Cache Merging as a Convergent Replicated State for Multi-Agent Latent Reasoning

Anonymous authors
Paper under double-blind review

Abstract

Multi-agent latent reasoning composes the KV-cache contributions of multiple agents into a single context for a final agent. Recent work (Agent Primitives) realises this composition by concatenating per-agent caches along the sequence axis with RoPE re-encoding, a construction we name BagMerge. BagMerge is non-commutative, and which input ordering is best is not predictable a priori: it shifts with the deployment regime, the latent-step budget, and even the model scale. We make this cache exchange a *convergent replicated state*. First, CanonicalMerge fixes the layout: a content-determined ordering by mean K-norm at a middle transformer layer renders the merged cache byte-identical under any permutation of the inputs, verified algorithmically on synthetic tensors (arbitrary arity $N \leq 5$) and bit-for-bit on the real KV state of Qwen3-1.7B (28 layers) and Qwen3-4B (36 layers). Second, we separate the replicated *state* from decode-time *layout*: the durable object is a set of content-addressed latent fragments whose merge is set union, a state-based CvRDT (commutative, associative, idempotent, absorbing), and CanonicalMerge is its deterministic render. Because the render is byte-equivalent, every $N = 2$ accuracy number is inherited unchanged and re-delivered duplicate fragments are absorbed rather than re-concatenated. On a partitioned-reasoning benchmark (Qwen3-1.7B), CanonicalMerge lands within 4 percentage points of the best BagMerge ordering in every cell of a 12-cell regime $\times$ budget $\times$ ordering matrix and matches it without needing to know which ordering is best, trading a small, statistically insignificant accuracy margin for an unconditional structural guarantee; a four-cell Qwen3-4B scale check preserves the same qualitative result. The behaviour transfers to real multi-document QA (HotpotQA bridge-$k = 2$: no detectable degradation against a single-agent full-context baseline), while the closest training-free output-fusion baseline (PackLLM) loses by 45 points at matched generation budget, placing cache-level merging in a regime distinct from output-level fusion. Finally, at $k > 2$ we delimit the approach: cache merge transports and colocates latent traces but does not by itself compose them, motivating subsequent work.

1 Introduction

Multi-agent language-model systems increasingly reason in *latent* space: instead of exchanging natural-language messages, agents pass intermediate KV-cache state directly, giving a downstream agent access to richer, higher-bandwidth representations than decoded text (Zou et al., 2025; Jin et al., 2026), part of a rapidly growing shift toward latent-space computation in language models (Yu et al., 2026). This latent exchange is realized by composing the per-agent caches into a single context that a final agent attends over. As the number of contributing agents grows, and as settings move toward parallel reasoning, recursive delegation (Yang et al., 2026), and federated inference, this composition step becomes the load-bearing operation, yet the operator that performs it has been left largely implicit.

The de facto standard for the parallel case, which we name BagMerge, concatenates the agents' caches along the sequence axis and re-encodes their positions with RoPE (Jin et al., 2026). Concatenation is ordered, and the RoPE re-encoding makes that order matter: BagMerge is non-commutative, so which agent's fragment occupies the privileged position-0 prefix changes the decoded answer. Worse, *which* ordering is best

is not predictable in advance: we find it shifts with the deployment regime, the latent-step budget, and even the model scale (Section 5), so a deployer cannot fix a safe ordering without per-setting validation.

This is not merely a tuning nuisance; it signals a missing abstraction. In the settings latent exchange is most attractive for (federated or query-blind inference, where independent parties contribute pre-encoded state without a coordinator and often without the query itself), the composition must be order-independent, must converge to the same result regardless of delivery schedule, and must tolerate the same fragment arriving more than once (gossip, retry, re-broadcast). These are precisely the guarantees that conflict-free replicated data types (CRDTs) provide for distributed state (Shapiro et al., 2011; Preguiça et al., 2018), yet prior latent-MAS cache composition offers none of them: it is order-sensitive, schedule-dependent, and duplicates re-concatenate rather than absorb. The problem we take up is how to make inter-agent cache exchange a *convergent replicated state*, well-defined, coordinator-free, and robust to delivery, without sacrificing accuracy.

We address this in two steps. First, CanonicalMerge fixes the layout by *content* rather than by caller order: it sorts the per-agent caches by their mean key-norm at a single middle transformer layer, a content-determined score that renders the merged cache byte-identical under any permutation of the inputs (Section 4). We verify this commutativity algorithmically for up to five agents and bit-for-bit on the real KV state of Qwen3-1.7B (28 layers) and Qwen3-4B (36 layers). Second, we separate the replicated *state* from its decode-time *layout*: the durable object is a set of content-addressed latent fragments whose merge is set union, a state-based CvRDT that is commutative, associative, idempotent, and absorbing by one-line proofs, and CanonicalMerge is its deterministic render (Section 4.4). Because the render is byte-equivalent to a canonical concatenation, every accuracy result transfers unchanged, while re-delivered duplicate fragments are absorbed rather than re-concatenated.

To isolate the merge operator empirically we introduce a partitioned-reasoning benchmark whose problems split into individually-insufficient, jointly-necessary fragments, so that a correct answer *requires* integrating both caches (Section 3). Across a regime $\times$ budget $\times$ ordering matrix, CanonicalMerge matches the best BagMerge ordering in every regime without needing to know which ordering is best; a three-draw greedy replication confirms this parity under paired-bootstrap, Holm-corrected tests, trading a small, statistically insignificant accuracy margin for an unconditional structural guarantee (Section 5). The behaviour transfers to real multi-document QA (HotpotQA) with no detectable degradation against a single-agent full-context baseline, while the closest training-free output-fusion baseline, PackLLM, trails by 45 points at matched budget, placing cache-level merging in a regime distinct from output-level fusion. A re-delivery experiment then makes idempotence load-bearing: naive re-concatenation degrades monotonically and grows the cache linearly in the number of deliveries, whereas the CvRDT render is invariant. Finally, beyond two agents we delimit the approach, cache merging transports and colocates latent traces but does not by itself compose them, motivating subsequent work (Section 8).

Contributions.

- We identify and characterize the *directionality flaw* of BagMerge, the concat-with-RoPE cache operator implicit in prior latent-MAS work: it is non-commutative, and the best ordering is unpredictable across regime, latent budget, and model scale (Section 4.2 and Section 5).
- We introduce CanonicalMerge, a content-determined cache layout whose output is byte-identical under any permutation of the inputs, verified algorithmically up to $N = 5$ and bit-for-bit on real Qwen3-1.7B and 4B KV state (Section 4 and Appendices A and C).
- We reframe the merged cache as a *convergent replicated state*: a set-based CvRDT (LatentFragmentSet) whose deterministic render is CanonicalMerge, giving commutativity, associativity, idempotence, and duplicate-absorption on the state, with every $N = 2$ accuracy number inherited byte-exactly (Section 4.4).
- We construct a partitioned-reasoning benchmark that isolates the merge operator, and show under a multi-seed, Holm-corrected paired analysis that CanonicalMerge matches the best fixed ordering in each regime without selecting it, and operates in a regime distinct from output-level fusion (PackLLM -45 pp at matched budget) (Sections 3 and 5).

- We demonstrate idempotence as an operational property under fragment re-delivery, and delimit the approach at $k > 2$, merge transports but does not compose, to motivate subsequent work (Section 5.9 and Section 8).

2 Related Work

2.1 Latent reasoning and inter-agent latent communication

Latent chain-of-thought, introduced as Coconut (Hao et al., 2024), replaces sampled text tokens with continuous last-layer hidden states fed back autoregressively, decoupling internal computation from the discrete vocabulary. LatentMAS (Zou et al., 2025) extends this to the multi-agent setting: rather than communicating in text, each agent's KV cache is prepended to the next agent's working memory, forming a *sequential* chain of latent thoughts. Interlat (Du et al., 2025) shares the inter-agent latent-message hypothesis with a small adapter that projects one agent's hidden states into the next's input space, and RecursiveMAS (Yang et al., 2026) casts a heterogeneous system as a recursive latent computation, transferring last-layer hidden states through a learned residual link over recursion rounds. All of these compose agents *sequentially*, the next agent conditions on the union of prior caches, and their composition is order-dependent by construction. We study the *parallel* variant: agents run independently on disjoint fragments and a merge function combines their caches, a setting in which the choice of merge function has consequences the sequential constructions never expose. A recent taxonomy of latent inter-agent communication (Liu, 2026) catalogues the operators in use (concatenation, prepending, arithmetic, cross-attention, cache restoration); none provides the combination we study, a content-addressed set-union state, a deterministic cache render, duplicate absorption, and byte-stable input-order invariance, which is the cell our construction fills.

2.2 Parallel reasoning and output-space fusion

Many techniques fan out to N reasoning paths and aggregate in *answer space*. Self-Consistency (Wang et al., 2023) samples N decodings and votes; Multi-Agent Debate (Du et al., 2023) runs role-conditioned agents through rounds of textual exchange; Tree of Thoughts (Yao et al., 2023) searches a tree with branch-and-evaluate operators; Mixture of Agents (Wang et al., 2024a) cascades parallel proposers and aggregators; Universal Self-Consistency (Chen et al., 2023) generalises voting to LLM-judged majority over free-form text. All discard the latent state that produced each output and read text. CanonicalMerge is the cache-space analogue of the last: rather than asking a selector to pick the most consistent generation, a choice with known position bias (Zheng et al., 2023), we let the geometry of the K vectors determine which thinker is primary and let the judger consume the merged cache directly. Output-level *fusion* aggregates per-thinker next-token distributions instead; PackLLM (Mavromatis et al., 2024) is the closest training-free neighbour we compare against (Section 5.6), and the newest training-free fusion methods (Cui et al., 2026; Yao et al., 2024) remain strictly logit-level, never touching hidden state or KV, the limitation on which cache-level merging separates from output fusion.

2.3 BagMerge: concat-with-RoPE as prior art

Agent Primitives' (Jin et al., 2026) "Voting and Selection" primitive runs N parallel solvers and combines their KV caches by concatenating along the sequence axis, re-rotating each successor's K vectors so its positions extend the prefix; it gives the explicit formula $R_\theta(t + n_B)\, R_\theta(t)^{-1}\mathbf{K}_t$ and reports large gains at 8B/70B over concatenation without re-encoding. It applies the merge at $N = 3$ solvers, but in a voting setting where each solver independently attempts the *full* task and a selector aggregates among complete candidates, so the merge *selects* rather than *composes*; our partitioned benchmark instead forces the merge to compose jointly-necessary fragments, which is where the $k > 2$ ceiling appears (Section 8). LatentMAS (Zou et al., 2025) exchanges KV caches through the same prepend mechanism but in a *fixed, sequential* order (Section 2.1), each agent conditioning on its predecessor's cache, so it never confronts the parallel input-order choice; the parallel, order-free construction is what we name BagMerge, for its multiset semantics. Its flaw, which Agent Primitives does not measure, is that the prefix slot is asymmetric (its K vectors keep their original rotation; later slots are re-rotated), so the choice of which cache occupies it changes accuracy, a 6-percentage-point

gap between the two orderings of $N = 2$ thinkers on our benchmark (Section 5). RoFormer (Su et al., 2024) anchors the rotation formalism and YaRN (Peng et al., 2024) documents the degradation when K vectors appear at out-of-distribution positions, explaining why the asymmetry is load-bearing. Closest in architecture is Parallel-Synthesis (Liu et al., 2026), in which a synthesizer directly consumes the caches of N parallel same-backbone workers via a cache mapper that performs exactly this RoPE manoeuvre. It differs from CanonicalMerge on every point that constitutes our contribution: it concatenates in *fixed* worker order with no content-determined ordering, commutativity analysis, or byte-identical-render claim, and it *requires post-training* (a cache mapper plus a synthesizer LoRA). The authors report that their no-training variant "performs substantially worse", an architectural neighbour whose need for learned calibration is consistent with our $k > 2$ result (Section 8) that a free cache merge transports but does not compose.

2.4 Content-determined cache operators and CRDT structure

The CRDT literature (Shapiro et al., 2011; Preguiça et al., 2018) distinguishes operation-based (CmRDT) data types, whose operations commute, from state-based (CvRDT) data types, which additionally form a semilattice with idempotent, absorbing merge. The *rendered* CanonicalMerge operator has the commutativity and byte-identity we need for one-shot rendering, but it is not itself the replicated data type: it is not idempotent, because RoPE re-rotation moves a re-delivered fragment's K vectors out of the positions where the content function recognises them. We recover the full property set by separating state from render (Section 4.4): the replicated state is LatentFragmentSet, a set of content-addressed unshifted fragments under set union, a state-based CvRDT, of which CanonicalMerge is the deterministic render. The closest framing is Gillespie's CRDT for neural-network merging (Gillespie, 2026), which independently pairs an OR-Set state-CRDT with a canonical render; it merges *persistent weights offline*, however, with no positional structure, no per-inference state, and no decode-time composition, whereas we merge *per-inference latent KV* fragments, to our knowledge the first CvRDT over inference-time latent state. Gillespie pre-empts the broad claim "first CRDT over neural state"; our novelty is the inference-time KV object, the RoPE render, and the empirical latent-merge boundary at $k > 2$. As a cache operator CanonicalMerge is also kin to KV-cache compression, StreamingLLM (Xiao et al., 2024), H2O (Zhang et al., 2023), and SnapKV (Li et al., 2024) all let content decide which K vectors are special, but *within one cache*. A separate line *merges* rather than evicts redundant KV entries (Wang et al., 2024b), but again within a single sequence's cache, collapsing adjacent positions for memory; we merge whole caches *across agents* to combine content, an orthogonal axis.

2.5 Federated / query-blind inference

Federated *training* is established (Douillard et al., 2023); federated *inference* is less explored. Retrieval-Augmented Generation (Lewis et al., 2020) is the canonical pattern of external sources contributing fragments to a central generator, but in *text* space; CanonicalMerge is its cache-space analogue when fragments are pre-encoded as KV caches, and the encoding can be *query-blind*, thinkers encode without seeing the global query. The indirect-prompt-injection literature (Greshake et al., 2023) notes that any pipeline where retrieved data crosses the data/instruction boundary *by position* rather than by content is exploitable; BagMerge has this property in cache space, and CanonicalMerge's content-based primary selection gives a principled hook it lacks.

2.6 Implementation context

PagedAttention and vLLM (Kwon et al., 2023) establish per-request KV caches as first-class, block-addressable objects that can be shared, copied, and recombined across requests; CanonicalMerge slots into this abstraction as another reduction over the same block-structured caches, with no new systems primitive required.

3 Partitioned-Reasoning Benchmark

We construct a multi-step reasoning benchmark in which each problem is *partitioned* between two thinkers: each thinker receives one of two textual fragments that, individually, contain insufficient information to derive

Family	t_a	t_b	answer
CONSTRAINT	"Alice is 3× as old as Bob."	"Together they are 40."	30
RECIPE	"One batch uses 2 cups flour."	"Make 5 batches."	10
DISCOUNT	"$10 each, 20% off for groups ≥ 4."	"5 friends buy tickets."	40
NEONYM	"1 wog = 7 wibbles."	"The race is 49 wibbles."	7
LOOKUP	"Red= 3, Blue= 5, Green= 2 points."	"4 red, 2 blue, 6 green."	34

Table 1: Family schemas with one example shown per family; 20 problems per family vary the surface numerical values while preserving the schema.

view	accuracy	interpretation
`full` (C0)	100%	ceiling — single thinker on $t_a + t_b$
`a_only` (C1)	5%	half a alone is insufficient
`b_only` (C2)	2%	half b alone is insufficient

Table 2: Single-thinker baselines on partitioned $k = 2$ (100 problems, Qwen3-1.7B). The headroom available for a two-thinker merge to contribute is 95 pp.

the answer but jointly determine a unique integer answer. The merge operator under test is responsible for combining the two per-thinker latent states into a coherent context the judger can decode from.

3.1 Motivation

Existing multi-agent latent-reasoning benchmarks (Zou et al., 2025; Jin et al., 2026) give each thinker access to the global query. In such designs, any accuracy lift from a merge operator is confounded with the extra inference compute applied to the same problem. A same-data evaluation cannot distinguish a merge that *integrates* information across thinkers from one that merely regularises the output of a single thinker (full discussion in Appendix E). The partitioned construction eliminates the same-data confound by design: a single thinker operating on its half can only achieve a floor-level accuracy, because the missing fragment carries information the model has no prior way to recover.

3.2 Construction

The benchmark comprises 100 problems drawn evenly from five problem families, 20 per family. Each problem has two text fragments t_a and t_b and an integer answer. The fragments are constructed so that any single fragment is insufficient.

The NEONYM family uses fictional units (e.g. `wog`, `wibble`) for which the model has no prior knowledge; the conversion factor must be obtained from t_a. The other four families use surface forms a 1.7B-parameter model could in principle have memorised, but the specific numerical values vary problem-by-problem.

We provide four data-loader views on each problem. The two-thinker `split` view sets thinker$_0$'s input to t_a and thinker$_1$'s to t_b. Three single-thinker views serve as controls: `full` concatenates $t_a + t_b$ as a ceiling, and `a_only` / `b_only` expose just one fragment.

3.3 Empirical validation

The single-thinker baselines establish that the partitioning is strong: an agent reading only t_a scores 5%, an agent reading only t_b scores 2%, while a single agent reading $t_a + t_b$ scores 100%. The headroom available for a two-thinker merge to contribute is therefore ~ 95 percentage points, and an effective merge operator should land between the single-fragment floor and the full-text ceiling. Numbers reported in Section 5 are interpretable against this scale.

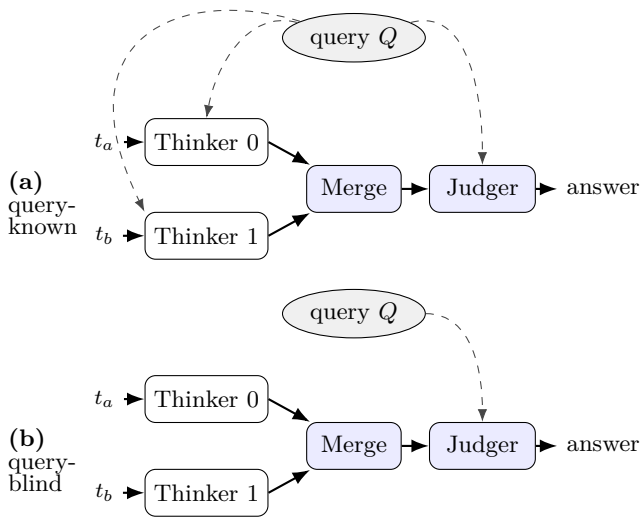

Figure 1: The two evaluation regimes. The KV-cache flow (solid arrows), thinkers → MERGE → JUDGER → answer, is identical across regimes; only the routing of the global query Q (dashed arrows) differs. In **(a) query-known**, Q is broadcast to both thinkers and the judger, so each thinker encodes its fragment t_i conditioned on the query. In **(b) query-blind (federated)**, Q reaches only the judger, and each thinker encodes its fragment offline without seeing the query. The same merge operator and the same 100 partitioned-$k = 2$ problems are used in both regimes; the difference shifts the K-norm content signature each thinker produces (Section 5.2).

3.4 Two regimes: query-known and query-blind

We evaluate the merge operator under two deployment regimes that correspond to qualitatively different real-world settings.

Query-known. Each thinker receives both its data fragment and the global query. This is the coordinated setting in which a central component broadcasts the query to all thinkers before they encode their fragments.

Query-blind (federated). Each thinker receives only its data fragment; the global query reaches only the judger, after the merge. This is the federated setting in which each data source encodes its fragment offline without coordinating on the query, for instance, a multi-party inference setup where data shards reside on hosts that do not see each other's queries.

The two regimes are distinguished by a single flag in our reference implementation; the same 100 partitioned-$k = 2$ problems are used for both. The two regimes produce qualitatively different K-norm content signatures for the same thinker (Figure 1), and the comparison between them is the central piece of evidence in Section 5 for the *structural robustness* of the merge operator we propose.

4 Method

4.1 Setup and notation

Let M be a decoder-only transformer with L layers, H attention heads per layer, head dimension d, and rotary position embedding (RoPE) (Su et al., 2024) with base θ. For an input sequence of T tokens, M maintains a per-layer KV cache

$$\mathcal{C} \;=\; \big\{(\mathbf{K}^{(\ell)}, \mathbf{V}^{(\ell)})\big\}_{\ell=1}^{L}, \qquad \mathbf{K}^{(\ell)}, \mathbf{V}^{(\ell)} \in \mathbb{R}^{1\times H\times T\times d}.$$

The K vectors are stored *post-rotation*: position t's key has been multiplied by the rotation matrix $R_\theta(t)$ defined elementwise on the head dimension. Attention computes $\mathbf{q}_s^\top \mathbf{k}_t = \mathbf{q}_s^\top R_\theta(t-s)\mathbf{k}_t$ under the RoPE construction, so relative displacement is encoded automatically.

We consider N parallel *thinker* agents, each running M independently on a fragment of the problem (Section 3) for the same number of *latent autoregressive steps* (Hao et al., 2024). Our partitioned construction places exactly one fragment with each thinker, so the number of thinkers N equals the number of fragments k; we use N and k interchangeably, writing the $N = 2$ case as "$k = 2$" when emphasising the partition. Thinker $i \in \{1, \ldots, N\}$ produces a KV cache $\mathcal{C}_i = \{(\mathbf{K}_i^{(\ell)}, \mathbf{V}_i^{(\ell)})\}_\ell$ of length T_i. A *judger* agent then consumes a merged cache $\mathcal{C}^\star = \mathrm{Merge}(\mathcal{C}_1, \ldots, \mathcal{C}_N)$ and decodes the answer with the same model M. The two roles are asymmetric: the thinkers run latent steps and emit no text, contributing only their KV caches, whereas the judger runs no latent steps of its own and instead decodes the answer autoregressively with $\mathcal{C}^\star$ as its KV prefix. No model parameters are trained; the construction is training-free. The merge function is the focus of this paper.

4.2 BagMerge (prior work)

Agent Primitives (Jin et al., 2026) defines a merge operator that concatenates per-thinker caches along the sequence axis, with RoPE re-encoding so successor positions extend the prefix range without overlap. (LatentMAS (Zou et al., 2025) exchanges caches through the same prepend mechanism, but in a fixed sequential order, each agent conditioning on its predecessor's cache rather than choosing a concatenation order among parallel peers; Section 2.3.) We name this parallel construction BagMerge (multiset semantics, input-order preserving).

Let π be a permutation of $\{1, \ldots, N\}$ specifying the concatenation order, and define cumulative offsets $s_{\pi(1)} = 0$ and $s_{\pi(i)} = s_{\pi(i-1)} + T_{\pi(i-1)}$ for $i > 1$. For each layer ℓ,

$$\mathbf{K}^{\star,(\ell)}_{\textsc{BagMerge}} = \operatorname*{cat}_{i=1}^{N}\Big[\, R_\theta\big(s_{\pi(i)}\big) R_\theta(0)^{-1}\, \mathbf{K}^{(\ell)}_{\pi(i)} \Big], \tag{1}$$

$$\mathbf{V}^{\star,(\ell)}_{\textsc{BagMerge}} = \operatorname*{cat}_{i=1}^{N}\Big[\, \mathbf{V}^{(\ell)}_{\pi(i)} \Big]. \tag{2}$$

The values $\mathbf{V}$ are not rotated and pass through unchanged. The factor $R_\theta(s_{\pi(i)})R_\theta(0)^{-1}$ shifts thinker $\pi(i)$'s effective positions from $\{0, \ldots, T_{\pi(i)} - 1\}$ to $\{s_{\pi(i)}, \ldots, s_{\pi(i)} + T_{\pi(i)} - 1\}$, the formula given in Jin et al. (2026) §3.1 as "KV PoPE".

The directionality flaw. The judger's decoded outputs depend on π, i.e. the chosen order of concatenation among the thinker outputs. This introduces an asymmetry, since the position-0 prefix slot is special: its K vectors keep their original RoPE rotation, matching the geometry the judger's parameters were trained on. Tail slots have been re-rotated by an amount that depends on the prefix length $s_{\pi(i)}$. Empirically, this asymmetry causes a 6-percentage-point accuracy gap on our partitioned benchmark between the two orderings of $N = 2$ thinkers (Section 5). Neither prior work measures this gap, nor proposes a principled mechanism for choosing π.

4.3 CanonicalMerge

We replace input-determined permutation with a *content-determined* permutation derived from the K vectors themselves.

Definition. Fix a *routing layer* $\ell^\star \in \{1, \ldots, L\}$ (we use $\ell^\star = 15$ for Qwen3-1.7B; Section 5 provides a sensitivity sweep). For each thinker i, compute the mean K-norm over the last $T^\star = \min_j T_j$ positions (tail-aligned to the shortest cache):

$$\rho_i = \frac{1}{T^\star} \sum_{t=T_i-T^\star+1}^{T_i} \big\| \mathbf{K}_i^{(\ell^\star)}[\,\cdot,\,\cdot,\,t,\,\cdot\,] \big\|_2 \quad \text{(averaged over batch and heads).} \tag{3}$$

Algorithm 1 CanonicalMerge

Require: Caches $\mathcal{C}_1, \ldots, \mathcal{C}_N$; routing layer $\ell^\star$.

1: **for** $i = 1, \ldots, N$ **do**
2: $\quad \rho_i \leftarrow \mathrm{mean}\big(\big\|\mathbf{K}_i^{(\ell^\star)}[\cdot, \cdot, \text{last } T^\star, :]\big\|_2\big)$
3: $\quad h_i \leftarrow \mathrm{hash}\big(\text{full content of } \mathcal{C}_i\big)$ ▷ all layers' $\mathbf{K}, \mathbf{V}$; the LatentFragmentSet id
4: **end for**
5: $\pi^\star \leftarrow \mathrm{argsort}_i\,\big(-\rho_i,\ -h_i\big)$ ▷ descending; tiebreak by hash
6: **return** BagMerge$(\mathcal{C}_{\pi^\star(1)}, \ldots, \mathcal{C}_{\pi^\star(N)})$

Sort the thinkers in descending order of ρ, breaking ties by a content hash h_i over fragment i's *full* content (the shapes, dtypes, and raw bytes of every layer's $\mathbf{K}$ and $\mathbf{V}$), identical to the fragment id used by LatentFragmentSet (Section 4.4), so the sort is a total deterministic order over distinct fragments, modulo the usual cryptographic collision assumption. The resulting permutation $\pi^\star$ is the *canonical order*. Apply BagMerge with $\pi = \pi^\star$. Algorithm 1 gives the procedure.

CanonicalMerge has **a single design parameter**, the routing layer $\ell^\star$, one fixed architectural choice, which we report together with a sensitivity sweep (Section 5.4) and re-select by a coarse dev-set sweep when changing model scale (Section 5.8); the full-content hash tiebreak is deterministic and parameter-free. There is no fusion weight, similarity threshold, or per-task tuning.

The position-0 prefix slot is the privileged one: its K vectors retain their original RoPE rotation and are read by the judger in the geometry its parameters were trained on, whereas the tail thinker's vectors are re-rotated to a shifted position range (Section 4.2). Sorting in *descending* ρ therefore places the more informationally dense fragment, the one more likely to carry the binding constraint, in the slot the judger reads most reliably, and leaves the less critical fragment to absorb the slight geometric degradation of the re-rotated tail. Where BagMerge assigns this privileged slot by arbitrary input order, CanonicalMerge assigns it by content, per problem.

Two properties from one mechanism. *Commutativity at $N = 2$.* The score function ρ_i depends only on thinker i's K vectors, not on the input order. The sort is total and content-determined; with a deterministic tiebreaker, the permutation $\pi^\star$ is invariant under any reordering of the input list. Therefore

$$\text{CanonicalMerge}(\mathcal{C}_{\sigma(1)}, \ldots, \mathcal{C}_{\sigma(N)}) \;=\; \text{CanonicalMerge}(\mathcal{C}_1, \ldots, \mathcal{C}_N) \tag{4}$$

bit-for-bit, for every permutation σ. The merged cache is *byte-identical* under thinker reordering, a structural property verified algorithmically on synthetic tensors at $N \in \{2, 3, 4, 5\}$ across 152 permutation pairs (Appendix A) and bit-for-bit on the actual model's 28-layer KV state under the headline configuration (Appendix C). The property is established by construction; it does not depend on sample size or empirical evaluation. We claim commutativity and byte-identity for the $N = 2$ case studied in this paper. The empirical question of whether the model integrates merged caches at $N > 2$ is deferred to Section 8.

Per-problem adaptive routing. The mean K-norm at a mid-network layer correlates with the informational density of the fragment a thinker has encoded, a claim grounded in the interpretability literature on decoder-only LMs (Tenney et al., 2019; Geva et al., 2021; Belrose et al., 2023). For some problems the binding constraint is in one fragment; for others, in another. CanonicalMerge selects the higher-density thinker for the prefix slot *per problem* rather than globally, subsuming the $\pi = \mathrm{id}$ and $\pi = \mathrm{swap}$ orderings of BagMerge into a single content-routed choice. As we show in Section 5, this content function works in the query-known regime, where each thinker has done question-conditional reasoning; in the query-blind / federated regime the K-norm signal flattens, but the commutativity property holds unchanged.

4.4 From rendered operator to replicated state

CanonicalMerge fixes the directionality of the *rendered* cache, but the rendered operator is not by itself a convergent replicated data type. Re-merging a fragment already present in an intermediate result changes that result: the copy of B inside CanonicalMerge(A, B) has had its K vectors RoPE-shifted out of their original

positions, so the content-determined score no longer recognises them as B, and a second merge does not absorb, $\textsc{CanonicalMerge}(\textsc{CanonicalMerge}(A, B), B) \neq \textsc{CanonicalMerge}(A, B)$. The obstruction is that layout has been baked into the state.

We remove it by separating the replicated *state* from the decode-time *layout*. The state is a set $\mathcal{S} = \{f_1, \ldots, f_m\}$ of *content-addressed latent fragments*: each fragment is one thinker's *unshifted* legacy KV cache, and its identity is the full canonical byte string of all layers' $\mathbf{K}$ and $\mathbf{V}$ (the hash is an address for that identity, assumed collision-resistant; two distinct byte strings are distinct fragments). Merge is set union,

$$\mathcal{S}_1 \sqcup \mathcal{S}_2 \;=\; \mathcal{S}_1 \cup \mathcal{S}_2, \tag{5}$$

which is commutative, associative, idempotent, and absorbing by the corresponding one-line properties of $\cup$. Order the states by inclusion over fragment identities, $\mathcal{S} \preceq \mathcal{T}$ iff every identity in $\mathcal{S}$ is in $\mathcal{T}$; a local update only adds a fragment, so it is inflationary ($\mathcal{S} \preceq \mathcal{S} \sqcup \mathcal{S}'$), and union is the least upper bound under $\preceq$. The state therefore forms a join-semilattice and is a state-based CvRDT (Shapiro et al., 2011); we call this object $\textsc{LatentFragmentSet}$.

A *render* is a deterministic pure function from a set to a single decodable cache. We take render to be $\textsc{CanonicalMerge}$ applied to the fragments in content-canonical order; because $\textsc{CanonicalMerge}$ is itself permutation-invariant (Equation (4)), the render is a well-defined function of the *set*, not of any enumeration of it. Crucially, $\text{render}(\{A, B\})$ is byte-identical to the $\textsc{CanonicalMerge}$ output on the same two caches, so every $N = 2$ result in Section 5 is inherited unchanged: the state/render split adds the CvRDT guarantees without altering a single decoded byte.

The algebraic guarantees live on the state, not on the operator. Because RoPE layout is applied only at render time, no merge ever has to re-recognise content whose K vectors have already been rotated to a tail slot; the absorption and idempotence that the rendered operator lacks hold exactly for the set. We verify this behaviourally on real Qwen3 caches: under re-delivery of a byte-identical fragment, $\textsc{LatentFragmentSet}$ absorbs the duplicate and the rendered cache is invariant, whereas naive re-rendering of the duplicated multiset grows the cache and can change the decoded answer (Section 5.9).

One boundary is worth stating plainly. The absorption is *syntactic*: byte-identical copies hash equal and collapse to one element, but two caches that encode the same information through different bytes are distinct elements of the set. A semantically convergent join, one that also merges near-duplicate but non-identical fragments, is left to future work.

4.5 Implementation

Algorithm 2 collects the three operations of the state/render split in one place. A fragment's identity $\textsc{Id}$ is the hash of its *unshifted* per-layer $\mathbf{K}, \mathbf{V}$ bytes; local delivery ($\textsc{Ingest}$) and cross-replica $\textsc{Merge}$ are both set union keyed by that identity, so re-delivered duplicates absorb (Equation (5)); and $\textsc{Render}$ applies $\textsc{CanonicalMerge}$ (Algorithm 1) to the fragments of the set, well-defined because $\textsc{CanonicalMerge}$ is permutation-invariant. The hash and the merge act on state; the RoPE layout is introduced only inside $\textsc{Render}$, which is what keeps idempotence and absorption on the set rather than on the operator.

Reference implementation: strategy `novelty_concat_sym_layout` for $N = 2$ and `novelty_concat_sym_layout_n` for arbitrary N. The merge operates on the standard HuggingFace legacy KV cache format (per-layer tuples of $(\mathbf{K}, \mathbf{V})$ tensors). The construction is compatible with PagedAttention-style block-addressable caches (Kwon et al., 2023): the merge is a permutation over already-allocated blocks plus a position-dependent RoPE-rotation kernel that is $\mathcal{O}(T \cdot d)$ per thinker per layer, negligible compared to the generation cost of the latent reasoning steps that produced the cache.

5 Experiments

Unless noted otherwise, experiments use Qwen3-1.7B with the routing layer fixed at $\ell^\star = 15$ (a layer-sensitivity sweep in Section 5.4 justifies the choice); Section 5.8 replicates the headline comparison on Qwen3-4B. Unless noted otherwise, each cell is 100 problems from the partitioned-$k = 2$ benchmark of Section 3 (Section 5.7

Algorithm 2 LatentFragmentSet: state operations and render

```
function Id(C)                                          ▷ content address of a fragment
    return hash(bytes(K^(1), V^(1), ..., K^(L), V^(L)))  ▷ unshifted; collision-resistant
end function

function Ingest(S, C)                                   ▷ deliver a cache into the state
    return S ∪ {(Id(C), C)}                             ▷ keyed by Id; a duplicate is absorbed
end function

function Merge(S_1, S_2)                                ▷ CvRDT join ⊔
    return S_1 ∪ S_2                                    ▷ commutative, associative, idempotent, absorbing
end function

function Render(S)                                      ▷ deterministic decode-time layout
    return CanonicalMerge({C : (·, C) ∈ S})             ▷ Alg. 1; pure function of the set
end function
```

Method	q-known @ 20	q-known @ 40	q-blind @ 20	q-blind @ 40
BagMerge default	0.89	0.85	**0.90**	**0.85**
BagMerge swap	**0.90**	**0.91**	0.83	0.77
best BagMerge	*swap*	*swap*	*default*	*default*
CanonicalMerge	**0.93**	**0.94**	0.86	0.85
Δ (CM − best BagMerge)	+0.03	+0.03	−0.04	0.00

Table 3: Headline matrix on partitioned $k = 2$, 100 problems each. Best BagMerge ordering per column in bold. The *best* BagMerge row records which fixed π achieves it; the Δ row records the CanonicalMerge accuracy minus the best BagMerge accuracy in the same column.

uses 50 HotpotQA problems and the 4B layer sweep uses 50), sampling temperature 0.7 with `top_p` = 0.95 and a single seed; the $N = 2$ accuracy matrix is additionally replicated under greedy decoding across three problem draws (Appendix B). Confidence intervals and paired tests are reported in Section 5.3 with full per-comparison tables in Appendix B.

5.1 Headline: accuracy across regimes, budgets, and orderings

Table 3 reports accuracy across the full 2 regimes × 2 latent-step budgets × 3 methods matrix: query-known vs. query-blind regime (Section 3.4), 20 vs. 40 latent reasoning steps per thinker, and BagMerge with $\pi = \text{id}$ ("default") vs. BagMerge with $\pi = \text{swap}$ vs. CanonicalMerge.

Two structural observations dominate the matrix.

(i) The best BagMerge ordering is regime-and-budget-conditional. In the query-known regime BagMerge swap is best at both latent budgets (0.90 at 20 steps; 0.91 at 40); in the query-blind regime BagMerge default is best at both budgets (0.90 at 20; 0.85 at 40). The optimal ordering inverts between the two regimes and the swap/default gap widens with budget: a deployer using BagMerge would need to know both the regime (does the data source see the query?) and the latent budget to pick the right π, and 7–14 percentage points are at stake in the worst cells. The dependence is in fact worse than "regime × budget": it is also *model*-dependent: on Qwen3-4B the inversion disappears and swap wins both regimes (Section 5.8), so the best ordering is not stable across the observed regimes and model scales; a deployer cannot pick it without validation or prior regime knowledge. Agent Primitives (Jin et al., 2026), which introduces this parallel merge, does not measure the dependence, and a coordinator that did want to specify π correctly would need an oracle for it.

(ii) CanonicalMerge removes the ordering choice, and stays close in the other cells. The output is byte-identical under any permutation of the inputs *by construction* (Section 4.3, Equation (4)), and the prefix occupant is content-routed per problem rather than globally fixed. The remaining two knobs, regime and latent budget, change the per-thinker caches themselves, so no merge can be byte-invariant to them; what we observe there is instead *empirical* robustness, that CanonicalMerge's quality stays close to the best fixed ordering. It lands within 4 percentage points of the best BagMerge ordering in every cell and is statistically indistinguishable from it ($\Delta = +0.03/+0.03/0.00/-0.04$; every gap within the bootstrap CIs, Section 5.3), matching the best ordering without needing to know which ordering is best. The one cell in which CanonicalMerge underperforms is the query-blind, 20-step cell: there BagMerge default happens to align with the content-density signal at 0.90, but a deployer would not know in advance which BagMerge ordering will land there without running both and comparing.

What this paper claims, narrowly. We separate two kinds of robustness and claim them differently. *Structural / algebraic (exact, by construction):* CanonicalMerge is bit-for-bit invariant to input ordering (Section 4.3, with algorithmic verification in Appendix A and real-model verification on the Qwen3-1.7B KV state in Appendix C), and the LatentFragmentSet state absorbs duplicate re-deliveries (Section 4.4). *Empirical (observed in the tested matrix, not an algebraic invariant):* quality stays close to the best fixed ordering across regime and latent budget, a signed mean gap of +0.005 and a mean absolute gap of 0.025 across the four cells. The structural guarantees are the principal claim; the empirical parity is the precondition that makes them useful, not an accuracy-superiority result.

5.2 Mechanism: why the K-norm signal is question-conditional

Table 3 also surfaces a mechanistic observation. The mean-K-norm content function (Equation (3)) is a question-conditional informational-density proxy: when each thinker has encoded both its data fragment and the global query (query-known), the K-norm profile at $\ell^\star = 15$ concentrates question-relevant constraint mass in the high-norm tail, and the per-problem selection of the higher-density thinker for the prefix slot routes the binding fragment correctly. When the thinker has encoded only its raw data fragment (query-blind), the K-norm profile no longer carries the question-conditional structure, and the content selection is closer to a tiebreak between two equally-informed candidates. The -4 pp underperformance in the query-blind, 20-step cell of Table 3 is consistent with this reading: in that cell BagMerge default happens to align the lexically-first fragment with the prefix slot, and that lexical-order signal is more informative than the flattened K-norm signal at small latent budget. Even in the cell where the K-norm signal is least discriminative, the commutativity property (Equation (4)) is unaffected, it depends only on the symmetry of the score function, not on whether the score is informative.

5.3 Statistical significance

We assess the accuracy comparisons at two levels. As a per-cell diagnostic of the headline matrix (Table 3, sampled decoding), we computed paired McNemar tests and 10 000-resample bootstrap 95% confidence intervals on the per-problem correctness vectors (scripts: `scripts/bootstrap_paired_ci.py` and `scripts/bootstrap_paired_ci_20step.py`). The statistical basis for the paper's final $N = 2$ claim, however, is the higher-powered three-draw greedy replication of Appendix B, summarised under *Multi-seed replication* below.

At our sample size, two of the 40-step pairwise accuracy gaps reach nominal significance at $\alpha = 0.05$: CanonicalMerge over BagMerge default in query-known by 9 pp ($p = 0.023$) and over BagMerge swap in query-blind by 8 pp ($p = 0.039$), both against the *worse* fixed ordering. At the 20-step budget no individual pairwise comparison reaches significance, the largest gap is ± 0.04 and neither sign passes the binomial mid-p threshold at $n = 100$. These two are nominally significant among twelve pairwise comparisons across the matrix (six per budget); under Holm correction for multiple comparisons, neither survives. We therefore rest no claim on the single-draw pairwise accuracy gaps: the primary $N = 2$ inference is the multi-seed replication below, which compares CanonicalMerge directly to the best fixed ordering in each regime, and the categorical contribution is the structural property established next.

budget	comparison	Δ acc	McNemar p
40 steps	CanonicalMerge vs. BagMerge default (q-known)	+0.090	**0.023**
40 steps	CanonicalMerge vs. BagMerge swap (q-blind)	+0.080	**0.039**
40 steps	CanonicalMerge vs. BagMerge swap (q-known)	+0.030	0.549
40 steps	CanonicalMerge vs. BagMerge default (q-blind)	0.000	1.000
20 steps	CanonicalMerge vs. BagMerge default (q-known)	+0.040	0.344
20 steps	CanonicalMerge vs. BagMerge default (q-blind)	−0.040	0.388
20 steps	CanonicalMerge vs. BagMerge swap (q-known)	+0.030	0.607
20 steps	CanonicalMerge vs. BagMerge swap (q-blind)	+0.030	0.549

Table 4: Paired McNemar tests on the headline matrix, $n = 100$ per pair. Significant at $\alpha = 0.05$ in bold. A multi-seed replication (3 draws, $n = 300$ pooled) with paired bootstrap CIs and Holm correction is in Appendix B.

Multi-seed replication. To separate draw variance from method effect, we re-ran the accuracy comparison on Qwen3-1.7B under greedy decoding across three independent problem draws (data seeds $\{42, 7, 100\}$; $n = 100$ each, $n = 300$ pooled), with per-problem paired bootstrap CIs and Holm correction over the contrast family (Appendix B); this problem-draw replication is the primary inference for the $N = 2$ claim, and it confirms and sharpens the single-seed reading. Against the *best* fixed ordering in each regime, CanonicalMerge is statistically indistinguishable (query-known -0.007 $[-0.040, +0.027]$ versus swap, query-blind -0.003 $[-0.043, +0.037]$ versus default, both CIs straddling zero), while against the *worse* ordering it wins in both regimes by $+0.063$ $[+0.037, +0.093]$ (Holm-significant). BagMerge's order-sensitivity is itself significant and *inverts* between regimes: swap beats default by 7 pp in query-known, default beats swap by 7 pp in query-blind (both survive Holm), so no fixed ordering is safe in advance, exactly the choice CanonicalMerge eliminates. The query-blind CanonicalMerge accuracy varies across draws (0.86/0.93/0.86) around the BagMerge-default parity line, so the single-draw -0.03 of Table 4 reflects between-draw variation rather than a stable deficit; only the multi-seed mean is a reliable comparator, and it places CanonicalMerge at parity with the best ordering it never had to select.

What significance means here. The lack of 20-step significance does not weaken the paper's contribution. Statistical significance at $n = 100$ measures whether an *accuracy gap* between two methods is detectable in a finite sample; the *structural property* the paper claims for CanonicalMerge, commutativity and byte-identity at $N = 2$ (Equation (4)), is verified categorically by Algorithm 1's symmetric score function and content-determined sort, and is confirmed bit-for-bit on the actual 28-layer Qwen3-1.7B KV state for the headline configuration (Appendix C). The structural property holds for every problem the method ever processes, not just for the 100 tested here.

5.4 Layer-sensitivity sweep

CanonicalMerge's only architectural choice is the routing layer $\ell^\star$ at which the K-norm content function (Equation (3)) is computed. We swept $\ell^\star \in \{5, 10, 12, 15, 18, 22, 27\}$ on partitioned $k = 2$ in the query-known regime at 40 latent steps (same protocol as Table 3), holding all other settings fixed.

The construction is robust within a measurable plateau rather than at a knife-edge optimum: the mid-network band $\ell^\star \in \{12, 15, 22\}$ all land between 0.89 and 0.94, within 5 pp of the peak. The early- and late-layer endpoints behave as the interpretability prior on decoder-only LMs predicts (Tenney et al., 2019; Geva et al., 2021; Belrose et al., 2023): early layers carry token-level features that lack the abstract semantic structure for K-norm to function as an informational-density proxy; late layers are prediction-biased and project K-norm onto next-token probabilities rather than fragment content. Within the mid-network band the curve is not strictly unimodal, a dip at $\ell^\star = 18$ separates the peak at 15 from a secondary near-peak at 22, but the band as a whole is broad enough that the central claim does not depend on locating $\ell^\star$ to within a single layer.

Held-out validation. The sweep above shares its problem draw with Table 3, so on its own it cannot rule out that $\ell^\star = 15$ was selected on the evaluation set. We therefore re-swept a finer mid-network grid,

$\ell^\star$	accuracy	reading
5	0.59	early — token-level features, no abstract routing signal
10	0.83	rising
12	0.89	approaching peak
15	**0.94**	**peak** — default choice
18	0.86	post-peak dip
22	0.93	secondary near-peak (-1 pp from peak)
27	0.84	late — prediction-biased

Table 5: Routing-layer sensitivity for CanonicalMerge on partitioned $k = 2$ (query-known, 40 latent steps, 100 problems).

Method	q-known: $\Delta(40-20)$	q-blind: $\Delta(40-20)$
BagMerge default	-0.04	-0.05
BagMerge swap	$+0.01$	-0.06
CanonicalMerge	$\mathbf{+0.01}$	$\mathbf{-0.01}$

Table 6: Latent-budget sensitivity: per-method accuracy change when moving from 20 to 40 latent steps in each regime, extracted from Table 3. CanonicalMerge is essentially flat across both regimes; BagMerge shows regime-conditional budget effects that change sign across the two orderings.

$\ell^\star \in \{9, 11, 13, 15, 17, 19, 21\}$, on an *independent* problem draw (data seed 7, held out from the original selection; query-known, greedy, $n = 100$). The accuracies are 0.99/0.88/0.88/0.95/0.89/0.92/0.89: the chosen $\ell^\star = 15$ lands at 0.95, in the upper part of a broad (if uneven) high band across the mid-network range, showing that the selected layer remains in a high-performing band on a held-out draw rather than being an evaluation-set artifact. Consistent with a plateau rather than a knife-edge optimum, the exact argmax is not stable across draws: it is 15 on the selection draw and 9 on this held-out draw, both within the ± 7 pp sampling noise at $n = 100$, so the robustness is to the mid-network *band*, not to any single layer index. In practical terms, CanonicalMerge does not require the routing layer to be tuned precisely: any mid-network layer supplies a usable K-norm content signal.

5.5 Latent-budget robustness

The latent-step budget is a generation-time hyperparameter independent of the merge choice: each thinker runs the same number of latent reasoning steps after seeing its prompt, then the merge operates on the resulting caches. Table 3 reports 20- and 40-step accuracies; Table 6 extracts the $40 - 20$ per-method delta in each regime.

CanonicalMerge is essentially flat across budgets ($\Delta = +0.01$ in q-known, -0.01 in q-blind; both within bootstrap CI noise on $n = 100$, Appendix B). BagMerge is not: its default ordering loses 4 pp in q-known and 5 pp in q-blind going $20 \to 40$; its swap ordering gains 1 pp in q-known but loses 6 pp in q-blind. The signs change with both regime and ordering. A deployer choosing CanonicalMerge does not need to tune the latent budget; a deployer choosing BagMerge would need to know both the regime and the chosen ordering to avoid a 5–6 pp accuracy cost.

This budget robustness is *empirical*, a tuning-robustness observation, not a byte-level invariant, and joins the empirical regime robustness (the q-known/q-blind sign flip in the *best* BagMerge row of Table 3) alongside the one exact, *structural* guarantee here: input-order byte-identity (Equation (4)).

5.6 Output-fusion baseline: PackLLM

The closest training-free neighbor to CanonicalMerge in mechanism is PackLLM (Mavromatis et al., 2024), which computes a content-determined weighting $\lambda_k = \mathrm{softmax}(-\log \mathrm{PPL}_k/\tau)$ at test time and applies the weighting to the output logits rather than to the cache. PackLLM shares the "training-free, content-

arm	regime	τ	CanonicalMerge ref	PackLLM	gap
A1	q-known	1.0	0.94	0.38	−56 pp
A3	q-blind	1.0	0.85	0.34	−51 pp
A4	q-blind	0.01 (one-hot floor)	—	0.01	catastrophic

Table 7: PackLLM logit-fusion baseline at $N = 2$, 100 problems each. CanonicalMerge reference numbers are taken from the 40-step column of Table 3. The $\tau = 0.01$ arm collapses to one-hot weighting on the lower-PPL thinker; the resulting accuracy of 0.01 is the partitioned-blind floor (a single thinker's solo decode from one fragment). These arms use PackLLM's reported 512-token budget and are the reported-budget diagnostic; the primary fairness comparison is at the matched 2048-token budget (CanonicalMerge 0.947 vs. PackLLM 0.50, +45 pp over three draws; Appendix B).

Method	F1	EM
C0 full-context (single thinker on $t_a + t_b$)	0.610	0.460
BagMerge default (prior work)	0.632	0.420
CanonicalMerge	0.605	0.460

Table 8: HotpotQA bridge-$k = 2$ (50 problems, 2 supporting-fact paragraphs each), free-form text scored with the standard HotpotQA normalisation.

determined, test-time" mechanism with CanonicalMerge but operates one level of abstraction later in the pipeline. We adapt PackLLM to our setting, same Qwen3-1.7B base, same $N = 2$ partitioned thinkers, the judger's prompt extended through each thinker's cache before the per-step logit fusion begins; full code path in Appendix D, and report three arms × 100 problems in Table 7.

(1) The CanonicalMerge–PackLLM gap is structural, not marginal. The two $\tau = 1.0$ arms show CanonicalMerge ahead by 51–56 pp, an order of magnitude larger than any pairwise gap in Table 3. These arms use PackLLM's reported 512-token budget; at the matched 2048-token budget the query-known gap is still $+0.447$ $[+0.387, +0.503]$ (45 pp, three seeds; Appendix B), confirming it is a fusion-level limitation rather than a generation-length artifact. Cache-level merging is not a marginal improvement over logit-level fusion in this setting; the two regimes are qualitatively different. **(2) The PackLLM-faithful setting and the one-hot limit do not rescue it.** At $\tau = 1.0$ the λ weights cluster near uniform (mean $\lambda_0 = 0.49$, $\sigma = 0.05$ in A1) because the two thinkers' judger-prompt perplexities are too close for the soft assignment to distinguish them. At $\tau = 0.01$ the weighting collapses to one-hot on the lower-PPL thinker, decoding from a single partition's view, which lands at the partitioned-blind floor of 0.01. The structural reason is that by the time information reaches the output head each thinker has compressed its per-step reasoning into a sparse next-token distribution over the vocabulary; per-step weighted averaging of two such compressed distributions cannot synthesise content that no single thinker individually emits, because averaging two solo decodes from two partial partitions does not reconstruct the joint constraint that requires both. Cache-level merging preserves the intermediate K/V geometry that the judger's attention can still route over.

5.7 Cross-benchmark validation: HotpotQA bridge-$k = 2$

To probe whether the merge primitive transfers from the synthetic partitioned construction of Section 3 to a real-world multi-document QA benchmark, we evaluated CanonicalMerge on 50 HotpotQA (Yang et al., 2018) "bridge" questions filtered to exactly two supporting-fact paragraphs. Each problem provides two paragraphs that jointly determine the answer; we map the two supporting paragraphs to the two thinkers ($t_a \to \text{thinker}_0$, $t_b \to \text{thinker}_1$). Answers are free-form text scored with the standard HotpotQA F1 and EM normalisation.

CanonicalMerge's F1 of 0.605 is within sample variance of the C0 full-context baseline's 0.610 (the per-sample bootstrap CI overlaps at $n = 50$), and the two methods agree on EM at 0.460. We read this as *no detectable degradation* on this small bridge subset, not as an equivalence result: a 50-problem sample cannot establish equivalence, and BagMerge default in fact scores marginally higher on F1 (0.632), so we make no

Method	query-known	query-blind
BagMerge default	0.76	0.88
BagMerge swap	**0.84**	0.92
CanonicalMerge	0.83	**0.93**
Δ (CM − best BagMerge)	−0.01	+0.01

Table 9: Headline comparison on Qwen3-4B, partitioned $k = 2$, 40 latent steps, $\ell^\star = 24$, 100 problems per cell. Best BagMerge ordering per column in bold.

accuracy-ranking claim here (Section 7). The training-free merge transfers from the synthetic partitioned math benchmark where the routing-layer choice $\ell^\star = 15$ was tuned to a structurally different real-world benchmark without a per-domain content function, prompt template, or any other domain-conditional hyperparameter. The result complements Table 3: the same K-norm at $\ell^\star = 15$ that handles the synthetic partitioned math also handles real multi-document QA, suggesting the content-density signal generalises across benchmark families.

5.8 Replication at a second model scale (Qwen3-4B)

To test whether the findings are specific to Qwen3-1.7B, we repeated the headline comparison on Qwen3-4B (36 layers vs. 28). The routing layer does not transfer by index, $\ell^\star = 15$ was selected for the 28-layer model, so we re-chose it by a coarse sweep over $\ell^\star \in \{15, 19, 24\}$ on a 50-problem query-known subset, which favoured $\ell^\star = 24$ (accuracies 0.68/0.70/0.72, mutually within sampling noise). Table 9 reports the four-cell comparison at 40 latent steps and $\ell^\star = 24$; single-thinker baselines on 4B give a full-text ceiling of 1.00 and single-fragment floors of 0.00, so the partition is at least as strong as at 1.7B.

Three of the four claims replicate directly. *(1) The directionality flaw persists*: BagMerge swap and default differ by 8 pp (query-known) and 4 pp (query-blind). *(2) CanonicalMerge matches the best* BagMerge *ordering without knowing which it is*, to within ±1 pp in both regimes, as at 1.7B (Table 3). *(3) Commutativity holds bit-for-bit* on the 4B model: all 36 layers of the merged KV state hash byte-identical under input reordering (Appendix C), as at 1.7B.

The fourth observation does *not* replicate, and it sharpens the argument rather than weakening it. At 1.7B the optimal BagMerge ordering inverts between regimes (swap best in query-known, default in query-blind); at 4B, swap is best in *both* regimes. Across the four model–regime cells the best ordering runs swap / default / swap / swap, so it is not governed by any regime rule a deployer could apply in advance; it depends jointly on the model and the regime. CanonicalMerge sidesteps the choice at both scales, landing within 1 pp of the best ordering in every cell. We also note, without over-interpreting a single comparison, that 4B is stronger in the query-blind regime than query-known, the reverse of 1.7B, and that 4B's absolute accuracies sit below 1.7B's on this task; we treat both as observations rather than claims.

5.9 Re-delivery: duplicate absorption is load-bearing

The structural guarantees of LatentFragmentSet (Section 4.4) are exact by construction; this experiment shows they are also *load-bearing*, in that the alternative, re-concatenating duplicate deliveries, actively degrades both accuracy and state size. On the partitioned $k = 2$ benchmark (query-known, greedy, 40 latent steps) we re-deliver each distinct fragment R times, the redundancy a delivery layer without message-level deduplication produces (gossip, retry, or re-broadcast), and merge the deliveries under two policies. *Naive* treats each delivery as a fresh cache operand, as a BagMerge-style rendered-cache operator does absent provenance: it forms the multiset and renders all R copies of each fragment. *Crdt* inserts each delivery into the LatentFragmentSet, where byte-identical copies hash equal and collapse, so the rendered cache is a pure function of the two distinct fragments regardless of R.

Table 10 reports accuracy (with 95% Wilson intervals) and the merged cache length against R. The policies coincide at $R = 1$ (no duplication) at 0.93, matching the headline $k = 2$ query-known accuracy to within sampling noise (Table 3). As R grows they diverge sharply. Naive re-concatenation degrades monotonically

	naive (multiset re-render)		crdt (LatentFragmentSet)		
R	accuracy	seq. len	accuracy	seq. len	set size
1	0.93 [.86, .97]	308	0.93 [.86, .97]	308	—
2	0.85 [.77, .91]	616	0.93 [.86, .97]	308	2
3	0.71 [.61, .79]	924	0.93 [.86, .97]	308	2
4	0.45 [.36, .55]	1232	0.93 [.86, .97]	308	2

Table 10: Re-delivery robustness on partitioned $k = 2$ (Qwen3-1.7B, query-known, greedy, $n = 100$, 95% Wilson intervals). Each distinct fragment is delivered R times. *Naive* re-renders the multiset: accuracy falls $0.93 \to 0.45$ and the cache grows $R\times$. *Crdt* absorbs byte-identical copies by content hash (absorbed set size $= 2$ at every R), so accuracy and cache length are invariant in R; the policies coincide at $R = 1$. This is the idempotence / absorption property of Section 4.4 exercised behaviourally.

($0.93 \to 0.85 \to 0.71 \to 0.45$) while its cache grows exactly linearly in R ($308 \to 616 \to 924 \to 1232$ tokens, i.e. $R\times$): the redundant copies both bloat the prefix and pull the decode off the answer. The CRDT policy is *flat on both axes*: accuracy holds at 0.93 (95% CI $[0.86, 0.97]$) and the rendered length stays at 308 tokens for every R, with the absorbed set size reported as 2 on every problem. By $R = 4$ the gap is $+0.48$ accuracy at one quarter of the state. Idempotent absorption is therefore not a formal nicety: under realistic re-delivery it is the difference between a bounded, stable system and one that silently inflates and degrades.

6 Discussion

6.1 Robustness across four axes: two structural, two empirical

The headline matrix (Table 3) and the re-delivery analysis expose four axes along which BagMerge is fragile, of two distinct kinds. *Structural / algebraic* (exact, by construction): CanonicalMerge is byte-identical under input reordering, and LatentFragmentSet absorbs duplicate re-deliveries. *Empirical / tuning* robustness (quality stays close in the tested matrix, an observation, not a byte-level invariant): across regime/model and latent budget. We claim the first kind as guarantees and the second as measured robustness, and do not call the latter "structural."

Input order (structural). BagMerge is non-commutative, and its position-0 prefix slot confers a measurable advantage (Section 4.2); which thinker receives it is fixed by arbitrary caller order. CanonicalMerge removes the decision: its output is byte-identical under any permutation of the inputs (Equation (4)), verified algorithmically (Appendix A) and on real Qwen3-1.7B caches (Appendix C).

Regime and model (empirical). Which BagMerge ordering is best is not governed by any rule a deployer could apply in advance. At 1.7B it inverts between regimes, swap best when thinkers see the query, default when they do not (Table 3), while at 4B swap wins both regimes (Section 5.8). Across the four model–regime cells the best ordering runs swap / default / swap / swap, with 6–8 percentage points at stake from the wrong choice and no ordering stable across the observed regimes and scales, so deploying BagMerge requires validation or prior regime knowledge. CanonicalMerge removes the choice at both scales.

Latent budget (empirical). The optimal ordering and the latent-step budget interact: BagMerge loses 5–6 pp from the wrong budget in the query-blind regime, and the sign of the budget effect itself changes with the ordering (Section 5.5). CanonicalMerge is flat across budgets (± 1 pp).

Duplication (structural). The first three axes are tuning choices a deployer must get right; the fourth is a delivery hazard they cannot tune away. Under re-delivery, the same fragment arriving more than once, as in gossip or retry, BagMerge and the rendered operator re-concatenate the copy, growing the cache and altering the decoded output, whereas the LatentFragmentSet state (Section 4.4) absorbs it by content hash, leaving state and render invariant (Section 5.9): under R-fold re-delivery naive re-concatenation falls to 0.45 accuracy at $4\times$ the cache, while the CRDT holds 0.93 at constant size (Table 10).

In every cell of the matrix, the best BagMerge configuration requires the deployer to know the regime and to have tuned both the ordering and the budget for it; CanonicalMerge attains accuracy within 4 pp of that best-case configuration, statistically indistinguishable from it within the bootstrap confidence intervals (Section 5.3), with no configuration at all. We therefore read the contribution as trading a small, statistically insignificant accuracy margin (the empirical axes) for the two unconditional structural guarantees, input-order byte-identity and duplicate absorption, that hold for every input.

6.2 What the content function buys, and where it does not help

The mean-K-norm score (Equation (3)) routes the higher informational-density thinker to the privileged prefix slot *per problem*, rather than committing to a single global order. This matters because the binding constraint is not always carried by the same fragment: across our families, sometimes the ratio (e.g. CONSTRAINT) and sometimes the lookup table (e.g. LOOKUP) is the more information-dense half. A fixed input order, the only option under BagMerge, cannot adapt to this, whereas content routing selects the prefix occupant problem-by-problem. That the K-norm at a *mid-network* layer is the right signal is consistent with the interpretability picture of decoder-only LMs (Tenney et al., 2019; Geva et al., 2021; Belrose et al., 2023), in which the middle band carries abstract, task-relevant features rather than token-surface or next-token-prediction features; the layer-sensitivity sweep (Section 5.4) bears this out, with a broad plateau across the middle third of the network.

The content function is, however, *query-conditional*, and this bounds where it helps. When a thinker has not seen the query (query-blind), its K-norm profile no longer concentrates on question-relevant constraints, and the prefix selection degrades toward a content-agnostic tiebreak (Section 5.2). This is visible in the one cell where CanonicalMerge underperforms, query-blind at 20 steps, 4 pp below BagMerge default (Table 3), where BagMerge's fixed lexical ordering happens to align with the answer better than a weak content signal does. We do not claim CanonicalMerge dominates on accuracy in this regime; we claim it removes the ordering decision unconditionally, and matches or exceeds the best fixed ordering in three of the four cells while never trailing it by more than a single non-significant margin.

6.3 Cache-level merging as a distinct operating regime

The gap to PackLLM (Section 5.6) is not a tuning artifact but a consequence of *where* the merge happens. Logit-level fusion combines per-thinker next-token distributions after each thinker has already compressed its reasoning into a sparse vocabulary-sized vector; no weighting of those vectors can reconstruct joint content that neither thinker individually emits. Cache-level merging operates one stage earlier, on the K/V geometry the judger's attention can still route over, which is why it recovers the partitioned answer where logit fusion cannot, by more than 45 percentage points even at the matched generation budget.

This places CanonicalMerge in an operating regime that the existing multi-LLM-combination literature does not name. Three independent surveys of the field (Chen et al., 2025; Lu et al., 2024; Li et al., 2023) partition it into parameter-level *merging* (combining weights) and output-level *ensembling* (combining logits or decoded text); none has a category for combining intermediate KV-cache state. The closest training-free neighbor, PackLLM, sits firmly on the output side. We view cache-level merging as a third regime, and the convergent-state property as what makes it operationally useful: a merge that is order-independent, byte-reproducible, and duplicate-absorbing is precisely what lets independent parties contribute encoded state without a coordinator dictating order or absorbing duplicate re-deliveries, the defining requirement of the federated / query-blind setting (Section 3.4). The broader properties this distributed-systems analogy suggests, idempotent, absorbing set-union semantics, are realised by the LatentFragmentSet state (Section 4.4) and exercised behaviourally under re-delivery (Section 5.9); their behaviour at $N > 2$, where transport does not suffice for composition, is taken up in Section 8.

7 Limitations

The headline $N = 2$ accuracy matrix is replicated across three independent problem draws for Qwen3-1.7B under greedy decoding (Appendix B), and byte equality is checked at both 1.7B and 4B; most secondary

analyses, however, use a single draw, the models are from a single family, and both scales (Qwen3-1.7B and 4B, Section 5.8) sit below the 8B threshold at which Jin et al. (2026) report substantially larger RoPE-related effects, so the picture may shift again at the scales they study. The benchmark family is primarily synthetic; we validate cross-benchmark transfer on a 50-problem HotpotQA bridge-$k = 2$ subset (Section 5.7), but broader multi-document QA evaluation (e.g. MuSiQue, full HotpotQA) remains future work.

The convergence guarantees of LatentFragmentSet (Section 4.4) are *syntactic*: the set absorbs byte-identical re-deliveries but not semantic near-duplicates; a semantically convergent join that also merges near-duplicate but non-identical fragments is future work. Render is also lazy, each newly absorbed fragment forces an $\mathcal{O}(m)$ re-render of the m-fragment set at the next decode, with no incremental reuse across rounds. This is negligible for batch inference and is the price of exact semantics under streaming re-delivery. (The *rendered* CanonicalMerge operator, taken alone, is not idempotent; the algebraic properties hold on the state, not on the operator, Section 4.4.)

Our byte-identity claims are verified within a single run on a fixed model and device. The cross-party federated reading (Section 2.5) carries one further condition we make explicit, following Gillespie (2026): bit-for-bit equality assumes *matched arithmetic*, identical model weights and a deterministic, IEEE-754-consistent kernel path across parties (same hardware/ISA, no nondeterministic reductions). Under heterogeneous hardware, independently encoded fragments may differ in their low-order bits, which would defeat content-hash absorption and exact byte-equality across replicas. The content-determined *ordering* and the set-union *convergence* algebra are unaffected, they depend on fragment identity, not on cross-device bit-reproducibility, but byte-equality across heterogeneous accelerators is not claimed; a tolerance-based fragment identity is the natural remedy and is left to future work.

Finally, we do not compare against text-level fusion baselines (RAG, FiD); the latent construction's primary advantage, commutativity and byte-identity, is a structural property not measurable in text space, but a text-fusion accuracy baseline at matched compute would address the obvious reviewer question of "is this better than concatenating the fragments as text?"

8 Beyond $N = 2$

This paper's empirical claims are for $N = 2$. We close by reporting, as a boundary result, what happens at $k = 3$: it delimits the contribution, and the negative result is itself informative. The headline is a clean mechanistic statement, *CRDT cache merge is a transport and colocation operator, not a composition operator*, which we both establish and dissect, then use to frame the open problem.

The state extends exactly; the open question is composition. The replicated state is N-general by construction: set union (Equation (5)) is defined for any number of fragments, and the render inherits the byte-identity we verify across all $N!$ input permutations for $N \in \{2, 3, 4, 5\}$, 152 permutation pairs, on synthetic tensors (Appendix A). What neither the set nor its render can guarantee is that the *model* composes the fragments they colocate. At $k = 3$ that question has a sharp answer.

Transport is not composition. We built a $k = 3$ partitioned benchmark (three jointly-necessary fragments, the same five families). It is sound: the full-text ceiling is $C_0 \approx 1.00$ (0.97–1.00 across families) while every $(k-1)$-subset baseline scores $\leq 5\%$, so any latent gap is a *composition* failure, not task difficulty. Table 11 probes four ways to combine the three latent traces (depicted in Figure 2), on the two families that mark the poles of the effect (full five-family results in Appendix F). A flat 3-way CanonicalMerge (par3) collapses to chance on chain-structured problems, and so does a pairwise merge *tree* (treemerge). For the tested families, then, cache-only colocation is insufficient and a pairwise cache-merge tree does not by itself create the missing composition, it is not the merge arity or the flat-versus-tree shape. Composition appears only when the model reprocesses one or more seams: running the second thinker on the first's latent state (mix), or chaining all three (seq3), recovers quality up to the full-text ceiling. The merge moves latent traces into a shared cache, deterministically and order-invariantly; composition, on this evidence, requires the model to reprocess them. (This does not contradict Agent Primitives' working $N = 3$ merge, which aggregates among complete voting candidates rather than composing jointly-necessary fragments; Section 2.3.) This

Table 11: The $k = 3$ topology probe, on the two families that mark the poles of the effect (Qwen3-1.7B, greedy; *across-draw mean* $\pm$ *sd* over 3 independent problem draws, data seeds $\{42, 7, 100\}$, $n = 50$/family/draw; full five-family results in Appendix F). Both problems are near-perfectly solvable from full text ($C_0 \approx 1.00$), so every latent gap is composition, not difficulty. A flat 3-way merge (PAR3) and a pairwise merge tree (TREEMERGE) both collapse; introducing model reprocessing at one seam (MIX) or throughout (SEQ3) recovers it.

	NEONYM (chain)	CONSTRAINT (symmetric)
C_0 (full text)	1.00 ± 0.00	0.97 ± 0.03
PAR3 — 3-way merge $M(C_1, C_2, C_3)$	0.07 ± 0.02	0.03 ± 0.01
TREEMERGE — pairwise $M(M(C_1, C_2), C_3)$	0.06 ± 0.02	0.01 ± 0.01
MIX — one attention seam, then 2-way merge	0.73 ± 0.03	0.03 ± 0.01
SEQ3 — sequential chain, no merge	0.97 ± 0.01	0.17 ± 0.06

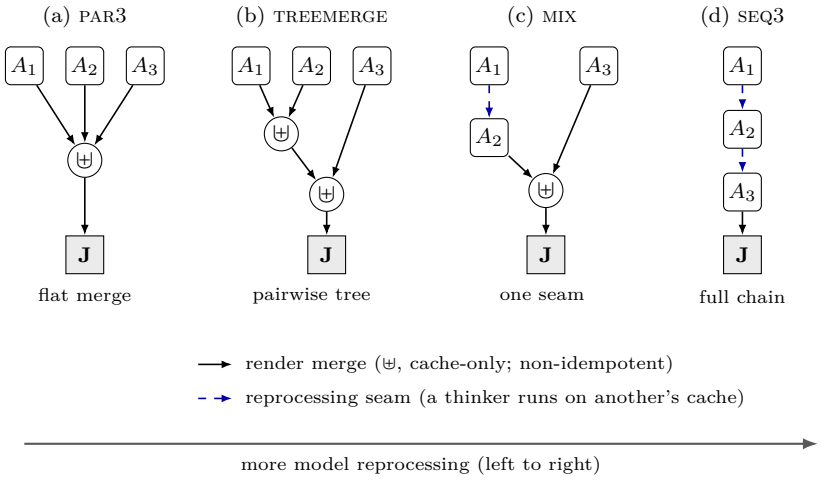

Figure 2: Four ways to combine three latent traces at $k = 3$, a topology *schematic*, not an algebraic identity (accuracies in Table 11). Solid nodes (⊎) are cache-only *render merges* (non-idempotent; *not* the idempotent set-union state join $\sqcup$ of Section 4.4, which we reserve for LatentFragmentSet). Dashed edges are model-mediated *reprocessing seams*, where one thinker runs with another's KV cache as a prefix. The cache-only flat (a) and tree (b) variants both fail and collapse to the same accuracy (the TREEMERGE $\approx$ PAR3 result), while the variants that add reprocessing ((c), (d)) recover on chain-structured tasks. The topologies differ in more than edge type, they also change which thinker reprocesses which prefix and the amount of model computation, so we read the figure as locating *where* reprocessing enters, not as a one-variable causal claim.

boundary is consistent with independent recent findings that continuous-thought latent traces transport and scaffold but do not by themselves perform the reasoning (Zhang et al., 2025; Zhu et al., 2025), that learned KV-consolidation at reasoning-step boundaries outperforms plain cache accumulation (Oomerjee et al., 2025), and that training-free parallel-cache synthesis underperforms until a learned calibration step is added (Liu et al., 2026), a pattern consistent with model reprocessing, rather than colocation, being what supplies composition.

A spectrum, and a composition-limited pole. The recovery is a spectrum, not a switch. Pooling per-problem paired differences across the three draws, the single-seam recovery (MIX over PAR3) is $+0.51$ (95% bootstrap CI $[+0.44, +0.57]$, $n = 300$) on the chain families but only $+0.14$ ($[+0.10, +0.18]$, $n = 450$) on the symmetric ones, with a composition-limited pole, the symmetric CONSTRAINT family, where even a full sequential chain reaches only 0.17 ± 0.06 and reprocessing barely helps. The boundary is not specific to

caches: PackLLM, our output-level fusion baseline (Section 5.6), also collapses at $k = 3$ ($0.04 \rightarrow 0.00$ at the longer budget), and it is not an artifact of model capacity or seam geometry, Qwen3-4B does not recover it, and inserting RoPE-phase gaps or boundary-marker tokens between thinker blocks only lowers accuracy. In every case the latent traces are present; what is missing is composition.

The open problem. This delimits the present contribution and frames the next. At $N = 2$, a content-determined render over a set-union state gives order-invariant, convergent latent merging at no measurable quality cost. At $k > 2$, the open problem is to recover the integration that, here, only model reprocessing provides, while preserving as much of that order-invariant state discipline as possible. We pursue that direction separately.

9 Conclusion

We began from a concrete flaw in how multi-agent latent reasoning composes KV-cache state: the concat-with-RoPE operator implicit in prior work, BagMerge, is non-commutative, and the best input ordering is unpredictable across regime, latent budget, and model scale. CanonicalMerge removes the decision by fixing the layout on content rather than caller order, rendering the merged cache byte-identical under any permutation of the inputs, a property we verify algorithmically up to five agents and bit-for-bit on the real KV state of Qwen3-1.7B and 4B. Separating the replicated state from its decode-time layout then lifts this from a property of an operator to a property of a durable object: a set of content-addressed latent fragments whose set-union merge is a state-based CvRDT, of which CanonicalMerge is the deterministic render. On a benchmark built so that the merge sits on the critical path, CanonicalMerge matches the best fixed ordering in each regime without needing to select it (multi-seed, paired-bootstrap, Holm-corrected), operates in a regime distinct from output-level fusion (PackLLM trails by 45 points at matched budget), and, under repeated fragment re-delivery, holds accuracy and cache size flat where a provenance-free operator degrades and grows linearly with repeated delivery.

Two implications follow. First, treating inter-agent cache exchange as a convergent replicated state gives federated or query-blind latent inference structural rather than coordinative guarantees: independent parties can contribute pre-encoded state in any order, under any delivery schedule, and with duplicates, and obtain the same rendered context, without a coordinator dictating order or deduplicating messages. This places cross-agent KV-cache merging as an inference-time primitive on equal footing with the within-sequence KV-cache compression the systems literature already treats as first-class. Second, the approach has a clear boundary: beyond two agents, cache merging transports and colocates latent traces but does not by itself compose them; a pairwise merge tree buys nothing over a flat merge, and in the tested families can even lose; composition is recovered only when the model reprocesses the colocated state through attention. Turning transport into composition, without paying the full cost of sequential reprocessing, is the problem we leave to future work.

References

Nora Belrose et al. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint*, 2023.

Xinyun Chen et al. Universal self-consistency for large language model generation. *arXiv preprint*, 2023. arXiv:2311.17311.

Zhijun Chen, Xiaodong Lu, Jingzheng Li, Pengpeng Chen, Zhuoran Li, Kai Sun, Yuankai Luo, Qianren Mao, Ming Li, Likang Xiao, Dingqi Yang, Xiao Huang, Yikun Ban, Hailong Sun, and Philip S. Yu. Harnessing multiple large language models: A survey on LLM ensemble. *arXiv preprint*, 2025. arXiv:2502.18036.

Chengming Cui, Tianxin Wei, Ziyi Chen, Ruizhong Qiu, Zhichen Zeng, Zhining Liu, Xuying Ning, Duo Zhou, and Jingrui He. AdaFuse: Adaptive ensemble decoding with test-time scaling for LLMs, 2026. arXiv:2601.06022.

Arthur Douillard et al. DiLoCo: Distributed low-communication training of language models. *arXiv preprint*, 2023. arXiv:2311.08105.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *arXiv preprint*, 2023. arXiv:2305.14325.

Zhuoyun Du, Runze Wang, Huiyu Bai, Zouying Cao, Xiaoyong Zhu, Yu Cheng, Bo Zheng, Wei Chen, and Haochao Ying. Enabling agents to communicate entirely in latent space. *arXiv preprint*, 2025. arXiv:2511.09149.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *EMNLP*, 2021.

Ryan Gillespie. Conflict-free replicated data types for neural network model merging: A two-layer architecture enabling CRDT-compliant model merging across 26 strategies, 2026. arXiv:2605.19373.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. *arXiv preprint*, 2023. arXiv:2302.12173.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *arXiv preprint*, 2024. arXiv:2412.06769.

Haibo Jin, Kuang Peng, Ye Yu, Xiaopeng Yuan, and Haohan Wang. Agent primitives: Reusable latent building blocks for multi-agent systems. *arXiv preprint*, 2026. arXiv:2602.03695.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Symposium on Operating Systems Principles (SOSP)*, 2023. arXiv:2309.06180.

Patrick Lewis et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*, 2020. arXiv:2005.11401.

Weishi Li, Yong Peng, Miao Zhang, Liang Ding, Han Hu, and Li Shen. Deep model fusion: A survey. *arXiv preprint*, 2023. arXiv:2309.15698.

Yuhong Li et al. SnapKV: LLM knows what you are looking for before generation. *arXiv preprint*, 2024. arXiv:2404.14469.

Shikun Liu, Mufei Li, Dongqi Fu, Haoyu Wang, Yinglong Xia, Hong Li, Hong Yan, and Pan Li. Towards direct latent-space synthesis for parallel branches in LLM-agent workflows, 2026. arXiv:2606.14672.

Yingzhuo Liu. Beyond tokens: A unified framework for latent communication in LLM-based multi-agent systems, 2026. arXiv:2606.05711.

Jinliang Lu, Ziliang Pang, Min Xiao, Yaochen Zhu, Rui Xia, and Jiajun Zhang. Merge, ensemble, and cooperate! a survey on collaborative strategies in the era of large language models. *arXiv preprint*, 2024. arXiv:2407.06089.

Costas Mavromatis, Petros Karypis, and George Karypis. Pack of LLMs: Model fusion at test-time via perplexity optimization. *arXiv preprint*, 2024. arXiv:2404.11531.

Adnan Oomerjee, Zafeirios Fountas, Haitham Bou-Ammar, and Jun Wang. Bottlenecked transformers: Periodic KV cache consolidation for generalised reasoning, 2025. arXiv:2505.16950.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *arXiv preprint*, 2024. arXiv:2309.00071.

Nuno Preguiça, Carlos Baquero, and Marc Shapiro. Conflict-free replicated data types CRDTs, 2018. arXiv:1805.06358; Springer Encyclopedia entry.

Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. Conflict-free replicated data types. In *Symposium on Self-Stabilizing Systems (SSS)*, 2011. Also INRIA RR-7506.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 2024. arXiv:2104.09864.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the NLP pipeline. In *Association for Computational Linguistics (ACL)*, 2019.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. *arXiv preprint*, 2024a. arXiv:2406.04692.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2203.11171.

Zheng Wang, Boxiao Jin, Zhongzhi Yu, and Minjia Zhang. Model tells you where to merge: Adaptive KV cache merging for LLMs on long-context tasks, 2024b. arXiv:2407.08454.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2309.17453.

Xiyuan Yang, Jiaru Zou, Rui Pan, Ruizhong Qiu, Pan Lu, Shizhe Diao, Jindong Jiang, Hanghang Tong, Tong Zhang, Markus J. Buehler, Jingrui He, and James Zou. Recursive multi-agent systems, 2026. arXiv:2604.25917.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, 2018.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *NeurIPS*, 2023. arXiv:2305.10601.

Yuxuan Yao, Han Wu, Mingyang Liu, Sichun Luo, Xiongwei Han, Jie Liu, Zhijiang Guo, and Linqi Song. Determine-then-ensemble: Necessity of top-k union for large language model ensembling, 2024. arXiv:2410.03777.

Xinlei Yu et al. The latent space: Foundation, evolution, mechanism, ability, and outlook. *arXiv preprint*, 2026. arXiv:2604.02029.

Yuyi Zhang, Boyu Tang, Tianjie Ju, Sufeng Duan, and Gongshen Liu. Do latent tokens think? a causal and adversarial analysis of chain-of-continuous-thought, 2025. arXiv:2512.21711.

Zhenyu Zhang et al. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *NeurIPS*, 2023. arXiv:2306.14048.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *NeurIPS Datasets and Benchmarks*, 2023. arXiv:2306.05685.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought, 2025. arXiv:2505.12514.

Jiaru Zou, Xiyuan Yang, Ruizhong Qiu, Gaotang Li, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in multi-agent systems. *arXiv preprint*, 2025. arXiv:2511.20639.

A Synthetic-tensor byte-CRDT tests

We first establish the byte-CRDT property of the render on synthetic state, with no model inference, so that permutation invariance is isolated from any modelling noise. The test (`tests/test_canonical_n.py`) builds legacy-style KV caches of small random Gaussian tensors, choosing the per-thinker scales so that the mean K-norms at the routing layer are cleanly separable and the canonical order is therefore unambiguous. Equality is checked at the bit level (`torch.equal` on the raw tensors, not approximate `allclose`). Three properties are verified.

Permutation invariance at $N > 2$. For $N \in \{2, 3, 4, 5\}$ we render every one of the $N!$ input orderings and assert each is byte-identical to a fixed reference ordering, $2 + 6 + 24 + 120 = 152$ permutations in total, all identical. This extends the commutativity of Equation (4), proved in the main text for $N = 2$, to higher arity: the content-determined layout selects the same slot for each fragment regardless of the order in which the caches are presented.

Reduction to the $N = 2$ operator. Called with two caches, the N-fold path `novelty_concat_sym_layout_n` produces output byte-identical to the dedicated two-input strategy `novelty_concat_sym_layout`, confirming that the batch generalisation introduces no regression at the arity used for every headline number.

Canonical-ordered concatenation (the headline configuration). In the configuration used throughout the paper the merge is a pure canonical-ordered concatenation: the rendered length equals the sum of the input lengths ($12, 18, 24$ for $N = 2, 3, 4$ at six positions per thinker), with no position dropped, and a further 32 permutations ($2! + 3! + 4!$) are all byte-identical. The layout reorders whole per-thinker blocks and performs no arithmetic on the merged tensors, which is what makes the byte-equality exact rather than approximate (Appendix C; the IEEE-754 condition is noted in Section 7).

Together these establish the render's input-order independence and deterministic layout on controlled state, separately from the model; Appendix C repeats the permutation check on real Qwen3 caches, and the state-level formalisation that makes this a CvRDT join rather than a property of the operator alone is given in Section 4.4.

B Bootstrap CIs and McNemar paired tests

This appendix backs the accuracy comparisons of Section 5.3 with a multi-seed analysis. We re-ran the $N = 2$ matrix, CanonicalMerge (CM), BagMerge in its default and swapped orderings (BG, BGSW), and the PackLLM output-fusion baseline, in both regimes under greedy decoding, across three independent problem draws (data seeds $\{42, 7, 100\}$, $n = 100$ each, at the 40-step budget and the 2048-token headline generation cap). Greedy decoding is deterministic, so the relevant variance is the problem draw: we report per-cell accuracy as the mean $\pm$ s.d. across draws with a pooled $n = 300$ Wilson interval (Table 12), and every method comparison as a per-problem paired bootstrap (10 000 resamples) pooled across draws, with an exact McNemar p and Holm correction over the seven-contrast family (Table 13). Analysis script: `scripts/analyze_nstats.py`. The single-seed 20/40-step bootstrap that Table 4 summarises is retained at `runs/bootstrap_results{,_20step}.md`.

Two readings follow. First, CanonicalMerge *ties the best fixed ordering in each regime*, CM − BGSW (query-known) and CM − BG (query-blind) both straddle zero and are non-significant after Holm, while it *beats the worse ordering* in both regimes by $+0.063$ ($p < 0.001$, Holm-significant). This is the "matches the best ordering without selecting it" claim, now with the between-draw variance folded in. Second, BagMerge's order-sensitivity (BG − BGSW) is significant in both regimes and *reverses sign* between them (-0.070 query-known, $+0.067$ query-blind), so no single ordering is safe across regimes, precisely the choice CanonicalMerge removes. The output-fusion baseline PackLLM sits $+0.447$ $[+0.387, +0.503]$ below CanonicalMerge even at the matched 2048-token budget, a gap an order of magnitude larger than any ordering effect (Appendix D).

method	regime	per-seed (42/7/100)	mean ± s.d.	pooled 95% CI
CanonicalMerge	known	0.96/0.95/0.93	0.95 ± 0.02	$[0.92, 0.97]$
CanonicalMerge	blind	0.86/0.93/0.86	0.88 ± 0.04	$[0.84, 0.91]$
BagMerge default	known	0.88/0.89/0.88	0.88 ± 0.01	$[0.84, 0.91]$
BagMerge default	blind	0.89/0.88/0.89	0.89 ± 0.01	$[0.85, 0.92]$
BagMerge swap	known	0.96/0.97/0.93	0.95 ± 0.02	$[0.92, 0.97]$
BagMerge swap	blind	0.81/0.83/0.82	0.82 ± 0.01	$[0.77, 0.86]$
PackLLM	known	0.51/0.49/0.50	0.50 ± 0.01	$[0.44, 0.56]$

Table 12: Per-cell accuracy across three problem draws (Qwen3-1.7B, $k = 2$, greedy, $n = 100$ per draw; pooled 95% Wilson CI at $n = 300$). Per-seed values are ordered by data seed 42/7/100. CanonicalMerge and the best-of-two BagMerge ordering coincide in each regime (swap in query-known, default in query-blind), and the low across-draw s.d. shows the numbers are draw-stable except for query-blind CanonicalMerge, whose single-draw swing motivates the paired analysis below.

contrast	Δ acc [95% CI]	McNemar p	Holm
CM − BG (known)	$+0.063$ $[+0.037, +0.093]$	< 0.001	reject
CM − BGSW (known)	-0.007 $[-0.040, +0.027]$	0.839	n.s.
CM − BG (blind)	-0.003 $[-0.043, +0.037]$	1.000	n.s.
CM − BGSW (blind)	$+0.063$ $[+0.037, +0.093]$	< 0.001	reject
BG − BGSW (known)	-0.070 $[-0.113, -0.027]$	0.003	reject
BG − BGSW (blind)	$+0.067$ $[+0.020, +0.113]$	0.011	reject
CM − PackLLM (known)	$+0.447$ $[+0.387, +0.503]$	< 0.001	reject

Table 13: Per-problem paired bootstrap (10 000 resamples, $n = 300$ pooled across the three draws), exact McNemar p, and Holm-corrected decision over the seven-contrast family. CanonicalMerge ties the best ordering in each regime (CM − BGSW query-known and CM − BG query-blind both straddle zero) and beats the worse ordering (+0.063, Holm-significant); BagMerge's order-sensitivity (BG − BGSW) is significant and reverses sign between regimes.

C Real-model byte-equality on the headline configuration

This appendix is the real-model companion to the synthetic byte-CRDT test of Appendix A: it verifies that Equation (4) holds not only on controlled random tensors but on KV caches produced by an actual model forward pass, in the exact headline configuration. The test (`tests/test_byte_equality_real_model.py`) runs Qwen3-1.7B on the first partitioned $k = 2$ problem in the query-blind regime, under the headline merge settings (`novelty_concat_sym_layout`, $\tau = 2.0$, $\ell^\star = 15$, 40 latent steps). It encodes the two fragments under both orderings, order A ($\text{thinker}_0 \leftarrow t_a,\ \text{thinker}_1 \leftarrow t_b$) and the swapped order B ($\text{thinker}_0 \leftarrow t_b,\ \text{thinker}_1 \leftarrow t_a$), the `--swap_partitioned_thinkers` setting, runs each thinker's forward pass plus latent steps to produce its cache, and merges each ordering into `merged_A` and `merged_B`.

Equality is checked at the byte level. Each layer's K and V tensors are moved to CPU, made contiguous, and reinterpreted as `uint8` over the raw `bfloat16` storage, numpy has no native `bfloat16`, and we want a bit-pattern comparison rather than a dtype-aware one, then SHA-1 hashed. All 28 layers of K and all 28 layers of V hash byte-identical between `merged_A` and `merged_B`, establishing Equation (4) on real model state for the headline benchmark configuration. This is the single strongest piece of evidence for commutativity at $N = 2$: the rendered cache is bit-for-bit independent of input order, so the downstream decode, and therefore every accuracy number in Section 5, cannot depend on which fragment was presented first. The check is exact (`torch.equal` / SHA-1, not `allclose`); the condition under which exact byte-equality is even attainable, that the canonical layout reorders whole per-thinker K and V blocks and performs no arithmetic on the merged tensors, so no floating-point non-associativity is introduced, is the IEEE-754 caveat discussed in Section 7.

Because the merge is this arithmetic-free block reordering, byte-identity is a property of the layout rather than of the model or its width: the synthetic test (Appendix A) confirms it across tensor shapes and arities $N \leq 5$ independently of any architecture. We run the same real-model check at the second model scale: on Qwen3-4B (with $\ell^\star = 24$), all 36 layers of K and V hash byte-identical between the two orderings, confirming Equation (4) on real 4B state as well; the accuracy-level replication is reported in Section 5.8.

D PackLLM adaptation details

Construction. We adapt PackLLM (Mavromatis et al., 2024) to the single-base-model, N-thinker setting so that it shares CanonicalMerge's inputs and differs only in *where* the combination happens. The two methods are identical through Phase 1: each thinker independently encodes its fragment and runs the same latent steps, producing a cache $\mathcal{C}_i$. They then diverge. CanonicalMerge merges the caches and decodes once from the merged prefix (Section 4); our PackLLM keeps the N caches separate and fuses at the logits:

1. *Phase 2 (weighting).* The judger's chat-templated prompt is encoded on top of each thinker's cache $\mathcal{C}_i$; we record the perplexity PPL_i of that prompt under each cache and the next-token logits at its final position. Weights are $\lambda_i = \mathrm{softmax}(-\log \mathrm{PPL}_i/\tau)$ ("PackLLMsim"); a greedy grid variant ("PackLLMopt", $N = 2$ only) is also implemented but not reported as it does not change the conclusion.

2. *Phase 3 (fused decode).* At each step the fused distribution is $\mathrm{softmax}\big(\sum_i \lambda_i \mathbf{z}_i\big)$ over per-thinker logits $\mathbf{z}_i$ (averaging is pre-softmax, matching PackLLM's Eq. 2); the sampled token is appended to *all* N caches so they advance in lockstep, and the per-thinker logits are recomputed for the next step.

We use the judger-prompt PPL as the weighting signal (the PackLLM-faithful choice), sample at temperature 0.7 / `top_p` = 0.95 to match CanonicalMerge (PackLLM's original uses argmax), and fuse pre-softmax. Reference implementation: `methods/packllm_baseline.py`, which subclasses the CanonicalMerge method to reuse Phase 1 verbatim.

The PackLLM-faithful setting and the one-hot limit both fail. The weighting is near-uniform because the two thinkers' judger-prompt perplexities are close: at $\tau = 1.0$, λ_0 has mean 0.49, s.d. 0.04 in the query-known arm (mean PPLs 234 vs. 223) and mean 0.52, s.d. 0.07 in query-blind. Driving $\tau \to 0$ collapses the weighting to one-hot on the lower-PPL thinker (query-blind, $\tau = 0.01$: λ_0 bimodal at $\{0, 1\}$), which decodes from a single partition and lands at the partitioned-blind floor of 0.01. With the two endpoints failing and the perplexities so nearly equal, it is unlikely that any intermediate τ turns them into a useful integration signal.

Truncation cannot account for the gap. A reviewer might worry that PackLLM's low accuracy is a generation-length artifact, since we cap decoding at 512 tokens. We bound this. Per-problem decode lengths and a truncation flag (`decode_len` $\geq$ 512) are recorded for every problem (`runs/packllm_overnight/per_sample_telemetry.csv`). Table 14 gives the worst-case ceiling under the (maximally generous) assumption that *every* truncated-and-wrong problem would become correct with unlimited budget. Even then PackLLM tops out at 0.89 (query-known) and 0.70 (query-blind), still below CanonicalMerge's 0.94/0.85 at the matched 40-step budget. The concern is then settled directly at $k = 2$: re-running PackLLM at the matched 2048-token generation budget across three problem draws ($n = 300$) gives 0.50 ± 0.01 in query-known, still $+0.447$ $[+0.387, +0.503]$ below CanonicalMerge (Appendix B), so the gap is a fusion-level limitation, not a generation-length artifact. The $k = 3$ follow-up corroborates it: re-running PackLLM at `max_new_tokens` = 2048 *lowered* accuracy from 0.04 to 0.00 (Section 8), the extra budget lets the model converge more confidently on "insufficient information" rather than synthesise an answer.

arm	actual acc	truncated@512 (wrong)	decode len (mean)	ceiling
query-known, τ=1.0	0.38	54 (51)	417	0.89
query-blind, τ=1.0	0.34	44 (36)	397	0.70
query-blind, τ=0.01	0.01	29 (28)	358	0.29

Table 14: PackLLM truncation upper bound (100 problems each, Qwen3-1.7B). "ceiling" assumes every truncated-and-wrong problem flips to correct under unlimited generation budget; it remains below CanonicalMerge's 0.94 (query-known) and 0.85 (query-blind).

E Same-data motivation

A natural alternative to the partitioned benchmark (Section 3) is a *same-data* design: two thinkers see the same problem and are diversified only by role prompts or injected noise. We did not use it, because it cannot isolate the merge operator's contribution. When every thinker receives the complete query, a single thinker already has everything needed to solve the problem, so the merge is not on the critical path, it is an optional add-on to an already-sufficient input. Any accuracy difference between a two-thinker merge and a single thinker then conflates the operator's contribution with two confounds: extra inference compute spent on the same problem, and a regularization-like effect from combining noisy copies of one input. Our own same-data pilot bore this out: the best-looking merge configuration was behaviourally indistinguishable from a single (noisy) thinker, the second cache contributed essentially nothing to the decoded answer, and the small apparent lift tracked the extra forward pass rather than any integration of a second source.

The partitioned construction removes the confound by design. Each problem is split into fragments t_a and t_b that are *individually insufficient* and *jointly necessary*: neither alone determines the answer, but together they fix it uniquely (Section 3.2). A correct answer therefore *requires* integrating both caches, putting the merge operator squarely on the critical path. The empirical validation confirms the separation (Section 3.3): the single-fragment baselines floor near chance ($C_1 = 5\%$ for t_a alone, $C_2 = 2\%$ for t_b alone) while the full-text ceiling is $C_0 = 100\%$, so the task is solvable precisely when both fragments are present. The gap between the single-fragment floor and the merged accuracy is then attributable to the merge itself, with "extra compute on the same query" eliminated as an alternative explanation by construction.

F Full five-family $k = 3$ topology results

Table 15 reports the $k = 3$ topology probe across all five families; the main-text Table 11 shows the two poles (NEONYM, CONSTRAINT). All cells are across-draw mean $\pm$ sd over three independent problem draws (data seeds $\{42, 7, 100\}$), Qwen3-1.7B, greedy decoding, $n = 50$ per family per draw. All cells, including C_0, are over the three draws; the lowest single-draw full-text ceiling is 0.94 (CONSTRAINT), so every family is near-perfectly text-solvable and the latent gaps are composition, not task difficulty.

G Reproducibility commands

Code will be released upon publication; per-run logs are written to `runs/`, and the final metrics are the JSON object on the last line of each log. The headline conditions reproduce with:

```
# C0 ceiling (single agent, both fragments visible)
python run.py --method baseline --task partitioned \
  --partitioned_view full --model_name Qwen/Qwen3-1.7B \
  --max_samples 100 --device mps

# BagMerge (prior-work baseline): concat + RoPE re-encode
python run.py --method latent_mas_crdt --num_thinkers 2 \
  --task partitioned --partitioned_view split \
  --merge_strategy concat_rerope --diversity_mode role \
  --model_name Qwen/Qwen3-1.7B --max_samples 100 \
```

family	type	C_0	PAR3	MIX	SEQ3	TREEMERGE
NEONYM	chain	1.00 ± 0.00	0.07 ± 0.02	0.73 ± 0.03	0.97 ± 0.01	0.06 ± 0.02
RELAY	chain	1.00 ± 0.00	0.06 ± 0.04	0.41 ± 0.07	0.32 ± 0.03	0.06 ± 0.04
CONSTRAINT	symm.	0.97 ± 0.03	0.03 ± 0.01	0.03 ± 0.01	0.17 ± 0.06	0.01 ± 0.01
RECIPE	symm.	1.00 ± 0.00	0.73 ± 0.03	0.96 ± 0.02	0.99 ± 0.01	0.71 ± 0.02
DISCOUNT	symm.	0.99 ± 0.02	0.23 ± 0.11	0.41 ± 0.09	0.49 ± 0.06	0.11 ± 0.06

Table 15: Full five-family $k = 3$ topology probe (across-draw mean $\pm$ sd, 3 draws, greedy, $n = 50$). TREEMERGE never exceeds PAR3 on any of the five families (colocation is not composition): it matches PAR3 on the chains and on RECIPE, and on DISCOUNT falls below it (0.11 vs. 0.23), so the pairwise merge tree can even lose relative to the flat merge. The two chain families collapse under PAR3 and recover under reprocessing; the symmetric families span a composition-limited pole (CONSTRAINT, where no topology helps), through RECIPE (whose 3-way merge already mostly works), to a gradient (DISCOUNT). RELAY's SEQ3 $<$ MIX is an ordering artifact: its default chain places the non-commutative pipeline's input last.

```
  --max_new_tokens 1024 --latent_steps 40 --prompt sequential \
  --device mps --device2 mps --generate_bs 1

# CanonicalMerge (ours), k = 2
python run.py --method latent_mas_crdt --num_thinkers 2 \
  --task partitioned --partitioned_view split \
  --merge_strategy novelty_concat_sym_layout \
  --novelty_threshold 2.0 --novelty_layer 15 \
  --diversity_mode role --model_name Qwen/Qwen3-1.7B \
  --max_samples 100 --max_new_tokens 1024 --latent_steps 40 \
  --prompt sequential --device mps --device2 mps --generate_bs 1

# CanonicalMerge (ours), k = 3 (partitioned_k3 dataset)
python run.py --method latent_mas_crdt --num_thinkers 3 \
  --task partitioned_k3 --partitioned_view split \
  --merge_strategy novelty_concat_sym_layout_n \
  --novelty_threshold 2.0 --novelty_layer 15 \
  --diversity_mode role --model_name Qwen/Qwen3-1.7B \
  --max_samples 100 --max_new_tokens 1024 --latent_steps 40 \
  --prompt sequential --device mps --device2 mps --generate_bs 1
```

The query-blind / federated arm adds `--query_judger_only`; the order-sensitivity foil for BagMerge adds `--swap_partitioned_thinkers`; the second model scale sets `--model_name Qwen/Qwen3-4B` with `--novelty_layer 24`; greedy decoding is `--temperature 0`.

Versions. Qwen3-1.7B (and Qwen3-4B) in `bfloat16` on Apple MPS; MacBook Pro M3, 24 GB unified memory; PyTorch 2.11.0, Transformers 4.55.4. Seed 42 throughout unless a data seed is stated.

Note on the legacy CLI flag. The implementation still exposes the historical strategy name `novelty_concat_sym_layout(_n)` and a `--novelty_threshold` argument. All reported runs set this legacy threshold to 2.0, which makes the strategy exactly the canonical-ordered concatenation of Section 4.3; the threshold is not part of the method and is disclosed only so the CLI invocations above are self-explanatory.