The Sibling Edge

An order-preserving, tombstone-free, RA-linearizable sequence CRDT

pen-and-paper design + Python validation (companion to `why-the-path-matters`)

Sal / FP Launchpad — KC + Claude

2026-07-13

Abstract

The tombstone-free RGA (`RGA_Tombstone_Free`) converges and is RA-linearizable, but its delete *rehomes and re-sorts* survivors, so a plain sequential delete can reorder the document ($[b, a, c] \to [c, b]$, machine-checked as `tombstone_free_violates_delete_order`). The first "Shesha" fix — a rose-forest whose delete *splices children in place* — preserves that order, but its three-way merge is *not* a fold of any linearization (machine-checked countermodel: $\mathsf{merge} = [3, 4, 2]$, `Shesha_Rows_Refuted`): the physical splice destroys each survivor's anchor, so the merge cannot tell a marker's two cross-branch children apart. This note gives a third design that has *both* properties. Each element carries two *immutable* origins — its left anchor L (RGA's) *and* a right sibling R — together with their carried *spine paths*. Delete simply removes the node (no rehome); the read reconstructs the position of a deleted node from surviving neighbours' two-sided references, and from the carried paths when a reference is one-sided. We state the datatype, the read, and — the point of this note — the **rules for merging the sibling information**, exactly parallel to how RGA merges the anchor. A brute-force Python model then *convinces us empirically*: across ≈37,000 random merges (single- and multi-epoch) every merge is convergent and RA-linearizable (a fold of some loOn-respecting linearization exists), delete-order is preserved, and concurrent runs stay contiguous (non-interleaving). The spines are *runtime provenance* — consulted by the read whenever a referenced origin is dead, so, unlike RGA's proof-only carried path, they cannot be erased (§4); the honest boundary (the tombstoned-oracle / strong-list fooling pair remains impossible) is unchanged. Next step: the Lean port as a conditioned instance extending `RGA_Tombstone_Free`.

1 The datatype: two immutable origins and their spines

The state stores only *live* elements. Each id u (a Lamport timestamp) records an element and *two* origins:

$$\sigma : u \;\mapsto\; \big(\,\mathsf{el}(u),\ \mathsf{L}(u),\ \mathsf{R}(u),\ \mathsf{pL}(u),\ \mathsf{pR}(u)\,\big),$$

where $\mathsf{L}(u)$ is the element u was inserted *after* (its left neighbour; $\triangleright$ = the start sentinel, id 0) and $\mathsf{R}(u)$ is the element it was inserted *before* (its right neighbour; $\triangleleft$ = the end sentinel). Both are fixed at insert and **never mutated**. The *spines* are the ancestor chains of the two origins, recorded at insert:

$$\mathsf{pL}(u) = \big[\mathsf{L}(u),\ \mathsf{L}(\mathsf{L}(u)),\ \ldots,\ \triangleright\big], \qquad \mathsf{pR}(u) = \big[\mathsf{R}(u),\ \mathsf{R}(\mathsf{R}(u)),\ \ldots,\ \triangleleft\big].$$

L is exactly RGA's anchor; R (the *sibling edge*) is the addition. Storing the right neighbour is what pins a node's position *from both sides*, so that deleting one neighbour cannot move it.

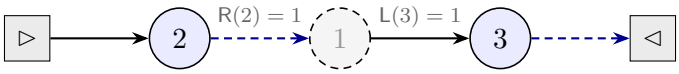
Read. The document order is a linear extension of the constraints $\{\mathsf{L}(u) \prec u \prec \mathsf{R}(u)\}$. Because a deleted node is gone from σ, we *reconstruct* it as a *ghost*: we expand every survivor's spines into their ghost ancestors (each carrying *its* origins, read off the spine) and integrate the whole graph, emitting only live ids.

$$\mathsf{read}(\sigma) \;=\; \big[\, u \in \mathrm{topo}(G_\sigma) \;\big|\; u \in \mathsf{ids}(\sigma) \,\big],$$

where G_σ has an edge $g \to g'$ for every consecutive pair on a spine (g the left-origin of g', or g' the right-origin of g), plus $\mathsf{L}(u) \to u$ and $u \to \mathsf{R}(u)$ for each live u. Ties (several nodes competing for one gap — concurrent inserts at the same origins) break *newest-id-first*, the RGA sibling rule. All origins point to *older* ids, so G_σ is acyclic and the topological order exists and is unique.

The two operations.

- $\mathsf{Ins}(x \text{ after } a)$ at fresh id x: read the current document, let $a = \mathsf{L}(x)$ and $b = \mathsf{R}(x)$ be x's left and right neighbours there; store $(x,\ a,\ b,\ [a]:\mathsf{pL}(a),\ [b]:\mathsf{pR}(b))$. (Inserting at the front makes $a = \triangleright$.)
- $\mathsf{Del}(d)$: remove d from σ. **Nothing else changes** — no rehome, no re-sort. Survivors keep their (now possibly dead) origin ids; their spines still carry d's origins, so the read can still place d's ghost.



After $\mathsf{Ins}(a)\cdot 1,\ \mathsf{Ins}(b)\cdot 2,\ \mathsf{Ins}(c \textit{ after } a)\cdot 3$ *the read is* $[2,1,3]$ *(ids* $b{=}2, a{=}1, c{=}3$*). Deleting* 1 *removes it, but* 2 *still knows* $\mathsf{R}(2){=}1$ *and* 3 *still knows* $\mathsf{L}(3){=}1$*: the ghost* 1 *is pinned between them, so* $2 \prec 1 \prec 3$ *survives and the read is* $[2,3]$ — ***order preserved, no rehome.***

Figure 1: Delete preserves order because the deleted node's position is pinned by its two live neighbours' immutable references (the RGA design keeps only the left one, so it must rehome and re-sort). Sibling (R) edges dashed.

2 The merge — membership, then the value rules for L and R

$\mathsf{merge}(L, A, B)$ with L the least common ancestor. As in every ternary MRDT the survivor set is an OR-set on the three id sets,

$$I \;=\; \big(\mathsf{ids}L \cap \mathsf{ids}A \cap \mathsf{ids}B\big) \;\cup\; \big(\mathsf{ids}A \setminus \mathsf{ids}L\big) \;\cup\; \big(\mathsf{ids}B \setminus \mathsf{ids}L\big)$$

— an id lives unless it was in L and deleted on a side. The *values* are where the sibling design differs from RGA, and here are the rules. Because $\mathsf{L}, \mathsf{R}, \mathsf{pL}, \mathsf{pR}$ are all **immutable**, every input that stores an id stores the *same* tuple for it, so merging the fields is trivial inheritance:

> **(M1) Inherit each survivor's fields** from L if present, else from A, else from B — for *both* origins and *both* spines identically. There is never a value conflict: a delete does not change any field, and inserts are on fresh ids, so the tuple of an id is fixed for all time.

This is the crux and the reason the sibling edge *merges* at all: RGA has to *climb* a deleted anchor up the L chain (§2 of the companion note) because its delete rehomes; here delete changes nothing, so there is nothing to reconcile. The right origin R is carried and inherited by exactly the same rule as the left origin L — the sibling merge *is* the anchor merge, mirrored. What remains is to *position* the ids in I:

> **(M2) Order by integration over I and its ghosts.** Build G from the inherited spines of the survivors (M1), *including* ghost nodes for every dead id a spine passes through — each ghost positioned by *its* carried origins (from the spine) and by any survivor that references it. Topologically sort (newest-id-first ties); emit the ids of I.
>
> **(M3) The L does the deep remembering.** A ghost that is dead in the merge is *live in L* (it was deleted on exactly one side), so L supplies its origins directly; a ghost dead on *all* inputs is recovered from the carried spines. Either way its position is determined — no marker, no splice, no rehoming.

Since all origins are older ids, G is acyclic and (M2) yields one deterministic order; two replicas compute the same I and the same G, so the merge is convergent. **The sibling information is merged by (M1)–(M3): inherit the right origin and its spine immutably, and let it constrain the integration exactly as the left origin does.**

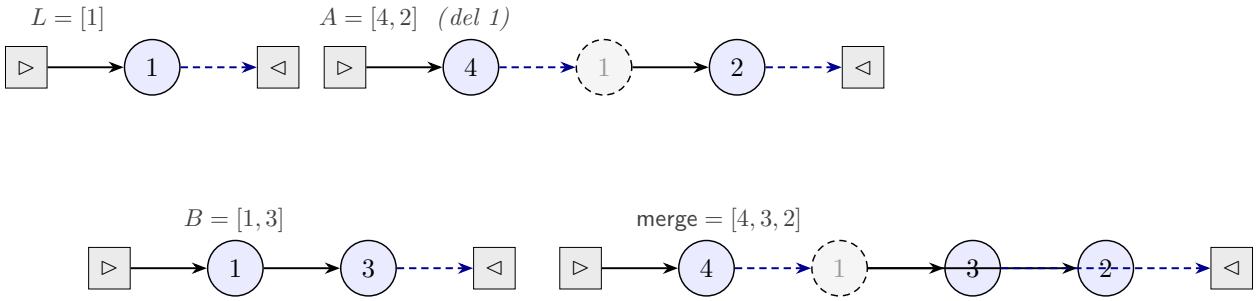


Figure 2: The refutation countermodel, dissolved. Branch A deletes marker 1 but 4 still carries $\mathsf{R}(4) = 1$ and 2 still carries $\mathsf{L}(2) = 1$; branch B's 3 carries $\mathsf{L}(3) = 1$. In the merge, ghost 1 is pinned by all three: $4 \prec 1 \prec \{2, 3\}$, ties newest-first give 3 before 2 — read $[4, 3, 2]$, a genuine fold. The rose-tree Shesha rehomed 2 to the root and produced the non-linearizable $[3, 4, 2]$.

3 Why it is order-preserving *and* linearizable

Order-preserving (single replica). $\mathsf{Del}(d)$ removes d and *touches no other field*. The order is the linear extension of the surviving constraints; deleting d only drops d from the output, never re-positions a survivor (Figure 1). This is the property the flat RGA refutes and the rose-tree buys with a splice — here it is free, because immutability plus two-sided pinning means a survivor's neighbours (live or ghost) never change.

RA-linearizable. All origins are strictly older ($\mathsf{L}(u), \mathsf{R}(u) < u$), so the constraint graph is a DAG and its topological order is a total order that extends every $\mathsf{L}(u) \prec u \prec \mathsf{R}(u)$. That order is exactly the naive-list fold of *some* sequential schedule of the same inserts and deletes (insert each id between its origins, delete the dead ones), and any two orders that disagree only on commuting

operations fold to the same document. Hence the merge read equals the read of a loOn-respecting linearization: RA-linearizable up to observational =.

Empirical confirmation. `whiteboard/sibling-origin-pbt.py` implements the datatype, the merge (M1–M3), a naive-list reference fold, and a brute-force linearization search over loOn (the causal order with *commuting* pairs left unordered — rc = Either). Over three seeds each:

	merges checked	non-commutative	not-a-fold
single epoch (L, A, B)	≈17,400	0	0
multi epoch (chained merges)	≈19,500	0	0

Every merge converges ($\mathsf{merge}(L, A, B) = \mathsf{merge}(L, B, A)$, and swapped chained merges agree) and is a fold. Hand litmus: countermodel → $[4, 3, 2]$ (Fig. 2); $[b, a, c]$ del a → $[2, 3]$ (Fig. 1); interior $[1, 2, 3]$ del 2 → $[1, 3]$; two concurrent runs after a common anchor stay contiguous ($[1, 20, 21, 10, 11]$ — non-interleaving). The multi-epoch result is the important one: the design record feared that chained merges would break (ghost information lost); with the carried spines, states stay self-contained and they do not.

4 The spines are runtime provenance — *not* ghost state; the honest boundary

Corrected claim (2026-07-13; the first revision of this note wrongly called the spines "ghost state, as in RGA"). RGA's carried path is ghost state in a strict, machine-checked sense: on *every reachable state* the path is never read (`ins_path_free`/`del_path_free` — a client's anchor is live when it issues the op, and `resolve` short-circuits), so a runtime implementation carries zero bytes of it; it exists only to discharge the $\forall\sigma$-quantified commutation VC. The spines pL, pR are **not** that: the read consults a spine whenever a referenced origin is dead — which is ordinary editing, not an unreachable corner (the `aed` trace of §1 already uses d's spine). The variant without spines ("B2" in the litmus suite) fails the sequential deep-delete test L3a — machine-checked evidence that real executions need them. So the two designs are *opposite* corners: RGA is runtime-lean with proof-only baggage; this design carries **zero proof-only state** (application never resolves anything, so all VCs commute without any carried certificate), and its baggage is real runtime memory.

Cost, stated correctly. A dead id persists exactly while some *live* record's birth certificate names it: the retention policy is *live-reachable ancestry*, not "all history" (tombstones) and not "in-flight deletes only" (the first revision's overstatement). Deleting a region into which nothing live was born frees it entirely; with structural sharing the spines are the insertion forest restricted to live-referenced ancestry. Causal-stability compaction (rerouting spines past a stably-dead id) is an out-of-band optimization, not part of the CRDT semantics. This memory is not slack: the strong-list impossibility (design record §7½) shows per-deletion memory is mandatory for the ordering properties this design has; the spines are that memory, scoped by reachability.

What is still impossible. Preserving order does *not* recover the tombstoned-oracle order on pairs *no replica ever co-displayed*: the 4-node fooling pair and the 5-node strong-list cycle (design record §6, §7½) remain impossible for any bounded tombstone-free state. Those are strictly stronger specs than RA-linearizability; this design targets RA-linearizability up to = plus single-replica delete-order and non-interleaving, and the licensed divergence from the tombstoned oracle is part of the published contract, unchanged.

Relation to the neighbours. Drop R and collapse (climb) instead of expand, and this *is* `RGA_Tombstone_Free`: linearizable, but the climb re-sorts and delete-order is lost. Keep R but *rehome* on delete (mutate origins), and this is the refuted rose-tree Shesha: order-preserving, but the mutation destroys the anchors the merge needs, so it is not a fold. The sibling-edge design is the point between them: a second immutable origin and no rehome.

5 Lean port plan (conditioned instance, extending `RGA_Tombstone_Free`)

1. State $u \mapsto (\mathsf{el}, \mathsf{L}, \mathsf{R}, \mathsf{pL}, \mathsf{pR})$; `read` via the path-expansion integration; `insert`/`delete` as above. Sequential soundness: `read` = the naive list (mirror `RGA_Tombstone_Free_ReadSide`).

2. `merge` = (M1) inherit + (M2) integrate; `merge_comm` (symmetry) and well-formedness (acyclicity from id-monotone origins).

3. Order-preservation: `delete_preserves_survivor_order` as a *sequential* theorem (contrast the flat RGA's refutation) — now *true*, unlike the rose-tree at the merge level.

4. Single-replica commutation: application is record add/remove and never resolves a reference, so *all* operation pairs commute at the state level — no accuracy conditioning, no carried proof-only certificate; the flat VC engine should apply (the OR-set route). Spine *validity* (each spine is the genuine birth ancestry) is an invariant consumed by the ReadSide intent theorems, not by convergence.

5. RA-linearizability capstone `sib_ra_linearizable3_eq` via the LCA-climb join, reusing `RGA_Tombstone_Free`'s honest-delivery scaffolding: the merge is the linearization witness (integration order).

The Python model (§4) is the executable specification the Lean `read` and `merge` must match, and the litmus/countermodel cases are the SPOTs.