(\widehat{\mathbb{W}}',\mathbb{S}))$$

$$\mathsf{RC\_prog}(\hat{\mathbb{P}}) \stackrel{\text{def}}{=} \forall \widehat{\mathbb{W}},\widehat{\mathbb{W}}',\mathbb{S}.\,(\hat{\mathbb{P}} \overset{load}{\Longrightarrow} \widehat{\mathbb{W}} \wedge \mathbb{S} = \mathsf{dom}(\widehat{\mathbb{W}}.\Sigma) \wedge \widehat{\mathbb{W}} \Rightarrow^* \widehat{\mathbb{W}}') \implies \mathsf{RC\_Trans}(\widehat{\mathbb{W}}',\mathbb{S})$$

$$\mathsf{ProgProp}(\hat{\mathbb{P}},E) \stackrel{\text{def}}{=} \mathsf{NPDRF}(\hat{\mathbb{P}}) \wedge \mathsf{Safe}(\hat{\mathbb{P}}) \wedge \mathsf{RC\_prog}(\hat{\mathbb{P}}) \wedge \mathsf{AC}(\hat{\mathbb{P}},E)$$

$$\hat{\mathbb{P}} \sqsupseteq \hat{P} \stackrel{\text{def}}{=} \forall \mathcal{B}.\, \mathit{PEtr}(\hat{P},\mathcal{B}) \implies \mathit{PEtr}(\hat{\mathbb{P}},\mathcal{B})$$

$$\hat{\mathbb{P}} \sqsupseteq_E \hat{P} \stackrel{\text{def}}{=} \mathsf{ProgProp}(\hat{\mathbb{P}},E) \implies (\mathsf{ProgProp}(\hat{P},E) \wedge \hat{\mathbb{P}} \sqsupseteq \hat{P})$$

Fig. 11. Auxiliary Refinement Relation

closed transition properties.

$$\mathcal{R} \in \mathcal{P}(\{R \mid \mathsf{wd}(R)\})$$

We define the refinement relation under self-related contexts below:

$$\Gamma \succcurlyeq_{\mathcal{R},E} \Pi \stackrel{\text{def}}{=} \forall c_1,c_2 \in \mathsf{Self}(\mathcal{R}),\mathtt{f}_1,\ldots,\mathtt{f}_n.\ \mathbf{let}\ (c_1 \cup \Gamma \cup c_2)\ \mathbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n \sqsupseteq_E \mathbf{let}\ (c_1 \cup \Pi \cup c_2)\ \mathbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$$

$$\mathsf{Self}(\mathcal{R}) \stackrel{\text{def}}{=} \{c \in \mathit{MdSet} \mid \forall R \in \mathcal{R},\ E \in \mathit{SyncEA}.\,(c,c) \in R(E)\}$$

We can prove that the refinement relation under self-related contexts satisfies the following properties.

Lemma 7 (Inclusion)**.** For any $\Gamma,\Pi,R \in \mathcal{R},E$, if $(\Gamma,\Pi) \in R(E)$, then $\Gamma \succcurlyeq_{\mathcal{R},E} \Pi$.

Lemma 8 (Adeqency)**.** For any Γ,Π,E, if $\Gamma \succcurlyeq_{\mathcal{R},E} \Pi$, then
$\forall \mathtt{f}_1,\ldots,\mathtt{f}_n.\ \mathbf{let}\ \Gamma\ \mathbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n \sqsupseteq_E \mathbf{let}\ \Pi\ \mathbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$.

Lemma 9 (VerComp)**.** For any Γ,Π',Π,E, if $\Gamma \succcurlyeq_{\mathcal{R},E} \Pi'$ and $\Pi' \succcurlyeq_{\mathcal{R},E} \Pi$, then $\Gamma \succcurlyeq_{\mathcal{R},E} \Pi$.

Lemma 10 (HorComp)**.**
For any $\Gamma,\Gamma',\Pi,\Pi' \in \mathsf{Self}(\mathcal{R}),E,E'$, if $\Gamma \succcurlyeq_{\mathcal{R},E} \Pi$ and $\Gamma' \succcurlyeq_{\mathcal{R},E'} \Pi'$, then $(\Gamma \cup \Gamma') \succcurlyeq_{\mathcal{R},E\cup E'} (\Pi \cup \Pi')$.

Lemma 11 (SelfComp)**.** For any $\Gamma,\Gamma' \in \mathsf{Self}(\mathcal{R})$, $\Gamma \cup \Gamma' \in \mathsf{Self}(\mathcal{R})$ holds.

We mode a sequential compiler SeqComp as following:

$$\mathsf{SeqComp} ::= (\mathsf{CodeT},\varphi)\,,\quad \text{where } \mathsf{CodeT} \in (\mathit{Module} \times \mathit{SyncEA}) \rightharpoonup \mathit{Module},\ \varphi \in \mathit{Addr} \rightharpoonup \mathit{Addr}$$

Here, as we have introduced, the inputs of code transformation CodeT are source module and analysis result.

Definition 12 (Sequential Compiler Correctness)**.** Correct(SeqComp, $sl, tl, \mathcal{R}$) holds iff
for any γ, π, ge, ge' and E, if $\mathsf{SeqComp.CodeT}(\gamma,E) = \pi$, $\lfloor \mathsf{SeqComp}.\varphi \rfloor(ge) = ge'$, then there exists $M_0,\ldots,M_n$ such that:

(1) $M_0 = \{(sl,ge,\gamma)\}$ and $M_n = \{(tl,ge',\pi)\}$; and
(2) $\forall i \in [0,n), \exists R \in \mathcal{R}, (M_i,M_{i+1}) \in R(E)$.

Theorem 13 (Final Theorem)**.** For any $\mathsf{SeqComp}_1,\ldots,\mathsf{SeqComp}_m$, $sl_1,\ldots,sl_m$, $tl_1,\ldots,tl_m$ such that $\forall i \in \{1,\ldots,m\}$, we have $\mathsf{Correct}(\mathsf{SeqComp}_i,sl_i,tl_i,\mathcal{R})$, and for any $\mathtt{f}_1,\ldots,\mathtt{f}_n,\varphi,E = E_1 \cup \cdots \cup E_m,\Gamma = \{(sl_1,ge_1,\gamma_1),\ldots,(sl_m,ge_m,\gamma_m)\}$ and $\Pi = \{(tl_1,ge'_1,\pi_1),\ldots,(tl_m,ge'_m,\pi_m)\}$, if

(1) $\forall i \in \{1,\ldots,m\}.(\mathsf{SeqComp}_i.\mathsf{CodeT}(\gamma_i, E_i) = \pi_i) \wedge \mathsf{injective}(\varphi)$
$\wedge (\mathsf{SeqComp}_i.\varphi = \varphi) \wedge \lfloor\varphi\rfloor(ge_i) = ge_i' \wedge \{(sl_i, ge_i, \gamma_i), (tl_i, ge_i', \pi_i)\} \in \mathsf{Self}(\mathcal{R})$;
(2) $\mathsf{Safe}(\mathbf{let}\ \Gamma\ \mathbf{in}\ \mathtt{f}_1 \parallel \ldots \parallel \mathtt{f}_n)$, $\mathsf{DRF}(\mathbf{let}\ \Gamma\ \mathbf{in}\ \mathtt{f}_1 \parallel \ldots \parallel \mathtt{f}_n)$;
(3) $\mathsf{AC}(\mathbf{let}\ \Gamma\ \mathbf{in}\ \mathtt{f}_1 \parallel \ldots \parallel \mathtt{f}_n, E)$; (* the analysis result is correct *)
(4) $\forall i \in \{1,\ldots,m\}.\ \mathsf{ReachClose}(sl_i, ge_i, \gamma_i)$;

then $\mathbf{let}\ \Gamma\ \mathbf{in}\ \mathtt{f}_1 \parallel \ldots \parallel \mathtt{f}_n \sqsupseteq \mathbf{let}\ \Pi\ \mathbf{in}\ \mathtt{f}_1 \parallel \ldots \parallel \mathtt{f}_n$.

Proof. According to assumption (1), the definition of sequential compilation correctness (Def. 12), and Lemma 14, we get:

$$\forall i \in \{1,\ldots,m\}.\ \{(sl_i, ge_i, \gamma_i)\} \succcurlyeq_{\mathcal{R},E_i} \{(tl_i, ge_i, \pi_i)\} \tag{3.1}$$

According to assumption (3.1), (2) and Lemma 9 (HorComp), we get:

$$\{(sl_1, ge_1, \gamma_1), \ldots, (sl_m, ge_m, \gamma_m)\} \succcurlyeq_{\mathcal{R},E} \{(tl_1, ge_1', \pi_1), \ldots, (tl_m, ge_m', \pi_m)\}$$

So that we get:

$$\Gamma \succcurlyeq_{\mathcal{R},E} \Pi \tag{3.2}$$

From (3.2) and Lemma 8, the following holds:

$$\mathbf{let}\ \Gamma\ \mathbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n \supseteq_E \mathbf{let}\ \Pi\ \mathbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n \tag{3.3}$$

Accoring to Lemma 16, 17, 18, 15, and (3.3), we get:

$$\mathbf{let}\ \Gamma\ \mathbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n \sqsupseteq \mathbf{let}\ \Pi\ \mathbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$$

And the safety, no-data-race, reach closed transition and analysis correctness properties are also preserved during compilation.

$$\mathsf{NPDRF}(\mathbf{let}\ \Pi\ \mathbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n),\quad \mathsf{Safe}(\mathbf{let}\ \Pi\ \mathbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n),$$
$$\mathsf{RC\_prog}(\mathbf{let}\ \Pi\ \mathbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n),\quad \mathsf{AC}(\mathbf{let}\ \Pi\ \mathbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n, E)$$

The CASCompCert proof has told us that if the source and target program have refinement relation under non-preemptive semantics and the source program is data-race-free (Lemma 19), they also have refinement relation under preemptive semantics. So, we finish the proof. □

Lemma 14 (Modular Correctness). For any SeqComp, $sl, tl, \gamma, \pi, ge, ge', E, \mathcal{R}$, if Correct(SeqComp, $sl, tl, \mathcal{R}$), $\mathsf{SeqComp.CodeT}(\gamma, E) = \pi$, $\lfloor\mathsf{SeqComp}.\varphi\rfloor(ge) = ge'$, then $\{(sl, ge, \gamma)\} \succcurlyeq_{\mathcal{R},E} \{(tl, ge', \pi)\}$.

Proof. This lemma can be proved by applying Lemma 7 and 9. □

Lemma 15 (ReachClosed Program). For any $i \in \{1,\ldots,m\}$. $\mathsf{ReachClose}(sl_i, ge_i, \gamma_i)$ and $\Gamma = \{(sl_1, ge_1, \gamma_1), \ldots, (sl_m, ge_m, \gamma_m)\}$, then $\mathsf{RC\_prog}(\mathbf{let}\ \Pi\ \mathbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n)$.

Lemma 16. For any $\mathtt{f}_1, \ldots, \mathtt{f}_n$ and Π, if $\mathsf{Safe}(\mathbf{let}\ \Pi\ \mathbf{in}\ \mathtt{f}_1 \parallel \ldots \parallel \mathtt{f}_n)$, then $\mathsf{Safe}(\mathbf{let}\ \Pi\ \mathbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n)$.

Lemma 17. For any $\mathtt{f}_1, \ldots, \mathtt{f}_n$ and Π, if $\mathsf{DRF}(\mathbf{let}\ \Pi\ \mathbf{in}\ \mathtt{f}_1 \parallel \ldots \parallel \mathtt{f}_n)$, then $\mathsf{NPDRF}(\mathbf{let}\ \Pi\ \mathbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n)$.

Lemma 18. For any $\mathtt{f}_1, \ldots, \mathtt{f}_n, E$ and Π, if $\mathsf{AC}(\mathbf{let}\ \Pi\ \mathbf{in}\ \mathtt{f}_1 \parallel \ldots \parallel \mathtt{f}_n, E)$, then $\mathsf{AC}(\mathbf{let}\ \Pi\ \mathbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n, E)$.

Lemma 19 (Semantics Equivalence).
For any $\Pi, \mathtt{f}_1, \ldots, \mathtt{f}_m$, if $\mathsf{DRF}(\mathbf{let}\ \Pi\ \mathbf{in}\ \mathtt{f}_1 \parallel \ldots \parallel \mathtt{f}_m)$, then $\mathbf{let}\ \Pi\ \mathbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_m \approx \mathbf{let}\ \Pi\ \mathbf{in}\ \mathtt{f}_1 \parallel \ldots \parallel \mathtt{f}_m$.

4 MODULE SET RELATION

In this section, we define a simulation relation to instantiate the module set relation R introduced in the previous section. The simulation relation is based on the *footprint-preserving* simulation defined in CASCompCert and can prove the correctness of lock elision optimization pass.

$$f\{\!\{\mathbb{S}\}\!\} \stackrel{\text{def}}{=} \{l' \mid \exists l.\, l\in\mathbb{S} \wedge f(l)=l'\}$$
$$f|_{\mathbb{S}} \stackrel{\text{def}}{=} \{(l, f(l)) \mid l \in (\mathbb{S} \cap \mathsf{dom}(f))\}$$

$$\mathsf{FPmatch}(\mu,\Delta,\delta) \;\; \text{iff} \;\; (\delta.rs \cap \mu.S \subseteq \mu.f\{\!\{\Delta.rs \cup \Delta.ws\}\!\}) \wedge (\delta.ws \cap \mu.S \subseteq \mu.f\{\!\{\Delta.ws\}\!\})$$

$$\widehat{f}(v) \stackrel{\text{def}}{=} \begin{cases} v, & \text{if } v \notin Addr \\ f(v), & \text{if } v \in Addr \wedge v \in \mathsf{dom}(f) \\ \text{undefined}, & \text{otherwise} \end{cases}$$
$$\mathsf{Inv}(f,\Sigma,\sigma) \;\; \text{iff} \;\; \forall l, l'.\; (l \in \mathsf{dom}(\Sigma) \wedge f(l) = l') \Longrightarrow (l' \in \mathsf{dom}(\sigma) \wedge \widehat{f}(\Sigma(l)) = \sigma(l'))$$

$$\mathsf{HG}(\Delta,\Sigma,\mathbb{F},\mathbb{S}) \;\; \text{iff} \;\; \Delta \subseteq (\mathbb{F}\cup\mathbb{S}) \;\wedge\; \mathsf{closed}(\mathbb{S},\Sigma)$$

$$\mathsf{LG}(\mu,(\delta,\sigma,F),(\Delta,\Sigma)) \;\; \text{iff} \;\; \delta \subseteq (F\cup\mu.S) \wedge \mathsf{closed}(\mu.S,\sigma) \wedge \mathsf{FPmatch}(\mu,\Delta,\delta) \wedge \mathsf{Inv}(\mu.f,\Sigma,\sigma)$$

$$\mathsf{R}(\Sigma,\Sigma',\mathbb{F},\mathbb{S}) \;\; \text{iff} \;\; (\Sigma \overset{\mathbb{F}}{=\!=} \Sigma') \;\wedge\; \mathsf{closed}(\mathbb{S},\Sigma') \;\wedge\; \mathsf{forward}(\Sigma,\Sigma')$$

$$\mathsf{Rely}(\mu,(\Sigma,\Sigma',\mathbb{F}),(\sigma,\sigma',F)) \;\; \text{iff} \;\; \mathsf{R}(\Sigma,\Sigma',\mathbb{F},\mu.\mathbb{S}) \wedge \mathsf{R}(\sigma,\sigma',F,\mu.S) \;\wedge\; \mathsf{Inv}(\mu.f,\Sigma',\sigma')$$

$$\lfloor\varphi\rfloor(ge) \stackrel{\text{def}}{=} \begin{cases} \{(\varphi(l),\widehat{\varphi}(v)) \mid (l,v) \in ge\}, \\ \qquad \text{if } (\mathsf{dom}(ge) \cup (\mathsf{range}(ge) \cap Addr)) \subseteq \mathsf{dom}(\varphi) \\ \text{undefined}, \qquad \text{otherwise} \end{cases}$$
$$\mathsf{initM}(\varphi,ge,\Sigma,\sigma) \;\; \text{iff} \;\; ge\subseteq\Sigma \wedge \mathsf{closed}(\Sigma) \wedge \mathsf{dom}(\sigma)=\varphi\{\!\{\mathsf{dom}(\Sigma)\}\!\} \wedge \mathsf{Inv}(\varphi,\Sigma,\sigma)$$

$$\iota \in E \;\stackrel{\text{def}}{=}\; (\iota = \mathsf{id.EntA} \wedge (\mathsf{id},\_) \in E) \vee (\iota = \mathsf{id.ExtA} \wedge (\_,\mathsf{id}) \in E)$$

Fig. 12. Footprint matching and rely/guarantee conditions

Local downward simulation. We introduce the triple μ below to record the key information about the shared memory at the source and the target and the thread-local accessing locations in source.

$$\mu \stackrel{\text{def}}{=} (\mathbb{S}, S, f), \quad \text{where } \mathbb{S}, S \in \mathcal{P}(Addr) \text{ and } f \in Addr \rightharpoonup Addr.$$

Here $\mathbb{S}$ and S specify the shared memory locations at the source and the target respectively. The partial mapping f maps locations at the source level to those at the target. We require μ to be well-formed:

$$\mathsf{wf}(\mu) \;\; \text{iff} \;\; \mathsf{injective}(\mu.f) \wedge \;\mathsf{dom}(\mu.f) = \mu.\mathbb{S} \;\wedge \mu.f\{\!\{\mu.\mathbb{S}\}\!\} = \mu.S$$

Below we define $(sl, ge, \gamma) \preccurlyeq_{\varphi,E} (tl, ge', \pi)$ to relate the *non-preemptive* executions of the source module (sl, ge, γ) and the target one (tl, ge', π). The *injective function* φ maps source addresses to the target ones.

Definition 20 (Module-Local Downward Simulation)**.**
$(sl, ge, \gamma) \preccurlyeq_{\varphi,E} (tl, ge', \pi)$ iff

(1) $\lfloor\varphi\rfloor(ge) = ge'$;
(2) for all $\mathtt{f}, \Bbbk, \Sigma, \sigma, \mathbb{F}, F$, and $\mu = (\mathsf{dom}(\Sigma), \mathsf{dom}(\sigma), \varphi|_{\mathsf{dom}(\Sigma)})$, if $sl.\mathsf{InitCore}(\gamma, \mathtt{f}) = \Bbbk$, $\mathbb{F} \cap \mathsf{dom}(\Sigma) = F \cap \mathsf{dom}(\sigma) = \emptyset$, and $\mathsf{initM}(\varphi, ge, \Sigma, \sigma)$, then there exists $i \in \mathtt{index}$, κ such that:
$$tl.\mathsf{InitCore}(\pi, \mathtt{f}) = \kappa, \text{ and } (\mathbb{F}, (\Bbbk, \Sigma), \mathtt{emp}) \preccurlyeq_{\mu}^{i,E} (F, (\kappa, \sigma), \mathtt{emp}),$$
where $(\mathbb{F}, (\Bbbk, \Sigma), \Delta) \preccurlyeq_{\mu}^{i,E} (F, (\kappa, \sigma), \delta)$ is defined in Def. 2.

The case that the switch points of source and target program do not match seems a little strange. The target program does not interact with the environment, but we do not use the σ' to establish simulation continuously. The reason is that when source program enters and exits critical section but the target program can't, we will

undo the previous executions of target from the last switch point and redo them when it switches back. When redoing these executions, the state of target program will satisfy σ''.

Then, we define $\hat{R}(\varphi)$ below as an instantiation of the module set relation:

$$\begin{aligned}\hat{R}(\varphi) \quad \stackrel{\text{def}}{=} \quad & \lambda E.\, \{(\Gamma, \Pi) \mid |\Gamma| = |\Pi| \wedge (\forall i \in \{1, \ldots, |\Gamma|\}.\, \Gamma(i) = (sl_i, ge_i, \gamma_i) \wedge \Pi(i) = (tl_i, ge'_i, \pi_i) \\ & \wedge (\exists E_i \subseteq E.\, (sl_i, ge_i, \gamma_i) \preccurlyeq_{\varphi}^{E_i} (tl_i, ge'_i, \pi_i)) \wedge \mathsf{wd}(sl_i) \wedge \mathsf{wd}(tl_i) \wedge \mathsf{det}(tl_i))\}\end{aligned}$$

The subset of E is the set of synchronization pairs that the compiler is responsible to eliminate. The reason why we choose a subset of E is because some passes don't eliminate synchronization, and we require the synchronization in E_i must be eliminated to avoid that the compiler only eliminates only `lock()` or `unlock()` operation of a synchronization. For example, we need to avoid the following situation occuring:

```
lock()^id1;                          lock()^id1;
  x := 2;      ==compile==>              x := 2;
unlock()^id2;
```

We need to prove that $\hat{R}(\varphi)$ is well-defined. A module set relation is well-defined, if it satisfies the *HorComp* and *Adequency* properties as shown in Def. 6. The injective function φ will be instantiated as *identity*, *extension* and *injection* in the proof of compilation correctness. And the set $\mathcal{R}$ is instantiated as the following form in the proof of compilation correctness.

$$\mathcal{R} \stackrel{\text{def}}{=} \{\hat{R}(\varphi) \mid \varphi \in \{\mathit{identity},\ \mathit{extension},\ \mathit{injection}\}\}$$

Theorem 21 (HorComp)**.**
For any $\Gamma, \Gamma', \Pi, \Pi', E, E'$, and φ, if $(\Gamma, \Pi) \in \hat{R}(\varphi)(E)$ and $(\Gamma', \Pi') \in \hat{R}(\varphi)(E')$, then $(\Gamma \cup \Gamma', \Pi \cup \Pi') \in \hat{R}(\varphi)(E \cup E')$.

Proof. The correctness proof of this lemma can be achieved from the definition of $\hat{R}(\varphi)$. The only thing we need to prove that if there exists $E_i \subseteq E$ and $(sl_i, ge_i, \gamma_i) \preccurlyeq_{\varphi, E_i} (tl_i, ge'_i, \pi_i)$, then the following holds:

$$\exists E_i \subseteq (E \cup E').\, (sl_i, ge_i, \gamma_i) \preccurlyeq_{\varphi, E_i} (tl_i, ge'_i, \pi_i)$$

Its correctness proof is straight-forward. □

Theorem 22 (Adequency)**.** For any Γ, Π, E and φ, if $(\Gamma, \Pi) \in \hat{R}(\varphi)(E)$ holds, then

$$\forall \mathtt{f}_1, \ldots, \mathtt{f}_n.\, \textbf{let}\ \Gamma\ \textbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_1 \supseteq_E \textbf{let}\ \Pi\ \textbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$$

Proof. We first unfold $(\Gamma, \Pi) \in \hat{R}(\varphi)(E)$ according to its definition, and we get that for any $i \in \{1, \ldots, |\Gamma|\}$, there exists $E_i \subseteq E$, such that

$$(sl_i, ge_i, \gamma_i) \preccurlyeq_{\varphi, E_i} (tl_i, ge'_i, \pi_i) \wedge \mathsf{wd}(sl_i) \wedge \mathsf{wd}(tl_i) \wedge \mathsf{det}(tl_i) \tag{4.1}$$

Let $\hat{\mathbb{P}} = \textbf{let}\ \Gamma\ \textbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$, and $\hat{P} = \textbf{let}\ \Pi\ \textbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$. Then we unfold the proof goal by its definition, and do intro:

$$\mathsf{NPDRF}(\hat{\mathbb{P}}) \tag{4.2}$$

$$\mathsf{Safe}(\hat{\mathbb{P}}) \tag{4.3}$$

$$\mathsf{RC\_prog}(\hat{\mathbb{P}}) \tag{4.4}$$

$$\mathsf{AC}(\hat{\mathbb{P}}, E) \tag{4.5}$$

And we need to prove that the target program preserves these properties of source program:

$$\hat{\mathbb{P}} \sqsupseteq \hat{P} \wedge \mathsf{NPDRF}(\hat{P}) \wedge \mathsf{Safe}(\hat{P}) \wedge \mathsf{RC\_prog}(\hat{P}) \wedge \mathsf{AC}(\hat{P}, E) \tag{g}$$

According to (4.1) (4.4) (4.2) (4.5) and Lemma 23, we know that the module local simulation can compose to a whole program downward simulation,

$$\textbf{let } \Gamma \textbf{ in } \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n \preccurlyeq_E \textbf{let } \Pi \textbf{ in } \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n \tag{4.6}$$

By applying Lemma 24 on (E.1), we get that the whole program downward simulation can flip to a whole program upward simulation,

$$\textbf{let } \Pi \textbf{ in } \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n \leqslant_E \textbf{let } \Gamma \textbf{ in } \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n \tag{4.7}$$

According to (4.7) and Lemma 29, 25, 28 and 27, we know that the target program preserves the event trace, data-race-free, safety, reachclose and analysis correctness properties of source program. We finish the proof. □

Lemma 23 (Compositionality). For any $\mathtt{f}_1, \ldots, \mathtt{f}_n, \varphi, E, E_1, \ldots, E_m, \Gamma = \{(sl_1, ge_1, \gamma_1), \ldots, (sl_m, ge_m, \gamma_m)\}, \Pi = \{(tl_1, ge'_1, \pi_1), \ldots, (tl_m, ge'_m, \pi_m)\}$, if

- $\forall i \in \{1, \ldots, m\}. (sl_i, ge_i, \gamma_i) \preccurlyeq_{\varphi, E_i} (tl_i, ge'_i, \pi_i) \wedge \mathsf{wd}(sl_i) \wedge \mathsf{wd}(tl_i) \wedge E_i \subseteq E$;
- RC_prog(**let** Γ **in** $\mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$), NPDRF(**let** Γ **in** $\mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$), AC(**let** Π **in** $\mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n, E$);

then **let** Γ **in** $\mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n \preccurlyeq_E$ **let** Π **in** $\mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$.

Proof. The details of the proof of compositionality can be found in Sec. C □

Lemma 24 (Flip). For any $\mathtt{f}_1, \ldots, \mathtt{f}_n, \varphi, \Gamma = \{(sl_1, ge_1, \gamma_1), \ldots, (tl_m, ge'_m, \gamma_m)\}, \Pi = \{(tl_1, ge'_1, \pi'_1), \ldots, (tl_m, ge'_m, \pi_m)\}$, if $\forall 1 \leq i \leq m. \mathsf{det}(tl_i)$, Safe(**let** Γ **in** $\mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$) and **let** Γ **in** $\mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n \preccurlyeq_E$ **let** Π **in** $\mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$, then

$$\textbf{let } \Pi \textbf{ in } \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n \leqslant_E \textbf{let } \Gamma \textbf{ in } \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n.$$

Proof. The details of the proof of flip can be found in Sec. D. □

Lemma 25 (Safety Preservation). For any $\hat{P} \leqslant_E \hat{\mathbb{P}}$ and Safe($\hat{\mathbb{P}}$), then Safe($\hat{P}$).

Lemma 26 (NPDRF Preservation). For any $\hat{\mathbb{P}}, \hat{P}, E$, if $\hat{P} \leqslant_E \hat{\mathbb{P}}$, AC($\hat{\mathbb{P}}, E$), and NPDRF($\hat{\mathbb{P}}$), then NPDRF($\hat{P}$).

Proof. Details of proof of NPDRF Preservation can be found in Sec. E. □

Lemma 27 (Simulation ensures ReachClose). If $\hat{P} \leqslant_E \hat{\mathbb{P}}$, then RC_prog($\hat{P}$).

Proof. We finish the proof by the definition of whole program downward simulation and Lemma 86. (* on paper proof *) □

Lemma 28 (Analysis correctness Preservation). If $\hat{P} \leqslant_E \hat{\mathbb{P}}$, and AC($\hat{\mathbb{P}}, E$), then AC($\hat{P}, E$).

Proof. Details of proof of AC Preservation can be found in Sec. F. □

Lemma 29 (Soundness). If $\hat{P} \leqslant_E \hat{\mathbb{P}}$, then $\hat{\mathbb{P}} \sqsupseteq \hat{P}$.

4.1 Whole program simulation

We define whole program simulation in this section. In CASCompCert, there are two types of whole program simulation. The whole program downward simulation constructs the target program execution according to the execution of source program. The whole program downward simulation can be flipped to a whole program upward simulation and can be used to prove the DRF preserving in source program.

$$\frac{}{(T,\sigma) \Rrightarrow^{\mu}_{\emptyset} (T,\sigma)} \qquad \frac{\begin{array}{c}(T_0,\sigma_0) \Rrightarrow^{\mu}_{B\backslash\{\mathtt{t}\}} (T,\sigma) \quad T(\mathtt{t}) = (tl,F,\kappa) \quad B(\mathtt{t}) = (n,\delta,\widehat{W}) \\ F \vdash (\kappa,\sigma) \overset{\tau}{\underset{\delta}{\mapsto}}{}^{n} (\kappa',\sigma') \quad \delta \subseteq (F \cup \mu.S) \quad T' = T\{\mathtt{t} \rightsquigarrow (tl,F,\kappa')\}\end{array}}{(T_0,\sigma_0) \Rrightarrow^{\mu}_{B} (T',\sigma')}$$

$$d_1 \leq d_2 \overset{\text{def}}{=} d_1 = d_2 \vee (\exists \mathrm{id}.\, d_1 = 0 \wedge d_2 = \mathrm{id})$$

Fig. 13. Auxiliary definitions for Whole program downward simulation

Whole-program downward simulation. We introduce a tuple p, which records the some information for the previous switching point:

$$p \overset{\text{def}}{=} (\widehat{W}, B) \qquad (\text{where } B \overset{\text{def}}{=} \{\mathtt{t} \rightsquigarrow (n,\delta,\widehat{W})\}^*)$$

Definition 30 (Whole-Program Downward Simulation)**.** Let $\hat{\mathbb{P}} =$ **let** Γ **in** $\mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$, $\hat{P} =$ **let** Π **in** $\mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$, where $\Gamma = \{(sl_1, ge_1, \gamma_1), \ldots, (sl_m, ge_m, \gamma_m)\}$, $\Pi = \{(tl_1, ge'_1, \pi_1), \ldots, (tl_m, ge'_m, \pi_m)\}$. We say $\hat{\mathbb{P}} \preccurlyeq_E \hat{P}$ iff

- $\exists \varphi.\, (\forall 1 \leq i \leq m.\, \lfloor \varphi \rfloor ge_i = ge'_i)$, and
- for any $\widehat{\mathbb{W}}$, if $\hat{\mathbb{P}} :\overset{load}{\Longrightarrow} \widehat{\mathbb{W}}$, then there exists $\widehat{W}, i \in \mathtt{index}, \mu$, such that:
  $\hat{P} :\overset{load}{\Longrightarrow} \widehat{W} \wedge (\widehat{\mathbb{W}}, \mathtt{emp}) \preccurlyeq^{i,\mu,E}_{(\widehat{W},\emptyset),0} (\widehat{W}, \mathtt{emp}, \mathtt{emp})$.

Here we define $(\widehat{\mathbb{W}}, \Delta_0) \preccurlyeq^{i,\mu,E}_{p,n_0} (\widehat{W}, \delta_0, \delta_r)$ as the largest relation such that whenever $(\widehat{\mathbb{W}}, \Delta_0) \preccurlyeq^{i,\mu,E}_{p,n_0} (\widehat{W}, \delta_0, \delta_r)$ holds, then there exists $\mathtt{t}$, where $\widehat{W}.\mathtt{t} = \widehat{\mathbb{W}}.\mathtt{t} = \mathtt{t}$, such that the following are true:

(1) $\mathtt{t} \notin \mathrm{dom}(B)$, $\mathrm{dom}(p.B) \subseteq \mathrm{dom}(\mathbb{T})$, $\widehat{W}.\mathbb{d}(\mathtt{t}) \leq \widehat{\mathbb{W}}.\mathbb{d}(\mathtt{t})$ and $p.\widehat{W} :\overset{\tau}{\underset{\delta_r \cup \delta_0}{\Longrightarrow}}{}^{n_0} \widehat{W}$;

(2) $\forall \widehat{\mathbb{W}}', \Delta.$ if $\widehat{\mathbb{W}} \overset{\tau}{\underset{\Delta}{\Rightarrow}} \widehat{\mathbb{W}}'$, then one of the following holds:

(a) $\exists j < i.\, (\widehat{\mathbb{W}}', \Delta_0 \cup \Delta) \preccurlyeq^{j,\mu,E}_{p,n_0} (\widehat{W}, \delta_0, \delta_r)$; or

(b) $\exists \widehat{W}', \delta, j, n > 0$ such that:

(i) $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{n} \widehat{W}'$, and

(ii) $(\delta_0 \cup \delta) \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta, \delta_0 \cup \delta)$, and

(iii) $(\widehat{\mathbb{W}}', \Delta_0 \cup \Delta) \preccurlyeq^{j,\mu,E}_{p,n_0+n} (\widehat{W}', \delta_0 \cup \delta, \delta_r)$;

(3) $\forall \mathbb{T}', \mathtt{t}', \Sigma', o, \mathbb{d}'$, if $o \neq \tau$, and $\widehat{\mathbb{W}} :\overset{o}{\underset{\mathsf{emp}}{\Longrightarrow}} (\mathbb{T}', \mathtt{t}', \mathbb{d}', \Sigma')$, then one of the following holds:

(a) (* switch point match *) (* update the last switch point, and remove $\mathtt{t}''$ from reorder buffer *)
$\exists \widehat{W}', \delta, T', \sigma', \mathbb{d}'_1, j$, for any $\mathtt{t}'' \in \mathrm{dom}(\mathbb{T}')$, let $p' = ((T', \mathtt{t}'', \mathbb{d}'_1, \sigma'), B\backslash\{\mathtt{t}''\})$, such that

(i) $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{*} \widehat{W}'$, and $\widehat{W}' :\overset{o}{\underset{\mathsf{emp}}{\Longrightarrow}} (T', \mathtt{t}'', \mathbb{d}'_1, \sigma')$;

(ii) $(\delta_0 \cup \delta) \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0, \delta_0 \cup \delta)$;

(iii) $\exists T'', \sigma''.\, (T', \sigma') \Rrightarrow^{\mu}_{p.B} (T'', \sigma'') \wedge \mathsf{closed}(\mu.S, \sigma'')$;

(iv) if $p.B(\mathtt{t}'') = (n'', \delta'', \widehat{W}'')$, then:

- $\exists \widehat{W}''_1.\, (T', \mathtt{t}'', \mathbb{d}'_1, \sigma') :\overset{\tau}{\underset{\delta''}{\Longrightarrow}}{}^{n''} \widehat{W}''_1 \wedge \widehat{W}'' \overset{\delta''}{=\!=} \widehat{W}''_1$; (* redo executions of $\mathtt{t}''$ in buffer *)

- For any $\widehat{W}_1''$, if $(T', \mathtt{t}'', \mathbb{d}_1', \sigma') :\overset{\tau}{\underset{\delta''}{\Longrightarrow}}{}^{n''} \widehat{W}_1''$ and $\widehat{W}'' \overset{\delta''}{=\!=} \widehat{W}_1''$, then
$$((\mathbb{T}', \mathtt{t}'', \mathbb{d}', \Sigma'), \mathsf{emp}) \preccurlyeq^{j,\mu,E}_{p',n''} (\widehat{W}_1'', \mathsf{emp}, \delta'');$$

else (* no execution of $\mathtt{t}''$ in buffer *)
$$((\mathbb{T}', \mathtt{t}'', \mathbb{d}', \Sigma'), \mathsf{emp}) \preccurlyeq^{j,\mu,E}_{p',0} ((T', \mathtt{t}'', \mathbb{d}_1', \sigma'), \mathsf{emp}, \mathsf{emp}); \text{ or}$$

(b) (* switch point not match *)
$\exists \widehat{W}', \delta, n, j$, for any $\mathtt{t}'' \in \mathsf{dom}(\mathbb{T}')$, such that
(i) $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{n} \widehat{W}'$, $(n = 0 \implies j < i)$, $o \in E$; (* synchronization eliminated must in E *)
(ii) $(\delta_0 \cup \delta) \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0, \delta_0 \cup \delta)$;
(iii) (* undo the executions of current thread, and rollback to the last switch point *)
there exists $B = p.B\{\mathtt{t} \rightsquigarrow (n_0+n, \delta_r \cup \delta_0 \cup \delta, \widehat{W}')\}$, and $p' = (p.\widehat{W}^{\mathtt{t}''}, B\backslash\{\mathtt{t}''\})$ such that:
if $B(\mathtt{t}'') = (n'', \delta'', \widehat{W}'')$, then

- $\exists \widehat{W}_1''.\ p.\widehat{W}^{\mathtt{t}''} :\overset{\tau}{\underset{\delta''}{\Longrightarrow}}{}^{n''} \widehat{W}_1'' \wedge \widehat{W}'' \overset{\delta''}{=\!=} \widehat{W}_1''$; (* redo executions of $\mathtt{t}''$ in buffer *)
- For any $\widehat{W}_1''$, if $p.\widehat{W}^{\mathtt{t}''} :\overset{\tau}{\underset{\delta''}{\Longrightarrow}}{}^{n''} \widehat{W}_1''$ and $\widehat{W}'' \overset{\delta''}{=\!=} \widehat{W}_1''$, then
$$((\mathbb{T}', \mathtt{t}'', \mathbb{d}', \Sigma'), \mathsf{emp}) \preccurlyeq^{j,\mu,E}_{p',n''} (\widehat{W}_1'', \mathsf{emp}, \delta'');$$

else (* no execution of $\mathtt{t}''$ in buffer *)
$$((\mathbb{T}', \mathtt{t}'', \mathbb{d}', \Sigma'), \mathsf{emp}) \preccurlyeq^{j,\mu,E}_{p',0} (p.\widehat{W}^{\mathtt{t}''}, \mathsf{emp}, \mathsf{emp});$$

(4) if $\widehat{\mathbb{W}} :\overset{\tau}{\underset{\mathsf{emp}}{\Longrightarrow}}$ **done**, then $\exists \widehat{W}', \delta$.
(a) $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{*} \widehat{W}'$, $\widehat{W}' :\overset{\tau}{\underset{\mathsf{emp}}{\Longrightarrow}}$ **done**;
(b) $(\delta_0 \cup \delta) \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0, \delta_0 \cup \delta)$;

(** Done case proof **)

Lemma 31 (Whole Program Downward Simulation implies Done Preserving - II)**.** For any $\widehat{\mathbb{W}}$, Δ_0, $\widehat{W}$, δ_0, δ_r, i, μ, $\widehat{W}_0$, E, B, and n_0, if $\widehat{\mathbb{W}} \Rightarrow^{+}$ **done**, and $(\widehat{\mathbb{W}}, \Delta_0) \preccurlyeq^{i,\mu,E}_{(\widehat{W}_0,B),n_0} (\widehat{W}, \delta_0, \delta_r)$, then $\exists \mathtt{t}.\ Etr(\widehat{W}_0^{\mathtt{t}}, \textbf{done})$.

Proof. Prove by induction on the number of source steps. (* on paper proof *) □

(** Silent diverge case proof **)

Lemma 32 (A thread executes forever existing)**.** For any $\widehat{\mathbb{W}}$, $\widehat{W}$, Δ_0, δ_0, δ_r, $\widehat{W}_0$, B, k, μ, E, lt, if $(\widehat{\mathbb{W}}, \Delta_0) \preceq^{i,\mu,E}_{(\widehat{W}_0,B),n_0} (\widehat{W}, \delta_0, \delta_r)$, $\widehat{W}_0 \Downarrow B$, $Etr(\widehat{\mathbb{W}}, \epsilon)$, $lt \subseteq \mathsf{dom}(\widehat{W}_0.T)$, $\mathsf{isdone}_{\mu,E}(\widehat{\mathbb{W}}, \widehat{W}_0, B \cup \{\widehat{W}_0.\mathtt{t} \rightsquigarrow (n_0, \delta_r \cup \delta_0, \widehat{W})\}, lt)$, $k = |\widehat{W}_0.T| - |lt|$, then $\exists \mathtt{t} \in \mathsf{dom}(\widehat{W}_0.T)$, $\widehat{\mathbb{W}}', \Delta_0', \widehat{W}', \delta_0', \delta_r', \mathtt{t}_0, B', n_0', j, lt'$,

$$\widehat{\mathbb{W}} \Rightarrow^{*} \widehat{\mathbb{W}}' \wedge (\widehat{\mathbb{W}}', \Delta_0') \preceq^{j,\mu,E}_{(\widehat{W}_0^{\mathtt{t}_0},B'),n_0'} (\widehat{W}', \delta_0', \delta_r') \wedge Etr(\widehat{\mathbb{W}}', \epsilon) \wedge \widehat{W}_0 \Downarrow B' \wedge lt' \subseteq \mathsf{dom}(\widehat{W}_0.T)$$
$$\wedge \neg \mathsf{willdone}_{\mu,E}(\widehat{\mathbb{W}}', \widehat{W}_0, lt', \mathtt{t}) \wedge \mathsf{isdone}_{\mu,E}(\widehat{\mathbb{W}}', \widehat{W}_0, B' \cup \{\mathtt{t}_0 \rightsquigarrow (n_0', \delta_r' \cup \delta_0', \widehat{W}')\}, lt').$$

Proof. Prove by induction on the number of active threads k. (* on paper proof *) □

$$\frac{\widehat{\mathbb{W}} \Rrightarrow^* \widehat{\mathbb{W}}' \quad \widehat{\mathbb{W}}' \overset{o}{\Rrightarrow} \widehat{\mathbb{W}}'' \quad \widehat{W}_0^{\mathtt{t}} \Rrightarrow^* \widehat{W}_0' \quad \widehat{W}_0' \overset{o}{\Rrightarrow} \widehat{W}_0'' \quad (\widehat{\mathbb{W}}'', \Delta_0) \preccurlyeq^{i,\mu,E}_{(\widehat{W}_0'', B), n_0} (\widehat{W}'', \delta_0) \quad \widehat{W}_0'' \Downarrow B \quad Etr(\widehat{\mathbb{W}}'', \epsilon) \quad o \notin \{\tau, e\}}{\mathsf{SPmatchEx}_{\mu,E}(\widehat{\mathbb{W}}, \widehat{W}_0)}$$

$$(\widehat{\mathbb{W}}, \Delta_0) \preceq^{i,\mu,E}_{p,n_0} (\widehat{W}, \delta_0) \overset{\text{def}}{=} (\widehat{\mathbb{W}}, \Delta_0) \preccurlyeq^{i,\mu,E}_{p,n_0} (\widehat{W}, \delta_0) \wedge \neg \mathsf{SPmatchEx}_{\mu,E}(\widehat{\mathbb{W}}, p.\widehat{W}_0)$$

$$(n_1, \delta_1, \widehat{W}_1) \geq_{\widehat{W}_0} \bot \overset{\text{def}}{=} \widehat{W}_0 :\overset{\tau}{\underset{\delta_1}{\Rrightarrow}}{}^{n_1} \widehat{W}_1$$

$$(n_1, \delta_1, \widehat{W}_1) \geq_{\widehat{W}_0} (n_2, \delta_2, \widehat{W}_2) \overset{\text{def}}{=} \exists n, \delta.\, n_1 = n + n_2 \wedge \widehat{W}_1 \overset{\tau}{\underset{\delta}{\Rrightarrow}}{}^{n} \widehat{W}_2 \wedge \delta \cup \delta_2 = \delta_1$$

$$(n_1, \delta_1, \widehat{W}_1) >_{\widehat{W}_0} \bot \overset{\text{def}}{=} n_1 > 0 \wedge (n_1, \delta_1, \widehat{W}_1) \geq_{\widehat{W}_0} \bot$$

$$(n_1, \delta_1, \widehat{W}_1) >_{\widehat{W}_0} (n_2, \delta_2, \widehat{W}_2) \overset{\text{def}}{=} n_1 > n_2 \wedge (n_1, \delta_1, \widehat{W}_1) \geq_{\widehat{W}_0} (n_2, \delta_2, \widehat{W}_2)$$

$$B' >_{\mathtt{t}, \widehat{W}_0} B \overset{\text{def}}{=} B'(\mathtt{t}) >_{\widehat{W}_0^{\mathtt{t}}} B(\mathtt{t}) \wedge (\forall \mathtt{t}' \in \mathsf{dom}(B'), \mathtt{t}' \neq \mathtt{t}.\, B'(\mathtt{t}') \geq_{\widehat{W}_0^{\mathtt{t}'}} B(\mathtt{t}'))$$

$$\begin{aligned}\mathsf{isdone}_{\mu,E}(\widehat{\mathbb{W}}, \widehat{W}_0, B, lt) \overset{\text{def}}{=}\ & \forall \widehat{\mathbb{W}}', \widehat{W}', \Delta_0', \delta_0', \delta_r', i, B_0, n_0, \mathtt{t}. \\ & (\widehat{\mathbb{W}} \Rrightarrow^* \widehat{\mathbb{W}}' \wedge (\widehat{\mathbb{W}}', \Delta_0') \preceq^{i,\mu,E}_{(\widehat{W}_0^{\mathtt{t}}, B_0), n_0} (\widehat{W}', \delta_0', \delta_r') \wedge \widehat{W}_0 \Downarrow B_0 \\ & \wedge (B_0 \cup \{\mathtt{t} \rightsquigarrow (n_0, \delta_r' \cup \delta_0', \widehat{W}')\}) >_{\mathtt{t}, \widehat{W}_0} B \wedge Etr(\widehat{\mathbb{W}}', \epsilon)) \implies \mathtt{t} \notin lt\end{aligned}$$

$$\begin{aligned}\mathsf{willdone}_{\mu,E}(\widehat{\mathbb{W}}, \widehat{W}_0, lt, \mathtt{t}) \overset{\text{def}}{=}\ & \exists \widehat{\mathbb{W}}', \widehat{W}', i, B, n_0, \Delta_0, \delta_0, \delta_r, \mathtt{t}_0. \\ & \widehat{\mathbb{W}} \Rrightarrow^* \widehat{\mathbb{W}}' \wedge (\widehat{\mathbb{W}}', \Delta_0) \preceq^{i,\mu,E}_{(\widehat{W}_0^{\mathtt{t}_0}, B), n_0} (\widehat{W}', \delta_0, \delta_r) \wedge Etr(\widehat{\mathbb{W}}', \epsilon) \wedge \widehat{W}_0 \Downarrow B \\ & \wedge \mathsf{isdone}_{\mu,E}(\widehat{\mathbb{W}}', \widehat{W}_0, B \cup \{\mathtt{t}_0 \rightsquigarrow (n_0, \delta_r \cup \delta_0, \widehat{W}')\}, \{\mathtt{t}\} \cup lt)\end{aligned}$$

$$\widehat{W}_0 \Downarrow B \overset{\text{def}}{=} \forall \mathtt{t} \in \mathsf{dom}(B).\, B(\mathtt{t}) = (n, \delta, \widehat{W}) \implies \widehat{W}_0^{\mathtt{t}} \overset{\tau}{\underset{\delta}{\Rrightarrow}}{}^{n} \widehat{W}$$

$$(T, \mathtt{t}, \mathbb{d}, \sigma) \overset{\delta}{=\!=} (T', \mathtt{t}', \mathbb{d}', \sigma') \overset{\text{def}}{=} \mathtt{t} = \mathtt{t}' \wedge T(\mathtt{t}) = T'(\mathtt{t}') \wedge \sigma \overset{\delta.rs \cup \delta.ws}{=\!=\!=} \sigma'$$

Fig. 14. Auxiliary definition in proving revised whole program downward simulation implies refinement

Lemma 33 (Silent diverge implies a thread progress)**.** For any $\widehat{\mathbb{W}}$, $\widehat{W}$, Δ_0, δ_0, i, μ, E, $\widehat{W}_0$ and n_0, if $(\widehat{\mathbb{W}}, \Delta_0) \preceq^{i,\mu,E}_{(\widehat{W}_0, B), n_0} (\widehat{W}, \delta_0, \delta_r)$, $\widehat{W}_0 \Downarrow B$ and $Etr(\widehat{\mathbb{W}}, \epsilon)$, then $\exists \widehat{\mathbb{W}}', \widehat{W}', n_0', \Delta_0', \delta_0', \delta_r', j, \mathtt{t}, B'$.

$$\begin{aligned}\widehat{\mathbb{W}} \Rrightarrow^* \widehat{\mathbb{W}}' \wedge (\widehat{\mathbb{W}}', \Delta_0') \preceq^{j,\mu,E}_{(\widehat{W}_0^{\mathtt{t}}, B'), n_0'} (\widehat{W}', \delta_0', \delta_r') \wedge \widehat{W}_0 \Downarrow B' \\ \wedge (B' \cup \{\mathtt{t} \rightsquigarrow (n_0', \delta_r' \cup \delta_0', \widehat{W}')\}) >_{\mathtt{t}, \widehat{W}_0} (B \cup \{\widehat{W}_0.\mathtt{t} \rightsquigarrow (n_0, \delta_r \cup \delta_0, \widehat{W})\}) \wedge Etr(\widehat{\mathbb{W}}', \epsilon).\end{aligned}$$

Proof. Prove by induction on the index i. (* on paper proof *) □

Lemma 34 (Lemma for source silent diverge implies target diverge)**.** For any $\widehat{\mathbb{W}}_0$, $\widehat{\mathbb{W}}$, Δ_0, $\widehat{W}$, δ_0, δ_r, $\widehat{W}_0$, B, n_0, i, μ, E, lt and $\mathtt{t}$, if $\widehat{\mathbb{W}}_0 \Rrightarrow^* \widehat{\mathbb{W}}$, $(\widehat{\mathbb{W}}, \Delta_0) \preceq^{i,\mu,E}_{(\widehat{W}_0^{\mathtt{t}}, B), n_0} (\widehat{W}, \delta_0, \delta_r)$, $\neg \mathsf{willdone}_{\mu,E}(\widehat{\mathbb{W}}, \widehat{W}_0, lt, \mathtt{t})$, $lt \subseteq \mathsf{dom}(\widehat{W}_0.T)$, $\mathsf{isdone}_{\mu,E}(\widehat{\mathbb{W}}, \widehat{W}_0, B \cup \{\mathtt{t} \rightsquigarrow (n_0, \delta_r \cup \delta_0, \widehat{W})\}, lt)$, $\widehat{W}_0 \Downarrow B$, $Etr(\widehat{\mathbb{W}}, \epsilon)$, then $Etr(\widehat{W}, \epsilon)$.

Proof. Prove by cofix. (* on paper proof *) □

Lemma 35 (Source silent diverge implies target diverge). For any $\widehat{\mathbb{W}}$, $\widehat{W}$, Δ_0, δ_0, δ_r, i, μ, E, $\widehat{W}_0$, B, and n_0, if $(\widehat{\mathbb{W}}, \Delta_0) \preceq^{i,\mu,E}_{(\widehat{W}_0,B),n_0} (\widehat{W}, \delta_0, \delta_r)$, $\widehat{W}_0 \Downarrow B$ and $Etr(\widehat{\mathbb{W}}, \epsilon)$, then $\exists \mathtt{t}.\, Etr(\widehat{W}_0^{\mathtt{t}}, \epsilon)$.

Proof. We finish the proof by applying Lemma 32, 33, and 34. (* on paper proof *) □

Lemma 36 (Source diverge and switch point match existing implies target diverge). For any $\widehat{\mathbb{W}}$, $\widehat{\mathbb{W}}'$, $\widehat{\mathbb{W}}''$, $\widehat{W}_0$, $\widehat{W}_0'$, $\widehat{W}_0''$, $\widehat{W}''$, $\mathtt{t}$, o, Δ_0, δ_0, δ_r, i, μ, E, B, and n_0, if $\widehat{\mathbb{W}} \Rightarrow^* \widehat{\mathbb{W}}'$, $\widehat{\mathbb{W}}' \overset{o}{\Rightarrow} \widehat{\mathbb{W}}''$, $\widehat{W}_0^{\mathtt{t}} \Rightarrow^* \widehat{W}_0'$, $\widehat{W}_0' \overset{o}{\Rightarrow} \widehat{W}_0''$, $o \notin \{\tau, e\}$, $(\widehat{\mathbb{W}}'', \Delta_0) \preccurlyeq^{i,\mu,E}_{(\widehat{W}_0'',B),n_0} (\widehat{W}'', \delta_0, \delta_r)$, $\widehat{W}_0'' \Downarrow B$ and $Etr(\widehat{\mathbb{W}}'', \epsilon)$, then $Etr(\widehat{W}_0^{\mathtt{t}}, \epsilon)$.

Proof. Prove by cofix. (* on paper proof *) □

Lemma 37. $\mathsf{isdone}(\widehat{\mathbb{W}}, \widehat{W}_0, B, \emptyset)$

Lemma 38. $(\mathsf{isdone}(\widehat{\mathbb{W}}, \widehat{W}_0, B, lt) \wedge B' >_{\mathtt{t},\widehat{W}_0} B) \implies \mathsf{isdone}(\widehat{\mathbb{W}}, \widehat{W}_0, B', lt)$

Lemma 39. $(\widehat{\mathbb{W}} \Rightarrow^* \widehat{\mathbb{W}}' \wedge \mathsf{isdone}(\widehat{\mathbb{W}}, \widehat{W}_0, B, lt)) \implies \mathsf{isdone}(\widehat{\mathbb{W}}', \widehat{W}_0, B, lt)$

Lemma 40. $(\widehat{\mathbb{W}} \Rightarrow^* \widehat{\mathbb{W}}' \wedge \neg\mathsf{willdone}(\widehat{\mathbb{W}}, \widehat{W}_0, lt, \mathtt{t})) \implies \neg\mathsf{willdone}(\widehat{\mathbb{W}}', \widehat{W}_0, lt, \mathtt{t})$

Lemma 41 (Buffer execution exists). For any B and $\widehat{W}_0$, if $(\forall \mathtt{t}, n, \delta, \widehat{W}.\, B(\mathtt{t}) = (n, \delta, \widehat{W}) \implies (\exists \widehat{W}_1.\, \widehat{W}_0^{\mathtt{t}} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{n} \widehat{W}_1 \wedge \widehat{W} \overset{\delta}{=\!=} \widehat{W}_1))$, then $\exists B'.\, \widehat{W}_0 \Downarrow B' \wedge B' \simeq B \wedge \mathsf{dom}(B') = \mathsf{dom}(B)$.

Proof. Prove by induction on the number of elements in buffer B. □

$$B' \simeq B \overset{\text{def}}{=} \forall \mathtt{t} \in \mathsf{dom}(B'), n, \delta, \widehat{W}'.\, B'(\mathtt{t}) = (n, \delta, \widehat{W}') \implies \exists \widehat{W}.\, B(\mathtt{t}) = (n, \delta, \widehat{W}) \wedge \widehat{W} \overset{\delta}{=\!=} \widehat{W}'$$

Lemma 42 (Buffer execution simulation holds). For any $\widehat{\mathbb{W}}$, Δ_0, $\widehat{W}$, δ_0, δ_r, i, μ, E, $\widehat{W}_0$, B, n_0 and B', if $(\widehat{\mathbb{W}}, \Delta_0) \preccurlyeq^{i,\mu,E}_{(\widehat{W}_0,B),n_0} (\widehat{W}, \delta_0, \delta_r)$, $B' \simeq B$ and $\mathsf{dom}(B') = \mathsf{dom}(B)$, then $(\widehat{\mathbb{W}}, \Delta_0) \preccurlyeq^{i,\mu,E}_{(\widehat{W}_0,B'),n_0} (\widehat{W}, \delta_0, \delta_r)$.

Proof. Prove by cofix. □

Lemma 43 (Buffer increasing). For any $\widehat{W}_0$, B, $\mathtt{t}$, n_0, δ_0, $\widehat{W}$, δ and $\widehat{W}'$, if $\widehat{W}_0 \Downarrow B$, $B(\mathtt{t}) = (n_0, \delta_0, \widehat{W})$, $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{n} \widehat{W}'$ and $n > 0$, then $B\{\mathtt{t} \rightsquigarrow (n_0+n, \delta_0 \cup \delta, \widehat{W}')\} >_{\mathtt{t},\widehat{W}_0} B$.

Lemma 44 (Switch point match or not). For any $\widehat{\mathbb{W}}$, $\widehat{W}$, $\widehat{W}_0$, Δ_0, δ_0, i, μ, E, B, n_0, if $(\widehat{\mathbb{W}}, \Delta_0) \preccurlyeq^{i,\mu,E}_{(\widehat{W}_0,B),n_0} (\widehat{W}, \delta_0)$ and $Etr(\widehat{\mathbb{W}}, \epsilon)$ then $\mathsf{SPmatchEx}_{\mu,E}(\widehat{\mathbb{W}}, \widehat{W}_0) \vee \neg\mathsf{SPmatchEx}_{\mu,E}(\widehat{\mathbb{W}}, \widehat{W}_0)$.

Proof. Prove according to the Law of Excluded Middle. □

Lemma 45 (Whole Program Downward Simulation implies Diverge Preserving - II). For any $\widehat{\mathbb{W}}$, Δ_0, $\widehat{W}$, $\widehat{W}_0$, δ_0, δ_r, i, μ, E, B, and n_0, if $Etr(\widehat{\mathbb{W}}, \epsilon)$, $\widehat{W}_0 \Downarrow B$ and $(\widehat{\mathbb{W}}, \Delta_0) \preccurlyeq^{i,\mu,E}_{(\widehat{W}_0,B),n_0} (\widehat{W}, \delta_0, \delta_r)$, then $\exists \mathtt{t}.\, Etr(\widehat{W}_0^{\mathtt{t}}, \epsilon)$.

(** Event case proof **)

Lemma 46 (Event preservation). For any $\widehat{\mathbb{W}}$, Δ_0, $\widehat{\mathbb{W}}'$, e, $\widehat{W}$, δ_0, δ_r, $\widehat{W}_0$, i, E, μ and n_0, if $\widehat{\mathbb{W}} \overset{e}{\Rightarrow}{}^{+} \widehat{\mathbb{W}}'$, $(\widehat{\mathbb{W}}, \Delta_0) \preccurlyeq^{i,\mu,E}_{(\widehat{W}_0,B),n_0} (\widehat{W}, \delta_0, \delta_r)$, then $\exists \mathtt{t}, j, \widehat{W}_0', \widehat{W}', B', \delta'.\, \widehat{W}_0^{\mathtt{t}} \overset{e}{\Rightarrow}{}^{+} \widehat{W}_0' \wedge (\widehat{\mathbb{W}}', \mathsf{emp}) \preccurlyeq^{j,\mu,E}_{(\widehat{W}_0',B'),n'} (\widehat{W}', \mathsf{emp}, \delta') \wedge \widehat{W}_0' \Downarrow B'$.

Proof. This lemma can be proved by induction on the number of source steps. (* on paper proof *) □

Lemma 47 (Whole program downward sim implies event preservation aux)**.** For any $\widehat{\mathbb{W}}$, $\widehat{\mathbb{W}}'$, $\mathtt{t}$, $\widehat{W}_0$, $\widehat{W}_0'$, e, B, n_0, δ_r, if $\widehat{\mathbb{W}} \overset{e}{\Longrightarrow}^{+} \widehat{\mathbb{W}}'$, $\widehat{W}_0^{\mathtt{t}} \overset{e}{\Longrightarrow}^{+} \widehat{W}_0'$, $Etr(\widehat{\mathbb{W}}', B)$, $\widehat{W}_0' \Downarrow B$ and $(\widehat{\mathbb{W}}', \mathtt{emp}) \preccurlyeq^{i,\mu,E}_{(\widehat{W}_0', B), n_0} (\widehat{W}', \mathtt{emp}, \delta_r)$, then $Etr(\widehat{W}_0^{\mathtt{t}}, e :: B)$.

Proof. The lemma can be proved by cofix. □

Lemma 48 (Whole program downward sim implies event preservation)**.** For any $\widehat{\mathbb{W}}$, Δ_0, $\widehat{\mathbb{W}}'$, e, $\widehat{W}$, δ_0, δ_r, $\widehat{W}_0$, i, E, μ and n_0, if $\widehat{\mathbb{W}} \overset{e}{\Longrightarrow}^{+} \widehat{\mathbb{W}}'$, $(\widehat{\mathbb{W}}, \Delta_0) \preccurlyeq^{i,\mu,E}_{(\widehat{W}_0, B), n_0} (\widehat{W}, \delta_0, \delta_r)$ then $\exists \mathtt{t}.\, Etr(\widehat{W}_0^{\mathtt{t}}, e :: B)$.

Proof. Prove by applying Lemma 46 and 47. □

(** Whole progam downward simulation implies refinement **)

Theorem 49 (Whole-program Downward Simulation implies Refinement)**.**

$$\hat{\mathbb{P}} \preccurlyeq_E \hat{P} \implies \hat{\mathbb{P}} \sqsubseteq \hat{P}$$

where $\hat{\mathbb{P}} = \textbf{let}\ \Gamma\ \textbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$, $\hat{P} = \textbf{let}\ \Pi\ \textbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$, and $\Gamma = \{(sl_1, ge_1, \gamma_1), \ldots, (sl_m, ge_m, \gamma_m)\}$, $\Pi = \{(tl_1, ge_1', \gamma_1), \ldots, (tl_m, ge_m', \pi_m)\}$.

Proof. We first under $\hat{\mathbb{P}} \preccurlyeq_E \hat{P}$ according to its definition (Def. 30) and get:

$$\exists \varphi.\, (\forall 1 \le i \le i.\, \lfloor \varphi \rfloor ge_i = ge_i') \tag{4.8}$$

$$\begin{aligned} &\forall \widehat{\mathbb{W}}.\, (\hat{\mathbb{P}} :\overset{load}{\Longrightarrow} \widehat{\mathbb{W}}) \implies \\ &\quad \exists \widehat{W}, i \in \mathtt{index}, \mu.\, (\hat{P} :\overset{load}{\Longrightarrow} \widehat{W}) \wedge (\widehat{\mathbb{W}}, \mathtt{emp}) \preccurlyeq^{i,\mu,E}_{(\widehat{W}, \emptyset), 0} (\widehat{W}, \mathtt{emp}, \mathtt{emp}) \end{aligned} \tag{4.9}$$

And we need to prove: $PEtr(\hat{\mathbb{P}}, \mathcal{B}) \implies PEtr(\hat{P}, \mathcal{B})$.
We unfold $PEtr(\hat{\mathbb{P}}, \mathcal{B})$ according to its definition and get there exists $\widehat{\mathbb{W}}$, such that:

$$\hat{\mathbb{P}} :\overset{load}{\Longrightarrow} \widehat{\mathbb{W}} \tag{4.10}$$

$$Etr(\widehat{\mathbb{W}}, \mathcal{B}) \tag{4.11}$$

From (4.9) and (4.10), we get there exists $\widehat{W}$, $i \in \mathtt{index}$, and μ such that:

$$\hat{P} :\overset{load}{\Longrightarrow} \widehat{W} \tag{4.12}$$

$$(\widehat{\mathbb{W}}, \mathtt{emp}) \preccurlyeq^{i,\mu,E}_{(\widehat{W}, \emptyset), 0} (\widehat{W}, \mathtt{emp}, \mathtt{emp}) \tag{4.13}$$

We destruct $\mathcal{B}$, and discuss each case respectively in the following:

- if $\mathcal{B} = \textbf{done}$, we do inversion on (4.11) and get:

$$\widehat{\mathbb{W}} \Longrightarrow^{+} \textbf{done} \tag{4.14}$$

  According to (4.14) (4.13) and Lemma 31, we get:

$$\exists \mathtt{t}.\, Etr(\widehat{W}^{\mathtt{t}}, \textbf{done})$$

- if $\mathcal{B} = e :: \mathcal{B}'$, we do inversion on (4.11) and get:

$$\widehat{\mathbb{W}} \overset{e}{\Longrightarrow}^{+} \widehat{\mathbb{W}}' \wedge Etr(\widehat{\mathbb{W}}', \mathcal{B}) \tag{4.15}$$

  According to (4.15) (4.13) and Lemma 48, we get:

$$\exists \mathtt{t}.\, Etr(\widehat{W}^{\mathtt{t}}, e :: \mathcal{B}')$$

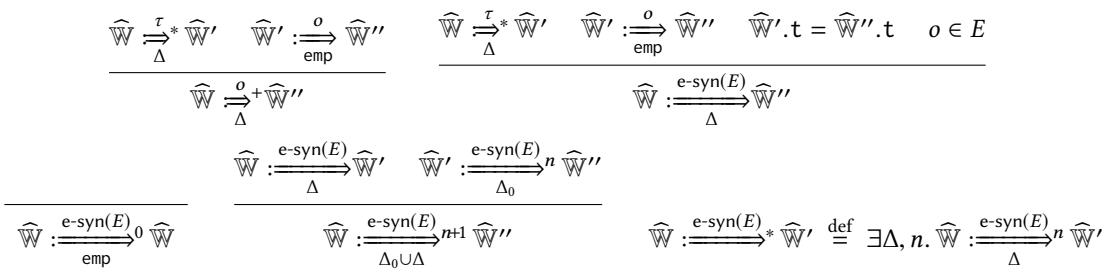


Fig. 15. Auxiliary Definition in Whole Program Upward Simulation

- if $\mathcal{B} = \epsilon$, we get:

$$Etr(\widehat{\mathbb{W}}, \epsilon) \tag{4.16}$$

According to (4.16), $\widehat{W} \Downarrow \emptyset$, (4.13) and Lemma 45, we get:

$$\exists \mathtt{t}.\ Etr(\widehat{W}^{\mathtt{t}}, \epsilon)$$

□

Whole-Program Upward Simulation. We define the whole-program upward simulation here.

Definition 50 (Whole-Program Upward Simulation)**.** Let $\hat{\mathbb{P}} = \textbf{let}\ \Gamma\ \textbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$, $\hat{P} = \textbf{let}\ \Pi\ \textbf{in}\ \mathtt{f}_1 \mid \ldots \mid \mathtt{f}_n$, $\Gamma = \{(sl_1, ge_1, \gamma_1), \ldots, (sl_m, ge_m, \gamma_m)\}$, $\Pi = \{(tl_1, ge'_1, \pi_1), \ldots, (tl_m, ge'_m, \pi_m)\}$. We say $\hat{P} \leqslant_E \hat{\mathbb{P}}$ iff.

- $\exists \varphi.\ (\forall 1 \le i \le m.\ \lfloor \varphi \rfloor ge_i = ge'_i)$, and
- for any $\widehat{W}$, if $\hat{P} :\xRightarrow{load} \widehat{W}$, then there exists $\widehat{\mathbb{W}}, i \in \mathtt{index}, \mu$, such that
  $\hat{\mathbb{P}} :\xRightarrow{load} \widehat{\mathbb{W}} \wedge (\widehat{W}, \mathtt{emp}) \leqslant_{\mu}^{i,E} (\widehat{\mathbb{W}}, \mathtt{emp})$.

Here we define $(\widehat{W}, \delta_0) \leqslant_{\mu}^{i,E} (\widehat{\mathbb{W}}, \Delta_0)$ as the largest relation such that whenever $(\widehat{W}, \delta_0) \leqslant_{\mu}^{i,E} (\widehat{\mathbb{W}}, \Delta_0)$ holds, then there exists $\mathtt{t}$, where $\widehat{W}.\mathtt{t} = \widehat{\mathbb{W}}.\mathtt{t} = \mathtt{t}$, such that the following are true:

(1) $\widehat{W}.\mathbb{d}\backslash\{\mathtt{t}\} = \widehat{\mathbb{W}}.\mathbb{d}\backslash\{\mathtt{t}\}$, $\widehat{W}.\mathbb{d}(\mathtt{t}) \le \widehat{\mathbb{W}}.\mathbb{d}(\mathtt{t})$;
(2) $\forall \widehat{W}', \delta$, if $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}} \widehat{W}'$, then one of the following holds:
  (a) $\exists j < i.\ (\widehat{W}', \delta_0 \cup \delta) \leqslant_{\mu}^{j,E} (\widehat{\mathbb{W}}, \Delta_0)$; or
  (b) $\exists \widehat{\mathbb{W}}', \Delta, j$ such that
    (i) $\widehat{\mathbb{W}} \overset{\tau}{\underset{\Delta}{\Rightarrow}}{}^{+} \widehat{\mathbb{W}}'$;
    (ii) $(\delta_0 \cup \delta) \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta, \delta_0 \cup \delta)$;
    (iii) $(\widehat{W}', \delta_0 \cup \delta) \leqslant_{\mu}^{j,E} (\widehat{\mathbb{W}}', \Delta_0 \cup \Delta)$; or
  (c) $\exists \widehat{\mathbb{W}}', \Delta, j$ such that (* high-level executes a eliminated synchronization *)
    (i) $\widehat{\mathbb{W}} :\xRightarrow[\Delta]{\text{e-syn}(E)} \widehat{\mathbb{W}}'$;
    (ii) $(\delta_0 \cup \delta) \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta, \delta_0 \cup \delta)$;
    (iii) $(\widehat{W}', \mathtt{emp}) \leqslant_{\mu}^{j,E} (\widehat{\mathbb{W}}', \mathtt{emp})$; or
  (d) $\exists \widehat{\mathbb{W}}_1, \widehat{\mathbb{W}}', \Delta_1, \Delta$ such that $\widehat{W}.\mathbb{d}(\mathtt{t}) = 0$ and (* high-level executes some eliminated synchronizations *)
    (i) $\widehat{\mathbb{W}} :\xRightarrow[\Delta_1]{\text{e-syn}(E)}{}^{+} \widehat{\mathbb{W}}_1$, $\widehat{\mathbb{W}}_1 \overset{\tau}{\underset{\Delta}{\Rightarrow}}{}^{*} \widehat{\mathbb{W}}'$;

(ii) $\mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta_1, \delta_0)$, and $\delta \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta, \delta)$;
(iii) $(\widehat{W}', \delta) \leqslant_{\mu}^{j,E} (\widehat{\mathbb{W}}', \Delta)$; or

(e) $\exists \widehat{\mathbb{W}}_1, \widehat{\mathbb{W}}', \Delta_1, \Delta$ such that $\widehat{W}.\mathbb{d}(\mathtt{t}) = 0$ and (* high-level executes some eliminated synchronizations *)
(i) $\widehat{\mathbb{W}} :\xRightarrow[\Delta_1]{\text{e-syn}(E)}{}^{+} \widehat{\mathbb{W}}_1$, $\widehat{\mathbb{W}}_1 :\xRightarrow[\Delta]{\text{e-syn}(E)} \widehat{\mathbb{W}}'$;
(ii) $\mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta_1, \delta_0)$, and $\delta \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta, \delta)$;
(iii) $(\widehat{W}', \mathsf{emp}) \leqslant_{\mu}^{j,E} (\widehat{\mathbb{W}}', \mathsf{emp})$;

(3) $\forall T', \mathbb{d}', \sigma', o, \mathtt{t}'$, if $o \neq \tau$, and $\widehat{W} :\xRightarrow[\mathsf{emp}]{o} (T', \mathtt{t}', \mathbb{d}', \sigma')$, then $\mathsf{closed}(\mu.S, \sigma')$, and one of the following holds:

(a) $\exists \widehat{\mathbb{W}}', \mathbb{T}', \Sigma', \Delta, j$ such that, for any $\mathtt{t}'' \in \mathsf{dom}(\mathbb{T}')$, we have
(i) $\widehat{\mathbb{W}} :\xRightarrow[\Delta]{\tau}{}^{*} \widehat{\mathbb{W}}'$, $\widehat{\mathbb{W}}' :\xRightarrow[\mathsf{emp}]{o} (\mathbb{T}', \mathtt{t}'', \mathbb{d}', \Sigma')$;
(ii) $\delta_0 \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta, \delta_0)$;
(iii) $((T', \mathtt{t}'', \mathbb{d}', \sigma'), \mathsf{emp}) \leqslant_{\mu}^{j,E} ((\mathbb{T}', \mathtt{t}'', \mathbb{d}', \Sigma'), \mathsf{emp})$; or

(b) $\exists \widehat{\mathbb{W}}_1, \widehat{\mathbb{W}}_2, \Delta_1, \Delta, \mathbb{T}', \Sigma', j$ such that, for any $\mathtt{t}'' \in \mathsf{dom}(\mathbb{T}')$, we have
(* high-level executes some eliminated synchronizations *)
(i) $\widehat{\mathbb{W}} :\xRightarrow[\Delta_1]{\text{e-syn}(E)}{}^{+} \widehat{\mathbb{W}}_1$, $\widehat{\mathbb{W}}_1 :\xRightarrow[\Delta]{\tau}{}^{*} \widehat{\mathbb{W}}_2$, $\widehat{\mathbb{W}}_2 :\xRightarrow[\mathsf{emp}]{o} (\mathbb{T}', \mathtt{t}'', \mathbb{d}', \Sigma')$;
(ii) $\delta_0 \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta_1, \delta_0)$;
(iii) $((T', \mathtt{t}'', \mathbb{d}', \sigma'), \mathsf{emp}) \leqslant_{\mu}^{j,E} ((\mathbb{T}', \mathtt{t}'', \mathbb{d}', \Sigma'), \mathsf{emp})$;

(4) if $\widehat{W} :\xRightarrow[\mathsf{emp}]{\tau}$ **done**, then one of the following are true:

(a) $\exists \widehat{\mathbb{W}}', \Delta$, such that:
(i) $\widehat{\mathbb{W}} :\xRightarrow[\Delta]{\tau}{}^{*} \widehat{\mathbb{W}}'$, $\widehat{\mathbb{W}}' :\xRightarrow[\mathsf{emp}]{\tau}$ **done**, and
(ii) $\delta_0 \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, and $\mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta, \delta_0)$; or

(b) $\exists \widehat{\mathbb{W}}_1, \widehat{\mathbb{W}}_2, \Delta_1, \Delta$, such that (* high-level executes some eliminated synchronizations *)
(i) $\widehat{\mathbb{W}} :\xRightarrow[\Delta_1]{\text{e-syn}(E)}{}^{+} \widehat{\mathbb{W}}_1$, $\widehat{\mathbb{W}}_1 :\xRightarrow[\Delta]{\tau}{}^{*} \widehat{\mathbb{W}}_2$, $\widehat{\mathbb{W}}_2 :\xRightarrow[\mathsf{emp}]{\tau}$ **done**;
(ii) $\delta_0 \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta_1, \delta_0)$.

REFERENCES

[1] Jonathan Aldrich, Emin Gün Sirer, Craig Chambers, and Susan J.Eggersa. 2003. Comprehensive synchronization elimination for Java. *Science of Computer Programming* 47 (May-June 2003), 91–120. Issue 2-3.
[2] Bogda Jeff and Hölzle Urs. 1999. Removing unnecessary synchronization in Java. In *OOPSLA*. 35–46.
[3] Hanru Jiang, Hongjin Liang, Siyang Xiao, Junpeng Zha, and Xinyu Feng. 2019. Towards Certified Separate Compilation for Concurrent Programs. In *PLDI*. 111–125.
[4] Youngju Song, Minki Cho, Dongjoo Kim, Yonghyun Kim, Jeehoon Kang, and Chung-Kil Hur. 2020. CompCertM: CompCert with C-Assembly Linking and Lightweight Modular Verification. In *POPL*.

$$\frac{P \overset{load}{\Longrightarrow} W \qquad W \Rightarrow^* W' \qquad W' \Longmapsto \mathsf{Race}}{P \Longmapsto \mathsf{Race}}$$

$$\frac{\mathtt{predict}(W, \mathtt{t}_1, (\delta_1, d_1)) \qquad \mathtt{predict}(W, \mathtt{t}_2, (\delta_2, d_2)) \qquad \mathtt{t}_1 \neq \mathtt{t}_2 \qquad (\delta_1, d_1) \frown (\delta_2, d_2)}{W \Longmapsto \mathsf{Race}}\ \mathsf{Race}$$

$$\frac{W = (T, \_, 0, \sigma) \quad T(\mathtt{t}) = (F, \kappa) \quad F \vdash (\kappa, \sigma) \overset{\tau}{\underset{\delta}{\mapsto}} (\kappa', \sigma')}{\mathtt{predict}(W, \mathtt{t}, (\delta, 0))}\ \text{Predict-0}$$

$$\frac{W = (T, \_, 0, \sigma) \qquad T(\mathtt{t}) = (F, \kappa) \qquad F \vdash (\kappa, \sigma) \overset{\mathsf{id.EntA}}{\underset{\mathtt{emp}}{\longmapsto}} (\kappa', \sigma) \quad F \vdash (\kappa', \sigma) \overset{\tau}{\underset{\delta}{\mapsto}}{}^* (\kappa'', \sigma'')}{\mathtt{predict}(W, \mathtt{t}, (\delta, \mathrm{id}))}\ \text{Predict-1}$$

Fig. 16. Data races in preemptive semantics

$$\frac{\hat{P} :\overset{load}{\Longrightarrow} \widehat{W} \qquad \widehat{W} :\Rightarrow^* \widehat{W}' \qquad \widehat{W}' :\Longmapsto \mathsf{Race}}{\hat{P} :\Longmapsto \mathsf{Race}}$$

$$\frac{\hat{P} :\overset{load}{\Longrightarrow} \widehat{W} \quad \mathtt{t}_1 \neq \mathtt{t}_2 \quad (\delta_1, d_1) \frown (\delta_2, d_2) \qquad \mathtt{NPpredict}(\widehat{W}, \mathtt{t}_1, (\delta_1, d_1)) \qquad \mathtt{NPpredict}(\widehat{W}, \mathtt{t}_2, (\delta_2, d_2))}{\hat{P} :\Longmapsto \mathsf{Race}}$$

$$\frac{\widehat{W} :\underset{\mathtt{emp}}{\overset{o}{\Longrightarrow}} \widehat{W}' \quad o \neq \tau \quad \mathtt{t}_1 \neq \mathtt{t}_2 \quad (\delta_1, d_1) \frown (\delta_2, d_2) \qquad \mathtt{NPpredict}(\widehat{W}', \mathtt{t}_1, (\delta_1, d_1)) \quad \mathtt{NPpredict}(\widehat{W}', \mathtt{t}_2, (\delta_2, d_2))}{\widehat{W} :\Longmapsto \mathsf{Race}}\ \mathsf{Race}_{\mathsf{np}}$$

$$\frac{\widehat{W} = (T, \_, \mathbb{d}, \sigma) \qquad T(\mathtt{t}) = (F, \kappa) \qquad F \vdash (\kappa, \sigma) \overset{\tau}{\underset{\delta}{\mapsto}}{}^* (\kappa', \sigma') \qquad \mathbb{d}(\mathtt{t}) = d}{\mathtt{NPpredict}(\widehat{W}, \mathtt{t}, (\delta, d))}\ \mathsf{Predict}_{\mathsf{np}}$$

Fig. 17. Data Race in Non-Preemptive Semantics

A DATA-RACE-FREEDOM

Below we first define the conflict of footprints.

$$\begin{array}{rcl} \delta_1 \frown \delta_2 & \text{iff} & (\delta_1.ws \cap \delta_2 \neq \emptyset) \vee (\delta_2.ws \cap \delta_1 \neq \emptyset) \\ (\delta_1, d_1) \frown (\delta_2, d_2) & \text{iff} & (\delta_1 \frown \delta_2) \wedge (d_1 = 0 \vee d_2 = 0) \end{array}$$

Then we define $\mathsf{DRF}(P)$ and $\mathsf{NPDRF}(\hat{P})$:

$$\begin{array}{lcl} \mathsf{DRF}(P) & \text{iff} & \neg(P \Longmapsto \mathsf{Race}) \\ \mathsf{NPDRF}(\hat{P}) & \text{iff} & \neg(\hat{P} :\Longmapsto \mathsf{Race}) \end{array}$$

$$\sigma \overset{rs}{=\!=} \sigma' \quad \text{iff} \quad \forall l \in rs.\ l \notin (\mathsf{dom}(\sigma) \cup \mathsf{dom}(\sigma')) \ \vee\ l \in (\mathsf{dom}(\sigma) \cap \mathsf{dom}(\sigma')) \wedge \sigma(l) = \sigma'(l)$$

$$\delta \subseteq \delta' \quad \text{iff} \quad (\delta.rs \subseteq \delta'.rs) \wedge (\delta.ws \subseteq \delta'.ws)$$

$$\delta \cup \delta' \quad \overset{\text{def}}{=} \quad (\delta.rs \cup \delta'.rs,\ \delta.ws \cup \delta'.ws)$$

$$\mathsf{forward}(\sigma, \sigma') \ \text{ iff } \ (\mathsf{dom}(\sigma) \subseteq \mathsf{dom}(\sigma'))$$

$$\mathtt{LEqPre}(\sigma_1, \sigma_2, \delta, F) \quad \text{iff} \quad \sigma_1 \overset{\delta.rs}{=\!=} \sigma_2 \wedge (\mathsf{dom}(\sigma_1) \cap \delta.ws) = (\mathsf{dom}(\sigma_2) \cap \delta.ws) \wedge (\mathsf{dom}(\sigma_1) \cap F) = (\mathsf{dom}(\sigma_2) \cap F)$$

$$\mathtt{LEqPost}(\sigma_1, \sigma_2, \delta, F) \quad \text{iff} \quad \sigma_1 \overset{\delta.ws}{=\!=} \sigma_2 \wedge (\mathsf{dom}(\sigma_1) \cap F) = (\mathsf{dom}(\sigma_2) \cap F)$$

$$\mathtt{LEffect}(\sigma_1, \sigma_2, \delta, F) \quad \text{iff} \quad \sigma_1 \overset{\mathsf{dom}(\sigma_1) - \delta.ws}{=\!=\!=} \sigma_2 \wedge (\mathsf{dom}(\sigma_2) - \mathsf{dom}(\sigma_1)) \subseteq (\delta.ws \cap F)$$

Fig. 18. Auxiliary definitions about states and footprints

B WELL-DEFINED LANGUAGE AND REACHCLOSED MODULE

Definition 51 (Well-Defined Languages)**.** $\mathsf{wd}(tl)$ iff , for any execution step $F \vdash (\kappa, \sigma) \overset{\iota}{\underset{\delta}{\mapsto}} (\kappa', \sigma')$ in this language, all of the following hold (some auxiliary definitions are in Fig. 18):

(1) $\mathsf{forward}(\sigma, \sigma')$;
(2) $\mathtt{LEffect}(\sigma, \sigma', \delta, F)$;
(3) For any σ_1, if $\mathtt{LEqPre}(\sigma, \sigma_1, \delta, F)$, then there exists σ_1' such that $F \vdash (\kappa, \sigma_1) \overset{\iota}{\underset{\delta}{\mapsto}} (\kappa', \sigma_1')$ and $\mathtt{LEqPost}(\sigma', \sigma_1', \delta, F)$.
(4) Let $\delta_0 = \bigcup\{\delta \mid \exists \kappa', \sigma'.\ F \vdash (\kappa, \sigma) \overset{\tau}{\underset{\delta}{\mapsto}} (\kappa', \sigma')\}$.
For any σ_1, if $\mathtt{LEqPre}(\sigma, \sigma_1, \delta_0, F)$, then for any $\kappa_1', \sigma_1', \iota_1, \delta_1$,
$F \vdash (\kappa, \sigma_1) \overset{\iota_1}{\underset{\delta_1}{\mapsto}} (\kappa_1', \sigma_1') \implies \exists \sigma'.\ F \vdash (\kappa, \sigma) \overset{\iota_1}{\underset{\delta_1}{\mapsto}} (\kappa_1', \sigma')$.

Definition 52 (Reach Closed Module)**.** $\mathsf{ReachClose}(sl, ge, \gamma)$ iff , for all $\mathtt{f}$, $\Bbbk$, Σ, $\mathbb{F}$ and $\mathbb{S}$, if $sl.\mathsf{InitCore}(\gamma, \mathtt{f}) = \Bbbk$, $\mathbb{S} = \mathsf{dom}(\Sigma)$, $ge \subseteq \Sigma$, $\mathbb{F} \cap \mathbb{S} = \emptyset$, and $\mathsf{closed}(\mathbb{S}, \Sigma)$, then $\mathsf{RC}(\mathbb{F}, \mathbb{S}, (\Bbbk, \Sigma))$.

Here RC is defined as the largest relation such that, whenever $\mathsf{RC}(\mathbb{F}, \mathbb{S}, (\Bbbk, \Sigma))$, then for all Σ' such that $\mathsf{R}(\Sigma, \Sigma', \mathbb{F}, \mathbb{S})$, and for all $\Bbbk'$, Σ', Σ'', ι and Δ such that $\mathbb{F} \vdash (\Bbbk, \Sigma') \overset{\iota}{\underset{\Delta}{\mapsto}} (\Bbbk', \Sigma'')$, we have $\mathsf{HG}(\Delta, \Sigma'', \mathbb{F}, \mathbb{S})$, and $\mathsf{RC}(\mathbb{F}, \mathbb{S}, (\Bbbk', \Sigma''))$.

$$B \simeq (T,\sigma) \overset{\text{def}}{=} \forall \mathtt{t} \in \mathsf{dom}(B), n, \delta, \widehat{W}.\, B(\mathtt{t}) = (n,\delta,\widehat{W}) \implies (\widehat{W}.T(\mathtt{t}) = T(\mathtt{t}) \wedge \widehat{W}.\mathtt{t} = \mathtt{t} \wedge \widehat{W}.\sigma \xrightarrow{\delta.rs \cup \delta.ws} \sigma)$$

$$\frac{}{\widehat{\mathbb{W}} \overset{(\mathsf{emp},\mathsf{emp})}{\models\!\Longrightarrow}_{\mu,E} \widehat{\mathbb{W}}} \qquad \frac{\widehat{\mathbb{W}}^{\mathtt{t}} \overset{(\Delta_0,\delta_0)}{\models\!\Longrightarrow}_{\mu,E} \widehat{\mathbb{W}}^{\mathtt{t}}_1 \quad \mathsf{FPmatch}(\mu,\Delta,\delta) \quad \widehat{\mathbb{W}}^{\mathtt{t}}_1 \overset{\tau}{\underset{\Delta}{:\Longrightarrow^*}} \widehat{\mathbb{W}}^{\mathtt{t}}_2 \quad \widehat{\mathbb{W}}^{\mathtt{t}}_2 :\xrightarrow[\mathsf{emp}]{\mathsf{id.ExtA}} \widehat{\mathbb{W}}^{\mathtt{t}}_3}{\widehat{\mathbb{W}}^{\mathtt{t}} \overset{(\Delta_0\cup\Delta,\delta_0\cup\delta)}{\models\!\Longrightarrow}_{\mu,E} \widehat{\mathbb{W}}^{\mathtt{t}}_3}$$

$$\frac{\widehat{\mathbb{W}}^{\mathtt{t}} \overset{(\Delta_0,\delta_0)}{\models\!\Longrightarrow}_{\mu,E} \widehat{\mathbb{W}}^{\mathtt{t}}_1 \quad \widehat{\mathbb{W}}^{\mathtt{t}}_1 \overset{\tau}{\underset{\Delta}{:\Longrightarrow^*}} \widehat{\mathbb{W}}^{\mathtt{t}}_2 \quad \widehat{\mathbb{W}}^{\mathtt{t}}_2 :\xrightarrow[\mathsf{emp}]{\mathsf{id.ExtA}} \widehat{\mathbb{W}}^{\mathtt{t}}_3 \quad \widehat{\mathbb{W}}_2.\mathbb{d}(\mathtt{t}) = \mathrm{id}_0 \quad (\mathrm{id}_0,\mathrm{id}) \in E \quad \mathsf{FPmatch}(\mu,\Delta,\delta)}{\widehat{\mathbb{W}}^{\mathtt{t}} \overset{(\Delta_0\cup\Delta,\delta_0\cup\delta)}{\models\!\Longrightarrow}_{\mu,E} \widehat{\mathbb{W}}^{\mathtt{t}}_3}$$

$$\frac{}{\widehat{\mathbb{W}} \Rrightarrow^{\mu,E}_{\emptyset} \widehat{\mathbb{W}}} \qquad \frac{\widehat{\mathbb{W}} \Rrightarrow^{\mu,E}_{B\setminus\{\mathtt{t}\}} \widehat{\mathbb{W}}_1 \quad B(\mathtt{t}).\delta = \delta \quad \widehat{\mathbb{W}}^{\mathtt{t}}_1 \overset{(\Delta,\delta)}{\models\!\Longrightarrow}_{\mu,E} \widehat{\mathbb{W}}'}{\widehat{\mathbb{W}} \Rrightarrow^{\mu,E}_{B} \widehat{\mathbb{W}}'}$$

$\mathsf{wdPresw}(\mu, n_0, \widehat{W}, (\widehat{W}_0, B), \delta)$ iff

(1) $\widehat{W}_0 \overset{\tau}{\underset{\delta}{:\Longrightarrow^{n_0}}} \widehat{W} \wedge \widehat{W}.\mathtt{t} \notin \mathsf{dom}(B) \wedge \delta \subseteq (\mathsf{curF}(\widehat{W}_0) \cup \mu.S)$;

(2) $\forall \mathtt{t}_1, \mathtt{t}_2 \in \mathsf{dom}(B), \mathtt{t}_1 \neq \mathtt{t}_2.\, (B(\mathtt{t}_1).\delta = \delta_1 \wedge B(\mathtt{t}_2).\delta = \delta_2) \implies \neg(\delta_1 \frown \delta_2) \wedge \neg(\delta_1 \frown \delta)$.

$$\mathbb{d}' \lesssim_{\mathcal{E}} \mathbb{d} \overset{\text{def}}{=} (\forall \mathtt{t} \in \mathsf{dom}(\mathbb{d}), \mathrm{id}.\, (\mathbb{d}(\mathtt{t}) = \mathbb{d}'(\mathtt{t}) = \mathrm{id}) \implies (\mathrm{id}, \_) \notin \mathcal{E}(\mathtt{t}))$$
$$\wedge (\forall \mathtt{t} \in \mathsf{dom}(\mathbb{d}), \mathrm{id}.\, (\mathbb{d}(\mathtt{t}) = \mathrm{id} \wedge \mathbb{d}'(\mathtt{t}) = 0) \implies (\mathrm{id}, \_) \in \mathcal{E}(\mathtt{t})) \wedge (\forall \mathtt{t} \in \mathsf{dom}(\mathbb{d}).\, \mathbb{d}'(\mathtt{t}) \leq \mathbb{d}(\mathtt{t}))$$

Fig. 19. Auxiliary Definitions for Proof of Compositionality - II

C PROOF OF COMPOSITIONALITY

We prove that our module local simulation can compose to the revised whole program downward simulation in this section.

Lemma 53 (Lemma for compositionality)**.**

(1) $\forall i \in \mathsf{dom}(\mathbb{T}), i \neq \mathtt{t}.\, \mathbb{T}(i) = (sl_i, \mathbb{F}_i, \Bbbk_i) \wedge T(i) = (tl_i, F_i, \kappa_i) \wedge \mathcal{E}(i) \subseteq E \wedge$
$(\forall \sigma', \Sigma'.\, \mathsf{Rely}(\mu, (\Sigma, \Sigma', \mathbb{F}_i), (\sigma, \sigma', F_i)) \implies$
$\exists \mathsf{idx}'_i.\, (\beta_i = \circ \implies \mathsf{idx}'_i = \mathsf{idx}_i) \wedge (\mathbb{F}_i, (\Bbbk_i, \Sigma'), \mathsf{emp}) \preccurlyeq^{\mathsf{idx}'_i, \mathcal{E}(i)}_{\mu} (F_i, (\kappa_i, \sigma'), \mathsf{emp}))$;

(2) $(\mathbb{F}, (\Bbbk, \Sigma), \Delta_0) \preccurlyeq^{\mathsf{idx}_\mathtt{t}, \mathcal{E}(\mathtt{t})}_{\mu} (F, (\kappa, \sigma), \delta_0) \wedge \mathbb{T}(\mathtt{t}) = (sl, \mathbb{F}, \Bbbk) \wedge T(\mathtt{t}) = (tl, F, \kappa) \wedge \mathcal{E}(\mathtt{t}) \subseteq E$;

(3) $(T_c, \sigma_c) \Rrightarrow^{\mu}_{B} (T, \sigma) \wedge \mathsf{wdPresw}(\mu, n_0, (T_c, \mathtt{t}, \mathbb{d}', \sigma_c), (\widehat{W}_0, B), \delta_r \cup \delta_0) \wedge \mathbb{d}' \lesssim_{\mathcal{E}} \mathbb{d} \wedge B \simeq (T, \sigma)$;

(4) $\widehat{\mathbb{W}} \Rrightarrow^{\mu,E}_{B} \widehat{\mathbb{W}}_0 \wedge \widehat{\mathbb{W}}^{\mathtt{t}}_0 \overset{(\Delta_r,\delta_r)}{\models\!\Longrightarrow}_{\mu,E} \widehat{\mathbb{W}}'_0 \wedge \widehat{\mathbb{W}}'_0 \overset{\tau}{\underset{\Delta_0}{:\Longrightarrow^*}} (\mathbb{T}, \mathtt{t}, \mathbb{d}, \Sigma)$;

(5) $\neg(\hat{\mathbb{P}} :\!\models\!\Longrightarrow \mathsf{Race}) \wedge \neg((\hat{\mathbb{P}}, E) :\!\models\!\Longrightarrow \mathsf{WA})$, and $(\hat{\mathbb{P}} \overset{load}{:\Longrightarrow} \widehat{\mathbb{W}}_p \wedge \widehat{\mathbb{W}}_p \Rrightarrow^* \widehat{\mathbb{W}}'_p \wedge \widehat{\mathbb{W}}'_p :\xrightarrow[\mathsf{emp}]{o} \widehat{\mathbb{W}})$;

(6) $\forall \widehat{\mathbb{W}}'.\, ((\mathbb{T}, \mathtt{t}, \mathbb{d}, \Sigma) \Rrightarrow^* \widehat{\mathbb{W}}') \implies \mathsf{RC\_Trans}(\widehat{\mathbb{W}}', \mu.\mathbb{S})$;

(7) for all i, j in $\mathsf{dom}(\mathbb{T})$, and $i \neq j$: $\mathbb{F}_i \cap \mathbb{F}_j = F_i \cap F_j = \emptyset$, and $\mathsf{wd}(sl_i)$, $\mathsf{wd}(tl_i)$;

then $((\mathbb{T}, \mathtt{t}, \mathbb{d}, \Sigma), \Delta_0) \preccurlyeq^{\mathsf{idx}, \mu, E}_{(\widehat{W}_0, B), n_0} ((T_c, \mathtt{t}, \mathbb{d}', \sigma_c), \delta_0, \delta_r)$. (where $\mathsf{idx} = ((\beta_1, \mathsf{idx}_1), \ldots, (\circ, \mathsf{idx}_t), \ldots, (\beta_n, \mathsf{idx}_n))$)

Proof. Prove by co-induction, and we need to prove the following:

- $\mathtt{t} \notin \mathsf{dom}(B), \mathsf{dom}(B) \subseteq \mathsf{dom}(\mathbb{T})$, and $\widehat{W}_0 \overset{\tau}{\underset{\delta_r \cup \delta_0}{:\Longrightarrow^{n_0}}} \widehat{W}$; (* The correctness proof of this case is staright-forward from assumption (3). *)

- if $(\mathbb{T}, \mathtt{t}, \mathbb{d}, \Sigma) \overset{\tau}{\underset{\Delta}{\Rrightarrow}} \widehat{\mathbb{W}}'$, from the high-level step, we get:

$$\widehat{\mathbb{W}}' = (\mathbb{T}', \mathtt{t}, \mathbb{d}, \Sigma') \tag{C.1}$$

$$F \vdash (\mathbb{k}, \Sigma) \overset{\tau}{\underset{\Delta}{\mapsto}} (\mathbb{k}', \Sigma') \tag{C.2}$$

$$\mathbb{T}(\mathtt{t}) = (sl, \mathbb{F}, \mathbb{k}), \mathbb{T}' = \mathbb{T}\{\mathtt{t} \rightsquigarrow (sl, F, \mathbb{k}')\} \tag{C.3}$$

From (C.1), assumption (2) module local simulation, and (7) reachclosed transition, we get that the target may either executes zero or multiply steps, and we discuss each case repectively:

– $\exists \mathsf{idx}'_{\mathtt{t}} < \mathsf{idx}_{\mathtt{t}}. (\mathbb{F}, (\mathbb{k}', \Sigma'), \Delta_0 \cup \Delta) \preccurlyeq_{\mu}^{\mathsf{idx}'_{\mathtt{t}}, \mathcal{E}(\mathtt{t})} (F, (\kappa, \sigma), \delta_0)$;
By applying co-inductive hypothesis, we can prove: $\exists \mathsf{idx}'. ((\beta_1, \mathsf{idx}_1), \ldots, (\circ, \mathsf{idx}'_{\mathtt{t}}), \ldots, (\beta_n, \mathsf{idx}_n))$

$$\mathsf{idx}' < \mathsf{idx} \wedge ((\mathbb{T}', \mathtt{t}, \mathbb{d}, \Sigma'), \Delta_0 \cup \Delta) \preccurlyeq^{\mathsf{idx}', \mu, E}_{(\widehat{W}_0, B), n_0} ((T_c, \mathtt{t}, \mathbb{d}', \sigma_c), \delta_0, \delta_r)$$

And we finish the prove of this case.

– $\exists \kappa', \sigma', \delta, \mathsf{idx}'_{\mathtt{t}}$, such that

$$F \vdash (\kappa, \sigma) \overset{\tau}{\underset{\delta}{\mapsto}}{}^{n} (\kappa', \sigma'), n > 0 \tag{C.4}$$

$$(\delta_0 \cup \delta) \subseteq (F \cup \mu.S), \mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta, \delta_0 \cup \delta) \tag{C.5}$$

$$(\mathbb{F}, (\mathbb{k}', \Sigma'), \Delta_0 \cup \Delta) \preccurlyeq_{\mu}^{\mathsf{idx}'_{\mathtt{t}}, \mathcal{E}(\mathtt{t})} (F, (\kappa', \sigma'), \delta_0 \cup \delta) \tag{C.6}$$

Here, we need to prove that the footprint δ has no conflict with the footprints recorded in buffer B, which means: $(\forall \mathtt{t}' \in \mathsf{dom}(B). B(\mathtt{t}').\delta = \delta \implies \neg(\delta' \frown \delta))$. We prove it by contridiction, we assume that there exists a thread $\mathtt{t}'$, where $B(\mathtt{t}').\delta = \delta'$ and $(\delta' \frown (\delta_0 \cup \delta))$. So, we get there exists $l \in \mu.S$, and $(l \in (\delta'.ws \cap (\delta_0 \cup \delta)) \vee l \in ((\delta_0 \cup \delta).ws \cup \delta'))$. According to Lemma 66, we know that such location l doesn't exists, so we get:

$$\forall \mathtt{t} \in \mathsf{dom}(B). B(\mathtt{t}).\delta = \delta' \implies \neg(\delta' \frown \delta) \tag{C.7}$$

From assumption (3), (C.4), (C.7) and Lemma 63, we get there exists σ'_c, such that: (* reorder $(T_c, \sigma_c) \Rrightarrow_B^{\mu} (T, \sigma) \wedge F \vdash (\kappa, \sigma) \overset{\tau}{\underset{\delta}{\mapsto}}{}^{n} (\kappa', \sigma')$ *)

$$F \vdash (\kappa, \sigma_c) \overset{\tau}{\underset{\delta}{\mapsto}}{}^{n} (\kappa', \sigma'_c) \wedge (T_c, \sigma'_c) \Rrightarrow_B^{\mu} (T, \sigma') \tag{C.8}$$

Because $\mathtt{t}$ doesn't in the domain of B, according to Lemma 67 we get: $(T_c\{\mathtt{t} \rightsquigarrow (tl, F, \kappa')\}, \sigma'_c) \Rrightarrow_B^{\mu} (T\{\mathtt{t} \rightsquigarrow (tl, F, \kappa')\}, \sigma')$; From assumption (3), (C.4), (C.7), Lemma 54 and 55, we get: $B \simeq (T\{\mathtt{t} \rightsquigarrow (tl, F, \kappa')\}, \sigma')$.
From (C.5) (C.8), let $T'_c = T_c\{\mathtt{t} \rightsquigarrow (tl, F, \kappa')\}$, and we can prove that:

* $(T_c, \mathtt{t}, \mathbb{d}', \sigma_c) \overset{\tau}{\underset{\delta}{\Rrightarrow}}{}^{n} (T'_c, \mathtt{t}, \mathbb{d}', \sigma'_c)$;
* $(\delta_0 \cup \delta) \subseteq (F \cup \mu.S), \mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta, \delta_0 \cup \delta)$;

Applying co-inductive hypothesis, we can prove that: $\exists \mathsf{idx}' = ((\beta_1, \mathsf{idx}_1), \ldots, (\circ, \mathsf{idx}'_{\mathtt{t}}), \ldots, (\beta_n, \mathsf{idx}_n))$

* $((\mathbb{T}', \mathtt{t}, \mathbb{d}, \Sigma'), \Delta_0 \cup \Delta) \preccurlyeq^{\mathsf{idx}', \mu, E}_{(\widehat{W}_0, B), n_0+n} ((T'_c, \mathtt{t}, \mathbb{d}', \sigma'_c), \delta_0 \cup \delta, \delta_r)$;

We finish the proof of this case.

- if $(\mathbb{T}, \mathtt{t}, \mathbb{d}, \Sigma) :\xRightarrow[\mathsf{emp}]{o} (\mathbb{T}', \mathtt{t}', \mathbb{d}_1, \Sigma) \wedge o \neq \tau$. Here, we consider the high-level program executes existing atomic block operation. The other cases are similiar to it. From the high-level step, where o = id.ExtA, we get:

$$\mathbb{T}(\mathtt{t}) = (sl, \mathbb{F}, \Bbbk), \mathbb{d}(\mathtt{t}) = \mathrm{id}_0 \tag{C.9}$$

$$\mathtt{t}' \in \mathsf{dom}(\mathbb{T}) \tag{C.10}$$

$$\mathbb{F} \vdash (\Bbbk, \Sigma) \xmapsto[\mathsf{emp}]{\mathrm{id.ExtA}} (\Bbbk', \Sigma) \tag{C.11}$$

$$\mathbb{T}' = \mathbb{T}\{\mathtt{t} \rightsquigarrow (sl, \mathbb{F}, \Bbbk')\} \tag{C.12}$$

$$\mathbb{d}_1 = \mathbb{d}\{\mathtt{t} \rightsquigarrow 0\} \tag{C.13}$$

From (C.11), assumption (2) module local simulation, and (7) reachclosed transition, we get that either the target program exits the atomic block, or the atomic block is eliminated in the target program. We discuss each case respectively:

– $\exists \kappa', \kappa'', \sigma', \delta$, such that (* switch points match *)

$$F \vdash (\kappa, \sigma) \xmapsto[\delta]{\tau}{}^{*} (\kappa', \sigma') \wedge F \vdash (\kappa', \sigma') \xmapsto[\mathsf{emp}]{\mathrm{id.ExtA}} (\kappa'', \sigma') \tag{C.14}$$

$$(\_, \mathrm{id}) \notin \mathcal{E}(\mathtt{t}) \tag{C.15}$$

$$\mathsf{LG}(\mu, (\delta_0 \cup \delta, \sigma', F), (\Delta_0, \Sigma')) \tag{C.16}$$

$$\begin{aligned} &\forall \sigma'', \Sigma'. \mathsf{Rely}(\mu, (\Sigma, \Sigma', \mathbb{F}), (\sigma', \sigma'', F)) \implies \\ &\qquad \exists \mathrm{idx}'_{\mathtt{t}}. (\mathbb{F}, (\Bbbk', \Sigma'), \mathsf{emp}) \preccurlyeq_{\mu}^{\mathrm{idx}'_{\mathtt{t}}, \mathcal{E}(\mathtt{t})} (F, (\kappa'', \sigma''), \mathsf{emp}) \end{aligned} \tag{C.17}$$

Here, we need to prove that the footprint δ has no conflict with the footprints recorded in buffer B, which means: $(\forall \mathtt{t}' \in \mathsf{dom}(B). B(\mathtt{t}').\delta = \delta' \implies \neg(\delta' \frown \delta))$. We prove it by contridiction, we assume there exists a thread $\mathtt{t}'$, where $B(\mathtt{t}').\delta = \delta'$ and $(\delta' \frown \delta)$. So, we get there exists $l \in \mu.S$, and $(l \in (\delta'.ws \cap \delta) \vee l \in (\delta.ws \cap \delta'))$. According to Lemma 66, we know that such location l doesn't exists, so we get:

$$\forall \mathtt{t} \in \mathsf{dom}(B). B(\mathtt{t}).\delta = \delta' \implies \neg(\delta' \frown \delta) \tag{C.18}$$

From (C.14) (C.18), assumption (3), Lemma 63, and the current thread $\mathtt{t}$ doesn't in the domain of reorder buffer B, so we get there exists σ'_c, such that:

(* reorder $(T_c, \sigma_c) \Rrightarrow_B^{\mu} (T, \sigma) \wedge F \vdash (\kappa, \sigma) \xmapsto[\delta]{\tau}{}^{*} (\kappa', \sigma') \wedge F \vdash (\kappa', \sigma') \xmapsto[\mathsf{emp}]{\mathrm{id.ExtA}} (\kappa'', \sigma')$ *)

$$F \vdash (\kappa, \sigma_c) \xmapsto[\delta]{\tau}{}^{*} (\kappa', \sigma'_c) \wedge F \vdash (\kappa', \sigma'_c) \xmapsto[\mathsf{emp}]{\mathrm{id.ExtA}} (\kappa'', \sigma'_c) \wedge (T_c, \sigma'_c) \Rrightarrow_B^{\mu} (T, \sigma') \tag{C.19}$$

We discuss why the source and target atomic bit are equal. Assuming the source and target atomic bit are not equal, we get $(\mathbb{d}(\mathtt{t}) = \mathrm{id}_0) \wedge \mathbb{d}'(\mathtt{t}) = 0$, then according to $\mathbb{d} \lesssim_{\mathcal{E}}^{\mathtt{t}} \mathbb{d}'$, we get $\exists \mathrm{id}'_0. (\mathrm{id}_0, \mathrm{id}'_0) \in \mathcal{E}(\mathtt{t}) \subseteq E$. From the correctness of analysis result, we get $\mathrm{id}'_0 = \mathrm{id}$, and $(\mathrm{id}_0, \mathrm{id}) \in \mathcal{E}(\mathtt{t})$. This is contridiction with $(\_, \mathrm{id}) \notin \mathcal{E}(\mathtt{t})$.

From assumption (3), (C.14), (C.18), Lemma 54, and 55, we get:

$$B \simeq (T\{\mathtt{t} \rightsquigarrow (tl, F, \kappa'')\}, \sigma') \tag{C.20}$$

According to (C.14) and (C.16), we can prove that for any $\mathtt{t}'' \in \mathsf{dom}(\mathbb{T}')$,

$*$ $(T_c, \mathtt{t}, \mathbb{d}', \sigma_c) \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{*} (T_c', \mathtt{t}, \mathbb{d}', \sigma_c') \wedge (T_c', \mathtt{t}, \mathbb{d}', \sigma_c') :\overset{o}{\underset{\mathsf{emp}}{\Longrightarrow}} (T_c'', \mathtt{t}'', \mathbb{d}_1', \sigma_c')$
$\wedge T_c' = T_c\{\mathtt{t} \rightsquigarrow (tl, F, \kappa')\} \wedge T_c'' = T_c'\{\mathtt{t} \rightsquigarrow (tl, F, \kappa'')\} \wedge \mathbb{d}_1' = \mathbb{d}'\{\mathtt{t} \rightsquigarrow 0\}$;
$*$ $(\delta_0 \cup \delta) \subseteq (F \cup \mu.S) \wedge \mathsf{FPmatch}(\mu, \Delta_0, \delta_0 \cup \delta)$;
Because $\mathtt{t}$ doesn't in the domain of reorder buffer, we get the following according to Lemma 67:

$$(T_c'', \sigma_c') \Rrightarrow_B^{\mu} (T', \sigma') \quad (* \text{ where } T_c'' = T_c\{\mathtt{t} \rightsquigarrow (tl, F, \kappa'')\},\ T' = T\{\mathtt{t} \rightsquigarrow (tl, F, \kappa'')\}\ *) \tag{C.21}$$

From (C.21) and (C.16), we prove that:
$*$ $(T_c'', \sigma_c') \Rrightarrow_B^{\mu} (T', \sigma') \wedge \mathsf{closed}(\mu.S, \sigma')$;
Then, we discuss whether new thread $\mathtt{t}''$ is in the reorder buffer B.
We need to first prove the following. From $\widehat{\mathbb{W}} \Rrightarrow_B^{\mu,\mathcal{E}} \widehat{\mathbb{W}}_0 \wedge \widehat{\mathbb{W}}_0^{\mathtt{t}} \overset{(\Delta_r,\delta_r)}{\Longmapsto}_{\mu,\mathcal{E}} \widehat{W}_0' \wedge \widehat{\mathbb{W}}_0' :\overset{\tau}{\underset{\Delta_0}{\Longrightarrow}}{}^{*} (\mathbb{T}, \mathtt{t}, \mathbb{d}, \Sigma) \wedge (\mathbb{T}, \mathtt{t}, \mathbb{d}, \Sigma) :\overset{\mathsf{id.ExtA}}{\underset{\mathsf{emp}}{\Longrightarrow}} (\mathbb{T}', \mathtt{t}'', \mathbb{d}_1, \Sigma)$, and Lemma 69 and 70, let $\widehat{\mathbb{W}}_0.\mathtt{t} = \mathtt{t}_0$, we get: $\exists \widehat{\mathbb{W}}_1, \widehat{\mathbb{W}}_2, \widehat{\mathbb{W}}_3$.

$$\widehat{\mathbb{W}}^{\mathtt{t}} \overset{(\Delta_r,\delta_r)}{\Longmapsto}_{\mu,E} \widehat{\mathbb{W}}_1 \wedge \widehat{\mathbb{W}}_1 :\overset{\tau}{\underset{\Delta_0}{\Longrightarrow}}{}^{*} \widehat{\mathbb{W}}_2 \wedge \widehat{\mathbb{W}}_2 :\overset{\mathsf{id.ExtA}}{\underset{\mathsf{emp}}{\Longrightarrow}} \widehat{\mathbb{W}}_3 \wedge \widehat{\mathbb{W}}_3 \Rrightarrow_B^{\mu,E} (\mathbb{T}', \mathtt{t}_0, \mathbb{d}_1, \Sigma) \tag{C.22}$$

$*$ if $B(\mathtt{t}'') = (n'', \delta'', \widehat{W}'')$.
From (C.21), assumption (3), and Lemma 64, we get there exists σ_c'', $\kappa_{\mathtt{t}''}$, $\kappa_{\mathtt{t}''}'$, such that:

$$T_c''(\mathtt{t}'') = (tl_{\mathtt{t}''}, F_{\mathtt{t}''}, \kappa_{\mathtt{t}''}) \wedge T'(\mathtt{t}'') = (tl_{\mathtt{t}''}, F_{\mathtt{t}''}, \kappa_{\mathtt{t}''}') \wedge$$
$$F_{\mathtt{t}''} \vdash (\kappa_{\mathtt{t}''}, \sigma_c') \overset{\tau}{\underset{\delta''}{\mapsto}}{}^{n''} (\kappa_{\mathtt{t}''}', \sigma_c'') \wedge (T_c''\{\mathtt{t}'' \rightsquigarrow (tl_{\mathtt{t}''}, F_{\mathtt{t}''}, \kappa_{\mathtt{t}''}')\}, \sigma_c'') \Rrightarrow_{B\setminus\{\mathtt{t}''\}}^{\mu} (T', \sigma') \tag{C.23}$$

From (C.20) and $B(\mathtt{t}'') = (n'', \delta'', \widehat{W}'')$, we get:

$$\widehat{W}''.T(\mathtt{t}'') = T'(\mathtt{t}'') \wedge \widehat{W}''.\mathtt{t} = \mathtt{t}'' \wedge \widehat{W}''.\sigma \overset{\delta.rs \cup \delta.ws}{=\!=\!=} \sigma' \tag{C.24}$$

From (C.23), (C.24), assumption (3) buffer no conflict, and Lemma 59, we get:

$$\widehat{W}''.\sigma \overset{\delta.rs \cup \delta.ws}{=\!=\!=} \sigma_c'' \tag{C.25}$$

From (C.23) (C.24) and (C.25), we get that $\exists \widehat{W}_1'' = (\{T_c''\{\mathtt{t}'' \rightsquigarrow (tl_{\mathtt{t}''}, F_{\mathtt{t}''}, \kappa_{\mathtt{t}''}'\}, \mathtt{t}'', \mathbb{d}_1', \sigma_c'')$, such that:

$$(T_c'', \mathtt{t}'', \mathbb{d}_1', \sigma_c') :\overset{\tau}{\underset{\delta''}{\Longrightarrow}}{}^{n''} \widehat{W}_1'' \wedge \widehat{W}'' \overset{\delta''}{=\!=} \widehat{W}_1''$$

Then, we do intro and get the following holds:

$$(T_c'', \mathtt{t}'', \mathbb{d}_1', \sigma_c') :\overset{\tau}{\underset{\delta''}{\Longrightarrow}}{}^{n''} \widehat{W}_2'' \wedge \widehat{W}'' \overset{\delta''}{=\!=} \widehat{W}_2'' \tag{C.26}$$

From Lemma 61, we get: $\widehat{W}_1'' = \widehat{W}_2''$. We can apply co-inductive hypothesis to prove that source and target program can establish simulation relation continuously.

According to (C.22), $B(\mathtt{t}'') = (n'', \delta'', \widehat{W}'')$ and Lemma 68, then $\exists \widehat{\mathbb{W}}_4, \Delta''$.

$$\widehat{\mathbb{W}}_3 \Rrightarrow_{B\setminus\{\mathtt{t}''\}}^{\mu,E} \widehat{\mathbb{W}}_4 \wedge \widehat{\mathbb{W}}_4^{\mathtt{t}''} \overset{(\Delta'',\delta'')}{\Longmapsto}_{\mu,E} (\mathbb{T}', \mathtt{t}'', \mathbb{d}_1, \Sigma)$$

By applying co-inductive hypothesis,
we get there exists $\widehat{W}_0' = (T_c'', \mathtt{t}'', \mathbb{d}_1', \sigma_c')$ and $\mathsf{idx}' = ((\beta_1, \mathsf{idx}_1), \ldots, (\bullet, \mathsf{idx}_\mathtt{t}), \ldots, (\circ, \mathsf{idx}'_{\mathtt{t}''}), \ldots, (\beta_n, \mathsf{idx}_n))$, such that:

$$((\mathbb{T}', \mathtt{t}'', \mathbb{d}_1, \Sigma), \mathsf{emp}) \preccurlyeq^{\mathsf{idx}', \mu, E}_{(\widehat{W}_0', B\setminus\{\mathtt{t}''\}), n''} (\widehat{W}_1'', \mathsf{emp}, \delta'')$$

∗ else, $\mathtt{t}''$ doesn't in the reorder buffer, so by applying co-inductive hypothesis, we know there exists $\widehat{W}_0' = (T_c'', \mathtt{t}'', \mathbb{d}_1', \sigma_c')$ and $\mathsf{idx}' = ((\beta_1, \mathsf{idx}_1), \ldots, (\bullet, \mathsf{idx}_\mathtt{t}), \ldots, (\circ, \mathsf{idx}'_{\mathtt{t}''}), \ldots, (\beta_n, \mathsf{idx}_n))$ such that:

$$((\mathbb{T}', \mathtt{t}'', \mathbb{d}_1, \Sigma), \mathsf{emp}) \preccurlyeq^{\mathsf{idx}', \mu, E}_{(\widehat{W}_0', B), 0} ((T_c'', \mathtt{t}'', \mathbb{d}_1', \sigma_c'), \mathsf{emp}, \mathsf{emp})$$

– $\exists \kappa', \sigma', \delta, n, \mathsf{idx}'_\mathtt{t}$, such that (* switch point not match *)

$$F \vdash (\kappa, \sigma) \overset{\tau}{\underset{\delta}{\mapsto}}{}^{n} (\kappa', \sigma') \tag{C.27}$$

$$(\_, \mathsf{id}) \in \mathcal{E}(\mathtt{t}) \tag{C.28}$$

$$n = 0 \implies \mathsf{idx}'_\mathtt{t} < \mathsf{idx}_\mathtt{t} \tag{C.29}$$

$$\mathsf{LG}(\mu, (\delta_0 \cup \delta, \sigma', F), (\Delta_0, \Sigma)) \tag{C.30}$$

$$\forall \sigma'', \Sigma'. \mathsf{Rely}(\mu, (\Sigma, \Sigma', \mathbb{F}), (\sigma', \sigma'', F)) \implies (\mathbb{F}, (\Bbbk', \Sigma'), \mathsf{emp}) \preccurlyeq^{\mathsf{idx}'_\mathtt{t}, \mathcal{E}(\mathtt{t})}_{\mu} (F, (\kappa', \sigma''), \mathsf{emp}) \tag{C.31}$$

Here, we need to prove that the footprint δ has no conflict with the footprints recorded in buffer B, which means: $(\forall \mathtt{t}' \in \mathsf{dom}(B).\, B(\mathtt{t}').\delta = \delta' \implies \neg(\delta \frown \delta'))$. We prove it by contridiction, we assume that there exists a thread $\mathtt{t}'$, where $B(\mathtt{t}').\delta = \delta'$ and $(\delta' \frown \delta)$. So, we get there exists $l \in \mu.S$, such that $(l \in (\delta'.ws \cap \delta) \vee l \in (\delta.ws \cap \delta'))$. According to Lemma 66, we know that such location l doesn't exist, so we get:

$$\forall \mathtt{t} \in \mathsf{dom}(B).\, B(\mathtt{t}).\delta = \delta' \implies \neg(\delta' \frown \delta) \tag{C.32}$$

From (C.27), (C.32), assumption (3) and Lemma 63, so we get there exists σ_c', such that: (* reorder $(T_c, \sigma_c) \Rrightarrow^{\mu}_{B} (T, \sigma) \wedge F \vdash (\kappa, \sigma) \overset{\tau}{\underset{\delta}{\mapsto}}{}^{n} (\kappa', \sigma')$ *)

$$F \vdash (\kappa, \sigma_c) \overset{\tau}{\underset{\delta}{\mapsto}}{}^{n} (\kappa', \sigma_c') \wedge (T_c, \sigma_c') \Rrightarrow^{\mu}_{B} (T, \sigma') \tag{C.33}$$

Because the current thread $\mathtt{t}$ doesn't in the domain of reorder buffer and according to Lemma 67, we get:

$$(T_c', \sigma_c') \Rrightarrow^{\mu}_{B} (T', \sigma') \quad (\text{let } T_c' = T_c\{\mathtt{t} \rightsquigarrow (tl, F, \kappa')\}, T' = T\{\mathtt{t} \rightsquigarrow (tl, F, \kappa')\}) \tag{C.34}$$

We discuss why the source and target atomic bit are not equal. Assuming the source and target atomic bit are equal, we get $(\mathbb{d}(\mathtt{t}) = \mathbb{d}'(\mathtt{t}) = \mathsf{id}_0)$, then according to $\mathbb{d} \lesssim_{\mathcal{E}} \mathbb{d}'$, we get $(\mathsf{id}_0, \_) \notin \mathcal{E}(\mathtt{t})$. We also know there exists id_0', such that $(\mathsf{id}_0', \mathsf{id}) \in \mathcal{E}(\mathtt{t}) \subseteq E$. According to the correctness of analysis result, we get $\mathsf{id}_0 = \mathsf{id}_0'$ and $(\mathsf{id}_0, \mathsf{id}) \in \mathcal{E}(\mathtt{t})$. This is contridicted with $(\mathsf{id}_0, \_) \notin \mathcal{E}(\mathtt{t})$.

We can prove that: $\mathbb{d}(\mathtt{t}) = \mathsf{id}_0 \wedge \mathbb{d}'(\mathtt{t}) = 0 \wedge (\mathsf{id}_0, \mathsf{id}) \in \mathcal{E}(\mathtt{t}) \subseteq E$.

According to (C.33), and (C.30), we can prove that
for any $\mathtt{t}'' \in \mathsf{dom}(\mathbb{T}')$, $\exists \mathsf{idx}' = ((\beta_1, \mathsf{idx}_1), \ldots, (\circ, \mathsf{idx}'_\mathtt{t}), \ldots, (\circ, \mathsf{idx}'_\mathtt{t}), \ldots, (\beta_n, \mathsf{idx}_n))$ such that
∗ $(T_c, \mathtt{t}, \mathbb{d}', \sigma_c') \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{n} (T_c\{\mathtt{t} \rightsquigarrow (tl, F, \kappa')\}, \mathtt{t}, \mathbb{d}', \sigma_c')$;
∗ $(n = 0 \implies \mathsf{idx}' < \mathsf{idx})$, $\mathsf{id}.\mathsf{ExtA} \in E$;

∗ $(\delta_0 \cup \delta) \subseteq (F \cup \mu.S)$, FPmatch$(\mu, \Delta_0, \delta_0 \cup \delta)$;

Let $B' = B\{\mathtt{t} \rightsquigarrow (n_0 + n, \delta_r \cup \delta_0 \cup \delta, (T_c\{\mathtt{t} \rightsquigarrow (tl, F, \kappa')\}, \mathtt{t}, \mathbb{d}', \sigma_c'))\}$. From (C.33) (C.30) (C.34) (C.32), assumption (3) and Lemma 65, we get:

$$(\widehat{W}_0.T, \widehat{W}_0.\sigma) \Rrightarrow^{\mu}_{B'} (T', \sigma') \tag{C.35}$$

We discuss whether $\mathtt{t}''$ is in reorder buffer.

Before discuss whether the $\mathtt{t}''$ is in the reorder buffer, we first prove the following:
From $\widehat{\mathbb{W}} \Rrightarrow^{\mu,E}_{B} \widehat{\mathbb{W}}_0$, $\widehat{\mathbb{W}}^{\mathtt{t}}_0 \overset{(\Delta_r,\delta_r)}{\Longrightarrow}_{\mu,E} \widehat{\mathbb{W}}'_0$, $\widehat{\mathbb{W}}'_0 :\underset{\Delta_0}{\overset{\tau}{\Longrightarrow}}{}^{*} (\mathbb{T}, \mathtt{t}, \mathbb{d}, \Sigma), (\mathbb{T}, \mathtt{t}, \mathbb{d}, \Sigma) :\underset{\mathsf{emp}}{\overset{\mathsf{id.ExtA}}{\Longrightarrow}} (\mathbb{T}', \mathtt{t}'', \mathbb{d}_1, \Sigma)$, $\mathbb{d}(\mathtt{t}) = \mathrm{id}_0$, FPmatch$(\mu, \Delta_0, \delta_0 \cup \delta)$, we get:

$$\widehat{\mathbb{W}} \Rrightarrow^{\mu,E}_{B} \widehat{\mathbb{W}}_0 \wedge \widehat{\mathbb{W}}^{\mathtt{t}}_0 \overset{(\Delta_r \cup \Delta_0, \delta_r \cup \delta_0 \cup \delta)}{\Longrightarrow}_{\mu,E} (\mathbb{T}', \mathtt{t}, \mathbb{d}_1, \Sigma)$$

We can merge them together and get: $\widehat{\mathbb{W}} \Rrightarrow^{\mu,E}_{B'} (\mathbb{T}', \mathtt{t}, \mathbb{d}_1, \Sigma)$

From (C.34), $\sigma_c' \overset{(\delta_r \cup \delta_0 \cup \delta).rs \cup (\delta_r \cup \delta_0 \cup \delta).ws}{\Longrightarrow} \sigma_c'$ and Lemma 60, we get: $\sigma_c' \overset{(\delta_r \cup \delta_0 \cup \delta).rs \cup (\delta_r \cup \delta_0 \cup \delta).ws}{\Longrightarrow} \sigma'$
According to $B \simeq (T, \sigma)$, (C.32), (C.27), $\mathtt{t} \notin \mathrm{dom}(B)$, and Lemma 54 and 55, we get: $B \simeq (T', \sigma')$
Finally, we can prove:

$$B' \simeq (T', \sigma') \tag{C.36}$$

∗ If $B'(\mathtt{t}'') = (n'', \delta'', \widehat{W}'')$, from (C.22) and Lemma 64, we get there exists σ_0, $\kappa_{\mathtt{t}''}$, $\kappa'_{\mathtt{t}''}$, such that:

$$\begin{aligned} &\widehat{W}_0.T(\mathtt{t}'') = (tl_{\mathtt{t}''}, F_{\mathtt{t}''}, \kappa_{\mathtt{t}''}) \wedge T'(\mathtt{t}'') = (tl_{\mathtt{t}''}, F_{\mathtt{t}''}, \kappa_{\mathtt{t}''}) \\ &\quad \wedge F_{\mathtt{t}''} \vdash (\kappa_{\mathtt{t}''}, \widehat{W}_0.\sigma) \underset{\delta''}{\overset{\tau}{\mapsto}}{}^{n''} (\kappa'_{\mathtt{t}''}, \sigma_0) \wedge (\widehat{W}_0.T\{\mathtt{t}'' \rightsquigarrow (tl_{\mathtt{t}''}, F_{\mathtt{t}''}, \kappa'_{\mathtt{t}''})\}, \sigma_0) \Rrightarrow^{\mu}_{B\setminus\{\mathtt{t}''\}} (T', \sigma') \end{aligned} \tag{C.37}$$

And we get:

$$\widehat{W}_0^{\mathtt{t}''} :\underset{\delta''}{\overset{\tau}{\Longrightarrow}}{}^{n''} (\widehat{W}_0.T\{\mathtt{t}'' \rightsquigarrow (tl_{\mathtt{t}''}, F_{\mathtt{t}''}, \kappa'_{\mathtt{t}''})\}, \mathtt{t}'', \widehat{W}_0.\mathbb{d}, \sigma_0) \tag{C.38}$$

$$(\widehat{W}_0.T\{\mathtt{t}'' \rightsquigarrow (tl_{\mathtt{t}''}, F_{\mathtt{t}''}, \kappa'_{\mathtt{t}''})\}, \mathtt{t}'', \widehat{W}_0.\mathbb{d}, \sigma_0) \Rrightarrow^{\mu}_{B'\setminus\{\mathtt{t}''\}} (T', \sigma') \tag{C.39}$$

According to (C.36), $B'(\mathtt{t}'') = (n'', \delta'', \widehat{W}'')$, we get:

$$\widehat{W}''.T(\mathtt{t}'') = T'(\mathtt{t}'') \wedge \widehat{W}''.\mathtt{t} = \mathtt{t}'' \wedge \widehat{W}''.\sigma \overset{\delta.rs \cup \delta.ws}{\Longrightarrow} \sigma' \tag{C.40}$$

From (C.40) (C.39) and Lemma 59, the following holds:

$$\widehat{W}''.\sigma \overset{\delta.rs \cup \delta.ws}{\Longrightarrow} \sigma_0 \tag{C.41}$$

Then, we can prove the following holds, according to (C.39) (C.40) (C.41) and Lemma 62:

$$(\widehat{W}_0.T\{\mathtt{t}'' \rightsquigarrow (tl_{\mathtt{t}''}, F_{\mathtt{t}''}, \kappa'_{\mathtt{t}''})\}, \mathtt{t}'', \widehat{W}_0.\mathbb{d}, \sigma_0) \overset{\delta}{=\!=} \widehat{W}'' \tag{C.42}$$

We do intro and get the following holds:

$$\widehat{W}_0^{\mathtt{t}''} :\underset{\delta''}{\overset{\tau}{\Longrightarrow}}{}^{n''} \widehat{W}_2'' \wedge \widehat{W}_2'' \overset{\delta}{=\!=} \widehat{W}'' \tag{C.43}$$

From (C.38) (C.42) (C.43) and Lemma 61, we get: $(\widehat{W}_0.T\{\mathtt{t}'' \rightsquigarrow (tl_{\mathtt{t}''}, F_{\mathtt{t}''}, \kappa'_{\mathtt{t}''})\}, \mathtt{t}'', \widehat{W}_0.\mathbb{d}, \sigma_0) = \widehat{W}_2''$, we can apply co-inductive hypothesis to prove that source and target program can establish simulation

relation continuously.

According to $\widehat{\mathbb{W}} \Rrightarrow_{B'}^{\mu,E} (\mathbb{T}', \mathtt{t}, \mathbb{d}_1, \Sigma)$ and Lemma 68, we get: $\exists \widehat{\mathbb{W}}_1, \Delta''.$

$$\widehat{\mathbb{W}} \Rrightarrow_{B'\backslash\{\mathtt{t}''\}}^{\mu,E} \widehat{\mathbb{W}}_1 \wedge \widehat{\mathbb{W}}_1^{\mathtt{t}''} \overset{(\Delta'',\delta'')}{\Longrightarrow}_{\mu,E} (\mathbb{T}', \mathtt{t}'', \mathbb{d}_1, \Sigma)$$

By applying co-inductive hypothesis, we get:

$$((\mathbb{T}', \mathtt{t}'', \mathbb{d}_1, \Sigma), \mathsf{emp}) \preccurlyeq^{\mathsf{idx}',\mu,E}_{(\widehat{W}_0^{\mathtt{t}''}, B'\backslash\{\mathtt{t}''\}), n''} ((\widehat{W}_0.T\{\mathtt{t}'' \rightsquigarrow (tl_{\mathtt{t}''}, F_{\mathtt{t}''}, \kappa'_{\mathtt{t}''})\}, \mathtt{t}'', \widehat{W}_0.\mathbb{d}, \sigma_0), \mathsf{emp}, \delta'')$$

* else, if $\mathtt{t}''$ doesn't in reorder buffer B', by applying co-inductive hypothesis, we get:

$$((\mathbb{T}', \mathtt{t}'', \mathbb{d}_1, \Sigma), \mathsf{emp}) \preccurlyeq^{\mathsf{idx}',\mu,E}_{(\widehat{W}_0^{\mathtt{t}''}, B'), 0} (\widehat{W}_0^{\mathtt{t}''}, \mathsf{emp}, \mathsf{emp})$$

We finish the proof of this case.

- The proof of case **done** is staright-forward.

□

Lemma 54 (No conflict steps buffer sim stable)**.** For any B, T, κ, σ, κ' and σ', if $B \simeq (T, \sigma)$, $F \vdash (\kappa, \sigma) \overset{\tau}{\underset{\delta}{\mapsto}}{}^{n} (\kappa', \sigma')$ and $(\forall \mathtt{t} \in \mathsf{dom}(B).\, B(\mathtt{t}).\delta = \delta' \implies \neg(\delta' \frown \delta))$, then $B \simeq (T, \sigma')$.

Proof. Prove by induction on the number of elements in buffer. □

Lemma 55 (Upd no buffer thread buffer sim stable)**.** For any B, T, σ and $\mathtt{t}$, if $B \simeq (T, \sigma)$ and $\mathtt{t} \notin \mathsf{dom}(B)$, then $B \simeq (T\{\mathtt{t} \rightsquigarrow (tl, F, \kappa)\}, \sigma)$.

Lemma 56 (Buffer eliminate buffer sim stable)**.** For any B, T, σ and $\mathtt{t}$, if $B \simeq (T, \sigma)$, then $B\backslash\{\mathtt{t}\} \simeq (T, \sigma)$.

Lemma 57 (Location no effect equal)**.** For any σ, σ', δ and δ', if $\sigma \overset{\mathsf{dom}(\sigma)-\delta.ws}{=\!=\!=} \sigma'$, $(\mathsf{dom}(\sigma') - \mathsf{dom}(\sigma)) \subseteq \delta.ws$, $\neg(\delta' \frown \delta)$, then $\sigma \overset{\delta'.rs \cup \delta'.ws}{=\!=\!=} \sigma'$.

Lemma 58 (Memory set equal transitiviy)**.** For any σ, σ', σ'' and δ, if $\sigma \overset{\delta.rs \cup \delta.ws}{=\!=\!=} \sigma'$ and $\sigma' \overset{\delta.rs \cup \delta.ws}{=\!=\!=} \sigma''$, then $\sigma \overset{\delta.rs \cup \delta.ws}{=\!=\!=} \sigma''$.

Lemma 59 (Location no effect apply buffer stable)**.** For any T_c, σ_c, T', σ', σ_0 and δ, if $(T_c, \sigma_c) \Rrightarrow_B^{\mu} (T', \sigma')$, $\sigma_0 \overset{\delta.rs \cup \delta.ws}{=\!=\!=} \sigma'$ and $(\forall \mathtt{t} \in \mathsf{dom}(B).\, B(\mathtt{t}).\delta = \delta' \implies \neg(\delta' \frown \delta))$, then $\sigma_0 \overset{\delta.rs \cup \delta.ws}{=\!=\!=} \sigma_c$.

Lemma 60 (Location no effect apply buffer stable - II)**.** For any T_c, σ_c, T', σ', σ_0 and δ, if $(T_c, \sigma_c) \Rrightarrow_B^{\mu} (T', \sigma')$, $\sigma_0 \overset{\delta.rs \cup \delta.ws}{=\!=\!=} \sigma_c$ and $(\forall \mathtt{t} \in \mathsf{dom}(B).\, B(\mathtt{t}).\delta = \delta' \implies \neg(\delta' \frown \delta))$, then $\sigma_0 \overset{\delta.rs \cup \delta.ws}{=\!=\!=} \sigma'$.

Lemma 61 (Same effect world equal)**.** For any $\widehat{W}_0$, $\widehat{W}$, $\widehat{W}_1$, $\widehat{W}_2$, δ and n, if $\widehat{W}_0 \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{n} \widehat{W}_1$, $\widehat{W}_1 \overset{\delta}{=\!=} \widehat{W}$, $\widehat{W}_0 \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{n} \widehat{W}_2$ and $\widehat{W}_2 \overset{\delta}{=\!=} \widehat{W}$, then $\widehat{W}_1 = \widehat{W}_2$.

The correctness proof of Lemma 54, 59, 60 and 61 depends on the languages are well defined.

Lemma 62 (Apply buffer not in domain thread state stable)**.** For any T_c, σ_c, T, σ, μ, B and $\mathtt{t}$, if $(T_c, \sigma_c) \Rrightarrow_B^{\mu} (T, \sigma)$, $\mathtt{t} \notin \mathsf{dom}(B)$, then $T_c(\mathtt{t}) = T(\mathtt{t})$.

Lemma 63 (Reordering Reorder Buffer and Steps No-conflict).
For any $T_c, T, \sigma_c, \kappa, \sigma, \kappa', \sigma', \delta, n, \mu$ and B, if $(T_c, \sigma_c) \Rrightarrow_B^{\mu} (T, \sigma), F \vdash (\kappa, \sigma) \overset{\tau}{\underset{\delta}{\mapsto}}{}^{n} (\kappa', \sigma')$, and $(\forall \mathtt{t} \in \mathsf{dom}(B).\, B(\mathtt{t}).\delta = \delta' \implies \neg(\delta \frown \delta'))$, then $\exists \sigma_c'.\, F \vdash (\kappa, \sigma_c) \overset{\tau}{\underset{\delta}{\mapsto}}{}^{n} (\kappa', \sigma_c') \wedge (T_c, \sigma_c') \Rrightarrow_B^{\mu} (T, \sigma')$.

Proof. We prove it by induction on the number of elements in buffer B. (* on paper proof *) □

Lemma 64 (Redo executions in reorder buffer). For $T_c, T, \sigma_c, \sigma, \delta, \mu$ and n, if $(T_c, \sigma_c) \Rrightarrow_B^{\mu} (T, \sigma)$, $B(\mathtt{t}) = (n, \delta)$, and $(\forall \mathtt{t}_1, \mathtt{t}_2 \in \mathsf{dom}(B), \mathtt{t}_1 \neq \mathtt{t}_2.\, (B(\mathtt{t}_1).\delta = \delta_1 \wedge B(\mathtt{t}_2).\delta = \delta_2) \implies \neg(\delta_1 \frown \delta_2))$, then:

$$\exists \sigma_c', \kappa, \kappa'.\, T_c(\mathtt{t}) = (tl, F, \kappa) \wedge T(\mathtt{t}) = (tl, F, \kappa') \\ \wedge\ F \vdash (\kappa, \sigma_c) \overset{\tau}{\underset{\delta}{\mapsto}}{}^{n} (\kappa', \sigma_c') \wedge (T_c\{\mathtt{t} \rightsquigarrow (tl, F, \kappa')\}, \sigma_c') \Rrightarrow_{B\setminus\{\mathtt{t}\}}^{\mu} (T, \sigma).$$

Proof. We prove it by induction on the number of elements in buffer B. □

Lemma 65 (Applybuffer merge). For any $F_0, \kappa_0, \sigma_0, \kappa_0', \delta_0, n_0, T_0, \mu, B, T', \sigma'$ and $\widehat{W}_0$,
if $T_0(\mathtt{t}_0) = (tl_0, F_0, \kappa_0)$, $F_0 \vdash (\kappa_0, \sigma_0) \overset{\tau}{\underset{\delta_0}{\mapsto}}{}^{n_0} (\kappa_0', \sigma)$, $\mathtt{t}_0 \notin \mathsf{dom}(B)$, $\delta_0 \subseteq (F_0 \cup \mu.S)$, $(T_0\{\mathtt{t}_0 \rightsquigarrow (tl_0, F_0, \kappa_0)\}, \sigma) \Rrightarrow_B^{\mu} (T', \sigma')$, then $(T_0, \sigma_0) \Rrightarrow^{\mu}_{B \cup \{\mathtt{t}_0 \rightsquigarrow (n_0, \delta_0, \widehat{W}_0)\}} (T', \sigma')$.

Proof. Prove by induction on the number of elements in buffer B. (* on paper proof *) □

Lemma 66 (Contridiction for fp-conflict in reorder buffer).
For any $\widehat{\mathbb{W}}, \widehat{\mathbb{W}}_0, \widehat{\mathbb{W}}', \mu, B, E, \Delta, \delta, \mathtt{t}, \mathtt{t}', \delta', \hat{\mathbb{P}}, \widehat{\mathbb{W}}_p, \widehat{\mathbb{W}}_p'$ and o, if

(1) $\widehat{\mathbb{W}} \Rrightarrow_B^{\mu,E} \widehat{\mathbb{W}}_0, \widehat{\mathbb{W}}_0^{\mathtt{t}} \overset{\tau}{\underset{\Delta}{\Rightarrow}}{}^{*} \widehat{\mathbb{W}}', \mathtt{t} \notin \mathsf{dom}(B)$;
(2) $\mathsf{FPmatch}(\mu, \Delta, \delta), B(\mathtt{t}').\delta = \delta', \mathtt{t}' \neq \mathtt{t}$;
(3) $\neg((\hat{\mathbb{P}}, E) :\Longmapsto \mathsf{WA}) \wedge \neg(\hat{\mathbb{P}} :\Longmapsto \mathsf{Race})$;
(4) $\hat{\mathbb{P}} :\overset{load}{\Longmapsto} \widehat{\mathbb{W}}_p \wedge \widehat{\mathbb{W}}_p \Rrightarrow^{*} \widehat{\mathbb{W}}_p' \wedge \widehat{\mathbb{W}}_p' :\underset{\mathsf{emp}}{\overset{o}{\Longmapsto}} \widehat{\mathbb{W}}$;

then $\neg(\exists l \in \mu.S.\, (l \in (\delta'.ws \cap \delta)) \vee (l \in (\delta.ws \cap \delta')))$.

Proof. We prove it by induction on the number of elements in buffer B. (* on paper proof *) □

Lemma 67 (Applying reorder buffer stable). For any $T, \sigma, T', \sigma', B, \mu, \mathtt{t}, tl, F$, and κ, if $(T, \sigma) \Rrightarrow_B^{\mu} (T', \sigma')$ and $\mathtt{t}' \notin \mathsf{dom}(B)$, then $(T\{\mathtt{t} \rightsquigarrow (tl, F, \kappa)\}, \sigma) \Rrightarrow_B^{\mu} (T'\{\mathtt{t} \rightsquigarrow (tl, F, \kappa)\}, \sigma')$.

Lemma 68 (Source apply buffer splitting). For any $\widehat{\mathbb{W}}, \widehat{\mathbb{W}}_1, B, \mathtt{t}, n, \delta, E, \mu, \hat{\mathbb{P}}, \widehat{\mathbb{W}}_p, \widehat{\mathbb{W}}_p'$ and o, if

(1) $\widehat{\mathbb{W}} \Rrightarrow_B^{\mu,E} \widehat{\mathbb{W}}_1, B(\mathtt{t}) = (n, \delta)$;
(2) $\neg((\hat{\mathbb{P}}, E) :\Longmapsto \mathsf{WA}) \wedge \neg(\hat{\mathbb{P}} :\Longmapsto \mathsf{Race})$;
(3) $\hat{\mathbb{P}} :\overset{load}{\Longmapsto} \widehat{\mathbb{W}}_p \wedge \widehat{\mathbb{W}}_p \Rrightarrow^{*} \widehat{\mathbb{W}}_p' \wedge \widehat{\mathbb{W}}_p' :\underset{\mathsf{emp}}{\overset{o}{\Longmapsto}} \widehat{\mathbb{W}}$;

then $\exists \widehat{\mathbb{W}}_0, \Delta.\, \widehat{\mathbb{W}} \Rrightarrow^{\mu,E}_{B\setminus\{\mathtt{t}\}} \widehat{\mathbb{W}}_0 \wedge \widehat{\mathbb{W}}_0^{\mathtt{t}} \overset{(\Delta,\delta)}{\Longmapsto}_{\mu,E} \widehat{\mathbb{W}}_1^{\mathtt{t}}$.

Proof. Prove by induction on the number of elements in B, and applying Lemma 72. (* on paper proof *) □

Lemma 69 (Reordering of reorder buffer and thread local step).
For any $\widehat{\mathbb{W}}_0, \widehat{\mathbb{W}}_1, \widehat{\mathbb{W}}_2, \widehat{\mathbb{W}}, \mathtt{t}_1, \mathtt{t}_2, \mu, B, \Delta, \delta, \hat{\mathbb{P}}, \widehat{\mathbb{W}}_p, \widehat{\mathbb{W}}_p', E$ and o, if

(1) $\widehat{\mathbb{W}}_0 \Rrightarrow_B^{\mu,E} \widehat{\mathbb{W}}_1^{\mathtt{t}_1}, \widehat{\mathbb{W}}_1^{\mathtt{t}_2} \overset{(\Delta,\delta)}{\Longmapsto}_{\mu,E} \widehat{\mathbb{W}}, \mathtt{t}_2 \notin \mathsf{dom}(B)$;

(2) $\neg((\hat{\mathbb{P}}, E) :\Longmapsto \mathsf{WA}) \wedge \neg(\hat{\mathbb{P}} :\Longmapsto \mathsf{Race})$;
(3) $\hat{\mathbb{P}} :\xRightarrow{load} \widehat{\mathbb{W}}_p \wedge \widehat{\mathbb{W}}_p \rightrightarrows^* \widehat{\mathbb{W}}'_p \wedge \widehat{\mathbb{W}}'_p :\xRightarrow[\mathsf{emp}]{o} \widehat{\mathbb{W}}_0$;

then $\exists \widehat{\mathbb{W}}_2, \mathtt{t}.\ \widehat{\mathbb{W}}_0^{\mathtt{t}_2} \models\!\!\Longrightarrow^{(\Delta,\delta)}_{\mu,E} \widehat{\mathbb{W}}_2 \wedge \widehat{\mathbb{W}}_2^{\mathtt{t}} \Rrightarrow^{\mu,E}_B \widehat{\mathbb{W}}^{\mathtt{t}_1}$.

Proof. Prove by induction on the number of elements in B, and applying Lemma 72. (* on paper proof *) □

Lemma 70 (Reordering of reorder buffer and tau step).
For any $\widehat{\mathbb{W}}_0$, $\widehat{\mathbb{W}}_1$, $\widehat{\mathbb{W}}_2$, $\widehat{\mathbb{W}}$, $\mathtt{t}_1$, $\mathtt{t}_2$, μ, B, Δ, $\hat{\mathbb{P}}$, $\widehat{\mathbb{W}}_p$, $\widehat{\mathbb{W}}'_p$, E and o, if

(1) $\widehat{\mathbb{W}}_0 \Rrightarrow^{\mu,E}_B \widehat{\mathbb{W}}_1^{\mathtt{t}_1}, \widehat{\mathbb{W}}_1^{\mathtt{t}_2} \xRightarrow[\Delta]{\tau}{}^* \widehat{\mathbb{W}}$, $\mathtt{t}_2 \notin \mathsf{dom}(B)$;
(2) $\neg((\hat{\mathbb{P}}, E) :\Longmapsto \mathsf{WA}) \wedge \neg(\hat{\mathbb{P}} :\Longmapsto \mathsf{Race})$;
(3) $\hat{\mathbb{P}} :\xRightarrow{load} \widehat{\mathbb{W}}_p \wedge \widehat{\mathbb{W}}_p \rightrightarrows^* \widehat{\mathbb{W}}'_p \wedge \widehat{\mathbb{W}}'_p :\xRightarrow[\mathsf{emp}]{o} \widehat{\mathbb{W}}_0$;

then $\exists \widehat{\mathbb{W}}_2, \mathtt{t}.\ \widehat{\mathbb{W}}_0^{\mathtt{t}_2} \xRightarrow[\Delta]{\tau}{}^* \widehat{\mathbb{W}}_2 \wedge \widehat{\mathbb{W}}_2^{\mathtt{t}} \Rrightarrow^{\mu,E}_B \widehat{\mathbb{W}}^{\mathtt{t}_1}$.

Proof. Prove by induction on the number of elements in B, and applying Lemma 73. (* on paper proof *) □

Lemma 71 (Contridiction for fp-conflict in thread local step and tau step).
For any $\widehat{\mathbb{W}}$, $\widehat{\mathbb{W}}_1$, $\widehat{\mathbb{W}}_2$, $\mathtt{t}$, $\mathtt{t}_1$, Δ, δ, Δ', δ', μ, E, $\hat{\mathbb{P}}$, $\widehat{\mathbb{W}}_p$, $\widehat{\mathbb{W}}'_p$ and o, if

(1) $\widehat{\mathbb{W}}^{\mathtt{t}} \models\!\!\Longrightarrow^{(\Delta,\delta)}_{\mu,E} \widehat{\mathbb{W}}_1, \widehat{\mathbb{W}}_1^{\mathtt{t}_1} :\xRightarrow[\Delta']{\tau}{}^* \widehat{\mathbb{W}}_2$, $\mathtt{t} \neq \mathtt{t}_1$;
(2) $\mathsf{FPmatch}(\mu, \Delta', \delta')$;
(3) $\neg((\hat{\mathbb{P}}, E) :\Longmapsto \mathsf{WA}) \wedge \neg(\hat{\mathbb{P}} :\Longmapsto \mathsf{Race})$;
(4) $\hat{\mathbb{P}} :\xRightarrow{load} \widehat{\mathbb{W}}_p \wedge \widehat{\mathbb{W}}_p \rightrightarrows^* \widehat{\mathbb{W}}'_p \wedge \widehat{\mathbb{W}}'_p :\xRightarrow[\mathsf{emp}]{o} \widehat{\mathbb{W}}$;

then $\neg(\exists l \in \mu.S.\ (l \in (\delta'.ws \cap \delta)) \vee (l \in (\delta.ws \cap \delta')))$.

Lemma 72 (Reordering of thread local step). For any $\widehat{\mathbb{W}}$, $\widehat{\mathbb{W}}_1$, $\widehat{\mathbb{W}}_2$, $\mathtt{t}$, $\mathtt{t}_1$, $\hat{\mathbb{P}}$, E, $\widehat{\mathbb{W}}_p$, $\widehat{\mathbb{W}}'_p$ and o, if

(1) $\widehat{\mathbb{W}}^{\mathtt{t}} \models\!\!\Longrightarrow^{(\Delta,\delta)}_{\mu,E} \widehat{\mathbb{W}}_1, \widehat{\mathbb{W}}_1^{\mathtt{t}_1} \models\!\!\Longrightarrow^{(\Delta_1,\delta_1)}_{\mu,E} \widehat{\mathbb{W}}_2$, $\mathtt{t} \neq \mathtt{t}_1$;
(2) $\neg((\hat{\mathbb{P}}, E) :\Longmapsto \mathsf{WA}) \wedge \neg(\hat{\mathbb{P}} :\Longmapsto \mathsf{Race})$;
(3) $\hat{\mathbb{P}} :\xRightarrow{load} \widehat{\mathbb{W}}_p \wedge \widehat{\mathbb{W}}_p \rightrightarrows^* \widehat{\mathbb{W}}'_p \wedge \widehat{\mathbb{W}}'_p :\xRightarrow[\mathsf{emp}]{o} \widehat{\mathbb{W}}$;

then $\exists \widehat{\mathbb{W}}_3.\ \widehat{\mathbb{W}}^{\mathtt{t}_1} \models\!\!\Longrightarrow^{(\Delta_1,\delta_1)}_{\mu,E} \widehat{\mathbb{W}}_3 \wedge \widehat{\mathbb{W}}_3^{\mathtt{t}} \models\!\!\Longrightarrow^{(\Delta,\delta)}_{\mu,E} \widehat{\mathbb{W}}_2^{\mathtt{t}}$.

Proof. Prove by induction on the steps of thread local steps, and applying Lemma 73 (* on paper proof *) □

Lemma 73 (Reordering of thread local step and tau step).
For any $\widehat{\mathbb{W}}$, $\widehat{\mathbb{W}}_1$, $\widehat{\mathbb{W}}_2$, $\mathtt{t}$, $\mathtt{t}_1$, Δ, Δ_1, δ, $\hat{\mathbb{P}}$, E, $\widehat{\mathbb{W}}_p$, $\widehat{\mathbb{W}}'_p$ and o, if

(1) $\widehat{\mathbb{W}}^{\mathtt{t}} \models\!\!\Longrightarrow^{(\Delta,\delta)}_{\mu,E} \widehat{\mathbb{W}}_1, \widehat{\mathbb{W}}_1^{\mathtt{t}_1} :\xRightarrow[\Delta_1]{\tau}{}^* \widehat{\mathbb{W}}_2$, $\mathtt{t} \neq \mathtt{t}_1$;
(2) $\neg((\hat{\mathbb{P}}, E) :\Longmapsto \mathsf{WA}) \wedge \neg(\hat{\mathbb{P}} :\Longmapsto \mathsf{Race})$;
(3) $\hat{\mathbb{P}} :\xRightarrow{load} \widehat{\mathbb{W}}_p \wedge \widehat{\mathbb{W}}_p \rightrightarrows^* \widehat{\mathbb{W}}'_p \wedge \widehat{\mathbb{W}}'_p :\xRightarrow[\mathsf{emp}]{o} \widehat{\mathbb{W}}$;

then $\exists \widehat{\mathbb{W}}_3.\ \widehat{\mathbb{W}}^{\mathtt{t}_1} :\xRightarrow[\Delta_1]{\tau}{}^* \widehat{\mathbb{W}}_3 \wedge \widehat{\mathbb{W}}_3^{\mathtt{t}} \models\!\!\Longrightarrow^{(\Delta,\delta)}_{\mu,E} \widehat{\mathbb{W}}_2^{\mathtt{t}}$.

We use some figures to illustrate the technologies of compositionality proof. We first need to know:

(1) The memory of target program used in *module local simulation* is the memory **after** applying reorder buffer;
(2) The memory of target program used in *whole program downward simulation* is the memory **before** applying reorder buffer.

We consider each case respectively.

- The source program executes τ-step:

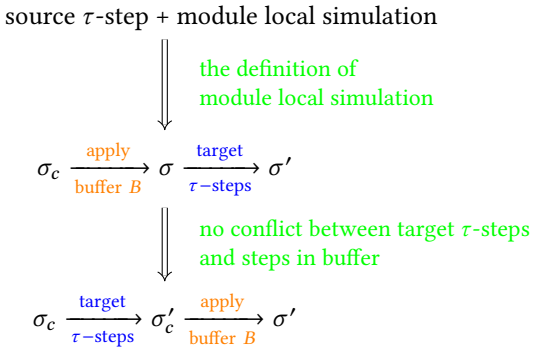


Fig. 20. Technology in Compositionality Proof —— τ-step

- The source program executes switch step:

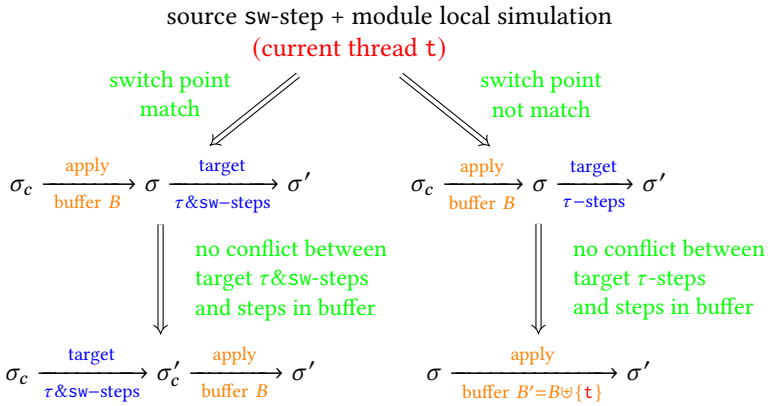


Fig. 21. Technology in Compositionality Proof —— sw-step

When the program (source and target) switches to the new thread $\mathtt{t}'$, we need to get the buffered step of $\mathtt{t}'$ from reorder buffer and do the following proof.

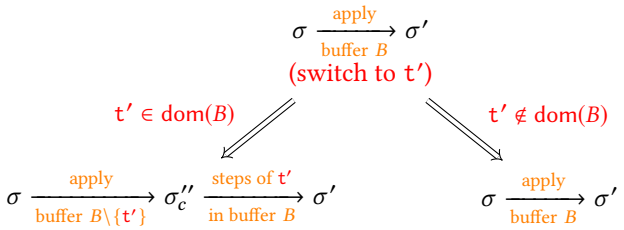


Fig. 22. Technology in Compositionality Proof —— redo buffered step

D PROOF OF FLIP

Definition 74 (Indexed Whole-Program Relation). $(\widehat{\mathbb{W}}, \Delta_0) \ltimes^{\mu,n,E}_{\widehat{W}_0} (\widehat{W}, \delta_0)$ holds, iff there exists $\mathtt{t}$, such that $\widehat{W}.\mathtt{t} = \widehat{\mathbb{W}}.\mathtt{t} = \mathtt{t}$, $\widehat{W}.\mathbb{d}\backslash\{\mathtt{t}\} = \widehat{\mathbb{W}}.\mathbb{d}\backslash\{\mathtt{t}\}$, $\widehat{W}.\mathbb{d}(\mathtt{t}) \leq \widehat{\mathbb{W}}.\mathbb{d}(\mathtt{t})$, and one of the following are true:

(1) either $\exists \widehat{W}', \widehat{\mathbb{W}}', \delta, \delta_r, \Delta, j, n_0$ such that
  (a) $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{n+1} \widehat{W}'$, $\widehat{\mathbb{W}} \overset{\tau}{\underset{\Delta}{\Rightarrow}}{}^{+} \widehat{\mathbb{W}}'$;
  (b) $(\delta_0 \cup \delta) \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta, \delta_0 \cup \delta)$;
  (c) $(\widehat{\mathbb{W}}', \Delta_0 \cup \Delta) \preccurlyeq^{j,\mu,E}_{(\widehat{W}_0, \emptyset), n_0} (\widehat{W}', \delta_0 \cup \delta, \delta_r)$;

(2) or $\exists \widehat{W}', \widehat{\mathbb{W}}', \delta, \delta_r, \Delta, j, n_0$ such that
  (a) $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{n+1} \widehat{W}'$, $\widehat{\mathbb{W}} :\overset{\text{e-syn}(E)}{\underset{\Delta}{\Longrightarrow}} \widehat{\mathbb{W}}'$;
  (b) $(\delta_0 \cup \delta) \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta, \delta_0 \cup \delta)$;
  (c) $(\widehat{\mathbb{W}}', \mathsf{emp}) \preccurlyeq^{j,\mu,E}_{(\widehat{W}_0, \emptyset), n_0} (\widehat{W}', \mathsf{emp}, \delta_r \cup \delta_0 \cup \delta)$;

(3) or $\exists \widehat{W}', \widehat{\mathbb{W}}_1, \widehat{\mathbb{W}}', \delta, \delta_r, \Delta_1, \Delta, j, n_0$ such that
  (a) $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{n+1} \widehat{W}'$, $\widehat{W}.\mathbb{d}(\mathtt{t}) = 0$, and $\widehat{\mathbb{W}} :\overset{\text{e-syn}(E)}{\underset{\Delta_1}{\Longrightarrow}}{}^{+} \widehat{\mathbb{W}}_1$, $\widehat{\mathbb{W}}_2 \overset{\tau}{\underset{\Delta}{\Rightarrow}}{}^{*} \widehat{\mathbb{W}}'$;
  (b) $\mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta_1, \delta_0)$, and $\delta \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta, \delta)$;
  (c) $(\widehat{\mathbb{W}}', \Delta) \preccurlyeq^{j,\mu,E}_{(\widehat{W}_0, \emptyset), n_0} (\widehat{W}', \delta, \delta_r \cup \delta_0)$;

(4) or $\exists \widehat{W}', \widehat{\mathbb{W}}_1, \widehat{\mathbb{W}}_2, \widehat{\mathbb{W}}', \delta, \delta_r, \Delta_1, \Delta, j, n_0$ such that
  (a) $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{n+1} \widehat{W}'$, $\widehat{W}.\mathbb{d}(\mathtt{t}) = 0$, and $\widehat{\mathbb{W}} :\overset{\text{e-syn}(E)}{\underset{\Delta_1}{\Longrightarrow}}{}^{+} \widehat{\mathbb{W}}_1$, $\widehat{\mathbb{W}}_2 :\overset{\text{e-syn}(E)}{\underset{\Delta}{\Longrightarrow}} \widehat{\mathbb{W}}'$;
  (b) $\mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta_1, \delta_0)$, and $\delta \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta, \delta)$;
  (c) $(\widehat{\mathbb{W}}', \mathsf{emp}) \preccurlyeq^{j,\mu,E}_{(\widehat{W}_0, \emptyset), n_0} (\widehat{W}', \mathsf{emp}, \delta_r \cup \delta_0 \cup \delta)$;

(5) or $\exists \widehat{W}', T', \mathbb{d}', \sigma', o \neq \tau, \widehat{\mathbb{W}}', \mathbb{T}', \Delta, j$, for any $\mathtt{t}' \in \mathrm{dom}(\mathbb{T}')$, such that
  (a) $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{n} \widehat{W}'$, $\widehat{W}' :\overset{o}{\underset{\mathsf{emp}}{\Longrightarrow}} (T', \mathtt{t}', \mathbb{d}', \sigma')$, $\mathsf{closed}(\mu.S, \sigma')$;
  (b) $\widehat{\mathbb{W}} \overset{\tau}{\underset{\Delta}{\Rightarrow}}{}^{*} \widehat{\mathbb{W}}'$, $\widehat{\mathbb{W}}' :\overset{o}{\underset{\mathsf{emp}}{\Longrightarrow}} (\mathbb{T}', \mathtt{t}', \mathbb{d}', \Sigma')$;
  (c) $((\mathbb{T}', \mathtt{t}', \mathbb{d}', \Sigma'), \mathsf{emp}) \preccurlyeq^{j,\mu,E}_{((T', \mathtt{t}', \mathbb{d}', \sigma'), \emptyset), 0} ((T', \mathtt{t}', \mathbb{d}', \sigma'), \mathsf{emp}, \mathsf{emp})$;

(6) or $\exists \widehat{W}', T', \mathbb{d}', \sigma', o \neq \tau, \widehat{\mathbb{W}}_1, \widehat{\mathbb{W}}_2, \mathbb{T}', \Delta_1, \Delta, j$, for any $\mathtt{t}' \in \mathrm{dom}(\mathbb{T}')$, such that
  (a) $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{n} \widehat{W}'$, $\widehat{W}' :\overset{o}{\underset{\mathsf{emp}}{\Longrightarrow}} (T', \mathtt{t}', \mathbb{d}', \sigma')$, $\mathsf{closed}(\mu.S, \sigma')$;

(b) $\widehat{\mathbb{W}} :\xrightarrow[\Delta_1]{\text{e-syn}(E)}{}^{+} \widehat{\mathbb{W}}_1$, $\widehat{\mathbb{W}}_1 \overset{\tau}{\underset{\Delta}{\Rightarrow}}{}^{*} \widehat{\mathbb{W}}_2$, $\widehat{\mathbb{W}}_2 :\overset{o}{\underset{\mathsf{emp}}{\Longrightarrow}} (\mathbb{T}', \mathtt{t}', \mathbb{d}', \Sigma')$;

(c) $\mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta_1, \delta_0)$, and $\delta_0 \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta, \delta)$;

(d) $((\mathbb{T}', \mathtt{t}', \mathbb{d}', \Sigma'), \mathsf{emp}) \preccurlyeq^{j,\mu,E}_{((T', \mathtt{t}', \mathbb{d}', \sigma'), \emptyset), 0} ((T', \mathtt{t}', \mathbb{d}', \sigma'), \mathsf{emp}, \mathsf{emp})$;

(7) or $\exists \widehat{W}', \delta, \widehat{\mathbb{W}}', \Delta$, such that:

(a) $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{n} \widehat{W}'$, $\widehat{W}' :\overset{\tau}{\underset{\mathsf{emp}}{\Longrightarrow}} \mathbf{done}$, $\widehat{\mathbb{W}} \overset{\tau}{\underset{\Delta}{\Rightarrow}}{}^{+} \widehat{\mathbb{W}}'$, $\widehat{\mathbb{W}}' :\overset{\tau}{\underset{\mathsf{emp}}{\Longrightarrow}} \mathbf{done}$;

(b) $(\delta \cup \delta_0) \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta, \delta_0 \cup \delta)$;

(8) or $\exists \widehat{W}', \widehat{\mathbb{W}}_1, \widehat{\mathbb{W}}_2, \delta, \Delta_1, \Delta$, such that:

(a) $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{n} \widehat{W}'$, $\widehat{W}' :\overset{\tau}{\underset{\mathsf{emp}}{\Longrightarrow}} \mathbf{done}$;

(b) $\widehat{\mathbb{W}} :\xrightarrow[\Delta_1]{\text{e-syn}(E)}{}^{+} \widehat{\mathbb{W}}_1$, $\widehat{\mathbb{W}}_1 \overset{\tau}{\underset{\Delta}{\Rightarrow}}{}^{*} \widehat{\mathbb{W}}_2$, $\widehat{\mathbb{W}}_2 \overset{\tau}{\underset{\Delta}{\Rightarrow}} \mathbf{done}$;

(c) $\delta \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta_1, \delta_0)$, $\mathsf{FPmatch}(\mu, \Delta, \delta)$.

Lemma 75 (Index Relation)**.** For any $\widehat{\mathbb{W}}, \widehat{W}, \Delta_0, \delta_0, \delta_r, i, n_0, \mu, E, \widehat{W}_0$ and $\mathtt{t}$, if

- $\mathsf{Safe}(\widehat{\mathbb{W}})$, $\widehat{W}.\mathtt{t} = \widehat{\mathbb{W}}.\mathtt{t} = \mathtt{t}$;
- $(\widehat{\mathbb{W}}, \Delta_0) \preccurlyeq^{i,\mu,E}_{(\widehat{W}_0, \emptyset), n_0} (\widehat{W}, \delta_0, \delta_r)$;
- $\widehat{W}.\mathbb{d}\backslash\{\mathtt{t}\} = \widehat{\mathbb{W}}.\mathbb{d}\backslash\{\mathtt{t}\}$, $\widehat{W}.\mathbb{d}(\mathtt{t}) \leq \widehat{\mathbb{W}}.\mathbb{d}(\mathtt{t})$;

then $\exists n. (\widehat{\mathbb{W}}, \Delta_0) \ltimes^{\mu,n,E}_{\widehat{W}_0} (\widehat{W}, \delta_0)$.

Proof. Prove by induction on the index i and applying Lemma 77. □

Lemma 76 (Lemma for Flip)**.** For any $\widehat{\mathbb{W}}, \widehat{W}, \Delta_0, \delta_0, \mu, E, \widehat{W}_0$ and n, if $\mathsf{Safe}(\widehat{\mathbb{W}})$ and all the target languages are deterministic, then

$$(\widehat{\mathbb{W}}, \Delta_0) \ltimes^{\mu,n,E}_{\widehat{W}_0} (\widehat{W}, \delta_0) \implies (\widehat{W}, \delta_0) \leqslant^{n,E}_{\mu} (\widehat{\mathbb{W}}, \Delta_0)$$

Proof. The proof of this lemma can be achieve by co-induction and applying Lemma 75 and 78. □

Lemma 77. If $\widehat{\mathbb{W}} \overset{\tau}{\underset{\Delta}{\Rightarrow}} \widehat{\mathbb{W}}'$ and $(\widehat{\mathbb{W}}', \Delta_0 \cup \Delta) \ltimes^{\mu,n,E}_{\widehat{W}_0} (\widehat{W}, \delta, \delta_r)$, then $(\widehat{\mathbb{W}}, \Delta_0) \ltimes^{\mu,n+1,E}_{\widehat{W}_0} (\widehat{W}, \delta_0)$.

Lemma 78. If $\widehat{\mathbb{W}} :\overset{o}{\underset{\mathsf{emp}}{\Longrightarrow}} \widehat{\mathbb{W}}'$, $\delta_0 \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$, $\mathsf{FPmatch}(\mu, \Delta_0, \delta_0)$ and $(\widehat{\mathbb{W}}', \mathsf{emp}) \ltimes^{\mu,n,E}_{\widehat{W}_0} (\widehat{W}, \mathsf{emp})$, then $(\widehat{\mathbb{W}}, \Delta_0) \ltimes^{\mu,n,E}_{\widehat{W}_0} (\widehat{W}, \delta_0)$.

The proof of Lemma 24 can be achieved by applying Lemma 75 and Lemma 76.

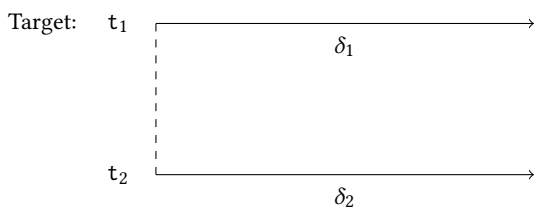


Fig. 23. Target conflicting footprint generating

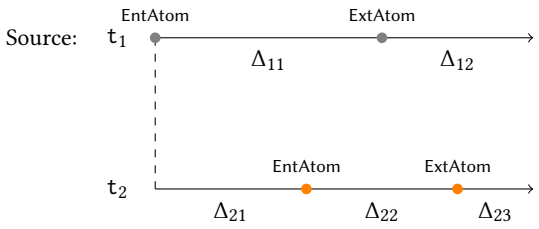


Fig. 24. Source conflicting footprint generating

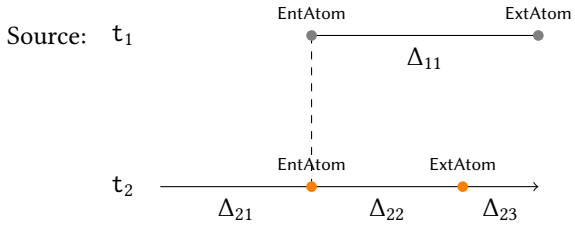
Fig. 25. Example of conflicting footprint generating

E PROOF OF DRF PRESERVING

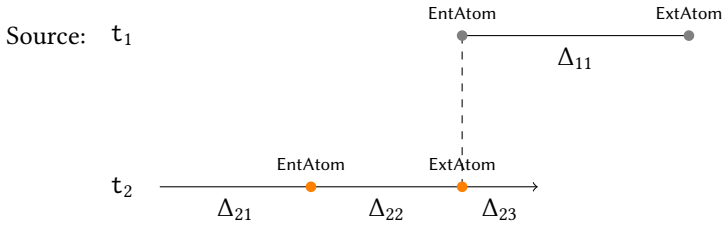
We prove the Lemma 26 in this section. The proof of DRF preserving in our work is more difficult than in CASCompCert. In CASCompCert, supposing there are two threads t_1 and t_2 generating δ_1 and δ_2 respectively in target program, we can know that there exists corresponding executions of t_1 and t_2, generating Δ_1 and Δ_2, in source program occuring footprint conflicting. However, the situation in our work is more complex, because of some eliminated atomic blocks during compilation in source program.

In Figure. 25, the execution of t_1 and t_2 in source program will execute some atomic blocks eliminated during compilation. Supposing there is a footprint conflicting in target program ($\delta_1 \frown \delta_2$). And we need to consider how to construct data race in source exeuction.

(1) Supposing $(\Delta_{11} \frown (\Delta_{21} \cup \Delta_{22} \cup \Delta_{23}))$, we discuss each case. If $\Delta_{11} \frown \Delta_{21}$, we construct a data race in source execution. Otherwise, we let the steps of t_2 generating Δ_{21} go first as shown below:

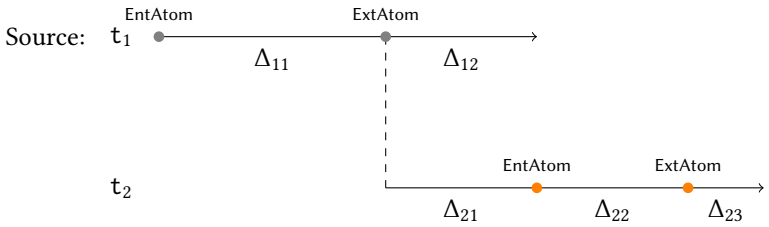


Then, we discuss whether $\Delta_{11} \frown \Delta_{22}$. We know that the execution of eliminated atomic block will only assess the thread local memory. So, Δ_{11} and Δ_{22} will never conflict. We let the steps of t_2 generating Δ_{21} go first as shown below:



We know that Δ_{11} must conflict with Δ_{23}, because $(\Delta_{11} \frown (\Delta_{21} \cup \Delta_{22} \cup \Delta_{23}))$, and $\neg(\Delta_{11} \frown \Delta_{21})$ and $\neg(\Delta_{11} \frown \Delta_{22})$. We construct a data race in source execution.

(2) If $\neg(\Delta_{11} \frown (\Delta_{21} \cup \Delta_{22} \cup \Delta_{23}))$, we let the steps of t_1 generating Δ_{11} go first as shown below:



We know $(\Delta_{12} \frown (\Delta_{12} \cup \Delta_{22} \cup \Delta_{23}))$, and discuss whether $(\Delta_{12} \frown \Delta_{21})$, or $(\Delta_{12} \frown \Delta_{22})$, or $(\Delta_{12} \frown \Delta_{23})$ in the similar way as before.

Proof. We first do intro and unfold NPDRF, we get the following assumptions:

$$\hat{P} \leqslant_E \hat{\mathbb{P}} \tag{E.1}$$

$$\mathsf{Safe}(\hat{\mathbb{P}}) \tag{E.2}$$

$$\mathsf{AC}(\hat{\mathbb{P}}, E) \tag{E.3}$$

$$\hat{P} :\Longmapsto \mathsf{Race} \tag{E.4}$$

And we need to prove the following goal:

$$\hat{\mathbb{P}} :\Longmapsto \mathsf{Race} \tag{E.5}$$

We do inversion on (E.4) and get two case. We just consider one of the case here, and the correctness proof of the other one is similar.

$$\hat{P} :\overset{load}{\Longrightarrow} \widehat{W} \tag{E.6}$$

$$\widehat{W} \Rightarrow^* \widehat{W}' \tag{E.7}$$

$$\widehat{W}' :\Longmapsto \mathsf{Race} \tag{E.8}$$

By applying Lemma 79 on (E.1), (E.6) and (E.7), we get there exists $i, \mu, \widehat{\mathbb{W}}, \widehat{\mathbb{W}}', \Delta_0, \delta_0$, such that:

$$\hat{\mathbb{P}} :\overset{load}{\Longrightarrow} \widehat{\mathbb{W}} \tag{E.9}$$

$$\widehat{\mathbb{W}} \Rightarrow^* \widehat{\mathbb{W}}' \tag{E.10}$$

$$(\widehat{W}', \delta_0) \leqslant^{i,E}_{\mu} (\widehat{\mathbb{W}}', \Delta_0) \tag{E.11}$$

Then, we do inversion on (E.8) and get:

$$\widehat{W}' :\underset{\mathsf{emp}}{\overset{o}{\Longrightarrow}} (T, \mathtt{t}, \mathbb{d}, \sigma), o \neq \tau \tag{E.12}$$

$$\mathtt{t}_1 \neq \mathtt{t}_2 \tag{E.13}$$

$$(\delta_1, \mathbb{d}(\mathtt{t}_1)) \frown (\delta_2, \mathbb{d}(\mathtt{t}_2)) \tag{E.14}$$

$$T(\mathtt{t}_1) = (tl_1, F_1, \kappa_1),\ F_1 \vdash (\kappa_1, \sigma) \underset{\delta_1}{\overset{\tau}{\mapsto}}{}^*(\kappa_1', \sigma_1') \tag{E.15}$$

$$T(\mathtt{t}_2) = (tl_2, F_2, \kappa_2),\ F_2 \vdash (\kappa_2, \sigma) \underset{\delta_2}{\overset{\tau}{\mapsto}}{}^*(\kappa_2', \sigma_2') \tag{E.16}$$

And we get the following hold:

$$(T, \mathtt{t}_1, \mathbb{d}, \sigma) :\overset{\tau}{\underset{\delta_1}{\Longrightarrow}}{}^{*} (T\{\mathtt{t}_1 \rightsquigarrow (tl_1, F_1, \kappa_1)\}, \mathtt{t}_1, \mathbb{d}, \sigma_1') \tag{E.17}$$

$$(T, \mathtt{t}_2, \mathbb{d}, \sigma) :\overset{\tau}{\underset{\delta_2}{\Longrightarrow}}{}^{*} (T\{\mathtt{t}_2 \rightsquigarrow (tl_2, F_2, \kappa_2)\}, \mathtt{t}_2, \mathbb{d}, \sigma_2') \tag{E.18}$$

From (E.12) (E.11), we get there exists $\widehat{\mathbb{W}}'', \mathbb{T}, \Sigma, \Delta, j$, for any $\mathtt{t}'' \in \mathsf{dom}(T)$, the following hold:

$$\widehat{\mathbb{W}}' \Longrightarrow^{*} \widehat{\mathbb{W}}'' \tag{E.19}$$

$$\widehat{\mathbb{W}}'' :\underset{\mathsf{emp}}{\overset{o}{\Longrightarrow}} (\mathbb{T}, \mathtt{t}'', \mathbb{d}, \Sigma) \tag{E.20}$$

$$((T, \mathtt{t}'', \mathbb{d}, \sigma), \mathsf{emp}) \leqslant_{\mu}^{j,E} ((\mathbb{T}, \mathtt{t}'', \mathbb{d}, \Sigma), \mathsf{emp}) \tag{E.21}$$

According to Lemma 80, (E.17), (E.18) and (E.2), we discuss each case:

- $\exists \widehat{W}_1', \delta_1', \widehat{\mathbb{W}}_1', \Delta_1'$, such that
  - $(T, \mathtt{t}_1, \mathbb{d}, \sigma) :\overset{\tau}{\underset{\delta_1'}{\Longrightarrow}}{}^{*} \widehat{W}_1', \delta_1 \subseteq \delta_1'$;
  - $(\mathbb{T}, \mathtt{t}_1, \mathbb{d}, \Sigma) :\overset{\tau}{\underset{\Delta_1'}{\Longrightarrow}}{}^{*} \widehat{\mathbb{W}}_1', \mathsf{FPmatch}(\mu, \Delta_1', \delta_1')$;

  and $\exists \widehat{W}_2', \delta_2', \widehat{\mathbb{W}}_2', \Delta_2'$, such that
  - $(T, \mathtt{t}_2, \mathbb{d}, \sigma) :\overset{\tau}{\underset{\delta_2'}{\Longrightarrow}}{}^{*} \widehat{W}_2', \delta_2 \subseteq \delta_2'$;
  - $(\mathbb{T}, \mathtt{t}_2, \mathbb{d}, \Sigma) :\overset{\tau}{\underset{\Delta_2'}{\Longrightarrow}}{}^{*} \widehat{\mathbb{W}}_2', \mathsf{FPmatch}(\mu, \Delta_2', \delta_2')$;

  According to (E.15), Lemma 86, we get there exists $l \in \mu.S$:
  $$(l \in \delta_1'.ws \wedge l \in \delta_2') \vee (l \in \delta_2'.ws \wedge l \in \delta_1')$$
  According to the definition of FPmatch, we get there exists $l' \in \mu.\mathbb{S}$:
  $$(l' \in \Delta_1'.ws \wedge l' \in \Delta_2') \vee (l' \in \Delta_2'.ws \wedge l' \in \Delta_1')$$
  And we construct a data race in source level.
- $\exists \widehat{W}_1', \delta_1', \widehat{\mathbb{W}}_1', \Delta_1'$, such that
  - $(T, \mathtt{t}_1, \mathbb{d}, \sigma) :\overset{\tau}{\underset{\delta_1'}{\Longrightarrow}}{}^{*} \widehat{W}_1', \delta_1 \subseteq \delta_1'$;
  - $(\mathbb{T}, \mathtt{t}_1, \mathbb{d}, \Sigma) :\overset{\tau}{\underset{\Delta_1'}{\Longrightarrow}}{}^{*} \widehat{\mathbb{W}}_1', \mathsf{FPmatch}(\mu, \Delta_1', \delta_1')$;

  and $\exists \widehat{W}_2, \delta_2', \widehat{\mathbb{W}}_2, \widehat{\mathbb{W}}_2', \Delta_2, \Delta_2'$, such that
  - $(T, \mathtt{t}_2, \mathbb{d}, \sigma) :\overset{\tau}{\underset{\delta_2'}{\Longrightarrow}}{}^{*} \widehat{W}_2, \delta_2 \subseteq \delta_2'$;
  - $(\mathbb{T}, \mathtt{t}_2, \mathbb{d}, \Sigma) :\overset{\mathsf{e\text{-}syn}(E)}{\underset{\Delta_2}{\Longrightarrow}}{}^{+} \widehat{\mathbb{W}}_2, \widehat{\mathbb{W}}_2 :\overset{\tau}{\underset{\Delta_2'}{\Longrightarrow}}{}^{*} \widehat{\mathbb{W}}_2', \mathsf{FPmatch}(\mu, \Delta_2 \cup \Delta_2', \delta_2)$;

  According to (E.15), Lemma 86, we get there exists $l \in \mu.S$:
  $$(l \in \delta_1'.ws \wedge l \in \delta_2') \vee (l \in \delta_2'.ws \wedge l \in \delta_1')$$
  According to the definition of FPmatch, we get there exists $l' \in \mu.\mathbb{S}$:
  $$(l' \in \Delta_1'.ws \wedge l' \in (\Delta_2 \cup \Delta_2')) \vee (l' \in (\Delta_2 \cup \Delta_2').ws \wedge l' \in \Delta_1')$$
  According to Lemma 83, we can prove that there is a data race in source level.

- The proof of this case is similiar with the previous one.
- $\exists \widehat{W}_1', \delta_1', \widehat{\mathbb{W}}_1', \widehat{\mathbb{W}}_1'', \Delta_1, \Delta_1'$, such that
  - $(T, \mathtt{t}_1, \mathbb{d}, \sigma) :\overset{\tau}{\underset{\delta_1}{\Longrightarrow}}{}^{*} \widehat{W}_1', \delta_1 \subseteq \delta_1'$;
  - $(\mathbb{T}, \mathtt{t}_1, \mathbb{d}, \Sigma) :\overset{\text{e-syn}(E)}{\underset{\Delta_1}{\Longrightarrow}}{}^{+} \widehat{\mathbb{W}}_1', \widehat{\mathbb{W}}_1' :\overset{\tau}{\underset{\Delta_1'}{\Longrightarrow}}{}^{*} \widehat{\mathbb{W}}_1'', \mathsf{FPmatch}(\mu, \Delta_1 \cup \Delta_1', \delta_1')$;

  and $\exists \widehat{W}_2', \delta_2', \widehat{\mathbb{W}}_2', \widehat{\mathbb{W}}_2'', \Delta_2, \Delta_2'$, such that
  - $(T, \mathtt{t}_2, \mathbb{d}, \sigma) :\overset{\tau}{\underset{\delta_2}{\Longrightarrow}}{}^{*} \widehat{W}_2', \delta_2 \subseteq \delta_2'$;
  - $(\mathbb{T}, \mathtt{t}_2, \mathbb{d}, \Sigma) :\overset{\text{e-syn}(E)}{\underset{\Delta_2}{\Longrightarrow}}{}^{+} \widehat{\mathbb{W}}_2', \widehat{\mathbb{W}}_2' :\overset{\tau}{\underset{\Delta_2'}{\Longrightarrow}}{}^{*} \widehat{\mathbb{W}}_2'', \mathsf{FPmatch}(\mu, \Delta_2 \cup \Delta_2', \delta_2')$;

  According to (E.15), Lemma 86, we get there exists $l \in \mu.S$:
  $$(l \in \delta_1'.ws \wedge l \in \delta_2') \vee (l \in \delta_2'.ws \wedge l \in \delta_1')$$
  According to the definition of FPmatch, we get there exists $l' \in \mu.\mathbb{S}$:
  $$(l' \in (\Delta_1 \cup \Delta_1').ws \wedge l' \in (\Delta_2 \cup \Delta_2')) \vee (l' \in (\Delta_2 \cup \Delta_2').ws \wedge l' \in (\Delta_1 \cup \Delta_1'))$$
  According to Lemma 84, we can prove that there is a data race in source level.

□

Lemma 79. For any $\hat{P}, \hat{\mathbb{P}}, \widehat{W}, \widehat{W}'$, and E, if $\hat{P} \leqslant_E \hat{\mathbb{P}}$, $\hat{P} :\overset{load}{\Longrightarrow} \widehat{W}$, $\widehat{W} \Rightarrow^{*} \widehat{W}'$, then there exists i, μ, $\widehat{\mathbb{W}}$, $\widehat{\mathbb{W}}'$, Δ_0, δ_0, such that: $\hat{\mathbb{P}} :\overset{load}{\Longrightarrow} \widehat{\mathbb{W}} \wedge \widehat{\mathbb{W}} \Rightarrow^{*} \widehat{\mathbb{W}}' \wedge (\widehat{W}', \delta_0) \leqslant_{\mu}^{i,E} (\widehat{\mathbb{W}}', \Delta_0)$.

Lemma 80. If $(\widehat{W}, \delta_0) \leqslant_{\mu}^{i,E} (\widehat{\mathbb{W}}, \Delta_0)$, $\mathsf{Safe}(\widehat{\mathbb{W}})$, $\widehat{W} :\overset{\tau}{\underset{\delta_1}{\Longrightarrow}}{}^{n} \widehat{W}_1$, then one of the following holds:

(1) either, $\exists \widehat{W}', \delta, \widehat{\mathbb{W}}', \Delta$, such that
  - $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{*} \widehat{W}', \delta_1 \subseteq \delta$;
  - $\widehat{\mathbb{W}} \overset{\tau}{\underset{\Delta}{\Rightarrow}}{}^{*} \widehat{\mathbb{W}}', \mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta, \delta_0 \cup \delta)$;

(2) or, $\exists \widehat{W}', \delta, \widehat{\mathbb{W}}', \widehat{\mathbb{W}}'', \Delta, \Delta'$, such that
  - $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{*} \widehat{W}', \delta_1 \subseteq \delta, \widehat{W}.\mathbb{d}(\widehat{W}.\mathtt{t}) = 0$;
  - $\widehat{\mathbb{W}} :\overset{\text{e-syn}(E)}{\underset{\Delta}{\Longrightarrow}}{}^{+} \widehat{\mathbb{W}}', \widehat{\mathbb{W}}' :\overset{\tau}{\underset{\Delta'}{\Longrightarrow}}{}^{*} \widehat{\mathbb{W}}'', \mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta \cup \Delta', \delta_0 \cup \delta)$;

Proof. We prove this lemma by induction on step n. If $n = 0$, we finish the proof by applying Lemma 81. If $n > 0$, we finish the proof by applying inductive hypothesis. □

Lemma 81. If $(\widehat{W}, \delta_0) \leqslant_{\mu}^{i,E} (\widehat{\mathbb{W}}, \Delta_0)$, $\mathsf{Safe}(\widehat{\mathbb{W}})$, then one of the following holds:

(1) either, $\exists \widehat{W}', \delta, \widehat{\mathbb{W}}', \Delta$, such that
  - $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{*} \widehat{W}'$;
  - $\widehat{\mathbb{W}} \overset{\tau}{\underset{\Delta}{\Rightarrow}}{}^{*} \widehat{\mathbb{W}}', \mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta, \delta_0 \cup \delta)$;

(2) or, $\exists \widehat{W}', \delta, \widehat{\mathbb{W}}', \widehat{\mathbb{W}}'', \Delta, \Delta'$, such that
  - $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{*} \widehat{W}', \widehat{W}.\mathbb{d}(\widehat{W}.\mathtt{t}) = 0$;
  - $\widehat{\mathbb{W}} :\overset{\text{e-syn}(E)}{\underset{\Delta}{\Longrightarrow}}{}^{+} \widehat{\mathbb{W}}', \widehat{\mathbb{W}}' :\overset{\tau}{\underset{\Delta'}{\Longrightarrow}}{}^{*} \widehat{\mathbb{W}}'', \mathsf{FPmatch}(\mu, \Delta_0 \cup \Delta \cup \Delta', \delta_0 \cup \delta)$;

Proof. We prove this Lemma by induction on index i. (* on paper proof *) □

Lemma 82 (Constructing source conflict aux)**.**
For any $\widehat{\mathbb{W}}_0, o, \mathbb{T}, \mathtt{t}, \mathbb{d}, \Sigma, \mathtt{t}_1, \widehat{\mathbb{W}}_1, \Delta_1, \mathtt{t}_2, \widehat{\mathbb{W}}_2, \Delta_2, n > 0$ and E, if

- $\widehat{\mathbb{W}}_0 :\xRightarrow[\mathsf{emp}]{o} (\mathbb{T}, \mathtt{t}, \mathbb{d}, \Sigma)$, $o \neq \tau$, $\mathtt{t}_1 \neq \mathtt{t}_2$;
- $(\mathbb{T}, \mathtt{t}_1, \mathbb{d}, \Sigma) :\xRightarrow[\Delta_1]{\tau}{}^{*} \widehat{\mathbb{W}}_1$;
- $(\mathbb{T}, \mathtt{t}_2, \mathbb{d}, \Sigma) :\xRightarrow[\Delta_2]{\text{e-syn}(E)} \widehat{\mathbb{W}}_2$;
- $\Delta_1 \frown \Delta_2$;
- $\neg(\exists \widehat{\mathbb{W}}.\ \widehat{\mathbb{W}}_0 \rightrightarrows^{*} \widehat{\mathbb{W}} \wedge (\widehat{\mathbb{W}}, E) :\Longrightarrow \mathsf{WA})$;

then $\widehat{\mathbb{W}}_0 :\Longrightarrow \mathsf{Race}$.

Proof. (* on paper proof *) □

Lemma 83 (Constructing source conflict - I)**.**
For any $\widehat{\mathbb{W}}_0, o, \mathbb{T}, \mathtt{t}, \mathbb{d}, \Sigma, \mathtt{t}_1, \widehat{\mathbb{W}}_1, \Delta_1, \mathtt{t}_2, \widehat{\mathbb{W}}_2, \widehat{\mathbb{W}}'_2, \Delta_2, \Delta'_2, n > 0$ and E, if

- $\widehat{\mathbb{W}}_0 :\xRightarrow[\mathsf{emp}]{o} (\mathbb{T}, \mathtt{t}, \mathbb{d}, \Sigma)$, $o \neq \tau$, $\mathtt{t}_1 \neq \mathtt{t}_2$;
- $(\mathbb{T}, \mathtt{t}_1, \mathbb{d}, \Sigma) :\xRightarrow[\Delta_1]{\tau}{}^{*} \widehat{\mathbb{W}}_1$;
- $(\mathbb{T}, \mathtt{t}_2, \mathbb{d}, \Sigma) :\xRightarrow[\Delta_2]{\text{e-syn}(E)}{}^{n} \widehat{\mathbb{W}}_2$, $\widehat{\mathbb{W}}_2 :\xRightarrow[\Delta'_2]{\tau}{}^{*} \widehat{\mathbb{W}}'_2$;
- $\Delta_1 \frown (\Delta_2 \cup \Delta'_2)$;
- $\neg(\exists \widehat{\mathbb{W}}.\ \widehat{\mathbb{W}}_0 \rightrightarrows^{*} \widehat{\mathbb{W}} \wedge (\widehat{\mathbb{W}}, E) :\Longrightarrow \mathsf{WA})$;

then $\exists \widehat{\mathbb{W}}.\ \widehat{\mathbb{W}}_0 \rightrightarrows^{*} \widehat{\mathbb{W}} \wedge \widehat{\mathbb{W}} :\Longrightarrow \mathsf{Race}$.

Proof. We prove it by induction on n and apply Lemma 82. (* on paper proof *) □

Lemma 84 (Constructing source conflict - II)**.**
For any $\widehat{\mathbb{W}}_0, \mathbb{T}, \mathtt{t}, \mathbb{d}, \Sigma, \mathtt{t}_1, \widehat{\mathbb{W}}_1, \Delta_1, \mathtt{t}_2, \widehat{\mathbb{W}}_2, \widehat{\mathbb{W}}'_2, \Delta_2, \Delta'_2, n > 0, m > 0$ and E, if

- $\widehat{\mathbb{W}}_0 :\xRightarrow[\mathsf{emp}]{o} (\mathbb{T}, \mathtt{t}, \mathbb{d}, \Sigma)$, $o \neq \tau$, $\mathtt{t}_1 \neq \mathtt{t}_2$;
- $(\mathbb{T}, \mathtt{t}_1, \mathbb{d}, \Sigma) :\xRightarrow[\Delta_1]{\text{e-syn}(E)}{}^{n} \widehat{\mathbb{W}}_1$, $\widehat{\mathbb{W}}_1 :\xRightarrow[\Delta'_1]{\tau}{}^{*} \widehat{\mathbb{W}}'_1$;
- $(\mathbb{T}, \mathtt{t}_2, \mathbb{d}, \Sigma) :\xRightarrow[\Delta_2]{\text{e-syn}(E)}{}^{m} \widehat{\mathbb{W}}_2$, $\widehat{\mathbb{W}}_2 :\xRightarrow[\Delta'_2]{\tau}{}^{*} \widehat{\mathbb{W}}'_2$;
- $(\Delta_1 \cup \Delta'_1) \frown (\Delta_2 \cup \Delta'_2)$;
- $\neg(\exists \widehat{\mathbb{W}}.\ \widehat{\mathbb{W}}_0 \rightrightarrows^{*} \widehat{\mathbb{W}} \wedge (\widehat{\mathbb{W}}, E) :\Longrightarrow \mathsf{WA})$;

then $\exists \widehat{\mathbb{W}}.\ \widehat{\mathbb{W}}_0 \rightrightarrows^{*} \widehat{\mathbb{W}} \wedge \widehat{\mathbb{W}} :\Longrightarrow \mathsf{Race}$.

Proof. We prove it by induction on n and apply Lemma 83. (* on paper proof *) □

Lemma 85 (Reachclosed step)**.** If $(\widehat{W}, \delta_0) \leqslant_{\mu}^{i,E} (\widehat{\mathbb{W}}, \Delta_0)$, $\mathsf{Safe}(\widehat{W})$, $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}} \widehat{W}'$, then $(\delta_0 \cup \delta) \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$.

Proof. Proof by induction on the index i. □

Lemma 86 (Reachclosed steps)**.** If $(\widehat{W}, \delta_0) \leqslant_{\mu}^{i,E} (\widehat{\mathbb{W}}, \Delta_0)$, $\mathsf{Safe}(\widehat{W})$, $\widehat{W} \overset{\tau}{\underset{\delta}{\Rightarrow}}{}^{n} \widehat{W}'$, then $\delta \subseteq (\mathsf{curF}(\widehat{W}) \cup \mu.S)$.

Proof. Proof by induction on the number of steps and applying Lemma 85. □

F ANALYSIS CORRECTNESS PRESERVATION

Proof. We first do intro and unfold AC, we get the following assumptions:

$$\hat{P} \leqslant_E \hat{\mathbb{P}} \tag{F.1}$$

$$(\hat{P}, E) :\Longmapsto \mathsf{WA} \tag{F.2}$$

And we need to prove the following goal:

$$(\hat{\mathbb{P}}, E) :\Longmapsto \mathsf{WA}$$

We do inversion on (F.2) and get:

$$\hat{P} :\overset{load}{\Longmapsto} \widehat{W} \tag{F.3}$$

$$\widehat{W} \rightrightarrows^* \widehat{W}' \tag{F.4}$$

$$(\widehat{W}', E) :\Longmapsto \mathsf{WA} \tag{F.5}$$

By applying Lemma 79 on (F.1), (F.3) and (F.4), we get there exists i, μ, $\widehat{\mathbb{W}}$, $\widehat{\mathbb{W}}'$, Δ_0, δ_0 such that:

$$\hat{\mathbb{P}} :\overset{load}{\Longmapsto} \widehat{\mathbb{W}} \tag{F.6}$$

$$\widehat{\mathbb{W}} \rightrightarrows^* \widehat{\mathbb{W}}' \tag{F.7}$$

$$(\widehat{W}', \delta_0) \leqslant_{\mu}^{i,E} (\widehat{\mathbb{W}}', \Delta_0) \tag{F.8}$$

Then, we do inversion on (F.5) and get:

$$(\widehat{W}', E) :\Longmapsto \mathsf{WA}_0 \vee (\widehat{W}', E) :\Longmapsto \mathsf{WA}_1 \tag{F.9}$$

We destruct (F.9) and discuss each case repectively:

- By the definition of $(\widehat{W}', E) :\Longmapsto \mathsf{WA}_0$, we get the following:

$$\widehat{W}' :\underset{\mathtt{emp}}{\overset{o}{\Longmapsto}} (T, \mathtt{t}, \mathbb{d}, \sigma), o \neq \tau \tag{F.10}$$

$$\mathtt{t}_1 \neq \mathtt{t}_2 \tag{F.11}$$

$$\delta_1 \frown \delta_2 \tag{F.12}$$

$$(\mathrm{id}_1, \_) \in E \tag{F.13}$$

$$T(\mathtt{t}_1) = (tl_1, F_1, \kappa_1), F_1 \vdash (\kappa_1, \sigma) \overset{\tau}{\underset{\delta_1}{\mapsto}}{}^*(\kappa_1', \sigma_1'), \mathbb{d}(\mathtt{t}_1) = \mathrm{id}_1 \tag{F.14}$$

$$T(\mathtt{t}_2) = (tl_2, F_2, \kappa_2), F_2 \vdash (\kappa_2, \sigma) \overset{\tau}{\underset{\delta_2}{\mapsto}}{}^*(\kappa_2', \sigma_2'), \mathbb{d}(\mathtt{t}_2) = d_2 \tag{F.15}$$

And we get the following holds:

$$(T, \mathtt{t}_1, \mathbb{d}, \sigma) :\overset{\tau}{\underset{\delta_1}{\Longrightarrow}}{}^* (T\{\mathtt{t}_1 \rightsquigarrow (tl_1, F_1, \kappa_1)\}, \mathtt{t}_1, \mathbb{d}, \sigma_1') \tag{F.16}$$

$$(T, \mathtt{t}_2, \mathbb{d}, \sigma) :\overset{\tau}{\underset{\delta_2}{\Longrightarrow}}{}^* (T\{\mathtt{t}_2 \rightsquigarrow (tl_2, F_2, \kappa_2)\}, \mathtt{t}_2, \mathbb{d}, \sigma_2') \tag{F.17}$$

From (F.10) and (F.8), we get $\exists \widehat{\mathbb{W}}'', \mathbb{T}, \Sigma, \Delta, j$, for any $\mathtt{t}'' \in \mathsf{dom}(T)$, the following hold:

$$\widehat{\mathbb{W}}' \Longrightarrow^* \widehat{\mathbb{W}}'' \tag{F.18}$$

$$\widehat{\mathbb{W}}'' :\underset{\mathsf{emp}}{\overset{o}{\Longrightarrow}} (\mathbb{T}, \mathtt{t}'', \mathbb{d}, \Sigma) \tag{F.19}$$

$$((T, \mathtt{t}'', \mathbb{d}, \sigma), \mathsf{emp}) \leqslant_{\mu}^{j,E} ((\mathbb{T}, \mathtt{t}'', \mathbb{d}, \Sigma), \mathsf{emp}) \tag{F.20}$$

According to Lemma 80, (F.16), (F.17), we discuss each case repectively:

– $\exists \widehat{W}_1', \delta_1', \widehat{\mathbb{W}}_1', \Delta_1'$, such that
  * $(T, \mathtt{t}_1, \mathbb{d}, \sigma) :\underset{\delta_1'}{\overset{\tau}{\Longrightarrow}}{}^* \widehat{W}_1', \delta_1 \subseteq \delta_1'$;
  * $(\mathbb{T}, \mathtt{t}_2, \mathbb{d}, \Sigma) :\underset{\Delta_1'}{\overset{\tau}{\Longrightarrow}}{}^* \widehat{\mathbb{W}}_1', \mathsf{FPmatch}(\mu, \Delta_1', \delta_1')$;

  and $\widehat{W}_2', \delta_2', \widehat{\mathbb{W}}_2', \Delta_2'$, such that
  * $(T, \mathtt{t}_2, \mathbb{d}, \sigma) :\underset{\delta_2'}{\overset{\tau}{\Longrightarrow}}{}^* \widehat{W}_2', \delta_2 \subseteq \delta_2'$;
  * $(\mathbb{T}, \mathtt{t}_2, \mathbb{d}, \Sigma) :\underset{\Delta_2'}{\overset{\tau}{\Longrightarrow}}{}^* \widehat{\mathbb{W}}_2', \mathsf{FPmatch}(\mu, \Delta_2', \delta_2')$;

  According to Lemma 86, we get there exists $l \in \mu.S$:

$$(l \in \delta_1'.ws \wedge l \in \delta_2') \vee (l \in \delta_2'.ws \wedge l \in \delta_1')$$

  According to the definition of FPmatch, we get there exists $l' \in \mu.\mathbb{S}$:

$$(l' \in \Delta_1'.ws \wedge l' \in \Delta_2') \vee (l' \in \Delta_2'.ws \wedge l' \in \Delta_1')$$

  Then, we get: $(\widehat{\mathbb{W}}'', E) :\Longrightarrow \mathsf{WA}_0$, and we finish the proof of this case.

– $\exists \widehat{W}_1', \delta_1', \widehat{\mathbb{W}}_1', \Delta_1'$, such that
  * $(T, \mathtt{t}_1, \mathbb{d}, \sigma) :\underset{\delta_1'}{\overset{\tau}{\Longrightarrow}}{}^* \widehat{W}_1', \delta_1 \subseteq \delta_1'$;
  * $(\mathbb{T}, \mathtt{t}_1, \mathbb{d}, \Sigma) :\underset{\Delta_1'}{\overset{\tau}{\Longrightarrow}}{}^* \widehat{\mathbb{W}}_1', \mathsf{FPmatch}(\mu, \Delta_1', \delta_1')$;

  and $\exists \widehat{W}_2', \delta_2', \widehat{\mathbb{W}}_2, \widehat{\mathbb{W}}_2', \Delta_2, \Delta_2'$, such that
  * $(T, \mathtt{t}_2, \mathbb{d}, \sigma) :\underset{\delta_2'}{\overset{\tau}{\Longrightarrow}}{}^* \widehat{W}_2', \delta_2 \subseteq \delta_2', \mathbb{d}(\mathtt{t}_2) = 0$;
  * $(\mathbb{T}, \mathtt{t}_2, \mathbb{d}, \Sigma) :\underset{\Delta_2}{\overset{\mathsf{e\text{-}syn}(E)}{\Longrightarrow}}{}^+ \widehat{\mathbb{W}}_2, \widehat{\mathbb{W}}_2 :\underset{\Delta_2'}{\overset{\tau}{\Longrightarrow}}{}^* \widehat{\mathbb{W}}_2', \mathsf{FPmatch}(\mu, \Delta_2 \cup \Delta_2', \delta_2)$;

  According to Lemma 86, we get there exists $l \in \mu.S$:

$$(l \in \delta_1'.ws \wedge l \in \delta_2') \vee (l \in \delta_2'.ws \wedge l \in \delta_1')$$

  According to the definition of FPmatch, we get there exists $l' \in \mu.\mathbb{S}$:

$$(l' \in \Delta_1'.ws \wedge l' \in (\Delta_2 \cup \Delta_2')) \vee (l' \in (\Delta_2 \cup \Delta_2').ws \wedge l' \in \Delta_1')$$

  Then, we finish the proof of this case by applying Lemma 87.

– $\exists \widehat{W}_1', \delta_1', \widehat{\mathbb{W}}_1, \widehat{\mathbb{W}}_1', \Delta_1, \Delta_1'$, such that
  * $(T, \mathtt{t}_1, \mathbb{d}, \sigma) :\underset{\delta_1'}{\overset{\tau}{\Longrightarrow}}{}^* \widehat{W}_1', \delta_1 \subseteq \delta_1', \mathbb{d}(\mathtt{t}_1) = 0$;
  * $(\mathbb{T}, \mathtt{t}_1, \mathbb{d}, \Sigma) :\underset{\Delta_1}{\overset{\mathsf{e\text{-}syn}(E)}{\Longrightarrow}}{}^+ \widehat{\mathbb{W}}_1, \widehat{\mathbb{W}}_1 :\underset{\Delta_1'}{\overset{\tau}{\Longrightarrow}}{}^* \widehat{\mathbb{W}}_1', \mathsf{FPmatch}(\mu, \Delta_1 \cup \Delta_1', \delta_1')$;

  and this is contridicted with $\mathbb{d}(\mathtt{t}) = \mathsf{id}_1$, according to (F.14).

– The proof of this case is similar with the previous one.

- By the definition of $(\widehat{W}', E) :\Longmapsto \mathsf{WA}_1$, we get the following:

$$\widehat{W}' \rightrightarrows^* \widehat{W}_1^{\mathtt{t}} \tag{F.21}$$

$$\widehat{W}_1^{\mathtt{t}} :\xRightarrow[\mathsf{emp}]{\mathsf{id}'.\mathsf{ExtA}} (T, \mathtt{t}', \mathbb{d}, \sigma) \tag{F.22}$$

$$\widehat{W}_1.\mathbb{d}(\mathtt{t}) = \mathsf{id} \tag{F.23}$$

$$((\mathsf{id}, \mathsf{id}_1) \in E \wedge \mathsf{id}_1 \neq \mathsf{id}') \vee ((\mathsf{id}_2, \mathsf{id}') \in E \wedge \mathsf{id}_2 \neq \mathsf{id}) \tag{F.24}$$

We finish the proof by applying Lemma 88.

□

Lemma 87 (Constructing wrong analysis - 0)**.**
For any $\widehat{\mathbb{W}}_0$, o, $\mathbb{T}$, $\mathtt{t}$, $\mathbb{d}$, Σ, $\mathtt{t}_1$, $\widehat{\mathbb{W}}_1$, Δ_1, $\mathtt{t}_2$, $\widehat{\mathbb{W}}_2$, $\widehat{\mathbb{W}}_2'$, Δ_2, Δ_2', n, id and E, if

- $\widehat{\mathbb{W}}_0 :\xRightarrow[\mathsf{emp}]{o} (\mathbb{T}, \mathtt{t}, \mathbb{d}, \Sigma)$, $o \neq \tau$, $\mathtt{t}_1 \neq \mathtt{t}_2$;
- $(\mathbb{T}, \mathtt{t}_1, \mathbb{d}, \Sigma) :\xRightarrow[\Delta_1]{\tau}{}^* \widehat{\mathbb{W}}_1$;
- $(\mathbb{T}, \mathtt{t}_2, \mathbb{d}, \Sigma) :\xRightarrow[\Delta_2]{\mathsf{e\text{-}syn}(E)}{}^n \widehat{\mathbb{W}}_2$, $\widehat{\mathbb{W}}_2 :\xRightarrow[\Delta_2']{\tau}{}^* \widehat{\mathbb{W}}_2'$;
- $\Delta_1 \frown (\Delta_2 \cup \Delta_2')$;
- $\mathbb{d}(\mathtt{t}_1) = \mathsf{id}$, $(\mathsf{id}, \_) \in E$;

then $\exists \widehat{\mathbb{W}}.\ \widehat{\mathbb{W}}_0 \rightrightarrows^* \widehat{\mathbb{W}} \wedge (\widehat{\mathbb{W}}, E) :\Longmapsto \mathsf{WA}_0$.

Proof. Prove by induction on n. (* on paper proof *) □

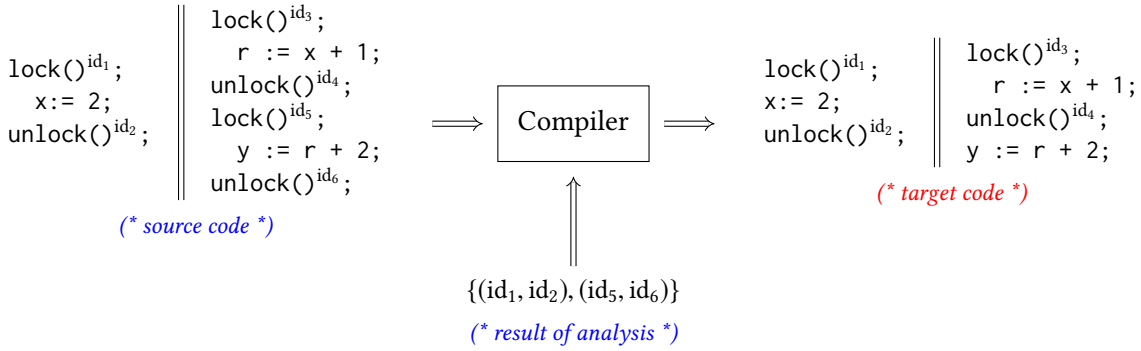
Lemma 88 (Constructing wrong analysis - 1)**.**
For any $\widehat{W}$, $\widehat{W}_1$, $\mathtt{t}$, $\mathtt{t}'$, id, id′, $\mathbb{d}$, σ, $\widehat{\mathbb{W}}$, δ_0, Δ_0, n, i, E and μ, if

- $\widehat{W} \rightrightarrows^n \widehat{W}_1^{\mathtt{t}}$, $\widehat{W}_1^{\mathtt{t}} :\xRightarrow[\mathsf{emp}]{\mathsf{id}'.\mathsf{ExtA}} (T, \mathtt{t}, \mathbb{d}, \sigma)$;
- $\widehat{W}_1.\mathbb{d}(\mathtt{t}) = \mathsf{id}$;
- $((\mathsf{id}, \mathsf{id}_1) \in E \wedge \mathsf{id}_1 \neq \mathsf{id}') \vee ((\mathsf{id}_2, \mathsf{id}') \in E \wedge \mathsf{id}_2 \neq \mathsf{id})$;
- $(\widehat{W}, \delta_0) \leqslant_\mu^{i,E} (\widehat{\mathbb{W}}, \Delta_0)$;

then there exists $\widehat{\mathbb{W}}_1$, $\mathbb{T}$ and Σ, such that:

- $\widehat{\mathbb{W}} \rightrightarrows^* \widehat{\mathbb{W}}_1^{\mathtt{t}}$, $\widehat{\mathbb{W}}_1^{\mathtt{t}} :\xRightarrow[\mathsf{emp}]{\mathsf{id}'.\mathsf{ExtA}} (\mathbb{T}, \mathtt{t}, \mathbb{d}, \Sigma)$;
- $\widehat{\mathbb{W}}_1.\mathbb{d}(\mathtt{t}) = \mathsf{id}$;
- $((\mathsf{id}, \mathsf{id}_1) \in E \wedge \mathsf{id}_1 \neq \mathsf{id}') \vee ((\mathsf{id}_2, \mathsf{id}') \in E \wedge \mathsf{id}_2 \neq \mathsf{id})$.

Proof. Prove by induction on n. □


lock()^id1;
x:= 2;
unlock()^id2;
lock()^id3;
r := x + 1;
unlock()^id4;
lock()^id5;
y := r + 2;
unlock()^id6;
(* source code *)
Compiler
{(id1, id2), (id5, id6)}
(* result of analysis *)
lock()^id1;
x:= 2;
unlock()^id2;
lock()^id3;
r := x + 1;
unlock()^id4;
y := r + 2;
(* target code *)