CORRECTING ERRORS IS NOT ENOUGH: EXACT AUDITS OF ANSWER PRESERVATION ON SHARED TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory-augmented large language models (LLMs) need to use helpful memory while preserving correct answers when memory is unreliable. We call the latter ability correct-answer preservation. Comparing this ability across model configurations is difficult because different configurations may answer different tasks correctly without memory, while missing responses and incomplete task alignment create further uncertainty. We propose a shared-task audit that compares corruption rates on tasks that every configuration in a declared group answers correctly at baseline. When task alignment or outcomes are incomplete, we derive sharp finite-set bounds over all task assignments and missing-result completions consistent with the retained summaries. We also prove an exact solver that needs to enumerate missing-correct counts only for the two target configurations while preserving the same outer bounds as full search. The framework further identifies when additional overlap information is sufficient to resolve an otherwise uncertain comparison and when task-level labels are required. On reported MEMIMPACTBENCH counts, pair-specific shared-task support determines 14 of 15 harmful-memory directions, while one all-six shared set determines eight misleading-memory and seven stale-memory directions. We also validate the audit on public row-level StrategyQA conflict records spanning four model configurations and 12 incorrect-prior conditions: all 144 count-based intervals contain the matched-task truth, and all 43 strict direction certificates match the directly counted sign. These results show that rates computed on different task subsets do not necessarily support the same cross-model ordering. The shared-task audit states which preservation differences are supported by the available evidence and which require additional information.

1 INTRODUCTION

Reliable memory has two jobs. It should help when the model is wrong. It should also keep correct answers correct. Many systems store or retrieve old text in different ways (Park et al., 2023; Packer et al., 2023; Wang et al., 2023; Zhong et al., 2024). Memory benchmarks then test whether agents can find facts, use earlier examples, and update old information (Wu et al., 2025; Hu et al., 2026; Tan et al., 2025). We study the second job. We call wrong-to-correct a *rescue*. We call correct-to-wrong a *corruption*. We call correct-to-correct *preservation*. Here, memory means external text added to a task.

A problem appears when we compare configurations. A corruption rate is usually computed only on tasks that one configuration answered correctly without memory. Different configurations can therefore use different task sets and different denominators. Their own-task rates can both be correct, but the difference may not tell us which configuration corrupts more answers on the same tasks. Earlier work already shows that conflicting context can both fix and break answers (Wu et al., 2024).

MEMIMPACTBENCH gives a clear example. We use this name for the nine-configuration memory-intervention study analyzed here. Under misleading memory, Meta has 179 corruptions among 349 correct baselines (51.29%). DeepSeek has 205 among 325 (63.08%). The own-task rates suggest more corruption for DeepSeek. But the same totals allow a 290-task shared set with differences of

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

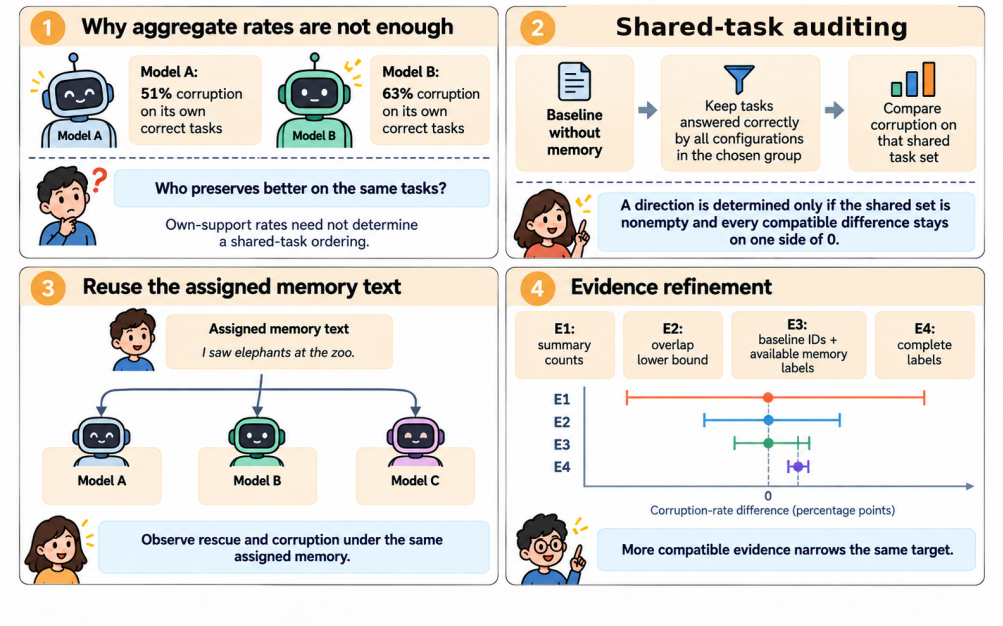


Figure 1: **Preservation auditing at a glance.** All four panels are illustrative. They show the comparison problem, shared-task auditing, reuse of the assigned external memory text, and evidence refinement. The zoo note and interval locations are not measured results, and the final point in panel 4 need not be zero.

either -3.10 or $+29.31$ percentage points. Both cases fit the reported summaries. The missing part is which baseline-correct tasks overlap and where the corruptions occur.

Simple fixes do not solve this problem. Own-task rates still use different task sets. Complete-case analysis changes the target to an observed subset. An interval midpoint gives one sign, but that sign is not guaranteed. Full enumeration is exact, but its search grows with missing counts from every configuration. We need an audit that keeps one shared-task target, gives a sign only when the data prove it, and stays practical for larger groups.

We propose an exact finite-set audit. It compares tasks that every configuration in a chosen group answered correctly at baseline. With missing task matches or outcomes, it returns the sharp interval of all corruption-rate differences that fit the retained data. We also prove that the exact search needs missing-correct counts only from the two target configurations. The outer bounds stay the same as in full search. If a comparison is unresolved, the audit asks what extra evidence is needed. A checked overlap is enough in some cases; other cases need task IDs and matched memory outcomes. Figure 1 shows the workflow. With full task-level labels, direct counting remains the final answer.

We test the audit in four ways. Reported MEMIMPACTBENCH counts show what summary data alone can prove. Public StrategyQA records test the bounds on real matched tasks. Provider-free ablations show which extra records matter. Stress and scaling tests show when the audit becomes less informative and how much computation the exact search saves.

Our contributions are:

- **Sharp shared-task bounds.** We define one shared-task target and exact bounds for missing task matches or answers, including an empty shared set.
- **Exact two-target search.** We prove that non-target missing-correct counts can stay at their smallest feasible values without changing the outer endpoints, giving $O(J + (u_A + 1)(u_B + 1))$ arithmetic states.
- **Evidence needed for certification.** We show when an overlap bound proves a direction and when task IDs and matched outcomes are needed. Public row-level data check these certificates.

The basic set-overlap bound and partial identification are not new. Our contribution is the shared-task target, the sharp bounds for this target, the exact two-target search, and the evidence query (McNemar, 1947; Manski, 1990; Tamer, 2010).

2 Related Work

Memory systems and agent memory. Many recent systems add an explicit memory layer around an LLM. Generative Agents stores experiences, writes reflections, and retrieves past events for later actions (Park et al., 2023). MemoryBank keeps long-term conversational memory and updates it over time (Zhong et al., 2024). LongMem adds a retriever and reader for long histories (Wang et al., 2023), while MemGPT manages several memory tiers to extend usable context (Packer et al., 2023). A recent survey groups this area by memory source, form, operation, and evaluation (Zhang et al., 2025). These works mainly ask how to build useful memory. Our paper asks how to compare preservation once different configurations have different baseline-correct task sets.

Memory and long-context evaluation. LongMemEval studies long-term memory in chat assistants (Wu et al., 2025). MemoryAgentBench tests retrieval, test-time learning, long-range understanding, and selective forgetting (Hu et al., 2026). MemBench also evaluates several parts of agent memory (Tan et al., 2025). Long-context benchmarks such as LongBench and RULER test whether models can use information across long inputs (Bai et al., 2024; Hsieh et al., 2024). Lost in the Middle shows that using long context can depend strongly on where useful information appears (Liu et al., 2024). Recent memory work also studies when stored information should be forgotten and how longer retention can create new risks (Pulipaka et al., 2026; Al-Tawaha et al., 2026). These benchmarks are related because they test information use, but they do not solve our cross-configuration denominator problem.

Retrieval, non-parametric memory, and harmful context. Retrieval-augmented models use external text as non-parametric knowledge. REALM learns a neural retriever during language-model pretraining (Guu et al., 2020), and RAG combines retrieval with generation for knowledge-intensive tasks (Lewis et al., 2020). Mallen et al. study when parametric knowledge is weak and retrieval is useful (Mallen et al., 2023). Retrieval can also hurt when added context is irrelevant; Yoran et al. analyze and reduce this failure mode (Yoran et al., 2024). Our setting is different: we do not propose a retrieval method. We audit whether reported rates support a same-task preservation comparison.

Conflicting and outdated context. Knowledge conflicts between model parameters and supplied context have been studied directly. Longpre et al. formalize entity-level conflicts in question answering (Longpre et al., 2021). ClashEval measures whether incorrect evidence changes an initially correct answer (Wu et al., 2024). HoH studies outdated information in RAG (Ouyang et al., 2025), while Astute RAG studies imperfect retrieval and knowledge conflict (Wang et al., 2025). DRUID and CUB also show that context quality and context use matter in retrieval-augmented evaluation (Hagström et al., 2025; 2026). Bajaj and Tiganj release aligned no-prior and prior-conditioned StrategyQA rows (Bajaj & Tiganj, 2026); we use them only for external validation of our audit, not as a memory benchmark.

Paired counts and partial identification. Rescue and corruption counts form a paired binary table (McNemar, 1947). When information is incomplete, partial identification asks for all values that remain compatible with the evidence instead of forcing a point estimate (Manski, 1990; Tamer, 2010). Related work studies bounds under contaminated or corrupted data (Horowitz & Manski, 1995), confidence regions for partially identified parameters (Imbens & Manski, 2004), and statistical matching when variables are not observed jointly (Ahfock et al., 2016). Sharp identified sets can also be written through linear systems or generic optimization (Balke & Pearl, 1997; Torgovitsky, 2019; Voronin, 2025). The multiway overlap inequality itself is standard. Our new result is its use inside the shared-task corruption contrast, together with monotone outer endpoints, an exact two-target reduction, and an evidence query. The intervals in this paper are finite-set logical bounds, not population confidence intervals.

Table 2: **Four paired outcomes.** The rows describe recorded correctness changes, not a causal effect with a shared hidden model draw.

Baseline	With memory	Outcome	Meaning
Incorrect	Incorrect	Still incorrect	No correct answer is produced.
Incorrect	Correct	Rescue	A baseline error is corrected.
Correct	Incorrect	Corruption	A baseline-correct answer is lost.
Correct	Correct	Preserved correct	The answer stays correct.

Table 1: **What the comparison methods answer.** "Exact" means a guaranteed interval or sign for the stated target under the retained information, not a population-confidence statement.

Method	Target	Missing outcomes	Exact sign	Task IDs required
Own-support difference	Each model's own correct tasks	Descriptive	No	No
Complete-case difference	Observed shared subset	Dropped	No	Yes
Interval midpoint	Full shared target	Allowed	No	No
Full enumeration / exact reference	Full shared target	Allowed	Yes	No
Generic MILP scan	Full shared target	Allowed	Yes	No
Two-target audit	Full shared target	Allowed	**Yes**	No

3 Shared-Task Preservation Audit

3.1 Setting and answer changes

Fix K task positions for one memory condition. A configuration includes the model, prompts, decoding controls, and service settings used for the recorded run. Memory is external text added to the task input. Valid memory supports the target answer; irrelevant memory does not resolve it; misleading memory supports a wrong answer; stale memory is out of date. These are separate conditions, not points on one severity scale.

For a scored baseline–memory pair, let $b_i, m_i \in \{0, 1\}$ denote correctness. Let $R_c = n_{01}$ be rescues, $C_c = n_{10}$ corruptions, $E = n_{00} + n_{01}$ baseline errors, $S = n_{10} + n_{11}$ correct baselines, and $N = E + S$. We report

$$R = R_c/N, \qquad C = C_c/N, \qquad r = R_c/E, \qquad h = C_c/S. \tag{1}$$

Preservation is $1 - h$. A conditional rate is NA when its opportunity denominator is zero. Net accuracy change is

$$\Delta = R - C = (1 - p)r - ph, \qquad p = S/N, \tag{2}$$

when both conditional rates are defined; otherwise the count form $(R_c - C_c)/N$ is used. Table 2 and Appendix A give the full decomposition.

3.2 The shared comparison target

Let B_j be configuration j's baseline-correct set and $H_j \subseteq B_j$ its corrupted set under the fixed memory condition. For a declared group G, define

$$I_G = \bigcap_{j \in G} B_j, \qquad \theta_{B-A|G} = \frac{|H_B \cap I_G| - |H_A \cap I_G|}{|I_G|}. \tag{3}$$

The contrast is defined only when I_G is nonempty. With $G = \{A, B\}$, each pair has its own shared set. With a larger fixed group, all pairs use one common set. Changing G therefore changes the quantity being measured; pair-specific directions are not automatically one leaderboard.

3.3 SHARP BOUNDS FROM COUNTS

Assume first that only completed correct-baseline counts $\tilde{s}_j = |B_j|$ and corruption counts $\tilde{c}_j = |H_j|$ are known. If $J = |G|$ and $k = |I_G|$, then

$$k_{\min} = \max\left(0, \sum_{j \in G} \tilde{s}_j - (J-1)K\right), \qquad k_{\max} = \min_{j \in G} \tilde{s}_j. \tag{4}$$

For fixed $k > 0$, configuration j has between

$$\ell_j(k) = \max(0, \tilde{c}_j - \tilde{s}_j + k), \qquad v_j(k) = \min(\tilde{c}_j, k) \tag{5}$$

corruptions inside the shared set. Hence

$$L(k) = \frac{\ell_B(k) - v_A(k)}{k}, \qquad U(k) = \frac{v_B(k) - \ell_A(k)}{k}. \tag{6}$$

Proposition 1 (Sharp count bounds)**.** *Under the retained margins, every feasible positive shared-set size between $k_{\min}$ and $k_{\max}$ can occur. At each size, both endpoints in equation 6 are attainable. Also, $L(k)$ cannot decrease and $U(k)$ cannot increase. The outer interval is therefore reached at the smallest feasible positive support.*

Appendix B proves feasibility, attainability, and monotonicity. If $k_{\min} = 0$, an empty set is possible and is flagged separately.

Now let (N_j, s_j, c_j) be observed pair, correct-baseline, and corruption counts, with $u_j = K - N_j$ unavailable pairs. A completion adds x_j correct baselines and y_j corruptions subject to $0 \le y_j \le x_j \le u_j$. These values describe possible missing labels, not new observations.

3.4 EXACT TWO-TARGET REDUCTION

Suppose a checked overlap lower bound $|I_G| \ge q$ is also supplied and let $r_0 = \max(1, q)$. For every non-target configuration, define

$$x_j^{\min} = \max(0, r_0 - s_j), \qquad j \notin \{A, B\}. \tag{7}$$

The reduced solver enumerates only x_A and x_B. For each target state, it uses the smallest feasible positive support implied by the completed sizes and evaluates the corresponding outer endpoints.

Theorem 1 (Two-target exactness)**.** *Fix x_A and x_B. Set every non-target missing-correct count to the minimum in equation 7. This cannot remove an outer endpoint. The reduced solver therefore returns the same lower and upper bounds as full enumeration.*

Intuition. Proposition 1 shows that the widest compatible interval occurs at the smallest feasible positive shared support. Increasing a non-target missing-correct count can only keep that minimum support unchanged or push it upward, so it cannot create a more extreme endpoint. This is why non-target counts can be fixed at their feasible minima without approximation.

The proof is in Appendix C. After $O(J)$ precomputation, the reduced query uses $O((u_A+1)(u_B+1))$ target states. This is an arithmetic-state result, not a hardware-independent runtime claim.

Generic exact optimization baseline. The same aggregate feasibility problem can also be solved without the two-target reduction. For a fixed positive shared-set size k, an integer program chooses missing-correct counts and the two targets' shared corruption counts subject to the count and overlap constraints. Scanning feasible k values and minimizing or maximizing the signed shared corruption count gives the exact interval. We use this generic MILP scan only as a computational baseline; Appendix K gives its implementation check and runtime comparison.

3.5 ADDING EVIDENCE WITHOUT CHANGING THE TARGET

A larger verified lower bound on $|I_G|$ removes feasible assignments, so the lower endpoint cannot decrease and the upper endpoint cannot increase. We can therefore ask for the smallest checked

overlap that proves a direction or a chosen minimum gap. If no feasible overlap size is enough, task identities or answer labels are required.

Matched task labels provide more detail than one overlap count. Known baseline labels can force tasks in or out of I_G, while known memory labels restrict their signed contribution. Appendix D gives exact task-aligned bounds. Complete labels reduce the calculation to direct counting. The target group and task universe stay fixed as evidence is added.

4 EXPERIMENTS

4.1 SETUP

The experiments ask four questions: what the reported MEMIMPACTBENCH counts imply, how the declared shared task set changes a comparison, whether count-only bounds agree with truth on independent row-level records, and how added evidence narrows the same target. MEMIMPACTBENCH reports nine completed configurations. The shared-task audit uses Claude, Kimi, Qwen, Meta, DeepSeek, and Grok because harmful-memory margins are available for this six-member group. The full nine-configuration overview, execution counts, and Representative results are retained in Appendices E, H, and I.

The MEMIMPACTBENCH case is intentionally a count-level certification exercise: it asks what the reported summaries alone are sufficient to prove and does not estimate unseen row-level ordering. For an independent matched-task check, we use the public StrategyQA portion of Experiment 1 from Bajaj & Tiganj (2026). It evaluates the same 300 items across four deterministic model configurations with a no-prior baseline and social-prior conditions. We use the 12 combinations of source $\in \{\text{friends, experts, LLMs}\}$ and group size $k \in \{1, 3, 5, 9\}$, retaining only items whose supplied prior is incorrect. Each harmful-condition universe contains 141–161 aligned items across all four models. Appendix J gives the data provenance and exact replay procedure. A separate synthetic study contains 160 task sets and 2,400 comparisons per scope; Section 4.7 summarizes its evidence stages.

4.2 REPORTED DATA REVEAL THE SHARED-TASK PROBLEM

The reported harmful-memory data first show the problem studied in this paper. Table 3 puts the key evidence in one place: the original own-task counts, three shared-task examples, and the result over all 15 pairs.

Panel (a) shows that the baseline-correct denominator is not the same across configurations. Panel (b) shows the effect: one direction survives both scopes, one survives only pair-specific support, and one is unresolved. Panel (c) shows that the pattern is broader than one pair. Valid memory has the same denominator issue; Appendix E gives the full transition margins.

4.3 SHARED-TASK COMPARISONS

Figure 2 shows the three cases from Table 3(b). The evidence is the same, but the declared task group changes the target. This can preserve a direction, remove a direction, or leave both scopes unresolved.

4.4 EXTERNAL ROW-LEVEL VALIDATION

The public StrategyQA records let us compare count-only conclusions with the matched-task truth rather than with a synthetic generator. For each of the 12 incorrect-prior conditions, we join the four models to the no-prior baseline by item ID. We then compute the exact pair-specific and all-four shared-task differences from the rows, discard the IDs, retain only each model's baseline-correct and corruption counts, and rerun the count audit.

All 144 count intervals contain the matched-task truth, and all 43 strict certificates match the directly counted sign. The shortcuts are not equivalent: across the 133 nonzero truths, the own-support sign is opposite to the corresponding shared-task sign in 14 cases, while the count-interval midpoint is opposite in 10. Restoring the public task IDs and post-intervention labels reduces each interval to the directly counted finite-set value. This experiment is not a memory benchmark and does not validate

Table 3: **The reported data reveal the shared-task problem.** The first panel shows the original harmful-memory margins. The lower panels show how the same data change when we require the same baseline-correct tasks. Bounds are exact logical intervals, not confidence intervals.

	Misleading memory			Stale memory		
Configuration	S	C_c	h (%)	S	C_c	h (%)
Claude	383	22	5.74	383	23	6.01
Kimi	384	78	20.31	384	87	22.66
Qwen	350	104	29.71	350	114	32.57
Meta	349	179	51.29	349	180	51.58
DeepSeek	325	205	63.08	325	199	61.23
Grok	384	270	70.31	384	270	70.31

(b) Selected misleading-memory contrasts

Comparison	Own diff.	Pair bound	All-six bound
DeepSeek − Kimi	+42.77	[39.08, 57.23]	[22.35, 80.39]
Kimi − Claude	+14.57	[14.32, 14.62]	[−8.98, 30.59]
DeepSeek − Meta	+11.79	[−3.10, 29.31]	[−17.25, 47.06]

(c) Directions determined

Target	Mis.	Stale
Own-task rate	15/15	15/15
Own-task bounds	15/15	15/15
Pair shared set	14/15	14/15
All-six shared set	8/15	7/15

S is the number of correct baselines, C_c is the number of corruptions, and $h = C_c/S$. Panel (b) uses first minus second in percentage points.

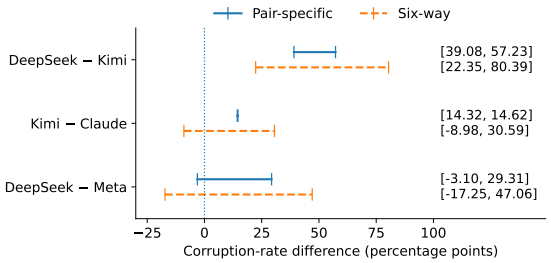


Figure 2: **The task group changes the result.** The same misleading-memory counts can keep a direction, remove it, or leave it unresolved. Lines are exact logical bounds.

Table 4: **Analysis baselines.** "Issued" is the share of nonzero-truth cases with a nonzero sign. "Opp." is the share with the opposite sign. The first two rows change the target; midpoint has no sign guarantee.

Method	Issued	Opp.	Guarantee
Own-support	100.00	6.09	No
Complete-case	97.67	4.99	No
Midpoint	33.19	2.05	No
Full exact ref.	0.97	0.00	Exact
Two-target	0.97	**0.00**	**Exact**

Percentages are from 4,727 nonzero-truth six-way stress contrasts. Exact rows abstain when the retained counts do not prove a sign.

MEMIMPACTBENCH task construction; it independently validates the audit's central measurement claim on real, aligned model outputs. Detailed results and source hashes appear in Appendix J.

4.5 ANALYSIS BASELINES

Table 4 gives a compact comparison on 4,727 nonzero-truth six-way stress contrasts. Own-support and complete-case points change the target. The midpoint keeps the full shared target but has no sign guarantee. Exact methods abstain when the retained counts do not prove a direction. The generic MILP uses the same exact target and is benchmarked against the proposed solver in Appendix K.

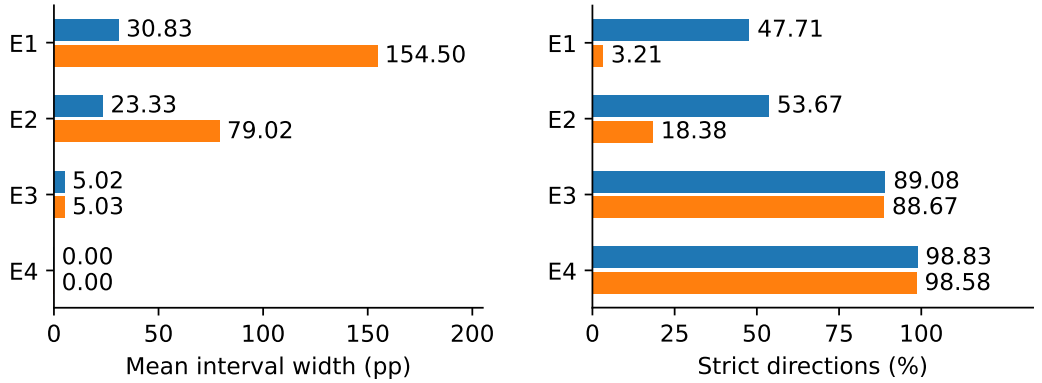
The 0.97% issuance rate is therefore a coverage result, not an accuracy failure: under this deliberately difficult count-only stress suite, the retained evidence rarely determines a six-way sign. Whenever the exact audit does issue a sign, no wrong certificate appears in these cases.

4.6 WHAT EXTRA EVIDENCE IS ENOUGH?

The reported counts allow an all-six overlap of 255–325 correct-baseline tasks. A verified lower bound of 325 would settle 11 misleading-memory and 13 stale-memory directions. For Kimi minus Claude under stale memory, 322 shared tasks are enough; under misleading memory, no feasible overlap count up to 325 settles the pair. The latter case needs task membership or matched answer

Table 5: **Public row-level validation on StrategyQA conflict records.** Each scope has six model pairs across 12 incorrect-prior conditions. "Opposite" counts are restricted to nonzero matched-task truths. Widths are percentage points.

Scope	N	Contained	Cert.	Wrong	Own opp.	Mid. opp.	Width
Pair-specific	72	72/72	35/72	0/35	8/66	3/66	13.36
All-four	72	72/72	8/72	0/8	6/67	7/67	29.94



(a) Mean interval width

(b) Determined nonzero directions

Figure 3: **Added evidence narrows the same synthetic target.** E1 is counts, E2 adds overlap, E3 adds baseline IDs and visible labels, and E4 uses all labels. Each scope has 2,400 comparisons.

labels. Appendix G reports all thresholds. This is the practical distinction between asking for *more overlap information* and asking for *which tasks and answers are shared*.

4.7 Synthetic information recovery and ablations

The reported synthetic study starts with complete labels and a known target, then hides information. E1 keeps counts, E2 adds a verified overlap lower bound, E3 restores baseline IDs plus available memory labels, and E4 restores all labels. The target task set stays fixed across stages.

For six-way support, E1 determines 3.21% of directions with mean width 154.50 points. E2 raises the direction rate to 18.38% and reduces width to 79.02. E3 gives 88.67% and 5.03 points. E4 gives zero width; the remaining non-strict cases are exact ties. The same sequence gives 47.71%, 53.67%, 89.08%, and 98.83% determined pair-specific directions.

Which part of E3 matters? We add a separate provider-free ablation on 1,200 six-way contrasts. The exact shared-set size is fixed first. Under that condition, baseline IDs alone only name which tasks belong to the shared set; without target memory outcomes, they still do not tell us where corruptions fall. Conversely, target memory labels without baseline membership do not identify which labelled outcomes belong to the shared set. Table 6 therefore isolates the value of linking these two evidence types. The large gain appears only when shared-task membership and target memory outcomes are aligned.

Stress and exact-solver checks. A provider-free extension varies hiding over 0–20% and group size over 2, 4, 6, and 9. Certificates remain sign-correct but become less frequent as information is removed or the common set becomes more selective. Appendix K gives all widths and settings. The same appendix compares full enumeration, a generic MILP exact baseline, and the proposed reduction. At $J = 6, u = 5$, the three runtimes are 1.215 s, 0.381 s, and 0.00064 s; at $J = 12, u = 20$, full enumeration has 7.36×10^{15} states, while MILP and the reduced solver take 0.439 s and 0.00760 s. MILP and the reduced solver agree on all 30 scaling workloads, all 20 timed full-enumeration cases agree, and 1,000 additional random finite checks match. The theorem establishes exactness; these tests check implementations and practical scaling.

Table 6: **Evidence-type ablation with exact shared-set size fixed.** These are new provider-free synthetic diagnostics. Width is the mean exact interval width over 1,200 contrasts. A strict direction excludes zero.

Evidence	Mean width (pp)	Determined (%)	Wrong certificates
Exact overlap size only	71.74	22.67	0
+ baseline IDs only	71.74	22.67	0
+ target memory labels only	59.64	22.92	0
+ both	0.00	98.75	0

4.8 SCOPE

The MEMIMPACTBENCH case certifies only what follows from its reported task universe, labels, and configuration links; it does not reconstruct unseen row-level ordering. The independent StrategyQA rows provide the complementary matched-task check. Shared-task support can become selective as a group grows, so shared-task results should accompany overall accuracy, coverage, and own-task rates. Recorded baseline-to-memory transitions are descriptive, not single-sample causal effects; repeated no-memory draws would be needed to separate ordinary generation variability. We compare recorded configurations rather than model weights alone, and service aliases may not reproduce immutable checkpoints. Exact replay therefore needs archived outputs or pinned snapshots plus prompt, decoding, endpoint, and service metadata.

5 CONCLUSION

Two valid corruption rates can still compare different tasks. We define the shared-task target, derive sharp finite-set bounds, and prove an exact two-target reduction. In MEMIMPACTBENCH, most pair-specific harmful-memory directions remain certifiable from summaries, while the stricter all-six target is less often determined. On public StrategyQA rows, all 144 count intervals contain the matched-task truth and all 43 strict certificates match the directly counted sign; own-support signs disagree with shared-task truth in 14 nonzero cases. The resulting lesson is simple: state the comparison set, report the full compatible interval, and request the specific evidence needed to resolve uncertainty rather than forcing a ranking.

REFERENCES

Daniel Ahfock, Saumyadipta Pyne, Sharon X. Lee, and Geoffrey J. McLachlan. Partial identification in the statistical matching problem. *Computational Statistics & Data Analysis*, 104:79–90, 2016.

Ahmad Al-Tawaha, Shangding Gu, Peizhi Niu, Ruoxi Jia, and Ming Jin. Remembering more, risking more: Longitudinal safety risks in memory-equipped llm agents. *arXiv preprint arXiv:2605.17830*, 2026.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 3119–3137, 2024. doi: 10.18653/v1/2024.acl-long.172.

Anooshka Bajaj and Zoran Tiganj. Who do llms trust? human experts matter more than other llms. *arXiv preprint arXiv:2602.13568*, 2026.

Alexander Balke and Judea Pearl. Bounds on treatment effects from studies with imperfect compliance. *Journal of the American Statistical Association*, 92(439):1171–1176, 1997.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3929–3938, 2020.

Lovisa Hagström, Sara Vera Marjanovic, Haeun Yu, Arnav Arora, Christina Lioma, Maria Maistro, Pepa Atanasova, and Isabelle Augenstein. A reality check on context utilisation for retrieval-augmented generation. In *Proceedings of ACL*, pp. 19691–19730, 2025.

Lovisa Hagström, Youna Kim, Haeun Yu, Sang-goo Lee, Richard Johansson, Hyunsoo Cho, and Isabelle Augenstein. Cub: Benchmarking context utilisation techniques for language models. In *Proceedings of ACL*, pp. 25101–25133, 2026.

Joel L. Horowitz and Charles F. Manski. Identification and robustness with contaminated and corrupted data. *Econometrica*, 63(2):281–302, 1995.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in llm agents via incremental multi-turn interactions. In *International Conference on Learning Representations*, 2026.

Guido W. Imbens and Charles F. Manski. Confidence intervals for partially identified parameters. *Econometrica*, 72(6):1845–1857, 2004. doi: 10.1111/j.1468-0262.2004.00555.x.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen-tau Yih, Tim Rocktaschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, 2020.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. Entity-based knowledge conflicts in question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7052–7063, 2021. doi: 10.18653/v1/2021.emnlp-main.565.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pp. 9802–9822, 2023. doi: 10.18653/v1/2023.acl-long.546.

Charles F. Manski. Nonparametric bounds on treatment effects. *American Economic Review*, 80(2): 319–323, 1990.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12:153–157, 1947. doi: 10.1007/BF02295996.

Jie Ouyang, Tingyue Pan, Mingyue Cheng, Ruiran Yan, Yucong Luo, Jiaying Lin, and Qi Liu. Hoh: A dynamic benchmark for evaluating the impact of outdated information on retrieval-augmented generation. In *Proceedings of ACL*, pp. 6036–6063, 2025.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–22, 2023. doi: 10.1145/3586183.3606763.

Sidharth Pulipaka, Oliver Chen, Manas Sharma, Taaha S. Bajwa, Vyas Raina, and Ivaxi Sheth. Persistbench: When should long-term memories be forgotten by llms? *arXiv preprint arXiv:2602.01146*, 2026.

Elie Tamer. Partial identification in econometrics. *Annual Review of Economics*, 2:167–195, 2010.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. Membench: Towards more comprehensive evaluation on the memory of llm-based agents. *arXiv preprint arXiv:2506.21605*, 2025.

Alexander Torgovitsky. Partial identification by extending subdistributions. *Quantitative Economics*, 10(1):105–144, 2019.

Andrei Voronin. Linear programming approach to partially identified econometric models. *arXiv preprint arXiv:2503.14940*, 2025.

Fei Wang, Xingchen Wan, Ruoxi Sun, Jiefeng Chen, and Sercan O. Arik. Astute rag: Overcoming imperfect retrieval augmentation and knowledge conflicts for large language models. In *Proceedings of ACL*, pp. 30553–30571, 2025.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. In *Advances in Neural Information Processing Systems*, 2023.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*, 2025. arXiv:2410.10813.

Kevin Wu, Eric Wu, and James Zou. Clasheval: Quantifying the tug-of-war between an llm's internal prior and external evidence. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. Making retrieval-augmented language models robust to irrelevant context. In *International Conference on Learning Representations*, 2024.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47, 2025. doi: 10.1145/3748302.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19724–19731, 2024. doi: 10.1609/aaai.v38i17.29946.

A Paired Answers and Net Change

A.1 Relations between the counts

Consider one scored pair set with the four counts $n_{00}, n_{01}, n_{10}, n_{11}$ from Table 2. There are $S = n_{10} + n_{11}$ correct baselines and $E = n_{00} + n_{01}$ baseline errors, with $N = S + E$. Baseline accuracy is $(n_{10} + n_{11})/N$; memory accuracy is $(n_{01} + n_{11})/N$. Their difference is

$$\Delta = \frac{n_{01} - n_{10}}{N} = \frac{R_c - C_c}{N} = R - C. \tag{A1}$$

When both baseline opportunity counts are positive, $R = (1 - p)r$ and $C = ph$, giving equation 2. If $E = 0$, the rescue count and all-pair rescue frequency are zero, but the conditional correction rate is undefined. NA records that lack of opportunity; neither zero nor 100% would describe it correctly. The same rule applies to conditional corruption when $S = 0$.

A.2 What two accuracy values do not tell us

Let p_b and p_m be baseline accuracy and memory accuracy on the *same* observed pair set. Let $z = n_{11}/N$. The share correct in both answers satisfies $\max(0, p_b + p_m - 1) \le z \le \min(p_b, p_m)$. Since $C = p_b - z$,

$$\max(0, p_b - p_m) \le C \le \min(p_b, 1 - p_m). \tag{A2}$$

Two accuracy values may leave the number of lost correct answers unknown. Paired records identify that count. This is a standard binary-table bound, not a claim that every real value in the interval occurs in the finite data. The symbol p_m denotes memory accuracy and is distinct from the overlap bound q.

A.3 Changes that cancel in the net result

The share of pairs whose correctness changes is $T = R + C$, also called turnover. Reporting only the absolute net change removes the following amount through cancellation:

$$T - |\Delta| = (R + C) - |R - C| = 2\min(R, C). \tag{A3}$$

For the four-model core under misleading memory, $(R_c, C_c, N) = (5, 654, 1536)$. Turnover is 42.90%, the absolute net change is 42.25 percentage points, and cancellation is 0.65 points. The five rescues under stale memory also give 0.65 points of cancellation. The net accuracy loss is therefore already visible in both conditions.

DeepSeek's irrelevant-memory condition has ten rescues and 83 corruptions among 384 pairs. These give 24.22% turnover, a -19.01-point net change and 5.21 points of cancellation. The paired counts explain both directions of change. In the reference core, the absolute misleading-memory loss is 8.11 times the valid-memory gain. This ratio is descriptive, not a general risk–benefit rule. The baseline opportunities differ, and the frequency of each memory condition in real use is unspecified.

B Exact Bounds with Complete Counts

B.1 Possible sizes of the shared task set

Fix $K \ge 1$ and a group of $J \ge 2$ configurations. The available complete counts are $\tilde{s}_j = |B_j|$ and $\tilde{c}_j = |H_j|$, where $H_j \subseteq B_j$. No task IDs or extra links between corruption sets are retained. No sampling model is assumed. Let D_j be the complement of B_j. Then

$$\left| \bigcap_j B_j \right| = K - \left| \bigcup_j D_j \right|. \tag{B1}$$

The union is at least as large as the largest complement and no larger than either K or the sum of complement sizes. These limits give equation 4.

Every integer between these limits is feasible. Choose a shared set of size k and let $L = K - k$ positions remain outside it. Each complement has size $d_j = K - \tilde{s}_j \le L$, while their sizes sum to at

least L. Arrange the complements as consecutive cyclic blocks over those L positions. Together they cover every outside position without entering the chosen shared set. The baseline intersection then has exactly k members. If $L = 0$, all baseline sets already equal the full universe.

B.2 Possible corruption counts inside the set

At a fixed $k > 0$, configuration j has $\tilde{s}_j - k$ correct baselines outside the shared set. Its shared corruptions therefore number at least $\max(0, \tilde{c}_j - \tilde{s}_j + k)$ and at most $\min(\tilde{c}_j, k)$. Every integer between these limits is attainable. Put the chosen number inside the shared set and the rest on the configuration's correct baselines outside it.

The model imposes no further links between configurations' corruption sets. Hence B can attain its minimum shared corruption count while A attains its maximum, and conversely. Other corruption sets can be filled separately. Both endpoints of equation 6 are attainable, so the bounds are sharp for the stated information. Additional matched labels may exclude some endpoint assignments.

B.3 Why the smallest positive set gives the widest range

For $d = \tilde{s}_B - \tilde{c}_B \geq 0$, write the lower bound as

$$L(k) = \max\left(0, 1 - \frac{d}{k}\right) - \min\left(\frac{\tilde{c}_A}{k}, 1\right). \tag{B2}$$

For positive k, both d/k and $\tilde{c}_A/k$ decrease as k grows. Thus the first term cannot fall, and the negative of the second term cannot fall either. It follows that $L(k)$ is nondecreasing. Swapping targets and reversing the sign shows that $U(k)$ is nonincreasing. The smallest feasible positive size therefore gives the outer bounds.

If an empty set is possible, the interval is conditional on positive support; it does not establish that every feasible comparison is defined. If no positive size is possible, the target is undefined. Feasible ratios in a finite task set may also have gaps. The reported bounds form the smallest interval containing them, not a claim that every real value in that interval is attainable.

C Missing Answers and the Exact Two-Target Search

C.1 Possible missing labels and the full search

For each configuration, retain (N_j, s_j, c_j) and $u_j = K - N_j$. A missing pair may have an incorrect baseline, a preserved correct baseline or a corrupted correct baseline. These cases give $0 \leq y_j \leq x_j \leq u_j$. The aggregate model does not use a known member of an otherwise unavailable pair unless that label is separately retained. It may therefore allow more completions than the task records permit. A direction proved over this larger set holds for every nonempty subset of its feasible completions. An unresolved interval need not imply a reversal in the actual records.

Fix x and complete each correct-baseline count to $s_j + x_j$. Let k_* be the smallest feasible positive overlap after any checked overlap rule. The extreme target corruption counts give

$$L(x) = \frac{\max(0, c_B - s_B - x_B + k_*) - \min(c_A + x_A, k_*)}{k_*}, \tag{C1}$$

$$U(x) = \frac{\min(c_B + x_B, k_*) - \max(0, c_A - s_A - x_A + k_*)}{k_*}. \tag{C2}$$

The lower endpoint corrupts every added correct A baseline and none of the added correct B baselines. The upper endpoint reverses those choices. Full enumeration takes the minimum of equation C1 and the maximum of equation C2 over all feasible x.

C.2 Why the smaller search gives the same result

Fix target counts x_A, x_B and an integer overlap bound $q \geq 0$. Set $r_0 = \max(1, q)$. Every completed correct-baseline set needs at least r_0 members. Each non-target therefore needs $x_j^{\min} = \max(0, r_0 - s_j)$ added correct baselines. If this exceeds u_j, the query is infeasible.

Algorithm 1 Exact bounds with an optional overlap lower bound

Require: $K \geq 1$, group G, counts (N_j, s_j, c_j), distinct $A, B \in G$, integer $q \geq 0$
Check $|G| \geq 2$ and $0 \leq c_j \leq s_j \leq N_j \leq K$ for every j
$u_j \leftarrow K - N_j$; $r_0 \leftarrow \max(1, q)$
if $\min_j(s_j + u_j) < r_0$ **then**
    return no nonempty shared set ($q = 0$), or conflicting evidence ($q > 0$)
end if
Set non-target $x_j^{\min} \leftarrow \max(0, r_0 - s_j)$; compute S_0, M_0 once
$L \leftarrow +\infty$; $U \leftarrow -\infty$; feasible $\leftarrow$ false
for $x_A = \max(0, r_0 - s_A), \ldots, u_A$ **do**
    for $x_B = \max(0, r_0 - s_B), \ldots, u_B$ **do**
        $k_* \leftarrow \max(r_0, S_0 + s_A + x_A + s_B + x_B - (|G| - 1)K)$
        if $k_* \leq \min(M_0, s_A + x_A, s_B + x_B)$ **then**
            Update L, U using (C1)–(C2)
            feasible $\leftarrow$ true
        end if
    end for
end for
if not feasible **then**
    return undefined or conflicting evidence
end if
$e \leftarrow (q = 0$ and $\sum_j s_j \leq (|G| - 1)K)$
return exact rational bounds $[L, U]$ and empty-set flag e
A strict direction needs $e =$ false and either $L > 0$ or $U < 0$.

Take any feasible completion with these target counts. Reduce every non-target addition to its minimum. The minimum feasible positive overlap cannot increase and remains at least r_0. Each reduced set still has at least r_0 members. Also, $\sum_j \tilde{s}_j - (J-1)K$ cannot exceed any completed set size, since the other $J - 1$ sizes are each at most K. The reduced minimum overlap is therefore feasible.

The target counts are unchanged. By Appendix B, lowering the minimum overlap cannot raise the outer lower bound or lower the outer upper bound. Non-target additions above their minima cannot produce more extreme endpoints. Since every reduced completion also belongs to the full search, both searches give the same outer bounds.

For $q = 0$, an empty intersection is possible exactly when

$$\sum_{j \in G} s_j \leq (J-1)K. \tag{C3}$$

Setting every x_j to zero gives the smallest sum of baseline-correct counts. A checked positive overlap rules out the empty set. A bound that conflicts with the counts is rejected rather than used to certify a direction. A zero endpoint does not establish a strict sign.

C.3 Implementation and number of search states

The non-target sum and minimum are computed before the target loops:

$$S_0 = \sum_{j \notin \{A,B\}} (s_j + x_j^{\min}), \qquad M_0 = \min_{j \notin \{A,B\}} (s_j + x_j^{\min}), \tag{C4}$$

With no non-targets, use $S_0 = 0$ and $M_0 = K$. For each target state, calculate $k_* = \max(r_0, S_0 + s_A + x_A + s_B + x_B - (J-1)K)$ and compare it with $\min(M_0, s_A + x_A, s_B + x_B)$. Precomputation takes $O(J)$ steps. Each state then requires constant arithmetic work, giving $O(J + (u_A + 1)(u_B + 1))$ states overall. Integer bit costs and endpoint construction are excluded. Large target missing counts can still make this search costly.

In the reported Kimi–DeepSeek six-way example, full enumeration has 144 missing-count states. The reduced search has one and returns the same endpoints. This compares state counts, not measured runtime. When complete task records are available, direct counting is the appropriate endpoint.

C.4 Examples at the bounds and reported tests

An endpoint witness is a constructed assignment that fits the counts and attains a bound. Missing markers can be placed on y_j corrupted correct baselines, $x_j - y_j$ preserved correct baselines and $u_j - x_j$ incorrect baselines. The remaining observed incorrect baselines can carry the reported rescues. These assignments establish attainability under the information model. Their indices are not original task IDs, and different endpoints may require different assignments.

The reported pairwise suite covers 46,656 constructed assignments; a support-group suite covers 72,900. Another 600 random complete-count tests compare the minimum-support formula with full search. Endpoint tests cover all 120 bounds from 30 condition–pair combinations at two scopes. The original and added suites report 1,387 and 4,589 passed checks. A separate exact-search comparison uses 350 small random problems with $1 \le K \le 7$, two to five configurations and at most two missing pairs per configuration. It also includes 60 reported-count cases and 150 overlap queries. All 560 comparisons agree. These are software tests on the stated inputs, not additional model observations.

D Bounds from Labels Matched by Task ID

D.1 Which tasks belong to the shared set?

For task t and configuration j, retain $b_{jt}, m_{jt} \in \{0, 1, ?\}$ separately. A known incorrect baseline excludes the task from I_G. All-known-correct baselines include it. Otherwise, membership is optional. Let F be the required set and V the optional set. Their sizes give $|F| \le |I_G| \le |F| + |V|$.

The method below assumes no cross-task links between unknown labels. Extra totals, shared responses or cross-condition rules require an optimization that keeps those constraints. They are not automatically enforced by sorting.

If the task belongs to the shared set, its signed contribution is

$$d_t = (1 - m_{Bt}) - (1 - m_{At}). \tag{D1}$$

Known memory labels fix the corresponding terms; unknown labels can be zero or one. Let a_t, z_t be the minimum and maximum contributions. Define $A_F = \sum_{t \in F} a_t$ and $Z_F = \sum_{t \in F} z_t$.

D.2 Exact bounds from sorted lists

Set $f = |F|$ and $v = |V|$. Sort lower contributions as $a_{(1)} \le \cdots \le a_{(v)}$ and upper contributions as $z_{[1]} \ge \cdots \ge z_{[v]}$. If exactly r optional tasks enter and $f + r > 0$, the bounds are

$$L_r = \frac{A_F + \sum_{i=1}^{r} a_{(i)}}{f + r}, \qquad U_r = \frac{Z_F + \sum_{i=1}^{r} z_{[i]}}{f + r}. \tag{D2}$$

The outer interval uses the smallest L_r and largest U_r over feasible r. An overlap bound restricts this to $f + r \ge q$. If $f = 0$ and $q = 0$, an empty set remains possible and no unconditional direction is certified. If no positive size is feasible, the target is undefined.

Every task in F contributes at a fixed set size. The minimum numerator takes the r smallest optional lower contributions; the maximum takes the r largest upper contributions. Selected optional tasks can have all unknown baselines set to correct. Every excluded optional task has an unknown baseline that can be set to incorrect. With no cross-task links, these choices attain the bounds. Membership checks take $O(KJ)$ steps, followed by $O(K \log K)$ steps for sorting and scanning.

Keep the information model fixed. Matched labels narrow the interval only when the target *and all earlier constraints* are retained. The sorted-list formula is exact for its stated partial-label model. Combining counts with partial labels may require extra constraints on totals. In the reported E3 setting, all baseline labels and all initially visible memory labels are restored. They preserve the observed-pair counts and fix the actual shared set.

D.3 A KNOWN SHARED SET WITH SOME MISSING MEMORY LABELS

Suppose baseline labels fix a shared set of size $k > 0$. Let c_A^o, c_B^o count known corruptions on that set. Let u_A^I, u_B^I count unknown target memory labels there. The range becomes

$$\left[\frac{c_B^o - c_A^o - u_A^I}{k}, \quad \frac{c_B^o - c_A^o + u_B^I}{k}\right]. \tag{D3}$$

This is the E3 calculation. Matching tasks does not remove uncertainty in the remaining memory labels. At E4, both unknown counts are zero and the interval becomes the direct finite-set difference.

D.4 WHY KEEPING ONLY COMPLETE CASES CHANGES THE TARGET

Consider two constructed tasks. On task 1, both baselines are known correct; memory preserves A and corrupts B, giving $+1$. On task 2, A has a known correct baseline that memory corrupts. The baseline for B is unknown, but its memory answer is correct.

Keeping only observed-correct common baselines retains task 1 and gives $+1$. The full-plan target allows two cases. If the missing B baseline on task 2 was incorrect, only task 1 is shared and the difference is $+1$. If it was correct, both tasks are shared and the contributions cancel. The full-plan interval is therefore $[0, 1]$, not the point $+1$. These summaries use different targets and must be labeled accordingly.

Reported full-search checks cover synthetic partial-label and complete-label cases, together with tests of the standard export interface. They test the implementation on those inputs, not the original MEMIMPACTBENCH task records.

E ALL AVAILABLE CONTROLLED RESULTS

E.1 ANSWER-CHANGE COUNTS FOR THE EXTERNAL CONFIGURATIONS

Table E1 retains the available external counts for all four Controlled conditions. It separates scored pairs, correct baselines, rescues, corruptions, conditional rates and net change. Each condition keeps its own denominator. NA denotes a zero opportunity count, not a failed request or measured zero performance.

Archived per-configuration results recover the full Controlled counts for the original four-model core (GPT, Grok, Gemini and GLM). Their sums exactly reproduce the historical core totals in Table E2: 80/0 rescues/corruptions for valid memory, 6/48 for irrelevant memory, 5/654 for misleading memory, and 5/638 for stale memory. This cross-check supports the per-model extraction used in Table I4.

E.2 THE ORIGINAL CORE AND ITS EXTENSION

Table E2 retains the original core of GPT, Grok, Gemini and GLM, and its extension with Kimi and DeepSeek. These historical groups differ from the shared-task audit group of Claude, Kimi, Qwen, Meta, DeepSeek and Grok.

These group summaries neither recover unreported individual cells nor establish task matches or independent model samples. Their frequencies use all observed pairs. Conditional rates in Table E1 instead use baseline errors or correct baselines.

Table E1: **Available Controlled answer-change counts.** N counts scored pairs and S counts correct baselines. R_c, C_c count rescues and corruptions. Rates are percentages. Δ is in percentage points. Each condition keeps its source denominator.

Condition	N	S	R_c	C_c	r (%)	h (%)	Δ (pp)
Qwen							
Valid	382	350	32	0	100.00	0.00	+8.38
Irrelevant	383	350	2	9	6.06	2.57	-1.83
Misleading	383	350	0	104	0.00	29.71	-27.15
Stale	383	350	0	114	0.00	32.57	-29.77
Meta							
Valid	350	350	0	0	NA	0.00	0.00
Irrelevant	350	350	0	18	NA	5.14	-5.14
Misleading	349	349	0	179	NA	51.29	-51.29
Stale	349	349	0	180	NA	51.58	-51.58
Claude							
Valid	383	383	0	1	NA	0.26	-0.26
Irrelevant	383	383	0	5	NA	1.31	-1.31
Misleading	383	383	0	22	NA	5.74	-5.74
Stale	383	383	0	23	NA	6.01	-6.01
Kimi							
Valid	384	384	0	0	NA	0.00	0.00
Irrelevant	384	384	0	0	NA	0.00	0.00
Misleading	384	384	0	78	NA	20.31	-20.31
Stale	384	384	0	87	NA	22.66	-22.66
DeepSeek							
Valid	384	325	59	0	100.00	0.00	+15.36
Irrelevant	384	325	10	83	16.95	25.54	-19.01
Misleading	384	325	0	205	0.00	63.08	-53.39
Stale	384	325	0	199	0.00	61.23	-51.82

Table E2: **Counts for the original core and its matched extension.** Rates use all observed pairs, not only baseline errors or correct baselines. This six-model extension is not the six-configuration preservation-audit group.

Measure	Valid	Irrelevant	Misleading	Stale
Four-model core: GPT, Grok, Gemini and GLM				
Observed pairs	1536	1536	1536	1536
Rescues	80	6	5	5
Corruptions	0	48	654	638
Net change (pp)	+5.21	-2.73	-42.25	-41.21
Historical six-model extension: core plus Kimi and DeepSeek				
Observed pairs	2304	2304	2304	2304
Rescues	139	16	5	5
Corruptions	0	131	937	924
Net change (pp)	+6.03	-4.99	-40.45	-39.89

F ALL SHARED-TASK COMPARISONS

F.1 BOUNDS FOR EACH PAIR AND FOR THE ALL-SIX SET

Tables F1 and F2 give every comparison among Claude, Kimi, Qwen, Meta, DeepSeek and Grok. Each label means first minus second. The order follows the reported own-task rates for readability; it is not an assumed shared-task ranking. Positive values mean greater corruption for the first configuration. All intervals allow the reported unavailable pairs.

Table F1: **Shared-task bounds under misleading memory.** Each row is first minus second, in percentage points. Each pair has its own shared set. The six-way columns use the shared set of one fixed group.

	Each pair's shared set		All-six shared set	
Comparison	Lower	Upper	Lower	Upper
Kimi – Claude	+14.32	+14.62	-8.98	+30.59
Qwen – Claude	+23.08	+30.00	-5.10	+41.02
Meta – Claude	+40.62	+53.19	+21.65	+73.79
DeepSeek – Claude	+56.00	+63.27	+44.14	+80.39
Grok – Claude	+64.32	+64.75	+46.48	+100.00
Qwen – Kimi	+7.41	+17.14	-27.06	+41.02
Meta – Kimi	+26.30	+38.97	+2.41	+73.79
DeepSeek – Kimi	+39.08	+57.23	+22.35	+80.39
Grok – Kimi	+50.00	+50.00	+24.71	+100.00
Meta – Qwen	+11.68	+34.92	-7.45	+66.67
DeepSeek – Qwen	+22.95	+54.98	+12.11	+76.86
Grok – Qwen	+37.61	+47.43	+14.45	+96.48
DeepSeek – Meta	-3.10	+29.31	-17.25	+47.06
Grok – Meta	+14.58	+26.07	-14.90	+68.52
Grok – DeepSeek	+1.85	+20.00	-25.10	+47.06

Table F2: **Shared-task bounds under stale memory.** Each row is first minus second, in percentage points. Each pair has its own shared set. The six-way columns use the shared set of one fixed group.

	Each pair's shared set		All-six shared set	
Comparison	Lower	Upper	Lower	Upper
Kimi – Claude	+16.41	+16.71	-9.38	+34.12
Qwen – Claude	+25.64	+32.86	-1.57	+44.92
Meta – Claude	+40.62	+53.33	+21.65	+74.14
DeepSeek – Claude	+53.85	+61.42	+41.41	+78.04
Grok – Claude	+64.06	+64.49	+46.09	+100.00
Qwen – Kimi	+7.69	+17.43	-26.67	+44.92
Meta – Kimi	+24.22	+36.68	-0.39	+74.14
DeepSeek – Kimi	+34.46	+52.62	+16.47	+78.04
Grok – Kimi	+47.66	+47.66	+21.18	+100.00
Meta – Qwen	+9.12	+32.06	-10.98	+63.14
DeepSeek – Qwen	+17.47	+49.48	+5.86	+70.59
Grok – Qwen	+34.76	+44.57	+10.55	+92.58
DeepSeek – Meta	-5.52	+26.90	-20.00	+44.31
Grok – Meta	+14.32	+25.79	-15.29	+68.15
Grok – DeepSeek	+3.69	+21.85	-22.75	+49.41

Under either harmful condition, the observed correct-baseline counts in the group order above are

$$(383, 384, 350, 349, 325, 384),$$

The counts sum to 2,175, giving a minimum overlap of $2175 - 5 \cdot 384 = 255$. DeepSeek's complete correct-baseline set fixes the maximum at 325. Feasible missing-pair completions preserve these outer limits. Endpoints from different comparisons need not occur together: each may require its own task assignment.

The intervals compare recorded configurations, including prompts, decoding, versions and provider services. Pair-specific rows have different task sets. Only the six-way columns use the intersection of one fixed group.

F.2 Two examples with the same summary counts

Table F3 expands the Introduction's Meta–DeepSeek example. A 290-task shared set leaves 59 Meta-only and 35 DeepSeek-only correct-baseline positions. The lower-bound assignment places 179 Meta and 170 DeepSeek corruptions inside it; the upper-bound assignment places 120 and 205. Remaining corruptions lie on each model's correct baselines outside the shared set.

Table F3: **Two artificial assignments with the same counts.** Both use 290 shared correct-baseline tasks, 179 Meta corruptions and 205 DeepSeek corruptions. They are examples that reach the bounds, not recovered model records.

Region	Positions	Meta C_c	DeepSeek C_c
Assignment at the lower bound: $(170 - 179)/290 = -3.10$ *pp*			
Both baseline-correct	290	179	170
Meta-only baseline-correct	59	0	0
DeepSeek-only baseline-correct	35	0	35
Assignment at the upper bound: $(205 - 120)/290 = +29.31$ *pp*			
Both baseline-correct	290	120	205
Meta-only baseline-correct	59	59	0
DeepSeek-only baseline-correct	35	0	0

For these constructed examples, Meta's 35 unavailable pairs receive incorrect baseline labels. Both assignments retain 179 Meta corruptions, 205 DeepSeek corruptions, the observed correct-baseline counts and the same overlap. The resulting differences are $(170 - 179)/290 = -3.10$ and $(205 - 120)/290 = 29.31$ points. Even exact overlap size cannot identify the direction without knowing where corruptions fall. These assignments are endpoint witnesses, not original model records.

G How Much Overlap Information Is Needed?

G.1 The query and what it means

Let $\mathcal{F}(q)$ contain all task assignments and binary completions that fit the condition's reported counts and the checked bound $|I_G| \geq q$. For an allowed nonempty shared set, define

$$L_G(q) = \min_{f \in \mathcal{F}(q)} \theta(f), \qquad U_G(q) = \max_{f \in \mathcal{F}(q)} \theta(f). \tag{G1}$$

Because $\mathcal{F}(q+1) \subseteq \mathcal{F}(q)$, a stronger feasible overlap bound cannot widen the interval. Searching feasible integers in order gives the first q with $L_G(q) > 0$. A chosen positive gap δ instead requires $L_G(q) \geq \delta$. Zero endpoints do not certify strict signs, and inconsistent overlap rules give no certificate.

A query returns one of three outcomes: the counts already suffice; a checked overlap count would suffice; or no feasible overlap count alone suffices. The last case calls for task membership or answer labels. It need not require new model calls if those records already exist.

G.2 Reported thresholds and tests

Table G1 lists the supplied positive-direction and ten-point-gap thresholds. "Already" means the query is settled at the count-implied minimum overlap of 255. "IDs" means that an overlap count

alone cannot settle it. The maximum feasible overlap is 325. These values are evidence requirements, not observed overlap sizes.

Table G1: **Minimum checked overlap needed for six-way comparisons.** "Already" means the counts are enough. "IDs" means no allowed overlap count alone proves the requested direction or gap. The maximum possible overlap is 325.

	Misleading memory		Stale memory	
Comparison	> 0	≥ 10 pp	> 0	≥ 10 pp
Kimi – Claude	IDs	IDs	322	IDs
Qwen – Claude	271	300	262	290
Meta – Claude	Already	Already	Already	Already
DeepSeek – Claude	Already	Already	Already	Already
Grok – Claude	Already	Already	Already	Already
Qwen – Kimi	IDs	IDs	325	IDs
Meta – Kimi	Already	315	292	324
DeepSeek – Kimi	Already	Already	Already	Already
Grok – Kimi	Already	Already	Already	Already
Meta – Qwen	311	IDs	320	IDs
DeepSeek – Qwen	Already	Already	Already	268
Grok – Qwen	Already	Already	Already	Already
DeepSeek – Meta	IDs	IDs	IDs	IDs
Grok – Meta	IDs	IDs	IDs	IDs
Grok – DeepSeek	320	IDs	314	IDs

The ten-point gap is a chosen threshold for this check. It is not a significance level or a loss in real use. *IDs* means more detailed evidence is needed. It does not state an observed direction.

A five-point query is mentioned in the source, but its full threshold table is not supplied and is not reconstructed here. Reported overlap tests cover 46,656 constructed assignments and 3,216 checks of nested bounds, minimum thresholds and infeasible queries. These check the algorithm rather than validate original model records.

The narrowing result concerns what remains possible as evidence grows. It does not say that a measured corruption difference must increase with actual overlap. Changing the set and learning more about a fixed set are different operations.

H Available Outputs, Full-Plan Effects and Representative Gains

H.1 Missing-pair bounds for one configuration

Each unknown binary pair contributes -1, 0 or $+1$ to net correctness change. With observed net count $D = R_c - C_c$ and $u = K - N$ unavailable pairs, the worst-case full-plan bounds are

$$\Delta_{\text{full}} \in \left[\frac{D-u}{K}, \frac{D+u}{K}\right], \qquad h_{\text{full}} \in \left[\frac{C_c}{S+u}, \frac{C_c+u}{S+u}\right], \quad S > 0. \tag{H1}$$

For conditional corruption, the minimum sets all unavailable baselines correct and preserved, while the maximum sets all unavailable baselines correct and corrupted. Swapping baseline-error and correct-baseline opportunities gives the corresponding correction-rate bounds. Possible zero denominators require separate handling. These worst-case bounds do not assume random missingness; known labels restrict the feasible completions further.

H.2 All incomplete Controlled conditions

Table H1 includes every reported Controlled condition with unavailable pairs. Fully observed conditions have point values for this missingness calculation, not zero population or generation uncertainty. Meta's valid and irrelevant effects allow either sign, while its misleading and stale effects

stay negative. Claude's valid-memory effect can reach zero. Both strong and weak conclusions are retained.

Table H1: **Observed net effects and bounds with missing pairs.** $U = 384 - N$. Both effect columns are in percentage points. Possible completions are not added to scored records.

Condition	N	U	Observed Δ	Full-plan interval
Qwen				
Valid	382	2	+8.38	$[+7.81, +8.85]$
Irrelevant	383	1	-1.83	$[-2.08, -1.56]$
Misleading	383	1	-27.15	$[-27.34, -26.82]$
Stale	383	1	-29.77	$[-29.95, -29.43]$
Claude				
Valid	383	1	-0.26	$[-0.52, 0.00]$
Irrelevant	383	1	-1.31	$[-1.56, -1.04]$
Misleading	383	1	-5.74	$[-5.99, -5.47]$
Stale	383	1	-6.01	$[-6.25, -5.73]$
Meta				
Valid	350	34	0.00	$[-8.85, +8.85]$
Irrelevant	350	34	-5.14	$[-13.54, +4.17]$
Misleading	349	35	-51.29	$[-55.73, -37.50]$
Stale	349	35	-51.58	$[-55.99, -37.76]$

H.3 Paired results in the Representative track

Table H2 uses each action's own baseline join on Representative tasks. Pair counts differ across actions even within one configuration because an action and its no-memory baseline must both have a scored terminal for the same archived cluster. Local abstention is not an observed correct answer, and these action results do not evaluate a trained selector.

Combining these gains with Controlled losses would require assumptions about the task distribution, memory quality and losses in real use. Retries on these different tasks also do not supply repeated Controlled no-memory baselines. The missing retrieval corpus and construction records limit further interpretation of the retrieved-memory condition.

H.4 Verified-correct delivery

Let K count planned opportunities, n count scoreable outputs, and a count verified-correct outputs. Report

$$\text{Scoreable accuracy} = \frac{a}{n}, \qquad \text{Coverage} = \frac{n}{K}, \qquad \text{Correct delivery} = \frac{a}{K}. \tag{H2}$$

The three rates separate correctness among observed outputs, output coverage and correct delivery per planned opportunity. Missing outputs have no semantic correctness label and supply no verified correct response. In the DeepSeek retrieved-memory example, $a = 62$, $n = 236$ and $K = 240$, giving 26.27%, 98.33% and 25.83%. The archived no-memory join contains 228 pair-complete clusters and gives $(R_c, C_c) = (60, 1)$, hence a paired net change of +25.88 percentage points. The marginal delivery rates and the paired transition rate answer different questions.

Table H2: **Representative actions matched to their own baselines.** Each action keeps its own pair-complete denominator. These rows do not test a trained selector.

Configuration	Action	Paired N	Rescues	Corruptions	Δ (pp)
GPT-5.6 Terra					
	Retry only	240	1	0	+0.42
	Self-memory	240	1	0	+0.42
	Retrieved memory	239	41	0	+17.15
Grok 4.5					
	Retry only	227	1	0	+0.44
	Self-memory	225	0	0	0.00
	Retrieved memory	226	39	0	+17.26
Gemini 3.6 Flash					
	Retry only	240	2	1	+0.42
	Self-memory	240	0	1	-0.42
	Retrieved memory	240	71	0	+29.58
GLM 5.2					
	Retry only	240	0	1	-0.42
	Self-memory	240	0	1	-0.42
	Retrieved memory	240	117	1	+48.33
Qwen 3.7 Max					
	Retry only	239	1	0	+0.42
	Self-memory	239	0	0	0.00
	Retrieved memory	239	63	0	+26.36
Claude Sonnet 5 V12					
	Retry only	239	1	0	+0.42
	Self-memory	238	0	1	-0.42
	Retrieved memory	237	27	0	+11.39
Meta Muse Spark 1.2					
	Retry only	240	2	1	+0.42
	Self-memory	238	2	1	+0.42
	Retrieved memory	240	211	0	+87.92
DeepSeek V4 Flash					
	Retry only	221	1	0	+0.45
	Self-memory	228	0	1	-0.44
	Retrieved memory	228	60	1	+25.88
Kimi K3					
	Retry only	238	2	0	+0.84
	Self-memory	238	1	0	+0.42
	Retrieved memory	239	53	0	+22.18

I Model Configurations, Sources and Release Limits

I.1 Reported execution counts

Table I1 records the nine completed configurations. Each has 3,120 planned units, including local abstentions, for 28,080 units overall. A run can retain 384 Controlled pairs per condition while missing outputs elsewhere. Overall execution availability must therefore remain separate from condition-specific pair counts.

Table I1: **Reported final execution counts.** Each configuration plans 3,120 units, including 240 local abstentions. Controlled pair counts are for each condition. They are not total question counts.

Configuration	Source group	Observed/ local	Unavail- able	Controlled pair N
GPT-5.6 Terra	Reference	3119	1	384
Grok 4.5	Reference	3068	52	384
Gemini 3.6 Flash	Reference	3120	0	384
GLM 5.2	Reference	3120	0	384
Kimi K3	Matched extension	3115	5	384
DeepSeek V4 Flash	Matched extension	3081	39	384
Qwen 3.7 Max	Near-complete	3116	4	382–383
Claude Sonnet 5 V12	Near-complete	3114	6	383
Meta Muse Spark 1.2	With missing pairs	2954	166	349–350
Total	All configurations	27807	273	Not combined

I.2 Recorded model and service IDs

Table I2 retains the recorded API IDs and their source descriptions. These identifiers do not all have the same reproducibility strength: dated snapshots are closest to immutable model references, while stable or service-bound aliases may later resolve to a changed backend. An echoed service ID therefore does not establish an immutable checkpoint. The reported `DeepSeek-V4-Flash-0731` designation is not treated as a separately selectable fixed API ID. We treat prompts, decoding controls, endpoint and provider service as part of the recorded configuration and avoid attributing observed differences to weights alone.

Table I2: **Recorded IDs and source descriptions.** These descriptions come from the source. The providers were not asked to verify them again for this revision.

Label	Recorded API ID	Source description
GPT	`gpt-5.6-terra`	Exact provider ID
Grok	`grok-4.5`	Exact provider ID
Gemini	`gemini-3.6-flash`	Stable/exact ID
GLM	`glm-5.2`	Release ID
Qwen	`qwen3.7-max-2026-06-08`	Dated snapshot
Claude	`claude-sonnet-5`	Provider-pinned dateless ID
Meta	`muse-spark-1.2`	Stable release ID
Kimi	`kimi-k3`	Service-ID-bound
DeepSeek	`deepseek-v4-flash`	Service-ID-bound

Historical Claude V11 is excluded from the current performance counts. Its exact error responses were reportedly not retained, so a suspected request-format problem is not presented as a recovered provider error. The current configuration is Claude V12. The identity descriptions in this appendix are retained source claims, not new provider authentication.

I.3 EVIDENCE SOURCES AND VALIDATION SCOPE

All count-level results in this paper are computed from frozen model summaries or archived result and terminal records. Table I3 separates evidence directly available for the shared-task audit from evidence that is unavailable at the original MEMIMPACTBENCH task-alignment level. Public StrategyQA rows provide an independent matched-task validation of the method, but they do not replace the unavailable original MEMIMPACTBENCH alignment records.

Table I3: **Evidence and what it supports.** The table distinguishes archived MEMIMPACTBENCH evidence, independent public row-level validation, and provider-free diagnostics.

Evidence	Availability	What it supports
Archived model summaries and terminal/result records	Available and checked	Controlled and Representative counts, execution accounting, and summary-evidence shared-task bounds.
Original MEMIMPACTBENCH task-level alignment records	Not available for this shared-task reconstruction	No direct row-level reconstruction of the original MEMIMPACTBENCH shared-task ordering.
Public StrategyQA row-level records	Available and replayed	Independent matched-task validation of interval containment and direction certificates.
Constructed endpoint and synthetic records	Generated and checked	Sharpness witnesses, evidence ablations, missingness stress, and finite-set diagnostics.
Exact-solver checks	Full/reduced/MILP agreement	Implementation checks for the exact endpoints and practical scaling.

Evidence limit. The unavailable item is specifically the original MEMIMPACTBENCH task-level alignment archive needed to reconstruct that study's shared-task ordering directly. The paper therefore treats the MEMIMPACTBENCH analysis as certification from retained summaries. This does not imply that the underlying records never existed, and it does not affect the independent public row-level validation in Appendix J.

Table I4: **Memory results for nine configurations.** Controlled results, Representative results and execution counts stay separate. All displayed cells have been extracted from the archived result or terminal records; the table gives no combined score or ranking on shared tasks.

(a) Controlled Challenge and execution

Configuration	Execution O/U	Valid	Irrelevant (N, S, R_c, C_c) ; r/h (%) ; Δ (pp)	Misleading	Stale
Original four-model core					
GPT-5.6 Terra	3,119 / 1	$(384, 351, 33, 0)$ 100.00/0.00 +8.59	$(384, 351, 0, 14)$ 0.00/3.99 −3.65	$(384, 351, 0, 173)$ 0.00/49.29 −45.05	$(384, 351, 0, 169)$ 0.00/48.15 −44.01
Grok 4.5	3,068 / 52	$(384, 384, 0, 0)$ NA/0.00 0.00	$(384, 384, 0, 12)$ NA/3.13 −3.13	$(384, 384, 0, 270)$ NA/70.31 −70.31	$(384, 384, 0, 270)$ NA/70.31 −70.31
Gemini 3.6 Flash	3,120 / 0	$(384, 377, 6, 0)$ 85.71/0.00 +1.56	$(384, 377, 6, 1)$ 85.71/0.27 +1.30	$(384, 377, 5, 56)$ 71.43/14.85 −13.28	$(384, 377, 5, 57)$ 71.43/15.12 −13.54
GLM 5.2	3,120 / 0	$(384, 343, 41, 0)$ 100.00/0.00 +10.68	$(384, 343, 0, 21)$ 0.00/6.12 −5.47	$(384, 343, 0, 155)$ 0.00/45.19 −40.36	$(384, 343, 0, 142)$ 0.00/41.40 −36.98
Other reported configurations					
Qwen 3.7 Max	3,116 / 4	$(382, 350, 32, 0)$ 100.00/0.00 +8.38	$(383, 350, 2, 9)$ 6.06/2.57 −1.83	$(383, 350, 0, 104)$ 0.00/29.71 −27.15	$(383, 350, 0, 114)$ 0.00/32.57 −29.77
Claude Sonnet 5 V12	3,114 / 6	$(383, 383, 0, 1)$ NA/0.26 −0.26	$(383, 383, 0, 5)$ NA/1.31 −1.31	$(383, 383, 0, 22)$ NA/5.74 −5.74	$(383, 383, 0, 23)$ NA/6.01 −6.01
Meta Muse Spark 1.2	2,954 / 166	$(350, 350, 0, 0)$ NA/0.00 0.00	$(350, 350, 0, 18)$ NA/5.14 −5.14	$(349, 349, 0, 179)$ NA/51.29 −51.29	$(349, 349, 0, 180)$ NA/51.58 −51.58
DeepSeek V4 Flash	3,081 / 39	$(384, 325, 59, 0)$ 100.00/0.00 +15.36	$(384, 325, 10, 83)$ 16.95/25.54 −19.01	$(384, 325, 0, 205)$ 0.00/63.08 −53.39	$(384, 325, 0, 199)$ 0.00/61.23 −51.82
Kimi K3	3,115 / 5	$(384, 384, 0, 0)$ NA/0.00 0.00	$(384, 384, 0, 0)$ NA/0.00 0.00	$(384, 384, 0, 78)$ NA/20.31 −20.31	$(384, 384, 0, 87)$ NA/22.66 −22.66

(b) Representative actions

Configuration	Retry only	Self-memory (N, R_c, C_c) ; Δ (pp)	Retrieved memory
Original four-model core			
GPT-5.6 Terra	$(240, 1, 0)$ +0.42	$(240, 1, 0)$ +0.42	$(239, 41, 0)$ +17.15
Grok 4.5	$(227, 1, 0)$ +0.44	$(225, 0, 0)$ 0.00	$(226, 39, 0)$ +17.26
Gemini 3.6 Flash	$(240, 2, 1)$ +0.42	$(240, 0, 1)$ −0.42	$(240, 71, 0)$ +29.58
GLM 5.2	$(240, 0, 1)$ −0.42	$(240, 0, 1)$ −0.42	$(240, 117, 1)$ +48.33
Other reported configurations			
Qwen 3.7 Max	$(239, 1, 0)$ +0.42	$(239, 0, 0)$ 0.00	$(239, 63, 0)$ +26.36
Claude Sonnet 5 V12	$(239, 1, 0)$ +0.42	$(238, 0, 1)$ −0.42	$(237, 27, 0)$ +11.39
Meta Muse Spark 1.2	$(240, 2, 1)$ +0.42	$(238, 2, 1)$ +0.42	$(240, 211, 0)$ +87.92
DeepSeek V4 Flash	$(221, 1, 0)$ +0.45	$(228, 0, 1)$ −0.44	$(228, 60, 1)$ +25.88
Kimi K3	$(238, 2, 0)$ +0.84	$(238, 1, 0)$ +0.42	$(239, 53, 0)$ +22.18

Reading the cells. Controlled cells show (N, S, R_c, C_c), then r/h in percent, then Δ in percentage points. Representative cells show (N, R_c, C_c) and Δ in percentage points. N counts observed pairs. S counts correct baselines. R_c and C_c count rescues and corruptions. The displayed rates are $100R_c/(N-S)$, $100C_c/S$ and $100(R_c-C_c)/N$.

NA and execution counts. NA means the rate's opportunity denominator is zero. O/U counts observed or locally available / unavailable units, not correct / wrong answers. Each row plans 3,120 units, including 240 local abstentions. Each Controlled condition plans 384 positions. Each Representative action plans 240.

Counts and task sets. Controlled cells use each configuration's own baseline pair set in each condition. Representative cells are pair-complete joins between an action and its archived no-memory baseline on the same cluster ID; therefore their N can be smaller than either action's marginal observed count. Individual cells are not inferred from group totals.

Archive cross-checks. The recovered four-model Controlled cells sum exactly to the historical core totals in Table E2. Reconstructing Grok retrieved-memory from terminal rows gives $(N, R_c, C_c) = (226, 39, 0)$, exactly matching its frozen scientific result. Sources are kept in Appendices E, H and I.

J PUBLIC ROW-LEVEL VALIDATION ON STRATEGYQA

J.1 PUBLIC RECORDS AND FIXED TASK ALIGNMENT

We use the public artifact released with Experiment 1 of Bajaj & Tiganj (2026). The release uses the same 300 StrategyQA items across four deterministic model configurations: DeepSeek V3.1 Terminus, Gemini 2.5 Flash-Lite, Llama 3.3 70B Instruct, and Grok 3 Mini. Each file has 3,900 rows, including 300 unique no-prior baselines. We select conditions whose prior is incorrect for each source in {friends, experts, LLMs} and each group size $k \in \{1, 3, 5, 9\}$. The selected item IDs are identical across all four model files for every condition, giving 12 aligned harmful-context universes of 141–161 items.

The four `results.csv` Git blob hash prefixes are `3f5f0df25a10` (DeepSeek), `26fb9149f45d` (Gemini), `20edf68375fb` (Llama), and `56e1bf0fa70c` (Grok); the supplementary manifest stores the full hashes. The supplementary script `analysis/run_public_row_validation.py` downloads the public CSVs, verifies the complete Git blob hashes, joins each condition to its no-prior baseline by item ID, and writes the raw contrasts and summaries used here. It makes no model-provider calls.

J.2 AUDIT WITH IDS HIDDEN, THEN RESTORED

For each harmful condition, the exact row-level reference first identifies the tasks on which each model is correct without the social prior and then records correct-to-incorrect changes after the incorrect prior is added. We compute pair-specific shared support for all six model pairs and one all-four shared support. We then discard task identities, retain only each model's correct-baseline and corruption counts, and apply the complete-margin audit in Section 3.3. Finally, restoring IDs gives the direct finite-set difference.

Across 12 conditions and six pairs, each scope has 72 contrasts. Every count-only interval contains its row-level truth. Pair-specific counts certify 35 directions and all-four counts certify eight; all 43 certificates match the direct sign. Among nonzero truths, own-support signs disagree with pair-specific truth in 8 of 66 contrasts and with all-four truth in 6 of 67. Midpoints disagree in 3 of 66 and 7 of 67, respectively. Mean count-only width is 13.36 points pair-specifically and 29.94 points on all-four support.

Table J1: **Public StrategyQA validation by prior source.** Each row aggregates four group sizes and six model pairs (24 contrasts). Width is the mean count-only interval width in percentage points. All intervals contain the row-level truth and every strict certificate has the correct sign.

Prior source	Scope	Certificates	Own-rate opposite	Midpoint opposite	Mean width
Experts	Pair-specific	21/24	0	0	16.11
Experts	All-four	8/24	0	0	49.32
Friends	Pair-specific	9/24	2	1	13.57
Friends	All-four	0/24	2	2	24.54
LLMs	Pair-specific	5/24	6	2	10.41
LLMs	All-four	0/24	4	5	15.96

The source-level pattern is descriptive, not a claim about memory systems or social trust. Its role here is methodological: with real aligned outputs, aggregate summaries can lose the sign of a shared-task comparison, while the exact count interval still contains the task-level answer and never certifies an unsupported sign.

K PROVIDER-FREE DIAGNOSTIC EXTENSIONS

This appendix reports provider-free diagnostic experiments. They use no model-provider calls. They test analysis choices, evidence types, missingness stress, and exact-solver scaling; they do not turn summary counts into observed task-level records.

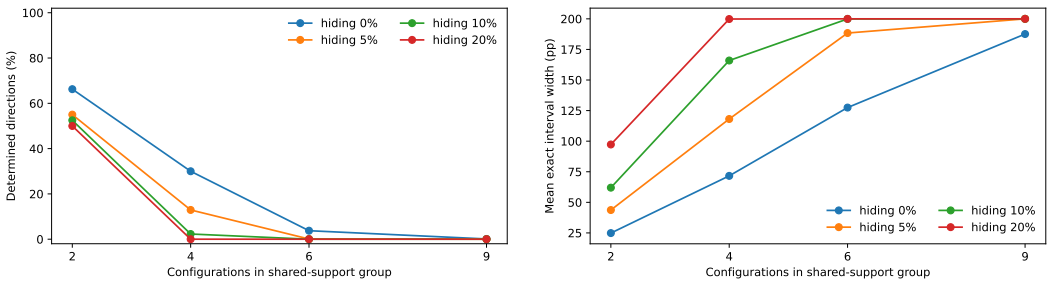
K.1 Analysis-baseline stress suite

The stress generator follows the documented binary-label mechanism but extends the hiding rate to $\eta \in \{0, 0.05, 0.10, 0.20\}$ and the shared-task group size to $J \in \{2, 4, 6, 9\}$. It uses $K \in \{128, 384\}$, baseline dependence $\rho \in \{0, 0.8\}$, separated and near-tied corruption regimes, task-harm slope $\beta \in \{0, 2\}$, and five repeats. A nested nine-configuration parameter grid is used; the six-member subset exactly preserves the reported six marginal baseline and corruption-rate vectors. The master seed is 20260925. This is a new stress design, not a replay of the reported 160-bank study.

Table K1 reports the mean strict-direction rate from the exact count audit. Every issued certificate matches the complete-label truth in this suite. The fall in determination rate is therefore conservatism under less information, not an increase in false certificates.

Table K1: **Stress test: determined directions (%).** Values average over task size, dependence, corruption regime, task-harm slope, and five repeats.

Group size J	0% hidden	5% hidden	10% hidden	20% hidden
2	66.25	55.00	52.50	50.00
4	30.00	12.92	2.29	0.00
6	3.75	0.08	0.00	0.00
9	0.10	0.00	0.00	0.00



(a) Determined directions (b) Mean interval width

Figure K1: **Count-only audit under larger groups and heavier hiding.** More configurations shrink the shared-task intersection target, while hiding enlarges the compatible completion set. Exact certificates remain valid but can become uninformative.

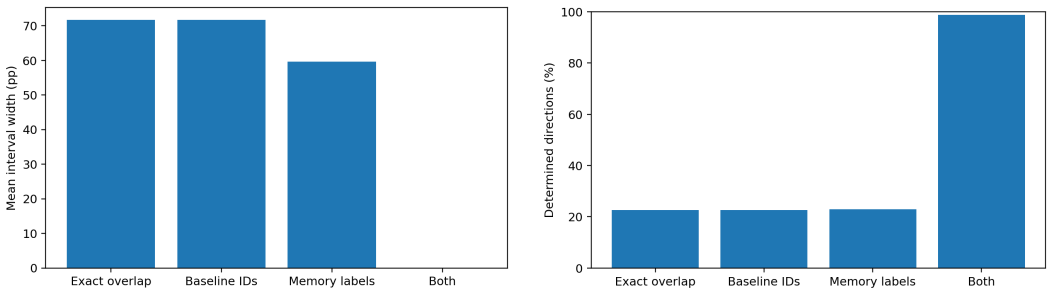
K.2 Evidence-type ablation

The ablation in Table 6 uses 80 complete six-configuration banks and all 15 target pairs, for 1,200 contrasts. It conditions on the true shared-set size so that the comparison isolates task identity from target memory-outcome identity. "Baseline IDs only" reveals the exact shared-task membership but withholds task-level target memory outcomes; under this information model it has the same outer bounds as exact overlap size alone. "Target memory labels only" reveals both target memory-correctness vectors but withholds baseline task membership; a four-cell integer flow computes the exact interval. "Both" links membership and target outcomes and therefore gives the direct finite-set contrast.

K.3 Solver scaling

For the scaling workload, $K = 384$ and each configuration has the same unavailable count u. Marginal baseline and corruption rates span the same ranges used by the stress generator. Full enumeration visits $(u+1)^J$ missing-correct states before feasibility pruning, whereas the reduced solver visits $(u_A+1)(u_B+1)$ target states after $O(J)$ precomputation. Both implementations stream states; neither materializes the Cartesian grid.

As a stronger exact baseline, we also solve the aggregate count model with a generic MILP. For each positive candidate support size k, integer variables choose each configuration's added correct-baseline



(a) Mean interval width (b) Determined directions

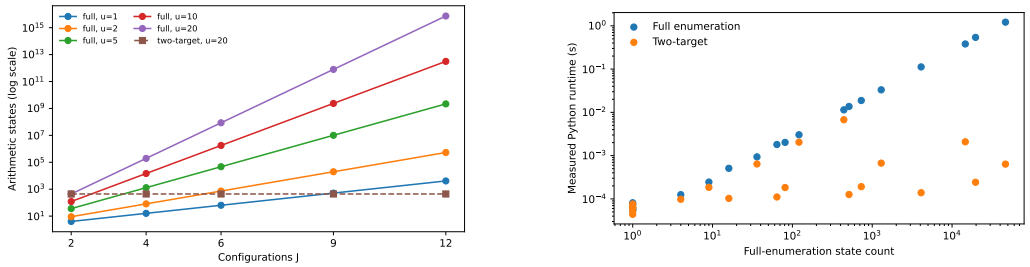
Figure K2: **Which task-level evidence matters?** Baseline IDs or target memory labels in isolation leave wide intervals. Linking shared-task membership with target outcomes identifies the finite-set contrast; remaining non-strict cases are exact ties.

count, the two targets' added corruptions, and the two shared corruption counts. Linear constraints enforce k-way overlap feasibility and the fixed-k corruption bounds; two objectives minimize and maximize the signed shared corruption count. Scanning feasible k values gives the exact interval without using Theorem 1. On all 30 scaling workloads, this MILP baseline and the reduced solver return identical endpoints.

Across cases timed with both Cartesian and reduced solvers, traced additional Python allocation is 1.44–2.27 KiB for full enumeration and 0.73–1.13 KiB for the reduced solver, so state count and runtime are the meaningful scaling differences. The MILP baseline is implemented with `scipy.optimize.milp` (SciPy 1.17.0). A separate 1,000-case random finite check also returns identical reduced/full endpoints.

Table K2: **Selected exact-solver scaling cases.** Runtimes are median Python wall-clock seconds on the manuscript-generation host. The generic MILP scans feasible support sizes and solves two integer programs per size. "–" means full enumeration exceeded the timing cap. All methods return identical endpoints on every row.

J	u	Full states	Two-target states	Full (s)	MILP (s)	Two-target (s)
4	10	14,641	121	0.381	0.385	0.00209
6	5	46,656	36	1.215	0.381	0.00064
9	2	19,683	9	0.540	0.407	0.00024
12	1	4,096	4	0.112	0.394	0.00014
12	20	7.36×10^{15}	441	–	0.439	0.00760



(a) State-space growth (b) Measured runtime where full search was feasible

Figure K3: **Exact solver scaling.** The theorem removes non-target missing-count dimensions. The plotted Cartesian and two-target implementations stream states; Table K2 also reports a generic exact MILP baseline. The meaningful gain is state count and runtime rather than a claimed memory reduction.