Grounding Thin-Air Reads with Event Structures
Technical Appendix

SOHAM CHAKRABORTY, MPI-SWS, Germany
VIKTOR VAFEIADIS, MPI-SWS, Germany

This is the technical appendix of the article “Grounding Thin-Air Reads with Event Structures.” It contains the proofs of the simulation of the promising semantics by weakest, DRF theorems, and various compilation correctness results along with the evaluation of the proposed models on the Java causality testcases and the construction rules of weakestmo-llvm.

CONTENTS

- Appendix A contains the proof of simulation of promising semantics by weakest.
- Appendix B contains the event structures for causality test cases.
- Appendix C contains the proofs of DRF theorems.
- Appendix D contains the event structure construction rules for weakestmo-llvm.
- Appendix E contains the proof of monotonicity property.
- Appendix F contains the proof of correctness of reorderings.
- Appendix G contains the proof of correctness of eliminations.
- Appendix H establishes the correctness of speculative load introduction in weakestmo-llvm.
A PROVING SIMULATION OF PROMISING SEMANTICS BY WEAKEST

We restate the definition of simulation relation.

**Definition 6.** Let \( P \) be a program with \( T \) threads, \( \Pi \subseteq T \) be a subset of threads, \( G \) be a weakest event structure, and \( MS = \langle TS_i, S_i, M_i \rangle \) be a promise machine state. We say that \( G \sim \Pi MS \) holds if there exist \( \mathcal{W}, \mathcal{S} \), and \( \mathcal{Sc} \) such that the following conditions hold:

1. \( G \) is consistent according to the weakest model: \( \text{isCons}_{\text{WEAKEST}}(G) \).
2. The local state of each thread in \( MS \) contains the program of that thread along with the sequence of covered events of that thread: \( \forall i. TS(i).\sigma = \langle P(i), \text{labels}(\text{sequence}_{\text{spo}}(S_i)) \rangle \).
3. Whenever \( \mathcal{W} \) maps an event of \( G \) to a message in \( MS \), then the location accessed and the written values match: \( \forall e \in \text{dom}(\mathcal{W}). e.\text{loc} = \mathcal{W}(e).\text{loc} \land e.\text{wval} = \mathcal{W}(e).\text{wval} \).
4. All outstanding promises of threads \( (T \setminus \Pi) \) have corresponding write events in \( G \) that are po-after \( \mathcal{S} \): \( \forall i \in (T \setminus \Pi). \forall e \in (\mathcal{S}_0 \cup \mathcal{S}_1). TS(i).P \subseteq \{\mathcal{W}(e') | (e, e') \in G.\text{po}\} \).
5. For every location \( x \) and thread \( i \), the thread view of \( x \) in the promise state \( MS \) records the timestamp of the maximal write visible to the covered events of thread \( i \).

\[
\forall i, x. TS(i).V(x) = \max\{\mathcal{W}(e).ts | e \in \text{dom}(\mathcal{W}), (e.\text{loc}, e.\text{wval}) \in MS(x)\}
\]

6. The \( \mathcal{S} \) events satisfy coherence: \( \text{shb}; \text{seco}^2 \) is irreflexive.
7. The atomicity condition holds for the \( \mathcal{S} \) events: \( \text{sfr}; \text{smo} \) is irreflexive.
8. The \( \mathcal{Sc} \) fences are appropriately ordered by \( \text{sc}: \{\text{shb}; \text{shb}; \text{seco}; \text{shb}; \{\text{fr}, \text{fr} \} \} \subseteq \text{sc} \).
9. The behavior of \( MS \) matches that of the \( \mathcal{S} \) events: \( \text{Behavior}(MS) = \text{Behavior}(G, \mathcal{W}, \mathcal{S}) \).

Before proceeding further we introduce certain definition and observations which we use in the proofs.

**Auxiliary Definitions.**

- We define immediate relation: given a relation \( R \) we use \( \text{imm}(R) \) to denote the immediate edges of \( R \), that is, \( \text{imm}(R) \triangleq R \setminus (R; R) \).
- Given the Behavior, \( \text{Behavior}|_x \) denotes the \( \{(x, v)\} \) where \( v \) is the value at location \( x \).
- We define \( \text{swe} \) the external synchronization relation, that is, \( \text{swe} \triangleq \text{sw} \setminus \text{po} \).
- In the following discussion \( \text{op} \) denotes the promise machine state transition operation which results in event \( a \) in the event structure and the promise machine reaches machine state \( MS_a \).
- \( E \mathcal{W} \) denotes the set of read write events where a write is \( \mathcal{W} \)-mapped to some PS message or a read reads from a \( \mathcal{W} \)-mapped write.

\[
E \mathcal{W} \triangleq \{e \in G.E | e \in \mathcal{W} \cap \text{dom}(\mathcal{W}) \lor \exists w \in \text{dom}(\mathcal{W}). G.rf(w, e)\}
\]

- \( \text{ts}(e) \) returns the timestamp of a write or view of a read on the respective locations.

\[
\text{ts}(e) \triangleq \begin{cases} 
\mathcal{W}(e).ts & \text{if } e \in \text{St} \cap E \mathcal{W} \\
\mathcal{W}(w).ts & \text{if } e \in \text{Ld} \cap E \mathcal{W} \text{ and } G.rf(w, e)
\end{cases}
\]

- In the promise machine \( \text{cur} \), \( \text{rel} \), \( \text{acq} \) denotes the current, release, acquire thread views similar to Kang et al. [2017]. The \( \text{cur} \) view is default.

Additionally, we enlist certain observations regarding the relation between the promise machine and event structure.

**Observations.** Considering the promising semantics and event structure we observe the followings.

1. The \((G.E \setminus \mathcal{S})\) events correspond to the certificate steps of a promise. The certificate steps do not have any release or fence operations. Hence there is no release or fence event in \((G.E \setminus \mathcal{S})\).
As a result, these events do not have outgoing $G.sw$ edges. Hence the source event of an incoming $G.sw$ edge is in $\mathcal{S}$, that is, $G.sw \subseteq (\mathcal{S} \times G.E)$. Also for $(G.E \setminus \mathcal{S})$ events the outgoing $G.hb$ edges are only $G.po$ edges.

(2) If a write event $w \in (G.E \setminus \mathcal{S})$ is mapped to some promise message, that is, $\mathcal{W}(w) \neq \bot$, then $w$ can have outgoing $G.rfe$ and $mo$ edges.

Now we state and prove Lemma 3 which use in further proofs.

**Lemma 3.**

Given a program $\mathcal{P}$, suppose $MS$ is a promise machine state and $G$ is an weakest event structure such that $MS$ simulates $MS_1;G \sim MS$. Then,

1. if two events $a, b \in E^{wh}$ on the same memory location are related by $(G.hb;G.eco^2_{strong})$ relation in $G$, then $ts(a) < ts(b)$. Moreover, if $b$ is a write event then $ts(a) < ts(b)$.
2. if two events $a, b \in \mathcal{S}$ on the same memory location are related by $(shb;seco^2)$, then $ts(a) < ts(b)$. Moreover, if $b$ is a write event then $ts(a) < ts(b)$.
3. If $r$ reads from $w$ such that $(w, r) \in (G.ew;G.jf)$ holds then $w$ and $r$ are not $hb$ related, that is $(w, r) \notin (G.hb \cup G.hb^{-1})$.
4. Whenever $imm(spo)(a, b)$ does not hold, $(a, b) \in [G.\mathcal{F}_{sc} \cap \mathcal{S}];shb \cup shb;seco;shb;[G.\mathcal{F}_{sc} \cap \mathcal{S}]$ implies $MS_1;S < MS_2;S$.

**Proof.** We study the component relations of $(G.hb;G.eco^2_{strong})$ and $(shb;seco^2)$.

- **case** $(a, b) \in G.po_x$
  
  Let $a$ and $b$ be in the $i^{th}$ thread in the event structure.
  
  In that case $ts(a) = MS_a.\mathcal{F}(i),V(x)$ and $ts(b) = MS_b.\mathcal{F}(i),V(x)$.
  
  We know that promise machine always extends thread view on each location.
  
  Hence $MS_a.\mathcal{F}(i),V(x) \leq MS_b.\mathcal{F}(i),V(x)$.
  
  As a result, $ts(op_a) \leq ts(op_b)$.

- **case** $(a, b) \in G.rf$
  
  In this case $op_a$ creates the message $\langle x : \neg @t \rangle$ and $op_b$ reads from the same message in the promise machine. As a result, $ts(a) = ts(b)$.

- **case** $(a, b) \in G.ew$
  
  We create $G.ew$ for the event pairs corresponding to the promise and fulfill operations. In this case $op_a, op_b$ are promise and fulfill operations respectively. The promise operation append a message and the fulfill operation removes the same message from the message queue. Hence, $ts(a) = ts(b)$.

- **case** $(a, b) \in G.jf$
  
  We know that
  
  $G.jf(a, b) \implies (ts(a) = ts(b))$
  
  $G.ew(a, b) \implies (ts(a) = ts(b))$, and
  
  $G.jf = G.ew^2;G.jf$
  
  As a result, $G.jf(a, b) \implies (ts(a) = ts(b))$.

- **case** $(a, b) \in G.hb$
  
  In this case $(a, b) \in (G.po \cup G.sw)^+$.
  
  If $G.po(a, b)$ then $(a, b) \in G.po_x$ and hence $ts(a) < ts(b)$.
  
  Otherwise there exists some event $c$ and $d$ such that $(a, c) \in G.po \land (c, d) \in G.sw \land G.hb^2(c, b)$.
  
  Following the promising semantics $ts(a) \leq MS_c.\mathcal{F}(c.tid),V(x)$.
  
  Then considering $c$ and $d$ access types
  
  - $c \in G.\mathcal{F}_{rel} \cap [\text{Rel}]$ and $d \in G.R \cap [\text{Acq}]$
In this case there exists some event \( w \in E_W \) such that \( G.po(c, w) \), \( w.loc = d.loc, w \in G.W_{rel} \), and \( (w, d) \in G.jf^+ \) and \( op_w \) results in message \( m = \langle -\rangle \). In this case view \( MS_a.\mathcal{TS}(a.tid).V(x) \) is included in the message view \( m.R, \) that is, \( MS_a.\mathcal{TS}(a.tid).V(x) \in m.R. \) Now if \( G.jf(w, d) \) then \( m.R \in MS_d.\mathcal{TS}(d.tid).\)cur and hence \( MS_a.\mathcal{TS}(a.tid).V(x) \in m.R \in MS_b.\mathcal{TS}(b.tid).\)cur. Otherwise if \( G.jf(w, u_1) \land G.jf(u_1, u_2) \land \ldots \land G.jf(u_n, d) \) where \( u_1, u_2, \ldots, u_n \in (G.U \cap E_W) \) then following the promising semantics

(i) if \( w.loc \neq c.loc \) then the view \( MS_a.\mathcal{TS}(a.tid).V(x) \) propagates through the messages created by \( u_1, u_2, \ldots, u_n \) and finally reaches \( d, \) that is, \( MS_2.\mathcal{TS}(a.tid).V(x) \in m.R \in MS_d.\mathcal{TS}(d.tid).\)cur holds.

(ii) if \( w.loc = c.loc \) then \( G.po_x(c, w) \) and hence \( ts(c) < ts(w) \) and in consequence \( ts(c) < MS_d.\mathcal{TS}(d.tid).V(x). \) Hence, considering (i) and (ii), \( MS_c.\mathcal{TS}(c.tid).V(x) \leq MS_d.\mathcal{TS}(d.tid).V(x). \)

- \( c \in G.W \cap [Rel] \) and \( d \in G.R \cap [Acq] \)

Similarly to above, the view \( MS_c.\mathcal{TS}(c.tid).V(x) \) propagates to \( MS_d.\mathcal{TS}(d.tid).\)cur by a read-from or release sequence and in that case \( MS_c.\mathcal{TS}(c.tid).V(x) \leq MS_d.\mathcal{TS}(d.tid).V(x). \)

- \( c \in G.F \cap [Rel] \) and \( d \in G.F \cap [Acq] \)

In this case there exists some event \( w, r \in E_W \) such that \( G.po(c, w), w \in G.W_{rel}, G.po(r, d), r \in G.R_{rel}, \) and \( (w, r) \in G.jf^+ \). Note that since a fence \( d \) is in \( E_W \), the \( G.po \)-predecessor \( r \) is also in \( E_W \).

Similar to the earlier case \( MS_c.\mathcal{TS}(c.tid).V(x) \) propagates to \( r \) and gets included in \( MS_r.\mathcal{TS}(r.tid).V.acq. \)

Finally \( MS_r.\mathcal{TS}(k).V.acq \) is included in \( MS_d.\mathcal{TS}(d.tid).\)cur and in turn \( MS_c.\mathcal{TS}(c.tid).V(x) \leq MS_d.\mathcal{TS}(d.tid).V(x). \)

- \( c \in G.W \cap [Rel] \) and \( d \in G.F \cap [Acq] \)

Similar to the earlier case \( MS_d.\mathcal{TS}(d.tid).\)cur gets the \( MS_c.\mathcal{TS}(i).V(x) \) or an updated view of \( x \) as and as a result, \( MS_c.\mathcal{TS}(c.tid).V(x) \leq MS_d.\mathcal{TS}(d.tid).V(x). \)

As a result, \( ts(a) \leq MS_d.\mathcal{TS}(d.tid).V(x) \) and following the \( G.\text{hb} \) path \( ts(a) \leq ts(b). \)

In all these \( G.\text{hb} \) cases the \( ts(a) \) propagates to \( b. \) If \( b \) is a write event then it extends the view and updates with a new timestamp. Hence if \( b \) is a write then \( ts(a) < ts(b). \)

Following from this argument, if \( (a, b) \in G.\text{mo}_{\text{strong}} \) then \( ts(a) < ts(b) \) holds.

- \((a, b) \in G.\text{fr}_{\text{strong}} \)

There exists a write \( c \) such that \( (a, c) \in G.\text{rf}^{-1} \land (c, b) \in G.\text{mo}_{\text{strong}}. \)

In this case \( ts(a) = ts(c) \) and \( ts(c) < ts(b) \) holds.

As a result, \( ts(a) < ts(b) \) holds.

Thus considering the component relations of \( (G.\text{hb}; G.\text{eco}_{\text{strong}})_{\text{loc}} \) results in \( \leq \)-order following the timestamps of the corresponding promise machine.

We now study the component relations of \( (shb; seco^2). \)

- \((a, b) \in shb \)

Considering the definition, in this case, \( shb \subseteq G.\text{hb} \cap (E_W \times E_W) \).

Hence \( shb(a, b) \) implies \( ts(a) \leq ts(b) \) and if \( b \) is a write event then \( ts(a) < ts(b) \).

- \((a, b) \in srf. \)

Considering the definition, in this case, \( srf \subseteq G.\text{rf} \cap (E_W \times E_W) \). Hence \( srf(a, b) \) implies \( ts(a) = ts(b) \)
• \((a, b)\in \text{sмо}\).

We know \(\text{sмо} \subseteq \text{мо}\) and hence following the definition of \text{мо}, \(\text{sмо}(a, b)\) implies \(\text{ts}(a) < \text{ts}(b)\).

• \((a, b)\in \text{sфр}\).

Hence \((a, b)\in (\text{sфр}^{-1}; \text{sмо})\). As a result, \(\text{ts}(a) < \text{ts}(b)\).

Thus considering the component relations of \((\text{шб}; \text{секо}^?)|\text{лос}\) results in \(\leq\)-order following the timestamps of the corresponding promise machine. Moreover, when \((a, b)\in (\text{шб}; \text{секо}^?)|\text{лос}\) and \(b\) is a write then \(\text{ts}(a) < \text{ts}(b)\).

We now study the relation between \(w\) and \(r\) when \((w, r)\in (G.\text{ew}; G.\text{jf})\).

We consider two cases

• \text{case} \(G'.\text{hb}(w, r)\) does not hold as \(w.\text{ord} \sqsubseteq \text{REL}\).

• \text{case} \(G'.\text{hb}(r, w)\).

From (1), in this case \(G'.\text{hb}(r, w)\) implies \(\text{ts}(r) < \text{ts}(w)\). However, we know, \(G.\text{rf}(w, r)\) implies \(\text{ts}(r) = \text{ts}(w)\).

Hence a contradiction and \(G'.\text{hb}(r, w)\) does not hold.

As a result, \((w, r)\notin (G.\text{hb} \cup G.\text{hb}^{-1})\). (3)

We have to show that \((a, b)\in [G.\text{fc} \cap \mathbb{S}]\); \text{shb} \cup \text{shb}; \text{seко}; \text{shb}; [G.\text{fc} \cap \mathbb{S}]\) implies \(\text{MS}_a.\mathbb{S} \leq \text{MS}_b.\mathbb{S}\).

We consider two cases

• \text{case} \(G'.\text{hb}(w, r)\) does not hold as \(w.\text{ord} \sqsubseteq \text{REL}\).

• \text{case} \(G'.\text{hb}(r, w)\).

From (1), in this case \(G'.\text{hb}(r, w)\) implies \(\text{ts}(r) < \text{ts}(w)\). However, we know, \(G.\text{rf}(w, r)\) implies \(\text{ts}(r) = \text{ts}(w)\).

Hence a contradiction and \(G'.\text{hb}(r, w)\) does not hold.

As a result, \((w, r)\notin (G.\text{hb} \cup G.\text{hb}^{-1})\). (3)

From the similar argument as (2), we can show that the timestamps increase or remain same through \text{seко} edges from \(c\) to \(d\) on location \(c.\text{loc}\).

Hence \(\text{seко}(c, d)\) implies \(\text{MS}_c.\mathbb{T}\mathbb{S}(c.\text{tid}).\mathbb{V} < \text{MS}_d.\mathbb{T}\mathbb{S}(d.\text{tid}).\mathbb{V}\) and

\(\text{shb}(d, b)\) implies \(\text{MS}_d.\mathbb{T}\mathbb{S}(d.\text{tid}).\mathbb{V} < \text{MS}_b.\mathbb{S}\).

As a result, whenever \(\text{imm}(\text{spo})(a, b)\) does not hold,

\((a, b)\in [G.\text{fc} \cap \mathbb{S}]\); \text{shb} \cup \text{shb}; \text{seко}; \text{shb}; [G.\text{fc} \cap \mathbb{S}]\) implies \(\text{MS}_a.\mathbb{S} < \text{MS}_b.\mathbb{S}\).

Lemma 4. Given a program \(\mathcal{P}\), suppose \(\text{MS}\) is a promise machine state and \(G\) is an weakest event structure such that \(G\) simulates \(\text{MS}\); \(G \sim \text{MS}\). In this case there is no outgoing external-synchronization from \(G.\text{E} \setminus \mathbb{S}\) events, that is, \(\text{dom}(G.\text{swe}) \subseteq \mathbb{S}\).

Proof. The simulation construction steps ensure that the conflicting events of \(\mathbb{S}\), that is, \(G.\text{E} \setminus \mathbb{S}\) events are created only as part of PS certificate steps in the respective threads.

In the promising semantics the certificate steps are not visible to any other thread. Similarly in event structure \(G\) the there is no outgoing \(\text{rfe}\) edge from \(G.\text{E} \setminus \mathbb{S}\) events except the event corresponding to the promise. Let that event be \(e_p\).

From PS we know that \(e_p.\text{ord} \sqsubseteq \text{RLX}\) and certificate steps do not have any release fence. Hence \(G.\mathcal{F}_{\text{zrel}} \cap (G.\text{E} \setminus \mathbb{S}) = \emptyset\).

Hence there is no outgoing \(G.\text{swe}\) edge from \(G.\text{E} \setminus \mathbb{S}\) events and \(\text{dom}(G.\text{swe}) \subseteq \mathbb{S}\) holds. ☐

Next we restate and prove Lemma 1.

Lemma 1. \(G \sim (i) \text{MS} \land \text{MS} \xrightarrow{np} I \text{MS'} \implies \exists G'. G \rightarrow_{\text{фик,weakest}^*} G' \land G' \sim (i) \text{MS'}\).
Before going to the proof we restate the proof idea.

**Proof Idea.** The $G'$ is constructed in two steps.

1. First, for a non-promise operation $np$ we either append a corresponding event $e'$ to $G$ or we identify an existing corresponding event $e'$ in $G$. In earlier case $G$ is extended to $G'$ and in later case $G' = G$.

2. Next, we check whether $TS_i$ has outstanding promises. If so, then we know that there is a promise-free certificate which fulfills the outstanding promises. In that case, for each non-promise certificate step we extend the event structure following the rules in *WEAKEST* and at each step the constructed event structure remains consistent.

In this construction $G$ and $MS$ are related by $S$, $\mathcal{W}$, and we define $S'$, $\mathcal{W}'$ to relate the $G'$ and $MS'$. By using the definitions of $S'$, $\mathcal{W}'$ we show that $G' \sim_{\{1\}} MS'$ holds. We use the results of Lemma 3 to establish the simulation relation.

**Proof.** We do a case analysis on the operation $op$ of the promise machine transition $MS \xrightarrow{np} i_i$, $MS'$ where $op = np$. From the definition of the simulation relation we know $\forall i. TS(i).\sigma = (\mathcal{P}(i), labels(sequence_{spo}(S_i)))$. Hence we can also make a step from the event structure $G$ to $G'$.

**Case Store** $St(o, x, v)$ creating message $m'$:

In the event structure we extend the event structure $G$ to $G'$. We extend the cover set $S_i$ as well as the relations $(spo, srf, smo)$ to $S'_i$ along with the respective relations $(spo', srf', smo')$ by including an event $e'$ where

1. $\text{dom}(G.po;\{e'\}) = S_0 \cup S_i$,
2. $e' \in S'_i \setminus S_i$, and
3. $\text{labels}(sequence_{G.po}(S_i)).(e'.\text{lab}) \in \mathcal{P}(i)$.

In this case the promise machine is updated as follows.

$M' = M \cup \{m'\}$, $S' = S$, and $TS' = TS[i \mapsto (\langle \mathcal{P}(i), labels(sequence_{spo}(S'_i)) \rangle, V', TS(i).P)]$ where $V' = TS(i).V[x \mapsto m'.ts]$. Now we do a case analysis on whether such a store event $e'$ exists in $G$ or we append a new event.

**Subcase** $\nexists e' \in (G.E_i \setminus S_i). \text{dom}(G.po;\{e'\}) = S_0 \cup S_i \land e'.\text{lab} = St_o(x, v)$:

We create $e'$ such that $e'.\text{lab} = St_o(x, v)$ and append $e'$ to event structure $G$ to create $G'$. Then,

- $G'.E = G.E \cup \{e'\}$
- $G'.po = (G.po \cup \{(e, e') | e \in (S_i \cup S_0)\})^+$
- $G'.jf = G.jf$
- $G'.ew = G.ew$

Let: $\mathcal{W}' \triangleq \mathcal{W}[e' \mapsto m']$.

Based on $\mathcal{W}'$, we derive following definitions in $MS'$.

- $\mathcal{S}' \triangleq \mathcal{S} \cup \{e'\}$
- $\mathcal{mo}' \triangleq \mathcal{mo} \cup \{(a, e') | a \in G.W_x \land \mathcal{W}(a) \not\models \mathcal{W}'(e').ts < \mathcal{W}'(e').ts\}
- $\mathcal{sc}' \triangleq \mathcal{sc}$
- $\mathcal{spo}' \triangleq \mathcal{spo} \cup \{(e, e') | e \in S_0 \cup S'_i\})^+$
- $\mathcal{srf}' \triangleq \mathcal{srf}$

Now we check whether $G' \sim_{\{1\}} (TS', S', M')$.

(1) Condition to show: $G'$ is consistent in *WEAKEST* model.
(CF) We know that $G$ satisfies constraint (CF). Considering the definition of $G'.ecf$, the only incoming $hb$ edge is $G'.po$ and there is no outgoing edge from event $e'$. Hence $G'.ecf$ is irreflexive and $G'$ satisfies (CF).

(CFJ) We know that $G$ satisfies constraint (CFJ). We also know that $G'.jf = G.jf$ and event $e'$ has no outgoing $G'.hb$ or $G'.jf$ edge. Hence $G'.jf \cap G'.ecf = \emptyset$ and $G'$ satisfies (CFJ).

(VISJ) Constraint (VISJ) is preserved in $G'$ as $G'.jf = G.jf$ and $G'$ satisfies constraint (VISJ).

(ICF) We know that $G$ satisfies (ICF). Suppose there exists an event $e_1 \in G$ which is in immediate conflict with $e'$ in $G'$, that is $G'.~(e_1, e')$ holds.

Then (1) $\text{dom}(G.po; \{\{e_1\}\}) = S_0 \cup S_i$,
(2) $e_1 \in S_i \setminus S_j$, and
(3) $\text{labels}(\text{sequence}_{G.po}(S_i)).(e_1.\text{lab}) \in P(i)$.

However, from definition of $e'$ we already know that
(1) $\text{dom}(G.po; \{\{e'\}\}) = S_0 \cup S_j$,
(2) $e' \in S_j \setminus S_i$, and
(3) $\text{labels}(\text{sequence}_{G.po}(S_i)).(e'.\text{lab}) \in P(i)$.

Hence following the determinacy condition we know either $e_1 = e'$ or there exists no such $e_1$. Hence (ICF) is preserved in $G'$.

(ICFJ) Constraint (ICFJ) is preserved in $G'$ as $e' \notin R$ and $G'$ satisfies constraint (ICFJ).

(COH) We know $G$ preserves (COH) constraint, that is, $(G.hb; G.eco^{str}_hf)$ is acyclic. The incoming edges to event $e'$ are $G'.po$, $G'.fr_{strong}$, $G'.hb$ and there is no outgoing edge concerning $G'.hb$ or $G'.eco^{str}_f$. As a result, $(G'.hb; G'.eco^{str}_f)$ is acyclic and $G'$ preserves (COH) constraint.

(2) Condition to show: The local state of each thread in MS' contains the program of that thread along with the sequence of covered events in $G'$ of that thread.

In this we have to show $\forall j. \mathcal{T}S'(j).\sigma = (\mathcal{P}(j), \text{labels}(\text{sequence}_{spo}(S'_j)))$.

We know that the relation holds between MS and $G$.

\textbf{case} For $j \neq i$, it is trivial because $\mathcal{T}S'(j) = \mathcal{T}S(j)$ holds from MS to MS' and $S'_j = S_j$ holds from $G$ to $G'$.

\textbf{case} For $j = i$, we know $\mathcal{T}S(i).\sigma = (\mathcal{P}(i), \text{labels}(\text{sequence}_{spo}(S_i)))$.

Hence following the definition of $\mathcal{T}S(i).\sigma, S'_i, spo'$ we get $\langle \mathcal{P}(i), \text{labels}(\text{sequence}_{spo}(S'_i)) \rangle$ $\Rightarrow (\mathcal{P}(i), \text{labels}(\text{sequence}_{spo}(S_i)).e'.\text{lab})$ $\Rightarrow (\mathcal{P}(i), \mathcal{T}S(i).\sigma.e'.\text{lab})$ $\Rightarrow \mathcal{T}S'(i).\sigma$

Hence the condition is preserved between MS' and $G'$.

(3) Condition to show: Whenever $\mathcal{W}'$ maps an event of $G'$ to a message in MS', then the location accessed and the written values match.

We know that the event to message mappings for existing events in $G.E$ and messages $M$ do not change.

$\forall e \in G'.E. e \neq e' \implies \mathcal{W}'(e) = \mathcal{W}(e)$

If $e = e'$ then $\mathcal{W}'(e') = m'$ and $e'.\text{loc} = m'.\text{loc} = x$ and $e'.\text{wval} = m'.\text{wval} = v$.

Hence $\mathcal{W}'$ preserves the condition.

(4) Condition to show: For all outstanding promises of threads ($T \setminus \{i\}$), there are corresponding write events in $G'$ that are po-after $S'$.
We know that for each thread \( j \neq i \) the set of promises are preserved from MS to MS', that is, 
\( \forall j \neq i. \; T S(j).P = T S'(j).P. \)
We also know that G satisfies this condition.
Hence the condition is preserved in G'.

(5) Condition to show: For every location \( \ell \) and thread \( j \), the thread view of \( \ell \) in the promise state MS' records the timestamp of the maximal write visible to the covered events in G' of thread \( j \).

Essentially we have to show
\[
\forall j, \ell. \; T S'(j).V(\ell) = \max\{W'(e).ts \mid e \in \text{dom}(W(\ell); G'; jf'; \; \text{shb}^O; \; \text{seco}^O; \; \text{shb}^O; [S'_j])}\]

\textbf{case} For \( j \neq i \) or \( j = i \land \ell \neq x \), it is trivial because \( T S'.V(\ell) = T S.V(\ell). \)

\textbf{case} For \( j = i \land \ell = x \),
following the promising semantics \( e' \in G.'W_x, W'(e') = m', m'.ts \) extends the view on \( x \) in thread \( i \), and hence \( T S(i).V(x) < T S'(i).V(x). \)
In this case \( e' \in S'_i \) and hence \( e' \in \text{dom}(W_x; G'.jf'; \; \text{shb}^O; \; \text{seco}^O; \; \text{shb}^O; [S'_j]) \) holds.
As a result,
\[ T S'(i).V(x) = m'.ts = \max\{W'(e).ts \mid e \in \text{dom}(W_x; G'.jf'; \; \text{shb}^O; \; \text{seco}^O; \; \text{shb}^O; [S'_j])\}. \]
Thus the relation holds between MS' and G'.

(6) Condition to show: The \( S' \) events in G' preserve coherence: \( \text{shb}^O; \; \text{seco}^O \) is irreflexive.

We know \( e' \in S' \) and let \( a \in S' \) such that \( (a, e') \in (\text{shb}^O; \; \text{seco}^O). \)

Hence following the definitions of \( \text{shb}^O, \; \text{seco}^O \), and from Lemma 3 (2) we know MS', \( T S'(a.tid).V(x) < MS'.T S'(e'.tid).V(x) \) as \( e' \in S_t. \)
As a result, \( (\text{shb}^O; \; \text{seco}^O) \) is irreflexive.

(7) Condition to show: The atomicity condition for update operations holds for \( S' \) events in G'.

We know that \[ G'.U \cap S' = [G.U \cap S] \text{ and } [G.U \cap S]; \text{fr; smo} = \emptyset \text{ holds}. \]
Assume there exists an update \( u \in G'.U \cap S' \), which reads from \( w \), such that \( \text{sfr'}(u, e') \) and \( \text{smo'}(e', u) \) holds.
By the definitions of \( \text{sfr'} \) and \( \text{smo'}, \; W'(w).ts < m'.ts < W'(u).ts. \)
But the promising semantics does not assign a timestamp in that range.
Hence a contradiction and \[ G'.U \cap S'; \text{fr'}; \text{smo'} = \emptyset \text{ holds}. \]

(8) Condition to show: The \( \text{sc} \) fences in G' are appropriately ordered by \( \text{sc}' \).

We know \[ [G.\text{Fsc}]; \text{shb} \cup \text{shb}; \text{seco}; \text{shb}; [G.\text{Fsc}] \subseteq \text{sc} \text{ holds in } G. \]

From definitions we know, \[ G'.\text{Fsc} = G.\text{Fsc}, \; \text{sc}' = \text{sc}, \; \text{shb} \subseteq \text{shb}', \; \text{seco} \subseteq \text{seco}' \text{.} \]
Consider \( a, b \) are two SC fences such that \( (a, b) \in [G.\text{Fsc}]; \text{shb} \cup \text{shb}; \text{seco}; \text{shb}; [G.\text{Fsc}] \text{, and sc}(a, b) \) holds.
In that case \( (a, b) \in (\text{shb} \cup \text{shb}'; \; \text{seco}'; \; \text{shb}') \) holds and \( \text{sc}'(a, b) \) holds.
To show \[ [G'.\text{Fsc}]; \text{shb} \cup \text{shb}; \text{seco}; \text{shb}; [G'.\text{Fsc}] \subseteq \text{sc}', \text{ we have to show } (b, a) \notin (\text{shb} \cup \text{shb}'; \; \text{seco}'; \; \text{shb}'). \text{ We show this by contradiction.} \]

Assume \( (b, a) \in (\text{shb} \cup \text{shb}'; \; \text{seco}'; \; \text{shb}'). \)
This is possible due to the relations created to/from event \( e' \).
Considering the relations in \( \text{shb} ', \; \text{sc} ', \; \text{seco} ', \) the incoming relations to event \( e' \) are \( \text{shb} ', \; \text{sfr} ', \; \text{smo} ' \) and the outgoing edges are \( \text{smo} '. \)
As there is no outgoing \( \text{sfr} \) edge from \( e' \), no new synchronization edge is created, that is, \( \text{ssw}' = \text{ssw}. \)
Thus a \( \text{smo}'(e', w) \) edge where \( w \) is a write event occurs in the \( (\text{shb} ' \cup \text{shb} ; \; \text{seco} '; \; \text{shb} ') \) path from \( b \) to \( a \).
In this case the path from \( b \) to \( a \) is \( (b, e') \in \text{shb} '; \; \text{seco}' \text{ and } (e', a) \in \text{smo} '; \; \text{seco}' \text{; shb}'. \)
We analyze the cases of \( (b, e') \in \text{shb} '; \; \text{seco}' \).
case \textit{shb}'(b, e'). 
In this case \textit{shb}(b, e) and \textit{spo}'(e, e') hold.
Hence MS_{T}S(b, t.id).V(x) ≤ MS_{T}'S(e, t.id).V(x) < MS_{T}'S(e', t.id).V(x).

- \textbf{case} \textit{shb'; seco}'(b, c) and \textit{smo}'(c, e'). 
  Hence \textit{shb; seco}(b, c) and \textit{smo}'(c, e') holds.
  So MS_{T}S(b, t.id).V(x) ≤ MS_{T}'S(c, t.id).V(x) < MS_{T}'S(e', t.id).V(x).
  Now we analyze \((e', a) \in \textit{smo}; \textit{seco}' \cup \textit{shb}'\).
  In this case there exist a write \(w \in S\) such that \textit{smo}'(e', w) and \((w, a) \in \textit{seco}'; \textit{shb}\) holds.
  Hence MS_{T}'S(e', t.id).V(x) < MS_{T}'S(w, t.id).V(x) ≤ MS_{T}'S(a, t.id).V(x).
  As a result, in all cases MS_{T}S(b, t.id).V(x) < MS_{T}'S(a, t.id).V(x) holds.
  However, we know that \textit{sc}(a, b) holds and therefore we have
  MS_{T}'S(a, t.id).V(x) ≤ MS_{T}S(b, t.id).V(x).
  This is a contradiction and hence \((b, a) \notin (\textit{shb'} \cup \textit{seco'} \cup \textit{shb}')\).
  As a result, \([G'.\mathcal{F}_{sc}]; \textit{shb}' \cup \textit{seco'} \cup \textit{shb}'; [G'.\mathcal{F}_{sc}] \subseteq \textit{sc}'\) holds.

(9) Condition to show: \textit{The behavior of MS'} matches that of the \(S'\) events in \(G'\).
Essentially we have to show, Behavior(MS') = Behavior(G', \(W', S'\)).
Following the definitions of Behavior(MS') and Behavior(G', \(W', S'\)); we know following cases for a location \(\ell\):

- \textbf{case} \(\ell \neq x\):
  The set of messages on \(\ell \neq x\) remains from MS to MS'.
  Hence in the promise machine Behavior|\(\ell\) (MS') = Behavior|\(\ell\) (MS) holds.
  Similarly Behavior|\(\ell\) (G', \(W', S'\)) = Behavior|\(\ell\) (G, \(W, S\)) holds in the event structure.
  We already know that Behavior|\(\ell\) (MS') \subseteq Behavior|\(\ell\) (G, \(W, S\)) holds for MS and G.
  As a result, Behavior|\(\ell\) (MS') = Behavior|\(\ell\) (G', \(W', S'\)).

- \textbf{case} \(\ell = x\):
  Let \(m\) be the message on \(x\) which results in the behavior of MS. In that case \(m.\text{loc} = x\),
  maxmsg(M \(\cup I\), TS(i).P.x) = \(m\), and let \(m.\text{wval} = v_1\). As a result, \((x, v_1) \in \text{Behavior}(MS)\).
  In this case there exists event \(e_1 \in G'.W_1 \cap S\) such that \(\mathcal{W}(e_1) = m, e_1.\text{loc} = x, e_1.\text{wval} = v_1,\) and \(\mathcal{T}(e_2.S, \text{mo}(e_1, e_2))\).
  Considering the new message is \(m'\), we know \(m' = \mathcal{W}'(e')\) and \(m'.\text{wval} = v\) holds.
  Comparing the \(m\) and \(m'\) we have two subcases:
  
  - \textbf{subcase} \(m.ts < m'.ts\).
    In this case maxmsg(M \(\cup I\), TS(i).P.x) = \(m'\) and Behavior|\(x\) (MS') = \{(x, v)\}.
    In the event structure \(G'\), \textit{mo}'(e_1, e') holds and hence Behavior|\(x\) (G', \(W', S'\)) = \{(x, v)\}.
  
  - \textbf{subcase} \(m.ts > m'.ts\).
    In this case maxmsg(M \(\cup I\), TS(i).P.x) = \(m'\) and Behavior|\(x\) (MS') = \{(x, v)\}.
    In the event structure \textit{mo}'(e', e_1) holds and hence Behavior|\(x\) (G', \(W', S'\)) = \{(x, v_1)\}.
    In both cases Behavior|\(x\) (G', \(W', S'\)) = Behavior|\(x\) (MS') holds.
  As a result, Behavior(G', \(W', S'\)) = Behavior(MS').

\textbf{Subcase} \(\exists e' \in (G.E_i \cup S_i). \text{dom}(G.p_w; \{e'\}) = S_0 \cup S_i \wedge e'.\text{lab} = St_o(x, v)\):
We take \(G' = G\) and let \(\mathcal{W}' \triangleq \mathcal{W}[e' \mapsto m']\).
Based on \(\mathcal{W}'\), we derive following definitions in MS'.

- \(S' \triangleq S \cup \{e'\}\)
• $\text{mo'} \triangleq \text{mo} \cup \{(a, e') \mid a \in G.W_x \land \varnothing(a) \neq \bot \land \varnothing'(a).ts < \varnothing'(e').ts\}$

• $\varnothing \{(e', a) \mid a \in G.W_x \land \varnothing(a) \neq \bot \land \varnothing'(e').ts < \varnothing'(a).ts\}$

• $\text{sc'} \triangleq \text{sc}$

• $\text{sro'} \triangleq (\text{sro} \cup \{(e, e') \mid e \in S_0 \cup S'_1\})^+$

Now we check whether $G' \sim_{\{1\}} (TS', S', M')$.

(1) Condition to show: $G'$ is consistent in weakest model.

$G'$ is consistent as $G$ is consistent.

(2) Condition to show: The local state of each thread in $MS'$ contains the program of that thread along with the sequence of covered events in $G'$ of that thread.

In this we have to show $\forall j. TS'(j).\sigma = \langle \mathcal{P}(j), \text{labels}(\text{sequence}_{\text{spo}}(S'_1))\rangle$.

We know that the relation holds between $MS$ and $G$.

Case For $j \neq i$, it is trivial because $TS'(j) = TS(j)$ holds from $MS$ to $MS'$ and $S'_j = S_j$ holds from $G$ to $G'$.

Case For $j = i$, we know $TS(i).\sigma = \langle \mathcal{P}(i), \text{labels}(\text{sequence}_{\text{spo}}(S_i))\rangle$.

Hence following the definition of $TS(i).\sigma, S'_i, \text{spo'}$ we get

$\langle \mathcal{P}(i), \text{labels}(\text{sequence}_{\text{spo}}(S'_1))\rangle$

$= \langle \mathcal{P}(i), \text{labels}(\text{sequence}_{\text{spo}}(S_i))\cdot e'.\text{lab} \rangle$

$= \langle \mathcal{P}(i), TS(i).\sigma\cdot e'.\text{lab} \rangle$

$= TS'(i).\sigma$

Hence the condition is preserved between $MS'$ and $G'$.

Note. This was same as the other scenario when we append a new $St_o(x, v)$.

(3) Condition to show: Whenever $\varnothing'$ maps an event of $G'$ to a message in $MS'$, then the location accessed and the written values match.

Case The event to message mappings for existing events in $G.E$ and messages $M$ do not change. Hence $\forall e \in G'.E. e \neq e' \implies \varnothing'(e) = \varnothing(e)$.

If $e = e'$ then $\varnothing'(e') = \text{wmsg}(\text{op}) = m'$ and $e'.\text{loc} = \text{wmsg}(\text{op}).\text{loc} = x$ and $e'.\text{wval} = m'.\text{wval} = v$.

Thus $\varnothing'$ preserves the condition between $MS'$ and $G'$.

(4) Condition to show: For all outstanding promises of threads ($T \setminus \{i\}$), there are corresponding write events in $G'$ that are po-after $S'$.

We know that for each thread $j \neq i$ the set of promises are preserved from $MS$ to $MS'$, that is, $\forall j \neq i. TS(j).P = TS'(j).P$.

We also know that $G$ satisfies this condition.

Hence the condition is preserved in $G'$.

Note. This was same as the other scenario when we append a new $St_o(x, v)$.

(5) Condition to show: For every location $\ell$ and thread $j$, the thread view of $\ell$ in the promise state $MS'$ records the timestamp of the maximal write visible to the covered events in $G'$ of thread $j$.

Essentially we have to show

$\forall j, \ell. TS'(j).V(\ell) = \max\{\varnothing'(e).ts \mid e \in \text{dom}([\mathcal{W}_1]; G'.j') \land \text{shb}'.(S'_j)\}$

For $j \neq i$ or $j = i \land \ell \neq x$, it is trivial because $TS'.V(\ell) = TS.V(\ell)$.

For $j = i \land \ell = x$, from the definition we know

(1) $TS(i).V(x) = \max\{\varnothing'(e).ts \mid e \in \text{dom}([\mathcal{W}_1]; G.j') \land \text{shb}'.(S'_1)\}$

(2) $TS'(i).V(x) = m'.ts$
(3) $\mathbb{W}'(e') = m'$ holds.
Following the promising semantics, we know $\mathcal{T}\mathcal{S}'(i).V(x)$ extends the thread view of $x$ from $\mathcal{T}\mathcal{S}(i).V(x)$ and $\mathcal{T}\mathcal{S}(i).V(x) < m'.ts$.
Hence following the construction, $\mathcal{T}\mathcal{S}'(i).V(x) = m'.ts = \max\{\mathbb{W}'(e).ts \mid e \in \text{dom}(\mathcal{W}_x; G'; jf'; \text{shb}'; \text{sc}'; \text{smo}'; S')\}$ holds.
Thus the relation holds between $\text{MS}'$ and $G'$.

(6) Condition to show: The $S'$ events in $G'$ preserve coherence: $\text{shb}'; \text{sec}''$ is irreflexive.
The argument is analogous to the case when we append a new $\text{St}_o(x, v)$.

(7) Condition to show: The atomicity condition for update operations holds for $\mathcal{T}\mathcal{S}'$ events in $G'$.
The argument is analogous to the case when we append a new $\text{St}_o(x, v)$.

(8) Condition to show: The $\text{sc}$ fences in $G'$ are appropriately ordered by $\text{sc}'$.
The argument is analogous to the case when we append a new $\text{St}_o(x, v)$.

(9) Condition to show: The behavior of $\text{MS}'$ matches that of the $\mathcal{T}\mathcal{S}'$ events in $G'$.
Essentially we have to show, $\text{Behavior}(\text{MS}') = \text{Behavior}(G', \mathbb{W}', S')$.
Following the definitions of $\text{Behavior}(\text{MS}')$ and $\text{Behavior}(G', \mathbb{W}', S')$; we know following cases for a location $\ell$:

- **case $\ell \neq x$**:
The set of messages on $\ell \neq x$ remains from MS to $\text{MS}'$.
Hence in the promise machine $\text{Behavior}_\ell(\text{MS}') = \text{Behavior}_\ell(\text{MS})$ holds.
Similarly $\text{Behavior}_\ell(G', \mathbb{W}', S') = \text{Behavior}_\ell(G, \mathbb{W}, S)$ holds in the event structure.
We already know that $\text{Behavior}_\ell(\text{MS}) = \text{Behavior}_\ell(G, \mathbb{W}, S)$ holds for MS and $G$.
As a result, $\text{Behavior}_\ell(\text{MS}') = \text{Behavior}_\ell(G', \mathbb{W}', S')$.

- **case $\ell = x$**:
Let $m$ be the message on $x$ which results in the behavior of MS. In that case $m.\text{loc} = x$,
$maxmsg(M \setminus \bigcup_j T\mathcal{S}(i).P, x) = m$, and let $m.\text{wval} = v_1$. As a result, $(x, v_1) \in \text{Behavior}(\text{MS})$.
In this case there exists event $e_1 \in G.\mathbb{W}_x \cap S$ such that $\mathbb{W}(e_1) = m, e_1.\text{loc} = x, e_1.\text{wval} = v_1$, and $\mathbb{W}(e_2) = S.\text{mo}(e_1, e_2)$.
Considering the new message is $m'$, we know $m' = \mathbb{W}'(e')$ and $m'.\text{wval} = v$ holds.
Comparing the $m$ and $m'$ we have two subcases:

  - **subcase $m.\text{ts} < m'.\text{ts}$**.
    In this case $maxmsg(M' \setminus \bigcup_j T\mathcal{S}'(i).P, x) = m'$ and $\text{Behavior}_x(\text{MS}') = \{(x, v)\}$.
    In the event structure $G'$, $\text{mo}'(e_1, e')$ holds and hence $\text{Behavior}_x(G', \mathbb{W}', S') = \{(x, v)\}$.

  - **subcase $m.\text{ts} > m'.\text{ts}$**.
    In this case $maxmsg(M' \setminus \bigcup_j T\mathcal{S}'(i).P, x) = maxmsg(M \setminus \bigcup_j T\mathcal{S}(i).P, x)$
    and $\text{Behavior}_x(\text{MS}') = \text{Behavior}_x(\text{MS}) = \{(x, v_1)\}$.
    In the event structure $\text{mo}'(e', e_1)$ holds and hence $\text{Behavior}_x(G', \mathbb{W}', S') = \text{Behavior}_x(G, \mathbb{W}, S) = \{(x, v_1)\}$.
    In both cases $\text{Behavior}_x(G', \mathbb{W}', S') = \text{Behavior}_x(\text{MS}')$ holds.
As a result, $\text{Behavior}(G', \mathbb{W}', S') = \text{Behavior}(\text{MS}')$.

Note. This was same as the other scenario when we append a new $\text{St}_o(x, v)$.

**Case Read** $\text{Ld}(a, x, v)$ reading from message $\text{wm} = (x : v@(-, i], R)$:
In the event structure we extend the event structure $G$ to $G'$. We extend the cover set $S_j$ as well as the relations ($\text{spo}$, $\text{srf}$, $\text{smo}$) to $S'_j$ along with the respective relations ($\text{spo}'$, $\text{srf}'$, $\text{smo}'$) by including an event $e'$ where

1. $\text{dom}(G.\text{po}; \{\{e'\}\}) = S_0 \cup S_i$. 
(2) \(e' \in S'_i \setminus S_i\), and
(3) \(\text{labels}(\text{sequence}_{G.po(S_i)})(e'.lab) \in P(i)\).

In this case the promise machine is updated as follows.
\[ M' = M, S' = S, \text{ and } TS' = TS[i \mapsto \langle \mathcal{P}(i), \text{labels}(\text{sequence}_{spo'(S'_i)}), V', TS(i).P \rangle] \] where \(V' = TS(i).V[x \mapsto \text{wm}.ts]\).

Now we do a case analysis on whether such an load event \(e'\) exists in \(G\) or we append a new event.

Subcase \(\#e' \in (G,E_i \setminus S_i). \text{ dom}(G.po; \{e'\}) = S_0 \cup S_i \wedge e'.lab = \text{Ld}_o(x, v) \wedge G.jf(\text{wm}, e')\) where \(\text{wm} = \mathcal{W}(\text{wm})\):

We create \(e'\) such that \(e'.lab = \text{Ld}_o(x, v)\) and append \(e'\) to event structure \(G\) to create \(G'\). In that case

- \(G'.E = G.E \cup \{e'\}\)
- \(G'.po = (G.po \cup \{(e, e') \mid e \in (S_i \cup S_0)\})^+\)
- \(G'.jf = G.jf \cup \{(\text{wm}, e') \mid \mathcal{W}(\text{wm}) = \text{wm} \land (S_0 \cup S'_i); G'.po^?; \{\{\text{wm}\}\}\)
- \(G'.ew = G.ew\)

Let: \(\mathcal{W}' \triangleq \mathcal{W}\).

Based on \(\mathcal{W}'\), we derive following definitions in \(MS'\):

- \(\mathcal{S}' \triangleq \mathcal{S} \cup \{e'\}\)
- \(\text{mo}' \triangleq \text{mo}\)
- \(\text{sc}' \triangleq \text{sc}\)
- \(\text{spo}' \triangleq (\text{spo} \cup \{(e, e') \mid e \in S_0 \cup S'_i\})^+\)
- \(\text{srf}' \triangleq \text{srf} \cup \{(w, e') \mid G'.rf(w, e') \land w \in \mathcal{S}\}\)

Now we check whether \(G' \sim_{\{i\}} (TS', S', M')\).

(1) Condition to show: \(G'\) is consistent in weakest model.

- (CF)
  We know \(G\) preserves (CF). Hence in \(G'\) we need to only consider the \(e'\).
  Assume there exists event \(e_1\) and \(e_2\) such that
  \(G'.hb(e_1, e'), G'.cf(e_1, e_2), G'.hb(e_2, e')\) hold.
  assert: \(e_1 \in \mathcal{S}\).
  We know \(G'.hb(e_1, e')\).
  Hence either \(G'.po(e_1, e')\) or \((e_1, e') \in G'.po^?; G'.swe; G.hb^?\).
  case \(G'.po(e_1, e')\). From the definitions \(e_1 \in \mathcal{S}\).
  case \((e_1, e') \in G'.po^?; G'.swe; G.hb^?\).
  Assume \(e_1 \notin \mathcal{S}\) and hence \(e_1 \in G.E \setminus \mathcal{S}\).
  All po-following events of \(e_1\) are in \(G.E \setminus \mathcal{S}\), that is, \(\text{dom}(\{e_1\}).G.po \in G.E \setminus \mathcal{S}\).
  However, from Lemma 4 we know that \(\text{dom}(G.swe) \subseteq \mathcal{S}\) and the events in \(G.E \setminus \mathcal{S}\) has no outgoing swe edge, that is, \(\text{dom}(G.swe) \notin (G.E \setminus \mathcal{S})\).
  Hence a contradiction and \(e_1 \notin \mathcal{S}\).
  assert: \(e_2 \notin \mathcal{S}\).
  Assume \(e_2 \in \mathcal{S}\).
  From the definition of \(\mathcal{S}\) it is conflict-free, that is, \(\mathcal{S} \cap G.cf = \emptyset\). Thus it is not possible and hence a contradiction.
  As a result, \(e_2 \notin \mathcal{S}\).
  Now we know that \(G'.hb(e_2, e')\) hold and thus \((e_2, e') \in G'.po^?; G'.swe; G'.hb^?\).
From Lemma 4 we know that $e_2$ has no $G'.po$ following event with outgoing $G'.swe$. Hence $G.po(e_2, e')$ holds.

In that case $G'.po(e_1, e'), G'.po(e_2, e'), G'.cf(e_1, e_2)$ result in a contradiction.

As a result, $G$ satisfies (CF).

• (CFJ) We know $G$ preserves (CFJ). Hence in $G$ we need to only consider the $G'.jf(w_m, e')$. Assume there exists event $e_1$ and $e_2$ such that $G'.hb(e_1, e'), G'.cf(e_1, e_2), G'.hb(e_2, w_m)$ hold.

**assert:** $e_1 \in \mathbb{S}$.

We know $G'.hb(e_1, e')$. Hence either $G'.po(e_1, e')$ or $(e_1, e') \in G'.po; G'.swe; G'.hb$.

**case** $(e_1, e') \in G'.po; G'.swe; G'.hb$. Assume $e_1 \notin \mathbb{S}$ and hence $e_1 \in G.E \setminus \mathbb{S}$.

In that case all po following events are in $G.E \setminus \mathbb{S}$, that is, $\text{codom}([\{e_1\}], G.po) \in G.E \setminus \mathbb{S}$.

However, from Lemma 4 we know that $\text{dom}(G.swe) \subseteq \mathbb{S}$ and the events in $G.E \setminus \mathbb{S}$ has no outgoing $swe$ edge, that is, $\text{dom}(G.swe) \notin (G.E \setminus \mathbb{S})$.

Hence a contradiction and $e_1 \in \mathbb{S}$.

**assert:** $e_2 \notin \mathbb{S}$.

Assume $e_2 \in \mathbb{S}$.

From the definition of $\mathbb{S}$ it is conflict-free, that is, $\mathbb{S} \cap G.cf = \emptyset$. Thus it is not possible and hence a contradiction.

As a result, $e_2 \notin \mathbb{S}$.

Now we know that $G'.hb(e_2, w_m)$ as well as $G.hb(e_2, w_m)$ hold and thus $(e_2, w_m) \in G'.po; G'.swe; G'.hb$.

From Lemma 4 we know that $e_2$ has no $G'.po$ following event with outgoing $G'.swe$. Hence $G.po(e_2, w_m)$ holds.

As a result, $e_1.tid = e_2.tid = w_m.tid$ holds.

However, from the definition of $G'.jf(w_m, e')$ we know that $G'.po(e_1, w_m)$ holds.

In that case $G'.po(e_1, w_m), G'.po(e_2, w_m), G'.cf(e_1, e_2)$ result in a contradiction.

As a result, $G$ satisfies (CFJ).

• (VISJ) We study the possible cases of $w_m$.

  - If $G'.po(w_m, e')$ then the condition holds as $(w_m, e') \notin G'.jfe$.
  
    - We will show that $G'$ satisfies (CFJ) constraint. Hence $w_m$ cannot be in conflict with $e'$, that is, $(w_m, e') \notin G'.cf$.
  
    - $w_m$ is in different thread and $G'.jfe(w_m, e')$ holds. We know that $G \sim (i)$ MS and the simulation rules ensures that there is no invisible event in the $(T \setminus \{i\})$ threads. Hence $w_m$ is a visible event in $G$ as well as in $G'$.

Considering the above mentioned cases $G'.jfe(w_m, e') \implies w_m \in \text{vis}(G')$ holds and $G'$ satisfies (VISJ) constraint.

• (ICF). We know $G$ satisfies constraint (ICF). Following the construction $e' \in G'.R$ and following the determinacy condition if $G'. \sim (e_1, e')$ then $e_1 \in \text{Ld}$. Thus $(e_1, e') \in (G'.R \times G'.R)$ and hence $G'$ satisfies (ICF).

• (ICFJ) From the construction we know there exists no $e_1$ such that $\text{imm}(cf)(e_1, e')$ and $G.Rf(\mathbb{W}^{-1}(w_m), e_1)$. Moreover, $G$ satisfies constraint (ICFJ). As a result, $G'$ satisfies (ICFJ).

• (COH) We know that $G$ satisfies (COH) constraint and hence $(G.hb; G.eco^?_{strong})$ is acyclic.

  We check if $(G'.hb; G'.eco^?_{strong})$ is acyclic.

The incoming edges to event $e'$ are $G'.hb$, $G'.rf$ and there is outgoing $G'.fr_{strong}$ edges.
If \((G'.\text{hb}; G'.\text{eco}\_\text{strong})\) forms a cycle then

(i) event \(e'\) is in the cycle.

(ii) \(G'.\text{fr}\_\text{strong}(e', w')\) is in the cycle where \(w'\) is some write on \(x\).

(iii) Either \(G'.\text{rf}(-, e')\) or \(G'.\text{hb}(-, e')\)

incoming edge is part of the \((G'.\text{hb}; G'.\text{eco}\_\text{strong})\) cycle.

Analyzing the cases on incoming edges to event \(e'\) the \((G'.\text{hb}; G'.\text{eco}\_\text{strong})\) cycle can be as follows.

- **case** \(G'.\text{rf}(-, e')\) completes the the \((G'.\text{hb}; G'.\text{eco}\_\text{strong})\) cycle.

The \(G'.\text{rf}(-, e')\) is either \(G'.\text{jf}(w_m, e')\) or there exists \(w_1\) such that

\[
G'.\text{ew}(w_m, w_1) \text{ and } (w_1, e') \in (G'.\text{ew}; G'.\text{jf}).
\]

Thus the cycle can be one of the followings ways.

(1) \(G'.\text{rf}(w_m, e'), G'.\text{fr}\_\text{strong}(e', w'), \text{ and } (w', w_m) \in (G'.\text{hb}; G'.\text{eco}\_\text{strong}).\)

(2) \(G'.\text{rf}(w_1, e'), G'.\text{fr}\_\text{strong}(e', w'), \text{ and } (w', w_1) \in (G'.\text{hb}; G'.\text{eco}\_\text{strong}).\)

Also note that \(G'.\text{fr}\_\text{strong}(e', w')\) implies either \(G'.\text{mo}\_\text{strong}(w_m, w')\) or \(G'.\text{mo}\_\text{strong}(w_1, w')\) already hold in \(G\).

Considering (1), (2), and possible reasons for \(G'.\text{fr}\_\text{strong}(e', w')\), we consider following subcases.

- **subcase**

(i) \(G'.\text{rf}(w_m, e'), G'.\text{fr}\_\text{strong}(e', w'), \text{ and } (w', w_m) \in (G'.\text{hb}; G'.\text{eco}\_\text{strong})\) is the cycle, and \(G'.\text{mo}\_\text{strong}(w_m, w')\)

(ii) \(G'.\text{rf}(w_1, e'), G'.\text{fr}\_\text{strong}(e', w'), \text{ and } (w', w_1) \in (G'.\text{hb}; G'.\text{eco}\_\text{strong})\) is the cycle, and \(G'.\text{mo}\_\text{strong}(w_1, w')\)

In case (i) \((w', w_m) \in (G'.\text{hb}; G'.\text{eco}\_\text{strong})\) implies

\((w', w_m) \in (G.\text{hb}; G'.\text{eco}\_\text{strong})\) holds in \(G\).

In that case \((w', w_m) \in (G.\text{hb}; G'.\text{eco}\_\text{strong})\) and \(G'.\text{mo}\_\text{strong}(w_m, w')\)

form a \((G.\text{hb}; G'.\text{eco}\_\text{strong})\) cycle in \(G\).

This is not possible as \((G.\text{hb}; G'.\text{eco}\_\text{strong})\) is acyclic and hence a contradiction.

Thus \((G'.\text{hb}; G'.\text{eco}\_\text{strong})\) is acyclic in this case.

Following the similar argument \((G'.\text{hb}; G'.\text{eco}\_\text{strong})\) is acyclic in case (ii).

- **subcase**

(i) \(G'.\text{rf}(w_m, e'), G'.\text{fr}\_\text{strong}(e', w'), \text{ and } (w', w_m) \in (G'.\text{hb}; G'.\text{eco}\_\text{strong})\) is the cycle, and \(G'.\text{mo}\_\text{strong}(w_m, w')\)

(ii) \(G'.\text{rf}(w_1, e'), G'.\text{fr}\_\text{strong}(e', w'), \text{ and } (w', w_1) \in (G'.\text{hb}; G'.\text{eco}\_\text{strong})\) is the cycle, and \(G'.\text{mo}\_\text{strong}(w_m, w')\)

In case (i) following Lemma 3,

(a) \((w', w_m) \in (G'.\text{hb}; G'.\text{eco}\_\text{strong})\) implies

\((w', w_m) \in (G.\text{hb}; G'.\text{eco}\_\text{strong})\) and in turn \(ts(w') < ts(w_m)\).

(b) \(G.\text{ew}(w_m, w_1)\) implies \(ts(w_m) = ts(w_1)\), and

(c) \(G'.\text{mo}\_\text{strong}(w_1, w')\) implies \(ts(w_1) < ts(w').\)

The combination of (a), (b), (c) contradicts the total order of timestamps.

Thus \((G'.\text{hb}; G'.\text{eco}\_\text{strong})\) is acyclic in this case.

Following the similar argument \((G'.\text{hb}; G'.\text{eco}\_\text{strong})\) is acyclic in case (ii).

- case \((G'.\text{hb}(-, e'))\) completes the \((G'.\text{hb}; G'.\text{eco}\_\text{strong})\) cycle.

In this case \(G'.\text{rf}(-, e')\) is not part of the \((G'.\text{hb}; G'.\text{eco}\_\text{strong})\) cycle.
Hence \((w', e') \in (G'_\text{hb}; G'_\text{eco}^2_{\text{strong}})\) and \(G'_\text{fr}^\text{strong}(e', w')\) form the \((G'_\text{hb}; G'_\text{eco}^2_{\text{strong}})\) cycle. 
\(G'_\text{fr}^\text{strong}(e', w')\) suggests two possibilities:

* **subcase** \(G'_\text{hb}(w_m, w')\).

  Following Lemma 3,
  1. \(ts(w_m) < ts(w')\).
  2. From \((w', e') \in (G'_\text{hb}; G'_\text{eco}^2_{\text{strong}})\) we know ts\((w') < ts(e')\).
  3. We also know \(G'_\text{rf}(w_m, e')\) implies ts\((w_m) = ts(e')\).
  4. However, \(G'_\text{fr}^\text{strong}(e', w')\) implies ts\((e') < ts(w')\).

  The combination of (a), (b), (c), (d) contradicts the total order of timestamps and hence \((G'_\text{hb}; G'_\text{eco}^2_{\text{strong}})\) is acyclic in this case.

* **subcase** \(G'_\text{hb}(w_1, w')\).

  Following Lemma 3,
  1. \(ts(w_1) < ts(w')\).
  2. From \((w', e') \in (G'_\text{hb}; G'_\text{eco}^2_{\text{strong}})\) we know ts\((w') < ts(e')\).
  3. We also know \(G'_\text{rf}(w_1, e')\) implies ts\((w_1) = ts(e')\).
  4. However, \(G'_\text{fr}^\text{strong}(e', w')\) implies ts\((e') < ts(w')\).

  The combination of (a), (b), (c), (d) contradicts the total order of timestamps and hence \((G'_\text{hb}; G'_\text{eco}^2_{\text{strong}})\) is acyclic in this case.

As a result, \(G'\) satisfies (COH).

Thus \(G'\) is consistent in weakest model.

(2) Condition to show: The local state of each thread in \(MS'\) contains the program of that thread along with the sequence of covered events in \(G'\) of that thread.

In this we have to show \(\forall j. TS'(j).\sigma = \langle P(j), labels(sequence_{spo'}(S'_j))\rangle\).

We know that the relation holds between MS and \(G\).

For \(j \neq i\), it is trivial because \(TS'(j) = TS(j)\) holds from MS to \(MS'\) and \(S'_j = S_j\) holds from \(G\) to \(G'\).

For \(j = i\), we know \(TS(i).\sigma = \langle P(i), labels(sequence_{spo}(S'_i))\rangle\).

Hence following the definition of \(TS(i).\sigma, S'_i, spo'\) we get
\[
\langle P(i), labels(sequence_{spo}(S'_j))\rangle = \langle P(i), labels(sequence_{spo}(S'_i))\rangle + e'.\text{lab}
\]
\[
= TS'(i).\sigma + e'\text{.lab}
\]
\[
= TS'(i).\sigma
\]

Hence the condition is preserved between \(MS'\) and \(G'\).

Note. This was same as the other scenario when we append a new \(St_e(x, v)\).

(3) Condition to show: Whenever \(W'\) maps an event of \(G'\) to a message in \(MS'\), then the location accessed and the written values match.

We know \(M' = M\) and \(W(e') = \bot\). Hence, if \(e \neq e'\) then \(W'(e) = \bot\). If \(e = e'\) then \(W'(e') = \bot\) and the assertion holds.

(4) Condition to show: For all outstanding promises of threads \(T \setminus \{i\}\), there are corresponding write events in \(G'\) that are po-after \(S'\).

We know that for each thread \(j \neq i\) the set of promises are preserved from MS to \(MS'\), that is, \(\forall j \neq i. TS(j).P = TS'(j).P\).

We also know that \(G\) satisfies this condition.

Hence the condition is preserved in \(G'\).
Condition to show: For every location \( \ell \) and thread \( j \), the thread view of \( \ell \) in the promise state \( MS' \) records the timestamp of the maximal write visible to the covered events in \( G' \) of thread \( j \).

Essentially we have to show
\[
\forall j, \ell. TS'(\ell).V(\ell) = \max\{W'(e).ts \mid e \in \text{dom}(\{W_\ell\}; G'.j\ell'; shb'^{\ell}; sc'^{\ell}; shb'; [S']_j)\}
\]
For \( j \neq i \) or \( j = i \land \ell \neq x \), it is trivial because \( TS'.V(\ell) = TS.V(\ell) \).
For \( j = i \land \ell = x \), we have to show
\[
TS'(i).V(x) = \max\{W'(e).ts \mid e \in \text{dom}(\{W_\ell\}; G'.j\ell'; shb'^{\ell}; sc'^{\ell}; shb'; [S']_j)\}.
\]
From the definitions we know
\[
\begin{align*}
(1) & \quad TS(i).V(x) = \max\{W(e).ts \mid e \in \text{dom}(\{W_\ell\}; G.j\ell'; shb'; sc'; shb'; [S_i])\} \\
(2) & \quad TS'(i).V(x) = ts(e') = \text{wm}.ts.
\end{align*}
\]
Following the promising semantics, we know \( TS'(i).V(x) \) extends the thread view of \( x \) from \( TS(i).V(x) \) by reading from \( \text{wm} \), and \( TS(i).V(x) \leq \text{wm}.ts \).
As a result,
\[
TS'(i).V(x) = \text{wm}.ts = \max\{W'(e).ts \mid e \in \text{dom}(\{W_\ell\}; G'.j\ell'; shb'^{\ell}; sc'^{\ell}; shb'; [S']_j)\}.
\]
Thus the condition is preserved between \( MS' \) and \( G' \).

Condition to show: The \( S' \) events in \( G' \) preserve coherence: \( shb'; seco'^{\ell} \) is irreflexive.

We know \( shb'; seco'^{\ell} \) is irreflexive in \( G \).
Let event \( a \in S' \) and assume \((a, e') \in \text{(shb'; seco'^{\ell})} \) and \((e', a) \in \text{(shb'; seco'^{\ell})}. \)
Following the definitions of \( shb', seco' \), and from Lemma 3 (2) we know
\[
MS_a.\text{TS}'(a.tid).V(x) \leq MS_a.\text{TS}'(e'.tid).V(x).
\]
However, the only outgoing edge from \( e' \) is \( fr' \) and from the definition we know \( sfr'(e', b) \) implies that \( MS_a.\text{TS}'(e'.tid).V(x) \leq MS_a.\text{TS}'(e'.tid).V(x) \).
Hence a contradiction and \( shb'; seco'^{\ell} \) is irreflexive.

Condition to show: The atomicity condition for update operations holds for \( S' \) events in \( G' \).

We know that \([G'.U \cap S'] = [G.U \cap S'] \) and \([G.U \cap S'] ; (sfr; smo) = 0 \) holds.
The \( e' \) does not introduce any \([G.U] ; G'.sfr' \) or \([G.U] ; G'.sfn' \) edge.
As a result, \([G'.U \cap S'] ; (sfr'; smo') = 0 \) holds.

Condition to show: The \( sc \) fences in \( G' \) are appropriately ordered by \( sc' \).

We know \([G.F_{sc}] ; shb \cup shb; seco; shb; [G.F_{sc}] \subseteq sc \) holds in \( G \).
From definitions we know, \([G'.F_{sc}] = G.F_{sc}, sc' = sc, shb \subseteq shb', seco \subseteq seco' \).
Consider \( a, b \) are two SC fences such that
\((a, b) \in [G'.F_{sc}] ; shb \cup shb; seco; shb; [G.F_{sc}] \), and \( se(a, b) \) holds.
In that case \((a, b) \in (shb' \cup shb'; seco'; shb') \) holds and \( sc'(a, b) \) holds.
To show \([G'.F_{sc}] ; shb' \cup shb'; seco'; shb'; [G.F_{sc}] \subseteq sc' \),
we have to show \((b, a) \notin (shb' \cup shb'; seco'; shb') \).
We show that by contradiction. Assume \((b, a) \in (shb' \cup shb'; seco'; shb') \).
This is possible due to the relations created to/from event \( e' \).
Considering the relations in \( shb' \) and \( seco' \), the incoming relations to event \( e' \) are \( shb' \) and \( sfr' \), and the outgoing edges are \( sfr' \).
Thus a \( sfr'(e', w) \) edge where \( w \) is a write event occurs in the \( (shb' \cup shb'; seco'; shb') \) path from \( b \) to \( a \).
In this case the path from \( b \) to \( a \) is \( (b, e') \in shb'; srf'^{\ell} \) and \((e', a) \in sfr'; seco'^{\ell}; shb' \).
It implies \((b, e') \in shb; srf'^{\ell} \) and \((e', a) \in sfr'; seco'; shb \).
In this case there exists \( w, w' \in G'.W_x \cap S \) such that \( srf'(w, e') \) and \( sfr'(e', w') \) holds.

However, from the definitions, in this case $\text{s}_\text{mo}(w, w')$ already holds and hence $(b, a) \in (\text{shb} \cup \text{shb'} \cup \text{seco} \cup \text{shb})$ already holds. This is a contradiction and hence $[G'.\text{f}_\text{sc}]; \text{shb'} \cup \text{shb'}; [G'.\text{f}_\text{sc}] \subseteq \text{sc'}$ holds.

(9) Condition to show: The behavior of $\text{MS}'$ matches that of the $\mathcal{S}'$ events in $G'$.

Essentially we have to show, $\text{Behavior}(\text{MS}') = \text{Behavior}(G', \mathcal{W}', \mathcal{S}')$.
We know $\text{Behavior}(\text{MS}) = \text{Behavior}(G, \mathcal{W}, \mathcal{S})$ holds.
From the definition we know, $\text{Behavior}(\text{MS}') = \text{Behavior}(\text{MS})$ and $\text{Behavior}(G', \mathcal{W}', \mathcal{S}') = \text{Behavior}(G, \mathcal{W}, \mathcal{S})$ hold.

As a result, $\text{Behavior}(\text{MS}') = \text{Behavior}(G', \mathcal{W}', \mathcal{S}')$ holds.

**Subcase** $\exists e' \in (G.E_i \setminus \mathcal{S}_i). \text{dom}(G.po; \{e'\}) = \mathcal{S}_0 \cup \mathcal{S}_i \land e'.\text{lab} = \text{Ld}_o(x, v) \land G.jf(\text{wm}, e')$ where $\text{wm} = \mathcal{W}(\text{wm})$:

We take $G' = G$ and let $\mathcal{W}' = \mathcal{W}$.

Based on $\mathcal{W}'$, we derive following definitions in $\text{MS}'$.

- $\mathcal{S}' \triangleq \mathcal{S} \cup \{e'\}$
- $\text{mo}' \triangleq \text{mo}$
- $\text{sc}' \triangleq \text{sc}$
- $\text{spo}' \triangleq \text{spo} \cup \{(e, e') \mid e \in \mathcal{S}_0 \cup \mathcal{S}_i\}^+$
- $\text{sr}' \triangleq \text{sr} \cup \{(w, e') \mid G'.rf(w, e') \land w \in \mathcal{S}\}$

Now we check whether $G' \sim_{\text{lin}} (\mathcal{T}\mathcal{S}', \mathcal{S}', \text{M}')$.

(1) Condition to show: $G'$ is consistent in weakest model.

We know $G'.E = G.E, G'.po = G.po, G'.jf = G.jf$, and $G$ is consistent. Hence $G'$ is also consistent.

(2) Condition to show: The local state of each thread in $\text{MS}'$ contains the program of that thread along with the sequence of covered events in $G'$ of that thread.

In this we have to show $\forall j. \mathcal{T}\mathcal{S}(j).\sigma = (\mathcal{P}(j), \text{labels}(\text{sequence}_{\text{spo}}(\mathcal{S}_j)))$.

We know that the relation holds between $\text{MS}$ and $G$.

For $j \neq i$, it is trivial because $\mathcal{T}\mathcal{S}(j) = \mathcal{T}\mathcal{S}(j)$ holds from $\text{MS}$ to $\mathcal{MS}'$ and $\mathcal{S}_j' = \mathcal{S}_j$ holds from $G$ to $G'$.

For $j = i$, we know $\mathcal{T}\mathcal{S}(i).\sigma = (\mathcal{P}(i), \text{labels}(\text{sequence}_{\text{spo}}(\mathcal{S}_i)))$.

Hence following the definition of $\mathcal{T}\mathcal{S}(i).\sigma$, $\mathcal{S}_i'$, spo' we get

- $(\mathcal{P}(i), \text{labels}(\text{sequence}_{\text{spo}}(\mathcal{S}_i)))$
- $(\mathcal{P}(i), \mathcal{T}\mathcal{S}(i).\sigma \cdot e'.\text{lab})$
- $\mathcal{T}\mathcal{S}'(i).\sigma$

Hence the condition is preserved between $\mathcal{MS}'$ and $G'$.

Note. This was same as the other scenario when we append a new $\text{St}_o(x, v)$ or $\text{Ld}_o(x, v)$.

(3) Condition to show: Whenever $\mathcal{W}'$ maps an event of $G'$ to a message in $\mathcal{MS}'$, then the location accessed and the written values match.

We know $\text{M}' = \text{M}$ and $\mathcal{W}(e') = \perp$. Hence, if $e \neq e'$ then $\mathcal{W}(e) = \perp(e)$. If $e = e'$ then $\mathcal{W}(e') = \perp$ and the assertion holds.

Note. This was same as the the scenario when we append a new $\text{Ld}_o(x, v)$.
(4) Condition to show: For all outstanding promises of threads \(T \setminus \{i\}\), there are corresponding write events in \(G'\) that are po-after \(S'\).

We know that for each thread \(j \neq i\) the set of promises are preserved from \(MS\) to \(MS'\), that is, \(\forall j \neq i. TS(j).P = TS'(j).P\).

We also know that \(G\) satisfies this condition.

Hence the condition is preserved in \(G'\).

Note. This was same as the other scenario when we append a new \(St_s(x, v)\) or \(Ld_d(x, v)\).

(5) Condition to show: For every location \(\ell\) and thread \(j\), the thread view of \(\ell\) in the promise state \(MS'\) records the timestamp of the maximal write visible to the covered events in \(G'\) of thread \(j\).

The argument is analogous to the case when we append a new \(Ld_d(x, v)\).

(6) Condition to show: The \(S'\) events in \(G'\) preserve coherence: \(shb'\); \(seco'\) is irreflexive.

The argument is analogous to the case when we append a new \(Ld_d(x, v)\).

(7) Condition to show: The atomicity condition for update operations holds for \(S'\) events in \(G'\).

The argument is analogous to the case when we append a new \(Ld_d(x, v)\).

(8) Condition to show: The \(sc\) fences in \(G'\) are appropriately ordered by \(sc'\).

The argument is analogous to the case when we append a new \(Ld_d(x, v)\).

(9) Condition to show: The behavior of \(MS'\) matches that of the \(S'\) events in \(G'\).

Essentially we have to show, \(Behavior(MS') = Behavior(G', \bar{W}', S')\).

We know \(Behavior(MS) = Behavior(G, \bar{W}, \bar{S})\) holds.

We have \(Behavior(MS') = Behavior(MS)\) and \(Behavior(G', \bar{W}', S') = Behavior(G, \bar{W}, \bar{S})\) by definition. As a result, \(Behavior(MS') = Behavior(G', \bar{W}', \bar{S}')\) holds.

**Case Update** \(U(o, x, v, v')\) reading from message \(wm = (x : v@(-, t], R)\) and creating message \(m' = (x : v'@[t', t], R')\):

In the event structure we extend the event structure \(G\) to \(G'\). We extend the cover set \(S_i\) as well as the relations \(spo, srf, smo\) to \(S'_i\) along with the respective relations \(spo', srf', smo')\) by including an event \(e'\) where

1. \(dom(G.p_o;\{(e')\}) = S_0 \cup S_i\),
2. \(e' \in S'_i \setminus S_i\), and
3. \(labels(sequence_{G.p_o(S_i)})(e'.lab) \in \bar{P}(i)\).

In this case the promise machine is updated as follows.

\[M' = M \uplus \{m'\}, S' = S, \text{ and } TS' = TS[i \mapsto ((\bar{P}(i), labels(sequence_{spo(S'_i)}), V'), TS(i).P)]\]

where \(V' = \bar{TS}(i).V[x \mapsto m'.ts]\).

Now we do a case analysis on whether such an update event \(e'\) exists in \(G\) or we append a new event.

**Subcase** \(\forall e' \in (G.E_i \setminus S_i)\). \(dom(G.p_o;\{(e')\}) = S_0 \cup S_i \land e'.lab = U(o, x, v, v') \land G.rf(w_m, e')\) where \(\bar{W}(w_m) = wm\):

We create \(e'\) such that \(e'.lab = U(o, x, v, v')\) and append \(e'\) to event structure \(G\) to create \(G'\). In that case

- \(G'.E = G.E \uplus \{e'\}\)
- \(G'.p_o = (G.p_o \cup \{(e, e') \mid e \in (S_i \cup \bar{S}_o)\})^+\)
- \(G'.j_f = G.j_f \uplus \{(w_m, e') \mid \bar{W}(w_m) = wm \land [S_0 \cup \bar{S}_i]; G'.p_o; \{(w_m)\}]\)
- \(G'.e_w = G.e_w\)

Let: \(\bar{W}' = \bar{W}[e' \mapsto m']\), and Based on \(\bar{W}'\), we derive following definitions in \(MS'\).
Grounding Thin-Air Reads with Event Structures

We consider the following subcases.

1. Condition to show: $G'$ is consistent in weakest model.
   - (CF) and (CFJ) constraints are preserved in $G'$. The arguments are analogous to the scenario when we append a new $\operatorname{Ld}_0(x, v)$.
   - (VISJ) We study the possible cases of $w_m$.
     - If $G'.\operatorname{po}(w_m, e')$ then the condition holds as $(w_m, e') \notin G'.\operatorname{rfe}$.
     - We will show that $G'$ satisfies (CFJ) constraint. Hence $w_m$ cannot be in conflict with $e'$, that is, $(w_m, e') \notin G'.\operatorname{cf}$.
     - $w_m$ is in different thread and $G'.\operatorname{jfe}(w_m, e')$ holds. We know that $G \sim \{1\} MS$ and the simulation rules ensure that there is no invisible event in the $(T \setminus \{1\})$ threads. Hence $w_m$ is a visible event in $G$ as well as in $G'$.
   - Considering the above mentioned cases $G'.\operatorname{jfe}(w_m, e') \implies w_m \in \operatorname{vis}(G')$ holds and $G'$ satisfies (VISJ) constraint.
   - Note. This was same as the other scenario when we append a new $\operatorname{Ld}_0(x, v)$.
   - (ICF) We know $G$ satisfies constraint (ICF). Following the construction $e' \in G'.\mathcal{R}$ and following the determinacy condition if $G'.\sim (e_1, e')$ then $e_1 \in \operatorname{Ld}$ or $e_1 \in \operatorname{U}$. Thus $(e_1, e') \in (G'.\mathcal{R} \times G'.\mathcal{R})$ and hence $G'$ satisfies (ICF).
   - Note. This was same as the other scenario when we append a new $\operatorname{Ld}_0(x, v)$.
   - (ICFJ) From the construction we know there exists no $e_1$ such that $\operatorname{imm}(\operatorname{cf})(e_1, e')$ and $G'.\operatorname{rf}(\varphi^{-1}(w_m), e_1)$. Moreover, $G$ satisfies constraint (ICFJ). As a result, $G'$ satisfies (ICF).
   - (COH) We know that $G$ satisfies (COH) constraint and hence $(G'.\operatorname{hb}; G'.\operatorname{eco}^{2}_{\operatorname{strong}})$ is acyclic.

We check if $(G'.\operatorname{hb}; G'.\operatorname{eco}^{2}_{\operatorname{strong}})$ is acyclic.

The incoming edges to event $e'$ are $G'.\operatorname{hb}, G'.\operatorname{jf}$ and there is an outgoing $G'.\operatorname{fr}^{\operatorname{strong}}$ edges.
If $(G'.\operatorname{hb}; G'.\operatorname{eco}^{2}_{\operatorname{strong}})$ forms a cycle then
(i) event $e'$ is in the cycle.
(ii) $G'.\operatorname{fr}^{\operatorname{strong}}(e', w')$ is in the cycle where $w'$ is some write on $x$.
(iii) Either $G'.\operatorname{rf}(-, e')$ or $G'.\operatorname{hb}(-, e')$ incoming edge is part of the $(G'.\operatorname{hb}; G'.\operatorname{eco}^{2}_{\operatorname{strong}})$ cycle.

Analyzing the cases on incoming edges to event $e'$ the $(G'.\operatorname{hb}; G'.\operatorname{eco}^{2}_{\operatorname{strong}})$ cycle can be as follows.
- **case** $G'.\operatorname{rf}(-, e')$ completes the the $(G'.\operatorname{hb}; G'.\operatorname{eco}^{2}_{\operatorname{strong}})$ cycle.

   The $G'.\operatorname{rf}(-, e')$ is either $G'.\operatorname{jf}(w_m, e')$ or there exists $w_1$ such that $G'.\operatorname{ew}(w_m, w_1)$ and $(w_1, e') \in (G'.\operatorname{ew}; G'.\operatorname{jf})$.
   - Thus the cycle can be one of the following ways.
     1. $G'.\operatorname{rf}(w_m, e'), G'.\operatorname{fr}^{\operatorname{strong}}(e', w'),$ and $(w', w_m) \in (G'.\operatorname{hb}; G'.\operatorname{eco}^{2}_{\operatorname{strong}})$.
     2. $G'.\operatorname{rf}(w_1, e'), G'.\operatorname{fr}^{\operatorname{strong}}(e', w'),$ and $(w', w_1) \in (G'.\operatorname{hb}; G'.\operatorname{eco}^{2}_{\operatorname{strong}})$.
   - Also note that $G'.\operatorname{fr}^{\operatorname{strong}}(e', w')$ implies either $G.\operatorname{mo}^{\operatorname{strong}}(w_m, w')$ or $G.\operatorname{mo}^{\operatorname{strong}}(w_1, w')$ already hold in $G$.
   - Considering (1), (2), and possible reasons for $G'.\operatorname{fr}^{\operatorname{strong}}(e', w')$, we consider following subcases.
* subcase
  (i) \( G'.rf(w_m, e'), G'.fr_{\text{strong}}(e', w'), \) and \((w', w_m) \in (G'.hb; G'.eco_{\text{strong}}')\) is the cycle, and \( G.mo_{\text{strong}}(w_m, w') \)
  (ii) \( G'.rf(w_1, e'), G'.fr_{\text{strong}}(e', w'), \) and \((w', w_1) \in (G'.hb; G'.eco_{\text{strong}}')\)
  is the cycle, and \( G.mo_{\text{strong}}(w_1, w') \)

In case (i) \((w', w_m) \in (G'.hb; G'.eco_{\text{strong}}')\) implies
\((w', w_m) \in (G.hb; G.eco_{\text{strong}}')\) holds in \( G \).
In that case \((w', w_m) \in (G.hb; G.eco_{\text{strong}}')\) and \( G.mo_{\text{strong}}(w_m, w') \) forms a
\((G.hb; G.eco_{\text{strong}}')\) cycle in \( G \).
This is not possible as \((G.hb; G.eco_{\text{strong}}')\) is acyclic and hence a contradiction.
Thus \((G'.hb; G'.eco_{\text{strong}}')\) is acyclic in this case.
Following the similar argument \((G'.hb; G'.eco_{\text{strong}}')\) is acyclic in case (ii).

* subcase
  (i) \( G'.rf(w_m, e'), G'.fr_{\text{strong}}(e', w'), \) and \((w', w_m) \in (G'.hb; G'.eco_{\text{strong}}')\) is the cycle, and \( G.mo_{\text{strong}}(w_1, w') \)
  (ii) \( G'.rf(w_1, e'), G'.fr_{\text{strong}}(e', w'), \) and \((w', w_1) \in (G'.hb; G'.eco_{\text{strong}}')\)
  is the cycle, and \( G.mo_{\text{strong}}(w_m, w') \)

In case (i) following Lemma 3,
(a) \((w', w_m) \in (G'.hb; G'.eco_{\text{strong}}')\) implies
\((w', w_m) \in (G.hb; G.eco_{\text{strong}}')\) and hence \( ts(w') < ts(w_m) \),
(b) \( G.ew(w_m, w_1) \) implies \( ts(w_m) = ts(w_1) \), and
(c) \( G.mo_{\text{strong}}(w_1, w') \) implies \( ts(w_1) < ts(w'). \)

The combination of (a), (b), (c) contradicts the total order of timestamps.
Thus \((G'.hb; G'.eco_{\text{strong}}')\) is acyclic in this case.
Following the similar argument \((G'.hb; G'.eco_{\text{strong}}')\) is acyclic in case (ii).

- case \((G'.hb; (−, e'))\) completes the \((G'.hb; G'.eco_{\text{strong}}')\) cycle.
In this case \( G'.rf(−, e') \) is not part of the \((G'.hb; G'.eco_{\text{strong}}')\) cycle.
Hence \((w', e') \in (G'.hb; G'.eco_{\text{strong}}')\) and \( G'.fr_{\text{strong}}(e', w') \)
forms the \((G'.hb; G'.eco_{\text{strong}}')\) cycle.
\( G'.fr_{\text{strong}}(e', w') \) suggests two possibilities:

  * subcase \( G'.hb(w_m, w') \).

Following Lemma 3,
(a) \( ts(w_m) < ts(w') \).
(b) From \((w', e') \in (G'.hb; G'.eco_{\text{strong}}')\) we know \( ts(w') < ts(e') \).
(c) We also know \( G'.jf(w_m, e') \) implies \( ts(w_m) < ts(e') \).
(d) However, \( G'.fr_{\text{strong}}(e', w') \) implies \( ts(e') < ts(w') \).

The combination of (a), (b), (c), (d) contradicts the total order of timestamps and hence
\((G'.hb; G'.eco_{\text{strong}}')\) is acyclic in this case.

  * subcase \( G'.hb(w_1, w') \).

Following Lemma 3,
(a) \( ts(w_1) < ts(w') \).
(b) From \((w', e') \in (G'.hb; G'.eco_{\text{strong}}')\) we know \( ts(w') < ts(e') \).
(c) We also know \( G'.rf(w_1, e') \) implies \( ts(w_1) = ts(e') \).
(d) However, \( G'.fr_{\text{strong}}(e', w') \) implies \( ts(e') < ts(w') \).
The combination of (a), (b), (c), (d) contradicts the total order of timestamps and hence \((G', \text{hb}; G', \text{eco}_{\text{strong}}')\) is acyclic in this case.

As a result, \(G'\) satisfies (COH).

Thus \(G'\) is consistent in weakest model.

(2) Condition to show: The local state of each thread in \(MS'\) contains the program of that thread along with the sequence of covered events in \(G'\) of that thread.

In this we have to show \(\forall j. TS'(j).\sigma = \langle P(j), \text{labels}(\text{sequence}_{\text{spo}}(S'_j))\rangle\).

We know that the relation holds between MS and G.

For \(j \neq i\), it is trivial because \(TS'(j) = TS(j)\) holds from MS to \(MS'\) and \(S'_j = S_j\) holds from \(G\) to \(G'\).

For \(j = i\), we know \(TS(i).\sigma = \langle P(i), \text{labels}(\text{sequence}_{\text{spo}}(S_i))\rangle\).

Hence following the definition of \(TS(i).\sigma, S'_i\), spo we get
\[
\langle P(i), \text{labels}(\text{sequence}_{\text{spo}}(S'_i))\rangle = \langle P(i), TS(i).\sigma.e'.lab\rangle
\]
\[
= TS'(i).\sigma
\]
Hence the condition is preserved between \(MS'\) and \(G'\).

Note. This was similar to the other scenario when we append a new \(St_o(x, v)\).

(3) Condition to show: Whenever \(W'\) maps an event of \(G'\) to a message in \(MS'\), then the location accessed and the written values match.

We know that the event to message mappings for existing events in \(G\). E and messages M do not change.

\[
\forall e \in G'.\text{E}. e \neq e' \implies W'(e) = W(e)
\]

If \(e = e'\) then \(W'(e') = m'\) and \(e'.\text{loc} = m'.\text{loc} = x\) and \(e'.\text{wval} = m'.\text{wval} = v\).

Hence \(W'\) preserves the condition.

Note. This was similar to the other scenario when we append a new \(St_o(x, v)\).

(4) Condition to show: For all outstanding promises of threads \((T \setminus \{i\})\), there are corresponding write events in \(G'\) that are po-after \(S'_i\).

We know that for each thread \(j \neq i\) the set of promises are preserved from MS to \(MS'\), that is,
\[
\forall j \neq i. TS(j).P = TS'(j).P
\]

We also know that \(G\) satisfies this condition.

Hence the condition is preserved in \(G'\).

Note. This was similar to the other scenario when we append a new \(St_o(x, v)\).

(5) Condition to show: For every location \(\ell\) and thread \(j\), the thread view of \(\ell\) in the promise state \(MS'\) records the timestamp of the maximal write visible to the covered events in \(G'\) of thread \(j\).

Essentially we have to show
\[
\forall j, \ell. TS'(j).V(\ell) = \max\{W'(e).ts \mid e \in \text{dom}(W'_\ell); G'.jf^j; \text{shb}^j ; \text{sc}^j; \text{shb}^j; [S'_j]\}
\]

For \(j \neq i\) or \(j = i \land \ell \neq x\), it is trivial because \(TS'.V(\ell) = TS.V(\ell)\).

For \(j = i \land \ell = x\), from the definition we know
\[
TS(i).V(x) = \max\{W(e).ts \mid e \in \text{dom}(W'_x); G.jf^i; \text{shb}^i; \text{sc}^i; \text{shb}^i; [S_i]\}
\]

Following the promising semantics, we know \(TS'(i).V(x)\) extends the thread view of \(x\) from \(TS(i).V(x)\) by reading from \(wm\), and hence \(TS'(i).V(x) < \text{wm.ts}\).

Moreover, following the semantics of update operation in promise machine \(\text{wm.ts} < m'.ts\).

Hence following the construction,
\[ TS'(i).V(x) = m'.ts = \max(\bar{\mathbb{W}}(e).ts \mid e \in \text{dom}(\{W_x\}; G'.jf'; shb'^i; \text{sc}'^i; shb'^{\mathbb{S}'^i}; \{S'\})). \]

Thus the condition is preserved between MS' and G'.

(6) Condition to show: The \( \mathbb{S}' \) events in G' preserve coherence: \( \text{shb}'^i; \text{seco}'^i \) is irreflexive.

The argument is analogous to the case when we append a new \( \text{St}_o(x, v) \).

(7) Condition to show: The atomicity condition for update operations holds for \( \mathbb{S}' \) events in G'.

Assume \( [G'.U \cap \mathbb{S}']; (\text{sfr}'; \text{smo}') \neq \emptyset \).

We know that \( [G.U \cap \mathbb{S}]; (\text{sfr}; \text{smo}) = \emptyset \) holds.

Hence \( e' \) is involved in atomicity violation. In that case two possibilities as follows:

• **case** There exists an update \( u \in (G.U_x \cap \mathbb{S}) \) such that \( \text{sfr}(u, e') \) and \( \text{smo}(e', u) \) holds.

Assume \( u \) reads from \( w_1 \), that is, \( \text{srf}(w_1, u) \).

\( \text{sfr}'(w_1, e') \) implies \( \text{mo}(w_1, e') \) holds.

\( \text{mo}'(w_1, e') \) implies \( \mathbb{W}'(w_1).ts < \mathbb{W}'(e').ts \).

However, \( \text{srf}'(w_1, u) \) implies \( \mathbb{W}'(w_1).ts < \mathbb{W}'(u).ts \) and there is no write on \( x \) in the time range \( (\mathbb{W}'(w_1).ts, \mathbb{W}'(u).ts) \), that is, \( \exists w' \in \mathbb{S}' \cap G'.W_x. \mathbb{W}'(w_1).ts < \mathbb{W}'(w').ts < \mathbb{W}'(u).ts \).

As a result, \( \mathbb{W}'(w_1).ts < \mathbb{W}'(e').ts < \mathbb{W}'(u).ts \) is not possible and hence \( \mathbb{W}'(u).ts < \mathbb{W}'(e').ts \) which implies \( \text{smo}'(u, e') \).

\( \text{smo}'(u, e') \) and \( \text{smo}'(e', u) \) both cannot hold.

Hence a contradiction and in this case atomicity holds in \( \mathbb{S}' \) events in G'.

• **case** There exists a write \( w' \in (G'.W_x \cap \mathbb{S}) \) such that \( \text{sfr}'(e', w') \) and \( \text{smo}'(w', e') \) hold.

\( \text{sfr}'(e', w') \) implies \( \text{mo}'(w, w') \), that is, \( \mathbb{W}'(w).ts < \mathbb{W}'(w').ts \).

However, \( \text{srf}'(w, e') \) implies \( \mathbb{W}'(w).ts < \mathbb{W}'(e').ts \) and there is no write on \( x \) in the time range \( (\mathbb{W}'(w).ts, \mathbb{W}'(e').ts) \), that is, \( \exists w' \in (G'.W_x \cap \mathbb{S}). \mathbb{W}'(w).ts < \mathbb{W}'(w').ts < \mathbb{W}'(e').ts \).

As a result, neither \( \mathbb{W}'(w).ts < \mathbb{W}'(e').ts < \mathbb{W}'(e').ts \) is not possible and hence \( \mathbb{W}'(e').ts < \mathbb{W}'(w').ts \) which implies \( \text{smo}'(e', w') \).

\( \text{smo}'(e', w') \) and \( \text{smo}'(w', e') \) both cannot hold.

Hence a contradiction and in this case atomicity holds in \( \mathbb{S}' \) events in G'.

(8) Condition to show: The \( \text{sc} \) fences in G' are appropriately ordered by \( \text{sc}' \).

We know \( [G.F_{sc}]; \text{shb} \cup \text{shb}; \text{seco}; \text{shb}; [G.F_{sc}]] \subseteq \text{sc} \) holds in G.

From definitions we know, \( G'.F_{sc} = G.F_{sc}; \text{sc}' = \text{sc}, \text{shb} \subseteq \text{shb}', \text{seco} \subseteq \text{seco}' \).

Consider \( a, b \) are two SC fences such that \( (a, b) \in [G.F_{sc}]; \text{shb} \cup \text{shb}; \text{seco}; \text{shb}; [G.F_{sc}] \subseteq \text{sc}' \), and \( \text{sc}(a, b) \) holds.

In that case \( (a, b) \in (\text{shb}' \cup \text{shb}' \cup \text{seco}' \cup \text{shb}') \) holds and \( \text{sc}'(a, b) \) holds.

To show \( [G'.F_{sc}]; \text{shb} \cup \text{shb}; \text{seco}; \text{shb}; [G'.F_{sc}] \subseteq \text{sc}' \),

we have to show \( (b, a) \notin (\text{shb} \cup \text{shb}; \text{seco}; \text{shb}) \).

We show that by contradiction. Assume \( (b, a) \in (\text{shb} \cup \text{shb}; \text{seco}; \text{shb}) \).

This is possible due to the relations created to/from event \( e' \).

Considering the relations in \( \text{shb}' \) and \( \text{seco}' \), the incoming relations to event \( e' \) are \( \text{shb}', \text{sfr}' \), \( \text{smo}' \) and the outgoing edges are \( \text{sfr}', \text{smo}' \).

Since \( e' \) is an update, for a write event \( w_1 \), relation \( \text{sfr}'(u, w_1) \) implies \( \text{smo}'(u, w_1) \).

Hence we consider only \( \text{smo}' \) as outgoing edge.

In this case the path from \( b \) to \( a \) is \( (b, e') \in \text{shb}'; \text{seco}' \) and \( (e', a) \in \text{smo}', \text{seco}' \cup \text{shb}' \).

As there is no outgoing \( \text{sfr} \) edge from \( e' \), no new synchronization edge is created, that is, \( \text{ssw}' = \text{ssw} \).

We analyze the cases of \( (b, e') \in \text{shb}' \cup \text{seco}' \).

In this case there exists some event $c$ such that

- $shb'(b, e')$.

Two possible subcases:

- **subcase** In this case $shb(b, e)$ and $spo'(e, e')$ holds.

  So $MS_b \cdot TS(b, tid).V(x) \leq MS_{e'}.TS(e, tid).V(x) < MS_{e'}.TS(e', tid).V(x)$.

- **subcase** $shb(b, c)$ and $ssw'(c, e')$ holds.

  Hence $MS_b \cdot TS(b, tid).V(x) \leq MS_{e'}.TS(e, tid).V(x)$ holds.

Moreover, consider the cases of $ssw'$, following from Lemma 3, we can show that $MS_{e'}.TS(e, tid).V(x) < MS_{e'}.TS(e', tid).V(x)$ holds.

Considering both subcases $MS_b \cdot TS(b, tid).V(x) < MS_{e'}.TS(e', tid).V(x)$ holds.

- $shb'; seco'(b, c)$ and $srf'(c, e')$.

  Hence $shb; seco(b, c)$ and $srf'(c, e')$ holds.

  As a result, following promising semantics, $MS_b \cdot TS(b, tid).V(x) \leq MS_{e'}.TS(e, tid).V(x) < MS_{e'}.TS(e', tid).V(x)$.

- $shb'; seco'(b, c)$ and $smo'(c, e')$.

  Hence $shb; seco(b, c)$ and $smo'(c, e')$ holds.

  As a result, following promising semantics, $MS_b \cdot TS(b, tid).V(x) \leq MS_{e'}.TS(e, tid).V(x) < MS_{e'}.TS(e', tid).V(x)$.

Now we analyze $(e', a) \in smo'; seco'; shb'$.

In this case there exist a $w \in S$ such that $smo'(e', w)$ and $(w, a) \in seco'; shb$ holds.

Hence $MS_{e'}.TS(e', tid).V(x) \leq MS_{w}.TS(w, tid).V(x) \leq MS_{a}.TS(a, tid).V(x)$.

As a result, in all cases $MS_b \cdot TS(b, tid).V(x) < MS_{e'}.TS(e, tid).V(x)$ holds.

However, we know that $sc(a, b)$ holds and hence $MS_a.V \leq MS_b.V$.

This is a contradiction and hence $(b, a) \notin (shb' \cup shb'; seco'; shb')$.

As a result, $[G'.F_{sc}]; shb' \cup shb'; seco'; shb'; [G'.F_{sc}] \subseteq sc'$ holds.

(9) Condition to show: The behavior of $MS'$ matches that of the $S'$ events in $G'$.

The argument is analogous to the case when we append a new $St_o(x, v)$.

**Subcase** $\exists e' \in (G.E_i \setminus S_i). \text{dom}(G.po; \{e'\}) = S_0 \cup S_i \wedge e'.lab = U(a, x, v, v') \wedge G.jf(w_m, e')$ where $\text{wm} = \mathbb{W}(w_m)$.

We take $G' = G$ and let $\mathbb{W'} = \mathbb{W}[e' \mapsto m']$.

Based on $\mathbb{W}'$, we derive the following definitions in $MS'$.

- $S' \triangleq S \cup \{e'\}$
- $mo' \triangleq mo \cup \{(a, e') \mid a \in G' \cdot W_x \wedge \mathbb{W}(a) \not\perp \wedge \mathbb{W'}(a).ts < \mathbb{W}'(e').ts\}$
- $sc' \triangleq sc$
- $spo' \triangleq (spo \cup \{(e, e') \mid e \in S_0 \cup S'_i\})^+$
- $srf' \triangleq srf \cup \{(w, e') \mid G'.rf(w, e') \wedge w \in S\}$

Now we check whether $G' \sim_{\mathbb{W}'} (TS', S', M')$.

(1) Condition to show: $G'$ is consistent in weakest model.
We know \(G'.E = G.E, G'.po = G.po, G'.jf = G.jf\), and \(G\) is consistent. Hence \(G'\) is also consistent in weakest model.

(2) Condition to show: The local state of each thread in \(MS'\) contains the program of that thread along with the sequence of covered events in \(G'\) of that thread.

In this we have to show \(\forall j. TS'(j).\sigma = \langle P(j), labels(sequence_{spo}(S'_j))\rangle\).

We know that the relation holds between \(MS\) and \(G\).

For \(j \neq i\), it is trivial because \(TS'(j) = TS(j)\) holds from \(MS\) to \(MS'\) and \(S'_j = S_j\) holds from \(G\) to \(G'\).

For \(j = i\), we know \(TS(i).\sigma = \langle P(i), labels(sequence_{spo}(S_i))\rangle\).

Hence following the definition of \(TS(i).\sigma, S'_i, spo\) we get\n\[
\langle P(i), labels(sequence_{spo}(S'_i))\rangle = \langle P(i), labels(sequence_{spo}(S_i))\rangle - e'.lab
\]
\[
= TS'(i).\sigma - e'.lab
\]
Hence the condition is preserved between \(MS'\) and \(G'\).

Note. This was same as the other scenario when we append a new \(St_s(x, v)\).

(3) Condition to show: Whenever \(\mathcal{H}'\) maps an event of \(G'\) to a message in \(MS'\), then the location accessed and the written values match.

The event to message mappings for existing events in \(G.E\) and messages \(M\) do not change.

\[
\forall e \in G'.E. e \neq e' \implies \mathcal{H}'(e) = \mathcal{H}(e)
\]

If \(e = e'\) then \(\mathcal{H}'(e') = \text{wmsg}(op) = m'\) and \(e'.loc = m'.loc = x\) and \(e.wval = m'.wval = v\).

Hence \(\mathcal{H}'\) preserves the condition.

(4) Condition to show: For all outstanding promises of threads \((T \setminus \{i\})\), there are corresponding write events in \(G'\) that are po-after \(S'\).

We know that for each thread \(j \neq i\) the set of promises are preserved from \(MS\) to \(MS'\), that is, \(\forall j \neq i. TS(j).P = TS'(j).P\).

We also know that \(G\) satisfies this condition.

Hence the condition is preserved in \(G'\).

Note. This was same as the other scenario when we append a new \(St_s(x, v)\).

(5) Condition to show: For every location \(\ell\) and thread \(j\), the thread view of \(\ell\) in the promise state \(MS'\) records the timestamp of the maximal write visible to the covered events in \(G'\) of thread \(j\).

The argument is analogous to the case when we append a new \(U_o(x, v, v')\).

(6) Condition to show: The \(S'\) events in \(G'\) preserve coherence: \(shb'; seco'\) is irreflexive.

The argument is analogous to the case when we append a new \(U_o(x, v, v')\).

(7) Condition to show: The atomicity condition for update operations hold for \(S'\) events in \(G'\).

The argument is analogous to the case when we append a new \(U_o(x, v, v')\).

(8) Condition to show: The sc fences in \(G'\) are appropriately ordered by \(sc'\).

We know \([G, F_{sc}]; shb \cup shb; seco; shb; [G, F_{sc}] \subseteq sc\) holds in \(G\).

The argument is analogous to the case when we append a new \(U_o(x, v, v')\).

(9) Condition to show: The behavior of \(MS'\) matches that of the \(S'\) events in \(G'\).

The argument is analogous to the case when we append a new \(U_o(x, v, v')\).
Case Release fence $F_{\text{rel}}$:

In the event structure we extend the event structure $G$ to $G'$. We extend the cover set $S_i$ as well as the relations (spo, srf, smo) to $S'_i$ along with the respective relations (spo', srf', smo') by including an event $e'$ where

1. $\text{dom}(G.\text{po};\{e'\}) = S_0 \cup S_i$,
2. $e' \in S'_i \setminus S_i$, and
3. $\text{labels}(\text{sequence}_{G.\text{po}}(S_i)).(e'.\text{lab}) \in \mathbb{P}(i)$.

In this case the promise machine is updated as follows.

$M' = M, S' = S,$

and $\mathcal{T}S' = \mathcal{T}S[i \mapsto \langle \mathbb{P}(i), \text{labels}(\text{sequence}_{\text{spo'}}(S'_i)), \langle V.\text{cur}, V.\text{acq}, V.\text{rel'} \rangle, \mathcal{T}S(i).P \rangle]$.

Now we do a case analysis on whether such an release fence event $e'$ exists in $G$ or we append a new event.

Subcase $\not\exists e' \in (G.E_i \setminus S'_i), \text{dom}(G.\text{po};\{e'\}) \subseteq S_i \land e'.\text{lab} = F_{\text{rel}}$:

We create $e'$ such that $e'.\text{lab} = F_{\text{rel}}$ and append $e'$ to event structure $G$ to create $G'$. Then,

- $G'.E = G.E \cup \{e' \;|\; e'.\text{lab} = F_{\text{rel}}\}$
- $G'.\text{po} = (G.\text{po} \cup \{(e, e') \;|\; e \in (S_i \cup S_0)\})^+$
- $G'.jf = G.jf$
- $G'.ew = G.ew$

Let: $\mathcal{W}' \triangleq \mathcal{W}$.

Based on $\mathcal{W}'$, we derive following definitions in $\mathcal{M}S'$.

- $S' \triangleq S \cup \{e'\}$
- $\text{mo'} \triangleq \text{mo}$
- $\text{sc'} \triangleq \text{sc}$
- $\text{spo'} \triangleq (\text{spo} \cup \{(e, e') \;|\; e \in S_0 \cup S'_i\})^+$
- $\text{srf'} \triangleq \text{srf}$

Now we check whether $G' \sim_{\{i\}} (\mathcal{T}S', S', M')$.

(1) Condition to show: $G'$ is consistent in weakest model.

- (CF) and (CFJ) constraints are preserved in $G'$. The arguments are analogous to the scenario when we append a new $\text{St}_o(x, v)$.
- (VISJ) Constraint (VISJ) is preserved in $G'$ as $G'.jf = G.jf$ and $G$ satisfies constraint (VISJ).
- (ICF)

We know that $G$ satisfies (ICF). Suppose there exists an event $e_1 \in G$ which is in immediate conflict with $e'$ in $G'$, that is $G'. \sim (e_1, e')$ holds.

Then (1) $\text{dom}(G.\text{po};\{e_1\}) = S_0 \cup S_i$,
(2) $e_1 \in S'_i \setminus S_i$, and
(3) $\text{labels}(\text{sequence}_{G.\text{po}}(S_i)).(e_1.\text{lab}) \in \mathbb{P}(i)$.

However, from definition of $e'$ we already know that
(1) $\text{dom}(G.\text{po};\{e'\}) = S_0 \cup S_i$,
(2) $e' \in S'_i \setminus S_i$, and
(3) $\text{labels}(\text{sequence}_{G.\text{po}}(S_i)).(e'.\text{lab}) \in \mathbb{P}(i)$.

Hence following the determinacy condition we know either $e_1 = e'$ or there exists no such $e_1$.

Hence (ICF) is preserved in $G'$.

Note. This was similar to the scenario when we append a new $\text{St}_o(x, v)$.

- (ICFJ) Constraint (ICFJ) is preserved in $G'$ as $e' \not\in \mathcal{R}$ and $G$ satisfies constraint (ICFJ).
(COH) We know $G$ preserves (COH) constraint, that is, $(G.hb; G.\eco_{strong})$ is acyclic. The incoming edges to event $e'$ are $G'.po$ and there is no outgoing edge concerning $G'.hb$ or $G'.\eco_{strong}$. As a result, $(G'.hb; G'.\eco_{strong})$ is acyclic and $G'$ preserves (COH) constraint.

(2) Condition to show: The local state of each thread in $MS'$ contains the program of that thread along with the sequence of covered events in $G'$ of that thread.

In this we have to show $\forall j. TS'(j).\sigma = (P(j), labels(sequence_{spo}(S'_j)))$.

We know that the relation holds between $MS$ and $G$.

For $j \neq i$, it is trivial because $TS'(j) = TS(j)$ holds from $MS$ to $MS'$ and $S'_j = S_j$ holds from $G$ to $G'$.

For $j = i$, we know $TS(i).\sigma = (P(i), labels(sequence_{spo}(S_i)))$.

Hence following the definition of $TS(i).\sigma, S'_i, spo'$ we get

$$(P(i), labels(sequence_{spo}(S'_i)))$$

Hence the condition is preserved between $MS'$ and $G'$.

(3) Condition to show: Whenever $W'$ maps an event of $G'$ to a message in $MS'$, then the location accessed and the written values match.

We know that the event to message mappings for existing events in $G.E$ and messages $M$ do not change, that is, $\forall e \in G'.E. e \neq e' \implies W'(e') = W(e)$. If $e = e'$ then $W'(e') = \bot$.

Hence $W'$ preserves the condition.

(4) Condition to show: For all outstanding promises of threads $(T \setminus \{i\})$, there are corresponding write events in $G'$ that are po-after $S'$.

We know that for each thread $j \neq i$ the set of promises are preserved from $MS$ to $MS'$, that is,

$\forall j \neq i. TS(j).P = TS(j').P$.

We also know that $G$ satisfies this condition.

Hence the condition is preserved in $G'$.

(5) Condition to show: For every location $\ell$ and thread $j$, the thread view of $\ell$ in the promise state $MS'$ records the timestamp of the maximal write visible to the covered events in $G'$ of thread $j$.

Essentially we have to show

$\forall j, \ell. TS'(j).V(\ell) = \max\{W'(e).Ts | e \in \text{dom}(W(\ell); G'.j; f; shb'; sc'; shb'; [S'])\}$

We know the relation holds in $G$.

In $G'$, for all $j, \ell, TS'(j).V(\ell) = TS(j).V(\ell)$ considering the mapping of $TS'$.

Hence $TS'$ satisfies the same condition and the relation holds between $MS'$ and $G'$.

(6) Condition to show: The $S'$ events in $G'$ preserve coherence: $shb'; seco'^{\bot}$ is irreflexive.

We know $shb; seco'$ is irreflexive.

Following the definition of components of $shb'$ and $seco'^{\bot}$ we know $shb'; seco'$ is irreflexive.

(7) Condition to show: The atomicity condition for update operations holds for $S'$ events in $G'$.

We know that $[G'.U \cap S'] = [G.U \cap S]$ and $[G.U \cap S]; (sfr; smo) = \emptyset$ holds.

The $e'$ does not introduce any $[G.U]; G'.sfr'$ or $[G.U]; G'.smo'$ edge.

As a result, $[G'.U \cap S']; (sfr'; smo') = \emptyset$ holds.

(8) Condition to show: The sc fences in $G'$ are appropriately ordered by $sc'$.

There is no outgoing edge from $e'$ to any event in $S'$.

Hence event $e'$ cannot introduce a new $(shb' \cup shb'; seco'; shb')$ path between two SC fences.
Hence \([G'.F_{sc}]; \text{shb}' \cup \text{shb}'; \text{seco}'; \text{shb}'; [G'.F_{sc}]\)
implies \([G.F_{sc}]; \text{shb} \cup \text{shb}; \text{seco}; \text{shb}; [G.F_{sc}]\).
We also know \(sc' = sc\).
We also know \([G.F_{sc}]; \text{shb} \cup \text{shb}; \text{seco}; \text{shb}; [G.F_{sc}] \subseteq sc\).
Hence \([G'.F_{sc}]; \text{shb}' \cup \text{shb}'; \text{seco}'; \text{shb}'; [G'.F_{sc}] \subseteq sc\) holds.

(9) Condition to show: The behavior of \(MS'\) matches that of the \(S'\) events in \(G'\).
Essentially we have to show, \(\text{Behavior}(MS') = \text{Behavior}(G', W', S')\).
We know \(\text{Behavior}(MS) = \text{Behavior}(G, W, S)\) holds.
From the definition we know,
\(\text{Behavior}(MS') = \text{Behavior}(MS)\) and \(\text{Behavior}(G', W', S') = \text{Behavior}(G, W, S)\) hold.
As a result, \(\text{Behavior}(MS') = \text{Behavior}(G', W', S')\) holds.

Subcase \(\exists e' \in (G.E_i \setminus \bar{S}_i). \text{dom}(G.po; \{e'\}) = \bar{S}_0 \cup \bar{S}_i \wedge e'.\text{lab} = F_{\text{rel}}\):
Note that promising semantics does not promise over a release fence. As a result, the certificate steps do not have any release fence. Hence there is no existing release fence event correspond to any certificate step which can be referred later in the simulation step. As a result, this case is not possible.

Case Acquire fence \(F_{\text{acq}}\):
In the event structure we extend the event structure \(G\) to \(G'\). We extend the cover set \(\bar{S}_i\) as well as the relations (spo, srf, smo) to \(\bar{S}'_i\) along with the respective relations (spo', srf', smo') by including an event \(e'\) where

1. \(\text{dom}(G.po; \{e'\}) = \bar{S}_0 \cup \bar{S}_i\),
2. \(e' \in \bar{S}'_i \setminus \bar{S}_i\), and
3. \(\text{labels(sequence}_{G.po}(\bar{S}_i)).(e'.\text{lab}) \in \mathbb{P}(i)\).

In this case the promise machine is updated as follows.
\(M' = M, S' = S, \) and
\(TS' = TS[i \mapsto ((\mathbb{P}(i), \text{labels(sequence}_{spo}(\bar{S}'_i))), (V.cur', V.acq, V.rel), TS(i).P)]\)
Now we do a case analysis on whether such an acquire fence event \(e'\) exists in \(G\) or we append a new event.

Subcase \(\exists e' \in (G.E_i \setminus \bar{S}_i). \text{dom}(G.po; \{e'\}) = \bar{S}_0 \cup \bar{S}_i \wedge e'.\text{lab} = F_{\text{acq}}\):
We create \(e'\) such that \(e'.\text{lab} = F_{\text{acq}}\) and append \(e'\) to event structure \(G\) to create \(G'\). Then,

- \(G'.E = G.E \cup \{e' \mid e'.\text{lab} = F_{\text{acq}}\}\)
- \(G'.\text{po} = G.po \cup \{\langle e, e' \rangle \mid e \in (\bar{S}_i \cup \bar{S}_0)\}\)
- \(G'.\text{jf} = G.jf\)
- \(G'.\text{ew} = G.ew\)

Let: \(W' \triangleq W\).
Based on \(W'\), we derive following definitions in \(MS'\).

- \(S' \triangleq S \cup \{e'\}\)
- \(\text{mo}' \triangleq \text{mo}\)
- \(sc' \triangleq sc\)
- \(\text{spo}' \triangleq (\text{spo} \cup \{(e, e') \mid e \in \bar{S}_0 \cup \bar{S}'_0\})^+\)
- \(\text{srf}' \triangleq \text{srf}\)

Note that there may be incoming synchronization edges to the acquire fence, that is, \(ssw \subseteq ssw'\) and hence \(\text{shb} \subseteq \text{shb}'\).
Now we check whether \(G' \sim_{\{i\}} (TS', S', M')\).
(1) Condition to show: \( G' \) is consistent in weakest model.

- (CF) The constraint is preserved in \( G' \). The argument is analogous to the scenario when we append a new \( \text{Ld}_v(x, v) \).
- (CFJ) Constraint (CFJ) is preserved in \( G' \). The argument is analogous to the scenario when we append a new \( \text{St}_v(x, v) \).
- (VISJ) Constraint (VISJ) is preserved in \( G' \) as \( G'.jf = G.jf \) and \( G \) satisfies constraint (VISJ).
- (ICF)
  
  We know that \( G \) satisfies (ICF). Suppose there exists an event \( e_1 \in G \) which is in immediate conflict with \( e' \) in \( G' \), that is \( G'. \sim (e_1, e') \) holds.
  
  Then (1) \( \text{dom}(G.p_o;\{e_1\}) = S_0 \cup S_i \),
  
  (2) \( e_1 \in S_i \setminus S_i \), and
  
  (3) \( \text{labels}(\text{sequence}_{G.p_o(S_i)})(e_1.\text{lab}) \in \mathbb{P}(i) \).

  However, from definition of \( e' \) we already know that

  (1) \( \text{dom}(G.p_o;\{e'\}) = S_0 \cup S_i \),
  
  (2) \( e' \in S_{i'} \setminus S_i \), and
  
  (3) \( \text{labels}(\text{sequence}_{G.p_o(S_i)})(e'.\text{lab}) \in \mathbb{P}(i) \).

  Hence following the determinacy condition we know either \( e_1 = e' \) or there exists no such \( e_1 \).

  Hence (ICF) is preserved in \( G' \).

  Note. This was similar to the scenario when we append a new \( \mathcal{T}_{\text{rel}} \).

- (ICFJ) Constraint (ICFJ) is preserved in \( G' \) as \( e' \notin \mathcal{R} \) and \( G \) satisfies constraint (ICFJ).

- (COH) We know \( G \) preserves (COH) constraint, that is, \( (G.\text{hb}; G.\text{eco}^2_{\text{strong}}) \) is acyclic. The incoming edges to event \( e' \) are \( G'.p_o \) and \( G'.\text{hb} \) (due to \( G'.\text{sw} \) edges), and there is no outgoing edge concerning \( G'.\text{hb} \) or \( G'.\text{eco}^2_{\text{strong}} \). As a result, \( (G.\text{hb}; G'.\text{eco}^2_{\text{strong}}) \) is acyclic and \( G' \) preserves (COH) constraint.

(2) Condition to show: The local state of each thread in \( MS' \) contains the program of that thread along with the sequence of covered events in \( G' \) of that thread.

In this we have to show \( \forall i. \mathcal{T}S'(i).\sigma = \langle \mathbb{P}(i), \text{labels}(\text{sequence}_{\text{sp}_0(S'_i)}(\mathcal{S}'_j))) \rangle \).

We know that the relation holds between \( MS \) and \( G \).

For \( j \neq i \), it is trivial because \( \mathcal{T}S'(j) = \mathcal{T}S(j) \) holds from \( MS \) to \( MS' \) and \( S'_j = S_j \) holds from \( G \) to \( G' \).

For \( j = i \), we know \( \mathcal{T}S(i).\sigma = \langle \mathbb{P}(i), \text{labels}(\text{sequence}_{\text{sp}_0(S_i)}(\mathcal{S}'_i)) \rangle \).

Hence following the definition of \( \mathcal{T}S(i).\sigma, S'_i, \text{sp}_0 \) we get

\[
\langle \mathbb{P}(i), \text{labels}(\text{sequence}_{\text{sp}_0(S'_i)}(\mathcal{S}'_i)) \rangle = \langle \mathbb{P}(i), \text{labels}(\text{sequence}_{\text{sp}_0(S_i)}(\mathcal{S}_i)) \rangle.
\]

Hence the condition is preserved between \( MS' \) and \( G' \).

(3) Condition to show: Whenever \( \mathbb{W}' \) maps an event of \( G' \) to a message in \( MS' \), then the location accessed and the written values match.

We know that the event to message mappings for existing events in \( G.E \) and messages \( M \) do not change, that is, \( \forall e \in G'.E. e \neq e' \implies \mathbb{W}'(e) = \mathbb{W}(e) \). If \( e = e' \) then \( \mathbb{W}'(e') = \bot \).

Hence \( \mathbb{W}' \) preserves the condition.

(4) Condition to show: For all outstanding promises of threads \( (T \setminus \{i\}) \), there are corresponding write events in \( G' \) that are po-after \( S' \).
We know that for each thread \( j \neq i \) the set of promises are preserved from \( MS \) to \( MS' \), that is, 
\[
\forall j \neq i. \ TS(j).P = TS'(j).P.
\]
We also know that \( G \) satisfies this condition.

Hence the condition is preserved in \( G' \).

(5) Condition to show: For every location \( \ell \) and thread \( j \), the thread view of \( \ell \) in the promise state \( MS' \) records the timestamp of the maximal write visible to the covered events in \( G' \) of thread \( j \).

Essentially we have to show
\[
\forall j, \ell. \ TS'(j).V(\ell) = \max\{W'(e).ts \mid e \in \text{dom}([W_i]; G'.jf^f; shb^f; sc^f; shb^f; \{S'_i\})\}.
\]
We know the relation holds in \( G \).

In \( G' \),
- for all \( j \neq i \), \( TS'(j).V(\ell) = TS(j).V(\ell) \) considering the mapping of \( TS' \).
- For \( j = i \), \( TS'(j).V.\text{cur} = TS(j).V.\text{acq} \).

We know that \( TS(i).V.\text{cur} \leq TS(i).V.\text{acq} \) for all location \( \ell \).

As a result, in this case \( TS'(i).V.\text{cur} \geq TS(i).V.\text{cur} \).

Hence
\[
\forall \ell. \ TS'(i).V(\ell) = \max\{W'(e).ts \mid e \in \text{dom}([W_i]; G'.jf^f; shb^f; sc^f; shb^f; \{S'_i\})\} \text{ holds}.
\]
Thus the relation holds between \( MS' \) and \( G' \).

(6) Condition to show: The \( S' \) events in \( G' \) preserve coherence: \( shb'; seco^f \) is irreflexive.

We know \( shb; seco^f \) is irreflexive.

Following the definition of components of \( shb' \) and \( seco^f \) we know \( shb'; seco^f \) is irreflexive.

(7) Condition to show: The atomicity condition for update operations holds for \( S' \) events in \( G' \).

The argument is analogous to the case when we append a new \( F_{\text{rel}} \).

(8) Condition to show: The sc fences in \( G' \) are appropriately ordered by \( sc' \).

The argument is analogous to the case when we append a new \( F_{\text{rel}} \).

(9) Condition to show: The behavior of \( MS' \) matches that of the \( S' \) events in \( G' \).

The argument is analogous to the case when we append a new \( F_{\text{rel}} \).

**Subcase** \( \exists e' \in (G.E_i \setminus S_i). \ \text{dom}(G.p_o;\{e'\}) = S_0 \cup S_i \land e'.\text{lab} = F_{\text{acq}} \):

Note that promising semantics does not promise over an acquire fence. As a result, the certificate steps do not have any acquire fence. Hence there is no existing acquire fence event correspond to any certificate step which can be referred later in the simulation step. As a result, this case is not possible.

**Case SC fence** \( F_{sc} \):

In the event structure we extend the event structure \( G \) to \( G' \). We extend the cover set \( S_i \) as well as the relations (spo, srf, smo) to \( S'_i \) along with the respective relations (spo', srf', smo') by including an event \( e' \) where

1. \( \text{dom}(G.p_o;\{e'\}) = S_0 \cup S_i \),
2. \( e' \in S'_i \setminus S_i \), and
3. \( \text{labels}(\text{sequence}_{G.p_o(S_i)})(e'.\text{lab}) \in \mathbb{P}(i) \).

In this case the promise machine is updated as follows.

\[
M' = M, S' = \{ (x, t) \mid x \in \text{Locs} \land \max(\mathcal{T}S(i).V.\text{cur}(x), t') \land (x, t') \in S \}, \text{ and}
\]
\[
\mathcal{T}S' = \mathcal{T}S[i \mapsto (\langle \mathbb{P}(i), \text{labels}(\text{sequence}_{spo}(S'_i)), S'_i, \mathcal{T}S(i).P \rangle)]
\]
Now we do a case analysis on whether such an sc fence event \( e' \) exists in \( G \) or we append a new event.
Subcase \( \exists e' \in (G.E_i \setminus S_i). \text{dom}(G.po; \{e'\}) \subseteq S_i \land e'.\text{lab} = F_{sc} \):

We create \( e' \) such that \( e'.\text{lab} = \mathcal{F}_{sc} \) and append \( e' \) to event structure \( G \) to create \( G' \). Then,

- \( G'.E = G.E \cup \{e' | e'.\text{lab} = \mathcal{F}_{sc}\} \)
- \( G'.po = G.po \cup \{(e, e') | e \in (S_i \cup S_0)\} \)
- \( G'.jf = G.jf \)
- \( G'.ew = G.ew \)

Let: \( \mathbb{W}' = \mathbb{W} \).

Based on \( \mathbb{W}' \), we derive following definitions in \( MS' \).

- \( \mathcal{S}' \triangleq \mathcal{S} \cup \{e'\} \)
- \( \text{mo}' \triangleq \text{mo} \)
- \( \text{sc}' \triangleq \text{sc} \cup \{(a, e') | a \in (G.\mathcal{F}_{sc} \cap \mathcal{S})\} \)
- \( \text{spo}' \triangleq \text{spo} \cup \{(e, e') | e \in S_0 \cup S'_0\} \)
- \( \text{srf}' \triangleq \text{srf} \)

Note that there may be incoming synchronization edges to the acquire fence, that is, \( ssw \subseteq ssw' \) and hence \( shb \subseteq shb' \).

Now we check whether \( G' \sim_{\{1\}} (\mathcal{T}\mathcal{S}', \mathcal{S}', M') \).

(1) Condition to show: \( G' \) is consistent in weakest model.

- (CF) The constraint is preserved in \( G' \). The argument is analogous to the scenario when we append a new \( \text{Ld}_o(x, \nu) \).
- (CFJ) Constraint (CF) is preserved in \( G' \). The argument is analogous to the scenario when we append a new \( \text{St}_o(x, \nu) \).
- (VISJ) Constraint (VISJ) is preserved in \( G' \) as \( G'.jf = G.jf \) and \( G \) satisfies constraint (VISJ).
- (ICF)

  We know that \( G \) satisfies (ICF). Suppose there exists an event \( e_1 \in G \) which is in immediate conflict with \( e' \) in \( G' \), that is \( G'. \sim (e_1, e') \) holds.

  Then (1) \( \text{dom}(G.po; \{e_1\}) = S_0 \cup S_i \),
  (2) \( e_1 \in S'_i \setminus S_i \), and
  (3) \( \text{labels}(\text{sequence}_{G.po}(S_i)).(e_1.\text{lab}) \in \mathcal{P}(i) \).

  However, from definition of \( e' \) we already know that
  (1) \( \text{dom}(G.po; \{e'\}) = S_0 \cup S_i \),
  (2) \( e' \in S'_i \setminus S_i \), and
  (3) \( \text{labels}(\text{sequence}_{G.po}(S_i)).(e'.\text{lab}) \in \mathcal{P}(i) \).

  Hence following the determinacy condition we know either \( e_1 = e' \) or there exists no such \( e_1 \).

  Hence (ICF) is preserved in \( G' \).

Note. This was similar to the scenario when we append a new \( \mathcal{F}_{ac}^1(x, \nu) \).

- (ICFJ) Constraint (ICFJ) is preserved in \( G' \) as \( e' \not\in \mathcal{R} \) and \( G \) satisfies constraint (ICFJ).
- (COH) We know \( G \) preserves (COH) constraint, that is, \( G.\text{hb}; G.\text{eco}_{\text{strong}}^1 \) is acyclic. The incoming edges to event \( e' \) are \( G'.po \) and \( G'.\text{hb} \) (due to \( G'.\text{sw} \) edges), and there is no outgoing edge concerning \( G'.\text{hb} \) or \( G'.\text{eco}_{\text{strong}}^1 \). As a result, \( G'.\text{hb}; G'.\text{eco}_{\text{strong}}^1 \) is acyclic and \( G' \) preserves (COH) constraint.

(2) Condition to show: The local state of each thread in \( MS' \) contains the program of that thread along with the sequence of covered events in \( G' \) of that thread.

In this we have to show \( \forall j. \mathcal{T}\mathcal{S}'(j).\sigma = (\mathcal{P}(j), \text{labels}(\text{sequence}_{\text{spo}}(S'_j))) \).

We know that the relation holds between \( MS \) and \( G \).
For \( j \neq i \), it is trivial because \( TS'(j) = TS(j) \) holds from MS to MS' and \( S'_j = S_j \) holds from G to G'.

For \( j = i \), we know \( TS(i).\sigma = (P(i), \text{labels}(\text{sequence}_{s_{po}}(S_i))) \).

Hence following the definition of \( TS(i).\sigma, S'_i \), spo' we get
\[
\langle P(i), \text{labels}(\text{sequence}_{s_{po}}(S'_i)) \rangle
= \langle P(i), TS(i).\sigma.e'.lab \rangle
= TS'(i).\sigma.
\]
Hence the condition is preserved between MS' and G'.

(3) Condition to show: Whenever \( \mathcal{W}' \) maps an event of G' to a message in MS', then the location accessed and the written values match.

We know that the event to message mappings for existing events in G,E and messages M do not change, that is, \( \forall e \in G'.E. e \neq e' \implies \mathcal{W}'(e) = \mathcal{W}(e). \)

Hence \( \mathcal{W}' \) preserves the condition.

(4) Condition to show: For all outstanding promises of threads \( (T \setminus \{i\}) \), there are corresponding write events in G' that are po-after \( S' \).

We know that for each thread \( j \neq i \) the set of promises are preserved from MS to MS', that is, \( \forall j \neq i. TS(j).P = TS'(j).P. \)

We also know that \( G \) satisfies this condition.

Hence the condition is preserved in G'.

(5) Condition to show: For every location \( \ell \) and thread \( j \), the thread view of \( \ell \) in the promise state MS' records the timestamp of the maximal write visible to the covered events in G' of thread \( j \).

Essentially we have to show
\[
\forall \ell, j. TS'(j).V(\ell) = \max\{\mathcal{W}'(e).ts \mid e \in \text{dom}(\{W_\ell\}; G'.jf^\ell; shb'^\ell; sc'^\ell; shb'^\ell; [S'_j])\}.
\]

We know the relation holds in G.

For \( j \neq i \), it is trivial because \( TS'.V(\ell) = TS.V(\ell) \).

For \( j = i \), we know that for a given location \( x \),
\( TS'(i).V(x) \) extends \( TS(i).V(x) \) by choosing between timestamp from \( TS(i).V(x) \) and timestamp from \( MS_c,TS'(c.id).V(x) \) where \( \text{imm}(sc'^\ell)(c,e') \) holds.

Hence \( \forall \ell. TS'(i).V(\ell) = \max\{\mathcal{W}'(e).ts \mid e \in \text{dom}(\{W_\ell\}; G'.jf^\ell; shb'^\ell; sc'^\ell; shb'^\ell; [S'_j])\} \)
holds.

Thus the relation holds between MS' and G'.

(6) Condition to show: The \( S' \) events in G' preserve coherence: \( shb'; seco'^\ell \) is irreflexive.

We know \( shb; seco'^{\ell} \) is irreflexive.

Following the definition of components of \( shb' \) and \( seco'^{\ell} \) we know \( shb'; seco'^{\ell} \) is irreflexive.

(7) Condition to show: The atomicity condition for update operations holds for \( S' \) events in G'.

The argument is analogous to the case when we append a new \( F_{rel} \).

(8) Condition to show: The sc fences in G' are appropriately ordered by \( sc'^\ell \).

There is no outgoing edge from \( e' \) to any event in \( S' \).

Hence event \( e' \) cannot introduce a new \( (shb' \cup shb'^\ell; seco'; shb') \) path between two SC fences.

Hence \( [G'.F_{sc}]; shb' \cup shb'; seco'; shb'; [G'.F_{sc}] \) implies \( [G.F_{sc}]; shb' \cup shb'; seco'; shb; [G.F_{sc}] \).
We also know \( sc \subseteq sc'^\ell \).

We also know \( [G.F_{sc}]; shb' \cup shb; seco; shb; [G.F_{sc}] \subseteq sc \).

Hence \( [G'.F_{sc}]; shb' \cup shb'; seco'; shb'; [G'.F_{sc}] \subseteq sc' \) holds.
(9) Condition to show: The behavior of \( S′ \) matches that of the \( \mathcal{S}' \) events in \( G' \).

The argument is analogous to the case when we append a new \( F_{rel} \).

**Subcase** \( \exists e' \in (G.E \setminus \mathcal{S}_i). \) dom\((G.po; \{\{e'\}\}) = \mathcal{S}_0 \cup \mathcal{S}_i \land e'.lab = F_{sc} \):

Note that promising semantics does not promise over an SC fence. As a result, the certificate steps do not have any SC fence. Hence there is no existing SC fence event correspond to any certificate step which can be referred later in the simulation step. As a result, this case is not possible.

**Case** \( \text{FULFILL op} = \text{fulfill}(m') \):

In the event structure we extend the event structure \( G \) to \( G' \). We extend the cover set \( \mathcal{S}_i \) as well as the relations \((spo, srf, smo)\) to \( \mathcal{S}'_i \) along with the respective relations \((spo', srf', smo')\) by including a write (store or update) event \( e' \) where

1. \( \text{dom}(G.po; \{\{e'\}\}) = \mathcal{S}_0 \cup \mathcal{S}_i \),
2. \( e' \in \mathcal{S}'_i \setminus \mathcal{S}_i \), and
3. \( \text{labels}(\text{sequence}_{G'.po}(\mathcal{S}'_i)).(e'.lab) \in \mathcal{P}(i) \).

In the promise machine let \( m' = \langle x : v' @ (f, t), \neg \rangle \).

Then the promise machine is updated as follows.
\[
M' = M \setminus \{m'\}, \quad S' = S, \\
\text{and } TS' = TS[i \mapsto (\langle \mathcal{P}(i), \text{labels}(\text{sequence}_{spo'}(\mathcal{S}'_i)), V', TS(i).P \setminus \{m'\} \rangle)]
\]

where \( V' = TS(i).V[x \mapsto t] \).

Now we do a case analysis on whether such an event \( e' \) exists in \( G \) or we append a new event.

Based on \( (\mathcal{P}(i), \text{labels}(\text{sequence}_{spo'}(\mathcal{S}'_i))) \) the event is either a store or an update event.

**Subcase** \( \exists e' \in (G.E \setminus \mathcal{S}_i). \) dom\((G.po; \{\{e'\}\}) = \mathcal{S}_0 \cup \mathcal{S}_i \land (e'.lab = St_o(x, v') \lor (e'.lab = U_o(x, v, v') \land G.jsf(w_m, e')) \) where \( w_m = \mathcal{W}(w_m) \):

We create \( e' \) such that \( e'.lab = St_o(x, v') \) or \( e'.lab = U_o(x, v, v') \) accordingly and append \( e' \) to event structure \( G \) to create \( G' \). Then,

- \( G'.E = G.E \cup \{e'\} \)
- \( G'.po = (G.po \cup \{(e, e') \mid e \in (\mathcal{S}_i \cup \mathcal{S}_0)\})^+ \)
- \( G'.jsf = G.jsf \cup \{(w_m, e') \mid e' \in U \land w_m \in G'.W_x \land w.mval = v \land \mathcal{W}(w_m) = m\} \)
- \( G'.ew = G.ew \cup \{(w_p, e') \mid w_p.id \neq e'.id \land \mathcal{W}(w_p) = m'\} \)

Let: \( \mathcal{W}' \triangleq \mathcal{W}[e' \mapsto m'] \).

Based on \( \mathcal{W}' \), we derive following definitions in \( MS' \).

- \( \mathcal{S}' \triangleq \mathcal{S} \cup \{e'\} \)
- \( \text{mo}' \triangleq \text{mo} \cup \{(a, e') \mid a \in G'.W_x \land \mathcal{W}(a) \neq \bot \land \mathcal{W}'(a).ts < \mathcal{W}'(e').ts\} \)
- \( \cup \{(e', a) \mid a \in G'.W_x \land \mathcal{W}(a) \neq \bot \land \mathcal{W}'(e').ts < \mathcal{W}'(a).ts\} \)
- \( \text{sc}' \triangleq \text{sc} \)
- \( \text{spo}' \triangleq (\text{spo} \cup \{(e, e') \mid e \in \mathcal{S}_0 \cup \mathcal{S}'_i\})^+ \)
- \( \text{srf}' \triangleq \text{srf} \cup \{(e', r) \mid (e', r) \in G'.rf(e', r) \land r \in \mathcal{S}'\} \)
- \( \cup \{(w_m, e') \mid e' \in G'.U \land G'.rf(w_m, e') \land w_m \in \mathcal{S}' \land w.mval = v \land \mathcal{W}'(w_m) = w.m\} \)

Now we check whether \( G' \sim_{\{i\}} (TS', \mathcal{S}', M') \).

(1) Condition to show: \( G' \) is consistent in weakest model.

- \( \text{(CF)} \)

We know that \( G \) satisfies (CF).

New \( G'.hb \) edges are created by the incoming edges to \( e' \). The outgoing \( G'.rf \) edge from \( e' \) does not result in any new synchronization.

The constraint is preserved in $G'$. If $e' \in G'.St$ then the argument is analogous to the scenario when we append a new $St_o(x, v)$ event. If $e' \in G'.U$ then the argument is analogous to the scenario when we append a new $U_o(x, v, v')$ event.

Hence $G'$ satisfies (CF).

- (CFJ)
  We know that $G$ satisfies (CF).
  Hence the new $hb$ edges are created by the incoming edges to $e'$. The outgoing $G'.rf$ edge from $e'$ does not result in any new synchronization.
  In that case the (CFJ) constraint is preserved in $G'$. If $e' \in G'.St$ then the argument is analogous to the scenario when we append a new $St_o(x, v)$ event. If $e' \in G'.U$ then the argument is analogous to the scenario when we append a new $U_o(x, v, v')$ event.

- (VISJ)
  - case $e' = St_o(x, v')$.
    Constraint (VISJ) is preserved in $G'$ as $G'.jf = G.jf$ and $G$ satisfies constraint (VISJ).
    Note. This was the same as the other scenario when we append a new $St_o(x, v')$.
  - case $e' = U_o(x, v, v')$.
    We study the possible cases of $w_m$.
    * If $G'.po(w_m, e')$ then the condition holds as $(w_m, e') \notin G'.jfe$.
    * We will show that $G'$ satisfies (CFJ) constraint. Hence $w_m$ cannot be in conflict with $e'$, that is, $(w_m, e') \notin G'.cf$.
    * $w_m$ is in different thread and $G'.jfe(w_m, e')$ holds. We know that $G \sim (i)$ MS and the simulation rules ensures that there is no invisible event in the $(T \setminus \{i\})$ threads. Hence $w_m$ is a visible event in $G$ as well as in $G'$.
    Considering the above mentioned cases $G'.jfe(w_m, e') \implies w_m \in vis(G')$ holds and $G'$ satisfies (VISJ) constraint.
    Note. This was the same as the other scenario when we append a new $U_o(x, v, v')$.

- (ICF) Constraint (ICF) is preserved in $G$. Now considering the cases of $e'$:
  - case $e' = St_o(x, v')$.
    Suppose there exists an event $e_1 \in G$ which is in immediate conflict with $e'$ in $G'$, that is $G'. \sim (e_1, e')$ holds.
    Then (1) $\text{dom}(G.po; \{e_1\}) = S_0 \cup S_i$,
    (2) $e_1 \in S_0 \setminus S_i$, and
    (3) $\text{label}(\text{sequence}_{G.po}(S_i)).(e_1.lab) \in \mathbb{P}(i)$.
    However, from definition of $e'$ we already know that
    (1) $\text{dom}(G.po; \{e'\}) = S_0 \cup S_i$,
    (2) $e' \in S_0 \setminus S_i$, and
    (3) $\text{label}(\text{sequence}_{G.po}(S_i)).(e'.lab) \in \mathbb{P}(i)$.
    Hence following the determinacy condition we know either $e_1 = e'$ or there exists no such $e_1$.
    Hence (ICF) is preserved in $G'$.
  - case $e' = U_o(x, v, v')$.
    Following the construction $e' \in G'.R$ and following the determinacy condition, if $G'. \sim (e_1, e')$ then $e_1 \in Ld$ or $e_1 \in U$. Thus $(e_1, e') \in (G'.R \times G'.R)$ and hence $G'$ satisfies (ICF).

- (ICFJ) From the construction we know either $e' \in St$ or there exists no $e_1$ such that $\text{imm}(cf)(e_1, e')$ and $G.rf(W^{-1}(w_m), e_1)$. Moreover, $G$ satisfies constraint (ICFJ). As a result, $G'$ satisfies (ICFJ).
• (COH) We know $G$ preserves (COH) constraint, that is, $(G.hb; G.eco^2_{str})$ is acyclic.

Now we check if $G'$ has $(G'.hb; G'.eco^2_{str})$ cycle.

If there exists $(G'.hb; G'.eco^2_{str})$ cycle then the cycle contains $G'.rf(e', r)$ and $(r, e') \in (G'.hb; G'.eco^2_{str})$ holds.

Since $(r, e') \notin G'.hb$, $(r, e') \in (G'.hb; G'.eco_{str})$ holds.

Now we consider the cases of event $e'$.

- case $e' = St_o(x, v')$.

  The incoming edges to event $e'$ are $G'.ew, G'.hb, G'.fr_{str}$ edges and the outgoing edges are $G'.ew, G'.rf$ edges.

  Note that as $e'$ is a newly appended event and no read event reads from $e'$ no new $G'.rf(w_m, -)$ is created.

  In that case the incoming edge to $e'$ is $G'.fr_{str}$ or $G'.mo_{str}$.

  * subcase $G'.mo_{str}$: Let $G'.mo_{str}(w_1, e')$ be the incoming edge. In that case, considering Lemma 3, $\forall'(w_m).ts < \forall'(w_1).ts, \forall'(w').ts < \forall'(e').ts$. However, we know $\forall'(w_m).ts = m'.ts = \forall'(e').ts$. Hence this is not possible.

  * subcase $G'.fr_{str}$: Let $G'.fr_{str}(r_1, e')$ be the incoming edge.

    Let $G'.fr_{str}(r_1, e')$ holds. In that case $G'.mo_{str}(w_1, e')$ holds and hence like the earlier case $\forall'(w_1).ts < m'.ts$ holds.

    However, we know that $(r, r_1) \in G'.hb; G'.eco^2_{str}$ and hence following Lemma 3, $m'.ts \leq \forall'(w_1).ts$. Hence a contradiction. As a result, $(G'.hb; G'.eco^2_{str})$ is irreflexive.

- case $e' = U_o(x, v, v')$.

  The incoming edges to event $e'$ are $G'.ew, G'.hb, G'.fr_{str}$, and $G'.rf$ edges and the outgoing edges are $G'.ew, G'.rf$ edges.

  Note that as $e'$ is a newly appended event and no read event reads from $e'$ no new $G'.rf(w_m, -)$ is created.

  The argument for incoming $G'.ew, G'.hb, G'.fr_{str}$ edges are same as the earlier cases where $e'$ is a store event.

  So now we consider the case where $G'.rf(\cdot, e')$ is the incoming edge to $e'$. Let the edge be $G'.rf(w'', e')$ and hence $(r, w'') \in (G'.hb; G'.eco^2_{str})$.

  Following Lemma 3, (1) $m'.ts \leq \forall'(w'').ts$. However, following the promising semantics for update operation we know that (2) $\forall'(e').ts > \forall'(w'').ts$ holds which implies $m'.ts > \forall'(w'').ts$.

  The (1) and (2) contradicts and hence there is no $(G'.hb; G'.eco^2_{str})$ cycle.

  Hence $(G'.hb; G'.eco^2_{str})$ is irreflexive.

Thus $G'$ satisfies (COH).

As a result, $G'$ is consistent in weakest model.

(2) Condition to show: The local state of each thread in MS' contains the program of that thread along with the sequence of covered events in $G'$ of that thread.

In this we have to show $\forall j. TS'(j).\sigma = \langle P(j), labels(sequence_{spo}(S_j)) \rangle$.

We know that the relation holds between MS and G.

For $j \neq i$, it is trivial because $TS'(j) = TS(j)$ holds from MS to $MS'$ and $S_j' = S_j$ holds from $G$ to $G'$.

For $j = i$, we know $TS(i).\sigma = \langle P(i), labels(sequence_{spo}(S_i)) \rangle$.

Hence following the definition of $TS(i).\sigma, S_i', spo'$ we get

$\langle P(i), labels(sequence_{spo}(S_i')) \rangle$ = $\langle P(i), labels(sequence_{spo}(S_i)) \rangle \cdot e'.lab$
\[ \langle P(i), TS(i).\sigma \cdot e'.lab \rangle = TS'(i).\sigma \]

Hence the condition is preserved between MS' and G'.

(3) Condition to show: Whenever \( W' \) maps an event of \( G' \) to a message in MS', then the location accessed and the written values match.

We know that the event to message mappings for existing events in G.E and messages M do not change.

\[ \forall e \in G'.E. \ e \neq e' \implies W'(e) = W(e) \]

If \( e = e' \) then \( W'(e') = m' \) and \( e'.loc = m'.loc = x \) and \( e'.wval = m'.wval = v' \).
Hence \( W' \) preserves the condition.

(4) Condition to show: For all outstanding promises of threads \( (T \setminus \{i\}) \), there are corresponding write events in \( G' \) that are po-after \( S' \).

We know that for each thread \( j \neq i \) the set of promises are preserved from MS to MS', that is, \( \forall j \neq i. TS(j).P = TS'(j).P \).
We also know that \( G \) satisfies this condition.
Hence the condition is preserved in \( G' \).

(5) Condition to show: For every location \( \ell \) and thread \( j \), the thread view of \( \ell \) in the promise state \( MS' \) records the timestamp of the maximal write visible to the covered events in \( G' \) of thread \( j \).

Essentially we have to show
\[ \forall j, \ell. \ TS'(j).V(\ell) = \max(\{W'(e).ts | e \in \text{dom}([W_\ell]; G'.j; f_i^j; sc_i^j; shb_i^j; [S_i']]) \}. \]
For \( j = i \) or \( j = i \land \ell = x \), it is trivial because \( TS'.V(\ell) = TS.V(\ell) \).
For \( j \neq i \land \ell = x \),
Based on the type of event \( e' \)
\[ \text{case } e' \in G.St_x, \]
following the promising semantics \( W'(e') = m', m'.ts \) extends the view on \( x \) in thread \( i \),
and hence \( TS(i).V(x) < TS'(i).V(x) \).
In this case, \( e' \in \text{dom}([W_\ell]; G'.f_i^j; shb_i^j; sc_i^j; shb_i^j; [S_i']) \).
So \( TS(i).V(x) = \max(\{W'(e).ts | e \in \text{dom}([W_\ell]; G'.f_i^j; shb_i^j; sc_i^j; shb_i^j; [S_i']) \} \) holds.
\[ \text{case } e' \in G.U_x, \]
Then, \( TS(i).V(x) = \max(\{W'(e).ts | e \in \text{dom}([W_\ell]; G'.j; f_i^j; shb_i^j; sc_i^j; shb_i^j; [S_i']) \} \) holds.
Following the promising semantics, we know \( TS'(i).V(x) \) extends the thread view of \( x \) from \( TS(i).V(x) \) by reading from some message \( \text{wm} \), and so \( TS(i).V(x) < \text{wm}.ts \).
Moreover, following the semantics of update in the promise machine, \( \text{wm}.ts < m'.ts \).
So \( TS'(i).V(x) = \max(\{W'(e).ts | e \in \text{dom}([W_\ell]; G'.f_i^j; shb_i^j; sc_i^j; shb_i^j; [S_i']) \} \) holds.
Thus the relation holds between \( MS' \) and \( G' \).

(6) Condition to show: The \( S' \) events in \( G' \) preserve coherence: \( shb'; seco' \) is irreflexive.

The argument is analogous to the new \( St_o(x, v, v') \) or new \( U_o(x, v, v') \) events.

(7) Condition to show: The atomicity condition for update operations holds for \( S' \) events in \( G' \).

The argument is analogous to the new \( St_o(x, v, v') \) or new \( U_o(x, v, v') \) events.

(8) Condition to show: The \( sc \) fences in \( G' \) are appropriately ordered by \( sc' \).

We know \([G.F_{sc}]; shb \cup shb; seco; shb; [G.F_{sc}] \subseteq sc \) holds in \( G \).
From definitions we know, \( G'.F_{sc} = G.F_{sc}, sc' = sc, shb \subseteq shb', seco \subseteq seco' \).
Consider \( a, b \) are two SC fences such that \( (a, b) \in [G.F_{sc}]; shb \cup shb; seco; shb; [G.F_{sc}] \), and \( sc(a, b) \) holds.
In that case \((a, b) \in (\text{shb}' \cup \text{shb}; \text{seco}'; \text{shb}')\) holds and \(\text{sc}'(a, b)\) holds.

To show \([G', F_{sc}]; \text{shb}' \cup \text{shb}; \text{seco}'; \text{shb}'; [G', F_{sc}] \subseteq \text{sc}'\),
we have to show \((b, a) \notin (\text{shb}' \cup \text{shb}; \text{seco}'; \text{shb}')\).

We show that by contradiction. Assume \((b, a) \in (\text{shb}' \cup \text{shb}; \text{seco}'; \text{shb}')\).

This is possible due to the relations created to/from event \(e'\).

Considering the relations in \(\text{shb}'\) and \(\text{seco}'\),
(1) when \(e' \in G'.\text{St}\), the incoming relations to event \(e'\) are \(\text{shb}',\text{srf}',\text{smo}'\) and the outgoing edges are \(\text{srf}', \text{smo}'\).

(2) when \(e' \in G'.\text{U}\), the incoming and outgoing relations to event \(e'\) are same as when \(e' \in G'.\text{St}\). Additionally, there are \(\text{srf}'\) incoming edges to \(e'\).

In this case the path from \(b\) to \(a\) is \((b, e') \in \text{shb}; \text{seco}'\),
and \((e', a) \in \text{srf}'; \text{seco}'\) or \((e', a) \in \text{smo}'\).

We analyze the cases of \((b, e') \in \text{shb}; \text{seco}'\).

Similar to the new \(\text{St}_0(x, v, v')\) or the new \(\text{U}_0(x, v, v')\), in this case also \(\text{MS}_b.\text{TS}(b.\text{tid}).V(x) < \text{MS}_e.\text{TS}(e'.\text{tid}).V(x)\) holds.

Now we consider the outgoing edges:

- \((e', a) \in \text{srf}'; \text{seco}'\) or \(\text{shb}'\).

There exists \(r\) such that \(\text{srf}'(e', a)\) and \((r, a) \in \text{seco}'\) or \(\text{shb}'\).

Hence, \(\text{MS}_e.\text{TS}(e'.\text{tid}).V(x) = \text{MS}_s.\text{TS}(r.\text{tid}).V(x) \leq \text{MS}_a.\text{TS}(a.\text{tid}).V(x)\).

- \((e', a) \in \text{smo}'\) or \(\text{sc}'\).

There exists a write \(w \in \mathcal{S}\) such that \(\text{smo}'(e', w)\) and \((w, a) \in \text{seco}'\) or \(\text{shb}'\).

Hence, \(\text{MS}_e.\text{TS}(e'.\text{tid}).V(x) < \text{MS}_w.\text{TS}(w.\text{tid}).V(x) \leq \text{MS}_a.\text{TS}(a.\text{tid}).V(x)\).

Considering both cases \(\text{MS}_b.\text{TS}(b.\text{tid}).V(x) < \text{MS}_a.\text{TS}(a.\text{tid}).V(x)\) holds.

This is a contradiction and hence \((b, a) \notin (\text{shb}' \cup \text{shb}; \text{seco}'\) or \(\text{shb}'\).

As a result, \([G', F_{sc}]; \text{shb}' \cup \text{shb}; \text{seco}'\) or \([G', F_{sc}] \subseteq \text{sc}'\) holds.

(9) Condition to show: The behavior of \(\text{MS}'\) matches that of the \(\mathcal{S}'\) events in \(G'\).

The argument is analogous to the case when we append a new store or update event.

**Subcase** \(\exists e' \in (G.\text{E} \setminus \mathcal{S}_i). \text{dom}(G.\text{po}; \{e'\}) = \mathcal{S}_0 \cup \mathcal{S}_i \land (e'.\text{lab} = \text{St}_0(x, v, v') \lor (e'.\text{lab} = \text{U}_0(x, v, v') \land \text{G.jf}(w_m, e'))\) where \(w_m = \mathcal{W}(w_m)\).

In this case an event created for the promise certificate corresponds to the fulfill operation.

We take \(G' = G\) and let \(\mathcal{W}' = \mathcal{W}[e' \mapsto m']\) and

Based on \(\mathcal{W}'\), we derive following definitions in \(\text{MS}'\):

- \(\mathcal{S}' := \mathcal{S} \cup \{e'\}\)
- \(\mathcal{mo}' := \mathcal{mo}\)
- \(\mathcal{sc}' := \mathcal{sc}\)
- \(\mathcal{spo}' := (\mathcal{spo} \cup \{(e, e') \mid e \in \mathcal{S}_0 \cup \mathcal{S}_i\}^+)\)
- \(\mathcal{sr}' := (\mathcal{sr} \cup \{(e', r) \mid (e', r) \in G'.\text{rf}(e', r) \land r \in \mathcal{S}'\})\)
- \(\mathcal{sm}' \cup \{(w_m, e') \mid e' \in G' \land G'.\text{rf}(w_m, e') \land w_m \in \mathcal{S}' \land w_m.\text{wval} = v \land w_m.\mathcal{W} = w_m = \mathcal{W}(w_m)\} \)

Now we check whether \(G' \sim_{\{1\}} (\text{TS}'', \mathcal{S}', \mathcal{M}')\).

(1) Condition to show: \(G'\) is consistent in weakest model.

\(G'\) is consistent as \(G\) is consistent in weakest model.

(2) Condition to show: The local state of each thread in \(\text{MS}'\) contains the program of that thread along with the sequence of covered events in \(G'\) of that thread.

In this we have to show \(\forall j. \text{TS}'(j).\sigma = (\mathcal{P}(j), \text{labels(sequence}_{\mathcal{spo}'(\mathcal{S}_j))})).\)

We know that the relation holds between \(\text{MS}\) and \(G\).
For \( j \neq i \), it is trivial because \( TS'(j) = TS(j) \) holds from \( MS \) to \( MS' \) and \( S'_j = S_j \) holds from \( G \) to \( G' \).
For \( j = i \), we know \( TS(i).\sigma = \langle P(i), labels(\text{sequence}_{spo}(S_i)) \rangle \).
Hence following the definition of \( TS(i).\sigma, S'_i, \text{spo} \) we get
\[
\langle P(i), labels(\text{sequence}_{spo}(S'_i)) \rangle = \langle P(i), TS(i).\sigma \cdot e'.\text{lab} \rangle = TS'(i).\sigma
\]
Hence the condition is preserved between \( MS' \) and \( G' \).

(3) Condition to show: Whenever \( W' \) maps an event of \( G' \) to a message in \( MS' \), then the location accessed and the written values match.

We know that the event to message mappings for existing events in \( G.E \) and messages \( M \) do not change.

\[
\forall e \in G'.E. e \neq e' \implies W'(e) = W(e)
\]
If \( e = e' \) then \( W'(e') = m' \) and \( e'.\text{loc} = m'.\text{loc} = x \) and \( e'.\text{wval} = m'.\text{wval} = v' \).
Hence \( W' \) preserves the condition.

(4) Condition to show: For all outstanding promises of threads \( T \setminus \{i\} \), there are corresponding write events in \( G' \) that are po-after \( S' \).

We know that for each thread \( j \neq i \) the set of promises are preserved from \( MS \) to \( MS' \), that is,
\[
\forall j \neq i. TS(j).P = TS'(j).P.
\]
We also know that \( G \) satisfies this condition.
Hence the condition is preserved in \( G' \).

(5) Condition to show: For every location \( \ell \) and thread \( j \), the thread view of \( \ell \) in the promise state \( MS' \) records the timestamp of the maximal write visible to the covered events in \( G' \) of thread \( j \).

The argument is analogous to the new \( St_o(x, v, v') \) or new \( U_o(x, v, v') \) events.
Thus the relation holds between \( MS' \) and \( G' \).

(6) Condition to show: The \( S' \) events in \( G' \) preserve coherence: \( \text{shb}'; \text{seco}' \) is irreflexive.

The argument is analogous to the case when we append a new store or update event for a fulfill operation.

(7) Condition to show: The atomicity condition for update operations holds for \( S' \) events in \( G' \).

The argument is analogous to the new store or update event.

(8) Condition to show: The \( \text{sc} \) fences in \( G' \) are appropriately ordered by \( \text{sc}' \).

The argument is analogous to the case when we append a new store or update event for a fulfill operation.

(9) Condition to show: The behavior of \( MS' \) matches that of the \( S' \) events in \( G' \).

The argument is analogous to the case when we append a new store or update event.

Now we prove Lemma 2.

**Lemma 2.** \( G \sim MS \wedge MS \rightarrow MS' \implies \exists G'. G \rightarrow_{P, \text{WEAKEST}^*} G' \wedge G' \sim MS' \).
Proof. Following the promise machine step:

\[
\begin{align*}
\langle TS(i), S, M \rangle \xrightarrow{\text{p-step}} \langle TS', S', M' \rangle & \xrightarrow{\text{p-step}} \langle TS'', S'', M'' \rangle \\
\langle TS'', S'', M'' \rangle & \text{ is consistent}
\end{align*}
\]

\[
\langle TS, S, M \rangle \xrightarrow{\text{op}} \langle TS[i \mapsto TS''], S'', M'' \rangle
\]

Case analysis on the op:

\[
\begin{align*}
\langle TS(i), S, M \rangle \xrightarrow{\text{op}_{i}} & \langle TS', S', M' \rangle \xrightarrow{\text{op}_{i}} \langle TS'', S'', M'' \rangle \\
\langle TS, S, M \rangle & \xrightarrow{\text{op}} \langle TS[i \mapsto TS''], S', M' \rangle \quad M''.P = \emptyset
\end{align*}
\]

Case Non-promise step:

From \( G \sim MS \), we get \( G \sim_{\{i\}} MS \).

By Lemma 1 and induction, we have

\[
\exists G', G \xrightarrow{*} G' \land G' \sim_{\{i\}} \langle TS[i \mapsto TS'], S', M' \rangle
\]

and by Lemma 1 and induction, we have

\[
\exists G''. G' \xrightarrow{*} G'' \land G'' \sim_{\{i\}} \langle TS[i \mapsto TS''], S'', M'' \rangle
\]

It remains to show \( G'' \sim MS' \).

We know that a certificate does not create any new message or SC fence. Hence \( M'' = M' \) and \( S'' = S' \).

We take \( \mathbb{W}'' = \mathbb{W}' \) as there exists a write event in the certificate which maps to the promise message and in this case \( mo'' = mo' \) and \( S'' = S', sc'' = sc', spo'' = spo', srf'' = srf', seco'' = seco' \) hold.

(1) From Eq. (i) we know that \( G'' \sim_{\{i\}} \langle TS[i \mapsto TS''], S'', M'' \rangle \). Hence \( G'' \) is consistent.

(2) From Eq. (i) we know that

\[
\forall j. TS'(j).\sigma = \langle P(j), \text{labels}_{\text{spo'}}(S_j') \rangle
\]

Hence \( \forall j. TS'(j).\sigma = \langle P(j), \text{labels}_{\text{spo'}}(S_j') \rangle \) also holds since \( S'' = S' \).

(3) From Eq. (i) we know \( G'' \sim_{\{i\}} \langle TS'[i \mapsto TS''], S'', M'' \rangle \). We also know that \( M'' = M' \) holds.

Hence whenever \( \mathbb{W}''(e) = m \) then \( e.\text{loc} = m.\text{loc} \) and \( e.\text{wval} = m.\text{wval} \).

(4) From Eq. (i) we know \( G' \sim_{\{i\}} \langle TS[i \mapsto TS'], S', M' \rangle \). Hence the following also holds.

\[
\forall j \in (T \setminus \{i\}). \forall e \in (S_0' \cup S_j'). TS'(j).P \subseteq \{ \mathbb{W}'(e') \mid (e, e') \in G'.po \}.
\]

It implies

\[
\forall j \in (T \setminus \{i\}). \forall e \in (S_0' \cup S_j'). TS'(j).P \subseteq \{ \mathbb{W}''(e') \mid (e, e') \in G''.po \} \quad (a)
\]
In thread $i$ events in $(S'_0 \cup S'_i)$ in $G'$ has $G'$-po-following events $e'$ corresponding to the certificate of outstanding promises. Hence $\forall e \in (S'_0 \cup S'_i). TS'(i).P \subseteq \{\mathbb{W}'(e') | (e, e') \in G'.po\}$.

It implies

$$\forall e \in (S''_0 \cup S''_i). TS'(i).P \subseteq \{\mathbb{W}''(e') | (e, e') \in G''.po\}$$

(b)

Thus considering Eq. (a), Eq. (b) the following also holds

$$\forall j \in T. \forall e \in (S''_0 \cup S''_j). TS'(j).P \subseteq \{\mathbb{W}''(e') | (e, e') \in G''.po\}$$

Thus the condition is satisfied between $G''$ and $MS'$.

(5) From Eq. (i) we know

$$\forall i, x. TS'(i).V(x) = \max\{\mathbb{W}(e).ts | e \in \text{dom}([W_x]; G'.jf); shb''; sc''; shb''; [S'_i]\})$$

We know that $G'.po \subseteq G''.po$, $G'.jf \subseteq G''.jf$, $G'.ew \subseteq G''.ew$.

Hence from the definitions following holds:

$$TS'(i).V(x) = \max\{\mathbb{W}''(e).ts | e \in \text{dom}([W_x]; G''.jf); shb''; sc''; shb''; [S'_i]\})$$

(6) From Eq. (ii) we already know $(shb''; seco''')$ is irreflexive.

(7) From Eq. (ii) we already know $[G''.U \cap S'']; (sfr''; smo'') = \emptyset$ holds.

(8) From Eq. (i) we know $[G'.F_{sc}]; shb' \cup shb; seco; shb; [G'.F_{sc}] \subseteq sc'$.

From Eq. (ii) we know $[G''.F_{sc}]; shb'' \cup shb; seco''; shb''; [G''.F_{sc}] \subseteq sc''$.

However, we know $sc'' = sc', G''.F_{sc} = G'.F_{sc}$, and $S'' = S'$.

Hence $[G''.F_{sc}]; shb'' \cup shb''; seco; shb''; [G''.F_{sc}] \subseteq sc'$.

(9) From Eq. (i) we know Behavior($MS' = Behavior(G', \mathbb{W}', S')$).

From Eq. (ii) we know Behavior($MS'' = Behavior(G'', \mathbb{W}'', S'')$).

However, Behavior($MS'' = Behavior(MS')$ holds and as a result, Behavior($MS' = Behavior(G', \mathbb{W}', S')$).

As a result, $G'' \sim MS'$ holds.

**Case Promise step:**

From $G \sim MS$, we get $G \sim_{\{i\}} MS$.

Also let $MS \xrightarrow{op_i} MS'$ holds where $op = \text{promise}(m)$ in the thread $i$.

We show: $\exists G'. G \rightarrow G' \wedge G' \sim_{\{i\}} MS'$

In this case $TS' = TS[i \mapsto TS']$, and $M' = M \cup \{m\}$, and we take $G' = G$.

Thus it remains to show that $G \sim_{\{i\}} MS'$.

We take $W = \mathbb{W}$

As a result $mo' = mo$ and $S' = S$, $sc' = sc$, spo' = spo, $sr' = sr$, $seco' = seco$ hold.

(1) From $G \sim MS$ we know $G$ is consistent and hence $G'$ is also consistent.

(2) From $G' \sim_{\{i\}} MS'$ we know that $\forall j \neq i. TS'(j).\sigma = (\mathbb{P}(j); labels(sequence_{spo}(S'_j)))$ holds.

Hence from the definitions $\forall j \neq i. TS'(j).\sigma = (\mathbb{P}(j); labels(sequence_{spo}(S'_j)))$ also holds.

For $j = i$, $TS'(i).\sigma = (\mathbb{P}(i); labels(sequence_{spo}(S'_i)))$ also holds.

It implies, $TS'(i).\sigma = (\mathbb{P}(i); labels(sequence_{spo}(S'_i)))$ also holds.

Hence $\forall j. TS'(i).\sigma = (\mathbb{P}(i); labels(sequence_{spo}(S'_i)))$ holds.

Thus the relation is preserved between $G$ and $MS'$. 
(3) From $G \sim MS$ we know whenever $\mathbb{W}(m) = e$ then $e$.loc = $m$.loc and $e$.wval = $m$.wval holds.
Since $\mathbb{W}' = \mathbb{W}$, the same also holds for $\mathbb{W}'$.

(4) We know $\forall j \in (T \setminus \{i\}) \forall e \in (S'_o \cup S'_i). TS'(j).P \subseteq \{\mathbb{W}'(e') | (e, e') \in G'.po\}$. Hence from the definitions $\forall j \in (T \setminus \{i\}) \forall e \in (S'_o \cup S'_j). TS'(j).P \subseteq \{\mathbb{W}(e) | (e, e) \in G.po\}$ holds.

(5) From $G \sim_{\{\}} MS$ we know
\[
\forall j \neq i. TS(j).V(e) = \max\{\mathbb{W}(e).ts | e \in \text{dom}([\mathbb{W}(e)]; G'.f_i; shb'; sc; shb; \{S_i\})\}
\]
Since $G' = G$, $\mathbb{W}' = \mathbb{W}$, and $TS' = TS[i \mapsto TS']$ the following also holds.
\[
\forall j \neq i. TS'(j).V(e) = \max\{\mathbb{W}(e).ts | e \in \text{dom}([\mathbb{W}(e)]; G'.f_i; shb'; sc; shb; \{S_i\})\}
\]

(6) From $G \sim_{\{i\}} MS$ we know $[G.F_{sc}]; shb \cup shb; seco; shb; [G.F_{sc}] \subseteq sc$ holds.
We know $G'.F_{sc} = G.F_{sc}$, $shb' = shb$, $seco' = seco$, and $sc' = sc$.
Hence, $[G'.F_{sc}]; shb' \cup shb'; seco'; shb'; [G'.F_{sc}] \subseteq sc$ also holds.

(7) From $G \sim_{\{i\}} MS$ we know $(shb; seco')$ is irreflexive.
From the definition $shb' = shb$ and $seco' = seco$ hold.
Hence $(shb'; seco')$ is irreflexive.

(8) From $G \sim_{\{i\}} MS$ we know $[G.U \cap S]; (sfr; smo) = \emptyset$ holds.
We also know $sfr' = sfr$ and $sma' = smo$, $S' = S$, and $G.U \subseteq G'.U$.
Hence $[G'.U \cap S']; (sfr'; sma') = \emptyset$ also holds.

(9) From $G \sim_{\{i\}} MS$ we know $\text{Behavior}(MS) = \text{Behavior}(G, \mathbb{W}, S)$. We also know that $S' = S$ and $G' = G$.
Now following the definitions of $MS'$ and $G'$, we get $\text{Behavior}(MS) = \text{Behavior}(MS')$ and $\text{Behavior}(G, \mathbb{W}, S) = \text{Behavior}(G', \mathbb{W}', S')$.
Hence $\text{Behavior}(MS') = \text{Behavior}(G', \mathbb{W}', S')$ holds.
Thus $G' \sim_{\{i\}} MS'$ holds.

Subcase Certificate step following the promise step:
From $G' \sim MS'$ we have $G' \sim_{\{i\}} MS'$ and also the following holds.

\[
\exists G''. G' \rightarrow^* G'' \land G'' \sim_{\{i\}} MS'' = \langle TS[i \mapsto TS''], M''\rangle
\]
It remains to show $G'' \sim MS'$
We know that $TS'' = TS'$. Moreover a certificate does not create any new message and hence $M'' = M'$.
We take $S'' = S'$, and $\mathbb{W}'' = \mathbb{W}'[e' \mapsto m]$ where $e'.loc = m$.loc, $e'.wval = m$.wval.
As a result, $mo' \subseteq mo''$, and $S'' = S', sc'' = sc'$.
However, $e' \not\in S''$ and hence $smo'' = smo'$.

(1) We know that $G'' \sim_{\{i\}} MS''$. Hence $G''$ is consistent.

(2) From $G' \sim MS'$ we know that
\[
\forall j. TS'(j).\sigma = \langle \mathcal{P}(j), labels(\text{sequence}_{spo}(S')) \rangle \text{ holds.}
\]
We also know that $S'' = S'$ and $TS'' = TS'$.
Hence $\forall j. TS'(j).\sigma = \langle \mathcal{P}(j), labels(\text{sequence}_{spo}(S'_j)) \rangle$ also holds.

(3) We know $G' \sim_{\{i\}} MS'$. We also know that $M'' = M'$ holds.
Hence whenever $\mathbb{W}'(e) = m$, then $e$.loc = $m$.loc and $e$.wval = $m$.wval holds.
(4) We know \( G' \sim (TS[i \mapsto TS'], S', M') \). Hence the following also holds.
\[
\forall j \in (T \setminus \{i\}). \forall e \in (S''_0 \cup S''_j). TS'(j).P \subseteq \{W'(e') | (e, e') \in G'.po\}.
\]
It implies
\[
\forall j \in (T \setminus \{i\}). \forall e \in (S''_0 \cup S''_j). TS'(j).P \subseteq \{W''(e') | (e, e') \in G''.po\}
\]
(\text{c})
In thread \( i \) events in \((S''_0 \cup S''_j)\) in \( G' \) has \( G' \).po-following events \( e' \) corresponding to the certificate of outstanding promises.
Hence \( \forall e \in (S''_0 \cup S''_j). TS'(j).P \subseteq \{W''(e') | (e, e') \in G'.po\} \).
It implies
\[
\forall e \in (S''_0 \cup S''_j). TS'(j).P \subseteq \{W''(e') | (e, e') \in G''.po\}
\]
(\text{d})
Thus considering Eq. (c), Eq. (d) the following also holds
\[
\forall j \in T. \forall e \in (S''_0 \cup S''_j). TS'(j).P \subseteq \{W''(e') | (e, e') \in G''.po\}
\]
Thus the condition is satisfied between \( G'' \) and \( MS' \).

(5) From \( G' \sim (TS') \) \( MS' \) We know
\[
TS'(i).V(t) = \max\{W'(e).ts | e \in \text{dom}(\{W_e; G'.j; jf; shb; sc; shb; [S'_i]\})\}
\]
We know that \( G'.e \subseteq G''.e, G'.po \subseteq G''.po, G'.jf \subseteq G''.jf, G'.ew \subseteq G''.ew, TS'' = TS', S'' = S', \) and \( W'' = W'[e' \mapsto m] \).
Hence from the definitions following holds:
\[
TS'(i).V(x) = \max\{W''(e).ts | e \in \text{dom}(\{W_x; G''.j; jf; shb; sc; shb; [S''_i]\})\}
\]
(\text{6})
We know \( (shb; seco') \) is irreflexive.
From the definition \( shb'' = shb' \) and \( seco'' = seco' \).
Hence \( (shb' ; seco') \) is irreflexive.

(7) From \( G' \sim (TS') \) \( MS' \) we know \([G'.U \cap S'] ; (sfr'; smo') = 0 \) holds.
We also know \( sfr'' = sfr' \) and \( smo'' = smo' \), \( S'' = S' \), and \( G'.U \subseteq G''.U \).
Hence \([G''.U \cap S''] ; (sfr'' ; smo'') = 0 \) also holds.

(8) We know \( S'' = S', \) \( mo' \subseteq mo'' \), \( sc'' = sc' \).
We also know that \([G'.F_{sc}] ; shb' \cup shb ; seco' ; shb' ; [G'.F_{sc}] \subseteq sc' \) holds.
Hence, \([G''.F_{sc}] ; shb'' \cup shb'' ; seco'' ; shb'' ; [G''.F_{sc}] \subseteq sc'' \) also holds.

(9) From \( G' \sim (TS') \) \( MS' \) we know \( \text{Behavior}(MS') = \text{Behavior}(G', W', S') \).
From \( G'' \sim (TS'') \) \( MS'' \) we know \( \text{Behavior}(MS'') = \text{Behavior}(G'', W'', S'') \).
From definitions \( \text{Behavior}(MS') = \text{Behavior}(MS') \) and \( \text{Behavior}(G'', W'', S'') = \text{Behavior}(G', W', S') \) holds.
Hence \( \text{Behavior}(MS') = \text{Behavior}(G'', W'', S'') \) holds.

Hence \( G'' \sim MS' \) holds.

\( \square \)

Finally we restate and prove Theorem 1.

**Theorem 1.** For a program \( P \), \( \text{Behavior}_{P}(P) \subseteq \text{Behavior}_{\text{WEAKEST}}(P) \).

**Formal statement:**
\[
\forall P. \forall MS. (MS_{\text{init}}(P) \rightarrow MS \land MS \rightarrow). \exists G, X. G_{\text{init}} \rightarrow P, \text{WEAKEST} \ast G \land X \in \text{ex}_{\text{WEAKEST}}(G).
\]
(\text{\land Behavior}(MS) = \text{Behavior}(X))
Proof. Step 1. Given a program $P$, from Lemma 2 we show that using the simulation relation in Definition 6, we can follow the promise machine steps and for a promise machine state state $MS$ we can construct an weakest event structure $G$, that is, $G_{init} \rightarrow_{P, {\text{WEAK}}^*} G$.

Step 2. Now we extract a consistent execution $X$ from $G$ where $X \in \text{ex}_{\text{WEAK}}(G)$, such that $\text{Behavior}(MS) = \text{Behavior}(X)$.

Given the event structure $G$ along with $S$ and related sets, the execution $X = (E, po, rf, mo)$ is as follows.

- $X.E = S$,
- $X.po = spo$,
- $X.rf = srf$, and
- $X.mo = smo$

Note that the events in $X.E$ is conflict-free as $S$ is conflict-free in $G$.

Now we check whether execution $X$ is consistent.

- From the definitions of $spo$, $srf$, $smo$, we know $X.po \subseteq (S \times S)$, $X.rf \subseteq (S \times S)$, and $X.mo \subseteq (S \times S)$.

Hence $X$ is (Well-formed).

- From the definition, we know $smo$ is total as the order on the timestamps on the same location is total in the promise machine.

Hence $X.mo$ is total and (total-MO) holds in $X$.

- From the construction of $G$ we know that $shb; seco^2$ is irreflexive.

Hence $(X.hu_{C11}; X.eco^2)$ is irreflexive and (Coherence) holds in $G$.

- From the construction we know that $[G.U \cap S]; (sfr; smo) = \emptyset$ holds. From the definition we know that $X.U = (G.U \cap S)$, $X.fr = sfr$, and also $X.mo = smo$ holds.

Hence $[X.U]; (X.fr; X.mo) = \emptyset$ hold and $X$ preserves (Atomicity).

- From the simulation relation in the construction we know that $sc$ is total in $G$ and $[G.F_{SC}; shb \cup shb; seco; shb; G.F_{SC}] \subseteq sc$ holds.

Hence $[G.F_{SC}; shb \cup shb; seco; shb; G.F_{SC}]$ is irreflexive.

From definition we know that $X.F_{SC} = G.F_{SC}$, $X.hu_{C11} = shb$, and $X.eco = seco$ hold.

As a result, $X.psc_f = [X.F_{SC}; X.hu_{C11} \cup X.hu_{C11}; X.eco; X.hu_{C11}; [X.F_{SC}]$ is irreflexive.

Note that $X$ does not have any SC memory access and hence $X.psc_{base} = \emptyset$.

Hence $X$ preserves (SC).

Thus $X$ is consistent and hence $X \in \text{ex}_{\text{WEAK}}(G)$.

Step 3. From the construction we know that $\text{Behavior}(MS) = \text{Behavior}(G, W, S)$.

Hence from the definitions $\text{Behavior}(MS) = \text{Behavior}(X)$.

Thus considering step 1, 2, 3 the theorem holds. \qed
B CAUSALITY TEST CASES

\[
\begin{align*}
  r_1 &= X; \\
  r_2 &= Y; \\
  \text{if}(r_1 \geq 0) &= Y = 1;
\end{align*}
\]

\[
\begin{align*}
  r_2 &= X; \\
  \text{if}(r_1 = r_2) &= X = r_3; \\
  Y &= 1;
\end{align*}
\]

Fig. 14. Case 1. Allowed \( r_1 == r_2 == 1 \).

\[
\begin{align*}
  r_1 &= X; \\
  r_2 &= X; \\
  \text{if}(r_1 = r_2) &= Y = 1;
\end{align*}
\]

\[
\begin{align*}
  r_3 &= Y; \\
  X &= r_3;
\end{align*}
\]

Fig. 15. Case 2. Allowed \( r_1 == r_2 == r_3 == 1 \).

\[
\begin{align*}
  r_1 &= X; \\
  r_2 &= X; \\
  \text{if}(r_1 = r_2) &= r_3 = 1; \\
  Y &= 1;
\end{align*}
\]

\[
\begin{align*}
  X &= 2;
\end{align*}
\]

Fig. 16. Case 3. Allowed \( r_1 == r_2 == r_3 == 1 \).
\( r_1 = X; \quad r_2 = Y; \quad Y = r_1; \quad r_2 = r_1 \)

\[ X = Y = 0 \]

Fig. 17. Case 4. Forbidden \( r_2 = r_1 = 1 \).

\( r_1 = X; \quad r_2 = Y; \quad r_3 = Z; \quad Z = 1; \quad X = r_2; \quad X = r_1 \)

\[ X = Y = Z = 0 \]

Fig. 18. Case 5. Forbidden \( r_1 = r_2 = 1, r_3 = 0 \). However, a sequence of transformations result this behavior.

\[ r_1 = A; \quad r_2 = B; \] (if \( r_1 = 1 \))

\[ B = 1; \quad r_2 = 0; \quad A = 1; \]

\[ A = B = 0 \]

Fig. 19. Case 6. Allowed \( r_1 = r_2 = 1 \).

\( r_1 = Z; \quad r_3 = Y; \quad r_2 = X; \quad Z = r_3; \quad Y = r_2; \quad X = 1; \)

\[ X = Y = Z = 0 \]

Fig. 20. Case 7. Allowed \( r_1 = r_2 = r_3 = 1 \).
\begin{align*}
    r_1 &= X; \\
    r_2 &= 1 + r_1 \ast r_1 - r_1; \\
    Y &= r_2;
\end{align*}

\begin{itemize}
    \item \hspace{1cm} \text{Fig. 21. Case 8. Allowed } r_1 == r_2 == 1.
\end{itemize}

\begin{align*}
    r_1 &= X; \\
    r_2 &= 1 + r_1 \ast r_1 - r_1; \\
    Y &= r_2;
\end{align*}

\begin{itemize}
    \item \hspace{1cm} \text{Fig. 22. Case 9. Allowed } r_1 == r_2 == 1.
\end{itemize}

\begin{align*}
    r_1 &= X; \\
    r_2 &= 1 + r_1 \ast r_1 - r_1; \\
    Y &= r_2;
\end{align*}

\begin{itemize}
    \item \hspace{1cm} \text{Fig. 23. Case 9a. Allowed } r_1 == r_2 == 1.
\end{itemize}

\begin{align*}
    r_1 &= X; \\
    \text{if} (r_1 == 1) \hspace{1cm} & \text{if} (r_2 == 1) \hspace{1cm} & \text{if} (r_3 == 1) \\
    Y &= 1; & X &= 1; & X &= 1;
\end{align*}

\begin{itemize}
    \item \hspace{1cm} \text{Fig. 24. Case 10. Forbidden } r_1 == r_2 == 1, r_3 == 0. \text{Same event structure as Fig. 18. Similar to test case 5, a sequence of transformations result this behavior.}
\end{itemize}
\[ r_1 = Z; \quad r_4 = W; \]
\[ W = r_1; \quad r_3 = Y; \]
\[ r_2 = X; \quad Z = r_3; \]
\[ Y = r_2; \quad X = 1; \]

Fig. 25. Case 11. Allowed \( r_1 = r_2 = r_3 = r_4 = 1 \).

\[ X = Y = 0; a[0] = 1; a[1] = 2; \]
\[ r_1 = X; \quad r_2 = Y; \]
\[ a[r_1] = 0; \quad r_3 = Y; \]
\[ r_2 = a[0]; \quad X = r_3; \]
\[ Y = r_2; \]

Fig. 26. Case 12. Forbids \( r_1 = r_2 = r_3 = 1 \).

\[ r_1 = X; \]
\[ \text{if}(r_1 == 1) \]
\[ Y = 1; \quad X = 1; \]
\[ \text{if}(r_2 == 1) \]

Fig. 27. Case 13. Forbids \( r_1 = r_2 = 1 \).

\[ r_1 = A; \]
\[ \text{if}(r_1 == 0) \]
\[ \text{do} \{ \]
\[ r_2 = Y_{sc}; \]
\[ Y_{sc} = 1; \]
\[ r_3 = B; \]
\[ \} \text{while}(r_2 + r_3 == 0); \]
\[ \text{else} \]
\[ B = 1; \]
\[ A = 1; \]

Fig. 28. Case 14. Forbids \( r_1 = r_3 = 1; r_2 = 0 \). In [Manson et al. 2004] \( Y \) is ‘volatile’ in Java. We map Java volatile to SC in C11 as the reordering rules are same.
\[ r_0 = X_{sc}; \]
\[ \text{if}(r_0 == 1) \quad r_1 = A; \]
\[ \text{else} \quad r_1 = 0; \]
\[ \text{if}(r_1 == 0) \quad Y_{sc} = 1; \]
\[ \text{else} \quad B = 1; \]
\[ \text{do} \quad \{ \quad r_2 = Y_{sc}; \quad r_3 = B; \quad \} \text{while}(r_2 + r_3 == 0); \]
\[ X_{sc} = 1; \]
\[ [A = B = X = Y = 0] \]

Fig. 29. Case 15. Forbids \( r_1 == r_3 == 1; r_2 == 0 \). In [Manson et al. 2004] \( X \) and \( Y \) are ‘volatile’ in Java. We map Java volatile to SC in C11 as the reordering rules are same.

\[ r_1 = X; \quad r_2 = X; \]
\[ X = 1; \quad X = 2; \]
\[ [X = Y = 0] \]

Fig. 30. Case 16. Behavior in question: \( r_1 = 2, r_2 = 1 \). This is allowed in Manson et al. [2004]. The behavior is allowed in basic event structure and in extracted execution as they do not enforce coherence. The weakest model constructs an event structure with these events but disallows the incoherent behavior in the extracted execution. The weakest model does not accommodate all these events together in any event structure and in consequence disallows the incoherent behavior in the extracted execution.
\( r_3 = X; \)
\( \text{if}(r_3 \neq 4) \quad r_2 = Y; \)
\( X = 4; \quad X = r_2; \)
\( r_1 = X; \)
\( Y = r_1; \)

\[ A = B = X = Y = 0 \]

[Fig. 31. Case 17 and 18. Allows \( r_1 = r_2 = r_3 = 4 \).]

\( r_3 = X; \)
\( \text{if}(r_3 = 0) \quad r_2 = Y; \)
\( X = 4; \quad X = r_2; \)
\( r_1 = X; \)
\( Y = r_1; \)

\[ A = B = X = Y = 0 \]

[Fig. 32. Case 19 and 20. Event Structure Forbids \( r_1 = r_2 = r_3 = 4 \).]
B.1 Allowing Forbidden Behaviors

Now we see certain behaviors which are disallowed by Manson et al. [2004] and our proposed scheme but are possible after a number of program transformations.

Testcase 5. The $r_1 == r_2 == 1, r_3 == 0$ outcome is possible after a sequence of transformations as follows.

\[
\begin{align*}
r_1 &= X; & r_2 &= Y; & r_3 &= Z; & Z &= 1; \\
Y &= r_1; & X &= r_2; & X &= r_3; & \leadsto \\
\end{align*}
\]

\[
\begin{align*}
r_1 &= X; & r_2 &= Y; & X &= r_3; & Z &= 1; \\
Y &= r_1; & \text{if}(r_2 == 1) X &= 1; & \text{else} X &= r_2; & X &= r_3; & \leadsto \\
\end{align*}
\]

\[
\begin{align*}
r_1 &= X; & r_2 &= Y; & \text{if}(r_2 == 1) X &= 1; & \text{else} X &= r_2; \\
Y &= r_1; & \{r_3 = Z; X = r_3; \} & || & \{Z = 1; \} \\
& \text{else}\{ \\
& \quad X &= r_2; \\
& \quad \{r_3 = Z; X = r_3; \} & || & \{Z = 1; \} \\
& \} \\
\leadsto \\
\end{align*}
\]

\[
\begin{align*}
r_1 &= X; & r_2 &= Y; & \text{if}(r_2 == 1) \{X = 1; r_3 = Z; X = r_3; Z = 1;\} & \text{else} \{X = r_2; Z = 1; r_3 = Z; X = r_3;\} & \leadsto \\
Y &= r_1; & \text{if}(r_2 == 1) \{X = 1; r_3 = Z; X = r_3; Z = 1;\} & \text{else} \{X = r_2; Z = 1; r_3 = 1; X = 1;\} \\
\leadsto \\
\end{align*}
\]

\[
\begin{align*}
a : r_1 &= X; & b : Y &= r_1; & \text{if}(r_2 == 1) \{e : r_3 = Z; X = r_3; Z = 1;\} & \text{else} \{Z = 1; r_3 = 1;\} \\
& \text{else} \{Z = 1; r_3 = 1;\} \\
\end{align*}
\]

Now it is possible to have an interleaving $c, a, b, d, e$ which results in $r_1 == r_2 == 1, r_3 == 0$.

Testcase 10. Similar to test case 5 the $r_1 == r_2 == 1, r_3 == 0$ outcome is possible after a sequence of transformations as follows.

\[
\begin{align*}
r_1 &= X; & r_2 &= Y; & r_3 &= Z; & Z &= 1; & \leadsto \\
\text{if}(r_1 == 1) & & \text{if}(r_2 == 1) & & \text{if}(r_3 == 1) & & \leadsto \\
Y &= 1; & X &= 1; & X &= 1; & \leadsto \\
\end{align*}
\]

\[
\begin{align*}
r_1 &= X; & r_2 &= Y; & \text{if}(r_2 == 1) & & \text{if}(r_3 == 1) & & \leadsto \\
\text{if}(r_1 == 1) & & X &= 1; & \text{else} & & X &= 1; & \leadsto \\
Y &= 1; & X &= 0; & X &= 1; & \leadsto \\
\end{align*}
\]
Now we can have an interleaving $d, a, b, c, e, f$ which results in $r_1 == r_2 == 1, r_3 == 0$. 
C PROOFS OF DRF THEOREMS

First we prove the following lemma.

**Lemma 5.** Given a program \( P \), suppose all its RC11-consistent executions are \( RLX \)-race-free. Let \( G \) be an event structure such that \( G_{init} \rightarrow_{P,\text{WEAKSMO}}^* G \). Then, \( G.jf \subseteq G.hb \) holds.

**Proof.** We show \( G.jf \subseteq G.hb \) holds by induction on the construction of \( G \). It holds trivially for \( G = G_{init} \) because \( G_{init}.jf = \emptyset \).

For the inductive case, we know that \( G_{init} \rightarrow_{P,\text{WEAKSMO}}^* G \rightarrow_{P,\text{WEAKSMO}}^* G' \) and \( G.jf \subseteq G.hb \), and have to show that \( G'.jf \subseteq G'.hb \). We do case analysis on the step \( G \rightarrow_{P,\text{WEAKSMO}}^* G' \); let \( e \) be the event appended to \( G \) to construct \( G' \).

**Case** \( e \notin R \). In this case, \( G'.jf = G.jf \) and \( G.hb \subseteq G'.hb \). Hence \( G'.jf \subseteq G'.hb \) holds.

**Case** \( e \in R \). In this case, there exists a write \( w \in G.E \) such that \( G'.jf = G.jf \cup \{(w,e)\} \). We consider the following cases for \( G.jf \cup \{(w,e)\} \):

**Subcase** \((w,e) \in G'.hb \). In this case, \( G'.jf \subseteq G'.hb \) holds.

**Subcase** \((e,w) \in G'.hb \). This case is not possible as it violates \( (COH') \) in \( G' \).

**Subcase** \((w,e) \notin G'.hb \). In this case, \((w,e) \in G'.\text{Race}(RLX) \).

We take \( A \) to be the \( G'.hb \)-prefixes of \( e \) and \( w \). From \( (CF) \), it follows that \( A \) is conflict-free.

Let \( G'' \) be the restriction of \( G' \) to \( A \). By construction, \( G'' \) is conflict-free \( \text{WEAKSMO} \) consistent event structure which is an RC11 execution and \((w,e) \in G''.\text{Race}(RLX) \). This contradicts the antecedent, and hence the statement holds. \( \Box \)

**Lemma 6.** Given a program \( P \), suppose all its RC11-consistent executions are \( RLX \)-race-free. Then \( X.rf \subseteq G.jf \) holds where \( X \) is an execution extracted from \( \text{WEAKSMO} \) event structure \( G \), that is, \( G_{init} \rightarrow_{P,\text{WEAKSMO}}^* G \) and \( X \in \text{ex}_{\text{WEAKSMO}}(G) \).

**Proof.** Assume \((w_1, r) \in X.rf \setminus G.jf \).

In this case there exists \( w_2 \) such that \( G.\text{ew}(w_1, w_2) \land (w_2, r) \in G.jf \).

From Lemma 5 we know \((w_2, r) \in G.hb \).

From the definition of \( X \) we know \( w_2 \in X.E \).

It contradicts that \( w_1 \in X.E \) and hence the statement holds. \( \Box \)

**Lemma 7.** Given a program \( P \), suppose all its RC11-consistent executions are \( RLX \)-race-free. Then \( X \) has no \((X.po \cup X.rf) \) cycle where \( X \) is an execution extracted from \( \text{WEAKSMO} \) event structure \( G \), that is, \( G_{init} \rightarrow_{P,\text{WEAKSMO}}^* G \) and \( X \in \text{ex}_{\text{WEAKSMO}}(G) \).

**Proof.** From Lemmas 5 and 6 we know \((X.po \cup X.rf) \subseteq (G.po \cup G.jf) \subseteq G.hb \). Hence \( X \) has no \((X.po \cup X.rf) \) cycle. \( \Box \)

Now we restate and prove the DRF-RLX theorem.

**Theorem 2** (DRF-RLX) Given a program \( P \), suppose its RC11-consistent executions are \( RLX \)-race-free. Then, \( \text{Behavior}_{\text{WEAKSMO}}(P) = \text{Behavior}_{\text{RC11}}(P) \).

**Proof.** Consider an extracted execution \( X \) from \( \text{WEAKSMO} \) event structure \( G \), that is, \( G_{init} \rightarrow_{P,\text{WEAKSMO}}^* G \) and \( X \in \text{ex}_{\text{WEAKSMO}}(G) \).

From Lemma 7 we know \( X \) has no \((X.po \cup X.rf) \) cycle.

Hence \( X \) is an RC11 execution where \( X.rf = X.jf \) and as a result, \( \text{Behavior}_{\text{WEAKSMO}}(P) = \text{Behavior}_{\text{RC11}}(P) \) holds. \( \Box \)
D weakestmo-llvm CONSTRUCTION RULES

\[ A \subseteq G.e.tid \quad \text{dom}([E.e.tid] ; \text{po} ; [A]) \subseteq A \quad \text{labels} \left( \text{sequence}_{\text{po}}(A) \right) \cdot e.\text{lab} \in \mathcal{P}(e.tid) \]

\[ E' = E \cup \{ e \} \quad \text{po}' = \text{po} \cup (A \times \{ e \}) \quad \text{isCons}_M((E', \text{po}', \text{jf}', \text{ew}', \text{mo}')) \quad CF = (E.e.tid \setminus A) \]

\[ \begin{align*}
\text{if } e \in R \text{ then } & \exists w \in E \cap 'W'. \text{jf}' = \text{jf} \cup \{ (w, e) \} \land w.\text{loc} = e.\text{loc} \land \\
& ((w, e) \in G'.\text{Race}(\text{NA}) \land e.\text{rval} = u \lor w.\text{wval} = e.\text{rval}) \\
\text{else } & \text{jf}' = \text{jf}
\end{align*} \]

\[ EW \subseteq \{ w \in 'W' \cap CF \mid w.\text{loc} = e.\text{loc} \land w.\text{wval} = e.\text{wval} \} \quad \text{ew}' = \text{ew} \cup (W \times \{ e \})^= \]

\[ W \subseteq AW = \{ w \in 'W' \cap E CF \mid w.\text{loc} = e.\text{loc} \land e \in 'W' \} \quad \text{mo}' = \text{mo} \cup W \times \{ e \} \cup \{ e \} \times (AW \setminus W) \]

\[ \langle E, \text{po}, \text{jf}, \text{ew}, \text{mo} \rangle \rightarrow_{P, M} \langle E', \text{po}', \text{jf}', \text{ew}', \text{mo}' \rangle \]

Fig. 33. weakestmo-llvm event structure construction rules where \( G' = \langle E', \text{po}', \text{jf}', \text{ew}', \text{mo}' \rangle \). The LLVM specific change is in green.
E  MONOTONICITY OF WEAKESTMO

The weaken transformation is as follows:

- \( \tau \cdot \text{Ld}_o(x, v) \cdot \tau' \xrightarrow{\text{WEAKEN}} \tau \cdot \text{Ld}_{o'}(x, v) \cdot \tau' \) where \( o' \subseteq o \)
- \( \tau \cdot \text{St}_o(x, v) \cdot \tau' \xrightarrow{\text{WEAKEN}} \tau \cdot \text{St}_{o'}(x, v) \cdot \tau' \) where \( o' \subseteq o \)
- \( \tau \cdot \text{U}_o(x, v, v') \cdot \tau' \xrightarrow{\text{WEAKEN}} \tau \cdot \text{U}_{o'}(x, v, v') \cdot \tau' \) where \( o' \subseteq o \)
- \( \tau \cdot \text{F}_o \cdot \tau' \xrightarrow{\text{WEAKEN}} \tau \cdot \text{F}_{o'} \cdot \tau' \) where \( o' \subseteq o \)
- \( \tau \cdot \text{F}_o \cdot \tau' \xrightarrow{\text{WEAKEN}} \tau \cdot \tau' \) where \( o' \subseteq o \)

\[ \forall X \in \text{weakestmo} \]

We prove that the weakestmo is a monotonic memory model.

**Theorem 9.** Given a program \( P_{\text{src}} \) if we weaken a program \( P_{\text{src}} \) to \( P_{\text{tgt}} \) then

1. For each consistent event structure of \( P_{\text{src}} \) there exists a consistent event structure of \( P_{\text{tgt}} \).
2. For each consistent execution extracted from a consistent event structure of \( P_{\text{src}} \) there exists a consistent execution extracted from a consistent event structure of \( P_{\text{tgt}} \).

Formal statement

\[ \forall P_{\text{src}}. \text{WEAKEN}(P_{\text{src}}, P_{\text{tgt}}) \implies \forall G_{\text{src}}, G_{\text{init}} \rightarrow P_{\text{src}}, \text{WEAKESTMO}^\ast(G_{\text{src}}). \exists G_{\text{tgt}}. G_{\text{init}} \rightarrow P_{\text{tgt}}, \text{WEAKESTMO}^\ast(G_{\text{tgt}}) \land \forall X_i \in \text{exWEAKESTMO}(G_{\text{src}}). \exists X_i \in \text{exWEAKESTMO}(G_{\text{tgt}}). \text{Behavior}(X_i) = \text{Behavior}(X_i) \]

**Proof.** (1) Given a target event structure \( G_{\text{init}} \rightarrow P_{\text{src}}, \text{WEAKESTMO}^\ast G_{\text{src}} \), we follow the construction steps of \( G_{\text{src}} \) and construct \( G_{\text{tgt}} \). In this construction, we can follow the write steps similar to that of \( G_{\text{tgt}} \). We can also follow the \( G_{\text{src}} \) fence step unless the fence is deleted. Hence we can append the reads with same labels by justifying from same writes compared to that of \( G_{\text{src}} \). Thus, \( G_{\text{tgt}}.E \subseteq G_{\text{src}}.E \), \( G_{\text{tgt}}.R^\ast W_{o'} \equiv G_{\text{tgt}}.R^\ast W_o \), \( G_{\text{tgt}}.po \subseteq G_{\text{src}}.po \), \( G_{\text{tgt}}.rf = G_{\text{src}}.rf \), and \( G_{\text{tgt}}.ew = G_{\text{src}}.ew \). While constructing \( G_{\text{tgt}} \) from \( G_{\text{src}} \), essentially we remove po edges to/from fences along with certain sw edges due to the removal of fences or replacing the Rel or Acq events with events with weaker or same memory order. As a result, we in turn remove certain hb relations and the relations between the SC accesses.

As a result, the \( G_{\text{tgt}} \) is less restrictive than \( G_{\text{src}} \) in terms of the relations involved in the weakest mo consistency conditions and \( G_{\text{tgt}} \) remains consistent.

(2) For each execution \( X_i \in \text{exWEAKESTMO}(G_{\text{src}}) \), we find an execution \( X_i \) such that \( X_i.E \subseteq X_i.E, X_i.R^\ast W_{o'} \equiv X_i.R^\ast W_o, X_i.po \subseteq X_i.po, X_i.rf = X_i.rf, X_i.mo = X_i.mo \).

Similar to the event structures, the \( X_i \) is less restrictive than \( X_i \) in terms of the relations involved in the execution consistency conditions. Hence \( X_i \) remains consistent and \( X_i \in \text{exWEAKESTMO}(G_{\text{src}}) \) holds. Moreover, in this case Behavior\((X_i) = \text{Behavior}(X_i) \) holds following the definitions of \( X_i \) and \( X_i \).

**Remark 3.** Consider we append a read \( r \) to consistent event structure \( G \) by justifying from a write \( w \in G.W \) from \((G'.hb \cup G'.jf)\)-prefix and create \( G' \) such that \( G' \) is consistent when \( \exists w(G', w, r) \) holds where

\[ \exists w(G', w, r) \triangleq (w, r) \in (G'.jf^2; G'.hb \setminus G'.ecf) \land w'. \exists w(G', w', r) \land G'.mo(w', w') \]

Note that there exists some write \( w \in G.W \) such that \( \exists w(G, w, r) \) holds as all locations are initialized.
F PROOFS OF CORRECTNESS OF REORDERINGS

We start with definitions and a lemma on \( \text{hb} \) in the \textsc{weakestmo} model.

We first define unique predecessor and unique successor.

\[ \text{Definition 8.} \quad \text{Upred}(R, a, b) \triangleq R(a, b) \land \forall c. \ G(c, b) \implies c = a \]

\[ \text{Definition 9.} \quad \text{Usucc}(R, a, b) \triangleq R(a, b) \land \forall c. \ G(R(a, c)) \implies c = b. \]

We derive the following lemma.

\[ \text{Lemma 8.} \quad \text{if Upred}(R, b, a) \text{ and Usucc}(R, b, a) \text{ holds then} \]
\[ (R \setminus \{(b, a)\}) \cup \{(a, b)\})^+ \subseteq R^+ \setminus \{(b, a)\} \cup \{(a, b)\} \text{ also holds.} \]

\[ \text{Proof.} \quad \text{We assume Upred}(R, b, a) \text{ and Usucc}(R, b, a) \text{ holds.} \]

Now we show \((R \setminus \{(b, a)\}) \cup \{(a, b)\})^+ \subseteq R^+ \setminus \{(b, a)\} \cup \{(a, b)\}.

We prove by induction on transitive closure.

Base Case: \((R \setminus \{(b, a)\}) \cup \{(a, b)\}) \subseteq (R^+ \setminus \{(b, a)\} \cup \{(a, b)\}).

The base case holds trivially by monotonicity.

The induction step:
\((R \setminus \{(b, a)\}) \cup \{(a, b)\}) \circ (R^+ \setminus \{(b, a)\} \cup \{(a, b)\}) \subseteq (R^+ \setminus \{(b, a)\} \cup \{(a, b)\}).

To prove the above mentioned induction, we consider following cases

\text{case 1.} (R \setminus \{(b, a)\}) \circ (R^+ \setminus \{(b, a)\}) \subseteq (R^+ \setminus \{(b, a)\} \cup \{(a, b)\}).

It is sufficient to show:
\((R \setminus \{(b, a)\}) \circ (R^+ \setminus \{(b, a)\}) \subseteq (R^+ \setminus \{(b, a)\})

Therefore it is sufficient to show,
\((R \setminus \{(b, a)\}) \circ (R^+ \setminus \{(b, a)\}) \subseteq R^+ \land (b, a) \notin (R \setminus \{(b, a)\}) \circ (R^+ \setminus \{(b, a)\}).

Now
(i) By monotonicity we know that \((R \setminus \{(b, a)\}) \circ (R^+ \setminus \{(b, a)\}) \subseteq R^+.

Therefore it is sufficient to show
(ii) \((b, a) \notin (R \setminus \{(b, a)\}) \circ (R^+ \setminus \{(b, a)\})

Assume \((b, a) \in (R \setminus \{(b, a)\}) \circ (R^+ \setminus \{(b, a)\})

By unfolding the definition of \( \circ \), it is sufficient to show
\[ \exists c. \ (R \setminus \{(b, a)\}) \cap (c, a) \in (R^+ \setminus \{(b, a)\}). \]

Assume \( \exists c. \ (b, c) \in R \setminus \{(b, a)\}. \)

Therefore \((b, c) \in R \land c \neq a \land (c, a) \in R^+ \land c \neq b.

From Usucc(R, b, a) we know \( c = a \) which is a contradiction.

Hence \( \exists c. \ (b, c) \in (R \setminus \{(b, a)\}). \)

\text{case 2.} (R \setminus \{(b, a)\}) \circ \{(a, b)\} \subseteq (R^+ \setminus \{(b, a)\} \cup \{(a, b)\}).

We know Upred(R, a, b) holds and hence \( \exists a, b, c. \ R(b, a) \land R(c, a) \neq b \neq a. \)

Hence, \( R \setminus \{(b, a)\} \circ \{(a, b)\} = \emptyset. \)

As a result, \( R \setminus \{(b, a)\} \circ \{(a, b)\} \subseteq (R^+ \setminus \{(b, a)\} \cup \{(a, b)\}). \)

\text{case 3.} \{(a, b)\} \circ (R^+ \setminus \{(b, a)\}) \subseteq (R^+ \setminus \{(b, a)\} \cup \{(a, b)\}).

We know \( \{(a, b)\} \circ R \setminus \{(b, a)\} = \emptyset \) because Usucc(R, a, b) holds, that is,
\[ \exists a, b, c. \ R(a, b) \land R(a, c) \land b = c. \]

As a result, \( \{(a, b)\} \circ R \setminus \{(b, a)\} \subseteq (R^+ \setminus \{(b, a)\} \cup \{(a, b)\}). \)

\text{case 4.} \{(a, b)\} \circ \{(a, b)\} \subseteq (R^+ \setminus \{(b, a)\} \cup \{(a, b)\}).

\{(a, b)\} \circ \{(a, b)\} = \emptyset \) and hence \( \{(a, b)\} \circ \{(a, b)\} \subseteq (R^+ \setminus \{(b, a)\} \cup \{(a, b)\}). \)

\[ \square \]
Now we relate the happens-before relations between the source and target executions. The safe reorderings from Table 1 as follows:

\[
\text{reord}(P_{\text{src}}, P_{\text{tgt}}) \text{ such that } P_{\text{tgt}}(i) \subseteq P_{\text{src}}(i) \cup \{r \cdot b \cdot r' \mid r \cdot a \cdot r' \in P_{\text{src}}(i) \} \land \forall j \neq i. P_{\text{tgt}}(j) = P_{\text{src}}(j)
\]

where \(a = a \cdot b, \beta = b \cdot a\), and \(a, b\) are labels of shared memory accesses or fences.

**Lemma 9.** Suppose

1. \(\text{reord}(P_{\text{src}}, P_{\text{tgt}})\) where the reordering is \(a \cdot b \sim b \cdot a\) and
2. \(X_s \in \text{ex}_{\text{WEAKESTMO}}(G_{\text{src}})\) where \(G_{\text{init}} \rightarrow_P_{\text{src}}, \text{WEAKESTMO}^* G_{\text{src}}\)
3. \(X_t \in \text{ex}_{\text{WEAKESTMO}}(G_{\text{tgt}})\) where \(G_{\text{init}} \rightarrow P_{\text{tgt}}, \text{WEAKESTMO}^* G_{\text{tgt}}\).

Then \(X_s, \text{hb}_{C11} \subseteq (X_t, \text{hb}_{C11} \setminus \{(b, a)\}) \cup \{(a, b)\}\).

**Proof.** We know \(X_s, \text{po} = X_t, \text{po} \cup \{(b, a)\} \cup \{(a, b)\}\). Let \(R = (X_t, \text{po} \cup R')\) where \(R'\) is some other relation independent of \(X_t, \text{po}\). Hence from Lemma 8,

\[
(R \setminus \{(b, a)\}) \cup \{(a, b)\})^+ \subseteq \left(R^+ \setminus \{(b, a)\} \cup \{(a, b)\}\right)
\]

\[
\implies \left((X_t, \text{po} \cup R') \setminus \{(b, a)\} \cup \{(a, b)\} \right)^+ \subseteq \left((X_t, \text{po} \cup R') \setminus \{(b, a)\} \cup \{(a, b)\}\right)
\]

\[
\implies \left((X_t, \text{po} \setminus \{(b, a)\} \cup \{(a, b)\} \cup R')^+ \subseteq \left((X_t, \text{po} \cup R') \setminus \{(b, a)\} \cup \{(a, b)\}\right)
\]

\[
\implies (X_s, \text{po} \cup R')^+ \subseteq \left((X_t, \text{po} \cup R') \setminus \{(b, a)\} \cup \{(a, b)\}\right)
\]

since \((X_s, \text{po} \cup R')^+ = (\text{imm}(X_s, \text{po}) \cup R')^+\) and \((X_t, \text{po} \cup R')^+ = (\text{imm}(X_t, \text{po}) \cup R')^+\),

substituting \(R' = X_s, \text{sw}_{C11} = X_t, \text{sw}_{C11}\) we get

\[
(\text{imm}(X_s, \text{po}) \cup X_s, \text{sw}_{C11})^+ \subseteq \left((X_t, \text{po} \cup X_t, \text{sw}_{C11}) \setminus \{(b, a)\} \cup \{(a, b)\}\right)
\]

It implies \(X_s, \text{hb}_{C11} \subseteq (X_t, \text{hb}_{C11} \setminus \{(b, a)\} \cup \{(a, b)\})\)

as \(X_s, \text{hb}_{C11} = (\text{imm}(X_s, \text{po}) \cup X_s, \text{sw}_{C11})^+\) and \(X_t, \text{hb}_{C11} = (\text{imm}(X_t, \text{po}) \cup X_t, \text{sw}_{C11})^+\). \(\square\)

### F.1 Reordering Theorem

We restate the definition of compilation correctness and the safe reordering theorem.

**Definition 7.** A transformation of program \(P_{\text{src}}\) in memory model \(M_{\text{src}}\) to program \(P_{\text{tgt}}\) in model \(M_{\text{tgt}}\) is **correct** if it does not introduce new behaviors: i.e., \(\text{Behavior}_{M_{\text{tgt}}}(P_{\text{tgt}}) \subseteq \text{Behavior}_{M_{\text{src}}}(P_{\text{src}})\).

**Theorem 6.** The safe reorderings in Table 1 are correct in both \text{WEAKESTMO} models.

The formal statement is as follows:

\[
\forall \exists \text{reord}(P_{\text{src}}, P_{\text{tgt}}) \implies \forall G_{\text{tgt}}. G_{\text{init}} \rightarrow P_{\text{tgt}}, \text{WEAKESTMO}^* G_{\text{tgt}} \cdot \exists G_{\text{src}}. G_{\text{init}} \rightarrow P_{\text{src}}, \text{WEAKESTMO}^* G_{\text{src}} \land \forall X_t \in \text{ex}_{\text{WEAKESTMO}}(G_{\text{tgt}}). \exists X_s \in \text{ex}_{\text{WEAKESTMO}}(G_{\text{src}}). \text{Behavior}(X_t) = \text{Behavior}(X_s) \land X_t, \text{Race} \cap E_{\text{NA}} \neq \emptyset \implies X_s, \text{Race} \cap E_{\text{NA}} \neq \emptyset
\]

To prove the theorem, given an extracted consistent target execution \(X_t \in \text{ex}_{\text{WEAKESTMO}}(G_{\text{tgt}})\) from a consistent target event structure \(G_{\text{tgt}}\), we construct a consistent source execution \(X_s\) from \(X_t\). Then we ensure that the behavior of the \(X_s\) and \(X_t\) are same and if \(X_t\) has undefined behavior due to data race then \(X_s\) also has undefined behavior due to data race. Finally, we show that the \(X_s \in \text{ex}_{\text{WEAKESTMO}}(G_{\text{src}})\) where \(G_{\text{src}}\) is a \text{WEAKESTMO} consistent source event structure.

Proof. In this proof we follow the above mentioned steps as follows.

Source Execution Consistency. From target execution $X_t$ we get source execution $X_s$ by reordering the respective events. Thus if $\text{imm}(X_t, \text{po})(b, a)$ then $\text{imm}(X_t, \text{po})(a, b)$ holds. We know, following the Lemma 9, $X_s, \text{hb} \subseteq X_t \setminus \{(b, a)\} \cup \{(a, b)\}$, that is, $X_s$ is more relaxed than $X_t$. We also know that $X_t$ is consistent. Hence the execution $X_s$ is consistent.

Same Behavior. The behaviors of $X_s$ and $X_t$ are same. The reordering does not introduce any new mo relation in $X_s$ and thus $X_t, \text{mo} = X_s, \text{mo}$. Hence the behaviors of $X_s$ and $X_t$ are same.

Race Preservation.
following the Lemma 9, $X_s, \text{hb} \subseteq X_t, \text{hb} \setminus \{(b, a)\} \cup \{(a, b)\}$. Hence if $X_t$ is racy, then $X_s$ is also racy. As a result, if the target execution has undefined behavior due to a data race, so does the source execution.

Source Event Structure Construction and Execution Extraction
It is left to show that we can construct a source event structure $G_{\text{init}} \rightarrow_{G_{\text{src}}, \text{WEAKEASTMO}} G'_{\text{src}}$ such that execution $X_t$ is an extracted execution from $G_{\text{src}}$, that is, $X_s \in \text{exWEAKEASTMO}(G_{\text{src}})$.

If $(X_s, \text{po} \cup X_s, \text{rf})^+$ is acyclic, then we follow the $(X_s, \text{po} \cup X_s, \text{rf})^+$ path to construct the source event structure and in this case $G_{\text{src}} = X_s$. From the definitions we know that WEAKEASTMO constraints are weaker than the execution constraints. Hence $G_{\text{src}}$ is consistent as $X_s$ is consistent. As a result, $X_s \in \text{exWEAKEASTMO}(G_{\text{src}})$.

However, if $X_s$ has $(X_s, \text{po} \cup X_s, \text{rf})^+$ cycle(s), then we construct $G_{\text{src}}$ and extract $X_s$ from $G_{\text{src}}$.

Source Event Structure Construction. To construct $G_{\text{src}}$, we follow the construction steps of $G_{\text{tgt}}$. For each target construction step that adds event $e$ to $G_{\text{tgt}}$ to get $G'_{\text{tgt}}$, we perform one or more corresponding steps going from $G_{\text{src}}$ to $G'_{\text{src}}$. We do a case analysis on the event $e$ of the target event structure. For the reordered events the construction is as follows:

![Diagram](image.png)

Fig. 34. $\{(c_s, c_t), (b_s, b_t), (a_s, a_t), (b'_s, b'_t), (d_s, d_t)\} \subseteq \mathcal{M}$.

We define $\text{pc} : N \rightarrow E$; a function that maps a thread identifier to an event in the respective thread in the execution.

We use $\mathcal{M}$ to keep track of the $X_s$ in $G_{\text{src}}$.

We define $\mathcal{M}$ relation which pairs a $G_{\text{src}}$ and $G_{\text{tgt}}$ event, that is,

$\mathcal{M} \triangleq \{(s, t) | s \in G_{\text{src}}.E \land t \in G_{\text{tgt}}.E \land s.\text{lab} = t.\text{lab} \land s.\text{tid} = t.\text{tid}\}$

Let $A \subseteq G_{\text{tgt}}.E$, $B \subseteq G_{\text{tgt}}.E$ denote the pair of sets of events which are created for the reordered access pairs.

We call $A \cup B$ as reordered events and $G_{\text{tgt}}.E \setminus (A \cup B)$ as non-reordered events.

Also let $C \subseteq G_{\text{tgt}}.E \setminus (A \cup B)$ be the immediate $G_{\text{tgt}}$.po-predecessors of the $B$ events.

We say $G_{\text{src}} \sim G_{\text{tgt}}$ holds iff

1. $G_{\text{src}}$, $G_{\text{tgt}}$ are consistent.
(2) there exists $M$ such that $G_{\text{src}}$ and $G_{\text{tgt}}$ preserves invariant which is a conjunction of following clauses.

(a) The non-reordered events in the target event structures are mapped to some non-reordered events in the source event structure.

$$\forall c_t \in G_{\text{tgt}}.E \setminus (A \cup B). \exists c_s \in G_{\text{src}}.E. M(c_s, c_t)$$

(b) If $b_t$ is po-successor of some event $c_t$ in the target event structure then there exists $a', b_s, c_s$ events in the source event structure such that $M(b_s, b_t), M(c_s, c_t)$ hold. In addition, memory location and memory order of $a'$ and $a_t$ match.

$$\forall c_t \in G_{\text{tgt}}.E \setminus (A \cup B), a_t \in A, b_t \in B \land G_{\text{tgt}}.po(c_t, b_t) \implies \exists c_s, a_s, b_s \in G_{\text{src}}.E. M(c_s, c_t) \land M(a_t, a_s) \land M(b_s, b_t)$$

(c) If $a_t$ is po-successor of some event $c_t$ in the target event structure then there exists $a_s, c_s$ events in the source event structure such that $M(a_s, a_t)$ and $M(c_s, c_t)$ hold.

$$\forall a_t \in G_{\text{tgt}}.E \setminus (A \cup B). a_t \in A. \land G_{\text{tgt}}.po(c_t, a_t) \implies \exists c_s, a_s \in G_{\text{src}}.E. M(c_s, c_t) \land M(a_s, a_t) \land G_{\text{src}}.po(c_s, a_s)$$

(d) If $a_t \in A$ is immediate-po successor of $b_t \in B$ in the target event structure then there exist $a_s, a', b_s, c_t, c_s$ such that

(i) $\{(c_s, c_t), (b_s, b_t), (a_s, a_t)\} \subseteq M$ holds.

(ii) $c_s$ and $c_t$ are non-reordered events such that if $c_t$ is immediate-po-predecessor of $b_t$ then $c_s$ is immediate-po predecessor of $a_s$.

(iii) $a'$ and $a$ are in immediate-conflict relation.

(iv) $b_s$ and $b'$ are immediate-po successors of $a'$ and $a_s$ respectively.

(v) $b'$ and $b_s$ are equal-writes.

$$\forall a_t \in A, b_t \in B. \text{imm}(G_{\text{tgt}}.po)(b_t, a_t) \implies (\exists c_t \in G_{\text{tgt}}.E \setminus (A \cup B), a', a_s, b_s, c_s \in G_{\text{src}}.E. M(c_s, c_t) \land M(a_t, a_s) \land M(b_s, b_t)$$

$$\land \text{imm}(G_{\text{tgt}}.po)(c_t, b_t) \land \text{imm}(G_{\text{src}}.po)(c_s, a_s) \land \text{imm}(G_{\text{src}}.po)(a_t, b')$$

$$\land b_s.\text{loc} = b'.\text{loc} \land b_s.\text{ord} = b'.\text{ord} \land G_{\text{src}}.\text{ew}(b_s, b')$$

(e) If non-reordered event $c_t$ is po-successor of $b_t$ in the target event structure then there exists $c_s$ in source event structure which maps to $c_t$ and $c_s$ is po-successor of $b'$ or $b_s$ where $b'$ and $b_s$ are equal-writes.

$$\forall c_t \in G_{\text{tgt}}.E \setminus (A \cup B), b_t \in B. G_{\text{tgt}}.po(b_t, c_t) \implies \exists b_s, b', c_s \in G_{\text{src}}.E. M(c_s, c_t) \land M(b_s, b_t) \land M(b_t, b_t)$$

$$\land G_{\text{src}}.\text{ew}(b_s, b') \land (G_{\text{src}}.po(b_s, c_s) \lor G_{\text{src}}.po(b', c_s))$$

(f) If $a_t \in A$ is immediate-po-successor of $b_t \in B$ in the target event structure then there is no po relation between $b_s$ and $a_t$ in source event structure where $a_s$ maps to $a_t$ and $b_s$ maps to $b_t$.

$$\forall a_t \in A, b_t \in B. G_{\text{tgt}}.po(b_t, a_t) \implies \exists a_s, b_s \in G_{\text{src}}.E. M(a_s, a_t) \land M(b_s, b_t) \land \neg G_{\text{src}}.po(b_s, a_s)$$
(g) For a pair of non-ordered events in the target event structure which are in po relation, there exists corresponding pair of events in the source event structure which are in po relation.

\[
\forall c_t, c'_t \in G_{tgt}.E \setminus (A \cup B). G_{tgt}.po(c_t, c'_t) \Rightarrow \\
\exists c_s, c'_s \in G_{src}.E. M(c_s, c_t) \land M(c'_s, c'_t) \land G_{src}.po(c_s, c'_t)
\]

(h) If \(a_t\) is justified from an event \(c_t\) in the target event structure then there exists corresponding \(a_s, c_s\) events in the source event structure such that \(a_s\) is justified from \(c_s\).

\[
\forall c_t \in G_{tgt}.E \setminus (A \cup B), a_t \in A. G_{tgt}.jf(c_t, a_t) \Rightarrow \\
\exists c_s, a_s \in G_{src}.E. M(a_s, a_t) \land M(c_s, c_t) \land G_{src}.jf(c_s, a_s)
\]

(i) If \(a_t\) justifies an event \(c_t\) in the target event structure then there exists corresponding \(a_s, c_s\) events in the source event structure such that \(a_s\) justifies \(c_s\).

\[
\forall c_t \in G_{tgt}.E \setminus (A \cup B), a_t \in A. G_{tgt}.jf(a_t, c_t) \Rightarrow \\
\exists c_s, a_s \in G_{src}.E. M(a_s, a_t) \land M(c_s, c_t) \land G_{src}.jf(a_s, c_s)
\]

(j) If \(b_t\) is justified from an event \(c_t\) in the target event structure then there exists corresponding \(b'_s\) and \(b_s, c_s\) events in the source event structure such that \(c_s\) justifies \(b_s\), \(b'_s\), and \(b_s, b'_s\) are equal-writes.

\[
\forall c_t \in G_{tgt}.E \setminus (A \cup B), b_t \in B. G_{tgt}.jf(c_t, b_t) \Rightarrow \\
\exists b_s, c_s \in G_{src}.E. M(b_s, b_t) \land M(c_s, c_t) \land G_{src}.jf(b_s, b_s) \\
\land (\exists b'_s \in G_{src}.E. M(b'_s, b_t) \land M(c_s, c_t) \land G_{src}.ew(b_s, b'_s) \Rightarrow G_{src}.jf(c_s, b'_s))
\]

(k) If \(b_t\) in the target event structure justifies \(c_t\) then either there exists \(b'_s\) corresponding to \(b_t\) such that \(b'_s\) justifies \(c_s\) where there is no \(b_s\) that maps to \(b_t\) or source event structure has \(b_s\) which is equal-writes to \(b'_s\) and justifies \(c_s\).

\[
\forall c_t \in G_{tgt}.E \setminus (A \cup B), b_t \in B. G_{tgt}.jf(b_t, c_t) \Rightarrow \\
((\exists b_s, c_s \in G_{src}.E. M(b_s, b_t) \land \exists b'_s \in G_{src}.E. M(b'_s, b_t) \land G_{src}.ew(b_s, b'_s)) \\
\Rightarrow G_{src}.jf(b_s, c_s) \\
\lor (\exists b'_s, b_s, c_s \in G_{src}.E. M(b'_s, b_t) \land M(b_s, b_t) \land M(c_s, c_t) \land G_{src}.ew(b_s, b'_s)) \\
\Rightarrow G_{src}.jf(b'_s, c_s)))
\]

(l) If a pair of non-reordered events are in justified-from relation, then there exists corresponding pair of events in the source event structure in justified-from relation.

\[
\forall c_t, c'_t \in G_{tgt}.E \setminus (A \cup B). G_{tgt}.jf(c_t, c'_t) \Rightarrow \\
\exists c_s, c'_s \in G_{src}.E. M(c_s, c_t) \land M(c'_s, c'_t) \land G_{src}.jf(c_s, c'_s)
\]

(m) If there is mo relation from a non-reordered event \(c_t\) to an ordered event \(a_t\) then there exists events \(c_s, a_s\) in mo relation in source event structure where non-reordered event \(c_s\) maps to \(c_t\) and ordered event \(a_s\) maps to \(a_t\).

\[
\forall c_t \in G_{tgt}.E \setminus (A \cup B), a_t, b_t \in B. G_{tgt}.mo(c_t, a_t) \Rightarrow \\
\exists c_s, a_s \in G_{src}.E. M(c_s, c_t) \land M(a_s, a_t) \land G_{src}.mo(c_s, a_s)
\]

(n) If there is mo relation from an ordered event \(a_t\) to a non-reordered event \(c_t\) then there exists mo relation from event \(a_s\) to \(c_s\) in source event structure where ordered event \(a_s\) maps to \(a_t\) and non-reordered event \(c_s\) maps to \(c_t\).

\[
\forall c_t \in G_{tgt}.E \setminus (A \cup B), a_t \in A, G_{tgt}.mo(a_t, c_t) \Rightarrow \\
\exists c_s, a_s \in G_{src}.E. M(c_s, c_t) \land M(a_s, a_t) \land G_{src}.mo(a_s, c_s)
\]
(o) If there is mo relation from a non-reordered event \( c_t \) to an ordered event \( b_t \) then there exists events \( c_s, b_s \) in mo relation in source event structure where non-reordered event \( c_s \) maps to \( c_t \) and ordered event \( b_s \) maps to \( b_t \).

\[
\forall c_t \in G_{tgt}.E \setminus (A \cup B), b_t \in B. G_{tgt}.mo(c_t, b_t) \implies \\
\exists c_s, b_s \in G_{src}.E. M(c_s, c_t) \land M(b_s, b_t) \land G_{src}.mo(c_s, b_s)
\]

(p) If there is mo relation from an ordered event \( b_t \) to a non-reordered event \( c_t \) then there exists mo relation from event \( b_s \) to \( c_s \) in source event structure where ordered event \( b_s \) maps to \( b_t \) and non-reordered event \( c_s \) maps to \( c_t \).

\[
\forall c_t \in G_{tgt}.E \setminus (A \cup B), b_t \in B. G_{tgt}.mo(b_t, c_t) \implies \\
\exists c_s, b_s \in G_{src}.E. M(c_s, c_t) \land M(b_s, b_t) \land G_{src}.mo(c_s, b_s)
\]

(q) If there is mo relation between a pair of non-reordered events \( c_t \) and \( c'_t \) in the target event structure then there exists mo relation from event \( c_s \) to \( c'_s \) in source event structure where \( c_s \) maps to \( c_t \) and \( c'_s \) maps to \( c'_t \).

\[
\forall c_t, c'_t \in G_{tgt}.E \setminus (A \cup B). G_{tgt}.mo(c_t, c'_t) \implies \\
\exists c_s, c'_s \in G_{src}.E. M(c_s, c_t) \land M(c'_s, c'_t) \land G_{src}.mo(c_s, c'_s)
\]

(r) If an event is unmapped in the source event structure then there is no outgoing mo edge from that event.

\[
\forall e_s \in G_{src}.E. (\# e_t \in G_{tgt}.E. M(e_s, e_t)) \implies \\
\# e'_s \in G_{src}.E. G_{src}.mo(e_s, e'_s)
\]

(s) For each equal-writes pair of events in the target event structure, there exists equal-writes pairs in the source event structure.

\[
\forall c_t, c'_t \in G_{tgt}.E. G_{tgt}.ew(c_t, c'_t) \implies \\
\exists c_s, c'_s \in G_{src}.E. M(c_s, c_t) \land M(c'_s, c'_t) \land G_{src}.ew(c_s, c'_s)
\]

(3) there exists pc such that

\[
X_s.E = \emptyset \\
X_s.po = G_{src}.po \cap (\mathbb{S} \times \mathbb{S}) \\
X_s.rf = G_{src}.rf \cap (\mathbb{S} \times \mathbb{S}) \\
X_s.mo = G_{src}.mo \cap (\mathbb{S} \times \mathbb{S})
\]

where \( \mathbb{S}(G_{src}.pc) \triangleq \{ e \mid e \in G_{src}.E \land G_{src}.po^t(e, pc(e.tid)) \} \). To prove the simulation we show the followings.

\[
G_{src} \sim G_{tgt} \land G_{tgt} \xrightarrow{\text{WEAKESTMO}} G'_{tgt} \implies \exists G'_{src}, G_{src} \xrightarrow{\text{WEAKESTMO}} G'_{src} \land G'_{src} \sim G'_{tgt}
\]

At each construction step, we extend \( G_{tgt} \) to \( G'_{tgt} \) by po-extending from an event \( e_t \in G_{tgt}.E \) with a new event \( e'_t \in G'_{tgt}.E \). We consider following cases:

**Case** \( e'_t \in B' \) where \( B' = B \cup \{ e'_t \} \):

In this case \( A' = A, \) and \( G'_{tgt}.E = G_{tgt}.E \cup \{ e'_t \} \).

We also append corresponding event(s) in \( G_{src} \) and construct \( G'_{src} \).

(1) Condition to show: \( G'_{src} \) is consistent.

The construction has two steps: \( G_{src} \rightarrow G''_{src} \rightarrow G'_{src} \). In \( G''_{src} \) we introduce \( a'' \) and in \( G'_{src} \) we introduce \( e'_t \).

**case.** event \( e_s \) has an immediate po successor \( a'' \) such that \( a.loc = a''.loc \) and \( a.ord = a''.ord \).

In this case \( a' = a'' \) and \( G''_{src} = G_{src} \).

**otherwise.**
We append an event \( a' \) in \( G_{\text{src}} \) and create \( G'_{\text{src}} \) such that
\[
G''_{\text{src}}.E = G_{\text{src}}.E \uplus \{ a' \} \\
G''_{\text{src}}.po = (G_{\text{src}}.po \uplus \{(e_s, a') \mid \overline{M}(e_s, e_t)) \}^+ \\
G''_{\text{src}}.jf = G_{\text{src}}.jf \\
\uplus \{(w, a') \mid (w, a') \in (G''_{\text{src}}.W \times G''_{\text{src}}.R) \}
\wedge \exists w' \in G'_{\text{tgt}}.E. \overline{M}(w, w') \wedge G_{\text{tgt}}.jf(w', a) \}
\uplus \{(w, a') \mid (w, a') \in (G''_{\text{src}}.W \times G''_{\text{src}}.R) \}
\wedge \nexists w' \in G'_{\text{tgt}}.E. \overline{M}(w, w') \wedge G_{\text{tgt}}.jf(w', a) \wedge \nexists W(G''_{\text{src}}, w, a') \}
\]
\[
G''_{\text{src}}.mo = G_{\text{src}}.mo \uplus \{(w, a') \mid (w, a') \in (G''_{\text{src}}.W \times G''_{\text{src}}.W) \}
\]
\[
G''_{\text{src}}.ew = G_{\text{src}}.ew
\]

Also in this case \( \overline{M}'' = \overline{M} \).
Now we check whether \( G''_{\text{src}} \) is consistent.
We know that \( G_{\text{tgt}} \sim G_{\text{src}} \). Hence \( G_{\text{src}} \) and \( G_{\text{tgt}} \) are consistent.

If \( G''_{\text{src}} = G_{\text{src}} \) then \( G''_{\text{src}} \) is consistent as \( G_{\text{src}} \) is consistent.
Otherwise, from definition of \( G''_{\text{src}} \) and observation from Remark 3 we know that \( G''_{\text{src}} \) satisfies (CF), (CFJ), (VIS), (ICF). (ICF).

There is no outgoing edge from \( a' \) and hence it does not result in any \( (G''_{\text{src}}.hb; G''_{\text{src}}.eco) \) cycle. Hence \( G''_{\text{src}} \) satisfies (COH').

As a result, \( G''_{\text{src}} \) remains consistent.
Next, we construct \( G'_{\text{src}} \) from \( G''_{\text{src}} \).

**case.** There exists \( e'_s \) where \( e'_s.lab = e'_t.lab \) and if \( e'_s, e'_t \in R \) then \( G''_{\text{src}}.jf(w_s, e'_s), G''_{\text{src}}.jf(w_t, e'_t), \overline{M}''(w_s, w_t) \) hold.

In this case \( G'_{\text{src}} = G''_{\text{src}} \) and \( b_s = e'_s \).

**Otherwise.** We append such a \( e'_s \) and thus
\[
G'_{\text{src}}.E = G''_{\text{src}}.E \uplus \{ e'_s \mid e'_s.lab = e'_t.lab \} \\
G'_{\text{src}}.po = (G''_{\text{src}}.po \uplus \{(a', e'_s) \})^+ \\
G'_{\text{src}}.jf = G''_{\text{src}}.jf \\
\uplus \{(w_s, e'_s) \mid (w_s, e'_s) \in (G''_{\text{src}}.W \times G''_{\text{src}}.R) \wedge G_{\text{tgt}}.jf(w_t, e'_t) \wedge \overline{M}''(w_s, w_t) \}
\]
\[
G'_{\text{src}}.mo = G''_{\text{src}}.mo \\
\uplus \{(w_s, e'_s) \mid (w_s, e'_s) \in (G''_{\text{src}}.W \times G''_{\text{src}}.W) \wedge \nexists (w_s, w_t) \wedge G_{\text{tgt}}.mo(w_t, e'_t) \}
\uplus \{(e'_s, w_s) \mid (w_s, e'_s) \in (G''_{\text{src}}.W \times G''_{\text{src}}.W) \wedge \nexists (w_s, w_t) \wedge G_{\text{tgt}}.mo(w'_t, e'_t) \}
\]
\[
G'_{\text{src}}.ew = G''_{\text{src}}.ew \uplus \{(w_s, e'_s), (e'_s, w_s) \mid (w_s, e'_s) \in (G''_{\text{src}}.W_{\leq \text{RLX}} \times G''_{\text{src}}.W_{\leq \text{RLX}}) \wedge \nexists (w_s, w_t) \wedge G_{\text{tgt}}.ew(w_t, e'_t) \}
\]

Also in this case \( M' = M'' \uplus \{(e'_s, e'_t)\} \).
Now we check whether \( G'_{\text{src}} \) is consistent.
If \( G'_{\text{src}} = G''_{\text{src}} \) then \( G_{\text{src}} \) is consistent as \( G''_{\text{src}} \) is consistent.
Otherwise, we check whether \( G'_{\text{src}} \) is consistent.

We know $G'_\text{src}$ and $G'_\text{tgt}$ preserve (CF). As a result, from the construction $(e'_s, e'_t) \notin G'_\text{src}.ecf$. Hence $G'_\text{src}$ preserves (CF).

We know $G'_\text{src}$ preserves (CFJ). Moreover, $G'_\text{tgt}.jf(w_t, e'_t)$ implies $\neg G'_\text{tgt}.ecf(w_t, e'_t)$. As a result, from the construction $\neg G'_\text{src}.ecf(w_s, e'_s)$ where $\mathcal{M}''(w_s, w_t)$ holds. Hence $G'_\text{src}$ preserves (CFJ).

We know $G'_\text{src}$ preserves (VISJ). Moreover, $G'_\text{tgt}.jf(w_t, e'_t)$ implies $w_t \in vis(G'_\text{tgt})$. As a result, from the construction $w_s \in vis(G'_\text{src})$ where $\mathcal{M}''(w_s, w_t)$ holds. Hence $G'_\text{src}$ preserves (VISJ).

We know $G'_\text{src}$ and $G'_\text{tgt}$ preserves (ICF). hence following the construction we know if $e'_s \notin G'_\text{src}.\mathcal{R}$ then there exists no event $e_1$ such that $G'_\text{src} \sim (e'_s, e_1)$. Hence $G'_\text{src}$ preserves (ICF).

We know $G'_\text{src}$ preserves (ICF). Moreover, following the construction of $G'_\text{src}$ from $G''_\text{src}$, $(w_s, w_t) \notin G'_\text{src}.jf; \text{imm}(cf); G'_\text{src}.rf^{-1}$. Hence $G'_\text{src}$ preserves (ICF).

We know $G''_\text{src}$ preserves (COH') and consider there is a $(G'_\text{src}.hb; G'_\text{src}.eco^-)$ cycle. In that case $e'_s$ is part of the $(G'_\text{src}.hb; G'_\text{src}.eco^-)$ cycle. However, following the construction of $G'_\text{src}$ in this case, there exists a $(G'_\text{src}.hb; G'_\text{src}.eco^-)$ cycle. This is not possible as $G'_\text{tgt}$ is consistent. Hence a contradiction and $G'_\text{src}$ preserves (COH'). As a result, $G'_\text{src}$ is consistent.

Thus finally $\mathcal{M}' = \mathcal{M} \cup \{(e'_s, e'_t)\}$ and $pc' = pc$.

(2) Condition to show: the simulation invariant holds between $G'_\text{src}$ and $G'_\text{tgt}$

(a) \[ \forall e_t \in G'_\text{tgt}.E \setminus (A' \cup B'), \exists e_s \in G'_\text{src}.E. \mathcal{M}'(e_s, e_t) \]

We know this condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. Hence the condition holds between $G'_\text{src}$ and $G'_\text{tgt}$ as $e'_t \notin G'_\text{tgt}.E \setminus (A' \cup B')$.

(b) \[ \forall e_t \in G'_\text{tgt}.E \setminus (A' \cup B'), a_t \in A', b_t \in B' \land G'_\text{tgt}.po(c_t, b_t) \implies \\
\exists e_s, a_s, b_s \in G'_\text{src}.E. \mathcal{M}'(e_s, c_t) \land \mathcal{M}'(a_s, a_t) \land \mathcal{M}'(b_s, b_t) \land (\exists a'' \in G'_\text{src}.E. a_s, \text{loc} = a'' \text{.loc} \land a_s, \text{ord} = a'' \text{.ord} \land G'_\text{src}.po(e_s, a'') \land \text{imm}(G'_\text{src}.po(a'', b_s))) \]

We know this condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. Considering the definitions of $G'_\text{src}$, $G'_\text{tgt}$, and $\mathcal{M}'$ the condition holds between $G'_\text{src}$ and $G'_\text{tgt}$ where $b_t = e'_t$, $b_s = e'_s$, and $a'' = a'$.

(c) \[ \forall e_t \in G'_\text{tgt}.E \setminus (A' \cup B'), a_t \in A', b_t \in B' \land G'_\text{tgt}.po(c_t, a_t) \implies \\
\exists e_s, a_s \in G'_\text{src}.E. \mathcal{M}'(e_s, c_t) \land \mathcal{M}'(a_s, a_t) \land G'_\text{src}.po(e_s, a_s) \]

We know this condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. Considering the definitions of $G'_\text{src}$, $G'_\text{tgt}$, $\mathcal{M}'$ this condition holds between $G'_\text{src}$ and $G'_\text{tgt}$ for all $e'_t$, $e'_s$, $a'$.

(d) \[ \forall a_t \in A', b_t \in B', \text{imm}(G'_\text{tgt}.po)(b_t, a_t) \implies \\
(\exists e_t \in G'_\text{tgt}.E \setminus (A' \cup B'), a'_t, b'_t, e'_s \in G'_\text{src}.E. \mathcal{M}'(e_s, c_t) \land \mathcal{M}'(a_s, a_t) \land \mathcal{M}'(b_s, b_t) \land \text{imm}(G'_\text{tgt}.po)(e_t, b_t) \land \text{imm}(G'_\text{src}.po)(a_s, a'_t) \land \text{imm}(G'_\text{src}.po)(e_s, a'_t) \land \text{imm}(G'_\text{src}.po)(a'_t, b'_t) \land \text{imm}(G'_\text{src}.efw)(b_s, b'_t) \land G'_\text{src}.e\langle a'_t, b'_t \rangle) \]

We know this condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. The event $e'_t$ is $G'_\text{tgt}.po$-maximal and hence $\text{imm}(G'_\text{tgt}.po)(b_t, a_t)$ does not hold when $b_t = e'_t$. Hence the condition holds between $G'_\text{src}$ and $G'_\text{tgt}$.
(e) 
\[ \forall \tau_{ij} \in G'_{tgt}.E \setminus (A' \cup B'), b_i \in B', G'_{tgt}.po(b_i, \tau_{ij}) \implies \exists b_s, b'_s, c_s \in G'_src.E. M'(c_s, \tau_{ij}) \land M'(b'_s, b_i) \land (G'_{src}.po(b_s, b'_s) \lor G'_{src}.po(b'_s, c_s)) \]

We know this condition holds between \( G_{src} \) and \( G_{tgt} \). The event \( e'_t \) is \( G'_{tgt}.po \)-maximal and hence \( G'_{tgt}.po(b_t, \tau_{ij}) \) does not hold when \( b_t = e'_t \). Hence the condition holds between \( G'_{src} \) and \( G'_{tgt} \).

(f) 
\[ \forall a_t \in A', b_t \in B', G'_{tgt}.po(b_t, a_t) \implies \exists a_s, b_s \in G'_{src}.E. M'(a_s, a_t) \land M'(b_s, b_t) \land \neg G'_{src}.po(b_s, a_s) \]

We know this condition holds between \( G_{src} \) and \( G_{tgt} \). The event \( e'_t \) is \( G'_{tgt}.po \)-maximal and hence \( G'_{tgt}.po(b_t, a_t) \) does not hold when \( b_t = e'_t \). Hence the condition holds between \( G_{src} \) and \( G_{tgt} \).

(g) 
\[ \forall \tau_{ij} \in G'_{tgt}.E \setminus (A' \cup B'), G'_{tgt}.po(\tau_{ij}, e'_t) \implies \exists c_s, c'_s \in G'_{src}.E. M'(c_s, \tau_{ij}) \land M'(c'_s, e'_t) \land G'_{src}.po(c_s, c'_s) \]

We know the condition holds between \( G_{src} \) and \( G_{tgt} \). Considering the definitions of \( G'_{src} \), \( G'_{tgt} \), \( M' \), the condition holds between \( G'_{src} \) and \( G'_{tgt} \) as \( e'_t \notin G'_{tgt}.E \setminus (A' \cup B') \).

(h) 
\[ \forall \tau_{ij} \in G'_{tgt}.E \setminus (A' \cup B'), a_t \in A', G'_{tgt}.jf(\tau_{ij}, a_t) \implies \exists c_s, a_s \in G'_{src}.E. M'(a_s, a_t) \land M'(c_s, \tau_{ij}) \land G'_{src}.jf(c_s, a_s) \]

We know the condition holds between \( G_{src} \) and \( G_{tgt} \). Considering the definitions of \( G'_{src} \), \( G'_{tgt} \), \( M' \), the condition holds between \( G'_{src} \) and \( G'_{tgt} \) as \( e'_t \notin G'_{tgt}.E \setminus (A' \cup B') \) or \( e'_t \notin A \).

(i) 
\[ \forall \tau_{ij} \in G'_{tgt}.E \setminus (A' \cup B'), a_t \in A', G'_{tgt}.jf(\tau_{ij}, a_t) \implies \exists c_s, a_s \in G'_{src}.E. M'(a_s, a_t) \land M'(c_s, \tau_{ij}) \land G'_{src}.jf(a_s, c_s) \]

We know the condition holds between \( G_{src} \) and \( G_{tgt} \). Considering the definitions of \( G'_{src} \), \( G'_{tgt} \), \( M' \), the condition holds between \( G'_{src} \) and \( G'_{tgt} \) as \( e'_t \notin G'_{tgt}.E \setminus (A' \cup B') \) and \( e'_t \notin A \).

(j) 
\[ \forall \tau_{ij} \in G'_{tgt}.E \setminus (A' \cup B'), b_i \in B', G'_{tgt}.jf(\tau_{ij}, b_i) \implies \exists b_s, c_s \in G'_{src}.E. M'(b_s, b_i) \land M'(c_s, \tau_{ij}) \land G'_{src}.jf(b_s, c_s) \land (\exists b'_t \in G'_{src}.E. M'(b'_t, b_i) \land G'_{src}.ew(b_s, b'_t) \implies G'_{src}.jf(b_s, c_s)) \]

We know the condition holds between \( G_{src} \) and \( G_{tgt} \). Considering the definitions of \( G'_{src} \), \( G'_{tgt} \), \( M' \), the condition holds between \( G'_{src} \) and \( G'_{tgt} \) where \( b_s = e'_t \) and there exists no \( b' \) such that \( M'(b_s, b'_t) \).

(k) 
\[ \forall \tau_{ij} \in G'_{tgt}.E \setminus (A' \cup B'), b_i \in B'. G'_{tgt}.jf(b_i, \tau_{ij}) \implies (\exists b_s, c_s \in G'_{src}.E. M'(b_s, b_i) \land \exists b'_t \in G'_{src}.E. M'(b'_t, b_i) \land G'_{src}.ew(b_s, b'_t) \implies G'_{src}.jf(b_s, c_s)) \lor (\exists b'_t, b_s, c_s \in G'_{src}.E. M'(b'_t, b_i) \land M'(b'_t, b_i) \land M'(c_s, \tau_{ij}) \land G'_{src}.ew(b_s, b'_t) \implies G'_{src}.jf(b'_t, c_s))) \]
We know the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. Considering the definitions of $G'_{\text{src}}$, $G'_{\text{tgt}}$, $\mathcal{M}'$, the condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$ where $b_s = e'_t$ and there exists no $b' \in G'_{\text{src}}.E$ such that $\mathcal{M}'(b_s, b'_t)$ and $G_{\text{src}}.Ew(b_s, b')$ holds.

\[(l)\]
\[
\forall c_t, c'_t \in G'_{\text{tgt}}.E \setminus (A' \cup B'). G'_{\text{tgt}}.jf(c_t, c'_t) \implies \\
\exists e_s, e'_s \in G'_{\text{src}}.E \mathcal{M}'(e_s, c_t) \land \mathcal{M}'(e'_s, c'_t) \land G'_{\text{src}}.jf(e_s, e'_s)
\]

We know the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. Considering the definitions of $G'_{\text{src}}$, $G'_{\text{tgt}}$, $\mathcal{M}'$, the condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$ as $e'_t \notin G'_{\text{tgt}}.E \setminus (A' \cup B')$.

\[(m)\]
\[
\forall c_t \in G'_{\text{tgt}}.E \setminus (A' \cup B'), a_t \in A', b_t \in B'. G'_{\text{tgt}}.mo(a_t, c_t) \implies \\
\exists c_s, a_s \in G'_{\text{src}}.E. \mathcal{M}'(c_s, c_t) \land \mathcal{M}'(a_s, a_t) \land G'_{\text{src}}.mo(a_s, c_s)
\]

We know the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. Considering the definitions of $G'_{\text{src}}$, $G'_{\text{tgt}}$, $\mathcal{M}'$, the condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$ as $e'_t \notin G'_{\text{tgt}}.E \setminus (A' \cup B')$ and for all $a_t \in A'$. $\neg \mathcal{M}'(a'_t, a_t)$ holds.

\[(n)\]
\[
\forall c_t \in G'_{\text{tgt}}.E \setminus (A' \cup B'), a_t \in A', c_t \in A'. G'_{\text{tgt}}.mo(a_t, c_t) \implies \\
\exists c_s, a_s \in G'_{\text{src}}.E. \mathcal{M}'(c_s, c_t) \land \mathcal{M}'(a_s, a_t) \land G'_{\text{src}}.mo(a_s, c_s)
\]

We know the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. Following the definitions of $G'_{\text{src}}$ and $G'_{\text{tgt}}$, $\mathcal{M}'$, the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$ where $b_t = e'_t$ and $b_s = e'_s$.

\[(o)\]
\[
\forall c_t \in G'_{\text{tgt}}.E \setminus (A' \cup B'), b_t \in B'. G'_{\text{tgt}}.mo(c_t, b_t) \implies \\
\exists c_s, b_s \in G'_{\text{src}}.E. \mathcal{M}'(c_s, c_t) \land \mathcal{M}'(b_s, b_t) \land G_{\text{src}}.mo(c_s, b_s)
\]

We know the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. Following the definitions of $G'_{\text{src}}$, $G'_{\text{tgt}}$, $\mathcal{M}'$, the condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$ where $b_t = e'_t$ and $b_s = e'_s$.

\[(p)\]
\[
\forall c_t \in G'_{\text{tgt}}.E \setminus (A' \cup B'), b_t \in B'. G'_{\text{tgt}}.mo(b_t, c_t) \implies \\
\exists c_s, b_s \in G'_{\text{src}}.E. \mathcal{M}'(c_s, c_t) \land \mathcal{M}'(b_s, b_t) \land G'_{\text{src}}.mo(b_s, c_s)
\]

We know the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. Following the definitions of $G'_{\text{src}}$, $G'_{\text{tgt}}$, $\mathcal{M}'$, the condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$ as $e'_t \notin G'_{\text{tgt}}.E \setminus (A' \cup B')$.

\[(q)\]
\[
\forall c, c' \in G'_{\text{tgt}}.E \setminus (A' \cup B'). G'_{\text{tgt}}.mo(c_t, c'_t) \implies \\
\exists c_s, c'_s \in G'_{\text{src}}.E. \mathcal{M}'(c_s, c_t) \land \mathcal{M}'(c'_s, c'_t) \land G'_{\text{src}}.mo(c_s, c'_s)
\]

We know the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. Following the definitions of $G'_{\text{src}}$, $G'_{\text{tgt}}$, $\mathcal{M}'$, the condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$ as $e'_t \notin G'_{\text{tgt}}.E \setminus (A' \cup B')$.
We know the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. Following the definitions of $G'_{\text{src}}$, $G'_{\text{tgt}}$, $M'$, the condition holds where $o_s = a'$. 

(s)  
\[ \forall c_t, c'_t \in G'_{\text{tgt}}.E. G'_{\text{tgt}}.e_w(c_t, c'_t) \implies \exists c_s, c'_s \in G'_{\text{src}}.E. \mathbb{M}'(c_s, c_t) \land \mathbb{M}'(c'_s, c'_t) \land G_{\text{src}}.e_w(c_s, c'_s) \]

We know the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. Following the definitions of $G'_{\text{src}}$, $G'_{\text{tgt}}$, $M'$, the condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$ where $c_t = e'_t$ or $c'_t = e'_t$ and $c_s = e'_s$ and $c'_s = e'_s$.

Hence the invariant holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$.

(3) Condition to show:

Case $e'_t \in A$ where $A' = A \cup \{ e'_t \}$:

The construction has two steps: $G_{\text{src}} \rightarrow G''_{\text{src}} \rightarrow G'_{\text{src}}$. In $G''_{\text{src}}$ we introduce $e'_s$ and in $G'_{\text{src}}$ we introduce $b'$. 

In this case $B' = B$, and $G'_{\text{tgt}}.E = G_{\text{tgt}}.E \cup \{ e'_t \}$. 

Let $c_t \in C$ be the immediate $G_{\text{tgt}}.\text{po-predecessor}$ of $e_t$, that is, $\text{imm}(G_{\text{tgt}}.\text{po})(c_t, e_t)$.

In $G_{\text{src}}$ the event $c_s$ is the corresponding event of $c_t$, that is, $\mathbb{M}(c_s, c_t)$.

We also append corresponding event(s) in $G_{\text{src}}$ and construct $G'_{\text{src}}$.

(1) Condition to show: $G'_{\text{src}}$ is consistent.

**case.** event $e'_s$ has an immediate $\text{po}$ successor $a''$ such that $e'_s.\text{lab} = a''.\text{lab}$ and if $e'_t \in R$ and $G'_{\text{tgt}}.jf(w_t, e'_t)$ then there exists $w_s$ such that $\mathbb{M}(w_s, w_t)$ and $G_{\text{src}}.jf(w_s, a'')$.

In this case $e'_s = a''$ and $G''_{\text{src}} = G_{\text{src}}$.

**otherwise.**
We append an event e′ s in G src by po-extending from e s and create G′′ src such that

\[
G′′_{src}.E = G_{src}.E \cup \{e'_s\}
\]

\[
G′′_{src}.po = (G_{src}.po \uplus \{(e_s, e'_s) \mid M(e_s, e_t)\})^+
\]

\[
G′′_{src}.jf = G_{src}.jf \uplus \{(w_s, e'_s) \mid (w_s, e'_s) \in (G′′_{src}.W \times G′′_{src}.R) \land G'_tgf.jf(w_t, e'_t) \land M(w_s, w_t)\}
\]

\[
G′′_{src}.mo = G_{src}.mo \uplus \{(w_s, e'_s) \mid (w_s, e'_s) \in (G′′_{src}.W \times G′′_{src}.W) \land M(w_s, w_t) \land G'_tgf.mo(w_t, e'_t)\}
\]

\[
G′′_{src}.ew = G_{src}.ew \uplus \{(w_s, e'_s), (e'_s, w_s) \mid (w_s, e'_s) \in (G′′_{src}.W_{RLX} \times G′′_{src}.W_{RLX}) \land M(w_s, w_t) \land G'_tgf.ew(w_t, e'_t)\}
\]

Also in this case M′′ = M \uplus \{(e_s, e'_s)\}.

Now we check whether G′′ src is consistent.

We know that G_tgt \sim G src and hence G src and G_tgt are consistent. Now we check whether G′′ src is consistent.

If G′′ src = G src then G′′ src is consistent as G src is consistent. Otherwise.

We know that G src preserves (ICFJ). Also from the construction of G′′ src, we know there is no G′′ src.jf(e'_s, -). Hence G′′ src preserves (ICFJ).

We know that G src preserves (CF), (CFJ), (VISJ), (CFJ). Also G'_tgf.jf(w_t, e'_t) implies e'_s \in R, w_t \in vis(G'_tgf) and \neg G'_tgf.ecf(w_t, e'_t), and M(w_s, w_t) holds. Following the construction, w_s \in vis(G′′ src), \neg G′′ src.ecf(w_s, e'_s) holds. Hence G′′ src preserves (CF), (CFJ), (VISJ), (ICFJ).

We know G src preserves (COH'). Consider there is (G′′ src.hb; G′′ src.eco s) cycle in G′′ src and e'_s is a part of this cycle. In that case there is a (G'_tgf.hb; G'_tgf.eco s) cycle in G'_tgf and e'_s is a part of the cycle. However, G'_tgf preserves (COH') and hence there is no (G'_tgf.hb; G'_tgf.eco s) cycle. Hence a contradiction and G′′ src preserves (COH').

As a result, G′′ src is consistent.

Next, we construct G′ src from G′′ src where we identify or create e'_s.

case. There exists e'_s where e'_s.lab = e'_s.lab and if e'_s, e'_s \in R, then G′′ src.jf(w_s, e'_s) and G′′ src.jf(w_t, e'_s) and M′′(w_s, w_t) hold.

In this case G′ src = G′′ src.

Otherwise. We append such a e'_s = b' and thus
\[G'_{\text{src}} \cdot \mathcal{E} = G''_{\text{src}} \cdot \mathcal{E} \cup \{b' \mid b'.\text{lab} = e_t.\text{lab}\}\]

\[G'_{\text{src}} \cdot \text{po} = (G''_{\text{src}} \cdot \text{po} \cup \{(e'_s, b')\})^+\]

\[G'_{\text{src}} \cdot \text{jf} = G''_{\text{src}} \cdot \text{jf} \cup \{(w_s, b') \mid (w_s, b') \in (G'_{\text{src}} \cdot \mathcal{W} \times G'_{\text{src}} \cdot \mathcal{R}) \land G'_{\text{targ}} \cdot \text{jf}(w_t, e_t)\}
\land \mathcal{M}''(w_s, w_t) \land \neg G''_{\text{src}}.\text{cf}(w_s, e_s)\}\]

\[G'_{\text{src}} \cdot \text{mo} = G''_{\text{src}} \cdot \text{mo} \cup \{(w_s, b') \mid (w_s, b') \in (G'_{\text{src}} \cdot \mathcal{W} \times G'_{\text{src}} \cdot \mathcal{W})\}
\land \mathcal{M}''(w_s, w_t) \land G'_{\text{targ}}.\text{mo}(w_t, e_t) \land \neg G''_{\text{src}}.\text{cf}(w_s, b')\}\]

\[\forall \{(w_s, b'), (b', w_s) \mid (w_s, b') \in (G'_{\text{src}} \cdot \mathcal{W}_{\text{RLX}} \times G'_{\text{src}} \cdot \mathcal{W}_{\text{RLX}}) \land \mathcal{M}''(w_s, e_t)\}\]

Also in this case \(\mathcal{M}' = \mathcal{M}'' \cup \{(e'_s, e'_t)\}\).

Now we check whether \(G'_{\text{src}}\) is consistent.

If \(G'_{\text{src}} = G''_{\text{src}}\) then \(G'_{\text{src}}\) is consistent as \(G''_{\text{src}}\) is consistent.

Otherwise we check the consistency of \(G'_{\text{src}}\).

We know \(G'_{\text{src}}\) and \(G'_{\text{targ}}\) preserve (CF). As a result, from the construction \((e'_t, e'_s) \notin G'_{\text{src}}.\text{ecf}\). Hence \(G'_{\text{src}}\) preserves (CF).

We know \(G'_{\text{src}}\) preserves (CFJ). Moreover, \(G'_{\text{targ}} \cdot \text{jf}(w_t, e'_t)\) implies \(\neg G'_{\text{targ}}.\text{cf}(w_t, e'_t)\). As a result, from the construction \(\neg G'_{\text{src}}.\text{cf}(w_s, e'_t)\) where \(\mathcal{M}''(w_s, w_t)\) holds. Hence \(G'_{\text{src}}\) preserves (CFJ).

We know \(G''_{\text{src}}\) preserves (VISJ). Moreover, \(G'_{\text{targ}} \cdot \text{jf}(w_t, e_t)\) implies \(w_t \in \text{vis}(G'_{\text{targ}})\). As a result, from the construction \(w_s \in \text{vis}(G'_{\text{src}})\) where \(\mathcal{M}''(w_s, w_t)\) holds. Hence \(G'_{\text{src}}\) preserves (VISJ).

We know \(G'_{\text{src}}\) and \(G'_{\text{targ}}\) preserves (ICF). Hence following the construction we know that \(G'_{\text{src}}\) preserves (ICF).

We know that \(G'_{\text{src}}\) preserves (ICF). Also from the construction of \(G'_{\text{src}}\), we know there is no \(G'_{\text{src}}.\text{jf}(e'_t, e'_t)\). Hence \(G'_{\text{src}}\) preserves (ICF).

We know \(G'_{\text{src}}\) preserves (COH') and consider there is a \((G'_{\text{src}}.\text{hb}; G'_{\text{src}}.\text{eco})\) cycle. In that case \(b'\) is part of the \((G'_{\text{src}}.\text{hb}; G'_{\text{src}}.\text{eco})\) cycle. However, following the construction of \(G'_{\text{src}}\), in this case, there exists a \((G'_{\text{targ}}.\text{hb}; G'_{\text{targ}}.\text{eco})\) cycle. This is not possible as \(G'_{\text{targ}}\) is consistent. Hence a contradiction and \(G'_{\text{src}}\) preserves (COH'). As a result, \(G'_{\text{src}}\) is consistent.

Thus finally \(\mathcal{M}' = \mathcal{M} \cup \{(e'_t, e'_s), (b', e_t)\}\) and \(\text{pc}' = \text{pc}[e_s, \text{tid} \mapsto b']\).

(2) Condition to show: the simulation invariant holds between \(G'_{\text{src}}\) and \(G'_{\text{targ}}\)

(a) \[\forall c_t \in G'_{\text{targ}} \cdot \mathcal{E} \setminus (A' \cup B'). \exists c_s \in G'_{\text{src}} \cdot \mathcal{E}. \mathcal{M}'(c_s, c_t)\]

In this case \(e'_t, e_t \notin G'_{\text{targ}} \cdot \mathcal{E} \setminus (A' \cup B')\). Hence the condition holds.

(b) \[\forall c_t \in G'_{\text{targ}} \cdot \mathcal{E} \setminus (A' \cup B'), a_t \in A', b_t \in B' \land G'_{\text{targ}} \cdot \text{po}(c_t, b_t) \implies \exists c_s, a_s, b_s \in G'_{\text{src}} \cdot \mathcal{E}. \mathcal{M}'(c_s, c_t) \land \mathcal{M}'(a_s, a_t) \land \mathcal{M}'(b_s, b_t)\]
\[\land (\exists a'' \in G'_{\text{src}} \cdot \mathcal{E}. a_s.\text{loc} = a''.\text{loc} \land a_s.\text{ord} = a''.\text{ord} \land G'_{\text{src}} \cdot \text{po}(a'', b_s))\]
We know this condition holds in $G_{\text{src}}$ and $G_{\text{tgt}}$. Considering the definitions of $G'_{\text{src}}$, $G'_{\text{tgt}}$, and $\mathcal{M}'$ the condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$ where $e_t \notin G'_{\text{tgt}}.E \setminus (A' \cup B')$ and $e'_t \notin B'$.

(c)

$$\forall c_t \in G'_{\text{tgt}}.E \setminus (A' \cup B'), a_t \in A', \land G'_{\text{tgt}}.\text{po}(c_t, a_t) \implies$$
$$\exists c_s, a_s \in G'_{\text{src}}.E, \mathcal{M}'(c_s, c_t) \land \mathcal{M}'(a_s, a_t) \land G'_{\text{src}}.\text{po}(c_s, a_s)$$

We know this condition holds in $G_{\text{src}}$ and $G_{\text{tgt}}$. Considering the definitions of $G'_{\text{src}}, G'_{\text{tgt}}, \mathcal{M}'$ this condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$ for $a_t = e'_t$ and $a_s = e'_s$.

(d)

$$\forall a_t \in A', b_t \in B'. \text{imm}(G'_{\text{tgt}}.\text{po})(b_t, a_t) \implies$$
$$\exists c_t \in G'_{\text{tgt}}.E \setminus (A' \cup B'), a_t, b_s, c_s \in G'_{\text{src}}.E, \mathcal{M}'(c_s, c_t) \land \mathcal{M}'(a_t, a_t) \land \mathcal{M}'(b_x, b_t)$$
$$\land \mathcal{G}'_{\text{src}}.\text{cf}(a_s, a') \land \mathcal{M}'(G'_{\text{src}}.\text{po})(a', b_s)$$
$$\land b_s, \text{loc} = b'. \text{loc} \land b_s, \text{ord} = b'. \text{ord} \land G'_{\text{src}}.\text{ew}(b_s, b')$$

We know this condition holds in $G_{\text{src}}$ and $G_{\text{tgt}}$. Considering the definitions of $G'_{\text{src}}, G'_{\text{tgt}}, \mathcal{M}'$ we have $b_1 = e_t, a_1 = e'_t, a_2 = e'_s, b_2 = e_s$ and from the construction we know there exists such an $a' \in G_{\text{src}}.E$ so that $\text{imm}(G'_{\text{src}}.\text{po})(a', b_1')$ holds. In this case $\mathcal{M}'(e_s, e_t), \mathcal{M}'(b', e_t)$, and $G'_{\text{tgt}}.\text{ew}(e_s, b')$ hold.

As a result, this condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$.

(e)

$$\forall c_t \in G'_{\text{tgt}}.E \setminus (A' \cup B'), b_t \in B'. G'_{\text{tgt}}.\text{po}(b_t, c_t) \implies$$
$$\exists b_t, b_x, c_s \in G'_{\text{src}}.E, \mathcal{M}'(c_s, c_t) \land \mathcal{M}'(b_t, b_x) \land \mathcal{M}'(b'_x, b_t)$$
$$\land \mathcal{G}'_{\text{src}}.\text{ew}(b_s, b') \land (G'_{\text{src}}.\text{po}(b_s, c_s) \lor G'_{\text{src}}.\text{po}(b'_x, c_s))$$

We know this condition holds in $G_{\text{src}}$ and $G_{\text{tgt}}$. Considering the definitions of $G'_{\text{src}}, G'_{\text{tgt}}, \mathcal{M}'$ we know $b', e_t \notin G'_{\text{tgt}}.E \setminus (A' \cup B')$. Hence the condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$.

(f)

$$\forall a_t \in A', b_t \in B'. G'_{\text{tgt}}.\text{po}(b_t, a_t) \implies$$
$$\exists a_s, b_s \in G'_{\text{src}}.E, \mathcal{M}'(a_s, a_t) \land \mathcal{M}'(b_s, b_t) \land \neg G'_{\text{src}}.\text{po}(b_s, a_s)$$

We know the condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$.

Considering the definitions of $G'_{\text{src}}, G'_{\text{tgt}}, \mathcal{M}'$ for $b_t = e_t, a_t = e'_t, a_s = e'_s, b_s = b'$ the condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$.

(g)

$$\forall c_t, c'_t \in G'_{\text{tgt}}.E \setminus (A' \cup B'), G'_{\text{tgt}}.\text{po}(c_t, c'_t) \implies$$
$$\exists c_s, c'_s \in G'_{\text{src}}.E, \mathcal{M}'(c_s, c_t) \land \mathcal{M}'(c'_s, c'_t) \land G'_{\text{src}}.\text{po}(c_s, c'_s)$$

We know the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. In this case $e'_t \notin G'_{\text{tgt}}.E \setminus (A' \cup B')$. Hence the condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$.

(h)

$$\forall c_t \in G'_{\text{tgt}}.E \setminus (A' \cup B'), a_t \in A', G'_{\text{tgt}}.\text{jf}(c_t, a_t) \implies$$
$$\exists c_s, a_s \in G'_{\text{src}}.E, \mathcal{M}'(a_s, a_t) \land \mathcal{M}'(c_s, c_t) \land G'_{\text{src}}.\text{jf}(c_s, a_s)$$

We know the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. Considering the definitions of $G'_{\text{src}}, G'_{\text{tgt}}, \mathcal{M}'$, the condition holds for $a_t = e'_t, a_s = e'_s$ between $G'_{\text{src}}$ and $G'_{\text{tgt}}$. 

We know the condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$. Considering the definitions of $G'_{\text{src}}$, $G'_{\text{tgt}}$, $M'$, for $a_t = e'_t$ there is no outgoing edge from $e'_t$. Hence the condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$.

We know the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. In this case the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$ as $e'_t \notin B'$.

We know the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. In this case the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$ as $e'_t \notin B'$ and $e'_t \notin G'_{\text{tgt}}.E \setminus (A' \cup B')$.

We know the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. Hence the condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$. Considering the definitions of $G'_{\text{src}}$, $G'_{\text{tgt}}$, $M'$, for $a_t = e'_t$ and $a_s = e'_s$ the condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$.

We know the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. Considering the definitions of $G'_{\text{src}}$, $G'_{\text{tgt}}$, $M'$, for $a_t = e'_t$ and $a_s = e'_s$ the condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$.

We know the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. Following the definitions of $G'_{\text{src}}$ and $G'_{\text{tgt}}$, $M'$, the condition holds between $G'_{\text{src}}$ and $G'_{\text{tgt}}$ as $e'_t \notin B'$ and $e'_t \notin G'_{\text{tgt}}.E \setminus (A' \cup B')$. 
We know the condition holds between $G_{src}$ and $G_{tgt}$. Following the definitions of $G'_{src}$ and $G'_{tgt}$, the condition holds between $G'_{src}$ and $G'_{tgt}$ as $e'_t \not\in B'$ and $e'_t \not\in G_{tgt} \cdot E \setminus (A' \cup B')$.

We know the condition holds between $G_{src}$ and $G_{tgt}$. In this case $e'_t \not\in G'_{tgt} \cdot E \setminus (A' \cup B')$. Hence the condition holds between $G'_{src}$ and $G'_{tgt}$.

We know the condition holds between $G_{src}$ and $G_{tgt}$. Following the definitions of $G'_{src}$ and $G'_{tgt}$, $M'$, $(e'_t, e_t), (b'_t, e_s) \in M'$. Hence the condition holds between $G'_{src}$ and $G'_{tgt}$.

We know the condition holds between $G_{src}$ and $G_{tgt}$. Following the definitions of $G'_{src}$ and $G'_{tgt}$, $M'$ the condition holds between $G'_{src}$ and $G'_{tgt}$ as $G'_{tgt} \cdot ew = G_{tgt} \cdot ew$.

Hence the invariant holds between $G'_{src}$ and $G'_{tgt}$.

(3) Condition to show:

there exists $pc'$ such that

$p'_c = S'$

$p'_c \cdot po = G'_{src} \cdot po \cap (S' \times S')$

$p'_c \cdot ref = G'_{src} \cdot ref \cap (S' \times S')$

$p'_c \cdot mo = G'_{src} \cdot mo \cap (S' \times S')$

where $S'(G_{src}, pc') \triangleq \{ e \mid e \in G'_{src} \cdot E \land G'_{src} \cdot po^1(e, pc'(e.tid)) \}$.

If $e'_t \not\in X'_t$ then $X'_t = X_t$. In this case $pc' = pc$, $S' = S$, and $X'_t = X_t$. 

Otherwise, when \( e'_t \in X'_t \) then \( X'_t \) is an extension of \( X_t \), that is,

\[
X'_t \cdot E = X_t \cdot E \cup \{e_t, e'_t\}
\]

\[
X'_t \cdot po = \{X'_t \cdot po \cup \{(a, e_t) \mid a \in X_t \cdot E \land G'_{tgt}.po(a, e_t)\}\} \cup \{(e_t, e'_t)\}^+
\]

\[
X'_t \cdot rf = X_t \cdot rf \cup \{(a, e_t) \mid a \in X_t \cdot E \land G'_{tgt}.rf(a, e_t)\}\cup \{(e_t, a) \mid a \in X_t \cdot E \land G'_{tgt}.rf(e_t, a)\}\cup \{(e'_t, a) \mid a \in X_t \cdot E \land G'_{tgt}.rf(e'_t, a)\}
\]

\[
X'_t \cdot mo = X_t \cdot mo \cup \{(a, e_t) \mid a \in X_t \cdot E \land G'_{tgt}.mo(a, e_t)\}\cup \{(e_t, a) \mid a \in X_t \cdot E \land G'_{tgt}.mo(e_t, a)\}\cup \{(e'_t, a) \mid a \in X_t \cdot E \land G'_{tgt}.mo(e'_t, a)\}
\]

We also know that the \( X_t \) and \( X_s \) are related as follows.

\[
X_t \cdot E = X_t \cdot E
\]

\[
X_s \cdot po = \{(a, b_s) \mid M(a, a_t) \land M(b_s, b_t) \land X_t \cdot po(a_t, b_t)\}
\]

\[
X_s \cdot rf = \{(a, b_s) \mid M(a, a_t) \land M(b_s, b_t) \land X_t \cdot rf(a_t, b_t)\}
\]

\[
X_s \cdot mo = \{(a, b_s) \mid M(a, a_t) \land M(b_s, b_t) \land X_t \cdot mo(a_t, b_t)\}
\]

**Source Execution Extraction.**

From \( X'_t \) we derive \( X'_s \) and relate \( X'_s \) to \( X_s \).

\[
X'_s \cdot E = X'_t \cdot E = X_t \cdot E \cup \{e_t, e'_t\} = X_t \cdot E \cup \{e_t, e'_t\}
\]

\[
X'_s \cdot po = \{(a, b_s) \mid X'_t \cdot po(a_t, b_t) \land X'_s \cdot po(a_t, b_t)\}
\]

\[
\Rightarrow X'_s \cdot po = \{(a, b_s) \mid X_t \cdot po(a_t, b_t) \land M'(a_t, a_t) \land M'(b_s, b_t)\}
\]

\[
\cup \{(a, e'_t) \mid X'_t \cdot po(a_t, e'_t) \land M'(a_t, a_t) \land M'(b'_s, b_t)\}
\]

\[
\cup \{(a, b'_t) \mid X'_t \cdot po(a_t, e_t) \land M'(a_t, a_t) \land M'(e'_t, e_t)\}
\]

\[
\Rightarrow X'_s \cdot po = X_s \cdot po
\]

\[
\cup \{(a, e'_t) \mid X'_t \cdot po(a_t, e'_t) \land M'(a_t, a_t) \land M'(e'_t, e'_t)\}
\]

\[
\cup \{(a, b'_t) \mid X'_t \cdot po(a_t, e'_t) \land M'(a_t, a_t) \land M'(e'_t, e'_t)\}
\]

\[
\Rightarrow X'_s \cdot rf = X_s \cdot rf
\]

\[
\cup \{(a, e'_t) \mid X'_t \cdot rf(a_t, e'_t) \land M'(a_t, a_t) \land M'(e'_t, e'_t)\}
\]

\[
\cup \{(a, b'_t) \mid X'_t \cdot rf(a_t, e'_t) \land M'(a_t, a_t) \land M'(b'_t, e'_t)\}
\]

\[
\cup \{(e'_t, a) \mid X'_t \cdot rf(e'_t, a_t) \land M'(a_t, a_t) \land M'(e'_t, e'_t)\}
\]

\[
\cup \{(b'_t, a) \mid X'_t \cdot rf(e'_t, a_t) \land M'(a_t, a_t) \land M'(b'_t, e'_t)\}
\]

\[
\Rightarrow X'_s \cdot mo = X_s \cdot mo
\]

\[
\cup \{(a, e'_t) \mid X'_t \cdot mo(a_t, e'_t) \land M'(a_t, a_t) \land M'(e'_t, e'_t)\}
\]

\[
\cup \{(a, b'_t) \mid X'_t \cdot mo(a_t, e'_t) \land M'(a_t, a_t) \land M'(b'_t, e'_t)\}
\]

\[
\cup \{(b'_t, a) \mid X'_t \cdot mo(e'_t, a_t) \land M'(a_t, a_t) \land M'(b'_t, e'_t)\}
\]

\[
\Rightarrow X'_s \cdot mo = X_s \cdot mo
\]
In this case we already have $X\subseteq S\cup\{e',b\}$.

Now we relate $X'_s$ and $S'$.

$X'_s.E = X.s.E \cup \{e',b\} = S \cup \{e',b\} = S'$

We already have

$X'_s.po = X.s.po$

$\cup \{(a_s,e'_t) \mid X'_s.po(a_s,e'_t) \land M'(a_s,a_t) \land M'(e'_t,e'_t)\}$

$\cup \{(a_s,b') \mid X'_s.po(a_s,e_t) \land M'(a_s,a_t) \land M'(b',e_t)\}$

$\cup \{(e'_s,a_s) \mid X'_s.po(e_s,e'_t) \land M'(e'_s,e'_t) \land M'(a_s,a_t)\}$

$\cup \{(b',a_s) \mid X'_s.po(b',a_s) \land M'(b',e_t) \land M'(a_s,a_t)\}$

$\Rightarrow X'_s.po = G_{src}.po \cap (S \times S) \cup \{G'_{src}.po(a_s,e'_t) \mid a_s,e_s \in S'\}$

$\cup \{(a_s,e'_t), (b',a_s) \mid a_s,b' \in S'\} \cup \{(e'_s,a_s) \mid a_s \in S'\}$

$\Rightarrow X'_s.po = G_{src}.po \cap (S' \times S')$

We already have

$X'_s.rf = X.s.rf$

$\cup \{(a_s,e'_t) \mid X'_s.rf(a_s,e'_t) \land M'(a_s,a_t) \land M'(e'_t,e'_t)\}$

$\cup \{(a_s,b') \mid X'_s.rf(a_s,e_t) \land M'(a_s,a_t) \land M'(b',e_t)\}$

$\cup \{(e'_s,a_s) \mid X'_s.rf(e'_s,a_s) \land M'(e'_t,e'_t) \land M'(a_s,a_t)\}$

$\cup \{(b',a_s) \mid X'_s.rf(b',a_s) \land M'(b',e_t) \land M'(a_s,a_t)\}$

$\Rightarrow X'_s.rf = G_{src}.rf \cap (S \times S) \cup \{G'_{src}.rf(a_s,e'_t) \mid a_s,e_s \in S'\}$

$\cup \{G'_{src}.rf(a_s,b') \mid a_s,b' \in S'\}$

$\cup \{G'_{src}.rf(e'_s,a_s) \mid a_s \in S'\}$

$\Rightarrow X'_s.rf = G_{src}.rf \cap (S' \times S')$

We already have

$X'_s.mo = X.s.mo$

$\cup \{(a_s,e'_t) \mid X'_s.mo(a_s,e'_t) \land M'(a_s,a_t) \land M'(e'_t,e'_t)\}$

$\cup \{(a_s,b') \mid X'_s.mo(a_s,e_t) \land M'(a_s,a_t) \land M'(b',e_t)\}$

$\cup \{(e'_s,a_s) \mid X'_s.mo(e_s,e'_t) \land M'(e'_s,e'_t) \land M'(a_s,a_t)\}$

$\cup \{(b',a_s) \mid X'_s.mo(b',a_s) \land M'(b',e_t) \land M'(a_s,a_t)\}$

$\Rightarrow X'_s.mo = G_{src}.mo \cap (S \times S)$

$\cup \{G'_{src}.mo(a_s,e'_t) \mid a_s,e_s \in S'\}$

$\cup \{G'_{src}.mo(a_s,b') \mid a_s,b' \in S'\}$

$\Rightarrow X'_s.mo = G'_{src}.mo \cap (S' \times S')$

As a result, $G'_{src}$ is close to $G_{tgt}$.

Case $e'_t \in G'_{tgt}E \setminus (A',B')$ where $A' = A$ and $B' = B$:

In this case $G'_{tgt}E = G_{tgt}E \cup \{e'_t\}$.

In $G_{src} e_s$ is the corresponding event of $e_t$, that is, $M(e_s,e_t)$.
We also append corresponding event in $G_{src}$ and construct $G'_{src}$.

(1) Condition to show: $G'_{src}$ is consistent.

Two possibilities: (1) either $e_s$ is po-maximal or (2) there exists an event $e''_s$ such that $\text{imm}(G_{src}.po)(e_s, e''_s)$ and $e''_s$ is $G_{src}.po$ maximal.

Let the maximal event be $e_m$.

We append an event $e'_s$ in $G_{src}$ by po-extending from $e_m$ and create $G'_{src}$ such that

\[
G'_{src}.E = G_{src}.E \cup \{e'_s\}
\]
\[
G'_{src}.po = (G_{src}.po \cup \{(e_m, e'_s)\})^+
\]
\[
G'_{src}.jf = G_{src}.jf \cup \{(w_s, e'_s) \mid (w_s, e'_s) \in (G_{src}. \mathcal{W} \times G_{src}. \mathcal{R}) \land \overline{M}(w_s, w_t) \land G'_{tgt}.jf(w_t, e'_t) \land -G'_{src}.cf(w_s, e'_s)\}
\]
\[
G'_{src}.mo = G_{src}.mo \cup \{(w_s, e'_s) \mid (w_s, e'_s) \in (G_{src}. \mathcal{W} \times G'_{src}. \mathcal{W}) \land \overline{M}(w_s, w_t) \land G'_{tgt}.mo(w_t, e_t) \land -G'_{src}.cf(w_s, e'_s)\}
\]
\[
G'_{src}.ew = G_{src}.ew \cup \{(w_s, e'_s, e'_t, w_s) \mid (w_s, e'_s) \in (G_{src}. \mathcal{W} \times G'_{src}. \mathcal{W}) \land \overline{M}(w_s, w_t) \land G'_{tgt}.ew(w_t, e_t)\}
\]

Also in this case $M' = M \cup \{(e'_s, e'_t)\}$.

Now we check whether $G'_{src}$ is consistent.

We know $G_{src}, G'_{tgt}$ are consistent hence satisfy (ICFJ). Hence from definition of $G'_{src}$ and $M'$ we know that $G'_{src}$ satisfies (ICFJ).

We know $G_{src}, G'_{tgt}$ are consistent hence satisfy (ICF). Hence following the definition of $G'_{src}$, and $M'$ we know $G'_{src}$ preserves (ICF).

We know that $G_{src}$ preserves (CF), (CFJ), (VISJ). Also $G'_{tgt}.jf(w_t, e'_t)$ implies $w_t \in \text{vis}(G'_{tgt})$ and $\neg G'_{src}.cf(w_t, e'_s)$ and $\overline{M}(w_s, w_t)$ holds. Following the construction, $w_s \in \text{vis}(G'_{src})$ as well as $\neg G'_{src}.cf(w_s, e'_s)$ hold. Hence $G'_{src}$ preserves (CF), (CFJ), (VISJ).

We know $G_{src}$ preserves (COH'). Consider there is $(G'_{src}.hb; G'_{src}.eco)$ cycle in $G'_{src}$ and $e'_s$ is a part of this cycle. In that case there is a $(G'_{tgt}.hb; G'_{tgt}.eco)$ cycle in $G'_{tgt}$ and $e'_t$ is a part of the cycle. However, $G'_{tgt}$ preserves (COH') and hence there is no $(G'_{tgt}.hb; G'_{tgt}.eco)$ cycle. Hence a contradiction and $G'_{src}$ preserves (COH').

As a result, $G'_{src}$ is consistent.

Thus finally $M' = M \cup \{(e'_s, e'_t)\}$ and $pc' = pc[e_s, tid \mapsto e'_s]$.

(2) Condition to show: the simulation invariant holds between $G'_{src}$ and $G'_{tgt}$

(a)

\[
\forall c_t \in G'_{tgt}.E \setminus (A' \cup B') \exists c_s \in G'_{src}.E. M'(c_s, c_t)
\]

We know this condition holds in $G_{src}$ and $G_{tgt}$. Considering the definitions of $G'_{src}$, $G_{tgt}$, and $M'$, the condition holds between $G'_{src}$ and $G'_{tgt}$ as $M'(e'_s, e'_t)$ holds.

(b)

\[
\forall c_t \in G'_{tgt}.E \setminus (A' \cup B'), a_t \in A', b_t \in B' \land G'_{tgt}.po(c_t, b_t) \implies \\
\exists c_s, a_s, b_s \in G'_{src}.E. M'(c_s, c_t) \land M'(a_s, a'_t) \land M'(b_s, b_t) \land (\exists a'' \in G'_{src}.E. a_s.loc = a'' locus \land a_s.loc = a'' locus) \land G'_{src}.po(c_s, a'') \land \text{imm}(G_{src}.po)(a'', b_s))
\]

We know this condition holds in $G_{src}$ and $G_{tgt}$. Considering the definitions of $G'_{src}$, $G'_{tgt}$, and $M'$, when $c_t = e'_t$ then $c_t$ is $G'_{tgt}.po$-maximal and there is no $G'_{tgt}.po(c_t, b_t)$. Hence the condition holds between $G'_{src}$ and $G'_{tgt}$.

\(\forall c_t \in G'_{tgt}.E \setminus (A' \cup B'), a_t \in A', \land G'_{tgt}.po(c_t, a_t) \implies \exists c_s, a_s \in G'_{src}.E. M'(c_s, c_t) \land M'(a_s, a_t) \land G'_{src}.po(c_s, a_s)\)

We know this condition holds in $G_{src}$ and $G_{tgt}$. Considering the definitions of $G'_{src}$, $G'_{tgt}$, and $M'$, when $c_t = e'_t$ then $c_t$ is $G'_{tgt}.po$-maximal and there is no $G'_{tgt}.po(c_t, a_t)$. Hence the condition holds between $G'_{src}$ and $G'_{tgt}$.

\(\forall a_t \in A', b_t \in B'. \land \text{imm}(G'_{tgt}.po)(b_t, a_t) \implies \exists b_s, b''_s, c'_s \in G'_{src}.E. M'(c_s, c_t) \land M'(a_s, a_t) \land M'(b_s, b_t) \land \text{imm}(G'_{src}.po)(c_s, b) \land \text{imm}(G'_{src}.po)(a_s, b') \land G'_{src}.cf(a_s, b') \land \text{imm}(G'_{src}.po)(a'_s, b) \land b_s.loc = b'.loc \land b_s.ord = b'.ord \land G'_{src}.ew(b_s, b'))\)

We know this condition holds in $G_{src}$ and $G_{tgt}$. Considering the definitions of $G'_{src}$, $G'_{tgt}$, $M'$, $e'_t \notin (A' \cup B')$. As a result, this condition holds between $G'_{src}$ and $G'_{tgt}$.

\(\forall c_t \in G'_{tgt}.E \setminus (A' \cup B'), b_t \in B'. G'_{tgt}.po(b_t, c_t) \implies \exists b_s, b''_s, c'_s \in G'_{src}.E. M'(c_s, c_t) \land M'(b_s, b_t) \land M''(b''_s, b_t) \land G'_{src}.ew(b_s, b'') \land (G'_{src}.po(b_s, c_s) \lor G'_{src}.po(b'', c_s))\)

We know this condition holds in $G_{src}$ and $G_{tgt}$. We consider two cases for $e_t$.

**case $e_t \in G'_{tgt}.E \setminus (A' \cup B')$:**

In this case there exists $b_t$ such that $G_{tgt}.po(b_t, e_t)$.

Hence $G'_{tgt}.po(e_t, e'_t)$ implies $G_{tgt}.po(b_t, e'_t)$ and the condition holds.

**case $e_t \in A'$:**

In this case there exists an event $e''_t$ such that $\text{imm}(G'_{src}.po)(e''_t, b_t)$ and $b_t \in B'$ and $\text{imm}(G'_{tgt}.po)(b_t, e_t)$. Thus the condition holds between $G'_{src}$ and $G'_{tgt}$.

\(\forall a_t \in A', b_t \in B'. G'_{tgt}.po(b_t, a_t) \implies \exists a_s, b_s \in G'_{src}.E. M'(a_s, a_t) \land M'(b_s, b_t) \land \neg G'_{src}.po(b_s, a_s)\)

We know this condition holds in $G_{src}$ and $G_{tgt}$. Considering the definitions of $G'_{src}$, $G'_{tgt}$, $M'$, $e'_t \notin (A' \cup B')$. As a result, this condition holds between $G'_{src}$ and $G'_{tgt}$.

\(\forall c_t, c'_t \in G'_{tgt}.E \setminus (A' \cup B'). G'_{tgt}.po(c_t, c'_t) \implies \exists c_s, c'_s \in G'_{src}.E. M'(c_s, c_t) \land M'(c'_s, c'_t) \land G'_{src}.po(c_s, c'_s)\)

We know this condition holds between $G_{src}$ and $G_{tgt}$. Considering the definitions of $G'_{src}$, $G'_{tgt}$, $M'$, this condition holds between $G'_{src}$ and $G'_{tgt}$ where $c'_t = e'_t$. 
∀c_t ∈ G'_tgt. E \ (A' \cup B'), a_t ∈ A'. G'_tgt. jf(c_t, a_t) \implies
\exists c_s, a_s ∈ G'_src. E. M'(a_s, a_t) \land M'(c_s, c_t) \land G'_src. jf(c_s, a_s)

We know the condition holds between G'_src and G'_tgt. Considering the definitions of G'_src, G'_tgt, M', the condition holds between G'_src and G'_tgt for c_t = e'_t where there is no outgoing G'_tgt. jf edge from e'_t.

∀c_t ∈ G'_tgt. E \ (A' \cup B'), a_t ∈ A'. G'_tgt. jf(a_t, c_t) \implies
\exists c_s, a_s ∈ G'_src. E. M'(a_s, a_t) \land M'(c_s, c_t) \land G'_src. jf(a_s, c_s)

We know the condition holds between G'_src and G'_tgt. Considering the definitions of G'_src, G'_tgt, M', the condition holds between G'_src and G'_tgt for c_t = e'_t.

∀c_t ∈ G'_tgt. E \ (A' \cup B'), b_t ∈ B'. G'_tgt. jf(c_t, b_t) \implies
\exists b_s, c_s ∈ G'_src. E. M'(b_s, b_t) \land M'(c_s, c_t) \land G'_src. jf(c_s, b_s)
∧ (\exists b' ∈ G'_src. E. M'(b', b_t) \land G'_src. eW(b_s, b') \implies G'_src. jf(c_s, b'))

We know the condition holds between G'_src and G'_tgt. Considering the definitions of G'_src, G'_tgt, M', the condition holds between G'_src and G'_tgt for c_t = e'_t.

∀c_t ∈ G'_tgt. E \ (A' \cup B'), b_t ∈ B'. G'_tgt. jf(b_t, c_t) \implies
((\exists b_s, c_s ∈ G'_src. E. (M'(b_s, b_t) \land \exists b' ∈ G'_src. E. M'(b', b_t) \land G'_src. eW(b_s, b'))) \implies
G'_src. jf(b_s, c_s))
∧ (\exists b_s, c_s ∈ G'_src. E. (M'(b', b_t) \land M'(b', b_t) \land M'(c_s, c_t) \land G'_src. eW(b_s, b')) \implies
G'_src. jf(b_s, c_s))

We know the condition holds between G'_src and G'_tgt. Considering the definitions of G'_src, G'_tgt, M', the condition holds between G'_src and G'_tgt for c_t = e'_t.

∀c_t, e'_t ∈ G'_tgt. E \ (A' \cup B'). G'_tgt. jf(c_t, e'_t) \implies
\exists e_s, e'_s ∈ G'_src. E. M'(e_s, e'_t) \land M'(e'_s, e'_t) \land G'_src. jf(e_s, e'_s)

We know the condition holds between G'_src and G'_tgt. Considering the definitions of G'_src, G'_tgt, M', (1) this condition holds between G'_src and G'_tgt where c'_t = e'_t. (2) the condition also holds when c_t = e'_t as in that case there is no outgoing edge from e'_t.

∀c_t ∈ G'_tgt. E \ (A' \cup B'), a_t ∈ A', b_t ∈ B'. G'_tgt. mo(c_t, a_t) \implies
\exists c_s, a_s ∈ G'_src. E. M'(c_s, a_t) \land M'(a_s, a_t) \land G'_src. mo(c_s, a_s)

We know the condition holds between G'_src and G'_tgt. Considering the definitions of G'_src, G'_tgt, M', for c_t = e'_t the condition holds between G'_src and G'_tgt.
We know the condition holds between $G'_{\text{src}}$ and $G_{\text{tgt}}$. Considering the definitions of $G'_{\text{src}}$, $G_{\text{tgt}}$, $\mathcal{M}'$, for $c_t = e'_t$, the condition holds between $G'_{\text{src}}$ and $G_{\text{tgt}}$.

We know the condition holds between $G'_{\text{src}}$ and $G_{\text{tgt}}$. Considering the definitions of $G'_{\text{src}}$, $G_{\text{tgt}}$, $\mathcal{M}'$, for $c_t = e'_t$, the condition holds between $G'_{\text{src}}$ and $G_{\text{tgt}}$.

We know the condition holds between $G'_{\text{src}}$ and $G_{\text{tgt}}$. Considering the definitions of $G'_{\text{src}}$, $G_{\text{tgt}}$, $\mathcal{M}'$, for $c_t = e'_t$ or $c'_t = e'_t$, the condition holds between $G'_{\text{src}}$ and $G_{\text{tgt}}$.

We know the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. Following the definitions of $G'_{\text{src}}$ and $G_{\text{tgt}}$, $\mathcal{M}'(e'_s, e_t)$ holds. Hence the condition holds between $G'_{\text{src}}$ and $G_{\text{tgt}}$.

We know the condition holds between $G_{\text{src}}$ and $G_{\text{tgt}}$. Following the definitions of $G'_{\text{src}}$ and $G_{\text{tgt}}$, $\mathcal{M}'(e'_s, e_t)$ holds. Hence the condition holds between $G'_{\text{src}}$ and $G_{\text{tgt}}$ for $c_t = e'_t$ or $c'_t = e'_t$.

Hence the invariant holds between $G'_{\text{src}}$ and $G_{\text{tgt}}$.

**Condition to show:**

*there exists pc’ such that*

$X'_s, E = S'$

$X'_s, po = G_{\text{src}}' \cdot po \cap (S' \times S')$

$X'_s, rf = G_{\text{src}}' \cdot rf \cap (S' \times S')$

$X'_s, mo = G_{\text{src}}' \cdot mo \cap (S' \times S')$

*where $S'(G_{\text{src}}' \cdot pc') \triangleq \{ e \mid e \in G_{\text{src}}' \cdot E \land G_{\text{src}}' \cdot po^\perp(e, pc'(e, tid)) \}$.*

If $e'_t \not\in X'_t$ then $X'_t = X_t$. In this case pc’ = pc, $S' = S$, and $X'_s = X_s$. 

Otherwise, when \( e'_t \in X'_t \) then \( X'_t \) is an extension of \( X_t \), that is,
\[
X'_t, E = X_t, E \cup \{e'_t\}
\]
\[
X'_t, \text{po} = (X_t, \text{po} \cup \{(a, e'_t) \mid a \in X_t, E \land G'_{\text{tgt}} \cdot \text{po}(a, e'_t)\} \cup \{(e_t, e'_t)\})^+
\]
\[
X'_t, \text{rf} = X_t, \text{rf} \cup \{(a, e'_t) \mid a \in X_t, E \land G'_{\text{tgt}} \cdot \text{rf}(a, e'_t)\}
\]
\[
\cup \{(e_t, e'_t) \mid a \in X_t, E \land G'_{\text{tgt}} \cdot \text{rf}(e_t, a)\}
\]
\[
X'_t, \text{mo} = X_t, \text{mo} \cup \{(a, e'_t) \mid a \in X_t, E \land G'_{\text{tgt}} \cdot \text{mo}(a, e'_t)\}
\]
\[
\cup \{(e'_t, a) \mid a \in X_t, E \land G'_{\text{tgt}} \cdot \text{mo}(e'_t, a)\}
\]

We also know that the \( X_t \) and \( X_s \) are related as follows.
\[
X_s, E = X_t, E \]
\[
X_s, \text{po} = \{(a_s, b_s) \mid \mathcal{M}(a_s, a_t) \land \mathcal{M}(b_s, b_t) \land X_t, \text{po}(a_t, b_t) \land \neg (a_t \in A \land b_t \in B)\}
\]
\[
\cup \{(a_s, b_s) \mid \mathcal{M}(a_s, a_t) \land \mathcal{M}(b_s, b_t) \land X_t, \text{po}(b_t, a_t) \land (a_t \in A \land b_t \in B)\}
\]
\[
X_s, \text{rf} = \{(a_s, b_s) \mid \mathcal{M}(a_s, a_t) \land \mathcal{M}(b_s, b_t) \land X_t, \text{rf}(a_t, b_t)\}
\]
\[
X_s, \text{mo} = \{(a_s, b_s) \mid \mathcal{M}(a_s, a_t) \land \mathcal{M}(b_s, b_t) \land X_t, \text{mo}(a_t, b_t)\}
\]

**Source Execution Extraction.**

From \( X'_t \) we derive \( X'_s \) and relate \( X'_s \) to \( X_s \).
\[
X'_s, E = X_t, E \cup \{e_t, e'_t\} = X_s, E \cup \{e_t, e'_t\}
\]
\[
X'_s, \text{po} = \{(a_s, b_s) \mid \mathcal{M}(a_s, a_t) \land \mathcal{M}(b_s, b_t) \land X'_t, \text{po}(a_t, b_t) \land \neg (a_t \in A' \land b_t \in B')\}
\]
\[
\cup \{(a_s, b_s) \mid \mathcal{M}(a_s, a_t) \land \mathcal{M}(b_s, b_t) \land X'_t, \text{po}(a_t, b_t) \land (a_t \in A' \land b_t \in B')\}
\]
\[
\Rightarrow X'_s, \text{po} = X_s, \text{po} \cup \{(a_s, e'_t) \mid X'_t, \text{po}(a_t, e'_t) \land \mathcal{M}'(a_s, a_t) \land \mathcal{M}'(e'_t, e'_t)\}
\]
\[
X'_s, \text{rf} = \{(a_s, b_s) \mid X'_t, \text{rf}(a_t, b_t) \land \mathcal{M}'(a_s, a_t) \land \mathcal{M}'(b_s, b_t)\}
\]
\[
\Rightarrow X'_s, \text{rf} = \{(a_s, b_s) \mid X_t, \text{rf}(a_t, b_t) \land \mathcal{M}'(a_s, a_t) \land \mathcal{M}'(b_s, b_t)\}
\]
\[
\cup \{(a_s, e'_t) \mid X'_t, \text{rf}(a_t, e'_t) \land \mathcal{M}'(a_s, a_t) \land \mathcal{M}'(e'_t, e'_t)\}
\]
\[
\cup \{(e'_t, a_t) \mid X'_t, \text{rf}(e'_t, a_t) \land \mathcal{M}'(e'_t, e'_t) \land \mathcal{M}'(a_s, a_t)\}
\]
\[
\Rightarrow X'_s, \text{rf} = X_s, \text{rf}
\]
\[
\cup \{(a_s, e'_t) \mid X_t, \text{rf}(a_t, e'_t) \land \mathcal{M}'(a_s, a_t) \land \mathcal{M}'(e'_t, e'_t)\}
\]
\[
\cup \{(e'_t, a_t) \mid X_t, \text{rf}(e'_t, a_t) \land \mathcal{M}'(e'_t, e'_t) \land \mathcal{M}'(a_s, a_t)\}
\]
\[
X'_s, \text{mo} = \{(a_s, b_s) \mid X'_t, \text{mo}(a_t, b_t) \land \mathcal{M}'(a_s, a_t) \land \mathcal{M}'(b_s, b_t)\}
\]
\[
\Rightarrow X'_s, \text{mo} = \{(a_s, b_s) \mid X_t, \text{mo}(a_t, b_t) \land \mathcal{M}'(a_s, b_s) \land \mathcal{M}'(b_s, b_t)\}
\]
\[
\cup \{(a_s, e'_t) \mid X'_t, \text{mo}(a_t, e'_t) \land \mathcal{M}'(a_s, a_t) \land \mathcal{M}'(e'_t, e'_t)\}
\]
\[
\cup \{(e'_t, a_t) \mid X'_t, \text{mo}(e'_t, a_t) \land \mathcal{M}'(e'_t, e'_t) \land \mathcal{M}'(a_s, a_t)\}
\]
\[
\Rightarrow X'_s, \text{mo} = X_s, \text{mo}
\]
\[
\cup \{(a_s, e'_t) \mid X_t, \text{mo}(a_t, e'_t) \land \mathcal{M}'(a_s, a_t) \land \mathcal{M}'(e'_t, e'_t)\}
\]
\[
\cup \{(e'_t, a_t) \mid X_t, \text{mo}(e'_t, a_t) \land \mathcal{M}'(e'_t, e'_t) \land \mathcal{M}'(a_s, a_t)\}
\]

In this case \( pc' = pc[e'_s.\text{tid} \mapsto e'_t] \) and hence \( S' = S \cup \{e'_s\} \).

Now we relate \( X'_s \) and \( S' \).
\[
X'_s, E = X_s, E \cup \{e'_s\} = S \cup \{e'_s\} = S'
\]

We already have
\[
X'_s, \text{po} = (X_s, \text{po} \cup \{(a_s, e'_t) \mid X'_t, \text{po}(a_t, e'_t) \land \mathcal{M}'(a_s, a_t) \land \mathcal{M}'(e'_t, e'_t)\})^+
\]
\[
\Rightarrow X'_s, \text{po} = G_{\text{src}} \cdot \text{po} \cap (S \times S) \cup \{G'_{\text{src}} \cdot \text{po}(a_s, e'_s) \mid a_s, e_s \in S\}
\]
\[
\Rightarrow X'_s, \text{po} = G'_{\text{src}} \cdot \text{po} \cap (S' \times S')
\]

We already have
$X'_s.rf = X_s.rf$
$\cup \{(a_s, e'_s) | X'_t.rf(at, e'_t) \land M'(a_s, at) \land M'(e'_s, e'_t)\}$
$\cup \{(e'_s, a_s) | X'_t.rf(e'_t, at) \land M'(e'_s, e'_t) \land M'(a_s, at)\}$
$\Rightarrow X'_s.rf = G_{src}.rf \cap (S \times S) \cup \{G'_{src}.rf(a_s, e'_s) | a_s, e_s \in S'\}$
$\cup \{G'_{src}.rf(e'_s, a_s) \cup a_s, e_s \in S'\}$
$\Rightarrow X'_s.rf = G_{src}.rf \cap (S' \times S')$

We already have
$X'_s.mo = X_s.mo$
$\cup \{(a_s, e'_s) | X'_t.mo(at, e'_t) \land M'(a_s, at) \land M'(e'_s, e'_t)\}$
$\cup \{(e'_s, a_s) | X'_t.mo(e'_t, at) \land M'(e'_s, e'_t) \land M'(a_s, at)\}$
$\Rightarrow X'_s.mo = G_{src}.mo \cap (S \times S) \cup \{G'_{src}.mo(a_s, e'_s) | a_s, e_s \in S'\}$
$\cup \{G'_{src}.mo(e'_s, a_s) \cup a_s, e_s \in S'\}$
$\Rightarrow X'_s.mo = G'_{src}.mo \cap (S' \times S')$

As a result, $G'_{src} \sim G'_tg$. Thus we complete the construction of the source event structure $G_{src}$ and the source execution $X_s$ can be extracted from $G_{src}$, that is, $X_s \in ex_{weakestmo}(G_{src})$. 

□
G PROOFS OF CORRECTNESS OF ELIMINATIONS

We restate the definition of compilation correctness and the safe elimination theorem.

**Definition 7.** A transformation of program $P_{\text{src}}$ in memory model $M_{\text{src}}$ to program $P_{\text{tgt}}$ in model $M_{\text{tgt}}$ is correct if it does not introduce new behaviors: i.e., $\text{Behavior}_{M_{\text{tgt}}}(P_{\text{tgt}}) \subseteq \text{Behavior}_{M_{\text{src}}}(P_{\text{src}})$.

**Theorem 7.** The eliminations in Fig. 12 are correct in both weakestmo models.

The safe eliminations from Fig. 12 are

**Definition 10.** $\text{elim}(P_{\text{src}}, P_{\text{tgt}})$ such that $P_{\text{tgt}}(i) \subseteq P_{\text{src}}(i) \cup \{ \tau \cdot \tau' \mid \tau \cdot \alpha \cdot \tau' \in P_{\text{src}}(i) \} \land \forall j \neq i. P_{\text{tgt}}(j) = P_{\text{src}}(j)$ where $\alpha$ is a label of shared memory accesses or fences.

Then The formal statement is as follows:

$$\forall P_{\text{src}}. \text{elim}(P_{\text{src}}, P_{\text{tgt}}) \implies$$

$$\forall G_{\text{tgt}}, G_{\text{init}} \rightarrow G_{\text{tgt}}. \text{WEAKESTMO} \cdot G_{\text{src}}. \exists G_{\text{src}}. G_{\text{init}} \rightarrow G_{\text{src}}. \text{WEAKESTMO} \cdot G_{\text{src}} \land$$

$$\forall X_f \in \text{exWEAKESTMO}(G_{\text{tgt}}). \exists X_s \in \text{exWEAKESTMO}(G_{\text{src}}). \text{Behavior}(X_f) = \text{Behavior}(X_s)$$

$$\land X_f. \text{Race} \cap E_{\text{NA}} \neq \emptyset \implies X_s. \text{Race} \cap E_{\text{NA}} \neq \emptyset$$

To prove the theorem, we construct a source event structure following a given target event structure. Then, for an extracted consistent target execution we extract a source execution from the source event structure. Then we show that the source execution is consistent and source and target execution has same behavior. Finally, we show race preservation: if target is racy, then the source execution is also racy. As a result, if the target execution has undefined behavior due to a data race, so does the source execution.

Now we study various safe eliminations.

G.1 Overwritten Write (OW)

**Proof.** Recall the relationship between the two programs for the thread $i$ affected by the transformation:

$$P_{\text{tgt}}(i) \subseteq P_{\text{src}}(i) \cup \{ \tau \cdot \text{St}_o(x, v) \cdot \tau' \mid \tau \cdot \text{St}_o'(x, v') \cdot \text{St}_o(x, v) \cdot \tau' \in P_{\text{src}}(i) \land o' \subseteq o \}$$

For all other threads $j \neq i$, we have $P_{\text{tgt}}(j) = P_{\text{src}}(j)$. Assume we have a target event structure, $G_{\text{tgt}}$, and an execution, $X_f \in \text{exWEAKESTMO}(G_{\text{tgt}})$, extracted from it.

Let $W$ be the set of stores of thread $i$ of $G_{\text{tgt}}$ with label $\text{St}_o(x, v)$, and whose $\text{po}$-prefix has some sequence of labels $\tau$ such that $\tau \cdot \text{St}_o(x, v) \notin P_{\text{src}}(i)$. Then, because of the relationship between the two programs, we know that for each such $w \in W$, $\tau \cdot \text{St}_o'(x, v') \cdot \text{St}_o(x, v) \in P_{\text{src}}(i)$ for the appropriate $\tau$. Let $C$ be the immediate $G_{\text{tgt}}.\text{po}$-predecessors of the events in $W$.

**Source Event Structure Construction.** To construct $G_{\text{src}}$, we follow the construction steps of $G_{\text{tgt}}$. For each target construction step that adds event $e$ to $G_{\text{tgt}}$ to get $G_{\text{tgt}}'$, we perform one or more corresponding steps going from $G_{\text{src}}$ to $G_{\text{src}}'$. We do a case analysis on the event $e$ of the target event structure.

Case $e \notin W$: In this case, we append event $e$ to the source event structure as follows:

$$G_{\text{src}}'.E = G_{\text{src}}.E \uplus \{ e \}$$

$$G_{\text{src}}'.\text{po} = (G_{\text{src}}.\text{po} \uplus \{ (a, e) \mid a \in \text{dom}(G_{\text{src}}.\text{po}; [e]) \})^+$$

$$G_{\text{src}}'.\text{jf} = G_{\text{tgt}}'.\text{jf}$$

$$G_{\text{src}}'.\text{mo} = G_{\text{tgt}}'.\text{mo} \cup \text{imm}(G_{\text{src}}.\text{po}; [W])$$

$$G_{\text{src}}'.\text{ew} = G_{\text{tgt}}'.\text{ew}$$
Now we check the consistency of $G'_{\text{src}}$. We already know that $G_{\text{src}}$ and $G'_{\text{tgt}}$ are consistent. Following the construction of $G'_{\text{src}}$, the (CF), (CFJ), (VISJ), (ICF), (ICFJ) constraints immediately hold. It remains to show that $G'_{\text{src}}$ satisfies (COH').

From the definition, there is no $G_{\text{src}}.\text{hb};G_{\text{src}}.\text{eco}$ as well as $G'_{\text{tgt}}.\text{hb};G'_{\text{tgt}}.\text{eco}$ cycle. Compared to $G_{\text{src}}$ and $G'_{\text{tgt}}$, the additional $G'_{\text{src}}.\text{mo}$ edges are from and to the event $d$. Assume the $\text{mo}$ edges to or from $d$ creates a $G'_{\text{src}}.\text{hb};G'_{\text{src}}.\text{eco}$ cycle. However, for each $G'_{\text{src}}.\text{mo}(d,e)$ or $G'_{\text{src}}.\text{mo}(e,d)$ already there exists $G'_{\text{src}}.\text{mo}(w,e)$ or $G'_{\text{src}}.\text{mo}(e,w)$ respectively where $w \in W$ and $\text{imm}(G_{\text{src}}.\text{po}(d,w))$. Thus event $e$ results no new $G'_{\text{src}}.\text{hb};G'_{\text{src}}.\text{eco}$ cycle and hence $G'_{\text{src}}$ satisfies (COH').

**Case** $e \in W$: In this case, we first append a new event $d$ with $d.\text{lab} = \text{St}_{\sigma'}(x,\nu')$ and then the event $e$ to $G_{\text{src}}$ as follows:

$$G'_{\text{src}}.E = G_{\text{src}}.E \cup \{d, e\} \quad \text{where } d.\text{lab} = \text{St}_{\sigma'}(x,\nu')$$

$$G'_{\text{src}}.\text{po} = (G_{\text{src}}.\text{po} \cup \{(d, e)\} \cup \{(c, d) \mid (c, e) \in G'_{\text{tgt}}.\text{po}\})^+$$

$$G'_{\text{src}}.\text{jf} = G'_{\text{tgt}}.\text{jf}$$

$$G'_{\text{src}}.\text{mo} = G'_{\text{tgt}}.\text{mo} \cup \{(d, a) \mid G'_{\text{tgt}}.\text{mo}(e, a)\} \cup \{(a, d) \mid G'_{\text{tgt}}.\text{mo}(a, e)\} \cup \{(d, e)\}$$

$$G'_{\text{src}}.\text{ew} = G'_{\text{tgt}}.\text{ew}$$

Now we check the consistency of $G'_{\text{src}}$. We already know that $G_{\text{src}}$ and $G'_{\text{tgt}}$ is consistent. Following the construction of $G'_{\text{src}}$, the (CF), (CFJ), (VISJ), (ICF), (ICFJ) constraints immediately hold. It remains to show that $G'_{\text{src}}$ satisfies (COH').

From the definition, there is no $G_{\text{src}}.\text{hb};G_{\text{src}}.\text{eco}$ as well as $G'_{\text{tgt}}.\text{hb};G'_{\text{tgt}}.\text{eco}$ cycle. Compared to $G_{\text{src}}$ and $G'_{\text{tgt}}$, the additional $G'_{\text{src}}.\text{mo}$ edges are from and to the event $d$. Assume the $\text{mo}$ edges to or from $d$ creates a $G'_{\text{src}}.\text{hb};G'_{\text{src}}.\text{eco}$ cycle.

However, for each $G'_{\text{src}}.\text{mo}(d, a)$ or $G'_{\text{src}}.\text{mo}(a, d)$ already there exists $G'_{\text{src}}.\text{mo}(w, e)$ or $G'_{\text{src}}.\text{mo}(e, w)$ respectively where $a \neq e$. Thus event $e$ results no new $G'_{\text{src}}.\text{hb};G'_{\text{src}}.\text{eco}$ cycle and hence $G'_{\text{src}}$ satisfies (COH').

**Source Execution Construction.** Next, we construct an execution $X_t \in \text{ex}_{\text{WEAKMO}}(G_{\text{tgt}})$.

If $W \subseteq (G_{\text{tgt}}.E \setminus X_t.E)$, then we find the corresponding execution $X_s \in \text{ex}_{\text{WEAKMO}}(G_{\text{src}})$ such that $X_s$ contains no event created for $\text{St}_{\sigma'}(x,\nu')$. Else if an event $w_t \in W$ is in $X_t$, then we know that we can find an execution with $w_s \in X_s.E$ and $X_s.E$ also contains an event $w'$ corresponding to store_{\sigma'}(x,\nu').$ Thus $X_s$ is as follows.

$$X_s.E = X_t.E \cup \{d \mid X_t.E \cap W \neq \emptyset\}$$

$$X_s.\text{po} = (X_t.\text{po} \cup \{(c, d), (d, w) \mid (c, w) \in \text{imm}(X_t.\text{po}) \cap (C \times W) \cap d \in (G_{\text{src}}.E \setminus G_{\text{tgt}}.E))\}^+$$

$$X_s.\text{rf} = X_t.\text{rf}$$

$$X_s.\text{mo} = X_t.\text{mo} \cup \{(d, w) \mid (d, w) \in ((G_{\text{src}}.E \setminus G_{\text{tgt}}.E) \times W)\}$$

$$\cup \{(a, d) \mid X_t.\text{mo}(a, w) \cap (d, w) \in ((G_{\text{src}}.E \setminus G_{\text{tgt}}.E) \times W) \cap \text{imm}(G_{\text{src}}.\text{po})\}$$

$$\cup \{(d, a) \mid X_t.\text{mo}(w, a) \cap (d, w) \in ((G_{\text{src}}.E \setminus G_{\text{tgt}}.E) \times W) \cap \text{imm}(G_{\text{src}}.\text{po})\}$$

**Source Execution Consistency.** Now we check the consistency of $X_s$.

Since $X_t$ is consistent, the (Well-formed), (total-MO), (Coherence), (Atomicity) constraints also hold for $X_s$. The (SC) constraint is affected only when $o = o' = sc$, in which case the new events introduce some $[SC], X_s.\text{po}_s;[SC]$ edges. These edges, however, can create a $(X_s.\text{psc}_0 \cup X_s.\text{psc}_F)$
cycle only when there is a \((X_t; \text{psc}_{\text{base}} \cup X_t; \text{psc}_F)\) cycle. Since \(X_t\) is consistent there is no \((X_t; \text{psc}_{\text{base}} \cup X_t; \text{psc}_F)\) cycle. Hence, \(X_s\) satisfies (SC) and, as a result, \(X_s\) is consistent.

**Same Behavior.** For locations \(y \neq x\), we have \(X_s.E_y = X.E_y\) and as a result \(\text{Behavior}(X_s)|_y = \text{Behavior}(X_t)|_y\) trivially holds. Now we check whether \(\text{Behavior}(X_s)|_x = \text{Behavior}(X_t)|_x\) holds. Note that any newly introduced event \(d \in X_s.E \setminus X_t.E\) is not \(X_s; \text{mo}\) maximal, because in that case there exists \(w \in W\) such that \(X_s; \text{mo}(d, w)\). Hence \(\text{Behavior}(X_s) = \text{Behavior}(X_t)\) holds.

**Race Preservation.** Moreover, if \(X_t\) is racy, then the new write \(d\) does not introduce any \(X_s; \text{sw}_{\text{C11}}\) edge in \(X_s\). Hence \(X_s\) is also racy. As a result, if the target execution has undefined behavior due to a data race, so does the source execution. 

**G.2 Read after Write (RAW)**

**Proof.** Recall the relationship between the two programs for the thread \(i\) affected by the transformation:

\[
\mathcal{P}_{\text{tgt}}(i) \subseteq \mathcal{P}_{\text{src}}(i) \cup \{\tau; \text{St}_o(x, v) \cdot \tau' | \tau; \text{St}_o(x, v) \cdot \text{Ld}_o'(x, _) \cdot \tau' \in \mathcal{P}_{\text{src}}(i) \land o' \subseteq o\}
\]

or

\[
\mathcal{P}_{\text{tgt}}(i) \subseteq \mathcal{P}_{\text{src}}(i) \cup \{\tau; \text{U}_o(x, v', v) \cdot \tau' | \tau; \text{U}_o(x, v', v) \cdot \text{Ld}_o'(x, _) \cdot \tau' \in \mathcal{P}_{\text{src}}(i) \land o' \subseteq o\}
\]

For all other threads \(j \neq i\), we have \(\mathcal{P}_{\text{tgt}}(j) = \mathcal{P}_{\text{src}}(j)\). Assume we have a target event structure, \(G_{\text{tgt}}\), and an execution, \(X_t \in \text{ex}_{\text{weakestmo}}(G_{\text{tgt}})\), extracted from it.

Let \(W\) be the set of writes with label \(\text{St}_o(x, v)\) or \(\text{U}_o(x, v', v)\) in the target event structure \(G_{\text{tgt}}\) for the respective accesses and whose po-suffix has some sequence of labels \(\tau\) such that \(\text{St}_o(x, v) \cdot \tau' \notin \mathcal{P}_{\text{src}}(i)\) or \(\text{U}_o(x, v', v) \cdot \tau' \notin \mathcal{P}_{\text{src}}(i)\) respectively. Then, because of the relationship between the two programs, we know that for each such \(w \in W\), \(\text{St}_o(x, v) \cdot \text{Ld}_o'(x, _) \cdot \tau' \in \mathcal{P}_{\text{src}}(i)\) or \(\text{U}_o(x, v', v) \cdot \text{Ld}_o'(x, _) \cdot \tau' \in \mathcal{P}_{\text{src}}(i)\) respectively for the appropriate \(\tau'\). Let \(C\) be the immediate \(G_{\text{tgt}}\)-po-successors of the events in \(W\).

**Source Event Structure Construction.**

To construct \(G_{\text{src}}\), we follow the construction steps of \(G_{\text{tgt}}\). For each target construction step that adds event \(e\) to \(G_{\text{tgt}}\) to get \(G'_{\text{tgt}}\), we perform one or more corresponding steps going from \(G_{\text{src}}\) to \(G'_{\text{src}}\). We do a case analysis on the event \(e\) of the target event structure.

**Case \(e \notin W\):** In this case we append event \(e\) to the source event structure as follows:

\[
\begin{align*}
G'_{\text{src}}.\text{E} &= G_{\text{src}}.\text{E} \uplus \{e\} \\
G'_{\text{src}}.\text{po} &= (G_{\text{src}}.\text{po} \uplus \{(a, e) \mid a \notin W \land \text{imm}(G'_{\text{tgt}}.\text{po})(a, e)\}) \\
&\uplus \{(r, e) \mid w \in W \land \text{imm}(G'_{\text{tgt}}.\text{po})(w, e)\}^+ \\
G'_{\text{src}}.\text{if} &= G_{\text{src}}.\text{if} \uplus \{(a, e) \mid G'_{\text{tgt}}.\text{if}(a, e)\} \\
G'_{\text{src}}.\text{mo} &= G'_{\text{tgt}}.\text{mo} \\
G'_{\text{src}}.\text{ew} &= G'_{\text{tgt}}.\text{ew}
\end{align*}
\]

Now we check the consistency of \(G'_{\text{src}}\) event structure. We already know that \(G_{\text{src}}\) and \(G'_{\text{tgt}}\) are consistent.

Following the definition of \(G'_{\text{src}}\) the \((\text{CF}), (\text{CFJ}), (\text{VISJ}), (\text{ICF}), (\text{ICFJ}), (\text{COH'})\) constraints immediately hold and hence \(G'_{\text{src}}\) is also consistent.
Case $e \in W$: In this case we first append event $e$ and then event $r$ with $r.\text{lab} = \text{Ld}_{o'}(x, v)$ to $G_{\text{src}}$ as follows:

$$G'_{\text{src}}.E = G_{\text{src}}.E \cup \{r, e\} \quad \text{where} \quad r.\text{lab} = \text{Ld}_{o'}(x, v)$$

$$G'_{\text{src}}.\text{po} = (G_{\text{src}}.\text{po} \cup \{(e, r), (a, e) \mid \text{imm}(G_{\text{tgt}}.\text{po})(a, e)\})^+$$

$$G'_{\text{src}}.\text{if} = G_{\text{src}}.\text{if} \cup \{(e, r)\}$$

$$G'_{\text{src}}.\text{mo} = G_{\text{tgt}}.\text{mo}$$

$$G'_{\text{src}}.\text{ew} = G_{\text{tgt}}.\text{ew}$$

Now we check the consistency of $G'_{\text{src}}$.

We already know that $G_{\text{src}}$ and $G_{\text{tgt}}$ is consistent. Following the construction of $G'_{\text{src}}$, the (CF), (CFJ), (VISJ), (ICF) constraints immediately hold. It remains to show that $G'_{\text{src}}$ satisfies (COH'). The outgoing edges from $r$ are $G'_{\text{src}}.\text{fr}$. Hence for an outgoing edge $G'_{\text{src}}.\text{fr}(r, a)$, there is $G_{\text{src}}.\text{mo}(e, a)$ edge. If $G'_{\text{src}}.\text{fr}(r, a)$ results in a $G'_{\text{src}}.\text{hb}; G'_{\text{src}}.\text{eco}$ cycle, then $G_{\text{src}}.\text{hb}; G_{\text{src}}.\text{eco}$ cycle is already there in $G_{\text{src}}$. But we know that $G_{\text{src}}$ is consistent and hence $G_{\text{src}}.\text{hb}; G_{\text{src}}.\text{eco}$ is not possible. Hence a contradiction and $G'_{\text{src}}.\text{hb}; G'_{\text{src}}.\text{eco}$ is also not possible. Thus $G'_{\text{src}}$ preserves (COH') and $G'_{\text{src}}$ is consistent.

Source Execution Construction. Next, we construct an execution $X_r \in \text{ex}_{\text{weakestmo}}(G_{\text{tgt}})$.

If $W \subseteq (G_{\text{tgt}} \setminus X_r.E)$, then we find the corresponding execution $X_s \in \text{ex}_{\text{weakestmo}}(G_{\text{src}})$ such that $X_s$ contains no $\text{St}_o(x, v)$ or $U_o(x, v', v)$. In that case $X_s$ also does not contain any event created for $\text{Ld}_{o'}(x, v)$ access.

Else if an event $w \in W$ is in $X_r$, then we know that we can find a source execution $X_s$ which contains both $w$ and $r$. Thus $X_s$ is as follows.

Thus $X_s$ is as follows.

$$X_s.E = X_r.E \cup \{r \mid X_r.E \cap W \neq \emptyset\}$$

$$X_s.\text{po} = (X_r.\text{po} \cup \{(w, r), (r, c) \mid (w, c) \in \text{imm}(X_r.\text{po}) \cap (W \times C) \land r \in (G_{\text{src}}.E \setminus G_{\text{tgt}}.E)\})^+$$

$$X_s.\text{rf} = X_r.\text{rf} \cup \{(w, r) \mid w \in X_r.E \cap W\}$$

$$X_s.\text{mo} = X_r.\text{mo}$$

Source Execution Consistency. Now we check the consistency of $X_s$.

We know that $X_r$ is consistent. The (Well-formed), (total-MO), (Coherence), (Atomicity) constraints hold as they hold for $X_r$. Considering the (SC) constraint we observe that if $o = o' = \text{sc}$, then $r'$ introduces a $[\text{SC}], X_s.\text{po}_s$; $[\text{SC}]$ edge. This edge can create a $(X_s.\text{psc}_\text{base} \cup X_s.\text{psc}_\text{f})$ cycle only when there is a $(X_r.\text{psc}_\text{base} \cup X_r.\text{psc}_\text{f})$ cycle. Since $X_r$ is consistent there is no $(X_r.\text{psc}_\text{base} \cup X_r.\text{psc}_\text{f})$ cycle. Hence there is no $(X_s.\text{psc}_\text{base} \cup X_s.\text{psc}_\text{f})$ cycle and $X_s$ satisfies (SC). As a result, $X_s$ is consistent.

Same Behavior.

Now we check whether $\text{Behavior}(X_s) = \text{Behavior}(X_r)$ holds.

For locations $y \neq x$, $\text{Behavior}(X_s) = \text{Behavior}(X_r)$ holds.

For $x$ load $r'$ does not introduce any new $\text{mo}$ edge and hence does not affect behavior of $X_s$.

Hence $\text{Behavior}(X_s) = \text{Behavior}(X_r)$ holds.

Race Preservation.

Moreover, if $X_r$ is racy, then the new read $r'$ does not introduce any new $(X_s.\text{sw}_{C11} \setminus X_s.\text{po})$ edge in $X_s$. Hence $X_s$ is also racy. As a result, if the target execution has undefined behavior due to data race then the source execution also has undefined behavior due to data race.

□
G.3 Read after Read (RAR)

**Proof.** Recall the relationship between the two programs for the thread $i$ affected by the transformation:

$$\mathbb{P}_{\text{tgt}}(i) \subseteq \mathbb{P}_{\text{src}}(i) \cup \{ \tau \cdot \text{ld}_o(x, v) \cdot \tau' \mid \tau \cdot \text{ld}_o(x, v) \cdot \text{ld}_o'(x, _) \cdot \tau' \in \mathbb{P}_{\text{src}}(i) \wedge o' \not\in o \}$$

For all other threads $j \neq i$, we have $\mathbb{P}_{\text{tgt}}(j) = \mathbb{P}_{\text{src}}(j)$. Assume we have a target event structure, $G_{\text{tgt}}$, and an execution, $X_i \in \text{exweakestmo}(G_{\text{tgt}})$, extracted from it.

Let $R$ be the set of loads with label $\text{ld}_o(x, v)$ in the target event structure $G_{\text{tgt}}$ whose po-suffix has some sequence of labels $\tau'$ such that $\text{ld}_o(x, v) \cdot \tau' \not\in \mathbb{P}_{\text{src}}(i)$. Then, because of the relationship between the two programs, we know that for each such $r \in W$, for the appropriate $\tau'$, $\text{ld}_o(x, v) \cdot \text{ld}_o'(x, _) \cdot \tau' \in \mathbb{P}_{\text{src}}(i)$ holds. Let $C$ be the immediate $G_{\text{tgt}}$-po-successors of the events in $R$.

**Source Event Structure Construction.**

To construct $G_{\text{src}}$, we follow the construction steps of $G_{\text{tgt}}$. For each target construction step that adds event $e$ to $G_{\text{tgt}}$ to get $G_{\text{tgt}}'$, we perform one or more corresponding steps going from $G_{\text{src}}$ to $G_{\text{src}}'$. We do a case analysis on the event $e$ of the target event structure.

**Case** $e \not\in R$: In this case we append event $e$ to the source event structure as follows:

- $G_{\text{src}}'.E = G_{\text{src}}.E \cup \{ e \}$
- $G_{\text{src}}'.po \equiv (G_{\text{src}}.po \cup \{(a, e) \mid a \not\in R \wedge \text{imm}(G_{\text{tgt}}.po)(a, e)\})^+$
- $G_{\text{src}}'.jf = G_{\text{tgt}}'.jf$
- $G_{\text{src}}'.mo = G_{\text{tgt}}'.mo$
- $G_{\text{src}}'.ew = G_{\text{tgt}}'.ew$

Now we check the consistency of $G_{\text{src}}'$ event structure. We already know that $G_{\text{src}}$ and $G_{\text{tgt}}'$ are consistent.

Following the definition of $G_{\text{src}}'$, the (CF), (CFJ), (VISJ), (ICF), (ICFJ), (COH') constraints immediately hold and hence $G_{\text{src}}'$ is also consistent.

**Case** $e \in R$: In this case we first append event $e$ and then event $r$ with $r.lab = \text{ld}_o'(x, v)$ to $G_{\text{src}}$ as follows:

- $G_{\text{src}}'.E = G_{\text{src}}.E \cup \{ d, e \} \quad \text{where } d.lab = \text{ld}_o'(x, v)$
- $G_{\text{src}}'.po \equiv (G_{\text{src}}.po \cup \{(e, d), (a, e) \mid \text{imm}(G_{\text{tgt}}.po)(a, e)\})^+$
- $G_{\text{src}}'.jf = G_{\text{src}}.jf \cup \{(a, e), (a, d) \mid G_{\text{tgt}}.jf(a, e)\}$
- $G_{\text{src}}'.mo = G_{\text{tgt}}'.mo$
- $G_{\text{src}}'.ew = G_{\text{tgt}}'.ew$

Now we check the consistency of $G_{\text{src}}'$.

We already know that $G_{\text{src}}$ and $G_{\text{tgt}}'$ is consistent. Following the construction of $G_{\text{src}}'$, the (CF), (CFJ), (VISJ), (ICF), (ICFJ) constraints immediately hold. It remains to show that $G_{\text{src}}'$ satisfies (COH'). The outgoing edges from $d$ are $G_{\text{src}}'.fr$. Hence for an outgoing edge $G_{\text{src}}.fr(d, a)$ there is $G_{\text{src}}'.fr(e, a)$ as well as $G_{\text{tgt}}'.fr(e, a)$ edges. Hence if $G_{\text{src}}.fr(d, a)$ results in a $G_{\text{src}}'.hb$; $G_{\text{src}}'.eco$ cycle, then there is also $G_{\text{tgt}}'.hb$; $G_{\text{tgt}}'.eco$ cycle. But we know that $G_{\text{tg}}'$ is consistent and hence $G_{\text{tg}}'.hb$; $G_{\text{tg}}'.eco$ cycle is not possible. Hence a contradiction and $G_{\text{src}}.hb$; $G_{\text{src}}.eco$ cycle is also not possible. Thus $G_{\text{src}}'$ preserves (COH') and $G_{\text{src}}'$ is consistent.
Source Execution Construction. Next, we construct an execution $X_t \in \text{ex}_{\text{WEAKESTMO}}(G_{tgt})$.

If $R \subseteq (G_{tgt} \setminus X_t, E)$, then we find the corresponding execution $X_s \in \text{ex}_{\text{WEAKESTMO}}(G_{src})$ such that $X_s$ contains no $Ld_o(x, v)$. In that case $X_s$ also does not contain any event created for $Ld_o(x, v)$ access.

Else if an event $r \in R$ is in $X_t$, then we know that we can find a source execution $X_s$ which contains both $r$ and $d$. Thus $X_s$ is as follows.

Thus $X_s$ is as follows.

$$X_s \cdot E = X_t \cdot E \cup \{d \mid X_t \cdot E \cap R \neq \emptyset\}$$

$$X_s \cdot po = (X_t \cdot po \cup \{(r, d), (d, c) \mid (r, c) \in \text{imm}(X_t \cdot po) \cap (R \times C) \land d \in (G_{src} \cdot E \setminus G_{tgt} \cdot E))\}^+$$

$$X_s \cdot rf = X_t \cdot rf \cup \{(a, d) \mid a \in \text{dom}(X_t \cdot rf) \setminus [R]\}$$

$$X_s \cdot mo = X_t \cdot mo$$

Source Execution Consistency. Now we check the consistency of $X_s$.

We know that $X_t$ is consistent. The (Well-formed), (total-MO), (Coherence), (Atomicity) constraints hold as they hold for $X_t$. Considering the (SC) constraint we observe that if $o = o' = sc$, then $r'$ introduces a $[SC], X_s.psc$; $[SC]$ edge. This edge can create a $(X_s.psc_{base} \cup X_s.psc_f)$ cycle only when there is a $(X_t.psc_{base} \cup X_t.psc_f)$ cycle. Since $X_t$ is consistent there is no $(X_t.psc_{base} \cup X_t.psc_f)$ cycle. Hence there is no $(X_s.psc_{base} \cup X_s.psc_f)$ cycle and $X_s$ satisfies (SC). As a result, $X_s$ is consistent.

Same Behavior.

Now we check whether $\text{Behavior}(X_s) = \text{Behavior}(X_t)$ holds.

For locations $y \neq x$, $\text{Behavior}_{|y}(X_s) = \text{Behavior}_{|y}(X_t)$ holds.

For $x$, load $r'$ does not introduce any new $mo$ edge and hence does not affect behavior of $X_s$.

Hence $\text{Behavior}(X_s) = \text{Behavior}(X_t)$ holds.

Race Preservation.

Moreover, if $X_t$ is racy, then the new read $d$ does not introduce any new $X_t.hb_{C11} \setminus X_s.po$ relation in $X_s$. Hence $X_s$ is also racy. As a result, if the target execution has undefined behavior due to data race then the source execution also has undefined behavior due to data race. \qed

G.4 Non-Atomic Read-Write (naRW)

PROOF. Recall the relationship between the two programs for the thread $i$ affected by the transformation:

$$P_{tgt}(i) \subseteq P_{src}(i) \cup \{r \cdot r' \mid r \cdot \text{Ld}_{na}(x, v) \cdot \text{St}_{na}(x, v) \cdot r' \in P_{src}(i)\}$$

For all other threads $j \neq i$, we have $P_{tgt}(j) = P_{src}(j)$. Assume we have a target event structure, $G_{tgt}$, and an execution, $X_t \in \text{ex}_{\text{WEAKESTMO}}(G_{tgt})$, extracted from it.

Let $C$ be the set of events the target event structure $G_{tgt}$ whose po-suffix has some sequence of labels $r'$ such that $c \cdot r' \notin P_{src}(i)$ where $c \in C$. Also let $D$ be the set of events which are immediate po-successors of events in $C$. Then, because of the relationship between the two programs, we know that for each such $c \in C$ and $c \in r$, $c \cdot \text{Ld}_{na}(x, v) \cdot \text{St}_{na}(x, v) \cdot r' \in P_{src}(i)$ for the appropriate $r'$.

Source Event Structure Construction.

To construct $G_{src}$, we follow the construction steps of $G_{tgt}$. For each target construction step that adds event $e$ to $G_{tgt}$ to get $G'_{tgt}$, we perform one or more corresponding steps going from $G_{src}$ to $G'_{src}$. We do a case analysis on the event $e$ of the target event structure.
Case $e \in C$: In this case we append event $e$ followed by $Ld_{sa}(x, wval)$ justified from a write $s$ and $St_{sa}(x, s.wval)$ to the source event structure as follows:

$$G'_{src}.E = G_{src} \cup \{e, r, w\} \quad \text{where } r.lab = Ld_{sa}(x, _) \text{ and } w = St_{sa}(x, _)$$

$$G'_{src}.po = (G_{src}.po \cup \{(a, e) \mid G.tgt.po(a, e)\})^+$$

$$G'_{src}.jf = G_{src}.jf \cup \{(a, e) \mid G.tgt.jf(a, e) \land \exists W(G'_{src}, s, r)\}$$

$$G'_{src}.mo = G_{src}.mo \cup \{(a, w) \mid a \in (G_{src}.W_x \setminus WA) \cup \{(w, a) \mid a \in WA\}$$

where $WA = \{a \mid (G'.ew^?; G'.mo)(s, a)\}$

$$G'_{src}.ew = G_{src}.ew \cup \{(a, e) \mid G.tgt.ew(a, e)\}$$

Now we check the consistency of $G'_{src}$.

We already know that $G_{src}$ and $G'_{tgt}$ is consistent. Following the construction of $G'_{src}$ and considering the definition of Remark 3, the (CF), (CFJ), (VISJ), (ICF), (ICFJ) constraints immediately hold. It remains to show that $G'_{src}$ satisfies (COH$'$). Again following the Remark 3 definition, additional events $r$ and $w$ do not create any $G'_{src}.hb; G'_{src}.eco^?$ cycle. Hence $G'_{src}$ satisfies (COH$'$) and is consistent. Case $e \notin C$: In this case we append event $e$ to the source event structure. However, if $e$ is justified-from $s$ in $G'_{tgt}$ and happens-after the newly newly appended non-atomic store from $(G_{src}.E \setminus G_{tgt}.E)$ in $G'_{src}$, then $e$ is justified-from the new store $St_{sa}(X, s.wval)$. Let $W \subseteq (G_{src}.E \setminus G_{tgt}.E)$ be the set of such store events. Note that id event $e$ happens-after event $w \in W$, then there exists an intermediate event $d \in D$. Thus we construct $G'_{src}$ as follows:

$$G'_{src}.E = G_{src}.E \cup \{e\}$$

$$G'_{src}.po = (G_{src}.po \cup \{(a, e) \mid G_{tgt}.po(a, e)\}$$

$$\cup \{(w, e) \mid w \in W \land e \in \text{codom}([C]; \text{imm}(G_{tgt}.po); [D]))^+$$

$$G'_{src}.jf = G_{src}.jf \cup \{(a, e) \mid G_{tgt}.jf(a, e) \land e \notin \text{codom}([D]; G_{src}.hb)\}$$

$$\cup \{(a, e) \mid G_{tgt}.jf(a, e) \land e \in \text{codom}([D]; G_{src}.hb)\}$$

$$G'_{src}.mo = G_{src}.mo \cup \{(a, e) \mid G_{tgt}.mo(a, e)\} \cup \{(e, a) \mid G_{tgt}.mo(e, a)\}$$

$$G'_{src}.ew = G_{src}.ew \cup \{(a, e) \mid G_{tgt}.ew(a, e)\}$$

Now we check the consistency of $G'_{src}$.

We already know that $G_{src}$ and $G'_{tgt}$ is consistent. Following the construction of $G'_{src}$, the (CF), (CFJ), (VISJ), (ICF), (ICFJ) constraints immediately hold. It remains to show that $G'_{src}$ satisfies (COH$'$).

Assume there is a $G'_{src}.hb; G'_{src}.eco^?$ cycle. We know there is no $G_{src}.hb; G_{src}.eco^?$ cycle. Hence the cycle involves event $e$. However, if event $e$ introduces a $G'_{src}.hb; G'_{src}.eco^?$, then from the definition, there is a $G'_{tgt}.hb; G'_{tgt}.eco^?$ cycle which is a contradiction. Hence $G'_{src}$ satisfies (COH$'$) and $G'_{src}$ is consistent.

Source Execution Construction. Next, we construct an execution $X_f \in \text{ex}_{\text{weakestmo}}(G_{tgt})$.

If $X_f.E$ does not contain any event in $C$ then we find the corresponding execution $X_s$ such that $X_s \in \text{ex}_{\text{weakestmo}}(G_{src})$ and $X_f.E$ contains no corresponding $St_{sa}(x, v)$ and $Ld_{sa}(x, v)$ events.

Else if an event $e \in C$ is in $X_f$, then we know that we can find an execution with $r, w \in X_s.E$ where $r.lab = Ld_{sa}(x, _) \text{ and } w.lab = St_{sa}(x, _)$. Thus $X_s$ is as follows:

\[
X_s.E = X_t \cup \{r, w \mid X_t.E \cap C \neq \emptyset \}
\]
\[
X_s.po = (X_t.po \cup \{(c, r), \ldots \})
\]

As a result, we cannot create \(Ld(X, 2)\) directly.

Given the program in consider the transformation deletes the \(X_{na} = 1\) access and hence results in an target execution as shown in . This execution has a defined behavior according to the \texttt{weakestmo-llvm} model as there is no write-write race in this execution.

The execution can be extracted from the target event structure in Fig. 35c.

Given this target event structure we cannot contruct the source event structure as once we introduce \(St_{na}(X, 1)\), we cannot create \(Ld(X, 2)\) directly.

\textbf{Race Preservation.} Moreover, if \(X_t\) is racy, then the new write \(d\) does not introduce any \(X_s.sw_{C11}\) edge in \(X_s\). Hence \(X_s\) is also racy. As a result, if the target execution has undefined behavior due to a data race, so does the source execution. \(\square\)

\section{Non-Adjacent Access Elimination (NA-OW)}

\textbf{Definition 11.} A trace \(\tau\) satisfies the intermediate condition for a location, \(x\), which is written as \(\text{GoodInt}(x, \tau)\), if:

- it contains no \(x\)-accesses, i.e., \(\tau \neq \tau_1\cdot[R\cdotW\cdot\tau_2]_{x}^{x}\) for all \(\tau_1\) and \(\tau_2\); and
- it contains no rel-acq pairs, i.e., \(\tau \neq \tau_1\cdot[\text{Rel}]\cdot\tau_2\cdot[\text{Acq}]\cdot\tau_3\) for all traces \(\tau_1\), \(\tau_2\), and \(\tau_3\).

Let \(E_{\tau}\) be the events corresponding to \(\tau\). If \(E_{\tau}\) has no release access then \(St_{na}(x, v')\) could reorder with \(E_{\tau}\) and placed in adjacency with \(St_{na}(x, v)\). Then \(St_{na}(x, v')\) could be deleted by overwritten write (OW) transformation. But if \(E_{\tau}\) contains a release operation then \(St_{na}(x, v')\) cannot be reordered with \(E_{\tau}\). Hence in this proof we consider the cases where \(C\) contains release access. Before going to the proof we discuss a special case for \texttt{weakestmo-llvm} model.

\textbf{Special Case.} Given the program in consider the transformation deletes the \(X_{na} = 1\) access and hence results in an target execution as shown in . This execution has a defined behavior according to the \texttt{weakestmo-llvm} model as there is no write-write race in this execution.

The execution can be extracted from the target event structure in Fig. 35c.
Soham Chakraborty and Viktor Vafeiadis

\[ X = 2; \]
\[ X_{\text{na}} = 1; \]
\[ Y_{\text{rel}} = 1; \]
\[ t = Z_{\text{rlx}}; \]
\[ X_{\text{na}} = 3; \]
\[ \text{if} \ (Y == 1) \]
\[ \text{if} \ (X == 2) \]

\( \tau \)

for the appropriate relationship between the two programs, we know that for each such \( \tau \) some sequence of labels \( \tau \) that \( X_{\text{na}}(x, v') \cdot \tau \cdot X_{\text{na}}(x, v) \notin P_{\text{src}}(i) \). Then, because of the relationship between the two programs, we know that for each such \( w \in W, X_{\text{na}}(x, v') \cdot \tau \cdot X_{\text{na}}(x, v) \notin P_{\text{src}}(i) \) for the appropriate \( \tau \).

Let
\( C \) be the set of first event in the sequence \( \tau \).
\( B \) be the set of immediate \( G_{\text{tgt}} \cdot \text{po-predecessor of} \ C \).
\( F = G_{\text{tgt}} \cdot \text{Rel}_{\neq x} \) are the release operations in \( \tau \).
\( W \) be the set of the respective \( X_{\text{na}}(x, v) \) labelled events and \( W \subseteq \text{codom}([F]; G_{\text{tgt}} \cdot \text{po}) \).
\( R \) be the set of reads such that \( R \subseteq (\text{codom}([B]; G_{\text{tgt}} \cdot \text{po}; [F]G_{\text{tgt}} \cdot \text{swr}; G_{\text{tgt}} \cdot \text{hr}) \cap G_{\text{tgt}} \cdot R_{x}) \) and \( M : R \mapsto G_{\text{src}} \cdot E \) maps a read in \( R \) to the corresponding read in source event structure. Let \( P \) be the

Fig. 35. NA-OW example executions and weakestmo-llvm event structures.

However, note that, \( \text{Ld}(X, 2) \) is in read-write race with \( \text{St}_{\text{na}}(X, 3) \). Hence the program has undefined behavior in weakestmo-c11 and in weakestmo-llvm the respective event may return \( u \) which can be evaluated to 2.

However, if \( \text{St}_{\text{na}}(X, 3) \) is appended after \( \text{Ld}(X, 2) \), then we cannot create \( \text{Ld}(X, u) \) in the source event structure directly. Hence \( G_{\text{src}} \) requires to create a \( \text{St}_{\text{na}}(X, _{-}) \) before \( \text{Ld}(X, u) \) as shown in.

**Proof.** Let \( W \) be the set of stores of thread \( i \) of \( G_{\text{tgt}} \) with label \( \text{St}_{\phi}(x, v) \), and whose po-prefix has some sequence of labels \( \tau \) such that \( \text{St}_{\text{na}}(x, v') \cdot \tau \cdot \text{St}_{\text{na}}(x, v) \notin P_{\text{src}}(i) \). Then, because of the relationship between the two programs, we know that for each such \( w \in W, \text{St}_{\text{na}}(x, v') \cdot \tau \cdot \text{St}_{\text{na}}(x, v) \notin P_{\text{src}}(i) \) for the appropriate \( \tau \).

Let
\( C \) be the set of first event in the sequence \( \tau \).
\( B \) be the set of immediate \( G_{\text{tgt}} \cdot \text{po-predecessor of} \ C \).
\( F = G_{\text{tgt}} \cdot \text{Rel}_{\neq x} \) are the release operations in \( \tau \).
\( W \) be the set of the respective \( \text{St}_{\phi}(x, v) \) labelled events and \( W \subseteq \text{codom}([F]; G_{\text{tgt}} \cdot \text{po}) \).
\( R \) be the set of reads such that \( R \subseteq (\text{codom}([B]; G_{\text{tgt}} \cdot \text{po}; [F]G_{\text{tgt}} \cdot \text{swr}; G_{\text{tgt}} \cdot \text{hr}) \cap G_{\text{tgt}} \cdot R_{x}) \) and \( M : R \mapsto G_{\text{src}} \cdot E \) maps a read in \( R \) to the corresponding read in source event structure. Let \( P \) be the
\[\tau_x\] be the sub-sequence from \(f \in F\) to \(w \in W\) such that \(G_{tgt}.po(f, w)\) holds and there is no \(f' \in F\) such that \(G_{tgt}.po(f', f)\).

\(pc(\tau_x)\) be the \(G_{src}.po\)-maximal event appended to the source event structure.

\(EW(\tau_x)\) be the set of writes on \(x\) with label \(St_{NA}(x, v)\) in \(G_{src}\). The writes in \(EW(\tau_x)\) are equal writes, that is, \(\forall w_1, w_2 \in EW(\tau_x), G_{src}.ew(w_1, w_2)\) holds.

\(D\) be the set of events deleted from source event structure.

\(S\) be the events of \(\tau_x\) that is, \(S \subseteq \text{codom}([F]), G_{tgt}.po \cup \text{dom}(G_{tgt}.po; [W])\).

**Source Event Structure Construction.** To construct \(G_{src}\), we follow the construction steps of \(G_{tgt}\). For each target construction step that adds event \(e\) to \(G_{tgt}\) to get \(G'_{tgt}\), we perform one or more corresponding steps going from \(G_{src}\) to \(G'_{src}\). We do a case analysis on the event \(e\) of the target event structure.

**Case** \(e \in C\):

We append a \(St_{NA}(x, v')\) event \(d\) followed by event \(e\) as follows. The immediate \(G_{tgt}.po\) predecessor of \(e\) is \(b\).

Let \(s\) be the maximal-visible write on \(x\) w.r.t \(b\), that is, \(\exists \text{exists} W(G_{src}, s, b)\) hold. We refer to the event \(s\) to create the \(mo\) relations to/from \(d\).

\[
\begin{align*}
G'_{src}.E & = G_{src}.E \cup \{d, e\} \quad \text{where } d.\text{lab} = St_{NA}(x, v') \\
G'_{src}.po & = (G_{src}.po \cup \{(d, e)\} \cup \{(b, d) | (b, e) \in G_{tgt}.po\})^+ \\
G'_{src}.jf & = G_{tgt}.jf \\
G'_{src}.mo & = G_{src}.mo \cup \{(s, d)\} \cup \{(p, d) | G_{src}.mo(p, s)\} \cup \{(d, p) | G_{src}.mo(p, s)\} \\
& \quad \text{where } \exists \text{exists} W(G_{src}, s, b). \\
G'_{src}.ew & = G_{src}.ew \cup \{(a, e) | G'_{src}.ew(a, e)\}
\end{align*}
\]

Also we update \(D\) to \(D \cup \{d\}\). Now we check the consistency of \(G'_{src}\). We already know that \(G_{src}\) and \(G'_{tgt}\) is consistent. Following the construction of \(G'_{src}\), the (CF), (CFJ), (VIS), (ICF), (ICFJ) constraints immediately hold. It remains to show that \(G'_{src}\) satisfies (COH').

From the definition, there is no \(G_{src}.hb; G_{src}.eco\) as well as \(G_{tgt}.hb; G_{tgt}.eco\) cycle. Compared to \(G_{src}\) and \(G_{tgt}\), the additional \(G_{src}.mo\) edges are from and to the event \(d\). Assume the \(mo\) edges to or from \(d\) creates a \(G'_{src}.hb; G'_{src}.eco\) cycle. However, for each \(G'_{src}.mo(d, a)\) or \(G'_{src}.mo(a, d)\) already there exists \(G'_{src}.mo(s, a)\) or \(G'_{src}.mo(a, s)\) respectively. Thus event \(d\) as well as \(e\) results no new \(G'_{src}.hb; G'_{src}.eco\) cycle and hence \(G'_{src}\) satisfies (COH').

**Case** \(e \in S\): Let \(e\) is in sequence \(\tau_x\). Two possibilities:

**Subcase** There exists an event \(e_s\) such that \(\text{imm}(G_{src}.po)(pc(\tau_x), e_s): pc' = pc[\tau_x \mapsto e_s]\). In this case \(G'_{src} = G_{src}\) and hence \(G'_{src}\) is consistent.

**Subcase Otherwise:** We take two steps where we first create an intermediate event structure \(G''\) by appending \(e\). Next, we append a sequence of events \(Q\) where a read \(r_c\) reads from a maximal visible write \(w_v\) in \(G_{src}\), that is, \(\exists \text{exists} W(G_{src}, w_v, r_c)\) until we append an event \(w_c = St_{NA}(x, v)\). Moreover, \(pc' = pc[\tau_x \mapsto e]\).

Next, we append a sequence of events \(Q\) where a read \(r_c\) reads from a maximal visible write \(w_v\) in \(G_{src}\), that is, \(\exists \text{exists} W(G_{src}, w_v, r_c)\) until we append an event \(w_c = St_{NA}(x, v)\).

Thus \(G'_{src}\) is as follows:
The outgoing structure corresponding to $G_{\text{src}}'$ may not create new edges from and to the event $\{e\} \cup Q$. From the definition, there is no $G_{\text{src}.\text{hb}}$; $G_{\text{src}.\text{eco}}$ as well as $G'_{\text{tgt}.\text{hb}}; G'_{\text{tgt}.\text{eco}}$ cycle. Compared to $G_{\text{src}}$ and $G'_{\text{tgt}}$, the additional $G'_{\text{src}.\text{hb}}$ and $G_{\text{src}.\text{eco}}$ edges are from and to the event $\{e\} \cup Q$. The edge from/to $e$ does not create new $G'_{\text{src}.\text{hb}}$; $G'_{\text{src}.\text{eco}}$ cycle as there is no $G'_{\text{tgt}.\text{hb}}; G'_{\text{tgt}.\text{eco}}$ cycle. Also the outgoing $G'_{\text{src}.\text{hb}}$ and $G_{\text{src}.\text{eco}}$ edges from events in $Q$ are only to other events in $Q$. Therefore, there is no $G'_{\text{tgt}.\text{hb}}; G'_{\text{tgt}.\text{eco}}$ cycle to/from $Q$ events. Thus $G'_{\text{src}}$ satisfies (COH') and $G'_{\text{src}}$ is consistent.

**Case $e \in R$:**

In this case event $e$ reads from a visible write $w_1$ which is now overwritten. $w_1$ has a $G'_{\text{tgt}.\text{po}}$-successor sequence $\tau$ which includes $f \in F$ such that $G'_{\text{tgt}.\text{po}}(w_1, f)$. From the construction, $f$ has a $G_{\text{src}.\text{po}}$ event $w_c$ such that $w_c.\text{lab} = St_{\text{Na}}(x, v)$. Consider we append event $r$ in source event structure corresponding to $e$.

Following the WEAKESTMO-C11 model, if we append an event corresponding to $e$ it results in race and hence the source has undefined behavior. Hence the transformation is correct.

Now we consider the WEAKESTMO-LLVM model. If $r \in U$, then there is a write-write race and in that case the source program has undefined behavior. Hence the transformation is correct.

The according to WEAKESTMO-LLVM read-write race has define behavior. Hence we continue the event structure construction when $r$ is a load, that is, $r \in \text{Ld}$.

We append $r$ to the $G_{\text{src}}$ as follows:

\[
G'_{\text{src}}.E = G_{\text{src}}.E \uplus \{r\} \quad \text{where } r.\text{lab} = \text{Ld}(x, u)\text{ which we evaluate } u \text{ to } w_1.\text{wval}
\]

\[
G'_{\text{src}.\text{po}} = (G_{\text{src}.\text{po}} \uplus \{(a, r) \mid G'_{\text{tgt}.\text{po}}(a, e)\})^+
\]

\[
G'_{\text{src}.\text{jf}} = G'_{\text{tgt}.\text{jf}} \uplus \{(w_c, r)\}
\]

\[
G'_{\text{src}.\text{mo}} = G_{\text{src}.\text{mo}}
\]

\[
G'_{\text{src}.\text{ew}} = G_{\text{src}.\text{ew}}
\]

Also we update the mapping $M' = M[e \mapsto r]$. 

Now we check the consistency of $G'_{\text{src}}$. We already know that $G_{\text{src}}$ and $G'_{\text{tgt}}$ is consistent. Following the construction of $G'_{\text{src}}$, the (CF), (CFJ), (VIS), (ICF), (ICFJ) constraints immediately hold. It remains to show that $G'_{\text{src}}$ satisfies (COH').

From the definition, there is no $G_{\text{src}}.hb; G_{\text{src}}.eco^2$ cycle. So any new $G'_{\text{src}}.hb; G'_{\text{src}}.eco^2$ cycle involves $r$. The incoming edges to $r$ is $G'_{\text{src}}.po; G'_{\text{src}}(w_c, r)$ and the outgoing edges are $G'_{\text{src}}.fr$ edges when $w_c \in G'_{\text{tgt}}.E$. As well. These edges cannot constitute a $G'_{\text{src}}.hb; G'_{\text{src}}.eco^2$ cycle as there is no $G'_{\text{tgt}}.hb; G'_{\text{tgt}}.eco^2$ cycle involving $w_c$. As a result, $G'_{\text{src}}$ preserves (COH') and $G'_{\text{src}}$ is consistent.

Case $e \in W$:
Either there already exists a write event $w_c \in EW(\tau_x)$ with $w_c.\text{lab} = St_{\text{NA}}(x, v)$ such that $\text{imm}(G_{\text{src}}.po)(pc(\tau_x), w_c)$ or we append event $e$.

Subcase $\exists w_c \in EW(\tau_x)$ such that $w_c.\text{lab} = St_{\text{NA}}(x, v)$, $\text{imm}(G_{\text{src}}.po)(pc(\tau_x), w_c)$:
In this case $pc' = pc[\tau_x \mapsto w_c]$ and $G'_{\text{src}}$ is as follows:

$G'_{\text{src}}.E = G_{\text{src}}.E$
$G'_{\text{src}}.po = G_{\text{src}}.po$
$G'_{\text{src}}.jf = G_{\text{src}}.jf$
$G'_{\text{src}}.mo = G_{\text{src}}.mo$
$G'_{\text{src}}.ew = G_{\text{src}}.ew \cup \{(a, w_c) \mid G'_{\text{tgt}}.ew(a, e)\}$

Now we check the consistency of $G'_{\text{src}}$. We already know that $G_{\text{src}}$ and $G'_{\text{tgt}}$ is consistent. Following the construction of $G'_{\text{src}}$, the (CF), (CFJ), (VIS), (ICF), (ICFJ) constraints immediately hold. It remains to show that $G'_{\text{src}}$ satisfies (COH').

From the definition, there is no $G_{\text{src}}.hb; G_{\text{src}}.eco^2$ cycle. So any new $G'_{\text{src}}.hb; G'_{\text{src}}.eco^2$ cycle involves new outgoing $G'_{\text{src}}.rf$ from $w_c$. However, $G'_{\text{tgt}}$ also has corresponding outgoing $G'_{\text{tgt}}.rf$ edge from $e$ and there is no $G'_{\text{tgt}}.hb; G'_{\text{tgt}}.eco^2$ cycle involving $e$. Hence there is no $G'_{\text{src}}.hb; G'_{\text{src}}.eco^2$ cycle involving $w_c$. As a result, $G'_{\text{src}}$ satisfies (COH') and $G'_{\text{src}}$ is consistent.

Subcase Otherwise: We append $e$ to $G_{\text{src}}$ and construct $G'_{\text{src}}$ as follows where $pc'(\tau_x) = e$.

$G'_{\text{src}}.E = G_{\text{src}}.E \cup \{e\}$
$G'_{\text{src}}.po = (G_{\text{src}}.po \cup \{(pc(\tau_x), e)\})^+$
$G'_{\text{src}}.jf = G'_{\text{tgt}}.jf$
$G'_{\text{src}}.mo = G_{\text{src}}.mo \cup \{(a, e) \mid G'_{\text{tgt}}.mo(a, e)\} \cup \{(a, e) \mid G'_{\text{tgt}}.po(e, a)\} \cup \{(w, e) \mid w.\text{lab} = St_{\text{NA}}(x, v') \land w \in \text{dom}([B]; G_{\text{src}}.po) \land \text{dom}(G_{\text{src}}.po; [C]) \land G_{\text{src}}.po(w, pc(\tau_x))\}$
$G'_{\text{src}}.ew = G_{\text{src}}.ew \cup \{(a, e) \mid G'_{\text{tgt}}.ew(a, e)\}$

Now we check the consistency of $G'_{\text{src}}$. We already know that $G_{\text{src}}$ and $G'_{\text{tgt}}$ is consistent. Following the construction of $G'_{\text{src}}$, the (CF), (CFJ), (VIS), (ICF), (ICFJ) constraints immediately hold. It remains to show that $G'_{\text{src}}$ satisfies (COH').

From the definition, there is no $G_{\text{src}}.hb; G_{\text{src}}.eco^2$ cycle. So any new $G'_{\text{src}}.hb; G'_{\text{src}}.eco^2$ cycle involves event $e$. However, if there is any outgoing $G_{\text{src}}.mo$ edge from $e$ then there is a write-write race and hence the source program has undefined behavior. Hence there is no $G'_{\text{src}}.hb; G'_{\text{src}}.eco^2$ cycle involving $e$. As a result, $G'_{\text{src}}$ satisfies (COH') and $G'_{\text{src}}$ is consistent.

Case $e \in G'_{\text{tgt}}.E \setminus (C \cup S \cup R \cup W)$:
We construct the $G'_{src}$ as follows:

$$G'_{src}.E = G_{src}.E \cup \{e\}$$

$$G'_{src}.po = (G_{src}.po \cup \{(a, e) \mid G'_{tgt}.po(a, e)\})^+$$

$$G'_{src}.jf = G'_{tgt}.jf \cup \{(a, e) \mid G'_{tgt}.jf(a, e)\}$$

$$G'_{src}.mo = G_{src}.mo \cup \{(a, e) \mid G'_{tgt}.mo(a, e)\}$$

$$\uplus \{(d, e) \mid d \in D \land G_{tgt}.mo(s, e) \land \exists W(G'_{src}, s, d)\}$$

$$\uplus \{(e, d) \mid d \in D \land G_{tgt}.mo(e, s) \land \exists W(G'_{src}, s, d)\}$$

$$\uplus \{(e, c) \mid c \in G'_{src}.E \setminus G'_{tgt}.E \land c.loc = e.loc \land \neg G'_{src}.cf(e, e)\}$$

$$G'_{src}.ew = G_{src}.ew \cup \{(a, e) \mid G'_{tgt}.ew(a, e)\}$$

Now we check the consistency of $G'_{src}$. We already know that $G_{src}$ and $G'_{tgt}$ is consistent. Following the construction of $G'_{src}$, the (CF), (CFJ), (VIS), (ICF), (ICFJ) constraints immediately hold. It remains to show that $G'_{src}$ satisfies (COH').

From the definition, there is no $G_{src}.hb; G_{src}.eco^2$ cycle. So any new $G'_{src}.hb; G'_{src}.eco^2$ cycle involves event $d \in D$ or the events in $G'_{src}.E \setminus G'_{tgt}.E$. However, following the definition, if there is any new $G'_{src}.hb; G'_{src}.eco^2$ cycle involving event $d$ then there is a cycle involving write event $s$ where $\exists W(G'_{src}, s, d)$. In that case there is also $G'_{tgt}.hb; G'_{tgt}.eco^2$ cycle which is a contradiction. The writes in $G'_{src}.E \setminus G'_{tgt}.E$ have no outgoing $G'_{src}.mo \setminus G'_{src}.po$ edge and hence cannot create a $G'_{src}.hb; G'_{src}.eco^2$ cycle. The reads in $G'_{src}.E \setminus G'_{tgt}.E$ may have outgoing $G'_{src}.fr$ edges. However, if any such $G'_{src}.fr$ edge creates a cycle then following the definition, there is already a $G_{src}.hb; G_{src}.eco^2$ cycle which is a contradiction. Hence $G'_{src}$ satisfies (COH') and $G'_{src}$ is consistent.

**Source Execution Construction.** Next, we construct an execution $X_I \in \text{ex}_{\text{weakestmo}}(G_{tgt})$.

If $W \subseteq (G_{tgt}.E \setminus X_I.E)$, then we find the corresponding execution $X_s \in \text{ex}_{\text{weakestmo}}(G_{src})$ such that $X_s$ contains no event created for $\text{store}_v(x, v')$. Else if an event $w \in W$ is in $X_I$, then we know that we can find an execution with $X_s.E$ and $X_s.E$ also contains an event $d \in D$ where $d.lab = \text{St}_{na}(x, v')$ Also let $r \in R \cap X_I.E$. Thus $X_s$ is as follows.

$$X_s.E = X_I.E \uplus \{d \mid X_I.E \cap W \neq \emptyset\} \cup \{r \mid r \in R \cap X_I.E\} \cup \{M(r) \mid r \in R \cap X_I.E\}$$

$$X_s.po = (X_I.po \uplus \{(b, d), (d, c) \mid (b, c) \in \text{imm}(X_I.po) \cap (B \times C) \land d \in (G_{src}.E \setminus G_{tgt}.E)\} \cup \{(p, r) \mid X_I.po(p, r) \land p \notin R \land r \in R \cap X_I.E\} \cup \{(p, M(r)) \mid X_I.po(p, r) \land p \notin R \land r \in R \cap X_I.E\}\}^+$$

$$X_s.rf = X_I.rf \cup \{(a, r) \mid r \in R\} \cup \{(w, M(r)) \mid G_{src}.rf(w, M(r)) \land r \in R \land w \in X_s.E\}$$

$$X_s.mo = X_I.mo \cup \{(d, w) \mid d \in D \land w \in \text{dom}(D) \cap G_{src}.mo \cap X_s.E\} \cup \{(w, d) \mid d \in D \land w \in \text{dom}(D) \cap X_s.E\}$$

**Source Execution Consistency.** Now we check the consistency of $X_s$.

- Following the definition of $X_s$ the (Well-formed) is satisfied.
- We know that $X_I$ follows (total-MO). The additional write $d$ introduced in $X_s$ has the label $St_{na}(x, v')$. However, from the definition of $G_{src}$ and $X_s$, event $d$ preserves (total-MO).
- Assume (Atomicity) does not hold in $X_s$. We know that (Atomicity) holds in $X_I$. Hence (Atomicity) is violated due to event $d$. In that case there exists $u \in X_s.U_x$ such that $X_s.fr(u, d)$
and $X_s.mo(d, u)$. However, in this case there is a write-write race and hence the source program has undefined behavior which is a contradiction. Hence (Atomicity) holds in $X_s$.

- Now we check if (SC) holds. As $d \notin SC$, it introduces no new $[SC]; X_s.hb_{C11}; [SC]$ path compared to $X_t$. We also know that SC holds on $X_t$. As a result, $X_s$ also preserves SC.

Thus $X_s$ is consistent and $X \in \text{ex}_{\text{WEAKESTMO}}(G_{src})$ holds.

**Same Behavior.**

For locations $y \neq x$, we have $X_s.E_y = X.E_y$ and so $\text{Behavior}(X_s)|_y = \text{Behavior}(X_t)|_y$ trivially holds. Now we check whether $\text{Behavior}(X_s)|_x = \text{Behavior}(X_t)|_x$ holds. Note that any newly introduced event $d \in X_s.E \setminus X_t.E$ is not $X_s.mo$ maximal, because in that case there exists a store $St_{xa}(x, v)$ which is $X_s.mo$ after $d$. Hence $\text{Behavior}(X_s) = \text{Behavior}(X_t)$ holds.

**Race Preservation.** Moreover, if $X_t$ is racy, then the new write $d$ does not introduce any $X_s.sw_{C11}$ edge in $X_s$. Hence $X_s$ is also racy. As a result, if the target execution has undefined behavior due to a data race, so does the source execution.

□
H PROOF OF CORRECTNESS OF SPECULATIVE LOAD

Theorem 8. The transformation $\epsilon \leadsto \text{Ld}_o(x,\_)$ is correct in \textsc{weakestmo-llvm}.

Proof. Let $R \subseteq G_{tgt}.E$ be the set of introduced events with label $\text{Ld}_o(x, v)$ in the target event structure $G_{tgt}$ such that

Let $R$ be the set of events of thread $i$ of $G_{tgt}$ with label $\text{Ld}_o(x, v)$ such that $\tau \cdot \text{Ld}_o(x, v) \cdot \tau' \notin P_{src}(i)$. Then, because of the relationship between the two programs, we know that for each such $r \in R$, $\tau \cdot \tau' \in P_{src}(i)$ holds. Let $C$ be the immediate $G_{tgt}.po$ successors of $R$ events.

Source Event Structure Construction.

To construct $G_{src}$, we follow the construction steps of $G_{tgt}$. For each target construction step that adds event $e$ to $G_{tgt}$ to get $G'_{tgt}$, we perform one or more corresponding steps going from $G_{src}$ to $G'_{src}$. We do a case analysis on the event $e$ of the target event structure.

Case $e \in R$:

In this case $G'_{src} = G_{src}$ and $G'_{src}$ is consistent as $G_{src}$ is consistent.

Case $e \in C$:

In this case we append $e$ to the event in $C$ as follows:

$$G'_{src}.E = G_{src}.E \uplus \{e\}$$

$$G'_{src}.po = (G_{src}.po \uplus \{(e, e) \mid (e, e) \in [C]; \text{imm}(G'_{tgt}.po); [R]; \text{imm}(G_{tgt}.po))\}^+$$

$$G'_{src}.jf = G_{src}.jf \uplus \{(a, e) \mid G'_{tgt}.jf(a, e)\}$$

$$G'_{src}.mo = G'_{tgt}.mo$$

$$G'_{src}.ew = G'_{tgt}.ew$$

Now we check the consistency of $G'_{src}$. We already know that $G_{src}$ and $G'_{tgt}$ is consistent. Following the construction of $G'_{src}$, the (CF), (CFJ), (VIS), (ICF), (ICFJ), (COH') constraints immediately hold.

Case $e \in G'_{tgt}.E \setminus (C \cup R)$:

Source Execution Construction. Next, we construct an execution $X_t \in \text{ex}_{\textsc{weakestmo}}(G_{tgt})$. If

If $R \subseteq (G_{tgt} \setminus X_t).E$, then we find the corresponding execution $X_s \in \text{ex}_{\textsc{weakestmo}}(G_{src})$ such that $X_s$ contains no event created for $\text{Ld}_o(x, v)$. Else if an event $r \in R$ is in $X_t$, then we know that we can find an execution with $r \notin X_s.E$. Thus $X_s$ is as follows.

$$X_s.E = X_t.E \setminus R$$

$$X_s.po = X_t.po \setminus \{(a, b) \mid a \in R \lor b \in R\}$$

$$X_s.rf = X_t.rf \setminus \{(a, b) \mid a \in R \lor b \in R\}$$

$$X_s.mo = X_t.mo$$

Source Execution Consistency. Now we check the consistency of $X_s$.

Since $X_t$ is consistent, the (Well-formed), (total-MO), (Coherence), (Atomicity), (SC) constraints also hold for $X_s$.

Same Behavior. The $R$ events are loads and hence do not affect program behavior. Hence, Behavior($X_s$) = Behavior($X_t$) holds.

Race Preservation. The $R$ events may introduce new read-write races in the target execution compared to the source execution. This is not correct in \textsc{weakestmo-C11} model, but it is fine in the \textsc{weakestmo-llvm} model. 

$\square$