A Proofs of the XMM Operational Semantics Properties

Definition 16. Let G and G' be two execution graphs, such that G is a prefix of G'. Let e be an event from $G'.E$.

A delta-graph $\Delta_e G$ is a graph, such that:

- $\Delta_e G.E \triangleq G.E \cup \{e\}$
- $\Delta_e G.lab \triangleq G.lab[e \mapsto G'.lab(e)]$
- $\Delta_e G.po \triangleq [\Delta_e G.E]; G'.po; [\Delta_e G.E]$
- $\Delta_e G.rf \triangleq [\Delta_e G.E]; G'.rf; [\Delta_e G.E]$
- $\Delta_e G.mo \triangleq [\Delta_e G.E]; G'.mo; [\Delta_e G.E]$
- $\Delta_e G.rmw \triangleq [\Delta_e G.E]; G'.rmw; [\Delta_e G.E]$

Essentially we construct it by adding e to G and all the accompanying edges from G'.

Lemma 5. *Let G and G' be two execution graphs, such that G is a prefix of G': $G'|_{G.E} = G$. Let $C \subseteq G'.E$ be a subset of committed events. Let e be an event such that:*

- $e \notin G.E$
- $e \in G'.E$
- $e \in G'.R$ *implies* $e \in \mathrm{codom}([G.E]; G'.rf) \cup C$

Finally, assume WellFormed(G, G', C) and WellFormed(G') hold. Then

$$\langle G', C\rangle \vdash G \xrightarrow{\langle e,\ell\rangle} \Delta_e G \quad \textit{where } \ell = G'.lab(e)$$

Proof. From *WellFormed*(G') it is obvious, that:

- $\Delta_e G.E = G.E \uplus \{e\}$
- $\Delta_e G.lab = G.lab[e \mapsto G'.lab(e)]$
- $\Delta_e G.po = G.po \cup \Delta_{po}(G, e)$
- $\Delta_e G.mo = G.mo \cup \Delta_{mo}(G, W_1, W_2, e)$
- $\Delta_e G.rmw = G.rmw \cup \Delta_{rmw}(G, r, e)$

The only tricky part is the delta for rf. We consider two cases:

- $e \in W$
  The following is obvious by definition of $\Delta_e G$:
  $$\Delta_e G.rf = G.rf \cup \Delta^{W}_{rf}(G, R, e)$$
- $e \in R$
  We have $e \in \mathrm{codom}([G.E]; G'.rf) \cup C$. From this we immediately get that
  $$\Delta_e G.rf = G.rf \cup \Delta^{R}_{rf}(G, w, e)$$

It is also obvious that the resulting configuration is well-formed. □

Lemma. *(Proof of Lemma 1)*

Let G and G' be two execution graphs, such that G is a prefix of G': $G'|_{G.E} = G$. Let $C \subseteq G'.E$ be a subset of committed events, $\mathcal{U} \triangleq G'.E \setminus C$ be a subset of uncommitted events, and $\Delta = G'.E \setminus G.E$ be a difference of the two event sets. Moreover, assume WellFormed(G, G', C) and WellFormed(G') hold.

Let $\ell \triangleq [e_1, \ldots, e_n]$ be an enumeration of a set Δ, such that the following conditions are met:

- $\langle e_i, e_j\rangle \in G'.po$ *implies* $i < j$;
- $\langle e_i, e_j\rangle \in G'.rf; [\mathcal{U}]$ *implies* $i < j$;
- $\forall r \in R \cap \Delta.\, \exists w \in G.W \cup \Delta.\, \langle w, r\rangle \in G'.rf$.

In other words

- $\mathrm{G}'.\mathrm{po}$ *and* $\mathrm{G}'.\mathsf{rf}; [\mathcal{U}]$ *respect the order induced by* ℓ
- G' *is* rf*-complete*

Then, the graph G' *can be constructed from* G*:*

$$\langle \mathrm{G}', C\rangle \vdash \mathrm{G} \overset{\ell}{\Rightarrow}^{*} \mathrm{G}'$$

Proof. The proof will proceed with induction by ℓ.

- Base: when ℓ is empty, Δ is empty too. Which means that $\mathrm{G}' = \mathrm{G}$. This immediately implies

$$\langle \mathrm{G}', C\rangle \vdash \mathrm{G} \overset{\ell}{\Rightarrow}^{*} \mathrm{G}'$$

- Step: we have $\ell = [e_1, e_2, \ldots, e_n]$.
  We will now show the following

$$\langle \mathrm{G}', C\rangle \vdash \mathrm{G} \xRightarrow{\langle e_1, \ell^*\rangle} \Delta_{e_1}\mathrm{G} \overset{\ell'}{\Rightarrow}^{*} \mathrm{G}' \quad \text{where } \ell^* = \mathrm{G}'.\mathsf{lab}(e_1),\ \ell' = [e_2, \ldots, e_n]$$

  From the assumptions, it is easy to show the following, which proves the first step

$$e_1 \in \mathrm{G}'.\mathrm{R} \textit{ implies } e_1 \in \mathsf{codom}([\mathrm{G}.\mathrm{E}]; \mathrm{G}'.\mathsf{rf}) \cup C$$

  The second step is proved by applying the induction hypothesis with $\mathrm{G} \triangleq \Delta_{e_1}\mathrm{G}$ and $\ell \triangleq \ell'$. All other conditions still hold, because we removed the first element of the list.

□

Corollary 1. *Given graphs* G, G'*, committed events* $C \subseteq \mathrm{G}'.\mathrm{E}$*, and a subset of determined events* $\mathcal{D}$*, such that* $\mathcal{D} \subseteq \mathsf{f} \uparrow C$ *and* $\mathsf{dom}(\mathrm{G}.\mathrm{po}; [\mathcal{D}]) \subseteq \mathcal{D}$*, re-execution can proceed from a graph restricted to* $\mathcal{D}$ *if and only if it can proceed from an initial graph:*

$$\langle \mathrm{G}', C\rangle \vdash \mathrm{G}|_{\mathcal{D}} \Rightarrow^{*} \mathrm{G}' \iff \langle \mathrm{G}', C\rangle \vdash \mathrm{G}_{\mathsf{init}} \Rightarrow^{*} \mathrm{G}'$$

Proof. Follows directly from Lemma 1 and the fact that the set of determined events is a program order closed subset of committed events — meaning that determined reads are not subject to $\mathsf{rf}; [\mathcal{U}]$ constraint and thus can be added to the graph at any point during re-execution. □

Lemma. *(Proof of Lemma 2)*
Let $\mathrm{G}, \mathrm{G}', C, \mathcal{U}, \Delta, \ell$ *be defined similarly as in Lemma 1. In particular, we have that* $\mathit{WellFormed}(\mathrm{G}, \mathrm{G}', C)$ *and* $\mathit{WellFormed}(\mathrm{G}')$ *hold. Moreover, suppose the predicate* $\mathit{ThreadOrderedUEvents}(\mathrm{G}', C, \prec_{\mathsf{tid}})$ *holds, restricting uncommitted events for a given* $\prec_{\mathsf{tid}}$ *relation.*
Then, the list of events ℓ *can be reordered and grouped by threads according to* $\prec_{\mathsf{tid}}$ *relation:*

$$\ell' \triangleq \ell_{t_{i_1}} \cdot \ldots \cdot \ell_{t_{i_n}} \quad | \quad k < m \implies t_{i_k} \prec_{\mathsf{tid}} t_{i_m}$$

Where $\ell_{t_{i_k}}$ *is a sublist of* ℓ*, which contains only the events with tid* t_i*. In other words*

$$\ell' \quad \textit{is ordered by} \quad \mathrm{G}'.\mathsf{tid} \downarrow \prec_{\mathsf{tid}} \cup\, \mathrm{G}'.\mathrm{po}$$

and the resulting list ℓ' *can still be used to re-execute the graph leading to the same result:*

$$\langle \mathrm{G}', C\rangle \vdash \mathrm{G} \overset{\ell}{\Rightarrow}^{*} \mathrm{G}' \quad \textit{if and only if} \quad \langle \mathrm{G}', C\rangle \vdash \mathrm{G} \overset{\ell'}{\Rightarrow}^{*} \mathrm{G}'.$$

Proof. With all the lemma assumptions in place, all we have to show that $\mathrm{G}'.\mathrm{po}$ and $\mathrm{G}'.\mathrm{rf}; [\mathcal{U}]$ respect the order induced by ℓ'.

The $\mathrm{G}'.\mathrm{po}$ constraint is trivial. The only part that needs clarification is $\mathrm{G}'.\mathrm{rf}; [\mathcal{U}]$. Consider the cases:

- The $\mathrm{G}'.\mathrm{rf}; [\mathcal{U}]$ edge is contained in the program order: $\mathrm{G}'.\mathrm{rf}; [\mathcal{U}] \subseteq \mathrm{G}'.\mathrm{po}$. This is trivial as our list respects the $\mathrm{G}'.\mathrm{po}$ order.
- The $\mathrm{G}'.\mathrm{rf}; [\mathcal{U}]$ edge starts in G.W (*i.e.,* determined event): $\mathrm{G}'.\mathrm{rf}; [\mathcal{U}] \subseteq [\mathrm{G.W}]; \mathrm{G}'.\mathrm{rf}; [\mathcal{U}]$. In this case the write event w from which the uncommited read event read-from does not belong to ℓ', and thus is not a subject of the constraint. In other words, because the write event is contained within the graph from which the re-execution starts (*i.e.,* it is a determined event), the read event reading from it can be added at any point during the re-execution.
- The $\mathrm{G}'.\mathrm{rf}; [\mathcal{U}]$ edge is between events from different threads, and does not start in G.W. This means the events have to be ordered by $\mathrm{G}'.\mathrm{tid} \downarrow \prec_{\mathsf{tid}}$, as dictated by the predicate $\mathit{ThreadOrderedUEvents}(\mathrm{G}', C, \prec_{\mathsf{tid}})$, making them immediately ordered by the ℓ' list order.

□

Corollary 2. *Let* $\mathrm{G}, \mathrm{G}', C, \mathcal{U}, \Delta, \ell$ *be defined similarly as in Lemma 2, and suppose that* $\Delta \subseteq \mathrm{G}'.\mathrm{E}|_t$ *for some thread* t*. Then* $\mathit{ThreadOrderedUEvents}(\mathrm{G}', C, \prec_{\mathsf{tid}})$ *holds.*

Proof. It is sufficient to notice that a choice of $\prec_{\mathsf{tid}}$ ordering that defines the thread t as a maximal element and lets all other threads to be unordered trivially satisfies the predicate $\mathit{ThreadOrderedUEvents}(\mathrm{G}', C, \prec_{\mathsf{tid}})$. □

Lemma 6. *Let* $\mathrm{G}, \mathrm{G}', C, \mathcal{U}, \Delta$ *be defined similarly as in Lemma 2. In particular,* $\mathit{WellFormed}(\mathrm{G}, \mathrm{G}', C)$ *and* $\mathit{WellFormed}(\mathrm{G}')$ *hold. Moreover, suppose the predicate* $\mathit{ThreadOrderedUEvents}(\mathrm{G}', C, \prec_{\mathsf{tid}})$ *holds. Additionally, assume the internal* $\mathrm{rf}; [\mathcal{U}]$ *respects the program order:* $\mathrm{G}'.\mathrm{rf}; [\mathcal{U}] \cap =_{\mathsf{tid}} \subseteq \mathrm{G}'.\mathrm{po}$*. Finally, assume that* G' *is* rf*-complete and* Δ *is finite.*

Then, the graph G' *can be constructed from* G*:*

$$\langle \mathrm{G}', C \rangle \vdash \mathrm{G} \Rightarrow^* \mathrm{G}'$$

Proof. To prove this lemma, all we need to do is to present a list ℓ, and apply the Lemma 1.

Let ℓ be any enumeration of Δ, that respects $\mathrm{G}'.\mathrm{po}$ and $\mathrm{G}'.\mathrm{tid} \downarrow \prec_{\mathsf{tid}}$ — we can construct such a list because $\mathit{ThreadOrderedUEvents}(\mathrm{G}', C, \prec_{\mathsf{tid}})$ holds. All we need to do is to show that $\mathrm{G}'.\mathrm{po} \cup \mathrm{G}'.\mathrm{rf}; [\mathcal{U}]$ respects the order induced by ℓ. This follows immediately from the lemma assumptions and the construction of ℓ.

□

Corollary 3. *A re-execution of a single thread is always feasible.*

That is, given $\mathrm{G}, \mathrm{G}', C, \mathcal{U}, \Delta$ *defined similarly as in Lemma 6, such that* $\Delta \subseteq \mathrm{G}'.\mathrm{E}|_t$ *for some thread* t*, and assuming internal* $\mathrm{rf}; [\mathcal{U}]$ *respects the program order, then:*

$$\langle \mathrm{G}', C \rangle \vdash \mathrm{G} \Rightarrow^* \mathrm{G}'$$

Proof. Follows directly from Lemma 6 and Corollary 2. □

B Proofs of C20 consistency properties

Definition 17. Given a relation $R \subseteq X \times X$, and two sets $\mathcal{A}, \mathcal{B} \subseteq X$, we define several auxiliary relations (for the convenience of the proofs, we let $\mathcal{A}, \mathcal{B}$ be arbitrary sets, but typically we use either singleton sets or empty sets):

- *InsertBefore*$(R, \mathcal{A}, \mathcal{B})$ — inserts $\mathcal{B}$ right before $\mathcal{A}$ in the relation R:

$$\mathit{InsertBefore}(R, \mathcal{A}, \mathcal{B}) \triangleq \mathcal{B} \times \mathcal{A} \cup \mathrm{dom}(R; [\mathcal{A}]) \times \mathcal{B} \cup \mathcal{B} \times \mathrm{codom}([\mathcal{A}]; R)$$

- *InsertAfter*$(R, \mathcal{A}, \mathcal{B})$ — inserts $\mathcal{B}$ right after $\mathcal{A}$ in the relation R:

$$\mathit{InsertAfter}(R, \mathcal{A}, \mathcal{B}) \triangleq \mathcal{A} \times \mathcal{B} \cup \mathrm{dom}(R; [\mathcal{A}]) \times \mathcal{B} \cup \mathcal{B} \times \mathrm{codom}([\mathcal{A}]; R)$$

- *Swap*$(R, \mathcal{A}, \mathcal{B})$ — swaps edges going between $\mathcal{A}$ and $\mathcal{B}$ in the relation R:

$$\mathit{Swap}(R, \mathcal{A}, \mathcal{B}) \triangleq R \setminus \mathcal{A} \times \mathcal{B} \cup \mathcal{B} \times \mathcal{A}$$

Additionally, let G be an execution graph, and let $e \in \mathrm{G.E}$ be its event. We give the following auxiliary definitions:

- *WriteOf*(G, e) — is either an event e itself, if it is a write event, or it is a write event from which e reads-from, if it is a read event (for the convenience of the proofs, we actually define *WriteOf* as a set):

$$\mathit{WriteOf}(\mathrm{G}, e) = \begin{cases} \{e\} & \text{if } e \in \mathrm{G.W} \\ \{w\} & \text{if } e \in \mathrm{G.R} \text{ and } \langle w, e\rangle \in \mathrm{G.rf} \\ \emptyset & \text{otherwise} \end{cases}$$

Proposition*. *The following statements are true:*

(1) $\mathsf{po}; \mathsf{sw} = \mathsf{rpo}; \mathsf{sw}$ *and* $\mathsf{sw}; \mathsf{po} = \mathsf{sw}; \mathsf{rpo}$
(2) $\mathsf{hb} = \mathsf{po} \cup \mathsf{rhb}$
(3) $\mathsf{hb}|_{\mathsf{loc}} = \mathsf{rhb}|_{\mathsf{loc}}$
(4) $\mathsf{hb} \cap \neq_{\mathsf{tid}} = \mathsf{rhb} \cap \neq_{\mathsf{tid}}$

Proof.

(1) **Proving** $\mathsf{po}; \mathsf{sw} = \mathsf{rpo}; \mathsf{sw}$.
By definition, sw starts with $[\mathcal{E}^{\mathrm{REL}\sqsubseteq}]$, which means that we have $\mathsf{po}; [\mathcal{E}^{\mathrm{REL}\sqsubseteq}]$ and this is an rpo-edge.

(2) **Proving** $\mathsf{sw}; \mathsf{po} = \mathsf{sw}; \mathsf{rpo}$.
By definition, sw ends with $[\mathcal{E}^{\mathrm{ACQ}\sqsubseteq}]$, which means that we have $[\mathcal{E}^{\mathrm{ACQ}\sqsubseteq}]; \mathsf{po}$ and this is an rpo-edge.

(3) **Proving** $\mathsf{hb} = \mathsf{po} \cup \mathsf{rhb}$.
Inclusion $\mathsf{po} \cup \mathsf{rhb} \subseteq \mathsf{hb}$ is trivial. To prove the other inclusion, we can rewrite hb and rhb by definition, and then do a case analysis of the path in hb.

$$(\mathsf{po} \cup \mathsf{sw})^+ \subseteq \mathsf{po} \cup (\mathsf{po}|_{\mathsf{loc}} \cup \mathsf{rpo} \cup \mathsf{sw})^+$$

We can use the following facts to simplify or replace the path:
- po is transitive;
- any po-edge followed or preceded by sw-edge can be replaced by rpo-edge, as was shown above.

So if the path contains po-edges only, it is contained in po and if it contains at least one sw-edge, all the continuous po-paths can firstly be collapsed into single po-edges and then replaced by rpo-edges, which is contained in $(\mathsf{po}|_{\mathsf{loc}} \cup \mathsf{rpo} \cup \mathsf{sw})^+$.

(4) **Proving** $\mathsf{hb}|_{\mathsf{loc}} = \mathsf{rhb}|_{\mathsf{loc}}$.
Applying the previous property, we need to show that $(\mathsf{po} \cup \mathsf{rhb})|_{\mathsf{loc}} \subseteq \mathsf{rhb}|_{\mathsf{loc}}$. Hence, we need to show that $\mathsf{po}|_{\mathsf{loc}} \subseteq \mathsf{rhb}|_{\mathsf{loc}}$. By definition, $\mathsf{rhb} = (\mathsf{po}|_{\mathsf{loc}} \cup \mathsf{rpo} \cup \mathsf{sw})^+$, and by definition, rpo contains $\mathsf{po}|_{\mathsf{loc}}$, which finalizes the proof.

(5) Applying the property (2) to the relation, we obtain the result.

□

Lemma. C20 *coherence is equivalent to the following statement:*

$$\mathsf{rhb};\mathsf{eco}^{?} \textit{ is irreflexive}$$

Proof. Since $\mathsf{eco}^{?}$ connects events of the same location, then hb is also restricted by the same location. Thus, applying Proposition 1 finalizes the proof. □

Definition 18. Memory consistency model $\mathcal{M}$ is called *monotone* if for every execution graphs G′ and G, an event mapping function $\mathsf{m} : \mathsf{G}'.\mathsf{E} \to \mathsf{G}.\mathsf{E}$ consistency of G′ implies consistency of G:

$$\mathit{Consistent}(\mathcal{M}, \mathsf{G}') \implies \mathit{Consistent}(\mathcal{M}, \mathsf{G})$$

provided that the following conditions are met:

- m is injective
- $\mathsf{G}.\mathsf{tid} \circ \mathsf{m} = \mathsf{G}'.\mathsf{tid}$
- $\mathsf{G}.\mathsf{lab} \circ \mathsf{m} = \mathsf{G}'.\mathsf{lab}$
- $\mathsf{G}.\mathsf{E} \subseteq \mathsf{m} \uparrow \mathsf{G}'.\mathsf{E}$
- $\mathsf{G}.\mathsf{rpo} \subseteq \mathsf{m} \uparrow \mathsf{G}'.\mathsf{rpo}$
- $\mathsf{G}.\mathsf{po}|_{\mathsf{loc}} \subseteq \mathsf{m} \uparrow \mathsf{G}'.\mathsf{po}|_{\mathsf{loc}}$
- $\mathsf{G}.\mathsf{rf} \subseteq \mathsf{m} \uparrow \mathsf{G}'.\mathsf{rf}$
- $\mathsf{G}.\mathsf{mo} \subseteq \mathsf{m} \uparrow \mathsf{G}'.\mathsf{mo}$
- $\mathsf{G}.\mathsf{rmw} \subseteq \mathsf{m} \uparrow \mathsf{G}'.\mathsf{rmw}$
- $\mathsf{G}.\mathsf{rhb} \subseteq \mathsf{m} \uparrow \mathsf{G}'.\mathsf{rhb}$

Lemma 7. C20 *is* monotone.

Proof.

(Coherence)

We need to prove that $\mathsf{G}.\mathsf{rhb};\mathsf{G}.\mathsf{eco}^{?}$ is irreflexive. Since G.rhb is irreflexive by inclusion into the $\mathsf{m} \uparrow \mathsf{G}.\mathsf{rhb}$, it is sufficient to prove that $\mathsf{G}.\mathsf{rhb};\mathsf{G}.\mathsf{eco}$ is irreflexive. Easy to see that $\mathsf{G}.\mathsf{fr} \subseteq \mathsf{m} \uparrow \mathsf{G}'.\mathsf{fr}$ and hence $\mathsf{G}.\mathsf{eco} \subseteq \mathsf{m} \uparrow \mathsf{G}'.\mathsf{eco}$. Therefore, $\mathsf{G}.\mathsf{rhb};\mathsf{G}.\mathsf{eco} \subseteq \mathsf{m} \uparrow (\mathsf{G}'.\mathsf{rhb};\mathsf{G}'.\mathsf{eco})$, which is irreflexive by the Coherence property of G′.

(Atomicity)

We need to prove that $\mathsf{G}.\mathsf{rmw} \cap (\mathsf{G}.\mathsf{fr};\mathsf{G}.\mathsf{mo}) \equiv \emptyset$. As already obtained, $\mathsf{G}.\mathsf{fr} \subseteq \mathsf{m} \uparrow \mathsf{G}'.\mathsf{fr}$, hence

$$\mathsf{G}.\mathsf{rmw} \cap (\mathsf{G}.\mathsf{fr};\mathsf{G}.\mathsf{mo}) \subseteq \mathsf{m} \uparrow (\mathsf{G}'.\mathsf{rmw} \cap \mathsf{G}'.\mathsf{fr};\mathsf{G}'.\mathsf{mo}) \equiv \emptyset$$

□

Definition 19. Memory consistency model $\mathcal{M}$ is called *maximal read extensible* if for every execution graphs G′ and G, an event mapping function $\mathsf{m} : \mathsf{G}'.\mathsf{E} \to \mathsf{G}.\mathsf{E}$, and a read event $r \in \mathsf{G}.\mathsf{E}$ consistency of G′ implies consistency of G:

$$\mathit{Consistent}(\mathcal{M}, \mathsf{G}') \implies \mathit{Consistent}(\mathcal{M}, \mathsf{G})$$

provided that the following conditions are met (where $R \triangleq \{r\}$):

- m is injective
- $\mathsf{G}.\mathsf{tid} \circ \mathsf{m} = \mathsf{G}'.\mathsf{tid}$
- $\mathsf{G}.\mathsf{lab} \circ \mathsf{m} =_{|\mathsf{G}'.\mathsf{E} \setminus \mathsf{m}\downarrow R} \mathsf{G}'.\mathsf{lab}$
- $\mathsf{G}.\mathsf{E} = \mathsf{m} \uparrow \mathsf{G}'.\mathsf{E} \cup R$
- $\mathsf{codom}([R];\mathsf{G}.\mathsf{rpo}) = \emptyset$
- $\mathsf{G}.\mathsf{rpo};[\mathsf{G}.\mathsf{E} \setminus R] \subseteq \mathsf{m} \uparrow \mathsf{G}'.\mathsf{rpo}$

- $\mathrm{codom}([R]; \mathsf{G.po}|_{\mathsf{loc}}) = \emptyset$
- $\mathsf{G.po}|_{\mathsf{loc}}; [\mathsf{G.E} \setminus R] \subseteq \mathsf{m} \uparrow \mathsf{G}'.\mathsf{po}|_{\mathsf{loc}}$
- $\mathsf{G.rf} = \mathsf{m} \uparrow \mathsf{G}'.\mathsf{rf} \cup \mathsf{G.srf}; [R]$
- $\mathsf{G.mo} = \mathsf{m} \uparrow \mathsf{G}'.\mathsf{mo}$
- $\mathsf{G.rmw} \subseteq \mathsf{m} \uparrow \mathsf{G}'.\mathsf{rmw}$

Lemma 8. C20 *is* maximal read extensible.

Proof. (Coherence)

We need to prove that $\mathsf{G.hb}; \mathsf{G.eco}^?$ is irreflexive. Let us first prove that $\mathsf{G.hb}$ is irreflexive. By Proposition 1, we know that $\mathsf{G.hb} = \mathsf{G.po} \cup \mathsf{G.rhb}$. Since po is irreflexive, it is left to show that $\mathsf{G.rhb}$ is irreflexive. There are two possible cases: it can be either $\mathsf{rhb}; [R]$ cycle or not. If not, then it is irreflexive by the Coherence property of G'. If it is, then we can use the fact that $\mathrm{codom}([R]; \mathsf{G.rhb}) = \emptyset$. Let us prove it as well.

It can be rewritten in the following way:

$$\begin{aligned}&\mathrm{codom}(([R]; \mathsf{G.rpo}); (\mathsf{G.po}|_{\mathsf{loc}} \cup \mathsf{G.rpo} \cup \mathsf{G.sw})^*)\\ \cup\ &\mathrm{codom}(([R]; \mathsf{G.sw}); (\mathsf{G.po}|_{\mathsf{loc}} \cup \mathsf{G.rpo} \cup \mathsf{G.sw})^*)\\ \cup\ &\mathrm{codom}(([R]; \mathsf{G.po}|_{\mathsf{loc}}); (\mathsf{G.po}|_{\mathsf{loc}} \cup \mathsf{G.rpo} \cup \mathsf{G.sw})^*) = \emptyset\end{aligned}$$

The first and thirds summand are empty, since $\mathrm{codom}([R]; \mathsf{G.rpo}) = \emptyset$ and $\mathrm{codom}([R]; \mathsf{G.po}|_{\mathsf{loc}}) = \emptyset$. And the second is empty, since sw starts with a write event or fence, and R is a read event.

So, since $\mathsf{G.hb}$ is irreflexive by construction, it is sufficient to prove that $\mathsf{G.hb}; \mathsf{G.eco}$ is irreflexive. The latter is equivalent to $\mathsf{G.rhb}; \mathsf{G.eco}$ being irreflexive.

We know that $\mathsf{G}'.\mathsf{hb}; \mathsf{G}'.\mathsf{eco}$ is irreflexive. It is also equivalent to $\mathsf{G}'.\mathsf{rhb}; \mathsf{G}'.\mathsf{eco}$ being irreflexive.

We also know that:

$$\begin{aligned}\mathsf{G.fr} &= \mathsf{G.rf}^{-1}; \mathsf{G.mo} \subseteq \mathsf{m} \uparrow \mathsf{G}'.\mathsf{fr} \cup [R]; \mathsf{G.srf}^{-1}; \mathsf{G.mo}\\ \mathsf{G.eco} &= \mathsf{G.rf} \cup \mathsf{G.mo}; \mathsf{G.rf}^? \cup \mathsf{G.fr}; \mathsf{G.rf}^?\\ &\subseteq \mathsf{m} \uparrow \mathsf{G}'.\mathsf{eco}\\ &\cup \mathsf{G.srf}; [R]\\ &\cup \mathsf{G.mo}; \mathsf{G.srf}; [R]\\ &\cup \mathsf{G.fr}; \mathsf{G.srf}; [R]\\ &\cup [R]; \mathsf{G.srf}^{-1}; \mathsf{G.mo}; \mathsf{G.rf}^?\\ \mathsf{G.rhb}; [\mathsf{G.E} \setminus R] &\subseteq \mathsf{m} \uparrow \mathsf{G}'.\mathsf{rhb}\\ \mathsf{G.rhb}; \mathsf{G.eco} &\subseteq \mathsf{m} \uparrow (\mathsf{G}'.\mathsf{rhb}; \mathsf{G}'.\mathsf{eco})\\ &\cup \mathsf{G.rhb}; \mathsf{G.srf}; [R]\\ &\cup \mathsf{G.rhb}; \mathsf{G.mo}; \mathsf{G.srf}; [R]\\ &\cup \mathsf{G.rhb}; \mathsf{G.fr}; \mathsf{G.srf}; [R]\\ &\cup \mathsf{G.rhb}; [R]; \mathsf{G.srf}^{-1}; \mathsf{G.mo}; \mathsf{G.rf}^?\end{aligned}$$

So it remains to show that the following relations are irreflexive:

- $\mathsf{G.rhb}; \mathsf{G.srf}; [R]$
- $\mathsf{G.rhb}; \mathsf{G.mo}; \mathsf{G.srf}; [R]$
- $\mathsf{G.rhb}; \mathsf{G.fr}; \mathsf{G.srf}; [R]$
- $\mathsf{G.rhb}; [R]; \mathsf{G.srf}^{-1}; \mathsf{G.mo}; \mathsf{G.rf}^?$

The first three irreflexivity proofs can be obtained from the fact that $\mathrm{codom}([R]; \mathrm{G.rhb}) = \emptyset$. For the last one, firstly, we can rotate the relation in the following way: $\mathrm{G.mo}; \mathrm{G.rf}^?; \mathrm{G.rhb}; [R]; \mathrm{G.srf}^{-1}$, since, we are aiming to proof irreflexivity. Then, we can rewrite it as

$$\mathrm{G.mo}; [W]; \mathrm{G.rf}^?; \mathrm{G.rhb}; [R]; \mathrm{G.srf}^{-1},$$

since mo-edge ends with write event. And now we can rewrite $[W]; \mathrm{G.rf}^?; \mathrm{G.rhb}^? \subseteq \mathtt{vf}$ and obtain $\mathrm{G.mo}; \mathrm{G.vf}; (\mathrm{G.po}|_{\mathrm{loc}} \cup \mathrm{G.rpo} \cup \mathrm{G.sw}); [R]; \mathrm{G.srf}^{-1}$. Since $\mathrm{codom}([R]; \mathrm{G.rpo}) = \emptyset$ and $\mathrm{codom}([R]; \mathrm{G.po}|_{\mathrm{loc}}) = \emptyset$, we only need to prove that $\mathrm{G.mo}; \mathrm{G.vf}; \mathrm{G.sw}; [R]; \mathrm{G.srf}^{-1}$ is empty. That would be true following from $\mathrm{G.sw} \subseteq \mathsf{m} \uparrow \mathrm{G}'.\mathrm{sw}$. Let us show that.

By definition,

$$\mathrm{sw} = [\mathcal{E}^{\mathrm{REL}\sqsubseteq}]; ([\mathrm{F}]; \mathrm{po})^?; [\mathrm{W}^{\mathrm{RLX}\sqsubseteq}]; (\mathrm{rf}; \mathrm{rmw})^*; \mathrm{rf}; [\mathrm{R}^{\mathrm{RLX}\sqsubseteq}]; (\mathrm{po}; [\mathrm{F}])^?; [\mathcal{E}^{\mathrm{ACQ}\sqsubseteq}]$$

As rmw-edges are preserved by m, they can not start with R-event, meaning that the only possibility of where R can exist in this construction is $[\mathrm{R}^{\mathrm{RLX}\sqsubseteq}]$ or in the last position. If there is no po at the end of the construction, then we obtain that R should be an acquire event, which is not possible. And if there is po in the end, then it can be replaced by rpo, leading to a contradiction with the fact that $\mathrm{codom}([R]; \mathrm{G.rpo}) = \emptyset$.

Now it is remaining to show that $\mathrm{G.rhb}; [\mathrm{G.E} \setminus R] \subseteq \mathsf{m} \uparrow \mathrm{G}'.\mathrm{rhb}$. Let us show that by induction. Firstly, we can easily obtain that

$$\mathrm{G.rhb} = (\mathrm{G.po}|_{\mathrm{loc}} \cup \mathrm{G.rpo} \cup \mathrm{G.sw}); [\mathrm{G.E} \setminus R] \subseteq \mathsf{m} \uparrow (\mathrm{G}'.\mathrm{po}|_{\mathrm{loc}} \cup \mathrm{G}'.\mathrm{rpo} \cup \mathrm{G}'.\mathrm{sw})$$

from the property on codomain and sw. From that, we can easily prove

$$\mathsf{m} \uparrow (\mathrm{G}'.\mathrm{po}|_{\mathrm{loc}} \cup \mathrm{G}'.\mathrm{rpo} \cup \mathrm{G}'.\mathrm{sw})^+; (\mathrm{G.po}|_{\mathrm{loc}} \cup \mathrm{G.rpo} \cup \mathrm{G.sw}); [\mathrm{G.E} \setminus R] \subseteq \mathsf{m} \uparrow (\mathrm{G}'.\mathrm{po}|_{\mathrm{loc}} \cup \mathrm{G}'.\mathrm{rpo} \cup \mathrm{G}'.\mathrm{sw})^+$$

which will lead us to

$$((\mathrm{G.po}|_{\mathrm{loc}} \cup \mathrm{G.rpo} \cup \mathrm{G.sw}); [\mathrm{G.E} \setminus R])^+ \subseteq \mathsf{m} \uparrow (\mathrm{G}'.\mathrm{po}|_{\mathrm{loc}} \cup \mathrm{G}'.\mathrm{rpo} \cup \mathrm{G}'.\mathrm{sw})^+$$

and the left part essentially equals to $(\mathrm{G.po}|_{\mathrm{loc}} \cup \mathrm{G.rpo} \cup \mathrm{G.sw})^+; [\mathrm{G.E} \setminus R]$, since none of the edges from $\mathrm{G.po}|_{\mathrm{loc}} \cup \mathrm{G.rpo} \cup \mathrm{G.sw}$ can not start in R and end in any other event, which finalizes the proof.

(Atomicity)

We need to prove that $\mathrm{G.rmw} \cap (\mathrm{G.fr}; \mathrm{G.mo}) \equiv \emptyset$. Using $\mathrm{G.fr} \subseteq \mathsf{m} \uparrow \mathrm{G}'.\mathrm{fr} \cup [R]; \mathrm{G.srf}^{-1}; \mathrm{G.mo}$, we can rewrite the intersection as

$$\mathsf{m} \uparrow \mathrm{G}'.\mathrm{rmw} \cap (\mathsf{m} \uparrow (\mathrm{G}'.\mathrm{fr}; \mathrm{G}'.\mathrm{mo}) \cup [R]; \mathrm{G.srf}^{-1}; \mathrm{G.mo}; \mathrm{G.mo}),$$

where we can remove $[R]; \mathrm{G.srf}^{-1}; \mathrm{G.mo}; \mathrm{G.mo}$, since $\mathsf{m} \uparrow \mathrm{G}'.\mathrm{rmw}$ can not begin with new event R. The latter is empty by the Atomicity property of G'.

□

Definition 20. Memory consistency model $\mathcal{M}$ is called *maximal write extensible* if for every execution graphs G' and G, an event mapping function $\mathsf{m} : \mathrm{G}'.\mathrm{E} \to \mathrm{G.E}$, and a write event $w \in \mathrm{G.E}$ consistency of G' implies consistency of G:

$$\mathit{Consistent}(\mathcal{M}, \mathrm{G}') \implies \mathit{Consistent}(\mathcal{M}, \mathrm{G})$$

provided that the following conditions are met (where $W \triangleq \{w\}$):

- m is injective
- $\mathrm{G.tid} \circ \mathsf{m} = \mathrm{G}'.\mathrm{tid}$

- $G.\mathsf{lab} \circ \mathsf{m} =_{|G'.E \setminus \mathsf{m}\downarrow W} G'.\mathsf{lab}$
- $G.E = \mathsf{m} \uparrow G'.E \cup W$
- $\mathsf{codom}([W]; G.\mathsf{rpo}) = \emptyset$
- $G.\mathsf{rpo}; [G.E \setminus W] \subseteq \mathsf{m} \uparrow G'.\mathsf{rpo}$
- $\mathsf{codom}([W]; G.\mathsf{po}|_{\mathsf{loc}}) = \emptyset$
- $G.\mathsf{po}|_{\mathsf{loc}}; [G.E \setminus W] \subseteq \mathsf{m} \uparrow G'.\mathsf{po}|_{\mathsf{loc}}$
- $G.\mathsf{rf} = \mathsf{m} \uparrow G'.\mathsf{rf}$
- $G.\mathsf{mo} = \mathsf{m} \uparrow G'.\mathsf{mo} \cup (G.\mathsf{W}_{\mathsf{loc}(w)} \setminus W) \times W$
- $G.\mathsf{rmw} \subseteq \mathsf{m} \uparrow G'.\mathsf{rmw}$

Lemma 9. C20 *is* maximal write extensible.

Proof. (Coherence)

We need to prove that $G.\mathsf{hb}; G.\mathsf{eco}^?$ is irreflexive. We will prove the irreflexivity of $G.\mathsf{hb}$ later, so now it is sufficient to prove that $G.\mathsf{hb}; G.\mathsf{eco}$ is irreflexive. The latter is equivalent to $G.\mathsf{rhb}; G.\mathsf{eco}$ being irreflexive.

We know that $G'.\mathsf{hb}; G'.\mathsf{eco}$ is irreflexive. It is also equivalent to $G'.\mathsf{rhb}; G'.\mathsf{eco}$ being irreflexive.

We also know that:

$$
\begin{aligned}
G.\mathsf{fr} &= G.\mathsf{rf}^{-1}; G.\mathsf{mo} \\
&= \mathsf{m} \uparrow G'.\mathsf{fr} \cup \mathsf{m} \uparrow G'.\mathsf{rf}^{-1}; ((G.\mathsf{W}_{\mathsf{loc}(w)} \setminus W) \times W) \\
G.\mathsf{eco} &= G.\mathsf{rf} \cup G.\mathsf{mo}; G.\mathsf{rf}^? \cup G.\mathsf{fr}; G.\mathsf{rf}^? \\
&= \mathsf{m} \uparrow G'.\mathsf{rf} \\
&\cup \mathsf{m} \uparrow (G'.\mathsf{mo}; G'.\mathsf{rf}^?) \\
&\cup \mathsf{m} \uparrow (G'.\mathsf{fr}; G'.\mathsf{rf}^?) \\
&\cup ((G.\mathsf{W}_{\mathsf{loc}(w)} \setminus W) \times W); (\mathsf{m} \uparrow G'.\mathsf{rf}^?) \\
&\cup (\mathsf{m} \uparrow G'.\mathsf{rf}^{-1}); ((G.\mathsf{W}_{\mathsf{loc}(w)} \setminus W) \times W); (\mathsf{m} \uparrow G'.\mathsf{rf}^?) \\
G.\mathsf{rhb}; [G.E \setminus W] &\subseteq \mathsf{m} \uparrow G'.\mathsf{rhb} \\
G.\mathsf{rhb}; G.\mathsf{eco} &\subseteq \mathsf{m} \uparrow (G'.\mathsf{rhb}; G'.\mathsf{eco}) \\
&\cup G.\mathsf{rhb}; ((G.\mathsf{W}_{\mathsf{loc}(w)} \setminus W) \times W); G.\mathsf{rf}^? \\
&\cup G.\mathsf{rhb}; G.\mathsf{rf}^{-1}; ((G.\mathsf{W}_{\mathsf{loc}(w)} \setminus W) \times W); G.\mathsf{rf}^?
\end{aligned}
$$

Irreflexivity of the first summand follows from the irreflexivity of $G'.\mathsf{rhb}; G'.\mathsf{eco}$. Then, in the last two summands, we can reduce rf-edge in the end, since no new rf-edges were added and they can not start with W. Therefore, we can rewrite the last two summands in the following way: $[W]; G.\mathsf{rhb}; \ldots$, which is empty, since $\mathsf{codom}([W]; G.\mathsf{rpo}) = \emptyset$ and $G.\mathsf{sw} \subseteq \mathsf{m} \uparrow G'\mathsf{sw}$, meaning that $G.\mathsf{rhb}$ can not begin with W.

Let us prove that $G.\mathsf{sw} \subseteq \mathsf{m} \uparrow G'.\mathsf{sw}$. By definition,

$$\mathsf{sw} = [\mathcal{E}^{\mathrm{REL}\sqsubseteq}]; ([\mathsf{F}]; \mathsf{po})^?; [\mathsf{W}^{\mathrm{RLX}\sqsubseteq}]; (\mathsf{rf}; \mathsf{rmw})^*; \mathsf{rf}; [\mathsf{R}^{\mathrm{RLX}\sqsubseteq}]; (\mathsf{po}; [\mathsf{F}])^?; [\mathcal{E}^{\mathrm{ACQ}\sqsubseteq}]$$

We should show that any $G.\mathsf{sw}$ edge has m-preimage in G'. Since rf and rmw-edges are preserved by m, they can not start or end with W, meaning that all the events in the $G.\mathsf{sw}$-construction have m-preimage in G'. Also, if there is a po-edge in the $G.\mathsf{sw}$-construction, it can be replaced by rpo-edge, following from the definition of rpo. Hence, all the edges in the construction have

m-preimage in G′ and the correspondent sw-preimage exists in G′. Moreover, in the following proof, we will need a weaker condition: $\mathrm{G.sw} \subseteq \mathsf{m} \uparrow \mathrm{G'.sw} \cup \mathrm{G.sw}; [W]$.

Now we can also prove irreflexivity of G.hb. By Proposition 1, we know that $\mathrm{G.hb} = \mathrm{G.po} \cup \mathrm{G.rhb}$. Since po is irreflexive, it is left to show that G.rhb is irreflexive. There are two possible cases: it can be either $\mathrm{rhb}; [W]$ cycle or not. If not, then it is irreflexive by the Coherence property of G′. If it is, then we can use the fact that $\mathrm{codom}([W]; \mathrm{G.rhb}) = \emptyset$. Let us prove it as well.

It can be rewritten in the following way:

$$\begin{aligned}\mathrm{codom}(([W]; \mathrm{G.rpo}); ((\mathrm{G.po}|_{\mathrm{loc}} \cup \mathrm{G.rpo} \cup \mathrm{G.sw}))^*) \cup \\ \mathrm{codom}(([W]; \mathrm{G.po}|_{\mathrm{loc}}); ((\mathrm{G.po}|_{\mathrm{loc}} \cup \mathrm{G.rpo} \cup \mathrm{G.sw}))^*) \cup \\ \mathrm{codom}(([W]; \mathrm{G.sw}); ((\mathrm{G.po}|_{\mathrm{loc}} \cup \mathrm{G.rpo} \cup \mathrm{G.sw}))^*) = \emptyset\end{aligned}$$

The first two summands are empty from the codomain conditions. And the second is empty, since $\mathsf{m} \uparrow \mathrm{G'.sw}$ can not start with W and $\mathrm{codom}([W]; \mathrm{G.sw}) = \emptyset$. The latter is easily obtained from the definition of sw.

Now it is remaining to show that $\mathrm{G.rhb}; [\mathrm{G.E} \setminus W] \subseteq \mathsf{m} \uparrow \mathrm{G'.rhb}$. Let us show that by induction. Firstly, we can easily obtain that

$$\mathrm{G.rhb} = (\mathrm{G.po}|_{\mathrm{loc}} \cup \mathrm{G.rpo} \cup \mathrm{G.sw}); [\mathrm{G.E} \setminus W] \subseteq \mathsf{m} \uparrow (\mathrm{G'.po}|_{\mathrm{loc}} \cup \mathrm{G'.rpo} \cup \mathrm{G'.sw})$$

from the property on codomain and sw. From that, we can easily prove

$$\begin{aligned}\mathsf{m} \uparrow (\mathrm{G'.po}|_{\mathrm{loc}} \cup \mathrm{G'.rpo} \cup \mathrm{G'.sw})^+; (\mathrm{G.po}|_{\mathrm{loc}} \cup \mathrm{G.rpo} \cup \mathrm{G.sw}); [\mathrm{G.E} \setminus W] \\ \subseteq \mathsf{m} \uparrow (\mathrm{G'.po}|_{\mathrm{loc}} \cup \mathrm{G'.rpo} \cup \mathrm{G'.sw})^+\end{aligned}$$

which will lead us to

$$((\mathrm{G.po}|_{\mathrm{loc}} \cup \mathrm{G.rpo} \cup \mathrm{G.sw}); [\mathrm{G.E} \setminus W])^+ \subseteq \mathsf{m} \uparrow (\mathrm{G'.po}|_{\mathrm{loc}} \cup \mathrm{G'.rpo} \cup \mathrm{G'.sw})^+$$

and the left part essentially equals to $(\mathrm{G.po}|_{\mathrm{loc}} \cup \mathrm{G.rpo} \cup \mathrm{G.sw})^+; [\mathrm{G.E} \setminus W]$, since none of the edges from $\mathrm{G.po}|_{\mathrm{loc}} \cup \mathrm{G.rpo} \cup \mathrm{G.sw}$ can not start in W and end in any other event, which finalizes the proof.

(Atomicity)
We need to prove that $\mathrm{G.rmw} \cap (\mathrm{G.fr}; \mathrm{G.mo}) \equiv \emptyset$. Using

$$\mathrm{fr} \subseteq \mathsf{m} \uparrow \mathrm{G'.fr} \cup \mathsf{m} \uparrow \mathrm{G'.rf}^{-1}; ((\mathrm{G.W}_{\mathrm{loc}(w)} \setminus W) \times W),$$

we can rewrite the intersection as

$$\begin{aligned}\mathsf{m} \uparrow \mathrm{G'.rmw} \cap (\mathsf{m} \uparrow (\mathrm{G'.fr}; \mathrm{G'.mo}) \cup (\mathsf{m} \uparrow \mathrm{G'.fr}); ((\mathrm{G.W}_{\mathrm{loc}(w)} \setminus W) \times W) \\ \cup (\mathsf{m} \uparrow \mathrm{G'.rf}^{-1}); ((\mathrm{G.W}_{\mathrm{loc}(w)} \setminus W) \times W); (\mathsf{m} \uparrow \mathrm{G'.mo}) \\ \cup (\mathsf{m} \uparrow \mathrm{G'.rf}^{-1}); ((\mathrm{G.W}_{\mathrm{loc}(w)} \setminus W) \times W); ((\mathrm{G.W}_{\mathrm{loc}(w)} \setminus W) \times W))\end{aligned}$$

The last three summands are empty as $\mathsf{m} \uparrow \mathrm{G'.rmw}$ can not end with W or $(\mathsf{m} \uparrow \mathrm{G'.mo})$ can not start with W. The remaining intersection is empty by the Atomicity property of G′.

□

Definition 21. Memory consistency model $\mathcal{M}$ is called *overwrite extensible* if for every execution graphs G′ and G, an event mapping function $\mathsf{m} : \mathrm{G'.E} \to \mathrm{G.E}$, a write event $w' \in \mathrm{G'}$, and a write event $w \in \mathrm{G.E}$, consistency of G′ implies consistency of G:

$$\mathit{Consistent}(\mathcal{M}, \mathrm{G'}) \implies \mathit{Consistent}(\mathcal{M}, \mathrm{G})$$

provided that the following conditions are met (where $W' \triangleq \{w'\}$ and $W \triangleq \{w\}$):

- m is injective

- $\mathsf{G.tid} \circ \mathsf{m} = \mathsf{G}'.\mathsf{tid}$
- $\mathsf{G.lab} \circ \mathsf{m} =_{|\mathsf{G}'.E \setminus \mathsf{m}\downarrow W}\ \mathsf{G}'.\mathsf{lab}$
- $\mathsf{G.E} = \mathsf{m} \uparrow \mathsf{G}'.\mathsf{E} \cup W$
- $\mathsf{G.po} = \mathsf{m} \uparrow \mathsf{G}'.\mathsf{po} \cup \mathit{InsertBefore}(\mathsf{m} \uparrow \mathsf{G}'.\mathsf{po}, \mathsf{m} \uparrow W', w)$
- $\mathsf{G.rf} = \mathsf{m} \uparrow \mathsf{G}'.\mathsf{rf}$
- $\mathsf{G.mo} = \mathsf{m} \uparrow \mathsf{G}'.\mathsf{mo} \cup \mathit{InsertBefore}(\mathsf{m} \uparrow \mathsf{G}'.\mathsf{mo}, \mathsf{m} \uparrow W', w)$
- $\mathsf{G.rmw} = \mathsf{m} \uparrow \mathsf{G}'.\mathsf{rmw}$

Lemma 10. C20 *is* overwrite extensible.

Proof. (Coherence)

We need to prove that $\mathsf{G.hb}; \mathsf{G.eco}^?$ is irreflexive. We will prove that $\mathsf{G.hb}$ is irreflexive later, so now it is sufficient to prove that $\mathsf{G.hb}; \mathsf{G.eco}$ is irreflexive.

We know that

$$\begin{aligned}
\mathsf{G.fr} &= \mathsf{G.rf}^{-1}; \mathsf{G.mo} \\
&\subseteq \mathsf{m} \uparrow \mathsf{G}'.\mathsf{fr} \cup \mathsf{G.rf}^{-1}; \mathit{InsertBefore}(\mathsf{m} \uparrow \mathsf{G}'.\mathsf{mo}, \mathsf{m} \uparrow W', w) \\
\mathsf{G.eco} &= \mathsf{G.rf} \cup \mathsf{G.mo}; \mathsf{G.rf}^? \cup \mathsf{G.fr}; \mathsf{G.rf}^? \\
&\subseteq \mathsf{m} \uparrow \mathsf{G}'.\mathsf{rf} \\
&\cup \mathsf{m} \uparrow (\mathsf{G}'.\mathsf{mo}; \mathsf{G}'.\mathsf{rf}^?) \\
&\cup \mathsf{m} \uparrow (\mathsf{G}'.\mathsf{fr}; \mathsf{G}'.\mathsf{rf}^?) \\
&\cup \mathit{InsertBefore}(\mathsf{m} \uparrow \mathsf{G}'.\mathsf{mo}, \mathsf{m} \uparrow W', w); \mathsf{G.rf}^? \\
&\cup \mathsf{G.rf}^{-1}; \mathit{InsertBefore}(\mathsf{m} \uparrow \mathsf{G}'.\mathsf{mo}, \mathsf{m} \uparrow W', w); \mathsf{G.rf}^?
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\mathsf{G.hb}; \mathsf{G.eco} \subseteq\ & \mathsf{G.hb}; \mathsf{m} \uparrow \mathsf{G}'.\mathsf{eco} \\
& \cup \mathsf{G.hb}; \mathit{InsertBefore}(\mathsf{m} \uparrow \mathsf{G}'.\mathsf{mo}, \mathsf{m} \uparrow W', w); \mathsf{G.rf}^? \\
& \cup \mathsf{G.hb}; \mathsf{G.rf}^{-1}; \mathit{InsertBefore}(\mathsf{m} \uparrow \mathsf{G}'.\mathsf{mo}, \mathsf{m} \uparrow W', w); \mathsf{G.rf}^?
\end{aligned}$$

To operate with hb-edges, we should notice that $\mathsf{G.hb} = \mathsf{m} \uparrow \mathsf{G}'.\mathsf{hb} \cup [w]; \mathsf{G.hb} \cup \mathsf{G.hb}; [w]$. For this, let us first prove that $\mathsf{G.sw} = \mathsf{m} \uparrow \mathsf{G}'.\mathsf{sw}$. So, we should show that all the $\mathsf{G}'.\mathsf{sw}$-edges are mapped to G.sw-edges and no new G.sw-edges are added. By definition,

$$\mathsf{sw} = [\mathcal{E}^{\mathrm{REL}\sqsubseteq}]; ([\mathsf{F}]; \mathsf{po})^?; [\mathsf{W}^{\mathrm{RLX}\sqsubseteq}]; (\mathsf{rf}; \mathsf{rmw})^*; \mathsf{rf}; [\mathsf{R}^{\mathrm{RLX}\sqsubseteq}]; (\mathsf{po}; [\mathsf{F}])^?; [\mathcal{E}^{\mathrm{ACQ}\sqsubseteq}]$$

Hence, if we have an sw-edge, it consists of po-edges, rf-edges, and rmw-edges. It is easy to see, that if there was an $\mathsf{G}'.\mathsf{sw}$-edge, then it would remain an G.sw-edge as all of its parts are mapped into the corresponding classes of edges. Now, let's consider the reverse direction. rf and rmw-edges are preserved by m, and G.po can appear if it is an image of $\mathsf{G}'.\mathsf{po}$ or if it is an edge between w and some other event a. The first case is fine and the second case is impossible to appear in sw, since that G.po-edge can be only in the first position of the sw-definition, and the following rf-edge can not start with a new event.

Now, let us proceed to $\mathsf{G.hb} = \mathsf{m} \uparrow \mathsf{G}'.\mathsf{hb} \cup [w]; \mathsf{G.hb} \cup \mathsf{G.hb}; [w]$. By definition, $\mathsf{G.hb} = (\mathsf{G.po} \cup \mathsf{G.sw})^+$, and in that path G.sw-edges connect only events from the target graph, and sequences of continuous G.po-edges can be merged into single G.po-edges by transitivity property. Therefore, all the events, in the $(\mathsf{G.po} \cup \mathsf{G.sw})^+$-path, except for the first and the last ones (followed or preceded by a new po-edge respectively), are events from the target graph. Moreover, if there is

an edge of the form $[w]; \mathrm{G.hb}$ or $\mathrm{G.hb}; [w]$, then in target graph there was a correspondent edge $[w']; \mathrm{G'.hb}$ or $\mathrm{G'.hb}; [w']$.

After this, we can return to the proof of irreflexivity of $\mathrm{G.hb}; \mathrm{G.eco}$. Let us consider the first component of the summands: $\mathrm{G.hb}; \mathsf{m} \uparrow \mathrm{G'.eco}$. We can rewrite it as

$$\mathsf{m} \uparrow (\mathrm{G'.hb}; \mathrm{G'eco}) \cup [w]; \mathrm{G.hb}; \mathsf{m} \uparrow \mathrm{G'.eco} \cup \mathrm{G.hb}; [w]; \mathsf{m} \uparrow \mathrm{G'.eco}.$$

The first part is irreflexive by the Coherence property of $\mathrm{G'}$, and in the latter two, we can see that in the target graph there was an edge $[w']; \mathrm{G'.hb}$ or $\mathrm{G'.hb}; [w']$, which contradicts the irreflexivity of $\mathrm{G'.hb}; \mathrm{G'.eco}$.

Now let's consider the second component of the summands: $\mathrm{G.hb}; \mathit{InsertBefore}(\mathsf{m} \uparrow \mathrm{G'.mo}, \mathsf{m} \uparrow W', w); \mathrm{G'.rf}^?$. If the rf-edge doesn't exist, we come to a simple contradiction, and if it exists, then we have a cycle of the form $[w]; \mathrm{G.mo}; [m(w')]; \mathrm{G.rf}; [m(r)]; \mathrm{G.hb}$, which means, that there was a $\mathrm{G'.hb}$-edge between w' and r, which contradicts the irreflexivity of $\mathrm{G'.hb}; \mathrm{G'.eco}$.

The last component of the summands is $\mathrm{G.hb}; \mathrm{G'.rf}^{-1}; \mathit{InsertBefore}(\mathsf{m} \uparrow \mathrm{G'.mo}, \mathsf{m} \uparrow W', w); \mathrm{G'.rf}^?$. This case is impossible as G.rf-edges can not start with w, which finally proves the irreflexivity of $\mathrm{G.hb}; \mathrm{G.eco}$.

Let us now prove the irreflexivity of G.hb. By definition, $\mathrm{hb} = (\mathrm{po} \cup \mathrm{sw})^+$, so essentially we have a path of po-edges and sw-edges. The po-edges can be merged into single po-edge by transitivity property, and sw-edges can not start with new event. Hence, if there are sw-edges in the path, then there are no new events inside of it, then a new event can be only in the beginning and the end of the path (since the end is the beginning in terms of irreflexivity). And if there are no sw-edges, then the path is a sequence of po-edges, which can be merged into a single po-edge. Now, if there is a new event w at the beginning of the path, then it is connected with something only via po-edges, hence there is respective hb-construction in the target graph with w replaced by w', which contradicts the irreflexivity of $\mathrm{G'.hb}$. And if there is $m(e)$ with $e \in \mathrm{G'.E}$ at the end of the path, then there is the same hb-construction in the target graph, which contradicts the irreflexivity of $\mathrm{G'.hb}$.

(Atomicity)
We need to prove that $\mathrm{G.rmw} \cap (\mathrm{G.fr}; \mathrm{G.mo}) \equiv \emptyset$. Using

$$\mathrm{fr} \subseteq \mathsf{m} \uparrow \mathrm{G'.fr} \cup \mathrm{G.rf}^{-1}; \mathit{InsertBefore}(\mathsf{m} \uparrow \mathrm{G'.mo}, \mathsf{m} \uparrow W', w),$$

we can rewrite the intersection as

$$\begin{aligned}
&\mathsf{m} \uparrow \mathrm{G'.rmw} \cap (\mathsf{m} \uparrow (\mathrm{G'.fr}; \mathrm{G'.mo}) \\
&\qquad \cup \mathsf{m} \uparrow \mathrm{G'.fr}; \mathit{InsertBefore}(\mathsf{m} \uparrow \mathrm{G'.mo}, \mathsf{m} \uparrow W', w) \\
&\qquad \cup \mathrm{G.rf}^{-1}; \mathit{InsertBefore}(\mathsf{m} \uparrow \mathrm{G'.mo}, \mathsf{m} \uparrow W', w); \mathsf{m} \uparrow \mathrm{G'.mo} \\
&\qquad \cup \mathrm{G.rf}^{-1}; \mathit{InsertBefore}(\mathsf{m} \uparrow \mathrm{G'.mo}, \mathsf{m} \uparrow W', w); \mathit{InsertBefore}(\mathsf{m} \uparrow \mathrm{G'.mo}, \mathsf{m} \uparrow W', w))
\end{aligned}$$

The second and the fourth summands are empty as G.rf-edges can not start with w and the third one as well, since fr-edge can not connect two writes. The remaining intersection is empty by the Atomicity property of $\mathrm{G'}$. □

Definition 22. Memory consistency model $\mathcal{M}$ is called *same-read extensible* if for every execution graphs $\mathrm{G'}$ and G, an event mapping function $\mathsf{m} : \mathrm{G'.E} \to \mathrm{G.E}$, an event $e' \in \mathrm{G'}$, and a read event $r \in \mathrm{G.E}$, consistency of $\mathrm{G'}$ implies consistency of G:

$$\mathit{Consistent}(\mathcal{M}, \mathrm{G'}) \implies \mathit{Consistent}(\mathcal{M}, \mathrm{G})$$

provided that the following conditions are met (where $R \triangleq \{r\}$):

- m is injective
- $\mathrm{G.tid} \circ \mathrm{m} = \mathrm{G}'.\mathrm{tid}$
- $\mathrm{G.lab} \circ \mathrm{m} =_{|\mathrm{G}'.E \setminus \mathrm{m}\downarrow R}} \mathrm{G}'.\mathrm{lab}$
- $\mathrm{G.E} = \mathrm{m} \uparrow \mathrm{G}'.\mathrm{E} \cup R$
- $\mathrm{G.po} = \mathrm{m} \uparrow \mathrm{G}'.\mathrm{po} \cup \mathit{InsertAfter}(\mathrm{m} \uparrow \mathrm{G.po}, \{\mathrm{m}(e')\}, R)$
- $\mathrm{G.rf} = \mathrm{m} \uparrow \mathrm{G}'.\mathrm{rf} \cup \mathrm{m} \uparrow \mathit{WriteOf}(\mathrm{G}', e') \times R$
- $\mathrm{G.mo} = \mathrm{m} \uparrow \mathrm{G}'.\mathrm{mo}$
- $\mathrm{G.rmw} = \mathrm{m} \uparrow \mathrm{G}'.\mathrm{rmw}$

Lemma 11. C20 *is* same-read extensible.

Proof. (Consistency)
We need to prove that $\mathrm{G.hb}; \mathrm{G.eco}^?$ is irreflexive. We will prove that G.hb is irreflexive later, so now it is sufficient to prove that $\mathrm{G.hb}; \mathrm{G.eco}$ is irreflexive.
We know that

$$\begin{aligned}
\mathrm{G.fr} &= \mathrm{G.rf}^{-1}; \mathrm{G.mo} \\
&\subseteq \mathrm{m} \uparrow (\mathrm{G}'.\mathrm{rf}^{-1}; \mathrm{G}'.\mathrm{mo}) \cup R \times \mathrm{m} \uparrow \mathit{WriteOf}(\mathrm{G}', e'); \mathrm{G.mo} \\
\mathrm{G.eco} &= \mathrm{G.rf} \cup \mathrm{G.mo}; \mathrm{G.rf}^? \cup \mathrm{G.fr}; \mathrm{G.rf}^? \\
&\subseteq \mathrm{m} \uparrow \mathrm{G}'.\mathrm{rf} \\
&\cup \mathrm{m} \uparrow (\mathrm{G}'.\mathrm{mo}; \mathrm{G}'.\mathrm{rf}^?) \\
&\cup \mathrm{m} \uparrow (\mathrm{G}'.\mathrm{fr}; \mathrm{G}'.\mathrm{rf}^?) \\
&\cup \mathrm{m} \uparrow \mathit{WriteOf}(\mathrm{G}', e') \times R \\
&\cup \mathrm{G.mo}; (\mathrm{m} \uparrow \mathit{WriteOf}(\mathrm{G}', e') \times R)^? \\
&\cup \mathrm{m} \uparrow \mathrm{G}'.\mathrm{fr}; (\mathrm{m} \uparrow \mathit{WriteOf}(\mathrm{G}', e') \times R)^? \\
&\cup R \times \mathrm{m} \uparrow \mathit{WriteOf}(\mathrm{G}', e'); \mathrm{G.mo}; \mathrm{G.rf}^? \\
&\cup R \times \mathrm{m} \uparrow \mathit{WriteOf}(\mathrm{G}', e'); \mathrm{G.mo}; (\mathrm{m} \uparrow \mathit{WriteOf}(\mathrm{G}', e') \times R)^?
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\mathrm{G.hb}; \mathrm{G.eco} &\subseteq \mathrm{G.hb}; \mathrm{m} \uparrow \mathrm{G}'.\mathrm{eco} \\
&\cup \mathrm{G.hb}; \mathrm{m} \uparrow \mathit{WriteOf}(\mathrm{G}', e') \times R \\
&\cup \mathrm{G.hb}; \mathrm{G.mo}; (\mathrm{m} \uparrow \mathit{WriteOf}(\mathrm{G}', e') \times R)^? \\
&\cup \mathrm{G.hb}; \mathrm{m} \uparrow \mathrm{G}'.\mathrm{fr}; (\mathrm{m} \uparrow \mathit{WriteOf}(\mathrm{G}', e') \times R)^? \\
&\cup \mathrm{G.hb}; R \times \mathrm{m} \uparrow \mathit{WriteOf}(\mathrm{G}', e'); \mathrm{G.mo}; \mathrm{G.rf}^? \\
&\cup \mathrm{G.hb}; R \times \mathrm{m} \uparrow \mathit{WriteOf}(\mathrm{G}', e'); \mathrm{G.mo}; (\mathrm{m} \uparrow \mathit{WriteOf}(\mathrm{G}', e') \times R)^?
\end{aligned}$$

Let us first show that $\mathrm{G.sw} = \mathrm{m} \uparrow \mathrm{G}'\mathrm{sw} \cup \mathrm{G.sw}; [r]$. We should show that all the $\mathrm{G}'.\mathrm{sw}$ edges are mapped to G.sw-edges and no new G.sw-edges are added. By definition,

$$\mathrm{sw} = [\mathcal{E}^{\mathrm{REL}\sqsubseteq}]; ([\mathrm{F}]; \mathrm{po})^?; [\mathrm{W}^{\mathrm{RLX}\sqsubseteq}]; (\mathrm{rf}; \mathrm{rmw})^*; \mathrm{rf}; [\mathrm{R}^{\mathrm{RLX}\sqsubseteq}]; (\mathrm{po}; [\mathrm{F}])^?; [\mathcal{E}^{\mathrm{ACQ}\sqsubseteq}]$$

Hence, if we have an sw-edge, it consists of po-edges, rf-edges, and rmw-edges. It is easy to see, that if there was an $\mathrm{G}'.\mathrm{sw}$-edge, then it would remain an G.sw-edge as all of its parts are mapped into the corresponding classes of edges. Now, let's consider the reverse direction. We have some path of sw-form between $m(a)$ and $m(b)$ in the source graph. If doesn't contain r, then it is mapped to the correspondent path in the target graph. If it contains r, we can consider all the possibilities

of where it can appear. It can not be a part of rmw-edge, since rmw-edges are preserved under m. Hence, the only possibility of r is to appear as $[\mathsf{R}^{\mathrm{RLX}\sqsubseteq}]$ in definition of sw. So, there are two possibilities: r can be equal to $m(b)$ or be between rf and po-edges. In the first case, we obtain $\mathsf{G.sw}; [r]$ and moreover, in the target graph there exists a correspondent sw-edge, starting from a and ending in e' (as we the rest of sw-elements are preserved and for the last G.rf-edge there is a correspondent G′.rf-edge ending in e'). In the second case, we can see that the respective preceding and following rf and po-edges exist in the target graph for e', hence, the correspondent sw-edge in the target graph exists as well.

Now, let us proceed to $\mathsf{G.hb} = \mathsf{m} \uparrow \mathsf{G'.hb} \cup [r]; \mathsf{G.hb} \cup \mathsf{G.hb}; [r]$. By definition, $\mathsf{G.hb} = (\mathsf{G.po} \cup \mathsf{G.sw})^+$, and sequences of continuous G.po-edges can be merged into single G.po-edges by transitivity property. Therefore, we have three cases. First, the first or the last event is r – this case is fine, and additionally, if there is an edge of the form $[r]; \mathsf{G.hb}$ or $\mathsf{G.hb}; [r]$, then in target graph there was a correspondent edge $[e']; \mathsf{G'.hb}$ or $\mathsf{G'.hb}; [e']$. Second, r is in the middle of the path and the beginning and the end of the path are events from the target graph (c and d). Hence, if we have a part of the form $[m(a)]; \mathsf{G.sw}; [r]; \mathsf{G.po}; [m(b)]$, then in the target graph there was a correspondent part of the form $[a]; \mathsf{G'.sw}; [e']; \mathsf{G'.po}; [b]$, which means that between the c and d there already was an G′.hb-edge (the rest of the edges between target events exist as well as m-preimages of correspondent edges). Lastly, if there is no r in the path, then all the events are from the target graph, the same path existed in the target path and the proof is finished.

After this, we can return to the proof of irreflexivity of G.hb; G.eco. Let us consider the first component of the summands: $\mathsf{G.hb}; \mathsf{m} \uparrow \mathsf{G'.eco}$. We can rewrite it as

$$\mathsf{m} \uparrow (\mathsf{G'.hb}; \mathsf{G'eco}) \cup [r]; \mathsf{G.hb}; \mathsf{m} \uparrow \mathsf{G'.eco} \cup \mathsf{G.hb}; [r]; \mathsf{m} \uparrow \mathsf{G'.eco}.$$

The first part is irreflexive by the Coherence property of G′, and in the latter two, we can see that in the target graph there was an edge $[e']; \mathsf{G'.hb}$ or $\mathsf{G'.hb}; [e']$, which contradicts the irreflexivity of G′.hb; G′.eco.

Now let's consider the rest of the summands.

- $\mathsf{G.hb}; \mathsf{m} \uparrow \mathit{WriteOf}(\mathsf{G}', e') \times R$ is empty, such hb; po cycle is impossible.
- $\mathsf{G.hb}; \mathsf{G.mo}; (\mathsf{m} \uparrow \mathit{WriteOf}(\mathsf{G}', e') \times R)^?$. If the latter edge doesn't exist, we come to a simple contradiction, and if it exists, e' is a write event, and since there is a hb-edge after r, then there is the same edge, starting from e', which contradicts the irreflexivity of G′.hb; G′.eco.
- $\mathsf{G.hb}; \mathsf{m} \uparrow \mathsf{G'.fr}; (\mathsf{m} \uparrow \mathit{WriteOf}(\mathsf{G}', e') \times R)^?$. If the latter edge doesn't exist, then we have a subcase of the very first summand. And if it exists, then e' is a write event, and there is a hb-edge coming from e' (let's say to $m(a)$), correspondent to the hb-edge coming from r into a, which leads contradiction with the irreflexivity of G′.hb; G′.eco.
- $\mathsf{G.hb}; R \times \mathsf{m} \uparrow \mathit{WriteOf}(\mathsf{G}', e'); \mathsf{G.mo}; \mathsf{G.rf}^?$. If the latter edge doesn't exist, then e' is a write event, there is a hb-edge incoming to e', correspondent to the hb-edge incoming to r, which leads to a contradiction with the irreflexivity of G′.hb; G′.eco within the cycle of hb; mo. If the rf-edge exists, then from one side, rf-edge should end with the read event, and from the other side, the mo-edge after that should start with the write event, which is impossible.
- $\mathsf{G.hb}; R \times \mathsf{m} \uparrow \mathit{WriteOf}(\mathsf{G}', e'); \mathsf{G.mo}; (\mathsf{m} \uparrow \mathit{WriteOf}(\mathsf{G}', e') \times R)^?$. If the latter edge doesn't exist, then we get a subcase of the fourth bullet. If it exists, then we have a mo-edge between $\mathit{WriteOf}(\mathsf{G}', e')$ and $\mathit{WriteOf}(\mathsf{G}', e')$, which is impossible.

Therefore, we have shown that G.hb; G.eco is irreflexive.

Let us now prove the irreflexivity of G.hb. There are two cases: a cycle of the form $\mathsf{G.hb}; [r]$ or $\mathsf{G.hb}; [m(e)]$, where $e \in \mathsf{G'.E}$. In the second case, we can see that there is a correspondent G′.*hb*-edge in the target graph, which contradicts the irreflexivity of G′.hb. In the first case, let us

recall the definition of hb : $\mathsf{hb} = (\mathsf{po} \cup \mathsf{sw})^+$, which is essentially a path of po-edges and sw-edges. The po-edges can be merged into a single po-edge by transitivity property. Now, we can substitute all the possible r-entries in the path by $m(e')$-entries, which means, that there was a correspondent $\mathrm{G}'.\mathsf{hb}$-edge in the target graph, which contradicts the irreflexivity of $\mathrm{G}'.\mathsf{hb}$. The only thing we have to ensure is that there are no $[m(e')]; \mathrm{G}.\mathsf{po}; [r]$-edges in this path, but this is impossible, since the following sw-edge can not start with r.

(Atomicity)
We need to prove that $\mathrm{G}.\mathsf{rmw} \cap (\mathrm{G}.\mathsf{fr}; \mathrm{G}.\mathsf{mo}) \equiv \emptyset$. Using

$$\mathsf{fr} \subseteq \mathsf{m} \uparrow (\mathrm{G}'.\mathsf{rf}^{-1}; \mathrm{G}'.\mathsf{mo}) \cup R \times \mathsf{m} \uparrow \mathit{WriteOf}(\mathrm{G}', e'); \mathrm{G}.\mathsf{mo},$$

we can rewrite the intersection as

$$\mathsf{m} \uparrow \mathrm{G}'.\mathsf{rmw} \cap (\mathsf{m} \uparrow (\mathrm{G}'.\mathsf{fr}; \mathrm{G}'.\mathsf{mo}) \cup R \times \mathsf{m} \uparrow \mathit{WriteOf}(\mathrm{G}', e'); \mathrm{G}.\mathsf{mo}; \mathrm{G}.\mathsf{mo})$$

The second summand is empty as G.rmw-edges can not start with r. The remaining intersection is empty by the Atomicity property of G'. □

C Proof of Data Race Freedom Guarantees

In this section, we present a detailed proof of Lemma 4 — a key lemma used to prove the DRF guarantees in Theorem 1 (see §3.4 for the proof of the theorem itself). Below we repeat the statement of the lemma and proceed with its proof.

Lemma. *Let $\mathcal{M}$ be a write-read coherent memory model, and* P *be a program, such that all of its* porf *acyclic $\mathcal{M}$-consistent execution graphs are* relax (RLX) race-free. *Suppose additionally that* G *is a* porf *acyclic $\mathcal{M}$ consistent graph of this program. Then every graph* G'*, such that* $\mathrm{G} \to \mathrm{G}'$*, is also* porf *acyclic.*

Proof. The statement is obviously true for the case of (Execute), as it only adds a single event that has no outgoing porf edges, and thus cannot form a new cycle.

The case of (Re-Execute) is more sophisticated. We need to consider a *re-execution frontier* $\mathcal{F}$ that is a set of events from G' immediately succeeding determined events $\mathcal{D}$ in program order. Without loss of generality, we can assume that the set $\mathcal{F}$ consists of uncommitted reads, which read-from a different write event compared to G. If some event in the frontier does not satisfy these constraints, then it can be safely added to the set $\mathcal{D}$ since it does not have to be re-executed.

Moreover, we can assume all reads in the frontier $\mathcal{F}$ are relaxed. Otherwise, an acquire read would induce an rpo edge, which is forbidden by the (Re-Execute) rule.

From all the relaxed read events in the frontier $\mathcal{F}$ we can pick one that is going to be re-executed first. Thus, during re-execution, it has to read from some write event lying in $\mathcal{D}$, and thus also belonging to G. So let r be the chosen read event, w_1 be a write event it reads-from in G, and w_2 be a write event it reads-from in G' (all three belong to G).

In case if either w_1 or w_2 is in race with r we have a contradiction with the assumption that porf acyclic graphs of P are relax race-free. So let's consider the case when both w_1 and w_2 happen-before r.

We have that $\langle w_1, r\rangle \in \mathrm{G}.\mathsf{hb}$ and $\langle w_2, r\rangle \in \mathrm{G}.\mathsf{hb}$. Note that it also implies $\langle w_1, r\rangle \in \mathrm{G}'.\mathsf{hb}$ and $\langle w_2, r\rangle \in \mathrm{G}'.\mathsf{hb}$, because all three events belong to a common prefix of both graphs consisting of $\mathcal{D}$ events.

Because G.mo is total it has to order w_1 or w_2 in one way or another.

Suppose the G.mo edge goes from w_1 to w_2. Now we have a cycle $r \xrightarrow{\text{fr}} w_2$ and $w_2 \xrightarrow{\text{hb}} r$ in the graph G, which contradicts to write-read coherence implied by $\mathcal{M}$-consistency of G.

Symmetrically, let the G.mo edge go from w_1 to w_2. Then we have a cycle $r \xrightarrow{\text{fr}} w_1$ and $w_1 \xrightarrow{\text{hb}} r$ in the graph G′, which contradicts to write-read coherence implied by $\mathcal{M}$-consistency of G′.

□

D Soundness of Compilation Mappings

We re-state the definition of the IMM model given in [Podkopaev et al. 2019].

Definition 23. We define *internal* and *external* versions of the reads-from and modification order:

$$\text{rfi} \triangleq \text{rf} \cap =_{\text{tid}} \qquad \text{moi} \triangleq \text{mo} \cap =_{\text{tid}}$$
$$\text{rfe} \triangleq \text{rf} \setminus =_{\text{tid}} \qquad \text{moe} \triangleq \text{mo} \setminus =_{\text{tid}}$$

Definition 24. IMM model augments the execution graphs with additional *syntactic dependency* relations, that model the dependencies between instructions of a program on the level of events. In IMM, the execution graph is supplemented with the following relations: data—*data dependency*, ctrl—*control dependency*, addr—*address dependency*, and casdep—*compare-and-set dependency*.

The combined *dependency* relation deps is defined as follows:

$$\text{deps} \triangleq \text{data} \cup \text{ctrl} \cup \text{addr} ; \text{po}^? \cup \text{casdep}.$$

Definition 25. IMM introduces a number of derived relations given below.

- ppo is *preserved program order*, a subset of program order which includes ordering constraints due to syntactic dependencies:

$$\text{ppo} \triangleq [\text{R}] ; (\text{deps} \cup \text{rfi})^+ ; [\text{W}]$$

- bob is *barrier-order-before* relation, a subset of program order which includes ordering constraints due to memory fence placement:

$$\text{bob} \triangleq \text{po} ; [\text{W}^{\text{REL}\sqsubseteq}] \cup [\text{R}^{\text{ACQ}\sqsubseteq}] ; \text{po} \cup \text{po} ; [\text{F}] \cup [\text{F}] ; \text{po} \cup [\text{W}^{\text{REL}\sqsubseteq}] ; \text{po}_{\text{loc}} ; [\text{W}]$$

- detour associates a write event with a subsequent read event reading externally from the same memory location:

$$\text{detour} \triangleq (\text{moe} ; \text{rfe}) \cap \text{po}$$

- ar is the *acyclic relation*, a global causality relation on all the events in the IMM graph:

$$\text{ppo} \cup \text{bob} \cup \text{detour} \cup \text{rfe}$$

Definition 26 (IMM consistency)**.**

- $\text{R} \subseteq \text{codom}(\text{rf})$ (rf-completeness)
- $\text{hb}; \text{eco}^?$ is irreflexive. (coherence)
- $\text{rmw} \cap (\text{fr}; \text{mo})$ is empty (atomicity)
- ar is an acyclic relation. (no-thin-air)

Lemma 12. *Every* IMM *consistent graph is* C20 *consistent.*

Proof. Trivial, since C20 is weaker than IMM. □

Next, we re-state the compilation correctness theorem 2 from §3.5 and provide its proof.

Theorem. *Let* P *be a program, and* G *be its* IMM *consistent execution graph. Then* G *is also an* XC20 *consistent execution graph.*

Proof. Note that by Lemma 12, G is is C20 consistent.

Let $[r_1, \ldots, r_n]$ be a sequence of all the read events G.R of the given graph, ordered in accordance with the ar relation, that is $\langle r_i, r_j \rangle \in \mathsf{ar} \implies i < j$.

We define a sequence of execution graphs $[\mathsf{G}_0, \mathsf{G}_1, \ldots, \mathsf{G}_n, \mathsf{G}_{n+1}]$, where $\mathsf{G}_0 \triangleq \mathsf{G}_{\mathsf{init}}$, $\mathsf{G}_{n+1} \triangleq \mathsf{G}$, and for any other i graph G_i is defined as follows:

- $\mathsf{G}_i.\mathsf{E} = \mathsf{G}.\mathsf{E} \setminus \mathsf{codom}([r_i]; (\mathtt{ctrl} \cup \mathtt{addr}; \mathsf{po}^? \cup \mathtt{casdep} \cup \mathsf{bob}); \mathsf{porf}^?)$
- $\mathsf{G}_i.\mathsf{tid} = \mathsf{G}.\mathsf{tid}|_{\mathsf{G}_i.\mathsf{E}}$
- $\mathsf{G}_i.\mathsf{lab} = \mathsf{G}.\mathsf{lab}|_{\mathsf{G}_i.\mathsf{E}}$
- $\mathsf{G}_i.\mathsf{po} = \mathsf{G}.\mathsf{po}|_{\mathsf{G}_i.\mathsf{E}}$
- $\mathsf{G}_i.\mathsf{rf} = \mathsf{G}.\mathsf{rf}|_{\{r_1,\ldots,r_{i-1}\}} \cup \mathsf{G}_{i+1}.\mathtt{srf}; [\{r_i\}] \cup \mathsf{G}_{i+1}.\mathsf{rf}|_{\{r_{i+1},\ldots,r_n\}}$
- $\mathsf{G}_i.\mathsf{mo} = \mathsf{G}.\mathsf{mo}|_{\mathsf{G}_i.\mathsf{E}}$

To show that each graph G_i is IMM consistent, we use the following facts:

- G_i it is a subgraph of IMM consistent graph G, where some subset of read events was redirected to read-from stable write using srf relation;
- $\mathsf{ar} \cup \mathtt{srf}$ is an acyclic relation.

Therefore, for each i we have $\mathit{Consistent}(\mathsf{G}_i, \mathsf{IMM})$, and by Lemma 12 we also derive $\mathit{Consistent}(\mathsf{G}_i, \mathsf{C20})$.

We prove that for any $i = 1 \ldots n$ it is true that $\mathsf{G}_i \xrightarrow[re-exec]{C_i} \mathsf{G}_{i+1}$ holds. To apply the re-execution rule, we set:

- $C_i \triangleq \mathsf{G}_{i+1}.\mathsf{E} \setminus \mathsf{codom}([\{r_i\}]; \mathsf{G}.\mathsf{ar}^?)$
- $\mathcal{D}_i \triangleq \mathsf{G}_i.\mathsf{E} \setminus \mathsf{codom}([\{r_i\}]; \mathsf{G}_i.\mathsf{po}^?)$
- $\mathsf{f}_i \triangleq \mathsf{id}$

The following properties follow immediately from these definitions:

- $C_i \subseteq \mathsf{G}_{i+1}.\mathsf{E}$
- $\mathcal{D}_i \subseteq \mathsf{id} \upharpoonright C_i = C_i$
- $\mathsf{dom}(\mathsf{G}.\mathsf{po}; [\mathcal{D}_i]) \subseteq \mathcal{D}_i$
- $\mathit{WellFormed}(\mathsf{G}_i|_{\mathcal{D}_i}, \mathsf{G}_{i+1}, C_i)$

$\mathit{ThreadOrderedUEvents}(\mathsf{G}_{i+1}, C_i, \preceq_{\mathsf{tid}\,i})$ holds trivially, since we re-execute single thread.

The $\mathit{CommitEmbedded}(\mathsf{G}_i, \mathsf{G}_{i+1}, C_i, \mathsf{f}_i)$ holds because C_i includes all the events, except those causally depending on r_i via ar relation. By definition of ar (as it includes syntactic dependency relation deps and external reads-from rfe), changing the reads-from for r_i can only affect the events in its ar suffix — which are uncommitted. As such, all the committed events are going to be preserved during re-execution.

Moreover, it can be shown that none of the events in the threads outside re-executed thread causally depend via ar relation on r_i. Indeed, any ar path from r_i to some external event should start with ppo; rfe. Therefore, due to rfe edge, it would necessarily involve a read event r_j. That is, $\langle r_i, r_j \rangle \in \mathsf{ppo}; \mathsf{rfe} \subseteq \mathsf{ar}$, and thus $i < j$ by construction of the ordered list of reads $[r_1, \ldots, r_n]$. As such, the read r_j had to be already redirected to read from stable write, and thus it cannot read externally via rfe — we get a contradiction.

Therefore, the re-execution is possible, and $\langle \mathsf{G}_{i+1}, C_i \rangle \vdash \mathsf{G}_i|_{\mathcal{D}_i} \Rightarrow^* \mathsf{G}_{i+1}$ holds.

Notice that in the graph G_0 all reads are reading from stable writes (*i.e.,* via srf relation), and thus this graph is porf acyclic. As such, it can be constructed using multiple execution steps: $\mathsf{G}_{\mathsf{init}} \xrightarrow[exec]{}{}^* \mathsf{G}_1$.

In conclusion, we have $\mathsf{G}_0 \to^* \mathsf{G}_1 \ldots \to^* \mathsf{G}_n \to^* \mathsf{G}_{n+1}$, implying that G is XC20 consistent.

□

E Soundness of Program Transformations

E.1 Elimination

We prove the soundness of the elimination transformations on the level of the execution graphs by eliminating one of the two adjacent events.

Definition 27. We define a relation $\mathcal{M}(G_s, G_t, m, a_t)$ asserting that the target graph G_t is obtained as a result of applying elimination transformation to the source graph G_s, where $m : G_t.E \to G_s.E$ is an event mapping function, a_t is the eliminated event in G_t:

- m is injective
- $G_s.E = m \uparrow G_t.E \cup \{a_s\}$
- $G_s.tid \circ m = G_t.tid$
- $G_s.lab \circ m = G_t.lab$
- $G_s.po = m \uparrow G_t.po \cup \mathcal{PO}$
- $G_s.rf = m \uparrow G_t.rf \cup \mathit{WriteOf}(G_t, a_t) \times (\{b_s\} \cap G_s.R)$
- $G_s.mo = m \uparrow G_t.mo \cup \mathit{InsertBefore}(m \uparrow G_t.mo, \{a_s\}, \{b_s\} \cap G_s.W)$
- $G_s.rmw = m \uparrow G_t.rmw$

where:

$$\mathcal{PO} \triangleq \begin{cases} \mathit{InsertBefore}(m \uparrow G_t.po, \{b_s\}, \{a_s\}) & \text{for Store-Store elimination} \\ \mathit{InsertAfter}(m \uparrow G_t.po, \{a_s\}, \{b_s\}) & \text{otherwise} \end{cases}$$

Store-Load and Load-Load Eliminations.

Definition 28. $\mathcal{R}(G_s, G_t, m, a_t)$ — is a *simulation relation* for store-load and load-load elimination transformations, where G_s is a source graph, G_t is a target graph, $m : G_t.E \to G_s.E$ is an event mapping function, $a_t \in (W \uplus R)$ is an event in G_t, as follows

- m is injective
- $G_s.tid \circ m = G_t.tid$
- $G_s.lab \circ m = G_t.lab$
- $G_s.E = m \uparrow G_t.E \cup \mathcal{B}_s$
- $G_s.po = m \uparrow G_t.po \cup \mathit{InsertAfter}(m \uparrow G_t.po, \mathcal{A}_s, \mathcal{B}_s)$
- $G_s.rf = m \uparrow G_t.rf \cup m \uparrow \mathit{WriteOf}(G_t, a_t) \times \mathcal{B}_s$
- $G_s.mo = m \uparrow G_t.mo$

where:

$$a_s \triangleq m(a_t)$$

$$\mathcal{A}_s \triangleq \{a_s\}$$

$$\mathcal{B}_s \triangleq \begin{cases} \{b_s\} & \text{if } a_t \in G_t.E\,, \quad \text{where } b_s \text{ is an event, } \textit{s.t. } \mathcal{P}(b_s) \text{ holds} \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathcal{P}(b_s) \triangleq (G_s.op(b_s) = R \wedge G_s.loc(b_s) = G_s.loc(a_s) \wedge G_s.val(b_s) = G_s.val(a_s))$$

Lemma 13. *Let G_t^{init} and G_s^{init} be initial execution graphs. Then $\mathcal{R}(G_s^{init}, G_t^{init}, \mathrm{id}, a_t, b_t)$ holds.*

Proof. Easy to see that $\mathcal{B}_s = \emptyset$, so the first four conditions of Definition 28 hold. As well as the rest of them, since the sets of po, rf and mo edges are empty for initial execution graphs and $\mathcal{B}_s$ is once again empty. □

Lemma 14. *Let* G_t *and* G_s *be* $\mathcal{M}$ *consistent and assume* $\mathcal{R}(G_s, G_t, m, a_t)$ *holds. If* $G_t \to G_t'$ *holds then there exists* G_s' *and* m' *such that* $G_s \to^* G_s'$ *and* $\mathcal{R}(G_s', G_t', m', a_t)$ *also hold.*

Proof. We do a case analysis on $G_t \to G_t'$ and use Lemmas 15 to 17 to derive G_s' and $\mathcal{R}(G_s', G_t', m, a_t, b_t)$. □

Lemma 15. *Let* G_t *and* G_s *be* $\mathcal{M}$ *consistent and assume* $\mathcal{R}(G_s, G_t, m, a_t)$ *holds. If* $G_t \xrightarrow[exec]{e \notin \{a_t\}} G_t'$ *holds then there exists* G_s' *and* m' *such that* $G_s \xrightarrow[exec]{m'(e)} G_s'$ *and* $\mathcal{R}(G_s', G_t', m', a_t)$ *also hold.*

Proof. Let us define e' as an event, s.t. $\mathcal{P}_e(e')$ holds:

$$\mathcal{P}_e(e') \triangleq (G_s.\mathsf{op}(e') = G_t.\mathsf{op}(e) \wedge G_s.\mathsf{loc}(e') = G_t.\mathsf{loc}(e) \wedge G_s.\mathsf{ord}(e') = G_t.\mathsf{ord}(e))$$

Therefore, we can construct G_s' and m' as follows:

- $m' = m[e \mapsto e']$
- $G_s'.E = G_s.E \cup \{e'\}$
- $G_s'.\mathsf{tid} = G_s.\mathsf{tid}[e' \mapsto G_t'.\mathsf{tid}(e)]$
- $G_s'.\mathsf{lab} = G_s.\mathsf{lab}[e' \mapsto G_t'.\mathsf{lab}(e)]$
- $G_s'.\mathsf{po} = G_s.\mathsf{po} \cup m \uparrow G_t'.\mathsf{po}; [\{e\}]$
- $G_s'.\mathsf{rf} = G_s.\mathsf{rf} \cup m \uparrow G_t'.\mathsf{rf}; [\{e\} \cap G_t'.R]$
- $G_s'.\mathsf{mo} = G_s.\mathsf{mo} \cup m \uparrow [\{e\} \cap G_t'.W]; G_t'.\mathsf{mo} \cup m \uparrow G_t'.\mathsf{mo}; [\{e\} \cap G_t'.W]$
- $G_s'.\mathsf{rmw} = m \uparrow G_t'.\mathsf{rmw}$

From this construction and Definition 28, it follows immediately that $\mathcal{R}(G_s', G_t', m', a_t)$ holds.

In case when $\mathcal{B}_s$ is empty, consistency of the source graph, $\mathit{Consistent}(\mathcal{M}, G_s')$, follows directly from the consistency of the target graph, $\mathit{Consistent}(\mathcal{M}, G_t')$, due to monotonicity property Lemma 7. In the other case when $\mathcal{B}_s \neq \emptyset$, it follows from Lemma 11. □

Lemma 16. *Let* G_t *and* G_s *be* $\mathcal{M}$ *consistent and assume* $\mathcal{R}(G_s, G_t, m, a_t)$ *holds. If* $G_t \xrightarrow[exec]{a_t} G_t'$ *holds then there exists* G_s' *and* m' *such that* $G_s \xrightarrow[exec]{a_s} \xrightarrow[exec]{b_s} G_s'$ *holds, with* $a_s \triangleq m'(a_t)$*, and* $\mathcal{R}(G_s', G_t', m', a_t)$ *also holds.*

Proof. We construct m' and G_s' as follows:

- $m' = m[a_t \mapsto a_s]$
- $G_s'.E = G_s.E \cup \{a_s\} \cup \{b_s\}$
- $G_s'.\mathsf{tid} = G_s.\mathsf{tid}[a_s \mapsto G_t'.\mathsf{tid}(a_t)][b_s \mapsto G_t'.\mathsf{tid}(a_t)]$
- $G_s'.\mathsf{lab} = G_s.\mathsf{lab}[a_s \mapsto G_t'.\mathsf{lab}(a_t)][b_s \mapsto \ell]$
- $G_s'.\mathsf{po} = G_s.\mathsf{po} \cup \mathit{InsertAfter}(m \uparrow G_t.\mathsf{po}, \{a_s\}, \{b_s\})$
- $G_s'.\mathsf{rf} = G_s.\mathsf{rf} \cup \mathit{WriteOf}(G_t, a_t) \times b_s$
- $G_s'.\mathsf{mo} = G_s.\mathsf{mo} \cup m \uparrow [\{a_t\} \cap G_t'.W]; G_t'.\mathsf{mo} \cup m \uparrow G_t'.\mathsf{mo}; [\{a_t\} \cap G_t'.W]$
- $G_s'.\mathsf{rmw} = m \uparrow G_t'.\mathsf{rmw}$

Where ℓ is label for b_s such that:

$$G_s.\mathsf{op}(b_s) = R \wedge G_s.\mathsf{loc}(b_s) = G_s.\mathsf{loc}(a_s) \wedge G_s.\mathsf{val}(b_s) = G_s.\mathsf{val}(a_s)$$

From this construction and Definition 28, it follows immediately that $\mathcal{R}(G_s', G_t', m', a_t)$ holds.

The consistency of the source graph follows from same-read extensibility Lemma 11 applied to the intermediate source graph we get after adding a_s, which is consistent due to Definition 18. □

Lemma 17. *Let* $\mathrm{G_t}$ *and* $\mathrm{G_s}$ *be* $\mathcal{M}$ *consistent and* $\mathcal{R}(\mathrm{G_s}, \mathrm{G_t}, \mathsf{m}, a_t)$ *holds. If* $\mathrm{G_t} \xrightarrow[re-exec]{C} \mathrm{G_t'}$ *holds then there exists* $\mathrm{G_s'}$ *and* m' *such that* $\mathrm{G_s} \xrightarrow[re-exec]{\mathsf{m}' \uparrow C} \mathrm{G_s'}$ *holds, and* $\mathcal{R}(\mathrm{G_s'}, \mathrm{G_t'}, \mathsf{m}', a_t)$ *also holds.*

Proof. We construct m' and $\mathrm{G_s'}$ as follows:

- $\mathsf{m}' = \textbf{if } a_t \in \mathrm{G_t.E} \textbf{ then } \mathsf{id}[a_t \mapsto a_s] \textbf{ else } \mathsf{id}$
- $\mathrm{G_s'.tid} = \mathrm{G_t'.tid} \circ \mathsf{m}$
- $\mathrm{G_s'.lab} = \mathrm{G_t'.lab} \circ \mathsf{m}$
- $\mathrm{G_s'.E} = \mathsf{m} \uparrow \mathrm{G_t'.E} \cup \mathcal{B}_\mathrm{s}$
- $\mathrm{G_s'.po} = \mathsf{m} \uparrow \mathrm{G_t'.po} \cup \mathit{InsertAfter}(\mathsf{m} \uparrow \mathrm{G_t.po}, \mathcal{A}_\mathrm{s}, \mathcal{B}_\mathrm{s})$
- $\mathrm{G_s'.rf} = \mathsf{m} \uparrow \mathrm{G_t'.rf} \cup \mathit{WriteOf}(\mathrm{G_t}, a_t) \times \mathcal{B}_\mathrm{s}$
- $\mathrm{G_s'.mo} = \mathsf{m} \uparrow \mathrm{G_t'.mo}$
- $\mathrm{G_s'.rmw} = \mathsf{m} \uparrow \mathrm{G_t'.rmw}$

From this construction and Definition 28, it follows immediately that $\mathcal{R}(\mathrm{G_s'}, \mathrm{G_t'}, \mathsf{m}', a_t)$ holds. We will define the components of the re-execution step as follows:

- $C_s \triangleq \mathsf{m} \uparrow C_t' \cup \textbf{ if } a_t \in C_t \textbf{ then } \{b_t\} \textbf{ else } \emptyset$
- $\mathcal{D}_s \triangleq \mathsf{m} \uparrow \mathcal{D}_t \cup \textbf{ if } a_t \in \mathcal{D}_t \textbf{ then } \{b_t\} \textbf{ else } \emptyset$
- $\mathsf{f}_s \triangleq \mathsf{m} \circ \mathsf{f}_t \circ \mathsf{m}'^{-1}$

In both cases, the following properties follow immediately from definitions:

- $C_s \subseteq \mathrm{G_s'.E}$
- $\mathcal{D}_s \subseteq C_s$
- $\mathsf{dom}(\mathrm{G.po}; [\mathcal{D}_s]) \subseteq \mathcal{D}_s$
- $\mathit{WellFormed}(\mathrm{G}|_{\mathcal{D}_s}, \mathrm{G_s'}, C_s)$
- $\mathit{CommitEmbedded}(\mathrm{G_s}, \mathrm{G_s'}, C_s, \mathsf{f}_s)$

The property $\mathsf{dom}(\mathrm{G_s'.rpo}; [\mathrm{G_s'.E} \setminus \mathcal{D}_s]) \subseteq \mathcal{D}_s$ follows from the fact that $\mathrm{G_s'.rpo} \subseteq \mathrm{G_t'.rpo}$ and the respective property of the target graph.

To prove $\mathit{ThreadOrderedUEvents}(\mathrm{G'}, C_s, \prec_{\mathsf{tid}})$ we use the following observation:

$$\begin{aligned}\mathrm{G_s'.vf}; \mathrm{G_s'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_s] = [\mathrm{G_s'.W}]; \mathrm{G_s'.rf}; \mathrm{G_s'.hb}^?; \mathrm{G_s'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_s] \subseteq \\ \subseteq \mathsf{m}' \uparrow ([\mathrm{G_t'.W}]; \mathrm{G_t'.rf}; \mathrm{G_t'.hb}^?; \mathrm{G_t'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_t]) \cup \\ \cup \mathrm{G_s'.rf}; [\mathcal{B}]; \mathrm{G_s'.po}; (\mathsf{m}' \uparrow \mathrm{G_t'.hb})^?; \mathrm{G_s'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_s] \cup\end{aligned}$$

- For the first disjunct, we notice that:

$$\mathsf{m}' \uparrow ([\mathrm{G_t'.W}]; \mathrm{G_t'.rf}; \mathrm{G_t'.hb}^?; \mathrm{G_t'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_t]) = \mathsf{m}' \uparrow (\mathrm{G_t'.vf}; \mathrm{G_t'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_t])$$

- For the second disjunct we can construct a similar path in $\mathrm{G_t'}$ to b_t, because a_s and b_s have the same set of outgoing hb edges (except $\langle a_s, b_s \rangle$ edge):

$$\begin{aligned}[\mathrm{G_s'.W}]; \mathrm{G_s'.rf}[\mathcal{B}_\mathrm{s}]; \mathrm{G_s'.po}; (\mathsf{m} \uparrow \mathrm{G_t'.hb})^?; \mathrm{G_s'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_s] \subseteq \\ [\mathrm{G_s'.W}]; \mathrm{G_s'.rf}^?; [\mathcal{A}_\mathrm{s}]; \mathsf{m} \uparrow (\mathrm{G_t'.hb}; \mathrm{G_t'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_t]) \subseteq \\ \mathsf{m} \uparrow (\mathrm{G_t'.vf}; \mathrm{G_t'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_t])\end{aligned}$$

So in the result, we get the desired property:

$$\mathrm{G_s'.vf}; \mathrm{G_s'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_s] \subseteq \mathsf{m} \uparrow (\mathrm{G_t'.vf}; \mathrm{G_t'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_t]) \subseteq \mathsf{m} \uparrow (\mathrm{G_t'.tid} \downarrow \preceq_{\mathsf{tid}}) = \mathrm{G_s'.tid} \downarrow \preceq_{\mathsf{tid}}$$

In case when $\mathcal{B}$ is empty, consistency of the source graph, $\mathit{Consistent}(\mathcal{M}, \mathrm{G}'_\mathrm{s})$, follows directly from the consistency of the target graph, $\mathit{Consistent}(\mathcal{M}, \mathrm{G}'_\mathrm{t})$, due to monotonicity property Lemma 7. And in the other case, it follows from same-read extensibility Lemma 11.

It is left to show that:

$$\langle \mathrm{G}'_\mathrm{s}, C_s \rangle \vdash \mathrm{G}_\mathrm{s}|_{\mathcal{D}_s} \Rightarrow^* \mathrm{G}'_\mathrm{s}$$

For this, we can apply Lemma 1 with ℓ inherited from the target graph, adding the b_s event to the end of the list if needed.

□

Lemma 18. *Suppose that $\mathcal{R}(\mathrm{G}_\mathrm{s}, \mathrm{G}_\mathrm{t}, \mathsf{m}, a_\mathrm{t})$ holds. Then G_s is also $\mathcal{M}$ consistent and terminal, and moreover $\mathcal{M}(\mathrm{G}_\mathrm{s}, \mathrm{G}_\mathrm{t}, \mathsf{m}, b_\mathrm{t})$ holds.*

Proof. $\mathcal{M}$ consistency follows from same-read extensibility property Lemma 11, since $\mathcal{B}$ is not empty, and all the prerequisites are met from the $\mathcal{R}(\mathrm{G}_\mathrm{s}, \mathrm{G}_\mathrm{t}, \mathsf{m}, a_\mathrm{t})$ definition. $\mathcal{M}(\mathrm{G}_\mathrm{s}, \mathrm{G}_\mathrm{t}, \mathsf{m}, b_\mathrm{t})$ holds by the same reasoning. □

Theorem 3.

$$\forall \mathbb{P}_{\mathrm{src}}.\ \mathsf{elim}(\mathbb{P}_{\mathrm{src}}, \mathbb{P}_{\mathrm{tgt}}) \implies \\ \forall \mathrm{G}_\mathrm{t}.\ \mathrm{G}_{\mathsf{init}} \rightarrow^* \mathrm{G}_\mathrm{t}.\ \exists \mathrm{G}_\mathrm{s}.\mathrm{G}_{\mathsf{init}} \rightarrow^* \mathrm{G}_\mathrm{s}.\ \mathsf{Behavior}(\mathrm{G}_\mathrm{s}) = \mathsf{Behavior}(\mathrm{G}_\mathrm{t})$$

Proof. We do induction on the construction steps: $\mathrm{G}_{\mathsf{init}} \rightarrow^* \mathrm{G}_\mathrm{t}$. We use Lemma 13 to show that initial graphs $\mathrm{G}_\mathrm{t}^{\mathsf{init}}$ and $\mathrm{G}_\mathrm{s}^{\mathsf{init}}$ are in simulation relation $\mathcal{R}$. Then using Lemma 14 we show that each step of the target graph construction $\mathrm{G}_\mathrm{t} \rightarrow \mathrm{G}'_\mathrm{t}$ can be simulated by the source graph construction step $\mathrm{G}_\mathrm{s} \rightarrow \mathrm{G}'_\mathrm{s}$. Finally, we use lemma Lemma 18 to derive that the final graph G_s is $\mathcal{M}$ consistent. □

Store-Store Elimination.

Definition 29. $\mathcal{R}(\mathrm{G}_\mathrm{s}, \mathrm{G}_\mathrm{t}, \mathsf{m}, b_\mathrm{t}, v)$ — is a *simulation relation* for store-store elimination transformation, where G_s is a source graph, G_t is a target graph, $\mathsf{m} : \mathrm{G}_\mathrm{t}.\mathsf{E} \rightarrow \mathrm{G}_\mathrm{s}.\mathsf{E}$ is an event mapping function, $b_\mathrm{t} \in \mathsf{W}$ is an event in G_t, $v \in \mathsf{VAL}$ is a value, as follows

- m is injective
- $\mathrm{G}_\mathrm{s}.\mathsf{tid} \circ \mathsf{m} = \mathrm{G}_\mathrm{t}.\mathsf{tid}$
- $\mathrm{G}_\mathrm{s}.\mathsf{lab} \circ \mathsf{m} = \mathrm{G}_\mathrm{t}.\mathsf{lab}$
- $\mathrm{G}_\mathrm{s}.\mathsf{E} = \mathsf{m} \uparrow \mathrm{G}_\mathrm{t}.\mathsf{E} \cup \mathcal{A}_\mathrm{s}$
- $\mathrm{G}_\mathrm{s}.\mathsf{po} = \mathsf{m} \uparrow \mathrm{G}_\mathrm{t}.\mathsf{po} \cup \mathit{InsertBefore}(\mathsf{m} \uparrow \mathrm{G}_\mathrm{t}.\mathsf{po}, \mathcal{B}_\mathrm{s}, \mathcal{A}_\mathrm{s})$
- $\mathrm{G}_\mathrm{s}.\mathsf{rf} = \mathsf{m} \uparrow \mathrm{G}_\mathrm{t}.\mathsf{rf}$
- $\mathrm{G}_\mathrm{s}.\mathsf{mo} = \mathsf{m} \uparrow \mathrm{G}_\mathrm{t}.\mathsf{mo} \cup \mathit{InsertBefore}(\mathsf{m} \uparrow \mathrm{G}_\mathrm{t}.\mathsf{mo}, \mathcal{B}_\mathrm{s}, \mathcal{A}_\mathrm{s})$

where:

$$b_\mathrm{s} \triangleq \mathsf{m}(b_\mathrm{t})$$

$$\mathcal{B}_\mathrm{s} \triangleq \{b_\mathrm{s}\}$$

$$\mathcal{A}_\mathrm{s} \triangleq \begin{cases} \{a_\mathrm{s}\} & \text{if } b_\mathrm{t} \in \mathrm{G}_\mathrm{t}.\mathsf{E}\,, \quad \text{where } a_\mathrm{s} \text{ is an event, } \textit{s.t. } \mathcal{P}(a_\mathrm{s}) \text{ holds} \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathcal{P}(a_\mathrm{s}) \triangleq (\mathrm{G}_\mathrm{s}.\mathsf{op}(a_\mathrm{s}) = \mathsf{W} \wedge \mathrm{G}_\mathrm{s}.\mathsf{loc}(a_\mathrm{s}) = \mathrm{G}_\mathrm{s}.\mathsf{loc}(b_\mathrm{s}) \wedge \mathrm{G}_\mathrm{s}.\mathsf{val}(a_\mathrm{s}) = v)$$

Lemma 19. *Let $\mathrm{G}_\mathrm{t}^{init}$ and $\mathrm{G}_\mathrm{s}^{init}$ be initial execution graphs. Then $\mathcal{R}(\mathrm{G}_\mathrm{s}^{init}, \mathrm{G}_\mathrm{t}^{init}, \mathsf{id}, a_\mathrm{t}, b_\mathrm{t})$ holds.*

Proof. Easy to see that $\mathcal{A}_s = \emptyset$, so the first four conditions of Definition 29 hold. As well as the rest of them, since the sets of po, rf and mo edges are empty for initial execution graphs and $\mathcal{A}_s$ is once again empty. □

Lemma 20. *Let* G_t *and* G_s *be* $\mathcal{M}$ *consistent and ssume* $\mathcal{R}(G_s, G_t, m, b_t, v)$ *holds. If* $G_t \to G'_t$ *holds then there exists* G'_s *and* m' *such that* $G_s \to^* G'_s$ *and* $\mathcal{R}(G'_s, G'_t, m', b_t, v)$ *also hold.*

Proof. We do a case analysis on $G_t \to G'_t$ and use Lemmas 21 to 23 to derive G'_s and $\mathcal{R}(G'_s, G'_t, m', b_t, v)$. □

Lemma 21. *Let* G_t *and* G_s *be* $\mathcal{M}$ *consistent and assume* $\mathcal{R}(G_s, G_t, m, b_t, v)$ *holds. If* $G_t \xrightarrow[exec]{e \notin \{b_t\}} G'_t$ *holds then there exists* G'_s *and* m' *such that* $G_s \xrightarrow[exec]{m'(e)} G'_s$ *and* $\mathcal{R}(G'_s, G'_t, m', b_t, v)$ *also hold.*

Proof. Let us define e' as an event, s.t. $\mathcal{P}_e(e')$ holds:

$$\mathcal{P}_e(e') \triangleq (G_s.\mathsf{op}(e') = G_t.\mathsf{op}(e) \wedge G_s.\mathsf{loc}(e') = G_t.\mathsf{loc}(e) \wedge G_s.\mathsf{ord}(e') = G_t.\mathsf{ord}(e))$$

Therefore, we can construct G'_s and m' as follows:

- $m' = m[e \mapsto e']$
- $G'_s.E = G_s.E \cup \{e'\}$
- $G'_s.\mathsf{tid} = G_s.\mathsf{tid}[e' \mapsto G'_t.\mathsf{tid}(e)]$
- $G'_s.\mathsf{lab} = G_s.\mathsf{lab}[e' \mapsto G'_t.\mathsf{lab}(e)]$
- $G'_s.\mathsf{po} = G_s.\mathsf{po} \cup m \uparrow G'_t.\mathsf{po}; [\{e\}]$
- $G'_s.\mathsf{rf} = G_s.\mathsf{rf} \cup m \uparrow G'_t.\mathsf{rf}; [\{e\} \cap G'_t.R]$
- $G'_s.\mathsf{mo} = G_s.\mathsf{mo} \cup m \uparrow [\{e\} \cap G'_t.W]; G'_t.\mathsf{mo} \cup m \uparrow G'_t.\mathsf{mo}; [\{e\} \cap G'_t.W]$
- $G'_s.\mathsf{rmw} = m \uparrow G'_t.\mathsf{rmw}$

From this construction and Definition 29, it follows immediately that $\mathcal{R}(G'_s, G'_t, m', b_t, v)$ holds.

In case when $\mathcal{A}_s$ is empty, consistency of the source graph, $\mathit{Consistent}(\mathcal{M}, G'_s)$, follows directly from the consistency of the target graph, $\mathit{Consistent}(\mathcal{M}, G'_t)$, due to the monotonicity property Lemma 7. And in the other case, it follows from Lemma 10. □

Lemma 22. *Let* G_t *and* G_s *be* $\mathcal{M}$ *consistent and assume* $\mathcal{R}(G_s, G_t, m, b_t, v)$ *holds. If* $G_t \xrightarrow[exec]{a_t} G'_t$ *holds then there exists* G'_s *and* m' *such that* $G_s \xrightarrow[exec]{a_s} \xrightarrow[exec]{b_s} G'_s$ *holds, with* $a_s \triangleq m'(a_t)$, *and* $\mathcal{R}(G'_s, G'_t, m', b_t, v)$ *also holds.*

Proof. We construct m' and G'_s as follows:

- $m' = m[b_t \mapsto b_s]$
- $G'_s.E = G_s.E \cup \{a_s\} \cup \{b_s\}$
- $G'_s.\mathsf{tid} = G_s.\mathsf{tid}[a_s \mapsto G'_t.\mathsf{tid}(b_t)][b_s \mapsto G'_t.\mathsf{tid}(b_t)]$
- $G'_s.\mathsf{lab} = G_s.\mathsf{lab}[a_s \mapsto \ell][b_s \mapsto G'_t.\mathsf{lab}(a_t)]$
- $G'_s.\mathsf{po} = G_s.\mathsf{po} \cup \mathit{InsertBefore}(m \uparrow G_t.\mathsf{po}, \{b_s\}, \{a_s\})$
- $G'_s.\mathsf{rf} = G_s.\mathsf{rf}$
- $G'_s.\mathsf{mo} = G_s.\mathsf{mo} \cup \mathit{InsertBefore}(m \uparrow G_t.\mathsf{mo}, \mathcal{B}_s, \mathcal{A}_s)$
- $G'_s.\mathsf{rmw} = m \uparrow G'_t.\mathsf{rmw}$

Where ℓ is label for a_s such that:

$$G_s.\mathsf{op}(a_s) = W \wedge G_s.\mathsf{loc}(a_s) = G_s.\mathsf{loc}(b_s) \wedge G_s.\mathsf{val}(a_s) = v$$

From this construction and Definition 29, it follows immediately that $\mathcal{R}(G'_s, G'_t, m', b_t, v)$ holds.

Consistency of the source graph follows from overwrite extensibility Lemma 10 applied to the intermediate source graph we get after adding a_s, which is consistent due to Lemma 7. □

Lemma 23. *Let* $\mathrm{G_t}$ *and* $\mathrm{G_s}$ *be* $\mathcal{M}$ *consistent and* $\mathcal{R}(\mathrm{G_s}, \mathrm{G_t}, \mathsf{m}, b_t, v)$ *holds. If* $\mathrm{G_t} \xrightarrow[re-exec]{C} \mathrm{G_t'}$ *holds then there exists* $\mathrm{G_s'}$ *and* m' *such that* $\mathrm{G_s} \xrightarrow[re-exec]{\mathsf{m}' \uparrow C} \mathrm{G_s'}$ *holds, and* $\mathcal{R}(\mathrm{G_s'}, \mathrm{G_t'}, \mathsf{m}', b_t, v)$ *also holds.*

Proof. We construct m' and $\mathrm{G_s'}$ as follows:

- $\mathsf{m}' = \textbf{if } b_t \in \mathrm{G_t.E} \textbf{ then } \mathsf{id}[b_t \mapsto b_s] \textbf{ else } \mathsf{id}$
- $\mathrm{G_s'.tid} = \mathrm{G_t'.tid} \circ \mathsf{m}$
- $\mathrm{G_s'.lab} = \mathrm{G_t'.lab} \circ \mathsf{m}$
- $\mathrm{G_s'.E} = \mathsf{m} \uparrow \mathrm{G_t'.E} \cup \mathcal{B}_s$
- $\mathrm{G_s'.po} = \mathsf{m} \uparrow \mathrm{G_t'.po} \cup \mathit{InsertBefore}(\mathsf{m} \uparrow \mathrm{G_t.po}, \mathcal{B}_s, \mathcal{A}_s)$
- $\mathrm{G_s'.rf} = \mathsf{m} \uparrow \mathrm{G_t'.rf}$
- $\mathrm{G_s'.mo} = \mathsf{m} \uparrow \mathrm{G_t'.mo} \cup \mathit{InsertBefore}(\mathsf{m} \uparrow \mathrm{G_t.mo}, \mathcal{B}_s, \mathcal{A}_s)$
- $\mathrm{G_s'.rmw} = \mathsf{m} \uparrow \mathrm{G_t'.rmw}$

From this construction and Definition 29, it follows immediately that $\mathcal{R}(\mathrm{G_s'}, \mathrm{G_t'}, \mathsf{m}', b_t, v)$ holds. We will define the components of the re-execution step as follows:

- $C_s \triangleq \uparrow C_t' \textbf{ if } b_t \in C_t \textbf{ then } \{a_t\} \textbf{ else } \emptyset$
- $\mathcal{D}_s \triangleq \mathsf{m} \uparrow \mathcal{D}_t' \textbf{ if } b_t \in \mathcal{D}_t \textbf{ then } \{a_t\} \textbf{ else } \emptyset$
- $\mathsf{f}_s \triangleq \mathsf{m} \circ \mathsf{f}_t \circ \mathsf{m}'^{-1}$

In both cases, the following properties follow immediately from definitions:

- $C_s \subseteq \mathrm{G_s'.E}$
- $\mathcal{D}_s \subseteq C_s$
- $\mathsf{dom}(\mathrm{G.po}; [\mathcal{D}_s]) \subseteq \mathcal{D}_s$
- $\mathit{WellFormed}(\mathrm{G}|_{\mathcal{D}_s}, \mathrm{G_s'}, C_s)$
- $\mathit{CommitEmbedded}(\mathrm{G_s}, \mathrm{G_s'}, C_s, \mathsf{f}_s)$

The property $\mathsf{dom}(\mathrm{G_s'.rpo}; [\mathrm{G_s'.E} \setminus \mathcal{D}_s]) \subseteq \mathcal{D}_s$ follows from the fact that $\mathrm{G_s'.rpo} \subseteq \mathrm{G_t'.rpo}$ and the respective property of the target graph.

To prove $\mathit{ThreadOrderedUEvents}(\mathrm{G}', C_s, \prec_{\mathsf{tid}})$ we use the following observation:

$$\begin{aligned}\mathrm{G_s'.vf}; \mathrm{G_s'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_s] = [\mathrm{G_s'.W}]; \mathrm{G_s'.rf}; \mathrm{G_s'.hb}^?; \mathrm{G_s'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_s] \subseteq \\ \subseteq \mathsf{m}' \uparrow ([\mathrm{G_t'.W}]; \mathrm{G_t'.rf}; \mathrm{G_t'.hb}^?; \mathrm{G_t'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_t]) \cup \\ \cup [\mathcal{A}]; \mathrm{G_s'.po}; \mathsf{m}' \uparrow (\mathrm{G_t'.hb}^?; \mathrm{G_t'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_t]) \cup\end{aligned}$$

- For the first disjunct, we notice that:

$$\mathsf{m}' \uparrow ([\mathrm{G_t'.W}]; \mathrm{G_t'.rf}; \mathrm{G_t'.hb}^?; \mathrm{G_t'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_t]) = \mathsf{m}' \uparrow (\mathrm{G_t'.vf}; \mathrm{G_t'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_t])$$

- For the second disjunct we can construct a similar path in $\mathrm{G_t'}$ to b_t, because a_s and b_s have the same set of outgoing hb edges (except $\langle a_s, b_s \rangle$ edge):

$$\begin{aligned}[\mathcal{A}]; \mathrm{G_s'.po}; \mathsf{m}' \uparrow (\mathrm{G_t'.rf}^?; \mathrm{G_t'.hb}^?; \mathrm{G_t'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_t]) \subseteq \\ \mathsf{m}' \uparrow ([\mathcal{B}]; \mathrm{G_t'.po}; \mathrm{G_t'.hb}^?; \mathrm{G_t'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_t]) \subseteq \\ \mathsf{m} \uparrow (\mathrm{G_t'.vf}; \mathrm{G_t'.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_t])\end{aligned}$$

So in the result, we get the desired property:

$$\mathrm{G}'_\mathrm{s}.\mathsf{vf}; \mathrm{G}'_\mathrm{s}.\underset{\mathsf{tid}}{=}; [\mathcal{U}_s] \subseteq \mathsf{m} \uparrow (\mathrm{G}'_\mathrm{t}.\mathsf{vf}; \mathrm{G}'_\mathrm{t}.\underset{\mathsf{tid}}{=}; [\mathcal{U}_t]) \subseteq \mathsf{m} \uparrow (\mathrm{G}'_\mathrm{t}.\mathsf{tid} \downarrow \preceq_{\mathsf{tid}}) = \mathrm{G}'_\mathrm{s}.\mathsf{tid} \downarrow \preceq_{\mathsf{tid}}$$

In case when $\mathcal{B}$ is empty, consistency of the source graph, $\mathit{Consistent}(\mathcal{M}, \mathrm{G}'_\mathrm{s})$, follows directly from the consistency of the target graph, $\mathit{Consistent}(\mathcal{M}, \mathrm{G}'_\mathrm{t})$, due to monotonicity property Lemma 7. In the other case, it follows from overwrite extensibility Lemma 10.

It is left to show that:

$$\langle \mathrm{G}'_\mathrm{s}, C_s \rangle \vdash \mathrm{G}_\mathrm{s}|_{\mathcal{D}_s} \Rightarrow^* \mathrm{G}'_\mathrm{s}$$

For this, we can apply Lemma 1 with ℓ inherited from the target graph, adding the a_s event at the end of the list if needed. □

Lemma 24. *Suppose that* $\mathcal{R}(\mathrm{G}_\mathrm{s}, \mathrm{G}_\mathrm{t}, \mathsf{m}, b_\mathrm{t}, v)$ *holds. Then* G_s *is also* $\mathcal{M}$ *consistent and terminal, and moreover* $\mathcal{M}(\mathrm{G}_\mathrm{s}, \mathrm{G}_\mathrm{t}, \mathsf{m}, a_\mathrm{t})$ *holds.*

Proof. $\mathcal{M}$ consistency follows from overwrite extensibility property Lemma 10, since $\mathcal{A}$ is not empty, and all the prerequisites are met from the $\mathcal{R}(\mathrm{G}_\mathrm{s}, \mathrm{G}_\mathrm{t}, \mathsf{m}, b_\mathrm{t}, v)$ definition. $\mathcal{M}(\mathrm{G}_\mathrm{s}, \mathrm{G}_\mathrm{t}, \mathsf{m}, a_\mathrm{t})$ holds by the same reasoning. □

Theorem 4.

$$\forall \mathbb{P}_{\mathsf{src}}.\ \mathsf{elim}(\mathbb{P}_{\mathsf{src}}, \mathbb{P}_{\mathsf{tgt}}) \implies \\ \forall \mathrm{G}_\mathrm{t}.\ \mathrm{G}_{\mathsf{init}} \to^* \mathrm{G}_\mathrm{t}.\ \exists \mathrm{G}_\mathrm{s}.\mathrm{G}_{\mathsf{init}} \to^* \mathrm{G}_\mathrm{s}.\ \mathsf{Behavior}(\mathrm{G}_\mathrm{s}) = \mathsf{Behavior}(\mathrm{G}_\mathrm{t})$$

Proof. We do induction on the construction steps: $\mathrm{G}_{\mathsf{init}} \to^* \mathrm{G}_\mathrm{t}$. We use Lemma 19 to show that initial graphs $\mathrm{G}^{\mathsf{init}}_\mathrm{t}$ and $\mathrm{G}^{\mathsf{init}}_\mathrm{s}$ are in simulation relation $\mathcal{R}$. Then using Lemma 20 we show that each step of the target graph construction $\mathrm{G}_\mathrm{t} \to \mathrm{G}'_\mathrm{t}$ can be simulated by the source graph construction step $\mathrm{G}_\mathrm{s} \to \mathrm{G}'_\mathrm{s}$. Finally, we use lemma Lemma 24 to derive that the final graph G_s is $\mathcal{M}$ consistent. □

E.2 Reordering

We prove the soundness of the reordering transformation on the level of the execution graphs by reordering two adjacent events.

Definition 30. Let G_s and G_t be the source and the target execution graphs respectively, and let $\mathsf{m} : \mathrm{G}_\mathrm{t}.\mathrm{E} \to \mathrm{G}_\mathrm{s}.\mathrm{E}$ be an event mapping function. We say that events $a, b \in \mathrm{G}_\mathrm{t}.\mathrm{E}$ form a reordering pair

$$a \cdot b \rightsquigarrow b \cdot a$$

if the following conditions are met:

- $\mathrm{G}_\mathrm{t}.\mathsf{loc}(a) \neq \mathrm{G}_\mathrm{t}.\mathsf{loc}(b)$
- $\mathrm{G}_\mathrm{s}.\mathsf{lab}(\mathsf{m}(a)) = \mathrm{G}_\mathrm{t}.\mathsf{lab}(a)$
- $\mathrm{G}_\mathrm{s}.\mathsf{lab}(\mathsf{m}(b)) = \mathrm{G}_\mathrm{t}.\mathsf{lab}(b)$
- $\mathsf{m} \uparrow \langle b, a \rangle \in \mathrm{G}_\mathrm{s}.\mathsf{po}_{\mathsf{imm}}$
- $\langle b, a \rangle \in \mathrm{G}_\mathrm{t}.\mathsf{po}_{\mathsf{imm}}$
- $\langle b, a \rangle \notin \mathrm{G}_\mathrm{t}.\mathsf{rpo}$

Definition 31. We define a relation $\mathcal{M}(\mathrm{G}_\mathrm{s}, \mathrm{G}_\mathrm{t}, \mathsf{m}, a, b)$ asserting that the target graph G_t is obtained as a result of applying the reordering transformation to the source graph G_s, where $\mathsf{m} : \mathrm{G}_\mathrm{t}.\mathrm{E} \to \mathrm{G}_\mathrm{s}.\mathrm{E}$ is an event mapping function, a and b are reordered events in G_t:

- m is injective
- $\mathrm{G}_\mathrm{s}.\mathrm{E} = \mathsf{m} \uparrow \mathrm{G}_\mathrm{t}.\mathrm{E}$
- $\mathrm{G}_\mathrm{s}.\mathsf{tid} \circ \mathsf{m} = \mathrm{G}_\mathrm{t}.\mathsf{tid}$

- $G_s.\mathsf{lab} \circ \mathsf{m} = G_t.\mathsf{lab}$
- $G_s.\mathsf{po} = \mathsf{m} \uparrow \mathit{Swap}(R, \{b\}, \{a\})$
- $G_s.\mathsf{rf} = \mathsf{m} \uparrow G_t.\mathsf{rf}$
- $G_s.\mathsf{mo} = \mathsf{m} \uparrow G_t.\mathsf{mo}$
- $G_s.\mathsf{rmw} = \mathsf{m} \uparrow G_t.\mathsf{rmw}$

Definition 32. $\mathcal{R}(G_s, G_t, \mathsf{m}, a_t, b_t)$ — is a *simulation relation* for the reordering transformation where G_s is a source graph, G_t is a target graph, and $\mathsf{m} : G_t.\mathsf{E} \to G_s.\mathsf{E}$ is an event mapping function, and a_t and b_t are reordered events in G_t.

- m is injective
- $G_s.\mathsf{E} = \mathsf{m} \uparrow G_t.\mathsf{E} \cup \mathcal{A}_s$
- $G_s.\mathsf{tid} \circ \mathsf{m} = G_t.\mathsf{tid}$
- $G_s.\mathsf{lab} \circ \mathsf{m} = G_t.\mathsf{lab}$
- $G_s.\mathsf{po} = \mathsf{m} \uparrow (\mathit{Swap}(R, \{b_t\}, \{a_t\}) \cup \mathsf{m} \uparrow (\mathsf{dom}(G_t.\mathsf{po}; [b_t]) \times \{\mathcal{A}\}))$
- $G_s.\mathsf{rf} = \mathsf{m} \uparrow G_t.\mathsf{rf} \cup G_s.\mathtt{srf}; [\mathcal{A}_s \cap \mathsf{R}]$
- $G_s.\mathsf{mo} = \mathsf{m} \uparrow G_t.\mathsf{mo} \cup (G_s.\mathsf{W}_\ell \setminus \mathcal{A}_s) \times (\mathcal{A}_s \cap \mathsf{W})$
- $G_s.\mathsf{rmw} = \mathsf{m} \uparrow G_t.\mathsf{rmw}$

where:

$$\ell \triangleq G_t.\mathsf{loc}(a_t)$$

$$b_s \triangleq \mathsf{m}(b_t)$$

$$\mathcal{A}_s \triangleq \begin{cases} \{a'_s\} & \text{if } a_t \notin G_t.\mathsf{E} \wedge b_t \in G_t.\mathsf{E}\,, \quad \text{where } a'_s \text{ is an event, } \textit{s.t. } \mathcal{P}(a'_s) \text{ holds} \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathcal{P}(a'_s) \triangleq (G_s.\mathsf{op}(a'_s) = G_t.\mathsf{op}(a_t) \wedge G_s.\mathsf{loc}(a'_s) = G_t.\mathsf{loc}(a_t) \wedge G_s.\mathsf{ord}(a'_s) = G_t.\mathsf{ord}(a_t))$$

Lemma 25. *Let G_t^{init} and G_s^{init} be initial execution graphs. Then $\mathcal{R}(G_s^{init}, G_t^{init}, \mathsf{id}, a_t, b_t)$ holds.*

Proof. Easy to see that $\mathcal{A}_s = \emptyset$, so the first four conditions of Definition 32 hold. As well as the rest of them, since the sets of $\mathsf{po}_{\mathsf{imm}}$, rf, and mo edges are empty for initial execution graphs and $\mathcal{A}_s$ is once again empty. □

Lemma 26. *Let G_t and G_s be $\mathcal{M}$ consistent and assume $\mathcal{R}(G_s, G_t, \mathsf{m}, a_t, b_t)$ holds. If $G_t \to G'_t$ holds then there exists G'_s and m' such that $G_s \to^* G'_s$ and $\mathcal{R}(G'_s, G'_t, \mathsf{m}', a_t, a_t)$ also hold.*

Proof. We do a case analysis on $G_t \to G'_t$ and use Lemmas 27 to 30 to derive G'_s and $\mathcal{R}(G'_s, G'_t, \mathsf{m}, a_t, b_t)$. □

Lemma 27. *Let G_t and G_s be $\mathcal{M}$ consistent and $\mathcal{R}(G_s, G_t, \mathsf{m}, a_t, b_t)$ holds. If $G_t \xrightarrow[exec]{e_t \notin \{a_t, b_t\}} G'_t$ holds then there exists G'_s and m' such that $G_s \xrightarrow[exec]{\mathsf{m}'(e_t)} G'_s$ and $\mathcal{R}_{rw}(G'_s, G'_t, \mathsf{m}', a_t, b_t)$ also hold.*

Proof. Let us define e_s as an event, s.t. $\mathcal{P}_e(e_s)$ holds:

$$\mathcal{P}_e(e_s) \triangleq (G_s.\mathsf{op}(e_s) = G_t.\mathsf{op}(e_t) \wedge G_s.\mathsf{loc}(e_s) = G_t.\mathsf{loc}(e_t) \wedge G_s.\mathsf{ord}(e_s) = G_t.\mathsf{ord}(e_t))$$

Therefore, we can construct G'_s and m' as follows:

- $\mathsf{m}' = \mathsf{m}[e_t \mapsto e_s]$
- $G'_s.\mathsf{E} = G_s.\mathsf{E} \cup \{e_s\}$
- $G'_s.\mathsf{tid} = G_s.\mathsf{tid}[e_s \mapsto G'_t.\mathsf{tid}(e_t)]$

- $G'_s.lab = G_s.lab[e_s \mapsto G'_t.lab(e_t)]$
- $G'_s.po = G_s.po \cup m \uparrow (dom(G_t.po; [\{e_t\}]) \times \{e_s\}))$
- $G'_s.rf = G_s.rf \cup m \uparrow G'_t.rf; [\{e_t\} \cap G'_t.R]$
- $G'_s.mo = G_s.mo \cup m \uparrow ([W_t]; G'_t.mo) \cup m(\uparrow G'_t.mo); [W_t] \cup (m \uparrow W_t) \times (\mathcal{A}_s \cap G_s.W_{loc(w)})$ where $W_t = \{e_t\} \cap G'_t.W$).
- $G'_s.rmw = G_s.rmw \cup$ **if** IsExclusiveWrite$(e_t) \uparrow G'_t.rmw; [\{e_t\}]$ **else** $\emptyset$

From this construction and Definition 32, it follows immediately that $\mathcal{R}(G'_s, G'_t, m', a_t, b_t)$ holds. In case when $\mathcal{A}_s$ is empty, consistency of the source graph, $Consistent(\mathcal{M}, G'_s)$, follows directly from the consistency of the target graph, $Consistent(\mathcal{M}, G'_t)$, due to monotonicity property Lemma 7. And in the other case, it follows from Lemma 8 if a_t is a read event, and from Lemma 9 if a_t is a write event. We apply these lemmas for graphs G'_t and G'_s, and the event a'_s. □

Lemma 28. *Let* G_t *and* G_s *be* $\mathcal{M}$ *consistent and* $\mathcal{R}(G_s, G_t, m, a_t, b_t)$ *holds. If* $G_t \xrightarrow[exec]{b_t} G'_t$ *holds then there exists* G'_s *and* m' *such that* $G_s \xrightarrow[exec]{a'_s} \xrightarrow[exec]{b_s} G'_s$ *holds, with* $b_s \triangleq m'(b_t)$*, and* $\mathcal{R}(G'_s, G'_t, m', a, b)$ *also holds.*

Proof. We construct m' and G'_s as follows:

- $m' = m[b_t \mapsto b_s]$
- $G'_s.E = G_s.E \cup \{a'_s\} \cup \{b_s\}$
- $G'_s.tid = G_s.tid[b_s \mapsto G'_t.tid(b_t)]$
- $G'_s.lab = G_s.lab[b_s \mapsto G'_t.lab(b_t)]$
- $G'_s.po = G_s.po \cup m \uparrow (dom(G_t.po; [\{b_t\}])) \times \{a'_s\} \cup \{\langle b_s, a'_s \rangle\}$
- $G'_s.rf = G_s.rf \cup m \uparrow G'_t.rf; [\{b_t\} \cap G'_t.R] \cup \{w_t\} \times \{a'_s\}$
- $G'_s.mo = G_s.mo \cup m \uparrow [\{b_t\} \cap G'_t.W]; G'_t.mo \cup m \uparrow G'_t.mo; [\{b_t\} \cap G'_t.W] \cup G_s.W \times \{a'_s\}$
- $G'_s.rmw = m \uparrow G'_t.rmw$

Where w_s is the $G_s.mo$ maximal write event to the same location as $G_s.loc(a'_s)$ from the view-front $G_s.vf$ of the $G_s.po$ maximal event of the thread $G_s.tid(a'_s)$. This ensures that we have an `srf`-edge from w_s to a'_s: $G_s.rf; [\{a'_s\}] = G_s.srf; [\{a'_s\}]$.

From this construction and Definition 32, it follows immediately that $\mathcal{R}(G'_s, G'_t, m', a_t, b_t)$ holds.

Consistency of the source graph, $Consistent(\mathcal{M}, G'_s)$, follows from Lemma 8 if a_t is a read event, and from Lemma 9 if a_t is a write event. We apply these lemmas for graphs G'_t and G'_s, and the event a'_s. □

Lemma 29. *Let* G_t *and* G_s *be* $\mathcal{M}$ *consistent and* $\mathcal{R}(G_s, G_t, m, a_t, b_t)$ *holds. If* $G_t \xrightarrow[exec]{a_t} G'_t$ *holds there exists* G'_s*,* m'*, and* C *such that* $G_s \xrightarrow[re-exec]{C} G'_s$ *and* $\mathcal{R}(G'_s, G'_t, m', a_t, b_t)$ *hold.*

Proof. We construct $m' \triangleq m$ and G'_s such that:

- $m' = m[a_t \mapsto a_s]$
- $G'_s.E = (G_s.E \setminus \{a'_s\}) \cup \{a_s\}$
- $G'_s.tid = G_s.tid[a'_s \mapsto \bot][a_s \mapsto G'_t.tid(a)]$
- $G'_s.lab = G_s.lab[a'_s \mapsto \bot][a_s \mapsto G'_t.lab(a)]$
- $G'_s.po = m \uparrow Swap(R, \{b\}, \{a\})$
- $G'_s.rf = G_s.rf; [G_s.E \setminus \{a'_s\}] \cup m \uparrow G'_t.rf; [\{a_t\} \cap G'_t.R]$
- $G'_s.mo = G_s.mo|_{G_s.E \setminus \{a'_s\}} \cup m \uparrow [\{a_t\} \cap G'_t.W]; G'_t.mo \cup m \uparrow G'_t.mo; [\{a_t\} \cap G'_t.W]$
- $G'_s.rmw = m \uparrow G'_t.rmw$

From this construction and Definition 32, it follows immediately that $\mathcal{R}(G'_s, G'_t, m', a_t, b_t)$ holds.

We set $\mathcal{C}$, $\mathcal{D}$, and f as follows:

- $\mathcal{C} \triangleq \mathrm{G}'_\mathrm{s}.\mathrm{E} \setminus \{a_\mathrm{s}\}$
- $\mathcal{D} \triangleq \mathrm{G}_\mathrm{s}.\mathrm{E} \setminus \{a'_\mathrm{s}, b_\mathrm{s}\}$
- $\mathrm{f} \triangleq \mathrm{id}$

The following properties follow immediately from these definitions:

- $\mathcal{C} \subseteq \mathrm{G}'_\mathrm{s}.\mathrm{E}$
- $\mathcal{D} \subseteq \mathrm{id} \upharpoonright \mathcal{C} = \mathcal{C}$
- $\mathrm{dom}(\mathrm{G.po}; [\mathcal{D}]) \subseteq \mathcal{D}$
- $\mathit{WellFormed}(\mathrm{G}|_{\mathcal{D}}, \mathrm{G}', \mathcal{C})$
- $\mathit{CommitEmbedded}(\mathrm{G}, \mathrm{G}', \mathcal{C}, \mathrm{f})$

Because we re-execute only a single thread, the predicate $\mathit{ThreadOrderedUEvents}(\mathrm{G}', \mathcal{C}, \prec_{\mathsf{tid}})$ holds due to Corollary 2.

Consistency of the source graph, $\mathit{Consistent}(\mathcal{M}, \mathrm{G}'_\mathrm{s})$, follows directly from the consistency of the target graph, $\mathit{Consistent}(\mathcal{M}, \mathrm{G}'_\mathrm{t})$, due to the monotonicity property Definition 18.

It is left to show that: $\langle \mathrm{G}'_\mathrm{s}, \mathcal{C}\rangle \vdash \mathrm{G}_\mathrm{s}|_{\mathcal{D}} \Rightarrow^* \mathrm{G}'_\mathrm{s}$. Here we note that we only re-execute a single thread, and thus the re-execution is possible due to Corollary 3.

□

Lemma 30. *Let* G_t *and* G_s *be* $\mathcal{M}$ *consistent and* $\mathcal{R}(\mathrm{G}_\mathrm{s}, \mathrm{G}_\mathrm{t}, \mathrm{m}, a_\mathrm{t}, b_\mathrm{t})$ *holds. If* $\mathrm{G}_\mathrm{t} \xrightarrow[re-exec]{\mathcal{C}} \mathrm{G}'_\mathrm{t}$ *holds then there exists* G'_s *and* m' *such that* $\mathrm{G}_\mathrm{s} \xrightarrow[re-exec]{\mathrm{m}'\upharpoonright\mathcal{C}} \mathrm{G}'_\mathrm{s}$ *holds, and* $\mathcal{R}(\mathrm{G}'_\mathrm{s}, \mathrm{G}'_\mathrm{t}, \mathrm{m}', a_\mathrm{t}, b_\mathrm{t})$ *also holds.*

Proof. Consider the cases.

(A.) $b_t, a_t \notin \mathrm{G}'_\mathrm{t}.\mathrm{E}_t$ or $b_t, a_t \in \mathrm{G}'_\mathrm{t}.\mathrm{E}_t$. In this case we set:

- $\mathrm{m}' = \mathrm{id}$
- $\mathrm{G}'_\mathrm{s}.\mathrm{E} = \mathrm{G}'_\mathrm{t}.\mathrm{E}$
- $\mathrm{G}'_\mathrm{s}.\mathrm{po} = \mathrm{m} \upharpoonright (\mathit{Swap}(R, \{b_t\}, \{a_t\}) \cup \mathrm{m} \upharpoonright (\mathrm{dom}(\mathrm{G_t.po}; [b_t]) \times \{\mathcal{A}\}))$
- $\mathrm{G}'_\mathrm{s}.\mathit{Rel} = \mathrm{m} \upharpoonright \mathrm{G}'_\mathrm{t}.\mathit{Rel}$ for $\mathit{Rel} \in \{\mathsf{rf}, \mathsf{mo}, \mathsf{rmw}\}$
- $\mathcal{C}_s \triangleq \mathrm{m}' \upharpoonright \mathcal{C}_t$
- $\mathcal{D}_s \triangleq \mathrm{m}' \upharpoonright \mathcal{D}_t$
- $\mathrm{f}_s \triangleq \mathrm{f}_t$
- $\mathrm{f}_s \triangleq \mathrm{m} \circ \mathrm{f}_t \circ \mathrm{m}'^{-1}$

To show that the re-execution sequence $\langle \mathrm{G}'_\mathrm{s}, \mathcal{C}\rangle \vdash \mathrm{G}_\mathrm{s}|_{\mathcal{D}} \Rightarrow^* \mathrm{G}'_\mathrm{s}$. despite the reordering of one program edge, we first apply Lemma 2 to sort the list ℓ_t, and then apply Lemma 1 to the list ℓ_s which is the same list as ℓ_t, except b_s and a_s being possibly reordered.

The consistency of G'_s follows from Lemma 7.

Almost all of the other constraints of the re-execution step hold trivially, as well as simulation relation $\mathcal{R}(\mathrm{G}'_\mathrm{s}, \mathrm{G}'_\mathrm{t}, \mathrm{m}', a_\mathrm{t}, b_\mathrm{t})$. Indeed, in this case both graphs are almost equal except potentially one program order edge being swapped, but most of the constraints are invariant under this change.

As such, in this case $\mathrm{G}_\mathrm{s} \xrightarrow[re-exec]{\mathrm{m}'\upharpoonright\mathcal{C}} \mathrm{G}'_\mathrm{s}$ and $\mathcal{R}(\mathrm{G}'_\mathrm{s}, \mathrm{G}'_\mathrm{t}, \mathrm{m}', a_\mathrm{t}, b_\mathrm{t})$ also hold.

(B.) $b_t \in \mathrm{G}'_\mathrm{t}.\mathrm{E}_t, a_t \notin \mathrm{G}'_\mathrm{t}.\mathrm{E}_t$ and $b_t \in \mathrm{G}'_\mathrm{t}.\mathcal{T}$. In this case, we construct m' and G'_s as follows (a'_s and $\mathcal{A}$ are defined similarly as in Definition 32):

- $\mathrm{m}' = \lambda e.\, \mathbf{if}\ e \neq a'_s \mathbf{then}\ e\ \mathbf{else}\ \bot$
- $\mathrm{G}'_\mathrm{s}.\mathrm{E} = \mathrm{m} \upharpoonright \mathrm{G}'_\mathrm{t}.\mathrm{E} \cup \mathcal{A}_\mathrm{s}$
- $\mathrm{G}'_\mathrm{s}.\mathrm{tid} = \mathrm{G}'_\mathrm{t}.\mathrm{tid} \circ \mathrm{m}$
- $\mathrm{G}'_\mathrm{s}.\mathrm{lab}{=}\mathrm{G}'_\mathrm{t}.\mathrm{lab} \circ \mathrm{m}$

- $\mathrm{G_s.po} = \mathsf{m} \uparrow (\mathit{Swap}(R, \{b_t\}, \{a_t\}) \cup \mathsf{m} \uparrow (\mathrm{dom}(\mathrm{G_t.po}; [b_t]) \times \{\mathcal{A}\}))$
- $\mathrm{G'_s.rf} = \mathsf{m} \uparrow \mathrm{G'_t.rf} \cup \mathrm{G'_s.srf}; [\mathcal{A}_\mathrm{s} \cap \mathrm{R}]$
- $\mathrm{G'_s.mo} = \mathsf{m} \uparrow \mathrm{G'_t.mo} \cup \mathrm{G'_s.W}_\ell \times (\mathcal{A}_\mathrm{s} \cap \mathrm{W})$
- $\mathrm{G'_s.rmw} = \mathsf{m} \uparrow \mathrm{G'_t.rmw}$

From this construction and Definition 32, it follows immediately that $\mathcal{R}(\mathrm{G'_s}, \mathrm{G'_t}, \mathsf{m}', a_\mathrm{t}, b_\mathrm{t})$ holds. Next we set:

- $C_s \triangleq \mathsf{m} \uparrow C_t$
- $\mathcal{D}_s \triangleq \mathsf{m} \uparrow \mathcal{D}_t$
- $\mathsf{f}_s \triangleq \mathsf{m} \circ \mathsf{f}_t \circ \mathsf{m}'^{-1}$

The following properties follow immediately from definitions, as committed and determined sets coincide in the source and target graphs:

- $C_s \subseteq \mathrm{G'_s.E}$
- $\mathcal{D}_s \subseteq C_s$
- $\mathrm{dom}(\mathrm{G.po}; [\mathcal{D}_s]) \subseteq \mathcal{D}_s$
- $\mathit{WellFormed}(\mathrm{G}|_{\mathcal{D}_s}, \mathrm{G'_s}, C_s)$
- $\mathit{CommitEmbedded}(\mathrm{G_s}, \mathrm{G'_s}, C_s, \mathsf{f}_s)$

The property $\mathrm{dom}(\mathrm{G'_s.rpo}; [\mathrm{G'_s.E} \setminus \mathcal{D}_s]) \subseteq \mathcal{D}_s$ follows from the fact that $\mathrm{G'_s.rpo} \subseteq \mathrm{G'_t.rpo}$ and the respective property of the target graph.

The consistency of the source graph follows from Lemma 8 if a_t is a read event, and from Lemma 9 if a_t is a write event.

To prove $\mathit{ThreadOrderedUEvents}(\mathrm{G}', C, \prec_{\mathsf{tid}})$ we use the following observation:

$$\begin{aligned}
\mathrm{G'_s.vf}; \mathrm{G'_s.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_s] = [\mathrm{G'_s.W}]; \mathrm{G'_s.rf}^?; \mathrm{G'_s.hb}^?; \mathrm{G'_s.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_s] \subseteq \\
\subseteq \mathsf{m}' \uparrow ([\mathrm{G'_s.W}]; \mathrm{G'_t.rf}^?; \mathrm{G'_t.hb}^?; \mathrm{G'_s.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_s] \setminus \{\langle b_t, a_t \rangle\}; \mathrm{G'_t.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_t]) \cup \\
\cup \mathrm{G'_s.rf}; \mathrm{G'_s.hb}; [\mathcal{A}_\mathrm{s}]; \mathrm{G'_s.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_s] \cup \\
\cup \mathrm{G'_s.srf}; [\mathcal{A}_\mathrm{s}]; \mathrm{G'_s.rhb}; \mathrm{G'_s.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_s] \cup \\
\cup \mathrm{G'_s.srf}; [\mathcal{A}_\mathrm{s}]; \mathrm{G'_s.po}^?; \mathrm{G'_s.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_s]
\end{aligned}$$

- For the first disjunct, we notice that:

$$\mathsf{m}' \uparrow ([\mathrm{G'_t.W}]; \mathrm{G'_t.rf}; \mathrm{G'_t.hb}^? \setminus \{\langle b_t, a_t \rangle\}; \mathrm{G'_t.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_t]) = \mathsf{m}' \uparrow \mathrm{G'_t.vf}; \mathrm{G'_t.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_t]$$

- For the second disjunct we can construct a similar path in $\mathrm{G'_t}$ to b_t, because a'_s and b_s have the same set of incoming hb edges (except $\langle a'_s, b_s \rangle$ edge):

$$\begin{aligned}
[\mathrm{G'_s.W}]; \mathrm{G'_s.rf}; \mathrm{G'_s.hb}; [\mathcal{A}_\mathrm{s}]; \mathrm{G'_s.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_s] = \\
= \mathsf{m}' \uparrow [\mathrm{G'_s.W}]; \mathrm{G'_t.rf}; \mathrm{G'_t.hb}; [b_t]; \mathrm{G'_t.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_t] \subseteq \\
\subseteq \mathsf{m}' \uparrow (\mathrm{G'_t.vf}; \mathrm{G'_t.}\underset{\mathsf{tid}}{=}; [\mathcal{U}_t])
\end{aligned}$$

- For the third disjunct, we notice that $[\mathcal{A}_\mathrm{s}]; \mathrm{G'_s.rhb} = \emptyset$ since a'_s is a rhb maximal event; and thus the whole disjunct is empty.
- For the last case of we first prove the following statement:

$$\mathrm{G'_s.rhb} \subseteq \mathrm{G'_s.po} \cup [\mathcal{D}_s]; \mathrm{G'_s.rhb} \cup \mathrm{G'_s.sw}; \mathrm{G'_s.rhb}^?$$

by using the fact that $\mathrm{rhb} \subseteq \mathrm{po} \cup \mathrm{po}^?; \mathrm{sw}; \mathrm{rhb}^?$ and also property $\mathrm{po}; \mathrm{sw} = \mathrm{rpo}; \mathrm{sw}$ (Proposition 1), and the fact that $\mathrm{dom}(\mathrm{G'_s.rpo}; [\mathrm{G'_s.E} \setminus \mathcal{D}_s]) \subseteq \mathcal{D}_s$.

Using that $\mathtt{srf} \subseteq \mathtt{vf}; \mathrm{po}$ we next arrive at the following equation:

$$\mathrm{G}'_{\mathrm{s}}.\mathtt{srf}; [\mathcal{A}_{\mathrm{s}}]; \mathrm{G}'_{\mathrm{s}}.\mathrm{po}^{?} \subseteq \mathrm{m}' \upharpoonright \mathrm{G}'_{\mathrm{t}}.\mathtt{vf}; \mathrm{G}'_{\mathrm{s}}.\mathrm{po}; [\mathcal{A}]; (\mathcal{A} \times \{b_t\})^{?} \subseteq \mathrm{m}' \upharpoonright (\mathrm{G}'_{\mathrm{t}}.\mathtt{vf}; \mathrm{G}'_{\mathrm{t}}.\underset{\mathtt{tid}}{=}; [\{b_t\}])$$

So in the result, we get the desired property:

$$\mathrm{m}' \upharpoonright (\mathrm{G}'_{\mathrm{s}}.\mathtt{vf}; \mathrm{G}'_{\mathrm{s}}.\underset{\mathtt{tid}}{=}; [\mathcal{U}_{\mathrm{s}}]) \subseteq \mathrm{m}' \upharpoonright (\mathrm{G}'_{\mathrm{t}}.\mathtt{vf}; \mathrm{G}'_{\mathrm{t}}.\underset{\mathtt{tid}}{=}; [\mathcal{U}_t]) \subseteq \mathrm{m}' \upharpoonright (\mathrm{G}'_{\mathrm{t}}.\mathrm{tid} \downarrow \preceq_{\mathtt{tid}}) = \mathrm{G}'_{\mathrm{s}}.\mathrm{tid} \downarrow \preceq_{\mathtt{tid}}$$

To show that:

$$\langle \mathrm{G}'_{\mathrm{s}}, C_s \rangle \vdash \mathrm{G}_{\mathrm{s}}|_{\mathcal{D}_s} \Rightarrow^{*} \mathrm{G}'_{\mathrm{s}}$$

we first apply Lemma 2 to sort ℓ_t, and then apply Lemma 1 to the list ℓ_s which is the same list as ℓ_t but with an additional event a'_s inserted before b_t.

As such, in this case $\mathrm{G}_{\mathrm{s}} \xrightarrow[re-exec]{\mathrm{m}' \upharpoonright C} \mathrm{G}'_{\mathrm{s}}$ and $\mathcal{R}(\mathrm{G}'_{\mathrm{s}}, \mathrm{G}'_{\mathrm{t}}, \mathrm{m}', a_{\mathrm{t}}, b_{\mathrm{t}})$ also hold.

(C.) $b_t \in \mathrm{G}'_{\mathrm{t}}.\mathrm{E}_t, a_t \notin \mathrm{G}'_{\mathrm{t}}.\mathrm{E}_t$ and $b_t \notin \mathrm{G}'_{\mathrm{t}}.\mathcal{T}$.

In this case, it has to be that $b_t \in \mathcal{D}_t$, and thus $b_t \in \mathrm{G}_{\mathrm{t}}.\mathrm{E}$ and it is the po last event in its thread. It means that $a'_s \in \mathrm{G}_{\mathrm{s}}.\mathrm{E}$.

We then show that it is possible to perform the re-execution: $\mathrm{G}_{\mathrm{s}} \xrightarrow[re-exec]{C_s} \mathrm{G}'_{\mathrm{s}}$ where:

- $\mathrm{m}' = \mathrm{id}$
- $\mathrm{G}'_{\mathrm{s}}.\mathrm{E} = \mathrm{m} \upharpoonright \mathrm{G}'_{\mathrm{t}}.\mathrm{E} \cup \{a'_s\}$
- $\mathrm{G}'_{\mathrm{s}}.\mathrm{po} = \mathrm{m} \upharpoonright \mathrm{G}'_{\mathrm{t}}.\mathrm{E} \cup \mathrm{m} \upharpoonright (\mathrm{dom}(\mathrm{G}_{\mathrm{t}}.\mathrm{po}; [b_t]) \times \{a'_s\}) \cup \{\langle a'_s, b_t \rangle\}$
- $\mathrm{G}'_{\mathrm{s}}.\mathtt{rf} = \mathrm{m} \upharpoonright \mathrm{G}'_{\mathrm{t}}.\mathtt{rf}|_{\mathrm{G}'_{\mathrm{s}}.\mathrm{E}} \cup \mathrm{G}_{\mathrm{s}}.\mathtt{srf}; [\{a'_s\}]$
- $\mathrm{G}'_{\mathrm{s}}.\mathit{Rel} = \mathrm{m} \upharpoonright \mathrm{G}'_{\mathrm{t}}.\mathit{Rel}|_{\mathrm{G}'_{\mathrm{s}}.\mathrm{E}}$ for $\mathit{Rel} \in \{\mathtt{rf}, \mathtt{mo}, \mathtt{rmw}\}$
- $C_s \triangleq \mathrm{m}' \upharpoonright C_t \cup \{a'_s\}$
- $\mathcal{D}_s \triangleq \mathrm{m}' \upharpoonright \mathcal{D}_t \cup \{a'_s\}$
- $\mathrm{f}_s \triangleq \mathrm{m} \circ \mathrm{f}_t \circ \mathrm{m}'^{-1}$

It is indeed possible to commit a'_s as the write it reads from belongs to C_s. We are using the following equation to demonstrate this:

$$\begin{aligned} \mathtt{srf}_s; [a'_s] &\subseteq (\mathrm{po}_s \cup \mathtt{rf}_s; \mathrm{po}_s \cup \mathtt{rf}_s^{?}; \mathtt{rhb}_s); [a'_s] \subseteq \\ &\subseteq [\mathcal{D}_s]; \mathrm{po}_s \cup [C_s]; \mathtt{rf}_s; [\mathcal{D}_s]; \mathrm{po}_s \cup [\mathcal{D}_s]; \mathtt{rf}_s^{?}; [\mathcal{D}_s]; \mathtt{rhb}_s \end{aligned}$$

From this construction and Definition 32, it follows immediately that $\mathcal{R}(\mathrm{G}'_{\mathrm{s}}, \mathrm{G}'_{\mathrm{t}}, \mathrm{m}', a_{\mathrm{t}}, b_{\mathrm{t}})$ holds.

The following properties follow immediately from definitions, as committed and determined sets coincide in the source and target graphs:

- $C_s \subseteq \mathrm{G}'_{\mathrm{s}}.\mathrm{E}$
- $\mathcal{D}_s \subseteq C_s$
- $\mathrm{dom}(\mathrm{G}.\mathrm{po}; [\mathcal{D}_s]) \subseteq \mathcal{D}_s$
- $\mathit{WellFormed}(\mathrm{G}|_{\mathcal{D}_s}, \mathrm{G}'_{\mathrm{s}}, C_s)$
- $\mathit{CommitEmbedded}(\mathrm{G}_{\mathrm{s}}, \mathrm{G}'_{\mathrm{s}}, C_s, \mathrm{f}_s)$

The property $\mathrm{dom}(\mathrm{G}'_{\mathrm{s}}.\mathtt{rpo}; [\mathrm{G}'_{\mathrm{s}}.\mathrm{E} \setminus \mathcal{D}_s]) \subseteq \mathcal{D}_s$ follows from the fact that $\mathrm{G}'_{\mathrm{s}}.\mathtt{rpo} \subseteq \mathrm{G}'_{\mathrm{t}}.\mathtt{rpo}$ and the respective property of the target graph.

The consistency of the source graph follows from Lemma 8 if $a'_{\mathrm{s_t}}$ is a read event, and from Lemma 9 if $a'_{\mathrm{s_t}}$ is a write event.

To prove $\mathit{ThreadOrderedUEvents}(G', C, \prec_{tid})$ we use the same observation as in the previous case:

$$\begin{aligned} G'_s.vf; G'_s.\underset{tid}{=}; [\mathcal{U}_s] = [G'_s.W]; G'_s.rf^?; G'_s.hb^?; G'_s.\underset{tid}{=}; [\mathcal{U}_s] &\subseteq \\ &\subseteq m' \uparrow ([G'_t.W]; G'_t.rf^?; G'_t.hb^?; G'_s.\underset{tid}{=}; [\mathcal{U}_s] \setminus \{\langle b_t, a_t\rangle\}; G'_t.\underset{tid}{=}; [\mathcal{U}_t]) \cup \\ &\cup G'_s.rf; G'_s.hb; [\mathcal{A}_s]; G'_s.\underset{tid}{=}; [\mathcal{U}_s] \cup \\ &\cup G'_s.srf; [\mathcal{A}_s]; G'_s.rhb; G'_s.\underset{tid}{=}; [\mathcal{U}_s] \cup \\ &\cup G'_s.srf; [\mathcal{A}_s]; G'_s.po^?; G'_s.\underset{tid}{=}; [\mathcal{U}_s] \end{aligned}$$

The last three disjuncts are empty, since a'_s does not belong to the re-executed threads, *i.e.,* $[\mathcal{A}_s]; G'_s.\underset{tid}{=}; [\mathcal{U}_s] = \emptyset$, because neither do b_t by the assumption of this case: $b_t \notin G'_t.\mathcal{T}$. For the first disjunct, we use the same equations as in the previous case, arriving at:

$$G'_s.vf; G'_s.\underset{tid}{=}; [\mathcal{U}_s] \subseteq m' \uparrow (G'_t.vf; G'_t.\underset{tid}{=}; [\mathcal{U}_t]) \subseteq m' \uparrow (G'_t.\mathrm{tid} \downarrow \preceq_{tid}) = G'_s.\mathrm{tid} \downarrow \preceq_{tid}$$

It is left to show that:

$$\langle G_s, C_s\rangle \vdash G_s \Rightarrow^* G'_s$$

For this, we first apply Lemma 2 to sort ℓ_t, and then apply Lemma 1 to the list ℓ_s which is the same list as ℓ_t but with an additional event a'_s inserted before b_t.

As such, in this case $G_s \xrightarrow[re-exec]{m'\uparrow C} G'_s$ and $\mathcal{R}(G'_s, G'_t, m', a_t, b_t)$ also hold.

□

Lemma 31. *Suppose that $\mathcal{R}(G_s, G_t, m, a_t, b_t)$ holds, and G_t is $\mathcal{M}$ consistent and terminal. Then G_s is also $\mathcal{M}$ consistent and terminal, and moreover $\mathcal{M}(G_s, G_t, m, a_t, b_t)$ holds.*

Proof. $\mathcal{M}$ consistency follows from monotonicity property Definition 18, since $\mathcal{A}_s$ is empty, and all the prerequisites are met from the $\mathcal{R}(G_s, G_t, m, a_t, b_t)$ definition. $\mathcal{M}(G_s, G_t, m, a_t, b_t)$ holds by the same reasoning. □

Theorem 5.

$$\begin{aligned} &\forall \mathbb{P}_{src}.\ \mathrm{reord}(\mathbb{P}_{src}, \mathbb{P}_{tgt}) \implies \\ &\quad \forall G_t.\ G_{init} \to^* G_t.\ \exists G_s.G_{init} \to^* G_s.\ \mathrm{Behavior}(G_s) = \mathrm{Behavior}(G_t) \end{aligned}$$

Proof. We do induction on the construction steps: $G_{init} \to^* G_t$. We use Lemma 25 to show that initial graphs G_t^{init} and G_s^{init} are in simulation relation $\mathcal{R}$. Then using Lemma 26 we show that each step of the target graph construction $G_t \to G'_t$ can be simulated by the source graph construction step $G_s \to G'_s$. Finally, we use lemma Lemma 31 to derive that the final graph G_s is $\mathcal{M}$ consistent. □

E.3 Sequentialization

Definition 33. $\mathcal{R}(G_s, G_t, m, t_1, t_2)$ — is a *simulation relation* for the sequentialization transformation, where G_s is a source graph, G_t is a target graph, and $m : G_t.E \to G_s.E$ is an event mapping function, and $G_s.t_1$ events are appended before the t_2 events in G_t.

- m is injective
- $G_t.\mathrm{tid} = |_{\neq t_2}\ G_s.\mathrm{tid} \circ m$
- $\forall e.\ G_s.\mathrm{tid}(m(e)) = t_2 \implies G_t.\mathrm{tid}(e) = t_1$
- $G_t.\mathrm{lab} = G_s.\mathrm{lab} \circ m$
- $m \uparrow G_t.E = G_s.E$

- $\mathsf{m} \uparrow \mathsf{G_t}.\mathsf{po} = \mathsf{G_s}.\mathsf{po} \cup \mathsf{po}_{seq}$
- $\mathsf{m} \uparrow \mathsf{G_t}.\mathsf{rf} = \mathsf{G_s}.\mathsf{rf}$
- $\mathsf{m} \uparrow \mathsf{G_t}.\mathsf{mo} = \mathsf{G_s}.\mathsf{mo}$

where:

$$\mathsf{po}_{seq} \triangleq \{(a, b) \mid \mathsf{G_s}.\mathsf{tid}(a) = \mathsf{t}_1 \wedge \mathsf{G_s}.\mathsf{tid}(b) = \mathsf{t}_2\}$$

Lemma 32. *Let* $\mathsf{G}_\mathsf{t}^{init}$ *and* $\mathsf{G}_\mathsf{s}^{init}$ *be initial execution graphs. Then* $\mathcal{R}(\mathsf{G}_\mathsf{s}^{init}, \mathsf{G}_\mathsf{t}^{init}, \mathsf{id}, \mathsf{t}_1, \mathsf{t}_2)$ *holds.*

Proof. po, rf, and mo are empty in the initial graphs, po_{seq} is empty as well. Hence, the last three conditions of the simulation relation hold. The remaining ones are trivial for initial graphs. □

Lemma 33. *Let* $\mathsf{G_t}$ *and* $\mathsf{G_s}$ *be* C20 *consistent and assume* $\mathcal{R}(\mathsf{G_s}, \mathsf{G_t}, \mathsf{m}, \mathsf{t}_1, \mathsf{t}_2)$ *holds. If* $\mathsf{G_t} \to \mathsf{G}'_\mathsf{t}$ *holds then there exists* G'_s *and* m' *such that* $\mathsf{G_s} \to^* \mathsf{G}'_\mathsf{s}$ *and* $\mathcal{R}(\mathsf{G}'_\mathsf{s}, \mathsf{G}'_\mathsf{t}, \mathsf{m}', \mathsf{t}_1, \mathsf{t}_2)$ *also hold.*

Proof. We do a case analysis on $\mathsf{G_t} \to \mathsf{G}'_\mathsf{t}$ and use Lemmas 34 and 35 to derive G'_s and $\mathcal{R}(\mathsf{G}'_\mathsf{s}, \mathsf{G}'_\mathsf{t}, \mathsf{m}, \mathsf{t}_1, \mathsf{t}_2)$. □

Lemma 34. *Let* $\mathsf{G_t}$ *and* $\mathsf{G_s}$ *be* C20 *consistent and* $\mathcal{R}(\mathsf{G_s}, \mathsf{G_t}, \mathsf{m}, \mathsf{t}_1, \mathsf{t}_2)$ *holds. If* $\mathsf{G_t} \xrightarrow[exec]{e} \mathsf{G}'_\mathsf{t}$ *holds then there exists* G'_s *and* m' *such that* $\mathsf{G_s} \xrightarrow[exec]{\mathsf{m}'(e)} \mathsf{G}'_\mathsf{s}$ *and* $\mathcal{R}(\mathsf{G}'_\mathsf{s}, \mathsf{G}'_\mathsf{t}, \mathsf{m}', \mathsf{t}_1, \mathsf{t}_2)$ *also hold.*

Proof. Let us define e' as an event, s.t. $\mathcal{P}_e(e')$ holds:

$$\mathcal{P}_e(e') \triangleq (\mathsf{G_s}.\mathsf{op}(e') = \mathsf{G_t}.\mathsf{op}(e) \wedge \mathsf{G_s}.\mathsf{loc}(e') = \mathsf{G_t}.\mathsf{loc}(e) \wedge \mathsf{G_s}.\mathsf{ord}(e') = \mathsf{G_t}.\mathsf{ord}(e))$$

Therefore, we can construct G'_s and m' as follows:

- $\mathsf{m}' = \mathsf{m}[e \mapsto e']$
- $\mathsf{G}'_\mathsf{s}.\mathsf{E} = \mathsf{m}' \uparrow \mathsf{G}'_\mathsf{t}.\mathsf{E}$
- $\mathsf{G}'_\mathsf{s}.\mathsf{lab} = \mathsf{G}'_\mathsf{t}.\mathsf{lab} \circ \mathsf{m}'$
- $\mathsf{G}'_\mathsf{s}.\mathsf{rf} = \mathsf{m}' \uparrow \mathsf{G}'_\mathsf{t}.\mathsf{rf}$
- $\mathsf{G}'_\mathsf{s}.\mathsf{mo} = \mathsf{m}' \uparrow \mathsf{G}'_\mathsf{t}.\mathsf{mo}$

For the last components, let us introduce c_t, such that c_t is the po-last event, appended before initial t_2 events in $\mathsf{G_t}$, and d_t is po-followed by c_t. Thread identifier function and program order are defined on case basis:

(1) if $\mathsf{tid}(e) \neq \mathsf{t}_2$ then
   - $\mathsf{G}'_\mathsf{s}.\mathsf{tid} = \mathsf{G_s}.\mathsf{tid}[e \mapsto \mathsf{G}'_\mathsf{t}.\mathsf{tid}(e)]$
   - $\mathsf{G}'_\mathsf{s}.\mathsf{po} = \mathsf{m} \uparrow \mathsf{G}'_\mathsf{t}.\mathsf{po}$

(2) if $\mathsf{tid}(e) = \mathsf{t}_2$:
   - $\mathsf{G}'_\mathsf{s}.\mathsf{tid} = \mathsf{G_s}.\mathsf{tid}[e \mapsto \mathsf{t}_2]$
   - $\mathsf{G}'_\mathsf{s}.\mathsf{po} = \mathsf{m} \uparrow \mathsf{G}'_\mathsf{t}.\mathsf{po} \cup \mathsf{G_s}.\mathsf{E}_{t_1} \times \{e\}$

$\mathcal{R}(\mathsf{G_s}, \mathsf{G_t}, \mathsf{m}, \mathsf{t}_1, \mathsf{t}_2)$ holds by construction. And consistency of G'_s follows from the consistency of G'_t due to monotonicity property Definition 18. □

Lemma 35. *Let* $\mathsf{G_t}$ *and* $\mathsf{G_s}$ *be* C20 *consistent and* $\mathcal{R}(\mathsf{G_s}, \mathsf{G_t}, \mathsf{m}, \mathsf{t}_1, \mathsf{t}_2)$ *holds. If* $\mathsf{G_t} \xrightarrow[re-exec]{C} \mathsf{G}'_\mathsf{t}$ *holds then there exists* G'_s, m', *and* rf'_{re} *such that* $\mathsf{G_s} \xrightarrow[re-exec]{\mathsf{m}' \uparrow C} \mathsf{G}'_\mathsf{s}$ *holds, and* $\mathcal{R}(\mathsf{G}'_\mathsf{s}, \mathsf{G}'_\mathsf{t}, \mathsf{m}', \mathsf{t}_1, \mathsf{t}_2)$ *also holds.*

Proof. We construct G'_s and m' as follows:

- $\mathsf{m}' = \mathsf{id}$
- $\mathsf{G}'_\mathsf{s}.\mathsf{E} = \mathsf{m}' \uparrow \mathsf{G}'_\mathsf{t}.\mathsf{E}$
- $\mathsf{G}'_\mathsf{s}.\mathsf{tid} = \mathsf{G_s}.\mathsf{tid}$

- $G'_s.\mathsf{lab} = G'_t.\mathsf{lab} \circ m'$
- $G'_s.\mathsf{po} = G_s.\mathsf{po}$
- $G'_s.\mathsf{rf} = m' \uparrow G'_t.\mathsf{rf}$
- $G'_s.\mathsf{mo} = m' \uparrow G'_t.\mathsf{mo}$

$\mathcal{R}(G'_s, G'_t, m', t_1, t_2)$ holds by construction.

Now, we set C_s, $\mathcal{D}_s$, f_s, and $\preceq_{\mathsf{tid}s}$ as follows:

- $C_s \triangleq m \uparrow C'_t$
- $\mathcal{D}_s \triangleq m \uparrow \mathcal{D}'_t$
- $f_s \triangleq \mathsf{id}$
- $\preceq_{\mathsf{tid}s} \triangleq \mathit{InsertBefore}(\preceq_{\mathsf{tid}t}, t_1, t_2)$

The following properties follow immediately from the definitions:

- $C_s \subseteq G'_s.\mathsf{E}$
- $\mathcal{D}_s \subseteq C_s$
- $\mathsf{dom}(G.\mathsf{po}; [\mathcal{D}_s]) \subseteq \mathcal{D}_s$
- $\mathit{WellFormed}(G|_{\mathcal{D}_s}, G'_s, C_s)$
- $\mathit{CommitEmbedded}(G_s, G'_s, C_s, f_s)$

The property $\mathsf{dom}(G'_s.\mathsf{rpo}; [G'_s.\mathsf{E} \setminus \mathcal{D}_s]) \subseteq \mathcal{D}_s$ follows from the fact that $G'_s.\mathsf{rpo} \subseteq G'_t.\mathsf{rpo}$ and the respective property of the target graph.

To prove $\mathit{ThreadOrderedUEvents}(G', C, \prec_{\mathsf{tid}})$ we use the following observation:

$$\begin{aligned} G'_s.\mathsf{vf}; G'_s.{=_{\mathsf{tid}}}; [\mathcal{U}_s] &= [G'_s.\mathsf{W}]; G'_s.\mathsf{rf}^?; G'_s.\mathsf{hb}^?; G'_s.{=_{\mathsf{tid}}}; [\mathcal{U}_s] \subseteq \\ &\subseteq m' \uparrow ([G'_t.\mathsf{W}]; G'_t.\mathsf{rf}^?; G'_t.\mathsf{hb}^?; G'_t.{=_{\mathsf{tid}}}; [\mathcal{U}_t] \subseteq \\ &\subseteq m' \uparrow \preceq_{\mathsf{tid}t} \subseteq m' \preceq_{\mathsf{tid}s} \end{aligned}$$

Consistency of the source graph, $\mathit{Consistent}(\mathcal{M}, G'_s)$, follows directly from the consistency of the target graph, $\mathit{Consistent}(\mathcal{M}, G'_t)$, due to monotonicity property Definition 18.

It is left to show that:

$$\langle G'_s, C_s \rangle \vdash G_s|_{\mathcal{D}_s} \Rightarrow^* G'_s$$

For this, we can apply Lemma 1 with ℓ inherited from the target graph. □

F XMC Model Checking

In this appendix we provide the proof of Theorem 4.1.

Proof. We note that whenever XMC adds a new event in porf order following the basic procedure Visit(P, G) provided by GenMC [Kokologiannakis et al. 2022], it mimics the (Execute) rule of XMM—the soundness of this procedure is guaranteed by the GenMC itself.

We only need to show that application of the new procedure reeXecute(P, G) performed by the XMC is in fact a particular case of applying the (Re-Execute) of XMM. Remember that XMC selects the subsets of committed and determined events judiciously as follows:

$$\mathcal{D} \triangleq G.\mathsf{E} \setminus \mathsf{codom}([r]; G.\mathsf{po}^? \quad \text{and} \quad C \triangleq \mathcal{D} \cup \mathsf{codom}([r]; G.\mathsf{po}^?) \cap \mathsf{dom}(G.\mathsf{rf} \setminus G.\mathsf{po})$$

We will denote the fact that the procedure reeXecute(P, G) derives from a graph G new graph G′ as $G \xrightarrow[xmc]{\langle \mathcal{D}, C \rangle} G'$. Also, let $G_i \xRightarrow[xmc]{\langle e, \ell \rangle} G_{i+1}$ denote intermediate graph construction steps performed by reeXecute(P, G), where e is an added event and ℓ is its label.

Thus we need to show that $G \xrightarrow[xmc]{\langle \mathcal{D}, C \rangle} G'$ implies $G \xrightarrow[re-exec]{C} G'$. This is true if the following properties hold:

- *WellFormed*$(\mathrm{G}|_{\mathcal{D}}, \mathrm{G}', C)$
- *WellFormed*$(\mathrm{G}_{i+1}, \mathrm{G}', C)$
- *CommitEmbedded*$(\mathrm{G}, \mathrm{G}', C, \mathrm{f})$
- *Consistent*(G')

Each of the above properties is proven to hold in Lemmas 37 to 40. □

Lemma 36. *Whenever* XMC *adds an event to a graph, the resulting graph is WellFormed.*

Proof. XMC uses GenMC to add events to graphs. By design, GenMC ensures that the resulting graph is *WellFormed*. Since XMC starts from an empty graph, which is trivially *WellFormed*, all graphs explored by XMC and their prefixes are also *WellFormed*. □

Lemma 37. *If* $\mathrm{G} \xrightarrow[xmc]{\langle \mathcal{D}, C \rangle} \mathrm{G}'$, *then WellFormed*$(\mathrm{G}|_{\mathcal{D}}, \mathrm{G}', C)$.

Proof. We show that *WellFormed*$(\mathrm{G}|_{\mathcal{D}}, \mathrm{G}', C)$ holds by unfolding the definition of configuration well-formedness and showing that each of its conditions holds:

- *WellFormed*$(\mathrm{G}|_{\mathcal{D}})$ follows from Lemma 36 because $\mathrm{G}|_{\mathcal{D}}$ is a prefix of G'.
- $\forall c \in C.\ \mathrm{G}|_{\mathcal{D}}.\mathsf{tid}(c) = \mathrm{G}'.\mathsf{tid}(c)$ because $\mathrm{G}|_{\mathcal{D}}$ is a prefix of G'
- $\forall c \in C.\ \mathrm{G}|_{\mathcal{D}}.\mathsf{lab}(c) = \mathrm{G}'.\mathsf{lab}(c)$ because $\mathrm{G}|_{\mathcal{D}}$ is a prefix of G'
- $[\mathrm{G}|_{\mathcal{D}}.C]; \mathrm{G}'.\mathsf{po}; [\mathrm{G}|_{\mathcal{D}}.C] \subseteq \mathrm{G}|_{\mathcal{D}}.\mathsf{po}$ because $\mathrm{G}|_{\mathcal{D}}$ is a prefix of G'
- $[\mathrm{G}|_{\mathcal{D}}.C]; \mathrm{G}'.\mathsf{rf}; [\mathrm{G}|_{\mathcal{D}}.C] \subseteq \mathrm{G}|_{\mathcal{D}}.\mathsf{rf}$ because $\mathrm{G}|_{\mathcal{D}}$ is a prefix of G'
- $[\mathrm{G}|_{\mathcal{D}}.C]; \mathrm{G}'.\mathsf{mo}; [\mathrm{G}|_{\mathcal{D}}.C] \subseteq \mathrm{G}|_{\mathcal{D}}.\mathsf{mo}$ because $\mathrm{G}|_{\mathcal{D}}$ is a prefix of G'
- $\mathrm{G}|_{\mathcal{D}}.\mathrm{R} \subseteq \mathsf{codom}(\mathrm{G}|_{\mathcal{D}}.\mathsf{rf}) \cup C$ because $\mathcal{D} \subseteq C$
- $\mathrm{G}'.\mathrm{R} \cap C \subseteq \mathsf{codom}([C]; \mathrm{G}'.\mathsf{rf})$ because $\mathcal{D} \subseteq C \wedge (\forall r \in \mathrm{G}'.\mathrm{R} \cap C.\ \mathsf{dom}(\mathrm{G}'.\mathsf{rf}; r) \in \mathcal{D} \vee$ r is matched to a committed write)

□

Lemma 38. *If* $\mathrm{G}_i \xrightarrow[xmc]{\langle e, \ell \rangle} \mathrm{G}_{i+1}$, *then WellFormed*$(\mathrm{G}_{i+1}, \mathrm{G}', C)$.

Proof. We show that *WellFormed*$(\mathrm{G}_{i+1}, \mathrm{G}', C)$ holds by unfolding the configuration well-formedness definition and showing that each of its conditions holds:

- *WellFormed*(G_{i+1}) follows from Lemma 36 because G_{i+1} is a prefix of G'.
- $\forall c \in C.\ \mathrm{G}_{i+1}.\mathsf{tid}(c) = \mathrm{G}'.\mathsf{tid}(c)$ because G_{i+1} is a prefix of G'.
- $\forall c \in C.\ \mathrm{G}_{i+1}.\mathsf{lab}(c) = \mathrm{G}'.\mathsf{lab}(c)$ because G_{i+1} is a prefix of G'.
- $[\mathrm{G}_{i+1}.C]; \mathrm{G}'.\mathsf{po}; [\mathrm{G}_{i+1}.C] \subseteq \mathrm{G}_{i+1}.\mathsf{po}$ because G_{i+1} is a prefix of G'.
- $[\mathrm{G}_{i+1}.C]; \mathrm{G}'.\mathsf{rf}; [\mathrm{G}_{i+1}.C] \subseteq \mathrm{G}_{i+1}.\mathsf{rf}$ because G_{i+1} is a prefix of G'.
- $[\mathrm{G}_{i+1}.C]; \mathrm{G}'.\mathsf{mo}; [\mathrm{G}_{i+1}.C] \subseteq \mathrm{G}_{i+1}.\mathsf{mo}$ because G_{i+1} is a prefix of G'.
- $\mathrm{G}_{i+1}.\mathrm{R} \subseteq \mathsf{codom}(\mathrm{G}_{i+1}.\mathsf{rf}) \cup C$ because, when a read r is added to G_i to obtain G_{i+1}, GenMC-XMM always creates an rf edge ending in r. Alternatively, if r was not added, it means that $r \in \mathcal{D}$, and thus $r \in C$.
- $\mathrm{G}'.\mathrm{R} \cap C \subseteq \mathsf{codom}([C]; \mathrm{G}'.\mathsf{rf})$ holds, as we showed in the proof of Lemma 37.

□

Lemma 39. *If* $\mathrm{G} \xrightarrow[xmc]{\langle \mathcal{D}, C \rangle} \mathrm{G}'$, *then CommitEmbedded*$(\mathrm{G}, \mathrm{G}', C, \mathrm{f})$.

Proof. We show that *CommitEmbedded*$(\mathrm{G}, \mathrm{G}', C, \mathrm{f})$ holds by unfolding Definition 7 and showing that each of its conditions holds:

- $\mathrm{G}|_{\mathcal{D}} \subseteq \mathrm{G}$ by definition of the XMC restriction step.
- $\mathrm{G}|_{\mathcal{D}} \subseteq \mathrm{G}'$ by definition of the XMC restriction step.

- f is injective, because only one write is matched to a $R_\perp$, so no two committed events in G can correspond to the same committed event in G′.
- $\forall c \in C.\ \mathrm{f}(c) \neq \perp$ because if there are left-over $R_\perp$ at the end of execution, XMC discards the graph.
- $\forall c \in C.\ \mathrm{G.tid}(\mathrm{f}(c)) = \mathrm{G}'.\mathrm{tid}(c)$ because either $c \in \mathrm{G}|_{\mathcal{D}}$, or c is a write matched to a $R_\perp$ that read from a write in G in the same thread as c.
- $\forall c \in C.\ \mathrm{G.lab}(\mathrm{f}(c)) = \mathrm{G}'.\mathrm{lab}(c)$ because either $c \in \mathrm{G}|_{\mathcal{D}}$, or c is a write matched to a $R_\perp$ that read from a write in G with the same label.
- $\mathrm{f} \uparrow ([C]; \mathrm{G}'.\mathsf{rpo}; [C]) = [\mathrm{f} \uparrow C]; \mathrm{G}.\mathsf{rpo}; [\mathrm{f} \uparrow C]$ because we have added a specific check that discards executions that do not satisfy this condition.
- $\mathrm{f} \uparrow ([C]; \mathrm{G}'.\mathsf{rf}; [C]) = [\mathrm{f} \uparrow C]; \mathrm{G}.\mathsf{rf}; [\mathrm{f} \uparrow C]$ because $\mathrm{G}|_{\mathcal{D}} \subseteq \mathrm{G}'$ so all rf edges in $\mathrm{G}|_{\mathcal{D}}$ remain the same. Committed writes matched to a $\mathrm{R}_\perp$ in G' have an equivalent write in G, so their rf edges are also the same.
- $\mathrm{f} \uparrow ([C]; \mathrm{G}'.\mathsf{mo}; [C]) = [\mathrm{f} \uparrow C]; \mathrm{G}.\mathsf{mo}; [\mathrm{f} \uparrow C]$ because we have added a specific check that discards executions that do not satisfy this condition.

□

Lemma 40. *If* $\mathrm{G} \xrightarrow[xmc]{C} \mathrm{G}'$, *then* $\mathit{Consistent}(\mathrm{G}')$.

Proof. This is true, because by construction XMC checks the consistency of the resulting graph G′ in the procedure reeXecute(P, G). □

G Benchmarks description

Litmus tests.

- **LB+acq** is like the classic LB test, but the loads use acquire memory order.
- **LB+coh-cyc** is taken from [Chakraborty and Vafeiadis 2019]. In this example, the outcome $r1 = 3 \wedge r2 = 2 \wedge r3 = 1$ is allowed by Promising, and forbidden by Weakestmo and XMM.
- **LB+equals** is a test where a compiler could potentially find invariant $X = Y$, and remove the redundant if statement if $(X == Y)$.
- **LB+rel** is like the classic LB test, but the stores use release memory order.
- **LB-invis-write+dep** is a test designed to verify that the execution obtained after applying the sequentialization optimization is outputted.
- **R-bot-multi-matching** is a test designed to verify that all writes that are compatible with a certain R_bot are matched.
- **java-test9a** is equal to Java causality test 9, taken from [Manson et al. 2005a], except that variable X is initialized to 2, and thread 3 writes 0 to X instead of 2.
- **LB+coh-cyc+Wd** is similar to *LB+coh-cyc*, but with a read from a distinct location in the second thread and a write to this location in a separate thread. $r1 = 3 \wedge r2 = 2 \wedge r3 = 1$ should be still forbidden.
- **java-test10** is Java causality test 10 [Manson et al. 2005a]. The Java model forbids outcome $r1 = 1 \wedge r2 = 1 \wedge r3 = 0$, but it is allowed by XMM.
- **java-test19** is Java causality test 19 [Manson et al. 2005a]. Both the Java model, and XMM allow outcome $r1 = 42 \wedge r2 = 42 \wedge r3 = 42$.
- **java-test20** is Java causality test 20 [Manson et al. 2005a]. Both the Java model, and XMM allow outcome $r1 = 42 \wedge r2 = 42 \wedge r3 = 42$.
- **LB+seq** is a test taken from the Promising paper [Kang et al. 2017]. It tests whether the sequentialization optimization is allowed.

- **java-test5** is Java causality test 5 [Manson et al. 2005a]. The Java model forbids outcome $r1 = 1 \wedge r2 = 1 \wedge r3 == 0$, but XMM allows it.
- **LB+coh+RR+cf** this test shows between the global mo order of the Weakestmo model, and the absence of such constraint in the XMM model. Due to the absence of this constraint, XMM allows more executions than Weakestmo in this test.

Data Structure Benchmarks.

- **mpmc-queue-bnd** is a multi-producer multi-consumer queue.
- **ticketlock** is an implementation of a ticketlock algorithm.
- **chase-lev** is a dynamic circular work-stealing deque
- **buf-ring** is a multi-producer multi-consumer ring buffer.
- **dq** is a multi-producer multi-consumer deque.
- **stc** is a stack implementation using Treiber's algorithm.
- **linuxrwlocks** is a read-write lock ported from the Linux kernel
- **twalock** is a ticket lock augmented with a waiting array
- **fcombiner** implements concurrent access to a set data structure using the flat combiner access paradigm.
- **mutex** is a mutual exclusive lock.

Synthetic Benchmarks.

- **reorder2**: 2 threads write to 2 variables, while 2 other threads read from them.
- **fib-bench**: 2 threads compute the Fibonacci sequence using two shared variables.
- **szymanski**: is an implementation of Szymański's Mutual Exclusion Algorithm.
- **dekker-bnd**: is an implementation of Dekker's Mutual Exclusion algorithm.
- **indexer**: N threads modify a shared array.
- **ainc**: N threads increment a shared variable by 1.
- **casrot**: N threads compare-and-swap a shared variable, rotating its value.
- **casw**: N threads compare-and-swap a shared variable, and write a new value to it.

Load Buffering Benchmarks.

- **LBn+ctrl**: N threads with a load buffering pattern spanning through all threads. All threads except the first one have a real control dependency between the load and the store.
- **LBn+data**: N threads with a load buffering pattern spanning through all threads. All threads except the first one have a real data dependency between the load and the store.
- **LBn**: N threads with a load buffering pattern spanning through all threads.
- **LBn-pairs**: N threads divided in pairs. Each pair reads/writes from/to a unique pair of variables forming a load-buffering race.

G.1 Duplicate Executions

In Table 3 we show the benchmarks with the respective executions in the model checkers and the number of duplicate executions in WMC and XMC. We also note the number of LB races. In Table 4 we note the execution times of the model checkers to explore these executions.

Table 3. Load Buffering benchmarks. XMC finds the same number of cyclic executions as other tools. The number of duplicates visited is slightly less than that of WMC.

	Number of Executions				Number of Duplicates		
Test Name	GenMC	HMC	WMC	XMC	WMC	XMC	LB Races
LBn+ctrl(10)	9	11	11	11	0	1	1
LBn+ctrl(12)	10	11	11	11	0	1	1
LBn+ctrl(14)	10	11	11	11	0	1	1
LBn+data(10)	19	1024	1024	1024	0	1	10
LBn+data(12)	1023	1024	1024	1024	0	1	10
LBn+data(14)	1023	1024	1024	1024	0	1	10
LBn(10)	1023	1024	1024	1024	9	10	10
LBn(12)	4095	4096	4096	4096	11	12	12
LBn(14)	16383	16384	16384	16384	13	14	14
LBn-pairs(10)	243	1024	1024	1024	2101	2184	2560
LBn-pairs(12)	729	4096	4096	4096	11529	10379	12288
LBn-pairs(14)	2187	16384	16384	16384	61741	48250	57344

Table 4. Load Buffering time benchmarks. XMC takes the same time as other tools when there are few LB races. It takes longer when the number of LB races increases.

	Execution Time				
Test Name	GenMC	HMC	WMC	XMC	LB Races
LBn+ctrl(10)	0.06s	0.05s	0.01s	0.04s	1
LBn+ctrl(12)	0.05s	0.04s	0.01s	0.04s	1
LBn+ctrl(14)	0.04s	0.04s	0.01s	0.05s	1
LBn+data(10)	0.04s	0.19s	0.10s	0.15s	10
LBn+data(12)	0.10s	0.18s	0.10s	0.13s	10
LBn+data(14)	0.11s	0.17s	0.10s	0.11s	10
LBn(10)	0.11s	0.14s	0.12s	0.13s	10
LBn(12)	0.33s	0.55s	0.46s	0.40s	12
LBn(14)	1.37s	2.40s	1.68s	1.47s	14
LBn-pairs(10)	0.06s	0.19s	0.34s	1.43s	2560
LBn-pairs(12)	0.12s	0.65s	2.27s	7.06s	12288
LBn-pairs(14)	0.34s	3.19s	11.60s	38.12s	57344