RECOVERY UNDER DRIFT: WHEN FAILED CONTEXT HELPS OR HURTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Failed context can contain both misleading content and correct task information, so removing it changes two things at once. We study this trade-off with controlled interventions that vary correct facts inside a partly wrong record, correct facts supplied separately, failed-code visibility, the source of failures used for training, and conversation retention at fixed files. In tool-use and ALFWorld recovery states, adding correct facts to the retained record raises its benefit for Qwen3-8B and Gemma-4-12B-it. A complete current reference lowers retention benefit in the tool setting, whereas the ALFWorld intervals do not establish the same change. A secondary Qwen tool comparison finds that partial external facts can first increase the record's benefit before a complete reference lowers it. Code experiments show that measured recovery also depends on restart context and failure source: for one 7B model, omitting failed code and feedback raises success within 64 attempts from 22.1% to 55.5%, and matched-source training gains exceed mismatched gains across five model settings. On 34 MettleBench failures at identical repository files, retaining the pre-checkpoint conversation raises mean recovery from 11.8% to 35.8%. Recovery evaluations should state what each condition retains, compare repair with fresh starts, and match inference budgets.

1 INTRODUCTION

Language-model agents work on multi-step tasks that leave behind files, tool observations, and conversation history. After an attempt fails, the agent must decide what to carry into its next attempt. A rejected program, an old tool map, or an earlier conversation may not only contain mistakes but also correct task relevant information. Keeping that history can lead the model toward an error; deleting it can remove information needed to recover. Prior work shows that failed code can hinder retries, that memory value depends on other information in context, and that agents can recover by rewinding files and conversation (Verma, 2026; Yang et al., 2026; Zhuang et al., 2026).

We study four factors that can change measured recovery. On fixed failed programs, we vary whether a retry sees the rejected code. On the same training tasks and targets, we compare training on automatically edited failures with training on failures from a frozen model. At fixed tool-use and embodied-agent states, we vary correct facts in a partly wrong retained context and in a separate reference, keeping the misleading content unchanged. At fixed repository checkpoints, we vary whether a coding agent keeps its earlier conversation.

The importance of these controls is clearest in the tool experiment. On the same 80 failed ScrambleToolBench states (Toh et al., 2026), the benefit of keeping a stale tool map changes from $+7.7$ to -6.6 points for Qwen3-8B, and from $+1.9$ to -8.3 points for Llama-3.1-8B, when a separate reference supplies current task mappings. Both paired interactions are below zero. Under the current reference, Llama's -8.3-point interval is below zero, whereas Qwen's -6.6-point interval ends at $+0.1$. Thus, the same stale map can help when current facts are unavailable and hinder when they are supplied. A second intervention holds the misleading content fixed and adds correct facts inside the record. The record's benefit increases in both the tool setting and ALFWorld (Shridhar et al., 2021), across two tested models. A complete reference reduces this benefit in the tool setting; the ALFWorld result is inconclusive.

The other experiments show why retention must also be evaluated for a particular model and restart boundary. For one recovery-trained 7B coding model, removing failed code and feedback raises

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.

the chance of generating at least one correct program among 64 candidates from 22.1% to 55.5%. A four-prompt factorial with the same model isolates code visibility: showing the failed program without feedback lowers success by 35.2 points relative to a code-hidden failure notice. Across five model settings, recovery training gives a larger relative benefit on the kind of failure used during training, extending work on failure-source mismatch (Kumar et al., 2025; Wu et al., 2026; Choi et al., 2026). On fixed model-generated failures, a recovery-trained model's 18.3-point lead over a task-only model under a shared repair prompt falls to 4.8 points when the task-only model starts fresh. Finally, on 34 failed MettleBench tasks (Zhuang et al., 2026), keeping GPT-5.6 Luna's conversation before a restored checkpoint raises average recovery success from 11.8% to 35.8% relative to a new conversation at identical files; both conditions discard the later failed work.

Our contribution is not the general observation that context can help or hurt, nor a new rule for selecting it. We isolate how facts inside and outside a partly wrong context change the value of retaining it, then test whether failed code is shown, training and evaluation failures share a source, and conversation is kept at fixed files. Measured recovery depends on the information available at restart, the failure distribution, and the restart boundary.

2 Related Work

Work on reflection, retrieval, and correction already shows that earlier information can help or hurt (Shinn et al., 2023; Kim et al., 2026; Shi et al., 2026). For memory retention, MeClear is the closest comparison: it estimates the utility of whole records jointly with other memories (Yang et al., 2026). We instead hold one partly wrong record and the failed state fixed, then vary correct facts in that record and in an independent reference.

Fresh sampling can rival repair, failed code can hinder retries, and correction training can depend on the errors used to train it (Olausson et al., 2024; Verma, 2026; Kumar et al., 2025; Wu et al., 2026; Choi et al., 2026). Our code studies cross training and evaluation failure sources under matched tasks, targets, and budgets; apply code-visibility controls to task-only and recovery-trained models; and compare repair with fresh starts on the same failures. AgentRewind restores files and conversation together and may summarize discarded work (Zhuang et al., 2026). We instead hold the selected files fixed and vary only whether the pre-checkpoint conversation is retained, isolating the conversational part of its intervention. Table 1 states these distinctions directly.

Table 1: Closest comparisons. The last column gives the additional comparison in this paper, not a claim that the broader question is new.

Work	Established result	Additional comparison here
SCoRe (Kumar et al., 2025)	Online self-correction training addresses own-output mismatch.	Edited versus model-generated training failures on fixed evaluation sets.
Verma (Verma, 2026)	Blind retries, failed-code exposure, feedback, and copying on MBPP+.	The same context controls for task-only and recovery-trained models.
Wu et al. (Wu et al., 2026)	Injected-error correction transfers poorly to model-generated errors.	Crossed code-failure sources with matched tasks, targets, and budgets.
Anchored Self-Play (Choi et al., 2026)	Bug-source generalization and repair versus solving from scratch.	Five model settings and fresh baselines on the same frozen failures.
MeClear (Yang et al., 2026)	Memory utility depends on other available records.	Correct facts varied inside one partly wrong record and separately.
AgentRewind (Zhuang et al., 2026)	Joint file/conversation restoration and summaries of discarded work.	Earlier conversation versus a fresh conversation at identical restored files.

3 Conditional Recovery Value

3.1 Available evidence and a policy's use of it

Let E denote information shared within a retention comparison, including the task and any separate reference, and S the earlier context shown only when retained: failed code, a map, a memory, or conversation. Define retention benefit as the paired difference in expected success at the same budget,

$$D_\theta(E, S; b) = J_\theta(E, S; b) - J_\theta(E; b). \tag{1}$$

The expectation is over fixed failed cases; the starting environment state, model, and budget are shared within each comparison. Positive D_θ means retention helps and negative D_θ means omission helps. Its value depends on the model, failures, information in E, and budget, not on S alone.

A benchmark-wide score averages retention effects over its particular failures and references. If the same record helps without a current reference but hurts with one, changing how often that reference is supplied can change which choice looks better overall. A single average may therefore give the wrong guidance for an agent facing a different information supply, even at the same failed states.

If observing S is free and a policy class can ignore it, access to S cannot reduce the class's best achievable success. A fixed model may nevertheless use misleading context poorly. Useful information in a record and successful use of that information are distinct; our experiments vary available information but measure their combined effect on recovery. Appendix F gives the assumptions and proof.

This motivates two conditional predictions. Holding misleading content fixed, an additional correct task fact inside S should increase its benefit if the model can use that fact. Supplying the same fact separately should reduce the record's benefit when that fact becomes redundant. Neither prediction is a monotonicity law: facts may complement one another, and a fixed model may not use them reliably.

4 Experimental Design

A benchmark's keep-or-remove score can depend on its mix of failures and on information supplied elsewhere. We test this by varying correct facts in the retained context and in a separate reference, failed-code visibility, training-failure source, and conversation at fixed file checkpoints. Table 2 shows what changes and what remains fixed in each comparison.

Table 2: Experimental map. Each row defines a separate comparison.

Question	What changes	What remains fixed	Primary quantity
1. When is an old record useful?	Map or memory retention; correct facts inside it and in a separate reference	Environment state, misleading content, and unrelated history	Retain-minus-remove effect by information condition
2. Can failed code hinder recovery?	Retry instruction, code or feedback visibility, code label, or sampling budget	Within each study: model, task, failed program, verifier, and prompt controls	Paired pass@b difference
3. Does recovery training generalize?	Training source crossed with evaluation source	Training tasks and targets; exact failures within each evaluation source	Matched- versus mismatched-source gain over the task-only model
4. Should conversation be retained at fixed files?	Fresh or retained conversation; generated summary is secondary	Task, selected files, model, tools, and recovery limits	Task-mean success

Information interventions. ScrambleToolBench and ALFWorld compare retention of a partly wrong tool map or object-location memory with its omission (Toh et al., 2026; Shridhar et al., 2021). We separately vary correct facts in the retained record and in a new reference, holding misleading content, environment state, and unrelated history fixed. Interactive success is defined as completion of the task in 30 actions. The state construction and model allocations appear with the results and in the appendix.

Code tasks, verification, and training. Questions 2 and 3 use MBPP+ from EvalPlus (Austin et al., 2021; Liu et al., 2023) and a BigCodeBench replication (Zhuo et al., 2025). All models see the task and public tests; depending on the condition, they also see the failed program and public-test error message. Hidden tests only select failures and score outputs. Training and evaluation tasks are disjoint; BigCodeBench retains only tasks whose canonical solution passes the pinned verifier.

In figures and tables, **Controlled** denotes automatically edited failures and **Natural** denotes failures from a frozen coding model. The MBPP+ sets contain 300 edited failures from 50 tasks and 73 model-generated failures from 42 tasks. We adapt Qwen2.5-Coder at 7B, 14B, and 32B (Hui et al., 2024) with response-only LoRA (Hu et al., 2021), and replicate with Qwen3-8B and Seed-Coder-8B (Yang et al., 2025; Zhang et al., 2025). The **Task-only** baseline learns from tasks and verified solutions; recovery conditions also receive a failed program and public feedback. The matched MBPP+ conditions share 43 training tasks, targets, response-token counts, optimizer steps, and three seeds. Direct Recovery uses a separate allocation. Appendix A gives details.

Context and checkpoint interventions. **Forced Fresh** keeps the task, public tests, and a generic failure notice but removes the failed program and feedback; **Blind** removes the notice too. Separate studies cross code and feedback visibility, or hold the failure notice, feedback, chat role, and output instruction fixed while changing code presence or label. MettleBench restores the same selected files while varying conversation; a generated-summary arm is secondary (Zhuang et al., 2026).

Metrics, pairing, and uncertainty. For static code generation, if one sample from history h passes the verifier with probability $p_\theta(h)$, success within b independent samples is

$$R_\theta(h; b) = 1 - (1 - p_\theta(h))^b. \tag{2}$$

We use empirical pass@b (Chen et al., 2021) at $b \in \{1, 8, 64\}$; interactive experiments use a fixed action limit. Code comparisons reuse failures and pair sampling streams where applicable. Information studies pair continuations at the same state; MettleBench reuses checkpoints but samples independent continuations. We aggregate repeated outcomes within task or prespecified environment cluster and report 95% cluster-bootstrap intervals and paired tests. Appendices give allocations, exclusions, and sensitivity analyses.

5 When Retained Context Preserves Useful Information

Question 1 separates three parts of a recovery context: misleading statements in an old record, correct task facts in that record, and correct facts supplied outside it. ScrambleToolBench uses a stale tool map; ALFWorld uses an object-location memory. Within each comparison, we fix the task, environment state, misleading content, and unrelated history. We then measure how the keep-minus-remove effect changes when facts move inside or outside the record.

A fixed record with two references. The first tool study keeps the stale map unchanged and varies only the separate reference. Across 80 states, Qwen3-8B's retention effect changes from $+7.7$ points with matched control facts to -6.6 points with current task mappings, an interaction of -14.3 points. Llama-3.1-8B-Instruct changes from $+1.9$ to -8.3 points, an interaction of -10.2 points. Both interactions have intervals below zero. The negative full-reference effect is individually established for Llama but not for Qwen, so the interaction and the effect within one reference condition should not be conflated.

Retained-fact and external-fact dose grids. A second tool study varies the number of correct mappings inside the record (0–2) and in the separate reference (0–4), while keeping stale and distractor mappings fixed. Averaged over the other axis, increasing retained facts raises retention benefit for Qwen3-8B by 7.2 points and for Gemma-4-12B-it by 7.6 points; increasing external facts lowers it by 9.3 and 13.5 points, respectively (Table 3). The full response curves differ. With the original Qwen map, retention benefit rises from 6.4 points with no external facts to 14.1 with two, then falls to -7.3 with four. The first increase is a secondary contrast ($+7.7$ points, 95% interval $[1.6, 14.4]$) and does not recur as a resolved increase for Gemma. With a complete reference, Gemma reaches 96–97% success in every keep/remove cell and the retention intervals include zero. This near-ceiling result limits detectable differences; it does not establish that the map is harmless or redundant.

A second environment. In 60 ALFWorld states, keeping one correct task fact instead of none raises the memory's value for Qwen3-8B ($+9.2$ points) and Gemma-4-12B-it ($+7.6$ points). Supplying locations separately lowers the point estimate for both models, but neither interval excludes zero (-10.2 and -3.8 points). The retained-fact contrast is therefore supported in all four model–environment grids, while the external-fact contrast is established only in the tool setting (Appendices G–H).

Table 3: Primary endpoint changes in retention benefit, averaged over the other dose axis. Retention benefit is success with the stale record minus success without it. Entries are percentage points with 95% paired cluster-bootstrap intervals: 21 episode clusters for tool use and 37 floor-plan clusters for ALFWorld. Tool doses span 0–2 correct retained facts and 0–4 external facts; ALFWorld spans 0–1 and 0–2. Models and environments are not pooled. Both ALFWorld external-fact intervals include zero.

Environment / model	Retained facts	External facts
Tool / Qwen3-8B	$+7.2\,[4.0, 10.6]$	$-9.3\,[-15.4, -3.3]$
Tool / Gemma-4-12B-it	$+7.6\,[3.1, 12.0]$	$-13.5\,[-20.2, -6.9]$
ALFWorld / Qwen3-8B	$+9.2\,[4.2, 14.8]$	$-10.2\,[-19.8, 0.1]$
ALFWorld / Gemma-4-12B-it	$+7.6\,[2.8, 11.8]$	$-3.8\,[-16.5, 7.6]$

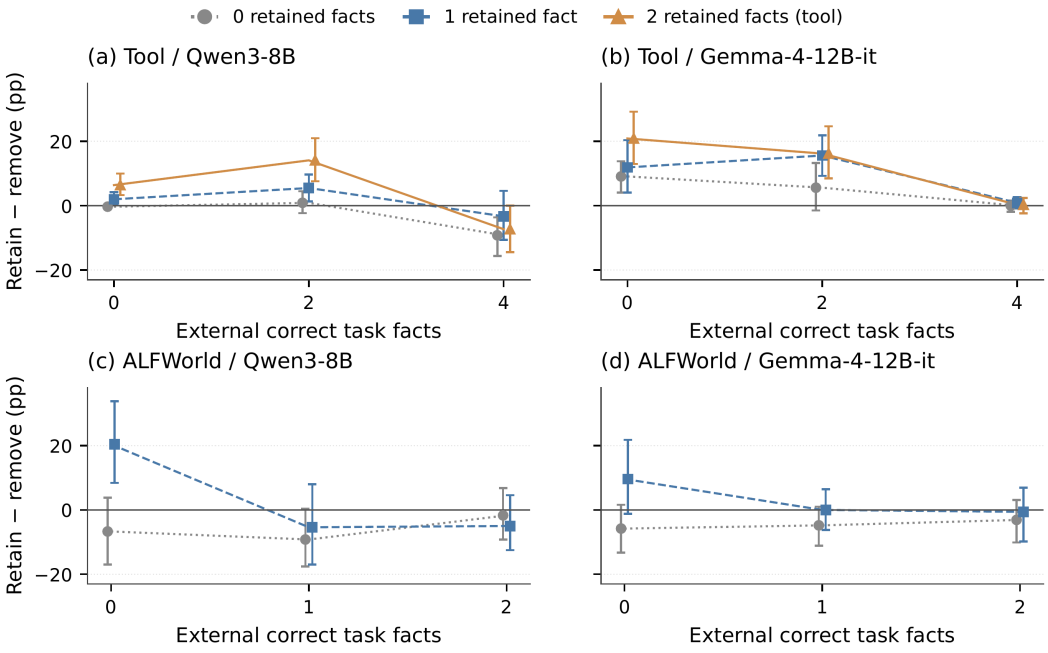


Figure 1: **Retention effects across the dose grid.** Reported retention effects with 95% paired cluster-bootstrap intervals. Tool results cluster by episode; ALFWorld results cluster by floor plan. Points are offset horizontally for legibility. Lines connect only observed dose levels. The number of correct retained task facts defines each series.

Confidence is not correctness. Separate probes ask the model to name the correct tool or location but their answers never enter the recovery history. With only the stale record, the model's prediction is often more concentrated yet less often correct than with no record at all. A confident prediction therefore does not show that the record holds useful knowledge, and low entropy alone cannot decide what to retain.

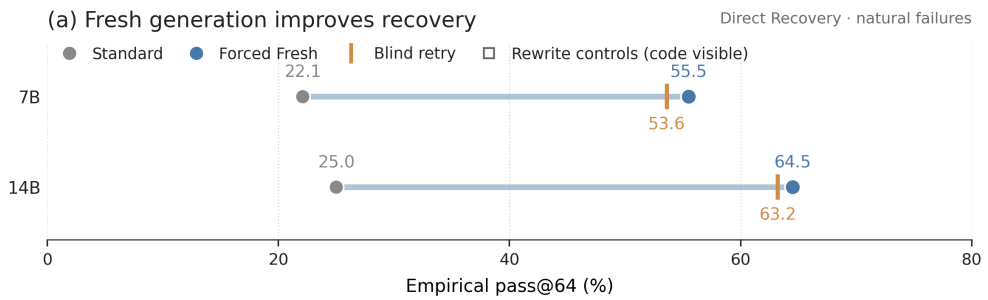


Figure 2: **Fresh generation on failures produced by the evaluated model.** Direct Recovery's success within 64 attempts under Standard, Forced Fresh, and Blind; squares show two 7B rewrite prompts that keep the failed code visible. Rates pool failures. Paired task-average effects and copying diagnostics appear in Appendix C.2.

6 Failed-Code Context Can Hinder Recovery

Fresh retries on the same failures. We first test Direct Recovery, a Qwen2.5-Coder model trained on 128 edited failures and separate from the matched models in Section 7. On programs generated incorrectly by each adapter, **Standard** shows the failed program and feedback, **Forced Fresh** reports a failure but omits both, and **Blind** shows only the task and public tests.

For the 7B model, success within 64 generated programs rises from 22.1% under Standard to 55.5% under Forced Fresh; Blind reaches 53.6% (Figure 2). The paired, task-averaged Forced Fresh gain is 33.7 percentage points (95% confidence interval: 23.7 to 44.0 points). At 14B, success rises from 25.0% to 64.5%. To narrow the source of this gain, we next compare retries with and without the failed program and its feedback.

What changes when code or feedback is shown? Four prompts cross failed-code and public-feedback visibility; traceback lines that quote the program are removed in the feedback-only condition. For Direct Recovery at 7B, showing code without feedback lowers success within 64 attempts by 35.2 points relative to a code-hidden failure notice (95% interval: 25.1 to 45.2 points lower). Feedback without code changes success by 1.1 points with an interval spanning zero, while adding feedback when code is visible improves it by 4.6 points (95% interval: 1.1 to 9.1). The prompts differ slightly in wording, and the full factorial appears in Appendix C.2.

Presentation controls. On a separate fixed set, relabeling identical code from *Previous attempt* to *Candidate program* changes per-program success by -1.9 points on model-generated failures and $+1.5$ points on edited failures, averaged across training conditions. Several code-presence effects are larger than these label effects; Appendix C.6 reports all contrasts and copying measures.

7 Recovery Training Specializes to Failure Sources

We train on one failure source and evaluate on both, holding tasks, targets, training allocation, and evaluation failures fixed. Controlled Recovery and Natural Recovery therefore differ only in the failed programs shown during training.

At 7B, training on edited failures raises success on edited failures by 23.7 points over the task-only model but changes success on model-generated failures by -3.8 points. Training on model-generated failures raises success on those failures by 14.4 points and lowers it on edited failures by 10.9. The matched-minus-mismatched interactions are 27.5 and 25.3 points, respectively, and the same specialization pattern holds in all five model settings (Figure 3; Table 7). Absolute matched gains are less stable: Qwen3-8B does not establish a gain on model-generated failures, and Seed-Coder-8B depends on the treatment of training-seed variation (Appendix B.3).

The baseline decides how large the gain looks. With 64 attempts, Natural Recovery is 18.3 points above the task-only model when both see the failed program, but 4.8 points above a task-only model that starts from the task description. No fresh-start comparison at 7B, 14B, 32B, or in the Qwen3-8B own-failure study establishes an advantage (Tables 12 and 16). Edited failures differ because hiding the program hurts the task-only model. On BigCodeBench, specialization replicates, but Natural Recovery's 35.9-point repair-prompt lead becomes a 6.6-point deficit against the fresh baseline at 64 attempts (Table 17).

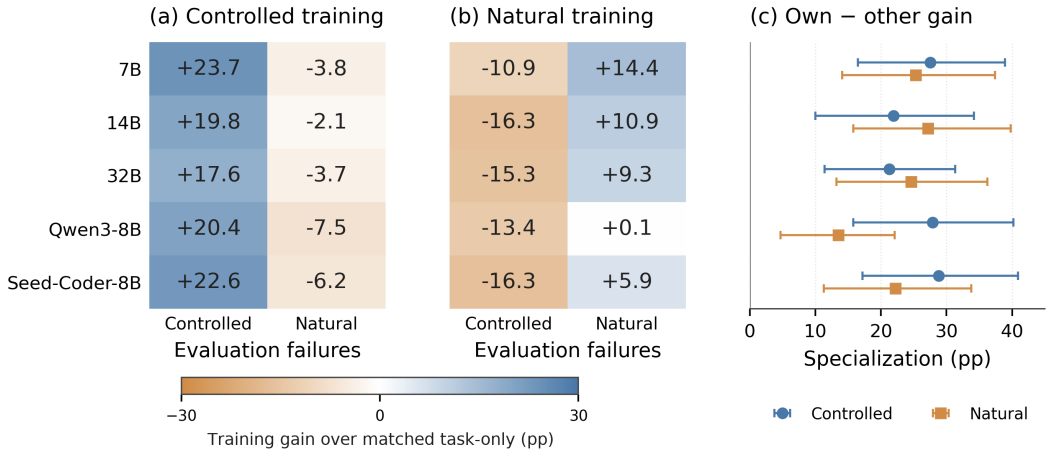


Figure 3: **Recovery training specializes by failure source.** One-shot task-balanced gains over the matched task-only model. Cells show percentage-point estimates; panel (c) shows matched-minus-mismatched interactions with 95% task-cluster bootstrap intervals. Table 7 reports all intervals. 7B/14B/32B denote Qwen2.5-Coder; color encodes effect size, not significance.

Table 4: **Fresh-start baselines change the measured recovery gain.** Natural Recovery at 64 attempts on fixed model-generated failures. Entries are task-balanced paired effects in percentage points with 95% task-cluster bootstrap intervals against task-only repair and fresh-start baselines. All models are Qwen2.5-Coder; the MBPP+ fresh-start intervals include zero.

Benchmark / model	vs task-only repair	vs task-only fresh
MBPP+ / 7B	$+18.3\,[9.1, 28.2]$	$+4.8\,[-4.0, 13.5]$
MBPP+ / 14B	$+13.5\,[2.4, 25.4]$	$-6.0\,[-14.7, 2.4]$
MBPP+ / 32B	$+18.3\,[7.9, 29.8]$	$+4.4\,[-0.4, 9.9]$
BigCodeBench / 7B	$+35.9\,[22.2, 49.5]$	$-6.6\,[-14.1, -0.5]$

Sampling-budget, training-dose, and cross-policy transfer results appear in Appendix E.

8 Conversation Retention at a Fixed Repository State

Question 4 moves from single programs to a long-horizon coding agent. Returning to a repository checkpoint involves two decisions: which files to restore and whether to keep the conversation that produced them. AgentRewind restores both together (Zhuang et al., 2026); we fix the files and vary only the conversation.

Design. We reserve 50 of MettleBench's 82 tasks, separate from development. GPT-5.6 Luna completes 49 initial attempts; 34 failures meet the eligibility rules. For each, the model chooses an earlier file checkpoint. We restore those same files with either a fresh conversation containing only the task or the model's conversation up to the checkpoint. A secondary condition adds a generated summary of the abandoned continuation. Each condition uses three continuations, high reasoning

effort, a 30-action limit, and a 60,000-output-token limit. No condition sees conversation after the checkpoint.

Results. Keeping the pre-checkpoint conversation raises task-mean success from 11.8% to 35.8%, a gain of 24.0 points (95% interval: 11.8 to 37.3). Twelve of 306 planned continuations lack an observed outcome. Extreme assignments move the gain between 20.6 and 26.5 points; these are bounds, not confidence intervals. The generated-summary condition reaches 40.2%, but its 4.4-point gain over the native prefix is unresolved. Full results and sensitivity checks appear in Appendix I.

The result answers a narrow question: given the files this model chose to return to, its earlier conversation was worth keeping on these tasks. It says nothing about whether the model chose good checkpoints, and it does not compare against AgentRewind's full system. Unlike Section 6, where removing the failed program helped, both conditions here already drop the failed continuation and removing the earlier conversation as well hurts.

9 Discussion and Limitations

What the results say together. The information experiments separate correct facts kept inside a partly wrong record from facts supplied elsewhere. More retained facts raise the record's value in both environments and for both tested models. A complete separate reference lowers that value in the tool setting, while ALFWorld does not establish the same change. The secondary Qwen tool pattern is not monotone, and Gemma's near-ceiling condition cannot show that the retained map is redundant or harmless. These are effects on success for fixed models and states; they do not identify which facts the model used.

The code and repository experiments show two other boundaries. Recovery training gives larger relative gains on failure sources similar to its training input, but those gains can shrink against a fresh-start task-only baseline. At fixed MettleBench files, keeping the pre-checkpoint conversation helps, but the experiment does not separate factual content from reasoning or other conversational state.

How recovery should be evaluated. Evaluations should report what is retained and what is supplied separately, compare recovery training with a task-only model under repair and fresh-start prompts, use failures like those expected at deployment, and match inference budgets. For agents, restoring files and keeping conversation should be treated as separate choices. A deployable retention policy would also need signals available at test time and must account for the cost of making the choice.

Scope of the evidence. The information states are constructed: the 80 tool states come from 21 episodes and 18 task templates, and ALFWorld uses 60 constructed states. Qwen3-8B rarely succeeds without a reference, Gemma nearly saturates with a complete one, and Gemma has four continuations per cell rather than eight. The ALFWorld external-reference intervals include zero. Most code results use MBPP+; the BigCodeBench replication keeps only 648 of 1,140 tasks whose canonical solutions pass the pinned verifier. Training on model-generated failures uses 43 tasks, and model-specific failure distributions differ. These experiments support the stated comparisons, not a general law of retention or an internal mechanism.

Agent experiment. MettleBench uses one hosted model and client on 34 failures at model-chosen checkpoints. Twelve continuations lack observed outcomes; extreme assignments preserve the direction of the primary effect but do not broaden its scope. Checkpoint choice and transfer to other agent scaffolds remain untested.

10 Conclusion

Retaining failed context trades exposure to misleading material against access to correct information. Its measured value changes when facts move inside or outside a record, when the model or failure source changes, and when a restart removes code or conversation. Recovery evaluations should therefore state what each condition retains, compare repair with fresh starts, and match inference budgets.

Reproducibility statement

Appendix A specifies code models, training, state construction, and estimators. Appendices B–E contain code results, controls, and sensitivity analyses. Appendix F gives the theoretical assumptions and proof; Appendices G and H describe the information interventions and probes. Appendix I reports agentic methods, missing outcomes, and inference. Mettle's statistics were independently reproduced from per-continuation outcomes, without rerunning trajectories or independently verifying scorer verdicts. The accompanying supplement preserves additional exploratory analyses and expanded model tables.

Ethics statement

This work uses public benchmarks with open-weight and hosted models; it involves no human subjects or private user data. Shareable agentic outcome tables exclude model reasoning, tool transcripts, repository contents, and simulated credential-like values. A principal risk is overgeneralizing from the tested failure distributions, models, and finite budgets. We state these limitations explicitly and avoid claims about internal neural mechanisms.

AI use statement

Generative AI tools assisted hypothesis refinement, experimental design and implementation, data analysis, literature review, figure preparation, and manuscript drafting and editing. The authors are responsible for verifying numerical claims against experimental results, checking citations against primary sources, reviewing AI-assisted material, and the final content.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Caroline Choi, Zeyneb Kaya, Shirley Wu, Tengyu Ma, Tatsunori Hashimoto, and Ludwig Schmidt. Anchored self-play for code repair. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2607.03523.

Neil Chowdhury, James Aung, Chan Jun Shern, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Aljubeh, Mia Glaese, Carlos E. Jimenez, John Yang, Leyton Ho, Tejal Patwardhan, Kevin Liu, and Aleksander Madry. Introducing SWE-bench Verified. OpenAI, 2024. URL https://openai.com/index/introducing-swe-bench-verified/.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. URL https://arxiv.org/abs/2106.09685.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2.5-Coder technical report. *arXiv preprint arXiv:2409.12186*, 2024. URL https://arxiv.org/abs/2409.12186.

Takyoung Kim, Jinseok Nam, Chandrayee Basu, Xing Fan, Chengyuan Ma, Heng Ji, Gokhan Tur, and Dilek Hakkani-Tür. ReIn: Conversational error recovery with reasoning inception. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=4J3kkHI6m5.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2409.12917.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. *arXiv preprint arXiv:2305.01210*, 2023. URL https://arxiv.org/abs/2305.01210.

mini-SWE-agent contributors. mini-SWE-agent documentation. Software documentation, 2026. URL https://mini-swe-agent.com/latest/. Accessed September 11, 2026.

Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. Is self-repair a silver bullet for code generation? In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/9ddc141bdbf9d1db510cefff56c586ad-Abstract-Conference.html.

Yaorui Shi, Yuxin Chen, Siyuan Wang, Sihang Li, Hengxing Cai, Qi Gu, Xiang Wang, and An Zhang. Look back to reason forward: Revisitable memory for long-context LLM agents. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=1cymflI2Lh.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2303.11366.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021.

Vernon Toh, Navonil Majumder, Zhengyuan Liu, Nancy F. Chen, and Soujanya Poria. Scramble-toolbench: Agents search exhaustively even when their own map points to the next step. *arXiv preprint arXiv:2608.02358*, 2026. URL https://arxiv.org/abs/2608.02358.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Yuvraj Verma. Try again, don't look back: Blind resampling outperforms self-repair in small code models. *arXiv preprint arXiv:2607.26117*, 2026. URL https://arxiv.org/abs/2607.26117.

David X. Wu, Shreyas Kapur, Anant Sahai, and Stuart Russell. Synthetic error injection fails to elicit self-correction in language models. In *ICLR Workshop on I Can't Believe It's Not Better: Where Large Language Models Need to Improve*, 2026. URL https://openreview.net/forum?id=S2gMXQvgy4.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Boyu Yang, Jiazheng Sun, Zilong Lu, Zhi Qiu, Xin Peng, and Jun Zheng. MeClear: Cooperative game-theoretic attribution and risk-aware memory clearance for long-horizon LLM agents. *arXiv preprint arXiv:2609.09115*, 2026. URL https://arxiv.org/abs/2609.09115.

Ruixiang Zhang, Richard He Bai, Huangjie Zheng, Navdeep Jaitly, Ronan Collobert, and Yizhe Zhang. Embarrassingly simple self-distillation improves code generation. *arXiv preprint arXiv:2604.01193*, 2026. URL https://arxiv.org/abs/2604.01193.

Yuyu Zhang, Jing Su, Yifan Sun, Chenguang Xi, Xia Xiao, Shen Zheng, Anxiang Zhang, Kaibo Liu, Daoguang Zan, Tao Sun, et al. Seed-Coder: Let the code model curate data for itself. *arXiv preprint arXiv:2506.03524*, 2025. URL https://arxiv.org/abs/2506.03524.

Yu Zhuang, Kefei Chen, Yitong Duan, Shuxin Zheng, Jian Li, and Xu-Yao Zhang. AgentRewind: Recoverable execution for long-horizon LLM agents. *arXiv preprint arXiv:2608.14380*, 2026. URL https://arxiv.org/abs/2608.14380.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. BigCodeBench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2406.15877.

APPENDIX GUIDE

Appendices A–E give the code methods and controls, F the theoretical argument, G–H the information studies, and I the agentic experiments. Additional exploratory analyses appear in the accompanying supplement.

A Code models, training conditions, failure construction, and estimators.
B Complete training-family results, seed sensitivity, and task-only capability controls.
C Prompt-context interventions, fresh baselines, and fixed-code presentation.
D Qwen3 own-provenance and BigCodeBench replications.
E Inference budgets, training dose, and organic-family transfer.
F Optional-evidence argument and entropy counterexample.
G Tool history intervention, reference factorial, and Qwen/Gemma dose grids.
H ALFWorld design, Qwen/Gemma results, and location probes.
I Fixed-checkpoint Mettle confirmation and supporting history-curation studies.

A CODE MODELS, TRAINING, AND EVALUATION

A.1 MODELS AND TRAINING CONDITIONS

We evaluate Qwen2.5-Coder-Instruct at 7B, 14B, and 32B, Qwen3-8B in non-thinking mode, and Seed-Coder-8B-Instruct. All main adapters use LoRA rank 16, alpha 32, dropout .05, response-only cross-entropy, bfloat16, batch size 1 with four-step gradient accumulation, learning rate 2×10^{-4}, 5% warmup, and 60 optimizer steps. Target modules are the attention projections and MLP gate/up/down projections. Each condition uses training seeds 0, 1, and 2.

Clean predicts a verified solution from the task alone. Controlled and Natural Recovery additionally receive, respectively, a mutated or genuine failed program and public feedback. Policy-Induced Recovery uses failures from a recovery-trained collector. Mixed Recovery combines 21 controlled and 22 natural examples from the same matched tasks. The context controls also include three policies from an earlier 128-example study (Table 5). Direct Recovery uses mutated programs; Trajectory Recovery additionally observes a less-corrupted failed program but supervises the same final target. SSD (Zhang et al., 2026) uses outcome-blind, self-generated clean-prompt targets instead of verified solutions. These conditions are separate from the matched 43-example comparisons; Direct Recovery is never pooled with Controlled Recovery.

Table 5: Training conditions. The main matched recovery comparisons share task IDs and verified targets. Legacy conditions use a separate allocation; SSD uses self-generated targets.

Condition	Failure shown in prompt	Examples	Used in
Clean	none	43 / 35 / 128	all grids
Controlled Recovery	AST mutation of reference	43 / 35	cross-drift, transfer
Natural Recovery	frozen base-model failure	43 / 35	cross-drift, transfer
Mixed Recovery	21 controlled + 22 natural	43	7B cross-drift
Policy-Induced Recovery	frozen collector failure	35	transfer
Direct Recovery (legacy Controlled)	AST mutation of reference	128	initial study, anchoring
Trajectory Recovery	two-step mutation history	128	initial study, anchoring
SSD	none; self-generated targets	128	initial study, anchoring

A.2 MATCHED DATA AND FAILURE CONSTRUCTION

The natural-failure harvest uses the frozen Qwen2.5-Coder-7B-Instruct base model for 8,192 generations over an evaluation-disjoint training pool. Forty-three tasks produce a nonempty verifier-failing natural state. Every main condition uses these task IDs and the same verified targets: 2,269 supervised response tokens per Qwen condition and 2,425 under Seed-Coder's tokenizer. Prompt tokens differ because recovery conditions include failure context; loss is masked over the prompt. Each 43-example dataset is presented approximately 5.6 times.

Controlled evaluation states are seeded AST mutations of verified programs. Each retained state fails the full verifier; the main grid has two states at each mutation count 1, 3, and 4 for each of 50 tasks. Natural evaluation states are sampled from the frozen Qwen2.5-Coder-7B base on a disjoint 50-task pool, yielding 73 nonempty failures across 42 tasks, reused unchanged across model settings. A separate exploratory study of static distance uses 960 states from 40 tasks; its methods and results appear in the accompanying supplement.

A frozen seed-0 Controlled-Recovery collector supplies the policy-induced states. A four-sample harvest, two-state-per-task cap, and 160-state cap yield 118 states across 64 tasks. Its training harvest covers 35 of the original 43 tasks; all four policies are retrained on this intersection with the same verified targets and 1,918 response tokens per condition. Table 6 summarizes the evaluation cohorts. Own-policy state counts are state–policy pairs, not common failure sets across models.

Table 6: Core evaluation blocks; 64 samples per fixed state. For the two own-policy blocks, the state count is a count of state–policy pairs pooled across policies.

Block	Tasks	States	Purpose
Controlled drift	50	300	matched synthetic recovery
Natural drift	42	73	organic recovery
Collector-induced drift	64	118	organic-family transfer
Fresh own-policy drift	170	1,458	deployment-facing validation
Own-policy anchoring (7B/14B)	170	1,596 / 1,065	context intervention

A.3 Generation and verification

Qwen2.5 and Seed-Coder generation use temperature .7 and top-$p = .95$. Qwen3 uses temperature .7, top-$p = .8$, top-$k = 20$, and minimum-$p = 0$. The original code comparisons allow 384 response tokens and use 64 samples per state. Public execution uses a 10-second timeout, no network, and a 2,048 MB memory limit. Hidden tests are excluded from model prompts. Inference settings are common across training conditions within each model. The separate presentation study's allocation and decoding settings are given in Appendix C.6.

A.4 Recovery estimators and uncertainty

For c verifier successes among $n = 64$ independently seeded samples, we use the standard empirical pass@b estimator (Chen et al., 2021),

$$\widehat{R}_{\text{pass}}(h; b) = 1 - \frac{\binom{n-c}{b}}{\binom{n}{b}}, \qquad b \in \{1, 8, 64\}. \tag{3}$$

Pass@64 is observed any-success coverage. The sensitivity analysis uses $\hat{p} = c/n$ and $1 - (1 - \hat{p})^b$; Table 18 reports both estimators beside the budget results. The fixed-state independence assumption is not applied within a sequentially changing agent trajectory.

For task-balanced contrasts, state metrics are averaged over the three training seeds and then within task. The primary bootstrap resamples task IDs, preserving condition pairing. Seed-wise effects and a paired seed–task bootstrap appear with the training results in Appendix B.3. Bounded C_{50} is a secondary descriptive statistic based on $\hat{p}$, censored at 128. Table cell means and task-balanced effects can use different aggregation weights, so the reported contrasts need not equal differences of pooled means.

A.5 Matched recovery-interaction design

Let $t \in \{0, C, N\}$ index Clean, Controlled-Recovery, or Natural-Recovery training; let $e \in \{C, N\}$ index the failure distribution μ_e; and let g_c transform the frozen history under Standard (S), Forced-Fresh (F), or Blind (B) context. Define

$$\boxed{\mathcal{A}_{t,e,c}(b) = \mathbb{E}_{h \sim \mu_e}[R_{\theta_t}(g_c(h); b)]} \quad \text{and} \quad \Delta_{t,e,c}(b) = \mathcal{A}_{t,e,c}(b) - \mathcal{A}_{0,e,c}(b). \tag{4}$$

Paired training-family and context comparisons are

$$\begin{aligned} \Gamma_C(b) &= \Delta_{C,C,S}(b) - \Delta_{C,N,S}(b), \\ \Gamma_N(b) &= \Delta_{N,N,S}(b) - \Delta_{N,C,S}(b), \\ \Lambda_{t,e}(b) &= \mathcal{A}_{t,e,F}(b) - \mathcal{A}_{t,e,S}(b). \end{aligned} \tag{5}$$

Positive Γ_C and Γ_N indicate relative specialization. Λ measures the declared context change, including Forced Fresh's instruction and feedback changes. Figure 4 summarizes the design.

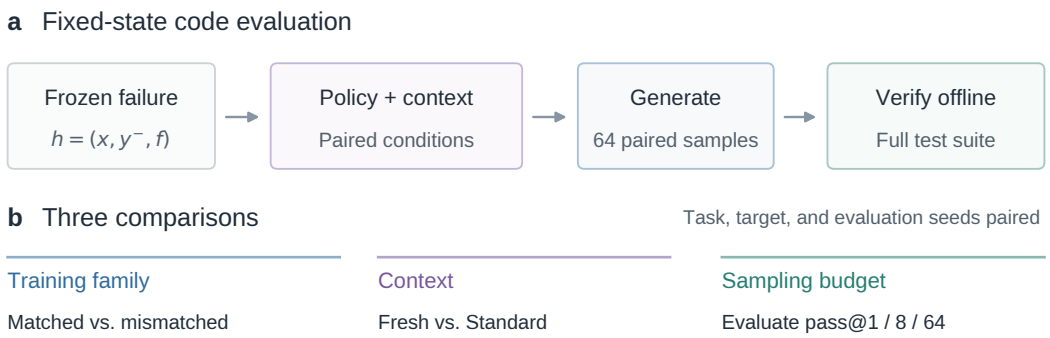


Figure 4: **Post-failure recovery as a matched interaction design.** (a) The original fixed-state code comparisons generate 64 paired samples per condition and verify them offline. (b) Matched-versus-mismatched gains over Clean measure specialization; Fresh-versus-Standard contrasts compare context policies; empirical pass@1, 8, and 64 test budget dependence.

B Recovery Training: Complete Results and Controls

B.1 Training-family specialization

Table 7 reports the matched comparisons defined in Appendix A.5. Both training-family specialization interactions replicate across the five model settings. At 7B, Controlled Recovery gains on controlled failures but not natural failures; Natural Recovery gains on natural failures and loses on controlled failures. The consistent result is a larger gain on the matched than the mismatched failure family, not an absolute improvement in every setting.

The table uses Clean under the shared repair prompt. Appendix C.4 evaluates fresh-generation baselines separately, preserving the distinction between a training effect at fixed context and a comparison across both training and context.

Mixed Recovery lies between the specialists at 7B: .504 one-shot recovery on controlled failures and .153 on natural failures (Table 8). Natural-over-Clean on natural failures is unresolved in Qwen3 and sensitive to seed inference in Seed-Coder; Appendix B.3 reports the sensitivity alongside the per-seed effects.

Table 7: Task-balanced one-shot gains over exactly matched Clean SFT and specialization interactions with 95% task-cluster bootstrap intervals, for every model setting. Each interaction subtracts a condition's gain on the other family from its gain on its own family.

Model	Training	Gain, controlled eval.	Gain, natural eval.	Interaction
7B	Controlled	+.237 [.154, .325]	−.038 [−.117, .031]	+.275 [.165, .389]
7B	Natural	−.109 [−.187, −.033]	+.144 [.062, .236]	+.253 [.141, .374]
14B	Controlled	+.198 [.119, .281]	−.021 [−.109, .065]	+.219 [.100, .342]
14B	Natural	−.163 [−.250, −.078]	+.109 [.034, .195]	+.272 [.158, .398]
32B	Controlled	+.176 [.105, .250]	−.037 [−.106, .030]	+.213 [.114, .313]
32B	Natural	−.153 [−.243, −.064]	+.093 [.026, .168]	+.246 [.132, .362]
Qwen3-8B	Controlled	+.204 [.104, .303]	−.075 [−.142, −.013]	+.279 [.158, .402]
Qwen3-8B	Natural	−.134 [−.211, −.054]	+.001 [−.045, .039]	+.135 [.047, .221]
Seed-Coder-8B	Controlled	+.226 [.136, .318]	−.062 [−.143, .007]	+.288 [.172, .409]
Seed-Coder-8B	Natural	−.163 [−.252, −.068]	+.059 [.001, .129]	+.222 [.113, .338]

B.2 Condition means

Table 8 collects the Standard-prompt means for all five model settings. These descriptive means use the original aggregation and should not be subtracted to reconstruct the task-balanced contrasts in Table 7. Appendix C compares fresh baselines, and Appendix E.1 reports budget sensitivity. The accompanying supplement retains the expanded per-model tables, including pass@8.

Table 8: Standard-prompt recovery means across three training seeds. Pass@64 is observed any-success coverage. Qwen3's original mean table reports only pass@1; its task-balanced higher-budget gains appear in Table 18.

		Controlled failures		Natural failures	
Model	Training	Pass@1	Pass@64	Pass@1	Pass@64
Qwen2.5-Coder-7B	Clean	.437	.817	.102	.265
	Controlled	.675	.887	.067	.146
	Natural	.328	.638	.246	.443
	Mixed	.504	.861	.153	.402
Qwen2.5-Coder-14B	Clean	.550	.841	.149	.301
	Controlled	.748	.894	.118	.196
	Natural	.387	.664	.274	.466
Qwen2.5-Coder-32B	Clean	.600	.880	.185	.447
	Controlled	.776	.929	.143	.320
	Natural	.447	.728	.299	.644
Qwen3-8B	Clean	.428	–	.117	–
	Controlled	.632	–	.053	–
	Natural	.294	–	.126	–
Seed-Coder-8B	Clean	.460	.797	.157	.301
	Controlled	.686	.868	.096	.192
	Natural	.298	.516	.223	.306

B.3 Training-seed sensitivity

The primary task bootstrap treats the three adapters as the realized intervention and quantifies task generalization. Table 9 reports effects separately by training seed and under a paired two-way bootstrap that independently resamples seed and task IDs. Qwen2.5 matched effects remain resolved; absolute Natural-over-Clean gains are unresolved for Qwen3 and Seed-Coder under this sensitivity analysis. All ten training-family specialization interactions remain resolved.

The two-way intervals for Controlled and Natural specialization are [.163, .393] and [.135, .380] at 7B; [.099, .347] and [.153, .395] at 14B; [.110, .320] and [.133, .363] at 32B; [.158, .402] and [.034,

Table 9: Matched one-shot gains by training seed and paired seed–task bootstrap sensitivity.

Scale	Eval./training drift	Seed 0	Seed 1	Seed 2	Two-way 95% CI
7B	Controlled	+.226	+.235	+.251	[.153, .324]
7B	Natural	+.142	+.150	+.140	[.062, .237]
14B	Controlled	+.205	+.189	+.201	[.119, .280]
14B	Natural	+.101	+.100	+.126	[.030, .200]
32B	Controlled	+.186	+.182	+.160	[.104, .250]
32B	Natural	+.088	+.084	+.107	[.020, .168]
Qwen3	Controlled	+.206	+.214	+.193	[.102, .307]
Qwen3	Natural	+.020	−.011	−.007	[−.056, .061]
Seed	Controlled	+.231	+.231	+.216	[.136, .320]
Seed	Natural	+.086	+.056	+.036	[−.003, .134]

.236] for Qwen3; and [.167, .414] and [.109, .336] for Seed-Coder. Three optimization seeds support a sensitivity check, not precise population inference over training randomness.

B.4 Task-only capability controls

Exactly paired clean solving tests whether recovery-training effects reflect a general capability shift (Table 10). The 7B Controlled condition incurs a resolved clean-solving cost; the other listed changes are unresolved.

Table 10: Exactly paired clean solving with 32 samples per task and policy.

Scale	Condition	Pass rate	Difference vs Clean [95% CI]
7B	Clean	.309	–
7B	Controlled	.231	−.077 [−.156, −.002]
7B	Natural	.334	+.026 [−.023, .071]
14B	Clean	.356	–
14B	Controlled	.384	+.028 [−.066, .126]
14B	Natural	.340	−.016 [−.076, .048]
Qwen3-8B	Clean	.225	–
Qwen3-8B	Controlled	.267	+.042 [−.039, .123]
Qwen3-8B	Natural	.226	+.001 [−.034, .036]
32B	Clean	.395	–
32B	Controlled	.407	+.011 [−.068, .093]
32B	Natural	.405	+.010 [−.035, .059]
Seed-Coder-8B	Clean	.324	–
Seed-Coder-8B	Controlled	.294	−.030 [−.112, .050]
Seed-Coder-8B	Natural	.310	−.014 [−.058, .025]

C Code Context and Presentation Controls

C.1 Prompt definitions and allocation

Standard repair Original task, previous failed code, public execution feedback, and a repair request.

Blind retry Original task and public tests only; no mention of the prior attempt.

Failure notice Original task plus a content-free statement that the previous attempt failed; failed code is hidden.

Code Only Failure notice plus the complete failed program, but no execution feedback.

Forced Fresh Failure notice plus an explicit instruction to solve again from scratch; failed code and execution feedback are hidden.

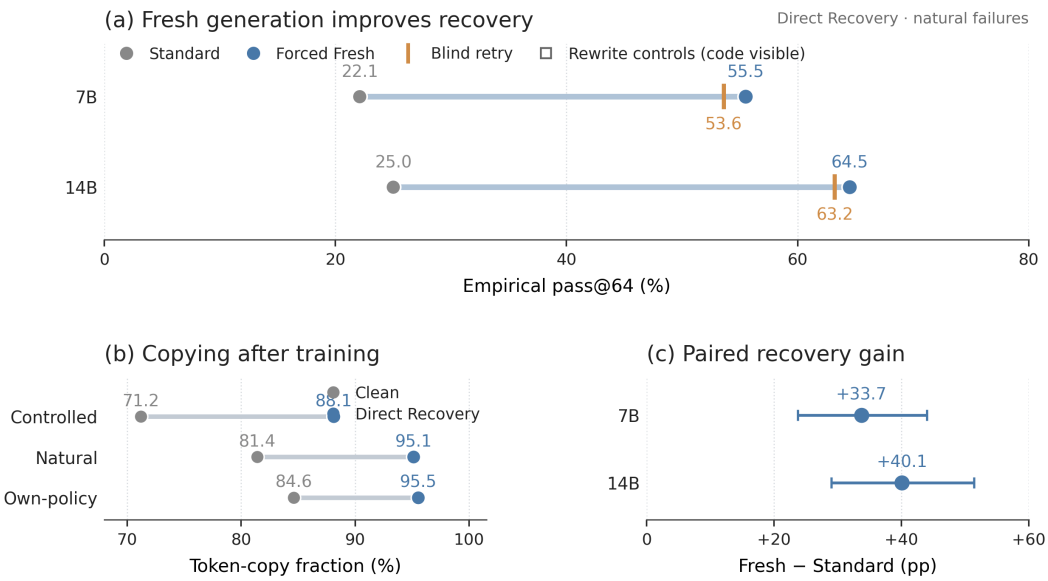


Figure 5: **Full retry-context diagnostics.** (a) Direct Recovery's success within 64 attempts under Standard, Forced Fresh, and Blind; squares show rewrite prompts that keep failed code visible. (b) Token copying under the task-only condition ("Clean" in the plot) and Direct Recovery. (c) Paired task-average Forced Fresh minus Standard effects with 95% task-bootstrap intervals. Copying is descriptive and does not identify why success changes.

Explicit Rewrite	Full failed code and feedback remain visible, with an instruction not to default to a local patch.
Diagnose–Rewrite	Full failed code remains visible; the model is asked to decide whether local repair or a new approach is warranted before returning complete code.

At 7B, the 13 legacy policies contribute 1,596 frozen own-policy state–policy pairs. The chain Blind → Failure Notice → Code Only → Standard varies failure framing, code exposure, and feedback; the rewrite prompts provide complementary controls. Each prompt variant contributes 102,144 generations across the full grid, with 64 per state–policy pair. At 14B, Standard, Blind, and Forced Fresh cover 1,065 state–policy pairs from ten policies. Prompt contrasts are paired within each policy's frozen failure set; cross-policy own-failure means are not state-paired.

C.2 Copying and fresh-generation controls

Direct Recovery (legacy Controlled, 128 examples) increases copying of the failed program: over 388,608 generations, mean token-copy fractions rise from .712 to .881, .814 to .951, and .846 to .955 on controlled, natural, and own-policy failures (Figure 5b). Nevertheless, changing the prompt exposes available capability. At 7B, Direct Recovery's pass@64 rises from .221 under Standard to .555 under Forced Fresh, compared with Blind at .536; the task-balanced effect is +.337 [.237, .440]. At 14B it rises from .250 to .645, compared with Blind at .632, with an effect of +.401 [.290, .514]. These are observed similarities to Blind, not equivalence tests.

Explicit Rewrite and Diagnose–Rewrite, which retain failed code, change Direct pass@64 by only +.010 and +.017 at 7B, with intervals crossing zero. In the sequential prompt decomposition, the code-exposure term is negative with an interval excluding zero for all 13 policies, whereas framing and feedback are resolved for 1/13 and 2/13 policies. These older variants also differ in wording.

C.3 Separating failed code from execution feedback

Raw feedback reproduces failed-code lines through traceback frames for 7% of states. Both feedback-present cells in the factorial therefore use a sanitized string containing the result header, failing public

assertion, and final exception line. Table 11 gives the complete contrasts. Feedback without code has no resolved effect in any condition, with intervals inside $[-.05, .04]$ at one and 64 samples; the feedback-by-code interactions are also unresolved. For Direct Recovery, code exposure changes recovery by $-.169$ $[-.245, -.095]$ at one sample and $-.352$ $[-.452, -.251]$ at 64. Sanitization moves the full-context mean by less than .01 at one sample in every condition.

Table 11: Failed-code $\times$ feedback factorial on the frozen 7B own-failure states for the four trained conditions (empirical pass@k; 95% task-cluster bootstrap intervals). N = notice only, C = code only, G' = sanitized feedback only, F' = code plus sanitized feedback, F = code plus raw feedback; DiD $= (F' - C) - (G' - N)$.

Condition	k	$G' - N$	$F' - C$	DiD	$C - N$	$F' - F$
Clean	1	−.002 [−.017, +.014]	−.011 [−.030, +.007]	−.010 [−.036, +.015]	−.103 [−.157, −.050]	−.001 [−.009, +.007]
Clean	64	−.008 [−.028, +.012]	+.005 [−.038, +.046]	+.013 [−.032, +.057]	−.109 [−.187, −.039]	−.002 [−.031, +.027]
SSD	1	−.011 [−.047, +.021]	+.022 [−.007, +.061]	+.034 [−.007, +.082]	−.123 [−.190, −.059]	−.006 [−.017, +.006]
SSD	64	+.001 [−.027, +.032]	−.023 [−.075, +.027]	−.025 [−.083, +.029]	−.200 [−.296, −.104]	−.018 [−.048, +.009]
Direct	1	+.005 [−.007, +.017]	+.028 [+.002, +.062]	+.023 [−.001, +.055]	−.169 [−.245, −.095]	−.009 [−.023, +.002]
Direct	64	+.011 [−.015, +.039]	+.046 [+.011, +.091]	+.035 [−.013, +.087]	−.352 [−.452, −.251]	+.005 [−.006, +.018]
Trajectory	1	+.016 [−.004, +.036]	−.004 [−.019, +.010]	−.020 [−.041, +.000]	−.153 [−.226, −.082]	−.004 [−.011, +.003]
Trajectory	64	−.005 [−.050, +.037]	+.000 [−.046, +.048]	+.005 [−.057, +.066]	−.337 [−.446, −.229]	−.032 [−.067, −.007]

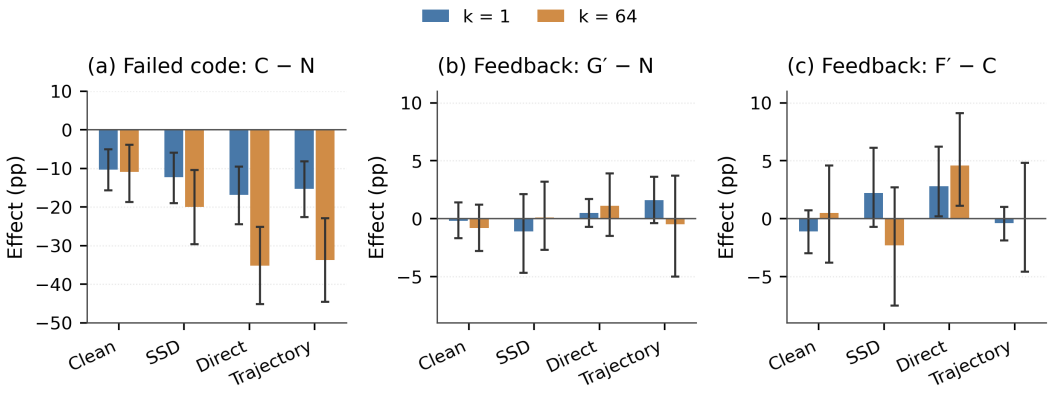


Figure 6: **Failed code and feedback have separable effects.** Frozen 7B own-failure states for four trained policies. Empirical pass@k contrasts with 95% task-cluster bootstrap intervals. N: notice only; C: code only; G': sanitized feedback only; F': code plus sanitized feedback. Panel (a) uses a wider vertical scale. Each contrast is within a policy; policy-specific failure sets do not support causal comparisons between policies.

C.4 Fresh baselines on fixed evaluation failures

Table 12 compares the recovery-trained policies under Standard with Clean under Standard, Forced Fresh, and Blind on the same failures and paired sampling seeds. At 7B on natural failures, Forced Fresh lifts Clean by +.112 [.021, .212] at one sample; Natural Recovery's gain over this baseline is +.032 $[-.033, .100]$, and +.048 $[-.040, .135]$ at 64 samples. Both are unresolved, as are the 14B and 32B fresh-baseline comparisons. On controlled mutations, hiding the program hurts Clean and Controlled Recovery. Thus fresh generation changes the baseline comparison without being uniformly preferable.

Table 12: Fixed-state context baseline at 7B (empirical pass@k; 95% task-cluster bootstrap intervals; paired sampling seeds). Top: gain of each recovery condition under the Standard prompt over Clean SFT evaluated under three contexts, at 7B for both families and at 14B and 32B for natural drift. Bottom: Forced Fresh minus Standard for each training condition on each family.

Gain of	k	vs Clean Standard	vs Clean Forced Fresh	vs Clean Blind
Controlled on controlled	1	+.237 [+.151, +.323]	+.372 [+.256, +.482]	+.386 [+.269, +.499]
Controlled on controlled	8	+.126 [+.038, +.223]	+.347 [+.230, +.469]	+.363 [+.246, +.484]
Controlled on controlled	64	+.070 [−.013, +.158]	+.323 [+.207, +.447]	+.327 [+.206, +.453]
Natural on natural	1	+.144 [+.064, +.235]	+.032 [−.033, +.100]	+.036 [−.030, +.105]
Natural on natural	8	+.198 [+.106, +.297]	+.044 [−.041, +.131]	+.057 [−.027, +.149]
Natural on natural	64	+.183 [+.091, +.282]	+.048 [−.040, +.135]	+.075 [−.020, +.175]
Natural on natural, 14B	1	+.109 [+.034, +.194]	−.041 [−.108, +.029]	−.026 [−.088, +.033]
Natural on natural, 14B	8	+.126 [+.018, +.240]	−.047 [−.129, +.034]	−.039 [−.120, +.039]
Natural on natural, 14B	64	+.135 [+.024, +.254]	−.060 [−.147, +.024]	−.040 [−.127, +.048]
Natural on natural, 32B	1	+.093 [+.026, +.166]	+.039 [−.027, +.108]	+.033 [−.030, +.096]
Natural on natural, 32B	8	+.166 [+.079, +.259]	+.044 [−.025, +.112]	+.032 [−.048, +.114]
Natural on natural, 32B	64	+.183 [+.079, +.298]	+.044 [−.004, +.099]	+.012 [−.063, +.087]
Family (FF − Standard)	**k**	**Clean**	**Controlled Recovery**	**Natural Recovery**
Controlled	1	−.135 [−.226, −.044]	−.363 [−.474, −.248]	+.003 [−.045, +.047]
Controlled	64	−.253 [−.374, −.136]	−.317 [−.447, −.193]	−.040 [−.098, +.012]
Natural	1	+.112 [+.021, +.212]	+.047 [−.031, +.120]	−.003 [−.048, +.043]
Natural	64	+.135 [+.012, +.262]	+.218 [+.067, +.373]	−.044 [−.135, +.044]
Natural, 14B	1	+.150 [+.052, +.248]	–	–
Natural, 14B	64	+.194 [+.075, +.317]	–	–
Natural, 32B	1	+.054 [−.041, +.153]	–	–
Natural, 32B	64	+.139 [+.012, +.266]	–	–

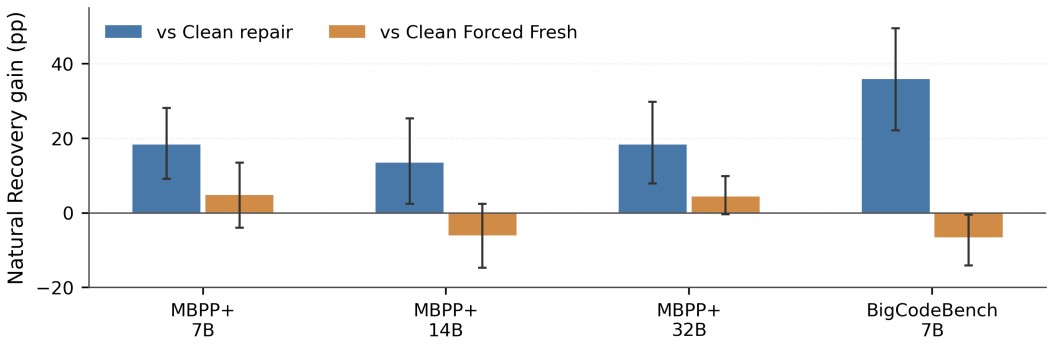


Figure 7: **The baseline changes the estimated recovery gain.** Natural Recovery under Standard minus the task-only condition under repair or Forced Fresh, at 64 attempts on fixed natural failures. Bars show reported paired effects, with 95% task-cluster bootstrap intervals. Qwen2.5-Coder models. Comparisons use each model and benchmark separately.

C.5 Context effects on fresh own-policy failures

On new failures generated by each adapted policy, Forced Fresh helps Clean and Controlled Recovery but hurts Natural and Policy-Induced Recovery (Table 13). These are paired within-policy prompt effects on different failure sets, not a causal comparison of training conditions on identical artifacts. The fixed-state comparisons above and the replication results in Appendix D address the latter question separately.

Table 13: Fresh own-policy one-shot context intervention. Means pool each training condition's own post-adaptation failures; effects are task-balanced within-policy Forced Fresh minus Standard contrasts.

Training condition	Standard	Forced Fresh	Effect [95% CI]
Clean	.115	.193	+.129 [.056, .200]
Controlled Recovery	.103	.222	+.157 [.082, .236]
Natural Recovery	.306	.222	$-.068$ [$-.116$, $-.023$]
Policy-Induced Recovery	.318	.229	$-.088$ [$-.137$, $-.042$]

C.6 Presentation of identical failed code

The context interventions in Appendix C change what the prompt contains. The presentation study holds the failed program's bytes fixed and changes only how it is labeled, with a code-omitted baseline. It tests sensitivity to the label without changing program content.

Design. The study reuses the 300 controlled and 73 natural frozen 7B states (92 distinct tasks) and the matched Qwen2.5-Coder-7B Clean, Controlled Recovery, and Natural Recovery adapters with training seeds 0, 1, and 2. Every prompt shares one system message, the task, the public tests, the failure notice "A program for this task failed evaluation. The public feedback below may cover only a subset of the evaluation tests.", the sanitized public feedback (result header, failing public assertion, and final exception line), the chat role and message order, and the instruction to return one complete program. A *Program context* block then shows the failed program under the label **Previous attempt** or **Candidate program**, or replaces the program with the label **Code omitted** and the line `[Program text omitted.]`. The two code-present variants are byte-identical apart from the label. Hidden tests, reference code, and mutation metadata never enter a prompt. Each of the 373 states, nine adapters, and three presentations receives 16 samples: 161,136 programs, all verified on the complete public and hidden suite. Sampling uses temperature .7, top-p .95, top-k 20, repetition penalty 1.1, and at most 384 new tokens; outputs reached the cap in under .03% of programs in any cell. Sampling parameters, prompts, states, adapters, and the analysis were specified before generation.

Analysis. The primary endpoint is verifier success per program. Within each failure family, samples are averaged within state, states within task, the three training seeds equally, and tasks equally. The cue effect is success with the candidate-program label minus success with the previous-attempt label. Six contrasts were prespecified: for each family, the cue effect averaged over the three policies, the Controlled Recovery cue effect minus the Clean cue effect, and the Natural Recovery cue effect minus the Clean cue effect. Intervals are 95% paired task bootstraps with 20,000 draws, reported pointwise and with a Bonferroni adjustment over the six primary contrasts. Intervals are conditional on the three observed training seeds. Unchanged-code rate (the output equals the failed program after stripping whitespace), ordered lexical token-copy fraction, completion length, cap hits, pass@8, and each policy's presentation-minus-omission contrasts are secondary and exploratory.

Results. Tables 14 and 15 report all cells and contrasts. The label effect is small and opposite in sign across families: averaged over policies, the candidate-program label lowers success by .019 on natural failures and raises it by .015 on controlled mutations, and both intervals exclude zero after Bonferroni adjustment. All four label-by-training interactions are unresolved after adjustment; the Natural Recovery minus Clean interaction on natural failures (+.032) has a pointwise interval excluding zero but an adjusted interval that does not. The label does change behavior: under the candidate-program label Clean returns the failed program unchanged in 47% of natural-failure samples versus 30%

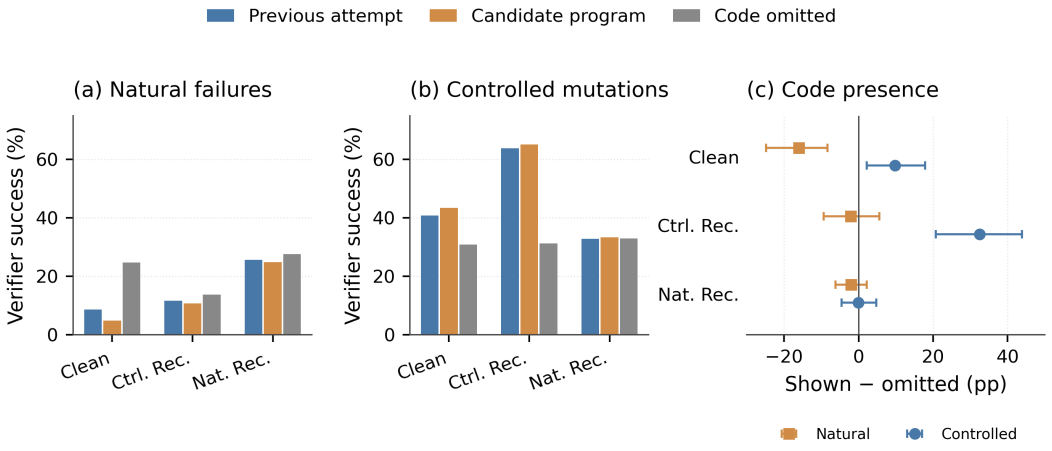


Figure 8: **Code presentation and code omission.** (a,b) Descriptive verifier-success means for three policies and three presentations. Means have no reported uncertainty. Natural failures: 73 states across 42 tasks; controlled mutations: 300 states across 50 tasks. (c) Secondary previous-attempt-minus-code-omitted contrasts, with pointwise 95% task-bootstrap intervals. Cue relabeling is a different contrast; its six adjusted primary results appear in Table 15.

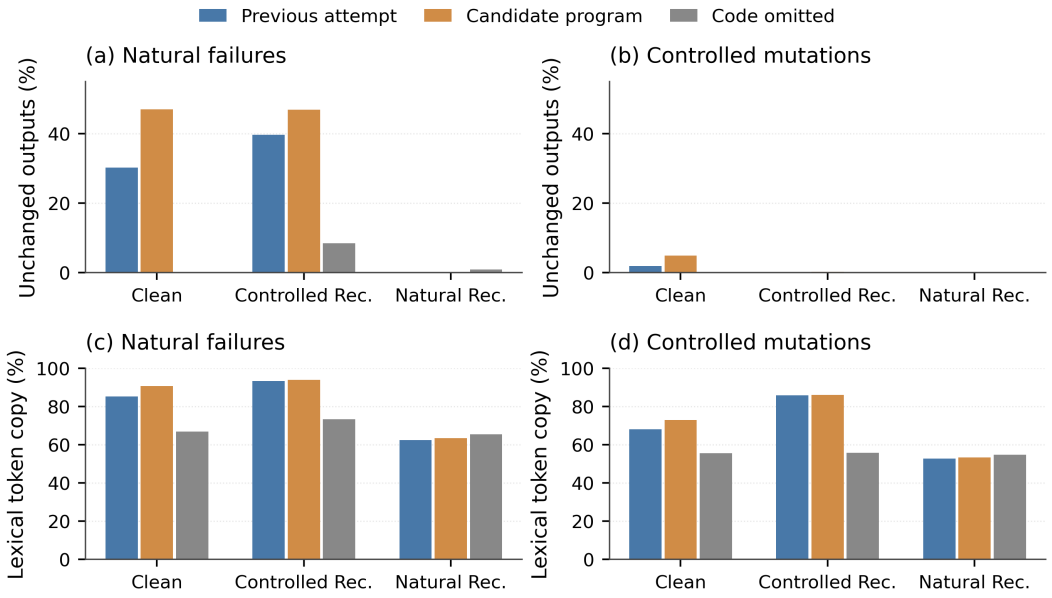


Figure 9: **Presentation effects on unchanged outputs and token copying.** Descriptive presentation-study means, using the same cells and weighting as Figure 8. Unchanged outputs equal the failed program after whitespace removal; lexical copying is the ordered token-copy fraction. The omitted condition can still match the hidden reference program and therefore need not have zero copy. No uncertainty is reported for these means; differences are descriptive.

Table 14: Presentation study cell means, equal weights over samples, states within task, training seeds, and tasks. Success is verifier pass rate; Unchanged is the fraction of outputs identical to the failed program; Copy is the lexical token-copy fraction. Natural: 73 states, 42 tasks. Controlled: 300 states, 50 tasks.

		Natural failures			Controlled mutations		
Policy	Presentation	Success	Unchanged	Copy	Success	Unchanged	Copy
Clean	Previous attempt	.087	.302	.852	.408	.019	.681
Clean	Candidate program	.048	.470	.906	.435	.049	.728
Clean	Code omitted	.247	.001	.669	.309	.000	.556
Controlled Rec.	Previous attempt	.117	.397	.933	.638	.001	.859
Controlled Rec.	Candidate program	.107	.469	.938	.651	.002	.861
Controlled Rec.	Code omitted	.138	.084	.734	.313	.000	.558
Natural Rec.	Previous attempt	.256	.000	.624	.329	.000	.527
Natural Rec.	Candidate program	.249	.000	.635	.334	.000	.534
Natural Rec.	Code omitted	.276	.009	.654	.330	.000	.548

Table 15: Presentation study contrasts on verifier success (proportion units). The six primary contrasts carry pointwise and Bonferroni-adjusted 95% task-bootstrap intervals. The presentation-minus-omission contrasts are secondary and carry pointwise intervals; they compare code shown under the previous-attempt label with code omitted.

Family	Contrast	Estimate [pointwise]	[Bonferroni]
Natural	Cue effect, policy mean	−.019 [−.037, −.005]	[−.044, −.002]
Natural	Cue: Controlled Rec. − Clean	+.029 [−.009, +.078]	[−.021, +.097]
Natural	Cue: Natural Rec. − Clean	+.032 [+.002, +.073]	[−.004, +.090]
Controlled	Cue effect, policy mean	+.015 [+.005, +.026]	[+.002, +.030]
Controlled	Cue: Controlled Rec. − Clean	−.014 [−.040, +.012]	[−.049, +.021]
Controlled	Cue: Natural Rec. − Clean	−.022 [−.052, +.008]	[−.065, +.017]
Natural	Clean: shown − omitted	−.160 [−.248, −.083]	
Natural	Controlled Rec.: shown − omitted	−.021 [−.094, +.056]	
Natural	Natural Rec.: shown − omitted	−.020 [−.062, +.022]	
Controlled	Clean: shown − omitted	+.098 [+.022, +.178]	
Controlled	Controlled Rec.: shown − omitted	+.325 [+.207, +.438]	
Controlled	Natural Rec.: shown − omitted	−.001 [−.046, +.048]	

under the previous-attempt label, so the cue raises verbatim copying more than it changes success. Seed-wise cue effects have the same sign in every training seed for Clean (natural −.034, −.043, −.040; controlled +.029, +.024, +.028) and are near zero for Natural Recovery in every seed.

Several presence-versus-omission effects are larger than the mean label effects. On natural failures, showing the code lowers Clean by .160 and .199 under the two labels, with intervals excluding zero, and has unresolved effects for Controlled and Natural Recovery. On controlled mutations, showing the code raises Clean by about .10 and Controlled Recovery by about .33, both resolved, while Natural Recovery has near-zero point estimates (+.004 and −.001), with intervals spanning zero. On natural failures its code-present minus omitted effects are −.020 and −.027, also unresolved. No unchanged programs were observed for Natural Recovery in either code-present condition, in either family; the natural-failure omitted condition nevertheless reproduces the failed program in .009 of samples. The success comparisons do not establish equivalence across presentations, and the copying results do not establish better information extraction.

Scope. The study tests one label pair on one base model with matched adapters; it does not vary chat role or authorship. Other cues may have different effects. Code omission changes syntax and length as well as content, and copying rates alone cannot identify how a policy uses information or which context it should retain.

D Code Replications: Failure Provenance and BigCodeBench

D.1 Qwen3-8B own-provenance control

The main Natural failures come from Qwen2.5-Coder-7B. A prespecified control retrains Clean and Natural Recovery on 30 tasks with Qwen3-8B's own failures and evaluates 62 own-policy states (Table 16). Natural Recovery gains +.031 [.009, .060] at one sample and +.124 [.038, .231] at 64 over Clean under Standard. Against Clean Forced Fresh, the corresponding contrasts are $-.026$ $[-.090, .028]$ and $-.000$ $[-.113, .102]$, both unresolved. Clean's own Forced Fresh lift is +.057 and +.124. Thus the baseline-context result also occurs with on-provenance supervision.

Table 16: Qwen3-8B own-provenance test (empirical pass@k; 95% task-cluster bootstrap intervals). Top: Natural minus Clean by training provenance (the completed 43-task adapters trained on Qwen2.5-Coder-7B failures versus new 30-task adapters trained on Qwen3's own failures) and evaluation provenance (the frozen 73 states versus 62 Qwen3-own states). Bottom: on the own states, the new Natural adapter under the repair prompt against the new Clean adapter under three contexts, and Clean's own Forced Fresh lift.

k	7B-trained, frozen 73	7B-trained, own states	own-trained, frozen 73	own-trained, own states
1	+.001 [−.043, +.039]	+.040 [+.006, +.084]	+.063 [−.003, +.127]	+.031 [+.009, +.060]
8	+.012 [−.042, +.069]	+.062 [−.009, +.146]	+.090 [+.015, +.166]	+.084 [+.028, +.149]
64	+.024 [−.036, +.087]	+.091 [−.011, +.199]	+.111 [+.044, +.187]	+.124 [+.038, +.231]

k	vs Clean Standard	vs Clean Forced Fresh	vs Clean Blind	Clean FF − Standard
1	+.031 [+.009, +.060]	−.026 [−.090, +.028]	−.030 [−.098, +.025]	+.057 [+.008, +.118]
8	+.084 [+.029, +.149]	−.014 [−.113, +.072]	−.024 [−.120, +.059]	+.098 [+.020, +.199]
64	+.124 [+.038, +.226]	−.000 [−.113, +.102]	−.027 [−.140, +.075]	+.124 [+.038, +.231]

D.2 BigCodeBench replication

BigCodeBench supplies a second code benchmark. Training uses 61 matched tasks and 24,579 supervised response tokens per condition; evaluation uses 192 controlled states from 32 tasks and 66 natural states from 33 tasks. The task pool is restricted to the 648 of 1,140 canonical solutions that pass environment verification. Table 17 reports the full results and paired fresh-generation comparisons.

Both training-family specialization interactions replicate. Natural Recovery's gain over Clean Standard on natural failures is +.218 at one sample and +.359 at 64. Against Clean Forced Fresh, it becomes $-.004$ $[-.066, +.059]$ and $-.066$ $[-.141, -.005]$; against Blind, $-.012$ $[-.073, +.053]$ and $-.030$ $[-.121, +.061]$. Forced Fresh improves Clean and Controlled Recovery, whereas its effect on Natural Recovery is unresolved. The natural-failure baseline result therefore extends beyond MBPP+ without implying that all recovery policies benefit from code omission.

Table 17: BigCodeBench replication (Qwen2.5-Coder-7B; empirical pass@k; 95% task-cluster bootstrap intervals). Training uses 61 matched tasks and 24,579 supervised response tokens per condition. Evaluation uses 192 controlled states (32 tasks) and 66 natural states (33 tasks), from the 648 of 1,140 tasks whose reference solutions pass verification. Top: condition means under Standard and, on natural drift, Forced Fresh and Blind. Middle: task-balanced gains over Clean, training-by-failure-family interactions, and comparisons against fresh generation using the same trained models and paired sampling seeds (64 samples per state for Forced Fresh and per task for Blind). Bottom: 32-sample clean control on 40 held-out tasks.

Eval. drift	Training	R@1	R@8	R@64
Controlled	Clean	.770	.927	.962
Controlled	Controlled	.900	.968	.969
Controlled	Natural	.460	.859	.976
Natural	Clean	.177	.364	.551
Natural	Controlled	.087	.149	.222
Natural	Natural	.395	.780	.909
Natural, Forced Fresh	Clean	.400	.869	.975
Natural, Forced Fresh	Controlled	.491	.850	.985
Natural, Forced Fresh	Natural	.391	.823	.965
Natural, Blind	Clean	.407	.842	.939
Contrast		$k = 1$	$k = 8$	$k = 64$
Controlled on controlled		+.131 [+.070, +.200]	+.041 [−.024, +.122]	+.007 [−.056, +.061]
Natural on controlled		−.310 [−.419, −.200]	−.068 [−.152, +.012]	+.014 [−.031, +.061]
Controlled on natural		−.090 [−.158, −.034]	−.214 [−.296, −.138]	−.328 [−.434, −.227]
Natural on natural		+.218 [+.095, +.338]	+.416 [+.275, +.558]	+.359 [+.222, +.495]
Controlled interaction		+.221 [+.136, +.315]	+.256 [+.153, +.366]	+.335 [+.216, +.456]
Natural interaction		+.528 [+.367, +.686]	+.484 [+.321, +.648]	+.345 [+.200, +.486]
Natural on natural vs Clean Forced Fresh		−.004 [−.066, +.059]	−.089 [−.180, −.006]	−.066 [−.141, −.005]
Natural on natural vs Clean Blind		−.012 [−.073, +.053]	−.062 [−.163, +.035]	−.030 [−.121, +.061]
Clean, Forced Fresh − Standard		+.222 [+.125, +.323]	+.505 [+.392, +.615]	+.424 [+.303, +.551]
Controlled, Forced Fresh − Standard		+.404 [+.294, +.512]	+.700 [+.596, +.799]	+.763 [+.646, +.874]
Natural, Forced Fresh − Standard		−.004 [−.068, +.056]	+.043 [−.058, +.145]	+.056 [−.005, +.126]
Clean, Blind − Standard		+.230 [+.130, +.333]	+.478 [+.362, +.595]	+.389 [+.242, +.530]
Clean control, Controlled		+.031 [−.057, +.120]		
Clean control, Natural		−.047 [−.098, −.002]		

E Code Recovery Across Budgets and Failure Families

E.1 Sampling budget and training dose

The Controlled–Clean advantage at 7B falls from +.237 at one sample to +.070 at 64 [−.012, .159]. Across 14B, 32B, Qwen3, and Seed-Coder, pass@64 gains lie between +.03 and +.07, with only the 14B interval excluding zero. Figure 10 and Table 18 report all budgets and the plug-in sensitivity check. The evidence concerns attenuation at measured budgets, not equality of unlimited-compute capability.

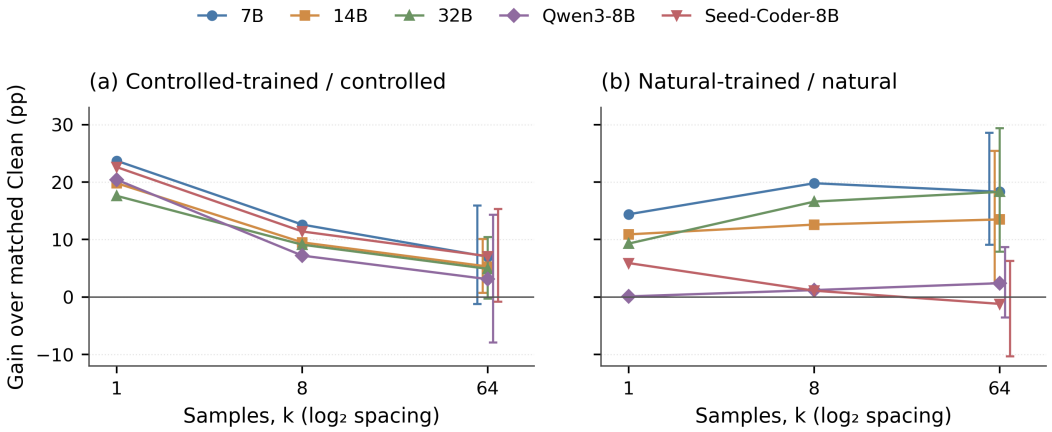


Figure 10: **Matched-training gains across sampling budgets.** Empirical pass@k gain over the matched task-only condition at k = 1, 8 and 64. Whiskers show the reported 95% task-cluster intervals at k = 64 only, with slight offsets for legibility. Table 18 supplies no intervals at the other budgets, so none are imputed. Lines connect reported points and do not represent fitted curves. 7B/14B/32B are Qwen2.5-Coder.

Natural Recovery's Standard-prompt gain persists at pass@64 in the Qwen2.5 series: at 7B it changes from +.144 to +.183 [.091, .286], while pass@64 rises from .265 to .443. Qwen3's gain is unresolved at both budgets; Seed-Coder's borderline one-shot gain is unresolved at 64. BigCodeBench also shows attenuation for Controlled Recovery and a larger high-budget Natural gain against Clean Standard (Table 17). In each case, the fresh-generation baseline remains a distinct comparison (Appendices C.4 and D).

Table 18: Matched-training gains under standard pass@k and the plug-in estimator. At $k = 1$ the estimators coincide. The last column is the 95% task-cluster bootstrap interval of the standard pass@64 gain.

		$k = 1$		$k = 8$		$k = 64$		
Scale	Eval. drift	Pass	Plug-in	Pass	Plug-in	Pass	Plug-in	95% CI
7B	Controlled	.237	.237	.126	.128	.070	.081	[−.012, .159]
7B	Natural	.144	.144	.198	.197	.183	.193	[.091, .286]
14B	Controlled	.198	.198	.095	.097	.053	.060	[.007, .101]
14B	Natural	.109	.109	.126	.126	.135	.132	[.024, .254]
32B	Controlled	.176	.176	.091	.093	.049	.056	[−.003, .104]
32B	Natural	.093	.093	.166	.164	.183	.180	[.079, .294]
Qwen3	Controlled	.204	.204	.072	.074	.031	.034	[−.079, .143]
Qwen3	Natural	.001	.001	.012	.012	.024	.023	[−.036, .087]
Seed	Controlled	.226	.226	.114	.116	.071	.074	[−.008, .153]
Seed	Natural	.059	.059	.011	.013	−.012	−.014	[−.103, .063]

Larger controlled-training dose. A dose control increases the training set to 231 mutants, or 5.4 times the examples and seven times the response tokens, with 322 steps to maintain approximately

constant presentations. Compared with a step-matched Clean control, the one-shot gain is +.248 [.155, .341], falling to +.047 [−.039, .134] at 64. The natural-failure effect at one sample is negative (−.055 [−.117, −.003]). Table 19 shows that the dose interactions remain unresolved, while the step-matched Clean control also changes natural-failure performance.

Table 19: Controlled-mutation dose control at 7B (empirical pass@k; 95% task-cluster bootstrap intervals). D_{10} = Controlled Recovery trained on 231 distinct mutants (5.4× examples, 7× response tokens) for 322 steps; C_{600} = Clean trained for the same 322 steps; K_1, C_1 = the completed 43-example, 60-step Controlled and Clean adapters. Dose effect = $(D_{10} - C_{600}) - (K_1 - C_1)$.

Eval. drift	k	$D_{10} - C_{600}$	$K_1 - C_1$	$C_{600} - C_1$	Dose effect
Controlled	1	+.248 [+.155, +.341]	+.237 [+.153, +.323]	−.000 [−.038, +.038]	+.011 [−.043, +.062]
Controlled	8	+.110 [+.018, +.206]	+.126 [+.036, +.224]	−.027 [−.061, +.012]	−.016 [−.066, +.030]
Controlled	64	+.047 [−.039, +.134]	+.070 [−.012, +.156]	−.046 [−.096, +.001]	−.023 [−.082, +.034]
Natural	1	−.055 [−.117, −.003]	−.038 [−.113, +.032]	−.005 [−.029, +.020]	−.017 [−.060, +.020]
Natural	8	−.053 [−.125, +.005]	−.063 [−.173, +.049]	−.049 [−.091, −.012]	+.009 [−.075, +.091]
Natural	64	−.044 [−.127, +.032]	−.115 [−.242, +.012]	−.103 [−.187, −.036]	+.071 [−.028, +.175]
Clean control (32 samples)	–	−.062 [−.133, +.004]	−.077 [−.156, −.003]	−.014 [−.062, +.029]	–

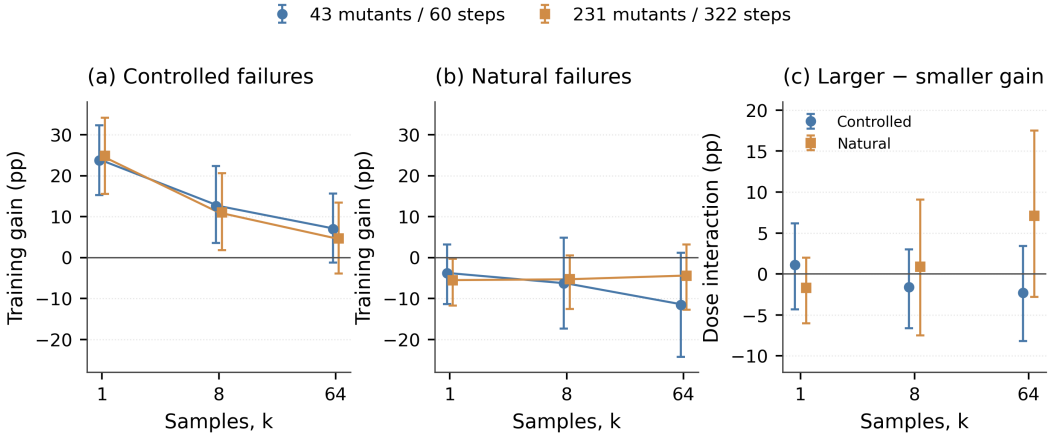


Figure 11: **Training-dose sensitivity with matched controls.** At 7B, the larger dose uses 231 mutants (5.4× examples and 7× response tokens) over 322 steps. The smaller dose uses 43 mutants over 60 steps. Gains subtract the corresponding step-matched task-only control. Panel (c) is the difference between these gains, with its directly reported 95% task-cluster bootstrap interval. All dose interactions remain unresolved. The third panel has a different vertical scale.

E.2 Transfer across organic failure families

The 35-task matched training set in Appendix A tests transfer to 118 failures from a recovery-trained collector on 64 evaluation tasks. Natural and Policy-Induced Recovery achieve one-shot means .320 and .314, compared with Clean at .119, with task-balanced gains +.208 [.136, .286] and +.203 [.128, .280]. Their difference is unresolved on collector-induced failures (−.005 [−.031, .023]) and natural failures (+.006 [−.024, .041]), despite exact failed-code overlap on only 2/35 training tasks. Both underperform Controlled Recovery on controlled mutations (Figure 12).

On the 42 collector states from 26 tasks not shared with the natural set, Natural and Policy-Induced Recovery still gain +.297 [.160, .441] and +.269 [.136, .411]. This supports transfer within an organic failure family rather than an explanation based solely on shared task identity.

On new own-policy failures, Policy-Induced Recovery reaches .318 and exceeds matched-budget Blind retry by +.119 [.029, .209]. Natural Recovery reaches .306; Clean and Controlled reach .115 and .103. Those means describe different failure sets and are not cross-policy causal contrasts. The corresponding within-policy context effects are reported in Table 13.

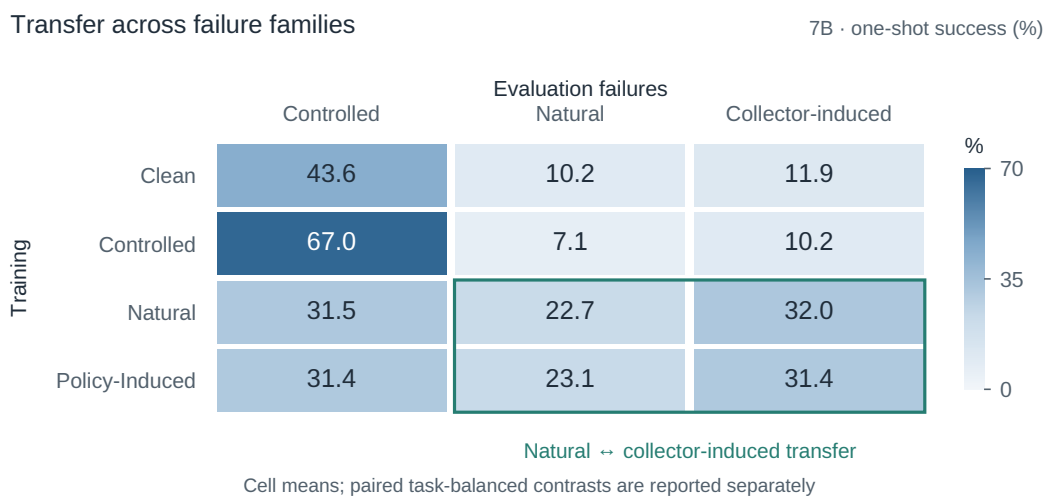


Figure 12: **Transfer follows failure-family structure at 7B.** Natural and policy-induced training transfer across independently generated organic failures, while Controlled Recovery remains specialized to controlled mutations.

F Supporting Argument: Optional Evidence and Fixed Policies

Fix a joint distribution over task-relevant latent state Z, shared observation E, and failed artifact S. Fix a success utility in $[0, 1]$ and a resource budget b. A policy maps its observations to a distribution over allowed actions or bounded trajectories. Define $J_\pi^- = \mathbb{E}[u(Z, \pi(E))]$ and $J_\pi^+ = \mathbb{E}[u(Z, \pi(E, S))]$, including policy randomness and any environment transitions in the expectation. Both experiments share the same latent-state distribution, utility, and budget.

Optional-evidence proposition. Suppose observing S is free and the plus-context policy class contains an extension of every minus-context policy that ignores S. Then $J_*^+ \geq J_*^-$, where J_*^c is the supremum over the corresponding class. For each minus-context policy π^-, its ignoring extension $\pi^+(E, S) = \pi^-(E)$ has the same utility. Taking suprema proves the inequality.

If S is conditionally independent of every utility- and transition-relevant aspect of Z given E, and the policy classes permit all relevant randomized decisions, the optima are equal: conditioning on S does not change the action-value function. This is a stipulated redundancy condition, not an assumption established by the injected-map reference. Observation costs, different computational classes, or an inability to ignore context can invalidate the premises.

For a fixed policy, let $\epsilon_\theta^c = J_*^c - J_\theta^c$ be its gap to the class optimum. Adding and subtracting the optima gives

$$D_\theta = (J_*^+ - J_*^-) - (\epsilon_\theta^+ - \epsilon_\theta^-). \tag{6}$$

The first term is ideal evidence benefit; the second is the change in the policy's optimality gap. This identity motivates separating evidence availability from successful use. Neither term is estimated, and a behavioral factorial interaction does not identify them separately.

Entropy and correctness. For a binary diagnostic whose true identity is 1, probabilities $(.4, .6)$ have entropy approximately .673 nats and correct-label log loss .916 nats. Probabilities $(.01, .99)$ have lower entropy, approximately .056 nats, but worse log loss, 4.605 nats. Thus concentration and correctness can move in opposite directions. The empirical probes report both alongside recovery; deploying a retention selector would require a separate held-out evaluation with available signals and measured decision costs.

G Tool Evidence: History, References, and Retained Facts

The ScrambleToolBench studies (Toh et al., 2026) share 80 injected-map states spanning 21 episode seeds and 18 task templates. The reference factorial and dose grids below hold the environment and surrounding history fixed while varying map retention and correct task facts.

G.1 Shared failure histories

Histories are collected with Qwen3-8B in non-thinking mode, light scramble (argument schemas shown), and no skipping. Each contains the previous task's goal, an assistant turn declaring an eight-tool map (four milestone tools and four distractors), the re-mapping of two milestone-tool names, the current goal, and a first call under a stale name with its environment response. The interventions below remove only map text. An earlier study instead removed the previous goal and the entire map-containing turn, including the previous submission; that broader intervention is reported separately in the accompanying supplement.

G.2 Map retention crossed with an independent reference

Intervention and allocation. The four-cell factorial crosses map retention/removal with a reference containing either four current milestone mappings or four valid control mappings. Removal replaces only the map text with an omission marker, preserving the previous submission and every other turn. Reference format, association count, and provenance wording are matched, although token lengths differ. Control associations are selected by an outcome-independent ordering from tools outside the old map and control commands. They can provide indirect clues, and the relevant reference can reveal task structure.

Qwen3-8B uses bfloat16, non-thinking inference, temperature .7, top-p .8, top-k 20, and at most 512 new tokens per action. Each state and condition receives 16 continuations of at most 30 further action attempts; parsing and invalid-action failures consume budget. The full grid contains 5,120 trajectories and 5,120 independent probes. An interim analysis near completion did not change prompts, sample count, stopping, or final analysis.

Analysis and results. The primary contrast is $(\text{keep} - \text{remove})_{\text{relevant}} - (\text{keep} - \text{remove})_{\text{control}}$ on success within 30 attempts. Continuations are averaged within state and states equally. A paired percentile bootstrap uses 10,000 draws over the 21 episode clusters, preserving all cells; 18-template clustering is a sensitivity analysis. Neither fully resolves crossed episode/template dependence. Shorter budgets, action errors, costs, and probes are secondary.

The four success counts are 99, 0, 866, and 950 out of 1,280 for control/keep, control/remove, relevant/keep, and relevant/remove. The interaction is -14.30 points $[-20.99, -7.66]$, with template interval $[-21.44, -6.98]$. Retention helps under control facts; its harmful effect under current mappings is individually unresolved for Qwen. Figure 13 and Table 20 show the estimates and uncertainty.

Second-model replication. The fixed candidate order uses a gate of six hash-selected states, one continuation per cell, at least 3 of 12 pooled relevant-reference successes, and no technical errors. Hermes-3-Llama-3.1-8B fails (2/12); official Llama-3.1-8B-Instruct passes (4/12) and runs the complete 5,120-continuation, 5,120-probe grid with an independent seed namespace. Llama's interaction is -10.16 points $[-16.80, -3.94]$; both individual retention effects exclude zero. Its interactions at 5, 10, and 20 attempts are -11.64, -11.33, and -10.31. Both models share the Qwen-harvested failure cohort, and results are not pooled.

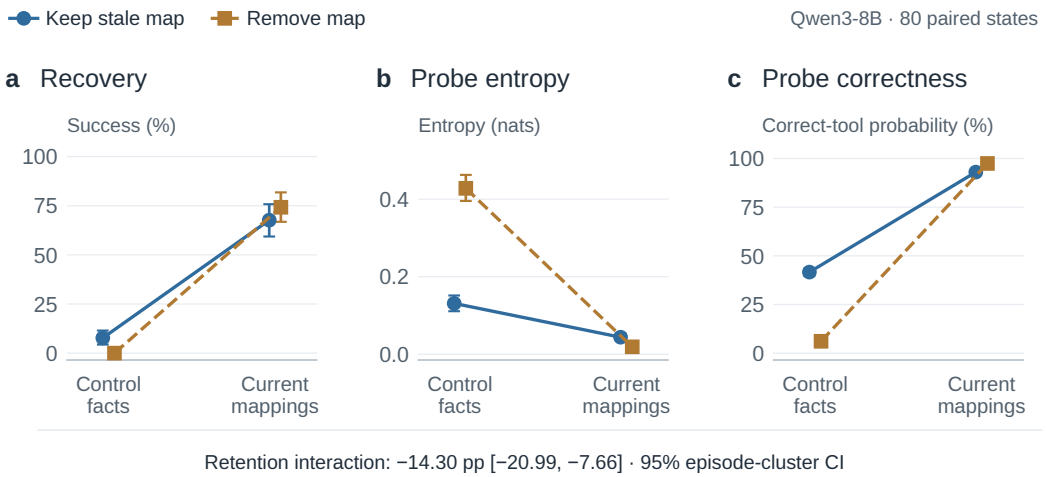


Figure 13: **Current information reduces the value of retaining a stale map.** Qwen3-8B, with all 80 states paired across four conditions. (a) Success within 30 actions. (b–c) Separate probes measure entropy and correct-tool probability over the same 28 callable identities. Whiskers are pointwise 95% episode-cluster bootstrap intervals. The change in retention benefit is resolved; the harmful effect with current mappings alone is unresolved for this model.

Table 20: Information factorial on two models, using the same 80 frozen states, intervention, sampling, allocation, and analysis, with independent continuation seeds across models. Success rates within 30 actions are percentages; retention effects and the interaction are percentage points. Brackets give 95% paired episode-cluster bootstrap intervals with equal state weights. Retention is keep minus remove; the interaction is the current-reference retention effect minus the control-reference effect.

Measure	Qwen3-8B	Llama-3.1-8B-Instruct
Control reference, map kept	7.73	1.95
Control reference, map removed	0.00	0.08
Current reference, map kept	67.66	37.66
Current reference, map removed	74.22	45.94
Retention: control reference	+7.73 [+4.37, +11.48]	+1.88 [+0.72, +3.24]
Retention: current reference	-6.56 [-13.13, +0.07]	-8.28 [-14.98, -1.79]
Interaction	-14.30 [-20.99, -7.66]	-10.16 [-16.80, -3.94]

Independent uncertainty diagnostic. Probes run in a copy of the context, replacing the system action-format suffix with a diagnostic instruction. Each question names a required operation and lists the same 28 public non-control callables. Four predeclared candidate orders are paired across conditions. The answer labels A–Z, a, and b each tokenize uniquely after a `Choice:` newline prefix. Temperature-one logits are normalized over labels. Entropy is computed before averaging over orders, operations, and states, not from an order-mixture distribution. Mean full-vocabulary choice mass exceeds 99.8% in every cell. Probe prompts and answers never enter recovery trajectories.

Table 21: Complete tool-factorial diagnostic means. Each cell averages 80 states, four operations, and four order permutations. Entropy and log loss are in nats. These are model-elicited scores, not the true entropy of possible environments.

Reference	Map	Entropy	p_{true}	Log loss
Control	kept	.131	.4165	8.625
Control	removed	.428	.0613	11.740
Relevant current	kept	.044	.9301	.342
Relevant current	removed	.019	.9745	.123

Without the map or relevant reference, entropy is .428 nats (uniform entropy is $\log 28 = 3.33$), yet correct-tool probability is 6.13%. Llama shows the same qualitative diagnostic pattern: with control facts, keeping the map lowers entropy from 2.514 to .627 nats and raises correct-label probability from .054 to .405; with current mappings both probabilities exceed .80. These probes distinguish confidence from correctness, but do not measure true world-state information gain or identify a retention rule.

G.3 Two-axis tool evidence-dose grid

Design. The dose grid uses the same 80 states, map-removal rule, reference format, probe, and analysis as the factorial. In each original eight-line map, two milestone associations are stale, two are correct, and four are correct distractors. External dose $e \in \{0, 2, 4\}$ selects how many slots of the four-line reference contain current milestone mappings; the rest retain the original control mappings. A fixed per-state hash ranking makes the doses nested, and endpoints reproduce the original references byte for byte.

Retained dose $s \in \{0, 1, 2\}$ selects how many of the two correct milestone lines remain in the map, under a second fixed ranking. Each removed line is replaced in place by a truthful association outside the map, reference, and control commands. The header, order, line count, two stale lines, distractors, failed call, and contradictory observation remain fixed. Replacement operations are not drifted, and $s = 2$ reproduces the original map. There are nine keep cells and three remove cells, since removal does not depend on s; a per-state audit verifies each fixture.

Qwen3-8B uses eight continuations per cell (7,680 trajectories) and four probe orders per operation (15,360 probes), with independent seeds from the factorial. All completed with zero technical errors. Gemma uses the same design with the allocation detailed below.

Primary contrasts. Define retention benefit $R(e, s) = K(e, s) - M(e)$, keep minus remove at fixed external dose. The two prespecified contrasts are $C_{\text{ext}} = \frac{1}{3}\sum_s [R(4, s) - R(0, s)]$, predicted negative, and $C_{\text{ret}} = \frac{1}{3}\sum_e [R(e, 2) - R(e, 0)]$, predicted positive. Each is read separately with its paired episode-cluster interval, without multiplicity adjustment; secondary intervals are pointwise. For Qwen, $C_{\text{ext}} = -9.3$ $[-15.4, -3.3]$ (template interval $[-15.6, -3.1]$) and $C_{\text{ret}} = +7.2$ $[+4.0, +10.6]$ ($[+3.9, +10.3]$). Both are supported. At 5, 10, and 20 attempts, external contrasts are -14.4, -9.6, -9.3 and retained contrasts are $+3.0$, $+5.4$, $+7.0$ points.

Table 22: Tool evidence-dose grid (Qwen3-8B, 80 frozen states, 8 continuations per cell). Success within 30 attempts in percent, retention effects $R(e, s)$ in percentage points, equal state weights; brackets are 95% paired episode-cluster bootstrap intervals (21 clusters). e = current task-relevant associations in the reference; s = still-correct task-relevant lines kept in the map; the remove cell does not depend on s.

e	remove	keep s=0	keep s=1	keep s=2	$R(e, 0)$	$R(e, 1)$	$R(e, 2)$
0	0.3	0.0	2.3	6.7	-0.3 $[-0.8, +0.0]$	$+2.0$ $[+0.2, +4.2]$	$+6.4$ $[+3.3, +10.0]$
2	3.6	4.5	9.1	17.7	$+0.9$ $[-2.3, +4.5]$	$+5.5$ $[+1.3, +9.7]$	$+14.1$ $[+7.6, +21.0]$
4	74.8	65.6	71.6	67.5	-9.2 $[-15.6, -3.6]$	-3.3 $[-10.6, +4.6]$	-7.3 $[-14.4, +0.0]$

Dose shape and secondary interactions. The retained-dose steps are resolved at $e = 0$ ($+2.3$ $[+0.6, +4.5]$, then $+4.4$ $[+0.8, +8.5]$) and $e = 2$ ($+4.5$ $[+1.0, +8.6]$, then $+8.6$ $[+3.4, +15.0]$). At $e = 4$ they are $+5.9$ $[-0.4, +13.0]$ and -4.1 $[-8.1, +0.0]$. Along the external axis, $R(2, s) - R(0, s)$ is $+1.3$, $+3.4$, and $+7.7$ $[+1.6, +14.4]$; $R(4, s) - R(2, s)$ is -10.2 $[-16.4, -4.1]$, -8.8 $[-15.7, -1.1]$, and -21.4 $[-32.3, -11.0]$ for $s = 0, 1, 2$. Thus partial reference information complements the map: two reference facts alone give 3.6% success and the original map alone 6.7%, compared with 17.7% together. The endpoint prompts reproduce the original factorial and yield an interaction of -13.7 $[-22.0, -5.5]$. The dose-by-dose interaction is -4.8 $[-11.8, +2.3]$, unresolved.

Probes and scope. Without external task facts, correct-callable probability is .037, .228, and .416 for $s = 0, 1, 2$, compared with .061 without the map. The $s = 0$ map has entropy .324 nats, below

.428 without it, despite worse probe correctness. With all current mappings, every cell exceeds .92 correct-callable probability. Qwen's low-reference cells are near the success floor. The two models share the same designer-constructed cohort, and their results are not pooled.

G.3.1 GEMMA-4-12B-IT REPLICATION

Design and allocation. Gemma-4-12B-it runs the same twelve-cell intervention on the same 80 states, using bfloat16, non-thinking inference, temperature .7, top-p .8, top-k 20, at most 512 output tokens per action, and a 30-attempt budget. Fixture histories are identical across models before their respective chat templates are applied; continuation seeds are paired across Gemma's cells in an independent namespace. The capability gate uses six predetermined states and one continuation per cell, requiring at least 6 of 24 pooled full-reference successes and zero technical errors. Gemma passes with 24 of 24.

A prespecified runtime rule selects eight continuations per cell if they fit a projected 48-hour budget, four if only that allocation fits, and otherwise the four endpoint cells. With four workers, projections of 82.2 hours for eight repeats and 41.1 for four select the full grid with four continuations per cell: 3,840 continuations and 15,360 independent probes. Completed samples were retained after an infrastructure memory interruption, and unfinished jobs resumed without changing the allocation or protocol. All cells completed. Gemma therefore has fewer repetitions per state than Qwen and uses failures collected from Qwen rather than its own policy.

Primary and secondary contrasts. Table 23 reports all Gemma cells, and Table 24 compares the two models' prespecified contrasts without pooling. Gemma supports both predictions: $C_{\text{ret}} = +7.6$ points $[+3.1, +12.0]$ and $C_{\text{ext}} = -13.5$ $[-20.2, -6.9]$. Both remain supported under task-template clustering. The two hypotheses are tested separately as prespecified, without multiplicity adjustment; secondary contrasts carry pointwise intervals. The within-run original interaction, $R(4,2) - R(0,2)$, is -20.6 $[-29.1, -12.3]$, and the secondary dose-by-dose interaction is -11.6 $[-18.0, -4.8]$.

Interpretation. With no external task facts, the original map ($s = 2$) raises success from 5.9% to 26.9%, a retention benefit of $+20.9$ points $[+13.0, +29.2]$. Even $s = 0$ helps ($+9.1$ $[+4.1, +13.8]$); this condition retains stale and distractor associations, so zero correct milestone facts is not zero possible contextual value. With the full reference, all conditions reach 96–97% success and retention estimates range from 0.0 to $+0.9$ points, with every interval including zero. The near-ceiling performance limits detectable additional gains; these estimates do not establish equivalence or complete information redundancy. The external midpoint increases retention benefit at $s = 1$, but that increase is unresolved. Thus the primary endpoint contrast replicates without a universal monotone dose response or resolved harmful retention under the full reference.

Independent probes. With no external task facts, the $s = 0$ map lowers entropy from 1.160 to 0.549 nats while lowering correct-callable probability from .078 to .051. Adding the retained correct facts raises that probability to .281 and .500 for $s = 1, 2$. With the full reference, every cell exceeds .998 correct-callable probability (Table 25). Concentration therefore remains distinct from correctness and from actual recovery. Differences in effect magnitude between models are descriptive; no model-by-intervention contrast is claimed.

Table 23: Gemma-4-12B-it tool evidence-dose grid: 80 paired states and four continuations per cell. Success is in percent and retention effects $R(e, s)$ in percentage points. Brackets are 95% paired episode-cluster intervals (21 clusters), with equal state weights. The remove cell is shared across s.

e	remove	keep s=0	keep s=1	keep s=2	$R(e,0)$	$R(e,1)$	$R(e,2)$
0	5.9	15.0	17.8	26.9	$+9.1$ $[+4.1, +13.8]$	$+11.9$ $[+4.1, +20.4]$	$+20.9$ $[+13.0, +29.2]$
2	30.3	35.9	45.9	46.6	$+5.6$ $[-1.5, +13.3]$	$+15.6$ $[+9.3, +21.9]$	$+16.2$ $[+8.5, +24.7]$
4	96.2	96.2	97.2	96.6	$+0.0$ $[-1.9, +1.6]$	$+0.9$ $[-0.9, +2.7]$	$+0.3$ $[-2.4, +2.4]$

Table 24: Prespecified tool-dose contrasts by model, in percentage points. Both hypotheses are supported under the primary episode clustering and task-template sensitivity. Results use eight continuations per cell for Qwen and four for Gemma, and are not pooled.

Model	Contrast	Estimate	Episode 95% CI	Template 95% CI
Qwen3-8B	C_{ret}	$+7.2$	$[+4.0, +10.6]$	$[+3.9, +10.3]$
Qwen3-8B	C_{ext}	-9.3	$[-15.4, -3.3]$	$[-15.6, -3.1]$
Gemma-4-12B-it	C_{ret}	$+7.6$	$[+3.1, +12.0]$	$[+3.1, +12.2]$
Gemma-4-12B-it	C_{ext}	-13.5	$[-20.2, -6.9]$	$[-21.7, -5.2]$

Table 25: Gemma tool-grid probe means: entropy in nats and probability of the correct callable. Each cell averages the 80 states, four operations, and four candidate orders. Probes are independent of the recovery trajectories.

e	Map	Entropy	p_{true}
0	removed	1.160	0.078
0	kept, $s = 0$	0.549	0.051
0	kept, $s = 1$	0.281	0.281
0	kept, $s = 2$	0.004	0.500
2	removed	0.598	0.539
2	kept, $s = 0$	0.420	0.525
2	kept, $s = 1$	0.278	0.644
2	kept, $s = 2$	0.147	0.737
4	removed	0.001	1.000
4	kept, $s = 0$	0.003	0.999
4	kept, $s = 1$	0.003	0.999
4	kept, $s = 2$	0.006	0.998

H ALFWORLD EVIDENCE-DOSE EXPERIMENTS

H.1 COHORT AND CONSTRUCTED FAILURE HISTORY

ALFWorld (Shridhar et al., 2021) is a household text environment with object locations hidden until a receptacle is visited. The stale-memory failures are constructed by fixed rules from simulator ground truth. From 474 validation games, we exclude movable-receptacle and sliced-object tasks and games not flagged solvable. Candidates are visited in a fixed per-family hash order. Eligibility requires at least two target-type instances at distinct receptacles, a non-openable receptacle without task objects, at least seven located non-task objects, 4–52 receptacles, and an inert scripted failure answered by *Nothing happens.* The first ten eligible games per family yield 60 states across six families and 37 floor plans; 61 candidates are skipped with reasons recorded. Initial observations, admissible commands, scripted observations, and object-placement facts are replay-checked before every continuation.

For each state, A is a hash-selected target object, R_{wrong} a non-openable receptacle without task objects, and B another target-type instance at a different receptacle. The history contains the task and initial observation; a six-line memory followed by a move to R_{wrong}; the arrival observation; a failed take of A; *Nothing happens.*; and an appended two-line verified reference. All bytes outside the memory and reference are fixed across cells.

H.2 EVIDENCE CELLS, GENERATION, AND PROBES

The memory always includes the false location of A. At retained dose $s = 1$, it also preserves B's true location; at $s = 0$, that slot contains a truthful non-task-object location. The remaining four lines are correct non-task facts. Remove cells replace the memory with an omission marker. At external dose $e \in \{0, 1, 2\}$, the first e slots in a fixed hash order reveal the true locations of A or B; remaining slots contain truthful non-task control facts disjoint from the memory. This gives six keep and three remove cells. A fixture audit checks counts, non-stale truth, disjointness, and fixed context.

Both models use non-thinking inference, temperature .7, top-p .8, top-k 20, 128 output tokens per action, and at most 30 actions. The system prompt requests one `Action: <command>` per turn. Missing action lines consume attempts; inadmissible commands are executed as typed. Success is simulator-verified completion. Each model runs eight continuations per cell: 4,320 continuations and 4,320 independent probe evaluations. The probe asks the location of A or B over the room's 9–52 receptacles using single-token letter labels under four orders; probes never enter recovery histories.

Gemma-4-12B-it follows a prespecified extension with the same states, fixture histories, paired reset seeds, contrasts, and reading rules. Its capability gate requires at least 5/18 pooled full-reference successes on six hash-selected states; it passes with seven and zero technical errors. Only the model and runtime library differ. Technical interruptions in the Qwen run are documented in the protocol amendment; completed results retain the same allocation and manifest.

H.3 Contrasts and results

Retention benefit is $R(e, s) = K(e, s) - M(e)$. The primary retained-dose contrast averages $R(e, 1) - R(e, 0)$ over e; the external-dose contrast averages $R(2, s) - R(0, s)$ over s. Means give states equal weight; paired floor-plan-cluster intervals are primary, with task-family sensitivity. Table 26 contains all cell means and retention effects.

Table 26: ALFWorld evidence-dose grid on two checkpoints (the same 60 constructed states, 8 continuations per cell each). Success within 30 actions in percent, retention effects in percentage points, equal state weights; brackets are 95% paired floor-plan-cluster bootstrap intervals (37 clusters). $R(e, s)$ is keep minus remove at reference dose e and memory dose s; the remove cell does not depend on s. Models are analyzed separately and never pooled.

Model	e	remove	keep, s=0	keep, s=1	$R(e, 0)$	$R(e, 1)$
Qwen3-8B	0	25.8	19.2	46.2	-6.7 [$-17.0, +3.8$]	$+20.4$ [$+8.4, +33.8$]
	1	53.8	44.6	48.3	-9.2 [$-17.6, +0.4$]	-5.4 [$-17.0, +8.0$]
	2	57.7	56.0	52.7	-1.7 [$-9.2, +6.8$]	-5.0 [$-12.5, +4.6$]
Gemma-4-12B-it	0	21.9	16.0	31.5	-5.8 [$-13.3, +1.6$]	$+9.6$ [$-1.2, +21.8$]
	1	43.8	39.0	43.8	-4.8 [$-11.1, +1.0$]	0.0 [$-6.2, +6.4$]
	2	46.5	43.3	45.8	-3.1 [$-10.1, +3.1$]	-0.6 [$-9.8, +6.9$]

Qwen3-8B. The retained-dose contrast is positive: $+9.2$ [$+4.2, +14.8$], with task-family interval [$+3.9, +14.8$]. The external-dose estimate is negative, -10.2 [$-19.8, +0.1$], but its primary floor-plan-cluster interval includes zero; the task-family interval is [$-17.1, -2.9$]. The latter averages a $+5.0$-point external-dose change for $s = 0$ and -25.4 for $s = 1$. The secondary dose-by-dose interaction is -30.4 [$-42.0, -18.0$]. The correct-fact memory's retention benefit drops from $+20.4$ at no reference facts to -5.4 at one and -5.0 at two; the latter effects are individually unresolved. Neither memory level has a monotone sequence of estimates. A supplied location may offer an alternative route to completion rather than duplicate the retained fact.

Gemma-4-12B-it. The retained-dose contrast is again positive: $+7.6$ [$+2.8, +11.8$], with task-family interval [$+2.2, +12.9$]. The external-dose estimate is -3.8 [$-16.5, +7.6$], with task-family interval [$-13.3, +4.9$]; both intervals include zero. It averages external-dose changes of -10.2 points for the correct-fact memory and $+2.7$ without that fact. The secondary interaction is -12.9 [$-25.5, -2.9$] under floor-plan clustering but unresolved under task-family clustering ([$-29.4, +1.5$]). Every individual retention interval includes zero, including the largest observed benefit, $+9.6$ [$-1.2, +21.8$]. All nine cell means are below Qwen's; model differences are descriptive, with no prespecified model-by-intervention contrast.

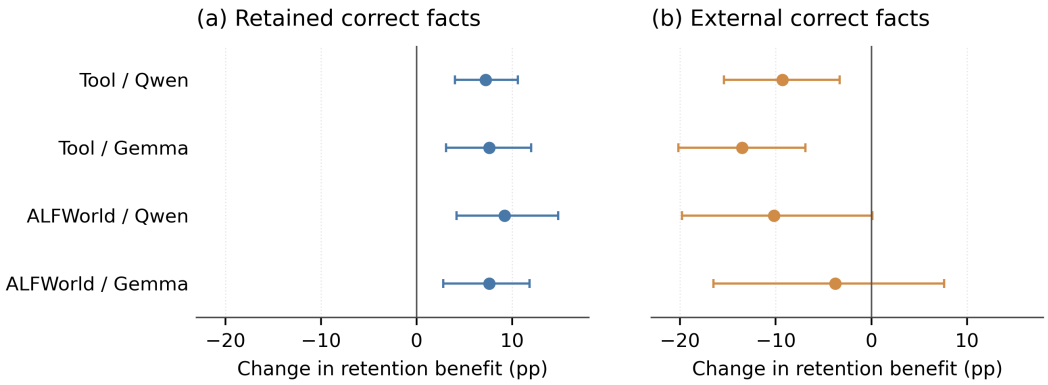


Figure 14: **Retained and external evidence have different effects.** Prespecified endpoint contrasts, averaging over the other evidence-dose axis. Intervals are 95% paired cluster bootstraps: 21 episode clusters for tool use and 37 floor-plan clusters for ALFWorld. Qwen = Qwen3-8B; Gemma = Gemma-4-12B-it. Dose ranges differ by environment, so these estimates are not pooled.

H.4 Probe diagnostics and precision

Without reference facts, Qwen's $s = 0$ memory yields entropy 0.17 nats and correct-location probability .006, compared with 0.32 and .031 without memory. Retaining the correct fact raises that probability to .48; with two reference facts every cell exceeds .99.

Gemma repeats this pattern: with no reference, the $s = 0$ memory yields entropy 0.41 nats and correct-location probability .023, compared with 0.79 and .056 without memory. The correct-fact memory yields 0.009 and .499, reflecting confident correct answers on one target and confident wrong answers on the other. With two reference facts, every cell exceeds .99 correct-location probability while task success stays at 43–46%.

An exploratory variance decomposition of Gemma's external-dose contrast gives a floor-plan sandwich standard error of 6.05 points, with a Monte Carlo component of 1.63. Quadrupling continuations on the same states projects a standard error of 5.9 points. This planning calculation suggests that additional independent states, rather than repeats alone, would be needed for substantially greater precision. Both model results remain scoped to constructed failures, a coarse two-fact reference, and the declared cluster analyses.

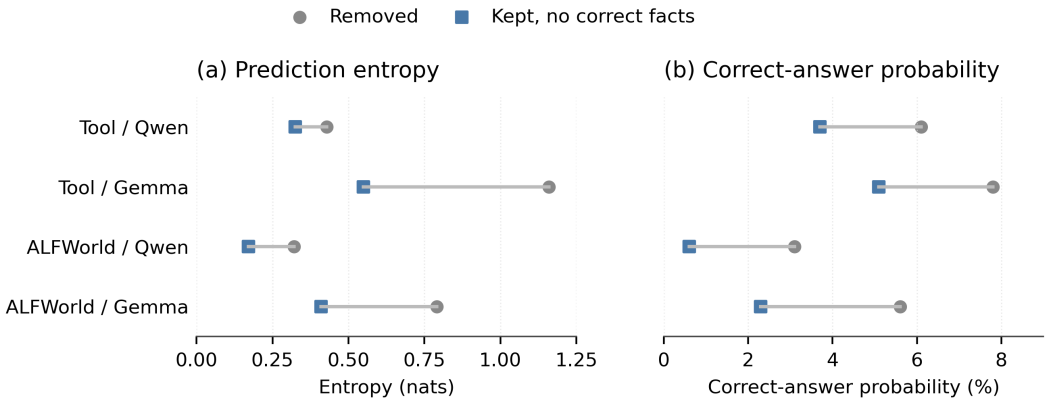


Figure 15: **Lower entropy can accompany lower correctness.** Descriptive probe means with no external task facts. Kept records contain zero correct task facts. The draft supplies no uncertainty for these means. Probes are excluded from recovery histories. Entropy is not normalized across candidate sets; comparisons are within rows. Qwen = Qwen3-8B; Gemma = Gemma-4-12B-it.

I Fixed-Checkpoint Recovery and History Curation

I.1 MettleBench confirmation: design and inference

Tasks and eligibility. MettleBench contains 82 long-horizon software tasks with deterministic, ordered acceptance checks (Zhuang et al., 2026). Before observing development outcomes, we fixed the task order and assigned the first 14 tasks to calibration, the next 18 to development, and the remaining 50 to confirmation. No tasks were substituted based on recovery outcomes. An initial trajectory could use up to 60 tool actions. A failure was eligible for recovery if it used at least 12 actions and contained a turn-aligned checkpoint with changed repository files; hidden checklist progress did not affect eligibility. Of 49 completed acquisitions, eight solved the task and seven failures were ineligible, leaving 34 eligible failures. One acquisition was abandoned because its verifier environment lacked a required data-analysis package. The acquisition completeness check therefore returned `STOP` at 49/50, although the separate prespecified minimum of 25 eligible failures was met.

Checkpoint selection and recovery context. A separate model call selected an earlier checkpoint from the eligible failed trajectory using the public assignment, available checkpoints, file-change summaries, public tool outputs, and public failure feedback. Hidden evaluator scores were not available. The call also generated a summary of potentially useful information from the abandoned continuation. We fixed both the selected checkpoint and this summary across recovery conditions. Selection could thus depend on the failed trajectory but not on subsequent recovery outcomes. Restoration covered repository files, not the complete machine or external-service state.

Fresh context contains the assignment and common recovery instructions without the previous conversation. The native-prefix condition retains the conversation through the selected model turn, including reasoning items retained by Codex. The memory condition adds the generated summary to that prefix. Both primary conditions omit conversation after the checkpoint. Contexts are not length-matched, and the comparison does not separate the value of facts from that of retained reasoning or other conversational state.

Budgets and feedback. All conditions use GPT-5.6 Luna with high reasoning effort, the same tools, a 30-action limit, and a 60,000-new-output-token interruption threshold. Acquisition uses a 100,000-output-token threshold. Each eligible failure receives three independently sampled continuations per condition; repetition indices are not matched random seeds. Conditions run in a deterministic, task-specific interleaved order. The online verifier supplies public feedback about the first unmet requirement, but not hidden checklist scores. Aggregate recovery outcomes were analyzed only after all planned allocations had finished or been censored. Equal action and output-token limits do not imply equal input-token use or total computational cost.

Table 27: MettleBench recovery allocations: 34 tasks and three repetitions per condition. Solved counts are continuation totals; the primary estimator weights tasks equally.

Context	Planned	Observed	Censored	Solved
Fresh task conversation	102	100	2	12
Native pre-checkpoint conversation	102	98	4	35
Native prefix + generated memory	102	96	6	40
Total	306	294	12	87

Estimation and uncertainty. For task i, condition a, and observed repetition r, let Y_{iar} indicate recovery success. With n_{ia} observed repetitions, define

$$\bar{Y}_{ia} = \frac{1}{n_{ia}} \sum_{r \in \mathcal{O}_{ia}} Y_{iar}, \qquad \widehat{\Delta}_{a-b} = \frac{1}{34} \sum_{i=1}^{34} (\bar{Y}_{ia} - \bar{Y}_{ib}). \tag{7}$$

Every eligible task has observed outcomes in all three conditions. Native prefix minus fresh context is the primary contrast; memory minus native prefix is secondary. We use 10,000 task-bootstrap resamples for percentile 95% intervals and two-sided paired sign-flip tests over task differences,

computed exactly by dynamic programming. Repetitions are averaged within task before inference, so the 294 continuations are not treated as independent tasks. The primary contrast has 13 positive, two negative, and 19 zero task differences; the memory contrast has four positive, two negative, and 28 zero differences. The secondary checklist-progress effect of native prefix versus fresh context is +27.7 percentage points [17.7, 38.4].

Table 28: Recovery from the same selected repository file checkpoints on 34 failed MettleBench tasks using GPT-5.6 Luna and a 30-action budget. Condition means first average observed continuations within each task, then weight tasks equally. Paired effects and intervals use tasks, with 95% task-bootstrap intervals. Of 306 planned continuations, 294 are observed and 12 are censored. The secondary memory contrast neither establishes a benefit nor satisfies the prespecified ± 10 pp equivalence criterion. Extreme-assignment missing-outcome bounds are separate from these confidence intervals.

Recovery context	Observed / planned	Success (%) [95% CI]
Fresh conversation	100/102	11.8 [4.9, 19.6]
Native prefix	98/102	35.8 [21.1, 51.0]
Native prefix + memory	96/102	40.2 [25.5, 55.9]

Paired contrast	Role	Effect (pp) [95% CI]
Native prefix - fresh	Primary	+24.0 [11.8, 37.3]
Memory - native prefix	Secondary	+4.4 [−1.5, 11.8]

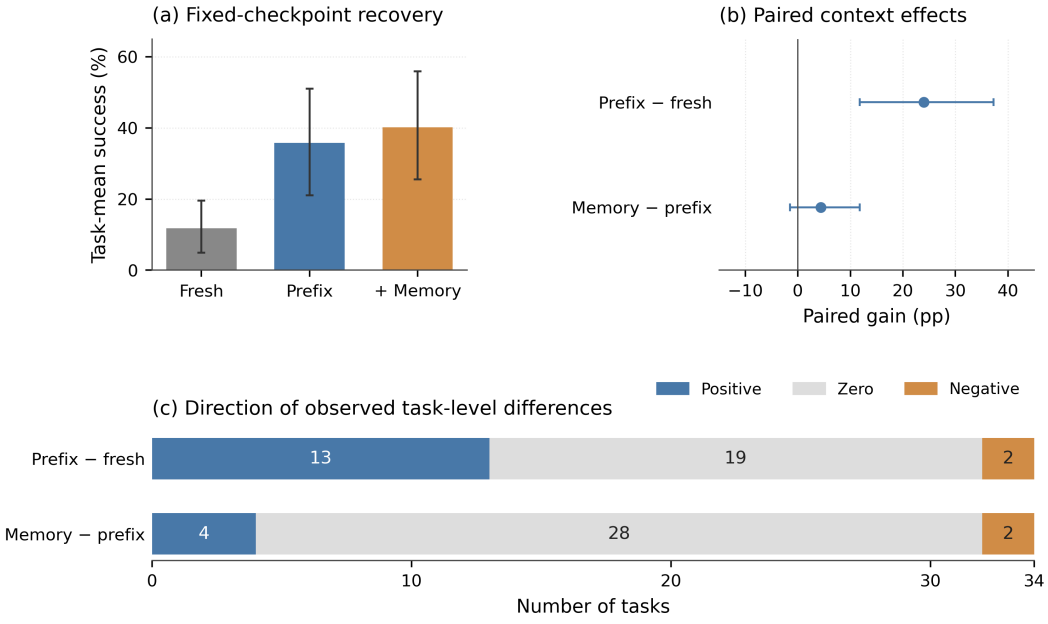


Figure 16: **Conversation history at a fixed repository checkpoint.** GPT-5.6 Luna on 34 failed MettleBench tasks at 30 actions. Means weight tasks equally; whiskers are 95% task-bootstrap intervals. Panel (c) counts the signs of observed within-task differences, rather than classifying statistical significance. The memory contrast is unresolved and fails the prespecified ± 10 pp equivalence criterion. There are 294 observed continuations out of 306 planned.

The prespecified equivalence test for memory requires the entire paired 95% interval to lie within $[-10, +10]$ percentage points. The observed interval $[-1.5, 11.8]$ fails this criterion; the nonsignificant sign-flip test does not establish equivalence.

Missing outcomes. Twelve allocations (3.9%) had no trustworthy persisted outcome and were censored without replay: six memory, four prefix, and two fresh-context allocations. One failed before generation and one was terminated with generation status uncertain; the other ten termination causes are unknown. These decisions preceded aggregate-outcome unblinding. The model, prompts, checkpoint, task inclusion, treatment allocation, and scoring rule remained unchanged.

The prespecified estimator excludes censored repetitions. For sensitivity, we assign all missing outcomes in one condition to failure and those in the other to success, then reverse the assignments, using all three planned repetitions per task. This bounds prefix minus fresh at [20.6, 26.5] points and memory minus prefix at [1.0, 10.8]. These are empirical extreme-assignment bounds, not confidence intervals or evidence of a population-level memory gain. The incomplete acquisition is separate from these 12 recovery allocations.

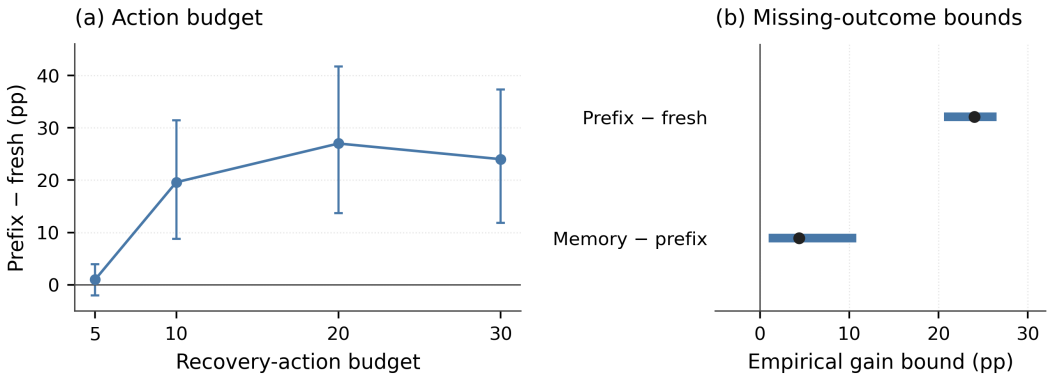


Figure 17: **Budget and missing-outcome sensitivity.** (a) Prefix-minus-fresh effects with 95% task-bootstrap intervals. Budgets share the same runs; only the 30-action endpoint is primary. (b) Extreme assignments of the 12 missing outcomes give empirical bounds [20.6, 26.5] and [1.0, 10.8] pp. These segments are NOT confidence intervals; dots are the primary observed-data estimates. The bounds do not establish a population-level memory effect.

Action budget. Table 29 reports within-run diagnostics alongside the prespecified 30-action comparison. These are not independent confirmatory tests. The retention advantage is small at five actions and becomes apparent by ten.

Table 29: Native prefix minus fresh context by recovery-action budget. Effects and intervals are in percentage points. The 30-action contrast is primary.

Actions	Effect	Task-bootstrap 95% interval	Exact p
5	1.0	[−2.0, 3.9]	1.0000
10	19.6	[8.8, 31.4]	0.0020
20	27.0	[13.7, 41.7]	0.0010
30	24.0	[11.8, 37.3]	0.00134

Reproduction limits. Independent analysis of the released acquisition and continuation outcomes reproduced the condition means, task-bootstrap intervals, exact paired tests, budget results, and missing-outcome bounds, and verified all 306 unique task–condition–repetition allocations. The release did not include the original complete analysis record or execution manifest. This check therefore verifies outcome-level statistics, not execution provenance, model trajectories, workspace restoration, or original scorer verdicts. The earlier development study is not pooled with the confirmation.

I.2 Supporting history-curation studies

We also compared selected, compressed, and recent histories with full history. These smaller studies condition on reaching their respective intervention and do not replicate Mettle's native-prefix versus fresh-context comparison. Table 30 summarizes task resolution.

Repository tasks. GLM-4.7-Flash used mini-SWE-agent on SWE-bench Verified (mini-SWE-agent contributors, 2026; Chowdhury et al., 2024). At action 12, we compared full history, model-selected turns, recent turns, random turns, shuffled history, and full history with selected turns replayed, at the same repository state. Recent and random histories matched the selected history's token count within 5%. Of 25 attempted tasks and 50 assigned repetitions, 20 tasks and 34 repetitions reached the intervention. Selected and recent histories coincided in 14 repetitions and shared the same outcome rather than being rerun independently. Limits were 70 actions, 98,304 output tokens, and 4,096 tokens per call.

Qwen3.8-27B compared recent turns, a summary plus recent turns, model-curated replay, and full history when context pressure arose: eight tasks, 16 repetitions, and 64 continuations. We then added an evidence-ledger condition, using new repetition seeds on the same task allocation. Seven tasks and 13 repetitions reached this intervention (65 continuations); three repetitions finished before exposure. We analyze the studies separately. Both allow 90 actions, 98,304 output tokens including the prefix, and 16,384 tokens per call. Bounded histories allow 12,000 tokens excluding the latest turn and anchors; full history allows 60,000. Output limits ended 43/64 and 37/65 continuations, respectively. Results from earlier versions with context-accounting and truncation faults are excluded.

AppWorld. Nemotron-3-Nano-30B-A3B in BF16 compared six context policies on eight development recovery states, with two repetitions each (96 continuations): recent turns, summary plus recent turns, model curation, full history, an exact code/output ledger, and full history with prior reasoning re-sent (Trivedi et al., 2024). Only the last condition re-sends reasoning. Earlier pilot tasks are excluded, but scenario families are not held out. Histories allow 12,000 tokens for bounded policies and 120,000 otherwise; continuations allow 80 actions, 262,144 output tokens, and 16,384 tokens per call. Curation was invoked in only two of its 16 continuations.

Table 30: Task- or state-mean success (%) in the history-curation studies. Qwen rows use separate samples and are not pooled. AppWorld is a development study. R denotes re-sent reasoning; dashes denote absent conditions. Equal means do not establish equivalence.

Study (scored units)	Recent	Summary	Selected	Full	Ledger	Full + R
GLM / SWE (20)	32.50	–	27.50	37.50	–	–
Qwen / SWE (8)	31.25	25.00	31.25	31.25	–	–
Qwen / SWE + ledger (7)	42.86	35.71	35.71	42.86	35.71	–
Nemotron / AppWorld (8)	12.50	6.25	0.00	0.00	6.25	6.25

Results and limitations. For GLM, selected minus recent is -5.0 points $[-17.5, 5.0]$ and selected minus full is -10.0 $[-25.0, 2.5]$. For Qwen, both contrasts are zero at every task average in the first study. With the ledger added, selected minus recent/full and ledger minus full are -7.1 $[-21.4, 0.0]$. AppWorld ledger minus full is +6.25 [0.00, 18.75]; ledger minus full with reasoning is zero at every state average, not every repetition. Its secondary requirement-fraction contrast favors the ledger over full history by +2.6 [0.8, 5.2]. Recency's two successes occur on different states, one repetition each.

We find no reliable task-resolution advantage from model curation, but the small, exposure-conditioned samples, coincident controls, output limits, and infrequent curation calls preclude an equivalence conclusion. Shorter retained context also does not establish lower total cost once curation calls are included; the Qwen ledger study does not demonstrate equal-success efficiency.