The Defeater, Rule by Rule:
why the published MRDT soundness proof cannot linearize one fully LCA-legal merge

Sal / FP Launchpad (KC + Claude)

July 18, 2026

Abstract

The consolidated Sal note (`sal-mrdts.pdf`, Part I) states that a fully LCA-legal execution defeats the soundness proof of the published MRDT metatheory, and compresses the argument into one example and one paragraph. This document decompresses it. We restate the published proof slowly (the induction, the six-set partition of a merge, and the eight-law toolbox it consumes), walk the defeater execution version by version with every state computed and every LCA checked, and then, at the critical merge, enumerate every move the published toolbox offers: each of the five merge rules in each orientation, the automation catalogue behind them, and the natural repairs of the peel architecture (closure-strengthened peels, block peels, widened induction classes). Each move fails for a stated, checked reason; every failure is anchored either to a computation displayed on the page or to a named kernel-checked theorem (`no_inter_lca_2op_rem_peel_of_defeater`, `crack1_witness`, `reunification_peel_obstruction`, `no_proper_back_block`, `killTest_no_common_U`). We then state precisely what is and is not defeated: the published *theorem* survives on this execution (the merged version has a witness, exhibited), the published *proof* does not, and neither does any peel-shaped repair of it. Finally we run the corrected metatheory on the very same merge and display the five-line derivation that discharges it, so the contrast between exhaustion and construction is visible on one page.

Contents

Provenance. The account of the published proof below follows the paper's own sources: the text of *Automatically Verifying Replication-aware Linearizability* (the Neem paper) and its F* artifact interface `App_mrdt.fsti`, both vendored in this repository under `_references/`; no statement about the published system is reconstructed from memory. The defeater and every claimed failure are taken from the Lean sources cited inline (all kernel-checked, zero `sorry`s in the cited chains); where this document and any compressed prose account could disagree, the Lean is the ground truth. The general corrected theory is in `sal-mrdts.pdf`, Part I; this document expands its § "Three facts the metatheory must respect" and does not replace it.

1 What the published proof does

1.1 The statement and the induction

An MRDT is a tuple $\mathcal{D} = \langle \Sigma, \sigma_0, \mathsf{do}, \mathsf{merge}, \mathsf{rc} \rangle$: states Σ with initial state σ_0; do applies an event $e = (t, r, o)$ (globally unique timestamp t, origin replica r, operation o), written $e(\sigma)$; a three-way merge $\mathsf{merge}(l, a, b)$ whose first argument is the state at the lowest common ancestor (LCA) of the two versions being merged; and a conflict-resolution relation rc that orders concurrent non-commuting operations. Executions form a version DAG: replicas fork, apply events, and merge; a version v carries a state σ_v and the set $L(v)$ of events that produced it; visibility $e_1 \xrightarrow{\mathsf{vis}} e_2$ records that e_2's issuing replica had seen e_1 when it issued e_2. Write $e_1 \parallel e_2$ for concurrency (neither sees the other) and $e_1 \rightleftarrows e_2$ for commutation ($e_1(e_2(\sigma)) = e_2(e_1(\sigma))$ for all σ).

Definition 1.1 (linearization order, per the paper)**.** $e_1 \ \mathsf{lo}\ e_2$ *iff* $\big(e_1 \xrightarrow{\mathsf{vis}} e_2 \wedge e_1 \not\rightleftarrows e_2\big)$ *or* $\big(e_1 \parallel e_2 \wedge e_1 \xrightarrow{\mathsf{rc}} e_2 \wedge \neg\exists e_3.\ e_2 \xrightarrow{\mathsf{vis}} e_3 \wedge e_2 \not\rightleftarrows e_3\big)$*. A visible non-commuting pair is ordered by visibility; a concurrent* rc*-resolved pair is ordered by* rc*, unless the losing event already has a later non-commuting observer (an absorber) somewhere in the configuration, in which case the* rc *edge is dropped. We write* lo^E *for the set-relative variant whose absorber clause ranges over*

a set E only; since an absorber anywhere kills a lo *edge,* $\mathsf{lo} \subseteq \mathsf{lo}^E$ *for every* E*. The paper uses* lo*; the set-relative variant enters in §2–§5 and the distinction turns out to matter.*

Definition 1.2 (RA-linearizability)**.** *A sequence* π *witnesses a version* v *when (i)* π *enumerates* $L(v)$ *without repetition, (ii)* π *extends* lo *(no later element of* π *is* lo*-before an earlier one), and (iii)* π *folds from* σ_0 *to* σ_v*. A configuration is RA-linearizable when every version admits a witness.*

The published soundness proof (the paper's Theorem 1, automated as its Theorem 2) runs an induction on the length of the execution:

- **Assumed at each step**: every version of the current configuration is linearizable; in particular each replica head v has a witness π_v that folds to σ_v.
- **Fork** and **query** change nothing. **Apply** extends a head's witness by the new event: $\pi \mapsto \pi e$.
- **Merge** is the hard case, and the entire content of the proof: given heads v_1, v_2 with LCA $v_\top$ (the store guarantees $L(v_\top) = L(v_1) \cap L(v_2)$), the new version holds $\sigma_m = \mathsf{merge}(\sigma_\top, \sigma_1, \sigma_2)$ and event set $L(v_1) \cup L(v_2)$, and the proof must *produce* a witness of it: a sequence π enumerating $L(v_1) \cup L(v_2)$, extending $\mathsf{lo}_m \coloneqq \mathsf{lo}$ restricted to that union, and folding to σ_m.

1.2 The merge case: the six-set partition and bottom-up peeling

The proof first fixes the *shape* of the witness it will build. With $L_1 = L(v_1)$, $L_2 = L(v_2)$, $L_\top = L(v_\top)$, it defines the local events $L'_i = L_i \setminus L_\top$ and splits them by whether they must precede an LCA event:

$$L^b_i = \{e \in L'_i \mid e \text{ is } \mathsf{lo}_m\text{-before an LCA event, directly or via one local event}\}, \quad L^a_i = L'_i \setminus L^b_i,$$

and splits the LCA events into $L^a_\top$ (those with a local lo_m-predecessor) and $L^b_\top$ (the rest). Causal closure of versions makes every pair in $L'_1 \times L'_2$ concurrent. The witness is then assembled in three segments,

$$\pi \;=\; \underbrace{S_1}_{L^b_\top} \;\; \underbrace{S_2}_{L^a_\top \cup L^b_1 \cup L^b_2} \;\; \underbrace{S_3}_{L^a_1 \cup L^a_2},$$

and the proof constructs π *bottom-up*: it repeatedly *peels* the current last event through the merge, using one of three interchange rules, and recurses on the smaller merge instance that remains. A peel does two things at once. It commits one event e to the current last slot of π, which is legal only if e is lo_m-maximal among the events not yet placed (otherwise π stops extending lo_m). And it rewrites the merge equation, which is possible only if the rule's arguments *syntactically factor*: the side an event is peeled from must literally be a do-image with the peeled operation outermost. There are no front peels in the published system; every rule extracts the last event.

1.3 The toolbox: eight laws

Table 1 lists the complete toolbox the published merge case consumes: three constraint laws on the update layer and five merge laws, of which three are the peel rules.

Two features of the peel rules deserve emphasis, because the defeater lives exactly on them.

Law	Statement (shape)	Role at a merge step
RC-NON-COMM	for distinct events on different replicas: rc returns Either iff the pair commutes	ties the conflict table to the commutation table; arbitration is exactly where it is needed
NO-RC-CHAIN	never $o_1 \xrightarrow{\mathsf{rc}} o_2$ and $o_2 \xrightarrow{\mathsf{rc}} o_3$	keeps lo acyclic, so maximal (peelable-last) events exist
COND-COMM	after $o_1 \xrightarrow{\mathsf{rc}} o_2$ is applied in either order, any later non-commuting o_3 erases the difference (base + inductive form)	lets differently-ordered replays agree; the convergence engine
MERGECOMMUTATIVITY	$\mathsf{merge}(l, a, b) = \mathsf{merge}(l, b, a)$	halves the rule set: every peel exists in both side orientations
MERGEIDEMPOTENCE	$\mathsf{merge}(a, a, a) = a$	terminates the derivation once only shared events remain
BOTTOMUP-2-OP	$e_1 \neq e_2,\ e_1 \xrightarrow{\mathsf{rc}} e_2 \vee e_1 \rightleftarrows e_2 \implies \mathsf{merge}(l, e_1(a), e_2(b)) = e_2(\mathsf{merge}(l, e_1(a), b))$	peels the last local event when *both* sides still hold unlinearized local events ($L_1^a, L_2^a \neq \emptyset$)
BOTTOMUP-1-OP	$\mathsf{merge}(e_\top(l), e_1(a), e_\top(b)) = e_1(\mathsf{merge}(e_\top(l), a, e_\top(b)))$	peels the last local event of the one remaining side, under a shared trailing LCA event ($L_2^a = \emptyset$)
BOTTOMUP-0-OP	$\mathsf{merge}(e_\top(l), e_\top(a), e_\top(b)) = e_\top(\mathsf{merge}(l, a, b))$	peels a shared LCA event, present outermost on *all three* arguments ($L_1^a = L_2^a = \emptyset$)

Table 1: The published toolbox: the three update-layer constraints and the five merge laws of the paper's Theorem 1. The derivation schedule is: 2-OP/1-OP until S_3 is exhausted, then alternating 0-OP (for $L_\top^a$ events) and 2-OP/1-OP (for the L_i^b events attached to them) through S_2, then MERGEIDEMPOTENCE for S_1.

- **The factorization requirement.** In BOTTOMUP-2-OP, the state written $e_2(b)$ is universally quantified as a *pattern*: to apply the rule to a concrete merge $\mathsf{merge}(\sigma_\top, \sigma_1, \sigma_2)$ one must exhibit b with $\mathsf{do}(b, e_2) = \sigma_2$. If no such b exists, the rule does not apply, whatever its side conditions say. The same holds for $e_1(a)$ in 1-OP and for all three arguments in 0-OP.
- **The arbitration orientations.** A 2-OP peel consults rc: the peeled e_2 must either commute with the opposite side's pending event e_1 or be its rc-winner ($e_1 \xrightarrow{\mathsf{rc}} e_2$: the winner is applied last). With MERGECOMMUTATIVITY each candidate peel therefore exists in two orientations (which head sits in the peelable slot), and an exhaustive check must cover both.

1.4 The automation layer

The paper's Theorem 2 replaces each rule of Table 1 by an induction scheme over the six event sets, Skolemizing "holds on feasible states" into base-and-step equations an SMT solver can discharge. In the F* artifact (`_references/neem_fstar_repo/code/interface/App_mrdt.fsti`) this is the familiar per-datatype catalogue: the four update-layer obligations (`rc_non_comm`, `no_rc_chain`, `cond_comm_base`, `cond_comm_ind`), the two global merge laws (`merge_comm`, `merge_idem`), and the `base_*`/`ind_*`/`inter_*` families for the 2-OP and 1-OP rules to-

gether with `lem_0op`. The catalogue proves the *same* rules, restricted to states reachable through the event-set structure; it adds no new derivation move. One member matters below: `inter_lca_2op` (`App_mrdt.fsti`, lines 137–143), the catalogue's incarnation of the 2-OP peel under one shared LCA event o_l; its conclusion

$$\begin{aligned}&\mathsf{merge}\big(\mathsf{do}(l, o_l),\ \mathsf{do}(\mathsf{do}(a, o_l), o_1),\ \mathsf{do}(\mathsf{do}(b, o_l), o_2)\big)\\ &\qquad = \mathsf{do}\big(\mathsf{merge}(\mathsf{do}(l, o_l),\ \mathsf{do}(a, o_l),\ \mathsf{do}(\mathsf{do}(b, o_l), o_2)),\ o_1\big)\end{aligned}$$

peels o_1, and requires the side it is peeled from to have the syntactic shape $\mathsf{do}(\mathsf{do}(a, o_l), o_1)$ with o_1 outermost: the factorization requirement again, in its exact artifact form. This is the statement the kernel arbiter of §3 tests against.

2 The defeater execution, drawn and walked

2.1 The datatype

The defeater is stated over a one-key add-wins skeleton with tagged adds (Lean: `AWSet` in `Sal/CRDTs/Metatheory/Convergence_CounterModel.lean`; it is the kernel of the production OR-set family). Component by component:

Σ	pairs $\sigma = (A, D)$ of timestamp sets: A = tags added, D = tags dead
σ_0	$(\emptyset, \emptyset)$
do	$\mathsf{add}_t(A, D) = (A \cup \{t\},\ D)$ $\quad$ $\mathsf{rem}(A, D) = (A,\ A \cup D)$ $\quad$ (kills everything it sees)
merge	$\mathsf{merge}(l, (A_1, D_1), (A_2, D_2)) = (A_1 \cup A_2,\ D_1 \cup D_2)$ $\quad$ (union; the LCA slot is ignored)
rc	$\mathsf{rem} \xrightarrow{\mathsf{rc}} \mathsf{add}$ (add wins a race; all other pairs Either)
query	the *live set* $\mathsf{live}(\sigma) = A \setminus D$

Commutation: $\mathsf{add} \rightleftarrows \mathsf{add}$ (both insert into A), $\mathsf{rem} \rightleftarrows \mathsf{rem}$ ($D := A \cup D$ is absorbing), and $\mathsf{add} \not\rightleftarrows \mathsf{rem}$: $\mathsf{rem}(\mathsf{add}_t(\sigma))$ has t dead, $\mathsf{add}_t(\mathsf{rem}(\sigma))$ has t live; the state-dependence of rem is what makes the pair non-commuting. RC-NON-COMM and NO-RC-CHAIN hold (rc orders exactly the non-commuting pairs, in one direction, and no operation both wins and loses); the skeleton also satisfies COND-COMM and the two global merge laws. It is a legal MRDT, inside the published theory's class.

2.2 The execution, step by step

Three replicas: p, q, and a *staging* replica s that only merges. Four events, timestamps 1 to 4:

$$A_p = (1, p, \mathsf{add}), \qquad A_q = (2, q, \mathsf{add}), \qquad R_p = (3, p, \mathsf{rem}), \qquad R_q = (4, q, \mathsf{rem}).$$

Figure 1 shows the version DAG; Table 2 tabulates every version. The steps:

1. **Initial.** All replicas at v_0: $L(v_0) = \emptyset$, $\sigma_{v_0} = (\emptyset, \emptyset)$.
2. p **adds.** p applies A_p at v_0: version p_1, $L = \{A_p\}$, state $(\{1\}, \emptyset)$, live $\{1\}$.
3. q **adds.** q applies A_q at v_0: version q_1, $L = \{A_q\}$, state $(\{2\}, \emptyset)$, live $\{2\}$.
4. **The staging merge.** Replica s merges p_1 and q_1. Their only common ancestor is v_0, so LCA $= v_0$, and $L(v_0) = \emptyset = \{A_p\} \cap \{A_q\}$: *LCA-legal.* New version v_s, $L = \{A_p, A_q\}$, state $\mathsf{merge}((\emptyset, \emptyset), (\{1\}, \emptyset), (\{2\}, \emptyset)) = (\{1, 2\}, \emptyset)$, live $\{1, 2\}$.

Version	Event set	State (A, D)	Live	Created by	LCA check
v_0	$\emptyset$	$(\emptyset, \emptyset)$	$\emptyset$	init	
p_1	$\{A_p\}$	$(\{1\}, \emptyset)$	$\{1\}$	apply A_p	
q_1	$\{A_q\}$	$(\{2\}, \emptyset)$	$\{2\}$	apply A_q	
v_s	$\{A_p, A_q\}$	$(\{1,2\}, \emptyset)$	$\{1,2\}$	merge(p_1, q_1)	$L(v_0) = \emptyset = L(p_1) \cap L(q_1)$
p_2	$\{A_p, R_p\}$	$(\{1\}, \{1\})$	$\emptyset$	apply R_p	
q_2	$\{A_q, R_q\}$	$(\{2\}, \{2\})$	$\emptyset$	apply R_q	
head_p	$\{A_p, R_p, A_q\}$	$(\{1,2\}, \{1\})$	$\{2\}$	merge(p_2, v_s)	$L(p_1) = \{A_p\} = L(p_2) \cap L(v_s)$
head_q	$\{A_q, R_q, A_p\}$	$(\{1,2\}, \{2\})$	$\{1\}$	merge(q_2, v_s)	$L(q_1) = \{A_q\} = L(q_2) \cap L(v_s)$
$v_\top$	$\{A_p, R_p, A_q, R_q\}$	$(\{1,2\}, \{1,2\})$	$\emptyset$	merge$(\text{head}_p, \text{head}_q)$	$L(v_s) = \{A_p, A_q\} =$ intersection

Table 2: Every version of the defeater configuration, with its state computed and, for each of the four merges, the LCA-legality check.

5. p **removes, having seen only its own add.** p applies R_p at p_1 (it has not yet seen v_s): version p_2, $L = \{A_p, R_p\}$, state $(\{1\}, \{1\})$, live $\emptyset$. Visibility gained: $A_p \xrightarrow{\text{vis}} R_p$.

6. q **removes symmetrically.** q applies R_q at q_1: version q_2, $L = \{A_q, R_q\}$, state $(\{2\}, \{2\})$, live $\emptyset$. Visibility: $A_q \xrightarrow{\text{vis}} R_q$.

7. p **merges in the staging version.** p merges p_2 and v_s. Common ancestors: v_0, p_1; the LCA is p_1, and $L(p_1) = \{A_p\} = \{A_p, R_p\} \cap \{A_p, A_q\}$: LCA-legal. New head head_p, $L = \{A_p, R_p, A_q\}$, state $(\{1\}, \{1\}) \sqcup (\{1,2\}, \emptyset) = (\{1,2\}, \{1\})$, **live** $\{2\}$: p's own tag is dead, the merged-in A_q lives.

8. q **merges in the staging version.** Symmetrically: LCA q_1, $L(q_1) = \{A_q\} = \{A_q, R_q\} \cap \{A_p, A_q\}$, LCA-legal. Head head_q, $L = \{A_q, R_q, A_p\}$, state $(\{1,2\}, \{2\})$, **live** $\{1\}$.

9. **The critical merge.** The two heads merge. Common ancestors: v_0, p_1, q_1, v_s; all four reach v_s ($p_1 \to v_s$, $q_1 \to v_s$ are DAG edges), so LCA $= v_s$, uniquely, and

$$L(v_s) = \{A_p, A_q\} = \{A_p, R_p, A_q\} \cap \{A_q, R_q, A_p\} = L(\text{head}_p) \cap L(\text{head}_q) :$$

fully LCA-legal. The merged version $v_\top$ holds $L = \{A_p, R_p, A_q, R_q\}$ and state $(\{1,2\}, \{1\}) \sqcup (\{1,2\}, \{2\}) = (\{1,2\}, \{1,2\})$: **all dead**.

Nothing about this execution is dishonest or ill-formed: every merge sits at a unique LCA whose event set is exactly the intersection of the two sides' histories; timestamps are globally unique; visibility is exactly program order ($A_p \xrightarrow{\text{vis}} R_p$ and $A_q \xrightarrow{\text{vis}} R_q$ are the only edges, since each remove was issued before its replica saw v_s). The staging replica is the one construction the DAG discipline permits that one might not think of: it *realizes the intersection of two histories as an honest store version*, and thereby hands the final merge a fully legal LCA. All nine states are faithful folds, checked in the kernel (`sp_eq`, `sq_eq`, `LCA_eq`, `head_p_eq`, `head_q_eq`, `mergedState_eq`, `head_p_live`, `head_q_live`, `mergedState_live` in `Refutations/InterLca2op_Defeater_Arbiter.lean`).

2.3 The order facts: forced witnesses and the four-event cycle

We now compute what witnesses each version admits. The visibility edges are $A_p \xrightarrow{\text{vis}} R_p$ and $A_q \xrightarrow{\text{vis}} R_q$; both are non-commuting pairs, so both are order edges in every variant of the

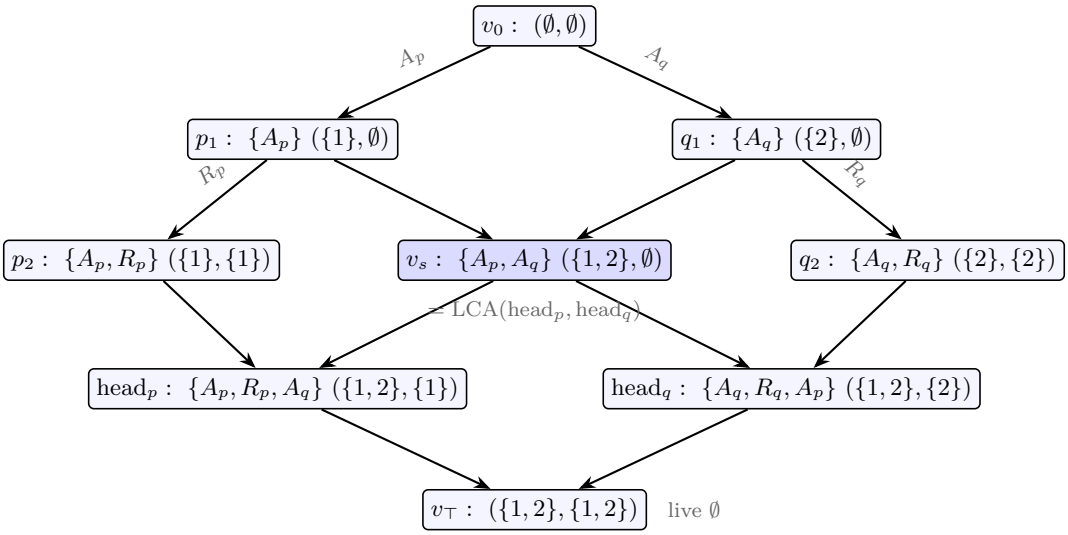


Figure 1: The defeater. Versions are labelled with their event sets and states (A, D); the shaded v_s is the staging version, an honest store version realizing $L(\mathrm{head}_p) \cap L(\mathrm{head}_q)$, so the final merge is fully LCA-legal. Live sets: head_p lives on $\{2\}$, head_q on $\{1\}$, $v_\top$ on nothing.

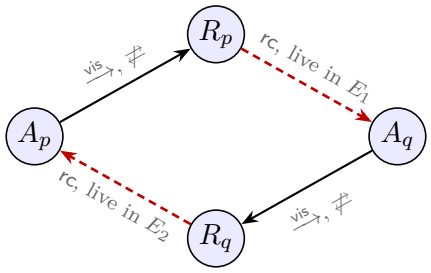
linearization order. The interesting edges are the rc edges between concurrent pairs: $R_p \parallel A_q$ and $R_q \parallel A_p$, both instances of $\mathsf{rem} \xrightarrow{\mathsf{rc}} \mathsf{add}$. Whether they survive depends on the absorber clause, and the absorber clause depends on *which event set you look at*:

- **Within** head_p**'s set** $E_1 = \{A_p, R_p, A_q\}$: the candidate absorber for the edge $R_p \to A_q$ would be a non-commuting vis-successor of A_q inside E_1; the only vis-successor of A_q anywhere is R_q, and $R_q \notin E_1$. The edge *survives*: $R_p \ \mathsf{lo}^{E_1}\ A_q$. Together with $A_p \to R_p$ this forces the order $[A_p, R_p, A_q]$: the *only* lo^{E_1}-respecting enumeration of E_1.
- **Within** head_q**'s set** $E_2 = \{A_q, R_q, A_p\}$: symmetrically, $R_q \to A_p$ survives ($R_p \notin E_2$), forcing $[A_q, R_q, A_p]$.
- **Within the union** $U = \{A_p, R_p, A_q, R_q\}$: both rc edges are *absorbed*: $A_q \xrightarrow{\mathsf{vis}} R_q$ with $A_q \not\rightleftarrows R_q$ and $R_q \in U$ kills $R_p \to A_q$; symmetrically R_p kills $R_q \to A_p$. So lo_m (equivalently lo^U: on U the configuration-wide and set-relative orders coincide, because the configuration has no events outside U) has exactly the two vis edges $A_p \to R_p$ and $A_q \to R_q$. Its maximal events are R_p **and** R_q**, and only they**.

The state side agrees with the order side, twice over. First, the forced side orders are also the only *state-correct* ones: brute-forcing the six enumerations of E_1,

$[A_p, R_p, A_q] \to (\{1,2\}, \{1\})$ ✓	$[A_p, A_q, R_p] \to (\{1,2\}, \{1,2\})$	$[A_q, A_p, R_p] \to (\{1,2\}, \{1,2\})$
$[A_q, R_p, A_p] \to (\{1,2\}, \{2\})$	$[R_p, A_p, A_q] \to (\{1,2\}, \emptyset)$	$[R_p, A_q, A_p] \to (\{1,2\}, \emptyset)$

only $[A_p, R_p, A_q]$ folds to head_p's state $(\{1,2\}, \{1\})$: **every witness of** head_p **ends in** A_q. Symmetrically every witness of head_q ends in A_p. Second, at the union: $v_\top$'s state is all-dead, and an enumeration of U ending in an add folds to a state in which that add's tag is live (nothing after it can kill it), so **every witness of** $v_\top$ **ends in** R_p **or** R_q: exactly the lo_m-maximal events, as it must be.



E_1 forces $A_p < R_p < A_q$; E_2 forces $A_q < R_q < A_p$.
In the union both rc *edges are absorbed, so the union order is satisfiable;*
but no union order restricts to both forced side orders.

Figure 2: The four-event cycle behind the burial. Solid edges are visibility edges (present in every event set containing both endpoints); dashed edges are rc edges, each live only in the side that lacks its absorber.

Now glue the two forced side orders (Figure 2): E_1 forces $A_p < R_p < A_q$, E_2 forces $A_q < R_q < A_p$. Their union is the cycle

$$A_p \xrightarrow{\text{vis}} R_p \xrightarrow{\text{rc (live in } E_1)} A_q \xrightarrow{\text{vis}} R_q \xrightarrow{\text{rc (live in } E_2)} A_p.$$

No total order on U restricts to both side orders. Consequently:

Claim 2.1 (the burial)**.** *No witness of $v_\top$ restricts, on both heads' event sets, to state-correct witnesses of the heads. Every valid union witness breaks at least one side: its restriction to that side's event set folds to the wrong state.*

This is checked in the kernel on a concrete instance: $w = [A_p, R_p, A_q, R_q]$ is a valid witness of $v_\top$ (it folds to the all-dead merged state, `w_fold`), yet its restriction to E_2 is $[A_p, A_q, R_q]$, which folds to $(\{1,2\},\{1,2\})$ and not to head_q's state $(\{1,2\},\{2\})$ (`crack1_witness`). The events the bottom-up induction must peel last, the removes, are *buried* mid-history inside the very sides that own them.

3 The critical merge: every move, one by one

3.1 What is owed

At step 9 the induction holds witnesses for head_p, head_q (and all earlier versions) and must produce, for the new version $v_\top$, a sequence π such that

(i) π enumerates $U = \{A_p, R_p, A_q, R_q\}$ without repetition;

(ii) π extends lo_m, whose edges are exactly $A_p \to R_p$ and $A_q \to R_q$;

(iii) π folds from σ_0 to $\sigma_{v_\top} = (\{1,2\},\{1,2\})$.

The derivation must build π back to front by peel steps on the concrete merge instance

$$\mathsf{merge}\big(\underbrace{(\{1,2\},\emptyset)}_{\sigma_{v_s}}, \underbrace{(\{1,2\},\{1\})}_{\text{head}_p}, \underbrace{(\{1,2\},\{2\})}_{\text{head}_q}\big) \;=\; (\{1,2\},\{1,2\}).$$

Before touching the rules, two facts pin down the last slot of π. By (ii), the last element must be lo_m-maximal: R_p or R_q (each add has an outgoing lo_m edge). By (iii), independently, the last element must be a remove (a trailing add leaves its tag live, §2.3). So **every derivation must begin by peeling a remove**, and the exhaustion below is finite.

One more preliminary, the algebraic heart of everything that follows (kernel: `awset_rem_output_empty`):

Lemma 3.1 (remove images are dead)**.** *For every state s and every remove event e, $\mathsf{live}(\mathsf{do}(s,e)) = \emptyset$: indeed $\mathsf{rem}(A,D) = (A, A \cup D)$ and $A \setminus (A \cup D) = \emptyset$. Hence a state with a nonempty live set is not a $\mathsf{do}(\cdot, \mathsf{rem})$-image of any state.*

Both heads have nonempty live sets: $\mathsf{live}(\mathrm{head}_p) = \{2\}$, $\mathsf{live}(\mathrm{head}_q) = \{1\}$. Now the moves.

3.2 The paper's own schedule lands on 2-OP

Computing the six-set partition of §1.2 for this merge: $L'_1 = \{R_p\}$, $L'_2 = \{R_q\}$; neither remove is lo_m-before any LCA event (their rc edges toward the adds are absorbed in lo_m), so $L^b_1 = L^b_2 = \emptyset$, hence $L^a_\top = \emptyset$, and

$$L^a_1 = \{R_p\}, \qquad L^a_2 = \{R_q\}, \qquad L^b_\top = \{A_p, A_q\}.$$

Both L^a_1 and L^a_2 are nonempty, so the proof's schedule (Table 1) prescribes BottomUp-2-OP as the first move, peeling the lo_m-maximal event of one of them. We check it first, then everything else.

3.3 Move 1: BottomUp-2-OP, peeling R_q from head_q

Instantiate $e_2 \coloneqq R_q$, $e_1 \coloneqq R_p$. The side condition holds: $R_p \neq R_q$ and $R_p \rightleftarrows R_q$ (removes commute). The conclusion requires writing the third argument as $e_2(b)$: a state b with $\mathsf{do}(b, R_q) = \mathrm{head}_q$. By Lemma 3.1 every such image has empty live set, and $\mathsf{live}(\mathrm{head}_q) = \{1\}$. **No such b exists; the rule cannot be instantiated.** Kernel: `no_rem_peelable_from_defeater_heads` (second conjunct).

3.4 Move 2: BottomUp-2-OP, peeling R_p from head_p (the other orientation)

Apply MergeCommutativity to swap the heads and peel $e_2 \coloneqq R_p$ with $e_1 \coloneqq R_q$. Side condition: holds, as before. Factorization: a state b with $\mathsf{do}(b, R_p) = \mathrm{head}_p$; $\mathsf{live}(\mathrm{head}_p) = \{2\} \neq \emptyset$. **No such b exists.** Kernel: `no_rem_peelable_from_defeater_heads` (first conjunct).

3.5 Move 3: BottomUp-2-OP, peeling an add

The heads *do* factor by adds:

$$\mathrm{head}_p = \mathsf{do}(p_2, A_q) \quad \text{since } \mathsf{do}((\{1\},\{1\}), A_q) = (\{1,2\},\{1\}), \qquad \mathrm{head}_q = \mathsf{do}(q_2, A_p),$$

and the side condition holds ($A_q \rightleftarrows A_p$). The rule's *equation* is even true here: both sides of

$$\mathsf{merge}(\sigma_{v_s}, \mathsf{do}(p_2, A_q), \mathsf{do}(q_2, A_p)) \;=\; \mathsf{do}\big(\mathsf{merge}(\sigma_{v_s}, \mathsf{do}(p_2, A_q), q_2), A_p\big)$$

evaluate to $(\{1,2\},\{1,2\})$ (left: the merged state; right: merge gives $(\{1,2\},\{1\}) \sqcup (\{2\},\{2\}) = (\{1,2\},\{1,2\})$, and A_p re-inserts tag 1, already present). But a peel is not only an equation: it

commits the peeled event to the last slot of π. Peeling A_p makes π end in A_p while R_p remains in the prefix, and A_p lo_m R_p: condition (ii) fails. Independently, condition (iii) fails: a π ending in an add folds to a live state, not to $(\{1,2\},\{1,2\})$; the equation above is consistent with this because the *recursive* obligation it leaves behind, linearizing $\mathsf{merge}(\sigma_{v_s}, \mathrm{head}_p, q_2)$ over the event multiset $\{A_p, R_p, A_q\} \cup \{A_q, R_q\}$, is not a witness obligation for $U \setminus \{A_p\}$ at all (A_p is still inside head_p; the event sets no longer partition). The paper's own induction never proposes this move: it peels only lo_m-maximal events, and neither add is lo_m-maximal. **The only factorizations that exist peel events that may not be peeled.**

3.6 Move 4: BottomUp-1-OP

The rule is licensed when exactly one of L_1^a, L_2^a is empty; here both are nonempty, so the schedule never reaches it. Read as a bare equation anyway: its conclusion peels a local event e_1 with the a-side factored as $e_1(a)$, under a shared trailing LCA event $e_\top$ on the LCA slot and the b-side. The shared-event shape is satisfiable ($e_\top = A_p$: $\sigma_{v_s} = \mathsf{do}((\{2\},\emptyset), A_p)$ and $\mathrm{head}_q = \mathsf{do}(q_2, A_p)$), but the peeled e_1 must be a lo_m-maximal event of the remaining side, i.e. R_p, and the factorization $\mathrm{head}_p = \mathsf{do}(a, R_p)$ is impossible by Lemma 3.1. Peeling $e_1 = A_q$ instead repeats Move 3's failure. **Dead in both instantiations.**

3.7 Move 5: BottomUp-0-OP

The rule peels one shared LCA event $e_\top$, present outermost on *all three* arguments.

- $e_\top$ a remove: the LCA slot must factor, $\sigma_{v_s} = \mathsf{do}(l, \mathsf{rem})$; but $\mathsf{live}(\sigma_{v_s}) = \{1,2\} \neq \emptyset$, impossible by Lemma 3.1. (A remove is not even in $L_\top$, so this instantiation was doubly unavailable.)
- $e_\top = A_q$ (or symmetrically A_p): all three factorizations exist, $\sigma_{v_s} = \mathsf{do}((\{1\},\emptyset), A_q)$, $\mathrm{head}_p = \mathsf{do}(p_2, A_q)$, $\mathrm{head}_q = \mathsf{do}((\{1\},\{2\}), A_q)$; the rule instantiates. But it commits A_q to the last slot of π, and A_q lo_m R_q with R_q unplaced: (ii) fails, and (iii) fails as in Move 3. The schedule agrees: 0-OP is reached only once $S_3 = L_1^a \cup L_2^a$ is exhausted, and $S_3 = \{R_p, R_q\}$ can never be exhausted (Moves 1 and 2).

Dead in all instantiations.

3.8 Move 6: MergeIdempotence and MergeCommutativity

MergeIdempotence requires all three arguments equal; here they are pairwise distinct $((\{1,2\},\emptyset), (\{1,2\},\{1\}), (\{1,2\},\{2\}))$. Not applicable. MergeCommutativity is not a peel; it only swaps orientations, and both orientations of every peel were covered above.

3.9 Move 7: the automation catalogue

Theorem 2's catalogue (§1.3) discharges the same rules by induction over the event sets; if no instantiation of a rule exists, no amount of automation of that rule helps. The kernel makes this concrete for the catalogue's central member. The arbiter file `Refutations/InterLca2op_Defeater_Arbiter.lean` (written as a neutral referee for a genuine dispute about exactly this point) proves:

- `crux_no_rem_peel_from_defeater`: in the `inter_lca_2op` conclusion shape $\mathsf{merge}(\mathsf{do}(l, o_l), \mathsf{do}(\mathsf{do}(a, o_l), o_1), \mathsf{do}(\mathsf{do}(b, o_l), o_2))$, none of the four assignments (peeled slot o_1 or context slot o_2; matched to head_p or head_q) can place a remove in the outermost position of a head: all four are refuted by Lemma 3.1.
- `no_inter_lca_2op_rem_peel_of_defeater`: therefore there is *no* instantiation of `inter_lca_2op`'s conclusion matching the defeater's merge $\mathsf{merge}(\sigma_{v_s}, \mathrm{head}_p, \mathrm{head}_q)$, in either head-to-slot orientation, in which the peeled operation o_1 is a remove.

The remaining catalogue members (`base_2op`, `ind_left/right_2op`, `inter_left/right(_base)_2op`, and the whole 1-OP/0-OP families) share the same do-outermost peel shape, so the same image fact closes them; the 1-OP/0-OP specifics were covered in Moves 4 and 5.

3.10 Move 8: fall back on the induction hypothesis directly

One might hope to bypass the rules: the sides are linearizable by assumption, so take their witnesses and interleave them into a union witness. Claim 2.1 closes this: the side witnesses are *unique* ($[A_p, R_p, A_q]$ and $[A_q, R_q, A_p]$) and no interleaving of them is even an order (the four-event cycle of Figure 2); conversely, every valid union witness restricts to a state-*incorrect* sequence on at least one side (kernel: `crack1_witness`). The induction hypothesis is not too weak in degree; it is the wrong *kind* of object: witnesses of the sides are not raw material for a witness of the union.

3.11 Interim summary

Finding. *At the defeater's final merge, the last slot of any witness must hold R_p or R_q (by order and by state); every rule of the published toolbox that could place a remove there requires a head, or the LCA state, to be a remove-image, and all three states have nonempty live sets while every remove-image is dead (Lemma 3.1); the only factorizations that exist peel adds, which are not lo_m-maximal and fold wrong; and no assembly of the (unique) side witnesses is an order. Every rule in the published toolbox has been tried, in every orientation, and each fails for the stated reason. This, precisely, is what "the defeater defeats the soundness proof" means.*

3.12 The repairs that keep the peel, and their refutations

The exhaustion above is relative to the published rule set. Three natural repairs suggest themselves, each keeping the peel-and-recurse architecture while strengthening what it may assume. All three are refuted in the kernel. The refutations are stated not on the AWSet defeater but on a four-event, two-key configuration K_2 distilled from the same cycle motif; we present it first and are explicit below about what each theorem does and does not cover.

The kill-test configuration K_2. Two keys x, y; four operations $\mathsf{add}_x, \mathsf{rem}_x, \mathsf{add}_y, \mathsf{rem}_y$; state a pair of per-key flags; same-key rem/add do not commute and are rc-ordered ($\mathsf{rem}_k \xrightarrow{\mathsf{rc}} \mathsf{add}_k$, add wins), cross-key operations commute (Lean: `K2` in `Refutations/Reunification_Peel_Obstruction.lean`; it satisfies the update-layer VC bundle, `K2_updateVCs`, so it is inside the theory's class). Replica p runs A_y then R_x (so $A_y \xrightarrow{\mathsf{vis}} R_x$); replica q runs A_x then R_y; no communication. Take $U = \{A_y, R_x, A_x, R_y\}$: it is *fully causally*

closed (every vis-predecessor of a member is a member). The rc edge $R_x \to A_x$ *survives in* lo^U, because A_x's only vis-successor R_y is cross-key and commutes with it, so the absorber clause cannot fire; symmetrically $R_y \to A_y$ survives. The result is the cycle

$$A_y \xrightarrow{\mathsf{vis}} R_x \xrightarrow{\mathsf{lo}^U} A_x \xrightarrow{\mathsf{vis}} R_y \xrightarrow{\mathsf{lo}^U} A_y$$

(kernel: `the_cycle`): the same alternating vis/rc cycle as Figure 2, pushed one level deeper. In the AWSet defeater the two rc edges are live only per-side and absorbed in the union; in K_2 the cross-key commutation defuses the absorbers, so the cycle survives *in the union order itself*. K_2 is the minimal habitat in which peel-style repairs must operate, and where they die.

Repair A: strengthen the bookkeeping to full causal closure. Store versions really are fully causally closed (the published proof itself uses this), so one might re-run a peel induction that maintains full closure of the shrinking event set, hoping the stronger invariant licenses stronger per-datatype facts. A closure-preserving peel of e from a fully closed U needs e to be simultaneously lo^U-maximal (placeable last) and *vis*-maximal (removing it must not orphan a vis edge). On K_2's U: the adds fail vis-maximality (each has a program-order successor) and the removes fail lo^U-maximality (each has a surviving rc edge). **No event of U is peelable** (kernel: `no_peelable_event`), and the packaged form

`reunification_peel_obstruction` (`Refutations/Reunification_Peel_Obstruction.lean`)

exhibits the signature, the configuration (transitive, irreflexive vis), and the nonempty fully closed $U \subseteq C.\mathsf{events}$ with no such event. Any closure-maintaining single-event peel induction is stuck at this U, whatever VCs it carries.

Repair B: peel a block instead of an event. Perhaps a *set* of events can be moved to the end of the enumeration jointly (on the defeater one is tempted by $\{R_p, R_q\}$). Two generic pins (kernel: `diff_fullClosure_iff`, `block_suffix_no_exit`): a back-block B preserves full closure iff B is vis-upward-closed within U, and joint late-placement forces no lo^U edge to leave B. On K_2's U these two conditions chase each other around the cycle: vis edges propagate membership by upward closure, surviving rc edges by exit-freeness, so *any* nonempty candidate block swallows all of U (kernel: `back_block_forces_all`), and

`no_proper_back_block`, `no_proper_front_block` (`Metatheory/JoinLemma3C.lean`)

conclude that no proper back-block and, by the direction-agnostic cycle, no proper front-block exists: block peels make no progress at any granularity, from either end. (Scope note, for honesty: on the AWSet defeater's own U the order-level back-block $\{R_p, R_q\}$ *does* exist, and indeed $[A_p, A_q, R_p, R_q]$ is a valid witness; what kills block peeling *there* is the same factorization fact as always, Lemma 3.1, since a published-style block peel must still extract the block's operations through merge from sides that are not remove-images. The K_2 theorem is the stronger, architecture-level statement: even the pure order-and-closure residue of block peeling, with all state content deleted, fails in general.)

Repair C: widen the induction class. If no peel preserves full closure, let the induction traffic in "fully closed minus what has already been peeled" (`AlmostClosed`: $U \setminus S$ with S upward-closed under lo^U). The order theory of this class is healthy (peels exist and stay in the

class: `AlmostClosed.peel`, `almostClosed_peel_exists`), but the induction on a *merge* needs both sides to decompose over one *common* fully closed U, and the Join hypothesis only supplies sides that are *individually* fully closed. On the kill-test: $ev_1 = \{A_y, R_x\}$ and $ev_2 = \{A_x, R_y\}$ are both fully closed, but any common U decomposing both forces $S_2 = \{A_y, R_x\}$, which the surviving cross-side edge $R_x \to A_x$ shows is *not* lo^U-upward-closed:

`killTest_no_common_U` (`Refutations/JoinLemma3F_Of_AlmostClosed.lean`).

The widened induction cannot be initialized from what a merge actually gives it. Concurrent rc edges between the two sides' exclusive events are exactly what closure, of any strength, does not control.

Scope of the three refutations. Repairs A–C and their refutations arose in the corrected theory's own development (the reunification study for the enable-wins flag, which needs full-closure hypotheses); they are cited here because they close, in machine-checked form, the natural "why not just…" branches that keep the peel architecture alive after §3.3–3.9. They are proved on K_2, not on the AWSet defeater; the two configurations share the alternating vis/rc cycle that is the common cause (Figure 2 vs. `the_cycle`), and K_2 is where that cycle penetrates the union order, which is what a peel must survive. Note also what the refutations do *not* say: K_2 itself is handled by the corrected weak-closure route (`weak_peel_exists`); only closure-*strengthened* peels die on it.

4 What the defeater does and does not defeat

Exactness matters here, because "defeats the soundness proof" is easily misread as "refutes the theorem." It does not.

The theorem survives on this execution. Every version of the defeater configuration is RA-linearizable. Witnesses, one per version, each checkable by the folds already computed in Table 2: $[\,]$ for v_0; $[A_p]$, $[A_q]$, $[A_p, A_q]$, $[A_p, R_p]$, $[A_q, R_q]$ for p_1, q_1, v_s, p_2, q_2; $[A_p, R_p, A_q]$ for head_p; $[A_q, R_q, A_p]$ for head_q; and for the critical $v_\top$ both $[A_p, A_q, R_p, R_q]$ and $w = [A_p, R_p, A_q, R_q]$ fold to $(\{1,2\}, \{1,2\})$ and extend lo_m (kernel, for w: `w_fold`). So the defeater is *not* a counterexample to the published Theorem 1 or Theorem 2 as statements about this datatype (the AWSet skeleton in fact satisfies a corrected contract and is linearizable at every reachable configuration, §5). Whether the published *statement* holds for every datatype satisfying its VCs is thereby left open by the defeater alone, refuted for no datatype and proved for none: what is refuted is the only proof on offer.

What is defeated, precisely.

1. **The derivation.** The bottom-up proof of Theorem 1 must, at the merge of step 9, produce a witness using the eight laws; §3 shows no first move exists. The proof, as written, does not establish its theorem. The automation of Theorem 2 automates the same rules and inherits the failure (§3.9).

2. **The proof's central invariant.** The induction manufactures merged witnesses out of *state-correct side witnesses*. On the defeater the side witnesses are unique, mutually cyclic,

Move	Why it fails here	Anchor
2-OP peel R_q from head_q	head_q has live $\{1\}$; remove-images are dead: no factorization	`awset_rem_output_empty`, `no_rem_peelable_…`
2-OP peel R_p from head_p (swapped)	same, live $\{2\}$	same, first conjunct
2-OP peel an add	factorization exists, but adds are not lo_m-maximal: witness stops extending lo_m, and folds live	computation, §3.5
1-OP	schedule: both L_i^a nonempty; bare equation: peeled local event must be a remove, no factorization	Lemma 3.1
0-OP	$e_\top$ a remove: LCA slot live $\{1,2\}$, not an image; $e_\top$ an add: not lo_m-maximal	Lemma 3.1, §3.5
MergeIdempotence	arguments pairwise distinct	Table 2
MergeCommutativity	not a peel; both orientations covered	
catalogue (`inter_lca_2op` etc.)	same rules, Skolemized; no instantiation with a peeled remove exists, either orientation	`no_inter_lca_2op_rem_peel_of_defeater`, `crux_no_rem_peel_…`
interleave the side witnesses	side witnesses unique and mutually cyclic; every union witness breaks a side	`crack1_witness`, Fig. 2
repair A: full-closure single peel	a fully closed U exists with no lo^U-maximal and vis-maximal event	`reunification_peel_obstruction`
repair B: block peel (back/front)	on the same U, every legal block is all of U	`no_proper_back_block`, `no_proper_front_block`
repair C: widened class (common U)	individually closed sides admit no common U: not initializable	`killTest_no_common_U`

Table 3: The exhaustion, summarized. Rows above the line are on the defeater itself; rows below are the architecture-level repairs, refuted on the K_2 kill-test (§3.12).

and no valid union witness restricts to both (Claim 2.1, `crack1_witness`): the invariant is not just unprovable here, it is false as a description of how the union witness relates to the sides.

3. **The configuration-wide order, as an internal tool.** With the paper's lo, the restriction to E_1 has no edge $R_p \to A_q$ (the absorber R_q exists in the configuration), so both $[A_p, R_p, A_q]$ and $[A_p, A_q, R_p]$ extend $\mathsf{lo}|_{E_1}$, yet they fold to different states (§2.3). Any internal lemma of the form "all order-respecting replays of a version agree" is false for lo; the corrected theory's set-relative lo^E exists precisely to make it true. (RA-linearizability itself only asserts existence of a witness, so this is a defect of the proof machinery, not of the definition; and since $\mathsf{lo} \subseteq \mathsf{lo}^E$, witnesses built against lo^E deliver the paper's definition verbatim.)

4. **Every peel-shaped repair** that Table 3 lists: closure-strengthened single peels, block peels in both directions, and the widened-class initialization, each by a named kernel theorem on K_2.

Adjacent but distinct: the store-level LCA lemma ($L(v_\ell) = L(v_1) \cap L(v_2)$) is *true* and the defeater exercises it honestly at all four merges; its published proof has a sepa-

rate, orthogonal gap (a missing generator-uniqueness induction), mechanized and repaired in `Metatheory/LCA_Lemma.lean` and discussed in `sal-mrdts.pdf` §4.1. Nothing in this document depends on that gap.

5 How the corrected theory handles the same execution

The corrected metatheory (`sal-mrdts.pdf`, Part I, §5–§8) draws the architectural conclusion of Claim 2.1: witnesses for a merged version cannot be manufactured out of witnesses for its sides, so the proof must not traffic in witness lists at all. Its objects are *canonical states*: the update layer proves (from the three update-layer VCs, now with the set-relative lo^E) that all lo^E-respecting enumerations of a finite E fold to the same state, written $\sigma(E)$; the merge layer then owes exactly one equation per merge, the ternary *Join Lemma*

$$\sigma(E_1 \cup E_2) \;=\; \mathsf{merge}\big(\sigma(E_1 \cap E_2),\, \sigma(E_1),\, \sigma(E_2)\big),$$

whose first argument is, by the LCA lemma, exactly the state the store hands the merge. The per-datatype contract behind it is eight VCs (`sal-mrdts.pdf` §7): the three update-layer laws (RC-NON-COMM with a different-replica guard, NO-RC-CHAIN, COND-COMM), merge symmetry, and four merge-layer laws quantified over *canonical* tuples at honest LCAs: the feasible unit law, two *redistribution* identities (LOCAL-REDISTRIBUTE and REDISTRIBUTE, the delta contract), and the *causal-delta* equation

$$\mathsf{CDVC3}: \qquad \mathsf{merge}\big(B,\; A,\; e(B)\big) \;=\; e(A)$$

for e a lo^U-maximal event of a backward-closed U, with $A = \sigma(U \setminus \{e\})$ and $B = \sigma(\downarrow e \setminus \{e\})$ the canonical state of e's own causal past: "what an event does is determined by what it saw." The master induction (`join_lemma3_of_cd_feasible`, `Metatheory/Adequacy.lean`) proves the Join Lemma from these by strong induction on $|E_1 \cup E_2|$, and the transition induction (`ra_linearizable_of_core_feasible_cd3`) maintains "every store version holds σ of its event set," which is RA-linearizability per version. We now run that induction on the defeater's critical merge and watch it construct what §3 proved unreachable.

5.1 The same merge, discharged in five lines

The instance: $E_1 = \{A_p, R_p, A_q\}$, $E_2 = \{A_q, R_q, A_p\}$, $E_1 \cap E_2 = \{A_p, A_q\}$, $U = E_1 \cup E_2$ all four events. The canonical states (all computed in §2; the unique respecting enumerations are the forced witnesses):

$$s_0 = \sigma(E_1 \cap E_2) = (\{1,2\}, \emptyset), \quad s_1 = \sigma(E_1) = (\{1,2\}, \{1\}), \quad s_2 = \sigma(E_2) = (\{1,2\}, \{2\}).$$

Goal: $\mathsf{merge}(s_0, s_1, s_2) = \sigma(U) = (\{1,2\}, \{1,2\})$.

Step 1: select a lo^U-maximal event (`exists_loOn_maximal_u`). The maximal events are R_p and R_q (§2.3); take $e = R_q$. Note what just happened: the corrected induction picks its pivot by the *union's* order, where R_q's rc edge is absorbed; it is entirely indifferent to the fact that R_q is buried inside E_2's own order. Its data:

$$\downarrow R_q = \{A_q, R_q\}, \qquad B = \sigma(\downarrow R_q \setminus \{R_q\}) = \sigma(\{A_q\}) = (\{2\}, \emptyset), \qquad e(B) = R_q(B) = (\{2\}, \{2\}),$$

$$A = \sigma(U \setminus \{R_q\}) = \sigma(\{A_p, R_p, A_q\}) = (\{1,2\}, \{1\}) \quad (= s_1 : \text{here } U \setminus \{R_q\} \text{ happens to equal } E_1).$$

Step 2: decompose the side that owns e (`side_decompositionF`). Since $R_q \in E_2$ only, the induction rewrites E_2 as a *join of event sets*, $E_2 = (E_2 \setminus \{R_q\}) \cup {\downarrow}R_q$ with intersection ${\downarrow}R_q \setminus \{R_q\}$, and applies the induction hypothesis at this smaller union (three events) to get

$$s_2 \;=\; \mathsf{merge}\big(B,\; t_2,\; R_q(B)\big), \qquad t_2 = \sigma(E_2 \setminus \{R_q\}) = \sigma(\{A_q, A_p\}) = (\{1,2\}, \emptyset).$$

Numerically: $\mathsf{merge}((\{2\},\emptyset),(\{1,2\},\emptyset),(\{2\},\{2\})) = (\{1,2\},\{2\}) = s_2$. ✓ This is the move the published system does not have: it does not ask how s_2 *factors through* do (it does not, by Lemma 3.1); it decomposes E_2 *through the join*, with the merge's own LCA slot (B, the state of R_q's causal past) doing the bookkeeping. Inside the recursive call, the union E_2 selects *its own* maximal event, which is A_p (in lo^{E_2} the edge $R_q \to A_p$ is live, so A_p is last), i.e. the recursion peels on that level exactly the event the forced witness $[A_q, R_q, A_p]$ ends with. Set-relativity is what lets different levels peel different events.

Step 3: redistribute the delta (`FeasibleDeltaVCs3.feasible_local_redistribute`). Read $c = R_q(B)$ as a delta relative to $m = B$. The LOCAL-REDISTRIBUTE law $\mathsf{merge}(l, \mathsf{merge}(m, x, c), y) = \mathsf{merge}(m, \mathsf{merge}(l, x, y), c)$, instantiated at $l = s_0$, $x = t_2$, $y = s_1$ (using merge symmetry to place the decomposed side), turns the goal's left side into

$$\mathsf{merge}(s_0, s_1, s_2) = \mathsf{merge}\big(s_0,\; s_1,\; \mathsf{merge}(B, t_2, R_q(B))\big) = \mathsf{merge}\big(B,\; \mathsf{merge}(s_0, s_1, t_2),\; R_q(B)\big).$$

Step 4: the inner join, by the induction hypothesis. The inner merge is a Join Lemma instance at $E_1 \cup (E_2 \setminus \{R_q\}) = U \setminus \{R_q\}$ with intersection $E_1 \cap (E_2 \setminus \{R_q\}) = \{A_p, A_q\}$: three events, strictly smaller, so the IH gives

$$\mathsf{merge}(s_0, s_1, t_2) = \sigma(U \setminus \{R_q\}) = A.$$

Numerically: $\mathsf{merge}((\{1,2\},\emptyset),(\{1,2\},\{1\}),(\{1,2\},\emptyset)) = (\{1,2\},\{1\}) = A$. ✓

Step 5: close with the causal-delta equation and snoc. The CDVC3 VC at U, $e = R_q$ (its hypotheses are exactly satisfied: U backward-closed, R_q maximal, A and B canonical):

$$\mathsf{merge}\big(B, A, R_q(B)\big) = R_q(A), \qquad \text{numerically:}$$
$$\mathsf{merge}\big((\{2\},\emptyset),(\{1,2\},\{1\}),(\{2\},\{2\})\big) = (\{1,2\},\{1,2\}) = R_q((\{1,2\},\{1\})). \checkmark$$

Finally $R_q(A) = \sigma(U)$ because A is canonical for $U \setminus \{R_q\}$ and R_q is lo^U-maximal (`isCanonicalState_snoc`): appending a maximal event to a canonical enumeration is again canonical. Chaining steps 2–5:

$$\begin{aligned}\mathsf{merge}(s_0, s_1, s_2) \;&=\; \mathsf{merge}\big(B,\; \mathsf{merge}(s_0, s_1, t_2),\; R_q(B)\big)\\ &=\; \mathsf{merge}\big(B,\; A,\; R_q(B)\big) \;=\; R_q(A) \;=\; \sigma(U). \qquad \blacksquare\end{aligned}$$

The witness object this constructs for $v_\top$ is any lo^U-respecting enumeration, concretely $[A_p, R_p, A_q]$ (the canonical enumeration of A's set) followed by R_q: that is, $w = [A_p, R_p, A_q, R_q]$, exactly the "tempting linearization" that §3 could see but never derive. The corrected proof derives it *without ever claiming* that w restricts to state-correct side witnesses (it does not: `crack1_witness`); the sides enter the argument only through their canonical *states* s_1, s_2 and the three merge-layer equations. The burial pathology has nothing left to grip.

5.2 Why the CDVC3 instance is true here, and in general

It is worth seeing why CDVC3 holds at step 5, since it is the one genuinely per-datatype fact consumed. For the add-wins skeleton, $\mathsf{merge}(B, A, R_q(B))$ has dead set $D_A \cup A_B \cup D_B$ while $R_q(A)$ has dead set $A_A \cup D_A$; the two agree iff every tag live in A was seen by R_q (is in A_B). That is false for arbitrary tuples, and true here *because* R_q *is* lo^U*-maximal*: a concurrent add in U either lies in $\downarrow R_q$ (then its tag is in A_B), or it has a non-commuting vis-successor in U that absorbs the rc edge (then it is already dead in A). On the instance: the concurrent $A_p \notin \downarrow R_q$, and indeed $1 \in D_A$, absorbed by R_p. This trichotomy is the uniform discharge recipe for CDVC3 across the catalogue, and it is precisely the feasibility conditioning the published catalogue's $\forall$-quantified VCs lack: the same equation quantified over all states is false (take A with a live tag R_q never saw and no absorber).

5.3 Where this instance sits in the corrected architecture

For the record, the route just walked is the weak-closure instance of the closure-indexed contract: the corrected Join Lemma is stated once as $\mathsf{JoinLemma3C}\,\mathcal{C}$, indexed by the closure predicate $\mathcal{C}$ the datatype declares (`Metatheory/JoinLemma3C.lean`); the add-wins family sits at weak closure (the hypotheses used in steps 1–5), the enable-wins flag at full closure, and the soundness bridge `ra_linearizable3_of_joinC` (`Metatheory/ConditionedContract.lean`) is uniform across strengths because store versions are fully closed and full closure implies every declared $\mathcal{C}$. That index exists *because* of §3.12: the refutations `reunification_peel_obstruction` / `no_proper_back_block` / `killTest_no_common_U` show the two strengths cannot be reunified into one induction, so the contract carries the index instead of pretending to. The production OR-set, whose kernel the defeater's skeleton is, is discharged end-to-end through this architecture (`ORSet_ra_linearizable3_eq`).

6 The lesson, in one paragraph

What the published toolbox could not see is that *being last* is not a property of an event, nor of a state, but of an event *relative to an event set*. Every published rule reads finality off a state factorization: "the side is $\mathsf{do}(b, e)$, so e may go last." The defeater manufactures, with one staging replica and full LCA legality, a merge whose sides are factorizable only by events that must *not* go last (the adds, absorbed in the union but binding per-side) while the events that must go last (the removes) are buried mid-history on their own sides and leave no trace in any factorization of the head states (remove-images are dead; the heads are live). Finality flips between E_1, E_2, and U because the absorber clause is set-relative, and a toolbox whose every move is surgery on side *witnesses* cannot follow it. The corrected theory follows it by construction: its induction is surgery on *histories* (decompose E_2 as $(E_2 \setminus \{e\}) \sqcup \downarrow e$, at the honest LCA $\downarrow e \setminus \{e\}$), its per-datatype laws are equations between canonical states of event sets, and the one place finality is consulted (lo^U-maximality, feeding CDVC3) names the set it is relative to.

Pointers. The general theory this document instantiates: `sal-mrdts.pdf`, Part I: §3 (the ternary setting and the set-relative order), §4.1 (the LCA lemma's separate gap), §4.2 (the defeater, compressed), §5 (canonical states and the Join Lemma), §6 (why the contract is a delta contract), §7 (the eight corrected VCs and the CDVC3 recipe), §8 (adequacy, the

closure-indexed contract, and the reunification finding), §9 (the discharged catalogue). The kernel anchors cited here: `Refutations/InterLca2op_Defeater_Arbiter.lean` (the defeater, its states, and the peel refutations), `Refutations/Reunification_Peel_Obstruction.lean` (the K_2 kill-test), `Refutations/JoinLemma3F_Of_AlmostClosed.lean` (the common-U refutation), `Metatheory/JoinLemma3C.lean` (block peels; the closure index), `Metatheory/Adequacy.lean` (the corrected master induction), all kernel-checked with clean axioms.