HARNESS^2: EVALUATING CODING-AGENT MEMORY AFTER A RESET

Anonymous authors
Paper under double-blind review

ABSTRACT

A coding agent receives a stored code change to help repair its next program. The resulting recovery score is taken as a verdict on what the memory transfers. The harness can settle that verdict before a program reaches the verifier. On two held-out functions at a 16,384-token per-call cap, 19 of 24 episodes at extra-high reasoning effort emit no program; forced closure prevents an observed end-token failure mode on Qwen3.8-27B-FP8. We introduce Harness^2 (Harness of Harnesses), which starts from a failing program, excludes prior interactions and the target solution, and compares reference-derived banks with the task, seed, model, verifier, and execution limits held fixed. On 60 completed PACE-Bench task–seed units, a sibling reference change resolves 23 episodes versus 22 with no bank: 12 paired wins, 11 losses, and 37 ties, a difference of $1/60$. A character-matched unrelated bank resolves 21, while direct patch application and whole-program transplant both fail on the tested cross-task transfers. Recovery is reported beside the output tokens consumed by the same evaluated episodes, excluding earlier executions that were replaced: the bank is evaluated through the harness that reads it.

1 INTRODUCTION

A coding agent that repairs one program can retain material for a later task. The next decision is what to carry forward. Existing systems store executable skills, trajectory-derived lessons, workflows, and adaptive notes (Wang et al., 2023; Zhao et al., 2024; Wang et al., 2025; Suzgun et al., 2026); other work organizes repository experience or extracts transferable skills from code (Gao et al., 2026; Feng et al., 2026). Evaluations ask whether that material improves a fresh coding episode, including what reaches the solver and how it is used (Fan et al., 2026a; Chen et al., 2026). A higher recovery rate offers an appealing verdict: the stored content transfers. Reading that score as a property of the memory itself makes the execution conditions disappear.

Measuring that contribution requires specifying the harness through which the agent reaches a task. The harness sets each call's output budget, handles reasoning-channel endings, limits attempts, and returns verifier feedback. A bank changes the prompt; under a fixed output cap, that change can alter how much generation remains for code after reasoning. Harness studies show that planning, action, and context choices interact with model and budget (Fan et al., 2026b; Yang et al., 2026), while work on agent working memory distinguishes stored state from delivered context and outcome (Chen et al., 2026; Saha, 2026). Reasoning-budget studies further motivate inspecting when deliberation displaces action (Cuadron et al., 2025; Kim et al., 2026). A final success rate alone cannot show whether stored content helped or whether the interface prevented code from reaching the verifier. A single acceptance rate collapses these different failures into the same zero, hiding where the episode was constrained.

We measure the harness before we measure the memory. On two held-out functions, 19 of 24 high-effort episodes exhaust a 16,384-token cap without emitting a program; at 49,152 tokens, every returned episode on the same seed groups emits one. An end token inside the reasoning channel supplies a second route to the same zero. Across 30 planned pairs on the 27B backbone, replacing that token with a closing tag yields three completed pairs favoring closure and none favoring closure off; accounting for the two incomplete pairs leaves a full-panel difference between 0.10 and 0.17. On Qwen3.5-4B, all 30 paired outcomes tie. The same termination rule meets two different models;

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

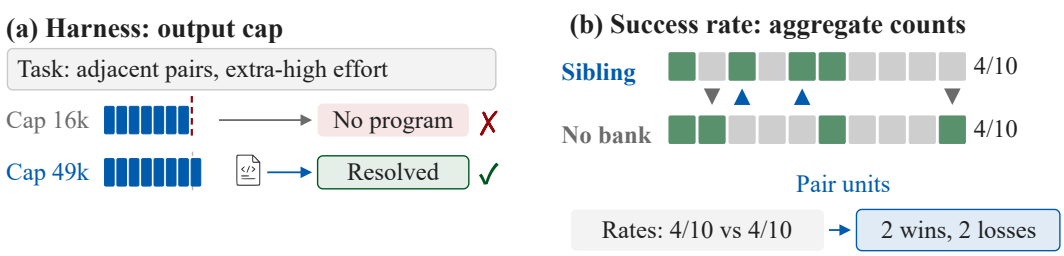

Figure 1: Limits of a recovery rate. (a) An output cap can prevent program production. (b) Equal recovery totals can conceal different paired outcomes.

its effect is measured where it acts. Even after code reaches the verifier, equal recovery totals can conceal wins and losses on different task–seed units.

Harness^2 makes the bank comparison follow those measurements. A *reset* starts from a program that fails the target test suite and excludes the target solution, prior agent interactions, and earlier verifier reports. For PACE-Bench adaptation pairs (Zhan et al., 2026), we take the change from a sibling task's released initial program to its solved program and place it in the prompt, with the target's own reference change excluded. A character-matched bank from unrelated families and an empty bank supply the controls. We fix this content before generation, then run matched task–seed units with the same model, verifier, output budget, attempt limit, and reasoning-channel rule, so the comparison varies the supplied bank. Verifier acceptance within these limits is *recovery*.

All 60 memory-comparison units returned. The sibling bank resolves 23 episodes, the empty bank 22, and the unrelated bank 21; the sibling-over-empty difference of $1/60$ comprises 12 wins, 11 losses, and 37 ties. Neither direct patch application nor whole-program transplant succeeds in the tested cross-task checks. With 49,152 tokens per call and up to five calls, mean episode output is 73.0k tokens for the sibling arm and 73.8k for no bank, averaged over their 60 evaluated episodes and excluding earlier executions that were replaced. The comparison puts the recovery difference beside each arm's episode-level output.

Our contributions are summarized as follows.

- We present Harness^2, a reset protocol that compares specified memory banks while holding the model, verifier, and execution settings fixed across task–seed units.
- We measure how a per-call output cap and reasoning-channel closure affect whether a program reaches the verifier, and bound the closure effect on recovery over its planned panel.
- We compare sibling reference content with empty and character-matched unrelated banks on 60 completed PACE-Bench units, reporting paired recovery differences alongside direct patch and transplant checks.

2 Related Work

Memory transfer. Agent memories retain executable skills, trajectory lessons, workflows, and notes for later tasks (Wang et al., 2023; Zhao et al., 2024; Wang et al., 2025; Suzgun et al., 2026). Coding studies evaluate repository experience and cross-task transfer, including irrelevant-memory controls and held-out solvers (Gao et al., 2026; Feng et al., 2026; Fan et al., 2026a). Harness^2 fixes reference-derived banks before generation and varies their content under matched execution, separating the bank comparison from memory writing.

Harness and context. Harness design governs planning, actions, and context delivery, with measured effects that depend on model and budget (Fan et al., 2026b; Yang et al., 2026; Lee et al., 2026). Working-memory studies distinguish stored information from the context delivered to a coding agent (Chen et al., 2026; Saha, 2026). We follow that context through generation: measure output truncation and reasoning-channel termination, then hold the harness fixed for the bank contrast.

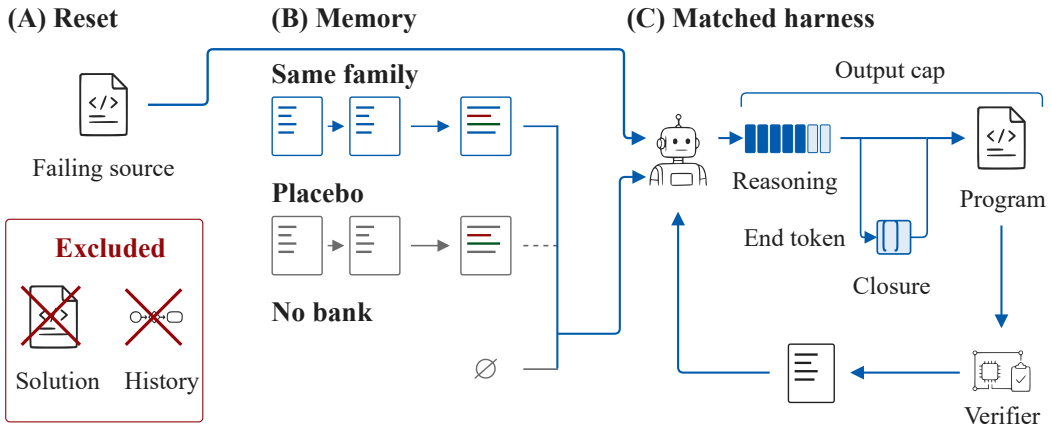

Figure 2: Reset-based evaluation. (A) Reset, (B) Memory, and (C) Matched harness. All banks share the generation limits and verifier.

Search and generation. Program evolution carries evaluated programs into later edits (Novikov et al., 2025; Sharma, 2025; Lange et al., 2026; Assumpção et al., 2025), while agent evolution changes the coding agent itself (Zhang et al., 2026). Test-time-compute studies examine how deliberation and budget allocation affect agentic coding (Cuadron et al., 2025; Kim et al., 2026). Our search-controller comparisons evaluate complete procedures on reset tasks; the bank contrast holds the procedure fixed and changes the supplied content.

3 Method

Harness[2] compares supplied banks on matched reset episodes (Figure 2). Each arm starts from the permitted source program; the target, seed, model, and execution limits are fixed within a comparison. Recovery means verifier acceptance under those limits. Each task–seed unit supplies one outcome per arm, so bank effects are compared within units before aggregation.

3.1 Reset Episodes and Memory Banks

Reset episodes. Our PACE-Bench pairs use the Initial environment as the source and its Stage-1 mutation as the target, where the source controller fails (Zhan et al., 2026). An episode starts from the pair's source-stage program; the target-stage verifier decides acceptance after each submitted program. Generation and verifier calls are bounded. An *open channel* is a generation call that ends inside the reasoning channel without returning a program, either at an end token or at the output cap. Such an ending does not imply that the episode submitted no earlier program. We call an episode resolved when the verifier accepts a program, and returned when a terminal agent outcome is available. Every returned episode enters the recovery-rate denominator; controller-process failures remain missing.

Harness rule. A protocol bounds each episode by its generation calls, verifier calls, and output tokens, and every arm of the reset contrast runs under one protocol; actual token use and call counts vary by episode. A reasoning model can emit its end token inside the reasoning channel, and the call then returns no program. *Forced closure* replaces that end token with `</think>` and continues generation for a bounded number of tokens inside the same output budget, with the combined output counted as one generation call and all tokens charged to the episode budget. The closure-off arm runs the same task–seed unit with this rule removed, sharing the model, prompt, and output budgets.

Memory banks. The reset excludes the target reference program, prior interactions, and earlier verifier reports. Matched arms differ only in the supplied bank. Each target pair has sibling tasks in the same physics family. The *same-family* arm places in the prompt one reference transition, the unified diff from a sibling task's official initial program to its Stage-1 reference program, com-

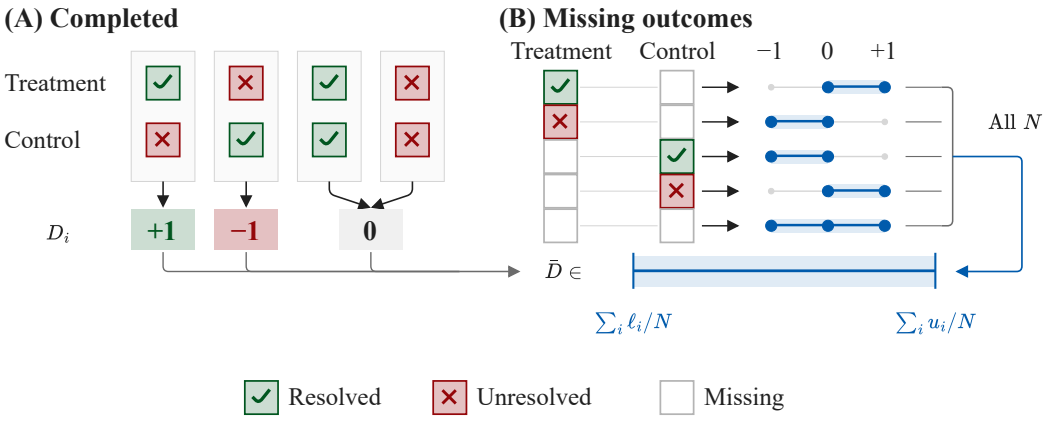

Figure 3: Paired outcomes and incomplete units. (A) Completed outcomes yield a win, loss, or tie. (B) Each missing binary outcome defines a feasible interval; averaging the unit endpoints over all planned units yields the sharp finite-panel bound.

puted from the released references with the pair's own task excluded. This is the sibling bank used throughout the results. The *placebo* arm places diffs from other physics families cut to the same character count, and the *no-bank* arm places nothing, so the prompt carries the task and the harness rule alone. Character matching controls input length in characters, and the tokenizer can produce different token counts. Every arm uses forced closure with the same pair, seed, model, and output budgets. The oracle-bank calibration supplies the target's own diff and resolves 23 of 30 episodes; it is reported separately from the cross-task memory arms.

3.2 Paired Evaluation

Figure 3 illustrates the paired differences and how missing outcomes contribute to the full-panel bound. Each task–seed unit produces a binary recovery outcome under each assigned arm, and a paired difference records whether the intervention changes acceptance within the fixed budget (Kotawala, 2026). Runs execute in parallel on shared hardware, where generation length and resource scheduling affect return order. At the analysis cutoff, completed pairs describe the available outcomes, and the full-panel bound covers all planned units with each missing binary outcome free to take either value.

Definition 1 (Paired value)**.** *Units $i = 1, \ldots, N$ are the fixed task–seed combinations in the evaluation. For a treatment arm and a control arm, $D_i \in \{-1, 0, 1\}$ is the treatment recovery indicator minus the control recovery indicator on unit i.*

Let C_t contain the units whose two arms have returned by calendar time t, with $D_i = 1$ for a win, -1 for a loss, and 0 for a tie. The completed-pair mean $\bar{D}_t = |C_t|^{-1} \sum_{i \in C_t} D_i$ is the difference in recovery rates on these shared units and uses every completed pair, ties included.

Full-panel bound. The panel mean $\bar{D} = N^{-1} \sum_{i=1}^{N} D_i$ covers all planned units. Let Y_i^T and Y_i^C be their treatment and control resolve indicators, with a returned arm fixing its indicator and a missing arm free to take either binary value. Completed units then have $\ell_i = u_i = D_i$, a unit with just the treatment returned has $[\ell_i, u_i] = [Y_i^T - 1, Y_i^T]$, one with just the control returned has $[-Y_i^C, 1 - Y_i^C]$, and a unit with neither returned spans $[-1, 1]$. Averaging the feasible endpoints gives

$$\bar{D} \in B_t = \left[\tfrac{1}{N} \textstyle\sum_{i=1}^{N} \ell_i,\ \tfrac{1}{N} \sum_{i=1}^{N} u_i \right]. \tag{1}$$

Proposition 1 establishes that this bound is sharp and needs no model of which episodes return. Its width is $(S_t + 2U_t)/N$ for S_t units with one returned arm and U_t with none, so every returned episode narrows it, and once it excludes zero the sign of the difference is fixed for the evaluated panel.

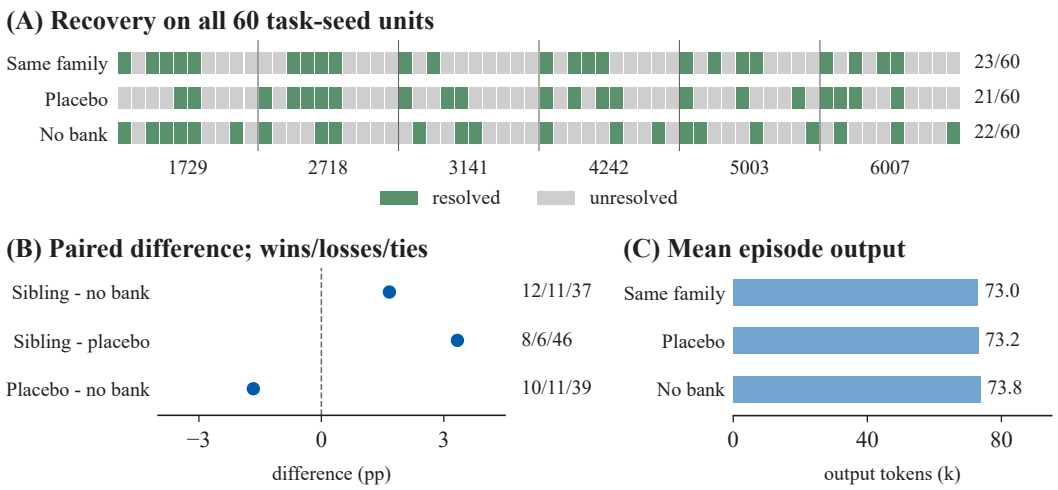

Figure 4: Completed memory panel. (A) Recovery on all 60 task–seed units; (B) Paired difference in percentage points (pp); (C) Mean episode output. Differences are descriptive, not confidence intervals.

Sign-flip reference. For inferential context, let W and L count the wins and losses among completed pairs. Under a joint law of outcomes and return times invariant to swapping the arms of any unit, the discordant signs are independent fair coins, and the exact two-sided sign-flip value is p_t in Equation 3 (Proposition 2, Appendix A.2). Ties leave it unchanged, and an all-tied set has $p_t = 1$.

4 Experiments

We compare the three banks under the fixed harness, ablate the output budget and forced closure, and test whether the stored changes work without the model.

4.1 Experimental Setup

Models and tasks. The PACE-Bench contrasts run Qwen3.8-27B-FP8 at low reasoning effort, temperature 1.0, and top-p 0.95, except the second closure contrast, on Qwen3.5-4B with thinking enabled. The memory contrast runs ten pairs from five physics families at six seeds, 60 units, and the analysis orders units by seed and then pair. The closure contrast uses the same pairs at three seeds, 30 units per backbone, and the oracle calibration runs on the Qwen3.8-27B-FP8 closure units. The four controllers use eight of those pairs at three seeds, 24 units each (Table 2), and the content checks cover 12 target pairs. An episode may submit at most 20 programs under a 49,152-token output budget per generation call. The 20-call protocol allowed 20 generation calls and 196,608 output tokens per episode, and the five-call protocol allows 5 generation calls and 98,304 tokens, both with 20 verifier calls. Every bank arm in the primary reset contrast runs under the five-call protocol. Each call is capped by the smaller of 49,152 tokens and the remaining episode allowance; forced closure continues generation for at most 4,096 tokens within that allowance.

Controllers and analysis populations. Four search implementations are adapted to the reset tasks; CodeEvolve starts from the source program without a bank (Assumpção et al., 2025). Each controller contributes 24 task–seed units, and the primary bank comparison contributes 60 units. For one task in the fluids family, transport failures required rerunning all 18 episodes: three bank conditions at six random seeds. Each rerun retains its original task, seed, and bank assignment. Acceptance, budget exhaustion, and agent-level errors are terminal outcomes; controller-process failures remain missing. The closure comparison has two incomplete pairs, covered by Equation 1. The bank, controller, and closure comparisons retain their own execution records and runtime settings, specified in Appendix B.

Table 1: Paired recovery differences and sharp finite-panel bounds. Ties equal Completed minus Wins and Losses. p_t is the conditional sign-flip reference.

			Completed pairs				Full-panel bound	
Contrast	Completed	Planned	Wins	Losses	$\bar{D}_t$	p_t	Lower	Upper
Closure, 27B	28	30	3	0	0.11	0.250	0.10	0.17
Closure, 4B	30	30	0	0	0.00	1.000	0.00	0.00
Same-family over no bank	60	60	12	11	0.02	1.000	0.02	0.02
Same-family over placebo	60	60	8	6	0.03	0.791	0.03	0.03
Placebo over no bank	60	60	10	11	−0.02	1.000	−0.02	−0.02
Same-family over OpenEvolve	24	24	3	3	0.00	1.000	0.00	0.00
Same-family over ShinkaEvolve	24	24	7	3	0.17	0.344	0.17	0.17
Same-family over Darwin Gödel	24	24	4	3	0.04	1.000	0.04	0.04
No bank over CodeEvolve	24	24	8	1	0.29	0.039	0.29	0.29

4.2 Paired Memory-Bank Outcomes

The bank contrast fixes output budget and closure across same-family, character-matched placebo, and no-bank arms. All 60 planned units in the primary bank contrast returned, so its finite-panel bound is the observed difference of $1/60$ (Figure 4B). The sibling bank resolves 23 episodes and no bank 22. Twelve units favor the sibling bank, eleven favor no bank, and thirty-seven tie (Table 1). Thus the observed positive sign comes from one more win than loss, with no evidence here of a reliably positive population effect.

The character-matched placebo probes whether a sibling relation helps beyond including a diff of similar length. The sibling-over-placebo difference is $2/60$, with eight wins and six losses; placebo resolves 21 episodes, one fewer than no bank (Table 1). Appendix C reports the completed results by family and seed.

Table 3 separates outcomes by physics family and seed; four of the six seeds favor the sibling bank or tie it, while the Cybernetics family resolves no episode in any bank arm. These subdivisions are descriptive at their smaller denominators.

4.3 Output-Budget and Closure Effects

A bank is read through an execution interface that can affect whether generated code reaches the verifier. We ablate two settings while holding the target, model, seed, and remaining limits fixed: the per-call output budget and the response to an end token inside the reasoning channel.

Output budget. We vary the per-call output budget between 16,384 and 49,152 tokens, labeled 16k and 49k in Figure 5, with Qwen3.8-27B-FP8 at low, medium, and extra-high API reasoning-effort settings. Two NetworkX functions, one generating adjacent pairs and one flattening nested iterables, start as empty implementations with two generation calls per episode, and each cap-matched comparison fixes the target, memory format, and seed. Low effort uses 24 episodes per target; other displayed conditions use 12. At extra-high reasoning effort and 16,384 tokens, 19 of 24 episodes reached the cap inside the reasoning channel and emitted no program. At 49,152 tokens every returned episode on the same seed groups emitted one (Figure 5A), and every adjacent-pairs episode then resolved (Figure 5B). For these episodes, the cap gates whether a program is produced, before its quality can be assessed.

The matched traces locate this gate in the first generation call. Let $L_{1,i}$ be the first-call length of comparison i at the raised cap, where every such call closes below 49,152 tokens. At the smaller cap $c = 16{,}384$, the observed no-program fraction $\hat{\theta}(c)$ satisfies

$$\hat{\theta}(c) = \frac{1}{24} \sum_{i=1}^{24} \mathbf{1}\{L_{1,i} > c\}. \tag{2}$$

Decoding is seed-coupled, with identical first-call prefixes up to the smaller cap. On all 24 cap-matched comparisons, an episode at 16,384 tokens has no program exactly when its first call at 49,152 runs past the smaller cap (Appendix C).

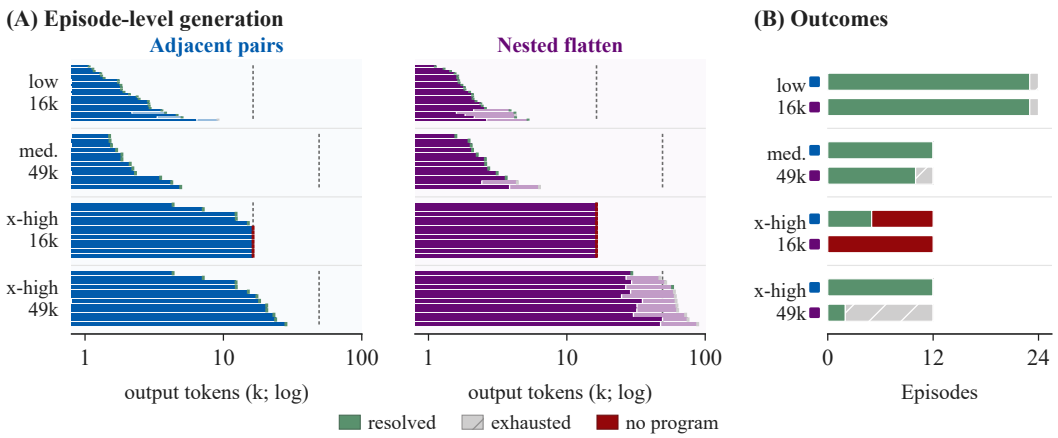

Figure 5: Generation and outcomes on two targets. (A) Episode-level generation; (B) Outcomes. Dashed lines mark the per-call caps on the cumulative-token axis.

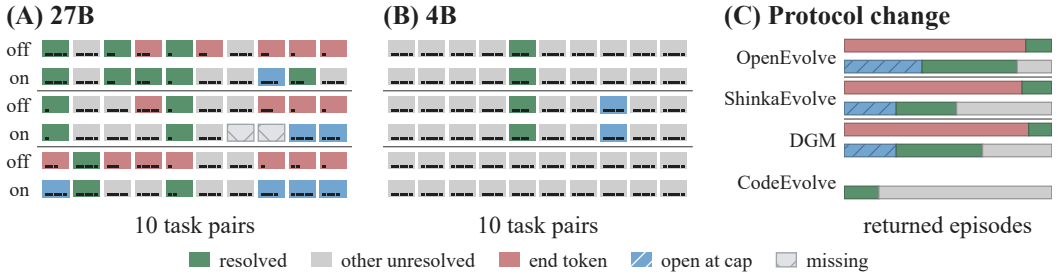

Figure 6: Closure comparisons. (A) 27B; (B) 4B; (C) Protocol change. Closure-off outcomes appear above their matched closure-on outcomes. Panel (C) also changes the call and token limits.

At extra-high effort, nested flatten reaches 21 times its spending at medium effort on the same two calls and resolves fewer episodes than at medium effort (Figure 5B). A raised cap therefore removes the no-program ending and leaves resolution open. For 19 of the 24 high-effort episodes at the smaller cap, failure is already determined before verification.

Forced closure. The 20-call protocol revealed a second no-program ending: an end token inside the reasoning channel. Across twelve adaptation pairs, 21 of 24 returned episodes in its controller panel had this ending for OpenEvolve, ShinkaEvolve, and the Darwin Gödel Machine (Figure 6C). Its occurrence across different procedures motivated a direct test of the shared end-token rule. In the initial five-call evaluation with forced closure, none of the 172 returned memory-arm episodes ended without a program. The open-channel endings in those executions each spent the full episode budget, reached the per-call output budget on the last call, and had scored at least one program before it. This termination diagnostic uses the initial executions; the primary bank contrast uses the 60-unit analysis described in Appendix B.

Motivated by this termination record, we intervene on the end-token rule. The matched comparison removes forced closure from the no-bank arm and keeps its other settings fixed, with Figure 6C describing the protocol change and panels A and B isolating the rule. The end-token counts show what the intervention changes. On Qwen3.8-27B-FP8, 16 of 30 returned closure-off episodes end on an end token inside the reasoning channel, and none of the closure-on episodes on the completed matched units ends that way (Figure 6A). This intervention isolates the end-token rule; the 20-call protocol comparison provides context because its call and token allowances also differ.

Recovery under Qwen3.8-27B-FP8 measures the episode-level outcome of this intervention. The full-panel bound over all 30 units is [0.10, 0.17] (Closure, 27B row of Table 1); every completion of the two missing pairs leaves a positive difference on this fixed panel. Three completed pairs favor

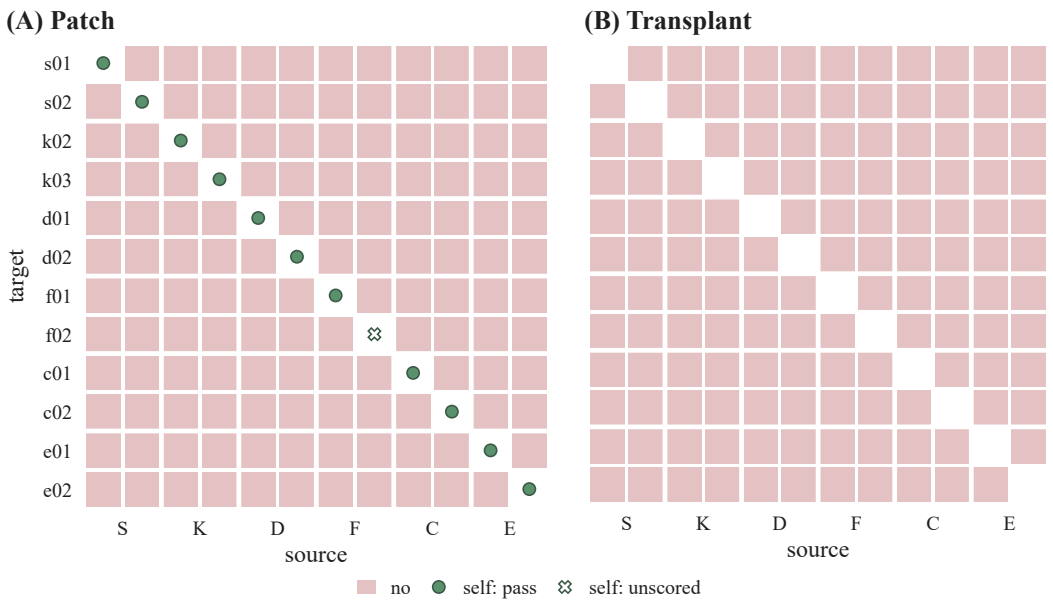

Figure 7: Direct reuse. (A) Patch; (B) Transplant. Each panel covers 132 cross-task transfers. S/K/D/F/C/E denote statics, kinematics, dynamics, fluids, cybernetics, and exotic physics. Diagonal Patch symbols are own-source controls.

closure and none favors closure off. In those wins, the closure-off episode stops inside the reasoning channel, consistent with the intervention's target. Qwen3.5-4B has no end-token stops inside its reasoning channel on these units, so closure has no opportunity to act. All 30 paired outcomes tie (Figure 6B; Closure, 4B row of Table 1). The closure effect therefore depends on whether the backbone emits end tokens inside its reasoning channel.

4.4 Mechanism Analysis

Direct reuse of stored changes. The paired outcomes do not show how the model uses a bank, so two model-free operators test whether the stored code works without model adaptation. Verbatim replay applies the stored diff from each of the other 11 pairs to a target's source-stage program as a patch and runs the verifier on every program a hunk changes, and transplant places each other pair's accepted Stage-1 program in the target and runs the target verifier. An operator check applies each pair's own diff to its own source and uses its Stage-1 verifier. None of the 132 stored cross-task transitions applies as a patch to its target source, because no hunk matches that source; each pair's own diff does apply to its own source (Figure 7A). No whole-program transplant passes a target verifier, including same-family siblings (Figure 7B). These checks exclude verbatim patch application and whole-program transplant as sufficient explanations for the observed outcomes. They do not determine whether the model uses particular concepts, code fragments, or merely the presence of a diff when it generates a new program.

Search controllers. The controller comparison evaluates four adapted search procedures on the same task pairs under their recorded budgets. All 24 units return for each controller. Table 1 pairs OpenEvolve, ShinkaEvolve, and the Darwin Gödel Machine with the same-family bank arm, and CodeEvolve with the no-bank arm; no controller wins more paired units than its matched arm. A controller's full procedure includes candidate selection, generated edits, and implementation-specific runtime behavior; these runs cannot isolate an archive effect or establish equivalence among procedures. Their outcomes are descriptive for the tested tasks and execution settings.

5 Conclusion

Harness^2 makes the execution conditions behind a memory score visible. Output truncation and end-token termination can prevent a program from reaching the verifier; after fixing these settings, the reset contrast compares what each supplied bank changes. On the completed 60-unit panel, sibling reference content resolves one more episode than no bank, with 12 wins, 11 losses, and 37 ties, while direct patch and transplant checks fail. These results put program production, repair, and cross-task content into one evaluation sequence. The next question is which memories turn available execution into repeatable recovery across tasks and harnesses.

Limitations. The sibling-versus-placebo difference and a population-level benefit remain unresolved. The closure diagnostic retains two incomplete pairs. One physics family resolves no episode in any bank arm. The primary memory panel covers ten PACE-Bench pairs and one backbone, while the closure diagnostic adds a second backbone and the cap study uses two NetworkX functions. New tasks, models, and agent-written banks require fresh matched evaluation.

Reproducibility Statement

Appendix B gives model identities, decoding settings, output budgets, the unit order, and core populations; Appendix A gives the paired-analysis proofs. Appendix E lists the evaluation records and reproduction commands. Code and analysis outputs will be released with the paper. The token-cap study uses two held-out NetworkX targets, and the primary memory and closure contrasts use PACE-Bench adaptation pairs.

AI Use Statement

Generative artificial intelligence tools assisted primary-source retrieval, research design, implementation, analysis, visualization, manuscript drafting, and proof development. The authors are responsible for the scientific claims and the submitted manuscript.

References

Henrique Assumpção, Diego Ferreira, Leandro Campos, and Fabricio Murai. CodeEvolve: An Open Source Evolutionary Coding Agent for Algorithm Discovery and Optimization, 2025. URL https://arxiv.org/abs/2510.14150.

Le Chen, Zishen Wan, Baixi Sun, Xiaolong Ma, Chih-Hsuan Yang, Feng Yan, Sheng Di, Franck Cappello, and Rajeev Thakur. Measure before you manage: Evaluating agent working memory in coding agents, 2026. URL https://arxiv.org/abs/2608.31057.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, Nicholas Thumiger, Aditya Desai, Ion Stoica, Ana Klimovic, Graham Neubig, and Joseph E. Gonzalez. The Danger of Overthinking: Examining the Reasoning-Action Dilemma in Agentic Tasks, 2025. URL https://arxiv.org/abs/2502.08235.

Liyang Fan, Yingcheng Shi, Yongbin Li, Chenghao Sun, Xin Chen, Xander Xu, Hu Wei, Shiwen Ni, Min Yang, and Jieping Ye. VibeMemBench: Evaluating memory systems for coding agents on real repository coding tasks, 2026a. URL https://arxiv.org/abs/2609.23570.

Run-Ze Fan, Zihao Zhang, Simin Ma, Yebowen Hu, Shouju Wang, Kaiqiang Song, Fei Liu, Hamed Zamani, and Xiaoyang Wang. An Empirical Study of Harness Design for Coding Agents, 2026b. URL https://arxiv.org/abs/2609.20804.

Yiyang Feng, Biddut Sarker Bijoy, Niranjan Balasubramanian, and Jiawei Zhou. Break it down, pass it on: Cross-task skill transfer in LLM agents, 2026. URL https://arxiv.org/abs/2608.20274.

Shuzheng Gao, Wenhao Zeng, Zhaojian Yu, Jianqiao Wangni, Chaozheng Wang, Kai Cai, Shilin He, and Michael R. Lyu. SWE-MeM: Learning Adaptive Memory Management for Long-Horizon Coding Agents, 2026. URL https://arxiv.org/abs/2606.28434.

Joongwon Kim, Wannan Yang, Kelvin Niu, Hongming Zhang, Yun Zhu, Eryk Helenowski, Ruan Silva, Zhengxing Chen, Srinivasan Iyer, Manzil Zaheer, Daniel Fried, Hannaneh Hajishirzi, Sanjeev Arora, Gabriel Synnaeve, Ruslan Salakhutdinov, and Anirudh Goyal. Scaling test-time compute for agentic coding, 2026. URL https://arxiv.org/abs/2604.16529.

Anany Kotawala. Resolution diagnostics for paired llm evaluation, 2026. URL https://arxiv.org/abs/2605.30315.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. ShinkaEvolve: Towards open-ended and sample-efficient program evolution. In *International Conference on Learning Representations*, 2026. URL https://iclr.cc/virtual/2026/poster/10007692.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-end optimization of model harnesses, 2026. URL https://arxiv.org/abs/2603.28052.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery, 2025.

Swapnanil Saha. Delivery, not storage: Cue-anchored working memory as a harness property for coding agents, 2026. URL https://arxiv.org/abs/2607.20972.

Asankhaya Sharma. OpenEvolve: An open-source evolutionary coding agent. https://github.com/algorithmicsuperintelligence/openevolve, 2025.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic Cheatsheet: Test-Time Learning with Adaptive Memory. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7080–7106, Rabat, Morocco, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.eacl-long.333. URL https://aclanthology.org/2026.eacl-long.333/.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models, 2023. URL https://arxiv.org/abs/2305.16291.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent Workflow Memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911, 2025. doi: 10.48550/arXiv.2409.07429. URL https://proceedings.mlr.press/v267/wang25bx.html.

Siqi Yang, Qianlan Yang, Yu-Xiong Wang, Saurabh Pujar, and Martin Hirzel. One recipe, many harnesses: What self-evolution encodes across languages and models, 2026. URL https://arxiv.org/abs/2608.10178.

Yuhao Zhan, Bingxiang He, Zecong Tang, and Chaojun Xiao. PACE-Bench: Benchmarking Physics Adaptation via Code Evolution in Dynamic Environments, 2026. URL https://arxiv.org/abs/2608.14441.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel Machine: Open-Ended Evolution of Self-Improving Agents. In *International Conference on Learning Representations*, 2026. doi: 10.48550/arXiv.2505.22954. URL https://iclr.cc/virtual/2026/poster/10007327.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM Agents Are Experiential Learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642. AAAI Press, 2024. doi: 10.1609/aaai.v38i17.29936.

APPENDIX

A STATISTICAL FOUNDATIONS OF PAIRED EVALUATION

The paired analysis separates the recovery difference on the evaluated panel from the conditional reference distribution of paired outcomes. For each of N fixed task–seed units, let $Y_i^T, Y_i^C \in \{0, 1\}$ indicate verifier acceptance under treatment and control, and write $D_i = Y_i^T - Y_i^C$. At time t, C_t contains units with both outcomes observed; W and L count its wins and losses. Incomplete outcomes determine the finite-panel bounds below. The sign-flip reference then specifies the joint invariance needed to interpret the observed wins and losses probabilistically.

A.1 PROOFS OF THE FINITE-PANEL BOUNDS

Proposition 1 (Sharp finite-panel bounds)**.** *For each unit, let $[\ell_i, u_i]$ contain all differences compatible with its observed outcomes: $[D_i, D_i]$ when complete, $[Y_i^T - 1, Y_i^T]$ when only treatment is observed, $[-Y_i^C, 1 - Y_i^C]$ when only control is observed, and $[-1, 1]$ when neither is observed. The sharp interval for $\bar{D} = N^{-1} \sum_i D_i$ is $B_t = [N^{-1} \sum_i \ell_i, N^{-1} \sum_i u_i]$, as in Equation 1. Its width is $(S_t + 2U_t)/N$, where S_t counts units with one returned arm and U_t counts units with neither arm returned.*

Proof. For each unit in Equation 1, the observed arm outcomes fix the known indicators and each missing binary indicator can take either value in $\{0, 1\}$. Choosing its favorable or unfavorable endpoint attains the stated unit bounds. Summing these choices attains both panel endpoints, proving sharpness without assumptions about return order or missingness. Each unit with one missing arm contributes width one, and each unit with two missing arms contributes width two. Averaging gives $(S_t + 2U_t)/N$. □

A.2 PROOFS OF THE SIGN-FLIP REFERENCE

Proposition 2 (Sign-flip reference)**.** *At a cutoff fixed in advance, suppose the joint law of all outcomes and return times is invariant under swapping either arm of any unit. Conditional on the completed set and its discordant units, all 2^{W+L} sign vectors are equally likely. Consequently,*

$$p_t = \min \left\{ 1,\ 2^{1-(W+L)} \sum_{k=0}^{\min(W,L)} \binom{W+L}{k} \right\}. \tag{3}$$

Proof. Completing a pair and being discordant are unchanged by an arm swap. Swapping one unit flips only its sign. Composing such swaps maps any discordant sign vector to any other, with equal probability under the joint invariance. Thus the conditional signs are uniform, even if the original units are dependent. Summing the two binomial tails gives Equation 3. □

This is a joint invariance assumption, including return times, not just equal marginal success rates. Closure changes termination itself; its displayed sign-flip value is descriptive. The experiment does not establish this symmetry through randomized arm assignment; the reported values are conditional references under the stated null.

B EXPERIMENTAL PROTOCOL

Tasks and units. The primary contrast uses ten PACE-Bench adaptation pairs: s01, s02, k02, d02, f01, f02, c01, c02, k03, and d01. The prefixes denote statics, kinematics, dynamics, fluids, and cybernetics. Fluids includes granular/fluid interaction; Cybernetics denotes control tasks (Zhan et al., 2026). Task labels omit the benchmark identifiers' underscores and use lower case; for example, f02 denotes PACE-Bench task `F_02` in the fluids family. Six seeds (1729, 2718, 3141, 4242, 5003, 6007) give 60 task–seed units, ordered by seed and then by the listed pair order. Closure and oracle contrasts use the first three seeds; controller comparisons use the first eight pairs and first three seeds. Direct-reuse checks cover 12 pairs. Verbatim replay and transplant are model-free checks

and use no generation calls or token budget. A seed is a repeated sampling condition on a task, not a new task.

Reset and bank construction. The PACE episode receives the source-stage program, target task description, and assigned bank. The donor pool additionally includes two exotic-physics tasks, e01 and e02; the prefix e denotes that sixth family, which is absent from the ten-task memory panel. The target reference program, prior target interactions, and earlier verifier reports are excluded. The fixed donor order is c01, c02, d01, d02, e01, e02, f01, f02, k02, k03, s01, s02. Each target has exactly one other donor in its family; that donor supplies the sibling bank. Each record contains the donor identifier and its Initial-to-Stage-1 unified diff with two context lines, capped at 12,000 diff characters. The placebo concatenates records from other families in the listed order and takes a prefix with the same character count as the sibling record, including its header and code delimiters. The no-bank arm supplies no record. Bank assignments are fixed across seeds. Character matching does not match token counts: on s01 the sibling and placebo increments are 207 and 224 prompt tokens, and on f01 they are 883 and 825. The oracle supplies the target's own diff and is a privileged calibration, not a cross-task competitor; it resolves 23 of 30 episodes.

Execution and accounting. The primary model is Qwen3.8-27B-FP8 at low reasoning effort, temperature 1.0, and top-p 0.95. The second closure model is Qwen3.5-4B with thinking enabled. Primary episodes allow five generation calls, 20 verifier submissions, and 98,304 output tokens in total. Each call receives the smaller of 49,152 tokens and the remaining episode allowance; no further call starts once that allowance is exhausted. Forced closure replaces an end token inside reasoning with `</think>` and continues for at most 4,096 tokens within the remaining call and episode limits; the continuation belongs to the same generation call. The 20-call protocol allowed 20 generation calls and 196,608 output tokens. Acceptance, budget exhaustion, and agent-level errors are terminal outcomes. Controller initialization failures are recorded separately. An open-channel final call can follow earlier scored programs. The 27B closure comparison reuses no-bank executions from before the fluids-task reruns. These controls overlap with the bank study, so population rows must not be added as independent samples. Mean output includes one terminal execution per task, seed, and bank condition. Where an execution was replaced, only its replacement enters this mean; the earlier execution and its token use remain available in the reproduction records.

Outcome mapping. An episode recovers when the target verifier accepts a submitted program within the execution limits. A terminal episode without an accepted program contributes zero; an episode without a terminal outcome remains missing. A pair is complete only when both compared arms have returned on the same task and seed. The paired difference is treatment success minus control success, so $+1$, -1, and 0 denote a win, loss, and tie.

Execution records and analysis populations. The primary memory analysis uses all 60 units after rerunning 18 episodes for task f02 because of transport failures. These are the six listed seeds in each of the three bank conditions: sibling, placebo, and no bank. Each rerun replaces the observation for the same task, seed, and bank condition; it does not add a new evaluation unit. The final panel has 12 wins, 11 losses, and 37 ties, with difference $1/60$. The closure and oracle diagnostics use the initial executions. Their 27B no-bank outcomes are therefore distinct from the reruns in the final bank comparison.

The NetworkX rows in Table 2 describe the output-cap study; Seed extensions denotes its four additional groups of random seeds.

C Supplementary Results

Memory by family and seed. Table 3 gives all strata of the completed memory panel; each arm returns all 60 episodes. Four of six seeds favor or tie the sibling bank against no bank, and cybernetics resolves no episode in any arm. These strata describe the measured tasks and seeds. Table 1 gives the corresponding aggregate recovery differences.

Direct reuse. Figure 7 reports the two model-free operators across 132 directed cross-task transfers. No diff applies to its target source, and no whole-program transplant passes the target verifier,

Table 2: Planned populations and per-episode limits. The three memory arms share 60 task–seed units (180 episodes); Gen. and Ver. are call limits, and Cap is output tokens per call.

Population	Tasks	Model	Gen.	Ver.	Cap	Unit	n
Memory contrast	10 PACE	27B	5	20	49,152	episode	180
Controllers	8 PACE	27B	5	20	49,152	episode	96
Closure contrast	10 PACE	27B	5	20	49,152	episode	60
Closure contrast	10 PACE	4B	5	20	49,152	episode	60
Oracle calibration	10 PACE	27B	5	20	49,152	episode	30
Held-out effort	2 NetworkX	27B	2	2	16,384	episode	72
Held-out effort	2 NetworkX	27B	2	2	49,152	episode	48
Seed extensions	2 NetworkX	27B	2	2	49,152	episode	24

Table 3: Completed memory-panel outcomes by physics family and seed. Counts are episodes within each bank arm; Resolved denotes verifier acceptance, not paired wins.

	Same family			Placebo			No bank		
Stratum	Planned	Returned	Resolved	Planned	Returned	Resolved	Planned	Returned	Resolved
Cybernetics	12	12	0	12	12	0	12	12	0
Dynamics	12	12	3	12	12	2	12	12	3
Fluids	12	12	9	12	12	9	12	12	9
Kinematics	12	12	6	12	12	4	12	12	3
Statics	12	12	5	12	12	6	12	12	7
Seed 1729	10	10	5	10	10	2	10	10	6
Seed 2718	10	10	4	10	10	5	10	10	3
Seed 3141	10	10	2	10	10	3	10	10	3
Seed 4242	10	10	4	10	10	4	10	10	3
Seed 5003	10	10	4	10	10	3	10	10	4
Seed 6007	10	10	4	10	10	4	10	10	3

including the 12 same-family transfers. As an operator control, each pair's own diff applies to its own source; 11 patched programs obtain score 100, while one verifier returns no score. These checks exclude the tested literal-reuse operations, not other ways for a model to use a bank.

Output-cap diagnostic. At extra-high effort, raising the per-call cap from 16,384 to 49,152 tokens moves adjacent-pairs recovery from 5 of 12 to 12 of 12; nested flatten at the raised cap resolves 2 of 12. The raised-budget first call exceeds the smaller cap on 7 of 12 adjacent-pairs comparisons and all 12 nested-flatten comparisons. These are exactly the smaller-cap episodes without a program; Equation 2 agrees on all 24 comparisons. First-call prefixes match up to the smaller cap. Four further seed groups plan 24 episodes but return only three; they remain outside the main cap panel, with their planned denominators retained in the repository results.

Closure diagnostic. On Qwen3.8-27B-FP8, 16 of 30 returned closure-off episodes end inside reasoning at an end token, seven before any program; none of the 28 closure-on episodes in the matched comparison has that ending. All three paired recovery wins are on units with the closure-off end-token failure. On Qwen3.5-4B each arm resolves 2 of 30 and every paired outcome ties.

D Scope of the Evaluation

Controller implementations. The three OpenEvolve, ShinkaEvolve, and Darwin Gödel Machine contrasts use the sibling-bank arm as treatment; only the CodeEvolve contrast uses no bank. All 24 CodeEvolve task–seed units use its editable-block interface. For the fluids task requiring reruns, its three random seeds use scaled-dot-product attention with 2,048-token prefill chunks after the initial executions exhausted device memory before generation; two runs of other controllers on the same task were also replaced. Results compare complete adapted procedures, including runtime conditions, rather than isolated archive or selection effects or reproductions of original published protocols. The reproduction records retain earlier executions and their token use.

Scope of supplied memory. A reference diff controls what enters the prompt without measuring how well an agent learns, summarizes, or retrieves its own experience. Excluding the target's reference change does not remove shared family structure or establish absence of model pretraining exposure. Character-matched diffs can differ in token count and semantic relevance. The primary memory contrast uses one backbone; the second backbone tests closure only. Further tasks and models require new matched evidence.

E Reproduction Resources

The repository entries below locate the evaluation records, analysis scripts, and archived supplementary studies. Paths are relative to the repository root. Use Python 3.12 with dependencies pinned in `uv.lock`; run `uv run python paper/tables/gen_numeric_tables.py` to regenerate tables and `uv run python paper/check_body_numbers.py` to check them. In the experiment source directory, `summarize_experiment_closure.py` recreates the reported analysis from the directory supplied to `--results` and saves it at the path supplied to `--output`.

Material	Repository entry
Memory and controller evaluation records	`experiments/reconstruction-frontier/results/experimental-closure-20260926/analysis.json`
Episode runners and analysis	`experiments/reconstruction-frontier/src/`
Archived supplementary studies	`paper/archive/appendix-paper-v36.tex`
Extended reconstruction theory and results	`theory/framework.md`; `experiments/pace-reconstruction/`
Finite-system enumeration and invariance	`experiments/recursive-interface-closure/`