What must be remembered about a deleted character?

v3: nothing beyond the coordinates — if the carve is timestamp-faithful
(working note — throwaway)

Sal / FP Launchpad — KC + Claude

2026-07-14

The question. When a character is deleted from a replicated list, what must the system remember about it, and for how long? The requirement throughout ("the contract"): once any user has seen character u displayed before character v, no user may ever see v before u; and replicas with the same events must agree.

Correction from v1. Version 1 argued that retention of a dead timestamp is forced whenever a deleted character has living descendants. That is true for designs whose merge *re-renders positions from identity keys*, but KC's proposal — keep every node's absolute range untouched through deletes and merges — refutes the general claim: on v1's own example it produces the right answer with *nothing* retained. This version records the corrected boundary, machine-checked: **distinct positions are permanent testimony; retention is forced only at ties.**

Encoding. Every node stores its full absolute range $Q = (lo, hi) \subseteq (0, 1)$. Insert takes the lower half of the remaining gap inside the parent's absolute range (first child of the root: $(0, \frac{1}{2})$). Display: depth-first, children by descending lo, equal lo broken by newer timestamp first. Characters are named by their timestamps (larger = later). Under KC's proposal, delete removes the record and re-parents the children; *no number ever changes.*

1 The four-event example — and why it forces nothing

step	action	screen	
1	type d (time 1)	d	
2	type a (time 4), same position	$a\ d$	
	— both replicas sync —	$a\ d$	
3	**Alice**: type x (time 6) *after* d	$a\ d\ x$	⇒ **Alice has seen a before x**
4	**Bob**: delete d	a	
5	*merge* M_1 (LCA = the synced version)	?	

Ranges: $d = (0, \frac{16}{32})$, $a = (\frac{16}{32}, \frac{24}{32})$, $x = (0, \frac{8}{32})$ (lower half *of* d). Four designs at M_1:

design	what survives d's death	M_1	verdict
RGA	d kept as a tombstone	$a\ x$	✓
always keep the dead time	stamp (1) on x	$a\ x$	✓
keep only if d conflicted	nothing (d never conflicted)	$x\ a$	× flips Alice's (a, x)
keep the ranges (KC)	**nothing — x stays at** $(0, \frac{8}{32})$	$a\ x$	✓

The keep-range row is the correction: x's position $(0, \frac{8}{32})$, below a's $(\frac{16}{32}, \frac{24}{32})$, *is* d's testimony — carried geometrically, permanently, for free. The conditional design fails here not because information must be retained but because it re-renders from keys and then forgets the key. This example refutes conditional forgetting in re-render designs; it does *not* force retention on position-keeping designs.

2 Keep-the-range survives everything the conditional designs failed

Machine-checked (regressions from `litmus/contest_tree.py` run against the keep-range design):

countermodel	killed	keep-range
4-event example (above)	conditional retention	survives: a x
subordination escape (7 events)	contest-only stamps	survives: $(4, 6)$ order preserved
retroactive subordination (6 events)	condition-on-current-level stamps	survives: $(3, 2)$ order preserved

The reason is uniform: in all three histories every deciding comparison is between *distinct* ranges, and kept ranges never change — so every verdict ever displayed is re-derivable from live positions alone, at every replica, forever.

3 The one failing shape: ties

Concurrent characters typed at the same position from identical states mint *identical* ranges — position carries zero information between them. Machine trace (the L25 shape, three branches):

step	state
R_0: type 6 at root	$6 = (0, \frac{16}{32})$
R_M (concurrent): type 10 at root	$10 = (0, \frac{16}{32})$ ← **identical mint: a tie**
M_1 = merge: tie broken by time	display $[10,\ 6]$
type 16, 22 under 6	$16 = (0, \frac{8}{32})$, $22 = (\frac{8}{32}, \frac{12}{32})$, both $\subset (0, \frac{16}{32})$
	display $[10,\ 6,\ 22,\ 16]$ (10 before 22 and 16)
R_D (from R_0): delete 6	
final merge, ranges kept	display [**22**, **16**, **10**] **both pairs flipped**

The mechanism: 6's range *equals* 10's, so "inside 6" is numerically also "inside 10" — 22's kept range has $lo = \frac{8}{32} > 0 = 10$'s lo, and the position sort puts 22 *above* 10. The verdict "10 before 6" was a fact about the *pair of timestamps*, decided at M_1; the heirs' sub-ranges nest inside *both* rivals' equal ranges and cannot stay subordinate by geometry once their side of the tie dies. At random-DAG scale keep-range fails 52/120 executions — this shape is the conjectured root of all of them (minimization pending).

4 The sibling-chain connection

The Q-ranges and the sibling links are two encodings of one structure, and the tie analysis says exactly where they coincide and where they part.

sibling-link view	Q-range view
node's birth pointers: (parent, displaced sibling)	node's mint range
distinct chain positions	distinct ranges
two nodes displacing the same head	**identical mint ranges — a tie**
splicing a dead node out of the chain	deleting a record, ranges kept
recording the spliced link through a dead node	the tie stamp / dig

The carve is deterministic: same parent and same displaced head ⇒ the same slice. So *tie-ness in ranges is exactly equality of the birth pointers* — a tie is two nodes claiming the same chain position, and the range encoding is the sibling chain *arithmetized*. Both encodings order distinct chain positions for free and both run out of positional information at the same place: a shared chain position, where only timestamps can decide. And the retention question is the same question in both wardrobes: keeping the tie verdict for a dead node's heirs *is* keeping the spliced-out link through the dead node.

The two encodings genuinely differ in one behavior (machine-checked, `litmus/sibling_tree.py`): what happens to the *displacers* of a tied node. Branch A stacks 2 then 5 over 1; branch B concurrently stacks 4 over 1 — so 2 and 4 tie (both displaced 1), and 5 displaced 2. The range read sorts the level globally: $[5, 4, 2, 1]$ (5 is newest, floats to the top). The sibling read keeps 5 glued above the 2 it displaced: $[4, 5, 2, 1]$. Sequentially the encodings agree on 300/300 random histories; they part exactly at ties. Run-gluing is the Fugue-direction behavior — a different datatype, not a different encoding of this one.

5 The sharpened claim

Retention is forced only at ties. Distinct concurrent mints are ordered by permanent geometry — equivalently, by their positions in the sibling chains. Identical concurrent mints (shared chain positions) are ordered by a timestamp verdict that some state must carry for as long as either side has living descendants — as a tombstone (RGA), a stamp (always-keep), a retained spliced link (sibling chains), or any equivalent.

This re-opens a door the conditional-retention refutations appeared to close. Their principle was: *the decision whether heirs inherit a dead rank must be a function of immutable data* — "was it contested" and "was it subordinated" are mutable, concurrently accruing knowledge, and both conditions fell to countermodels. But **tie-ness is immutable**: mint ranges (equivalently, birth pointers) are never rewritten, so "u and v claimed the same position" is a fixed fact about the pair. A design that keeps ranges verbatim and stamps heirs *only when their dead ancestor was tied* is therefore a live candidate, not a refuted one. Its retention would be O(tie families): zero on a purely sequential million-character history, growing only with genuine same-position races.

Pre-registered risks (the machine decides; not yet run). *(i) Posthumous ties*: a later insert can coincidentally mint a dead node's exact range (machine-observed) — though such pairs were never co-displayed, so their order may be legitimately free. *(ii) Tied orphans*: a tie whose dead side was born and deleted inside one branch, invisible to the merge that meets the survivor. *(iii) Mixed-comparator transitivity*: position where distinct, stamps at ties — the combined relation must remain a total order. *(iv) Nested tie families*: heirs of both sides of a tie meeting each other.

(v3, same day: the box above was never run — it was overtaken. KC: "we don't need metadata for races … the ranges fully capture it … it may be the case you don't see it or your merge may be

wrong." Correct. The next section removes the box's hidden premise, and all four risks dissolve with it.)

6 v3: the tie premise falls — timestamp-faithful carves

The boxed claim quantifies over designs whose carve is **timestamp-oblivious**: the quarter carve is a deterministic function of the seen state only, so concurrent same-slot mints produce *identical* ranges. Ties are not intrinsic to races — they are *manufactured by the carve*, which discards the one datum that distinguishes the rivals. Embed it instead. Mint, relative to the anchor:

$$I(t) \;=\; \Big(1 - 2^{-t},\;\; 1 - \tfrac{3}{4}\,2^{-t}\Big).$$

Everything else is unchanged — in particular **delete is the delta tree's isometric fold, verbatim**: the dead node's interval is composed into its children's, an exact affine substitution, so every survivor's absolute interval is untouched (the machine sentinel: the merge asserts that all three inputs agree on every survivor's absolute; it has never fired). The merge is OR-set survival plus re-anchoring to the nearest surviving ancestor; *it contains no repair and never decides an order.*

Theorem 1 (mint disjointness and order). For $s \neq t$, $I(s)$ and $I(t)$ are disjoint with a gap, and $s < t \Rightarrow I(s)$ lies entirely below $I(t)$. *Proof.* WLOG $s < t$, so $t \geq s + 1$. The claim is $\mathrm{hi}(s) < \mathrm{lo}(t)$, i.e. $1 - \frac{3}{4}2^{-s} < 1 - 2^{-t}$, i.e. $2^{\,t-s} > \frac{4}{3}$ — true since $2^{\,t-s} \geq 2$. Monotonicity of lo gives the order. □

Timestamps are globally unique, so *ties cannot exist*: no repair ever runs, no interval is ever rewritten, and a node's absolute interval is a **birth constant**

$$\alpha(u) \;=\; \varphi_{c_1} \circ \varphi_{c_2} \circ \cdots \circ \varphi_{c_k}\big(I(t_u)\big),$$

the affine composition of the mints along u's *birth chain* $c_1 c_2 \cdots c_k u$ (where φ_v maps $(0,1)$ onto $I(t_v)$, order-preservingly).

Theorem 2 (no overlap, globally and permanently). For any two distinct nodes u, v, either one's birth chain is a prefix of the other's (ancestor and descendant: α nested), or $\alpha(u) \cap \alpha(v) = \varnothing$. In particular, any two nodes sharing a *current* parent have disjoint intervals. *Proof.* If neither chain is a prefix of the other, let k be the first level where they differ, with distinct timestamps $s \neq t$ there. Both α's factor through the shared prefix $\varphi_{c_1} \circ \cdots \circ \varphi_{c_{k-1}}$, which is a monotone injection; inside it, one continuation lies in $I(s)$ and the other in $I(t)$, disjoint by Theorem 1 — and affine images of disjoint intervals are disjoint. For the "in particular": two nodes with the same current parent p cannot be chain-comparable — if u lay on v's birth chain and both were live, then v's nearest live birth-ancestor would be u or deeper, not p above u. And folds preserve α exactly (isometry), so the disjointness is permanent, not a mint-time accident. □

Corollary (the read needs no tie rule). Sibling intervals never tie, so the id tiebreak in the sort is dead code — machine-confirmed by an instrumented sweep asserting pairwise disjointness at *every read of every execution*: clean over $120 + 200$ randomized DAG executions.

Corollary (display ≡ chain-lex). Sibling order under any current parent is descending timestamp *at the first level where the birth chains diverge*: survivors by their own ts, folded heirs by their dead founder's — the subordination the countermodels demand, read off geometry. Instrumented check over 19,531 multi-member sibling groups: own-ts order violated 3,481 times (all post-fold heir groups — forcing own-ts there is exactly the dissolution failure), **birth-chain lex order violated 0 times**. Display order *is* the lex order on birth chains; the geometry realizes it with zero retained metadata.

Machine verdicts. The credential shape (a tie's heirs meeting a mid-timestamp third party under death-before vs. death-after topologies) plus the full gauntlet:

	RGA$_\dagger$	keep-range + o.-p. repair	embed
credential countermodel	✓	× diverges, 5 flips	✓(= RGA$_\dagger$'s read)
L25-third-party / CE-escape / CE-retro	✓	× / ✓/ ✓	all ✓
litmus battery	✓	—	clean exc. one-sided L19
DAG PBT	✓	28/120 fail	**120/120, 300/300 clean**
state beyond live (par, lo, hi)	tombstones	none	**none**

The control column matters: order-preserving repair is the strongest form of "the ranges capture the settled order" *without* changing the mint, and it fails exactly where predicted — ranking a fresh claimant against a dead founder's heirs needs the founder's rank, which a ts-oblivious state no longer holds and ts-faithful geometry does. All four pre-registered risks dissolve: identical ranges require identical timestamps, so posthumous ties, tied orphans, mixed comparators, and nested tie families have no instances.

The conservation law (what replaces the box).

> **Nothing must be remembered about the dead beyond the coordinates of the living — provided mints are timestamp-faithful.** The order information is conserved, not eliminated: it moves into coordinate precision. Ties force *metadata* only under timestamp-oblivious minting; under timestamp-faithful minting the same bits are carried *inside* the ranges — as a stamp, a link, a tombstone, or denominator bits, the verdict persists somewhere as long as it matters.

Measured on the 1000-character chain-then-delete-all: the survivor's denominator is $\sim 2^{502501}$ — $\approx \sum_i (t_i + 2)$ bits, i.e. the sum of the timestamps folded through, ∼61 KB for one character — versus 2^{2001} for the ts-oblivious quarter carve (the width+2 law). The carve choice is an information trade, not an escape: quarter carve = cheap and lossy (ties, hence metadata elsewhere); ts-embed = complete and exact (metadata inside the number). The open question is no longer *whether* to retain but *how to compress*: stability-gated renumbering of birth-constant coordinates without re-opening the mutable-geometry hole.

IEEE 754 floats as the range carrier (KC's question)

Binary64 values *are* dyadics — $\pm m \cdot 2^e$, $m < 2^{53}$ — so floats are the right *shape* for embed coordinates (every value in the design is dyadic; exact `Fraction`s here never need gcd). What the fixed 53-bit window costs, machine-measured (`embed_tree.py fp`):

- **F1, mint collapse at $t = 52$.** $I(52)$ already rounds to a zero-width point (lo = hi), and $1 - 2^{-t}$ rounds to 1.0 shortly after: every mint with a timestamp beyond ∼52 degenerates. A Lamport clock passes 52 within the first minute of any real system.
- **F2, depth underflow at 44.** On the sequential chain with $t = 1, 2, 3, \ldots$, the absolute width $2^{-\sum(t_i+2)}$ hits the subnormal floor 2^{-1074} at depth 44: deeper nodes have no interval.
- **F3, isometry survives only inside the window.** Shallow, small-ts folds are *exact* (dyadic products within 53 bits round to themselves) — but once a composition overflows the window it rounds, absolutes stop being birth constants, and Theorem 2's permanence is void: mutable geometry re-enters through the ulp.
- **DAG PBT: 114/120 clean, 6 executions die** — zero-width parents from F1/F2 produce `ZeroDivisionError` at merge re-relativization. The failures begin exactly when timestamps cross the window, and the 114 clean runs are those whose ids stayed small: binary64 works precisely as long as the workload fits 53 bits.

The general obstruction is a pigeonhole, not an artifact: the contract requires the state to order survivors by birth chains carrying $\Theta(\sum t_i)$ bits; a 64-bit pattern holds 64. *Any* fixed-width coordinate must eventually conflate two distinct chains — re-manufacturing ties — or reorder them. Fixed-precision floats do not evade the conservation law; they truncate it. What floats are good for: *(a)* a comparison cache — a binary64 key alongside the exact coordinate, compared first, falling back to the exact layer on equality; sound, since float equality flags exactly the pairs the window cannot separate; *(b)* the exact carrier itself in IEEE *shape* but with an unbounded mantissa (softfloat / big-dyadic), which is the natural implementation of these coordinates — cheaper than general rationals, no gcd ever; *(c)* fixed-width labels *with relabeling* (order-maintenance style) — but relabeling is geometry mutation, refuted throughout this campaign unless gated on causal stability, which is precisely the open compression question above.

The efficient representation (embed-code)

Binary64 fails, but the inefficiency it exposes is not intrinsic — it is the mint's: $I(t) = 1 - 2^{-t}$ writes the timestamp *in unary in the exponent*, spending t bits to carry $\log_2 t$ bits of information. Replace it with an **order-preserving prefix-free code** of the timestamp **delta** $\delta = t - t_{\text{anchor}}$ ($\delta \geq 1$ by causality — one inserts after something one has seen):

$$C(\delta) \;=\; \underbrace{1\cdots1}_{L-1}\,0\,\big(\delta \text{ minus its leading bit}\big), \qquad L = \text{bitlength}(\delta), \;\; |C| = 2L - 1,$$

minting the lower quarter of the dyadic cell $[\,0.C(\delta),\; 0.C(\delta) + 2^{-|C|}\,)$. Prefix-freeness makes distinct cells disjoint; monotonicity keeps newest-highest; both proofs are one line, and Theorems 1–2 and every corollary carry over verbatim — machine-confirmed: **the full gauntlet is clean** (credential countermodel, L25-third-party, both CEs, battery except one-sided L19, DAG PBT 120/120 and 300/300).

scenario	embed (unary)	**embed-code**	quarter carve (lossy)
1000-chain, sequential ($\delta = 1$)	2^{502501} (∼61 KB)	$\mathbf{2^{4001}}$ (∼0.5 KB)	2^{2001}
1000 root siblings ($\delta = t$)	max 2^{1003}, 125 KB total	**max $\mathbf{2^{23}}$, 5 KB total**	— (ties)

Sequential typing costs ∼4 bits per level — the quarter carve's constant, recovered — and a race with timestamp gap g costs $2\log_2 g + O(1)$ bits. Total coordinate size is $\Theta(\sum_i \log \delta_i)$: *the entropy of the birth chain.* This sharpens the conservation law to its final form: **the state pays for the magnitude of actual concurrency, and for nothing else**.

Two implementation corollaries. *(i)* The natural carrier is not a rational at all but a **bit string per node, parent-relative**: composition (the fold) is concatenation, comparison is lexicographic — $O(1)$ with structure sharing; the rationals are the *semantics* of the strings, kept for the proofs. (This is the Logoot/Fugue key space, reached here by construction rather than by choice, with the tree and the isometric fold retained on top.) *(ii)* The residual growth is exactly the dead chains concatenated into survivors' strings — the conservation law's irreducible content — and reclaiming it is the stability-gated compaction question, now the only open cost item.

7 Where the LCA helps — scoped honestly

The MRDT merge sees three states; "dead in a branch, live in the LCA" is visible for free, timestamp included — power that two-way-merge CRDTs lack. v1 argued this covers only the first merge after a death, with retention forced one merge later. That argument stands *only for re-render designs*: keep-the-range needs no consultation at all for distinct-range deaths, and for tie deaths the merge that processes the death typically still holds the tied record in an input (in the trace above, the final merge sees 6 live in one input). The honest residue is exactly risk *(ii)*: ties whose dead side no input remembers.

Status. Machine-checked: remembering nothing at all breaks — at ties (8 refuted designs); conditional retention on mutable conditions breaks (2 refuted rules); always-remembering works and equals the tombstoned RGA (O(history) worst case); keep-the-range with ts-oblivious mints works everywhere except ties (52/120 at scale); **keep-the-range with ts-faithful mints (§6) works everywhere** — full battery except one-sided L19, DAG PBT 120/120 and 300/300, no ids, no stamps, no ledger, no tombstones. The tie-only-stamp candidate is moot: ties no longer exist. The encoding question is settled (embed-code is entropy-optimal); open: stability-gated reclamation of dead-chain bits, and the L19 backward-run column (two-sided extension), both orthogonal to retention.

Provenance. `whiteboard/litmus/`: `contest_tree.py` (designs and countermodel regressions, and the always-keep control's lockstep equivalence with the tombstoned RGA), `sibling_tree.py` (the encoding-divergence probe), `relsplit_v2.py` (canonical refolding; the naive and frozen-block repairs), `embed_tree.py` (§6: the ts-faithful design, the v2c control, the credential countermodel, and the IEEE 754 measurements; `python3 embed_tree.py` runs the gauntlet, `python3 embed_tree.py fp` the float analysis). The keep-range runs reproduce from the session transcript of 2026-07-14; all table values machine-printed.