CO-BATCHING CAN CHANGE WHAT AGENTS DO: TOOL EXECUTION AND HISTORY EFFECTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-using agents turn model outputs into operations that read information, modify records, and complete tasks. Inference servers batch unrelated requests for efficiency, but can these shared execution conditions change an agent's behavior as well as its output text? We study this question in AppWorld by holding the target request fixed, varying co-batch conditions, executing the resulting actions, and following their consequences and action–observation histories. Repeatable operational forks occur in four of five tested deployment panels. Paired continuations differ on task checks in 36 of 97 evaluable selected states within an eight-decision window, including completed tasks. Restoring common measured database contents still leaves different supplied histories capable of changing subsequent actions. These differences persist across reversed comparison orders, while matched same-history controls agree. On new tasks, average task-check disagreement beyond repeated-condition background is positive but uncertain on both deployments. Deterministic configurations eliminate the tested initiating forks in controlled Qwen and DeepSeek audits; MiniMax shows partial prevention alongside residual and induced differences. The findings show that serving variation can affect executable operations and that database agreement alone is insufficient for behavioral agreement. Evaluating agent reproducibility therefore requires observing serving conditions, executed behavior, and retained interaction history together.

1 INTRODUCTION

An agent's generated output can become an operation: reading a note, updating a record, or submitting an answer. Variation in model serving can therefore become a question about behavior, not just wording. Inference servers group requests into batches to share computation. The other requests may concern unrelated users and tasks; their contents remain outside the target prompt. Can changing these shared execution conditions nevertheless alter what the agent does?

Answering this question requires following a generated action through execution. Different calls can have the same measured effect or leave different tool observations and database contents. The agent then acts on both the application data and its action–observation history, so database agreement alone may be insufficient for subsequent behavior to agree.

Consider an AppWorld task asking for the longest good-posture streak in habit logs. At a saved batching fork, one action reads one note and the other reads five. A later intervention restores common database contents and supplies either saved action–observation history. The new continuations answer six or four days; four is the benchmark answer, and all eight finish (Figure 1). These are repeated interventions on one selected state, rather than an uninterrupted run from the original fork.

Prior work documents batch-conditioned behavioral changes, cache-dependent tool-agent trajectories, and common-state context interventions (Kadadekar, 2026; Patodiya, 2026; Min et al., 2026). We ask which fixed-request action differences survive execution, which reach task checks, and whether the histories they leave affect behavior at common database contents. Real tool execution separates these consequences; reversed orders and same-history controls strengthen the selected-state evidence for history dependence. We then examine new tasks and deterministic configurations to locate the scope and practical limits of these findings.

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.

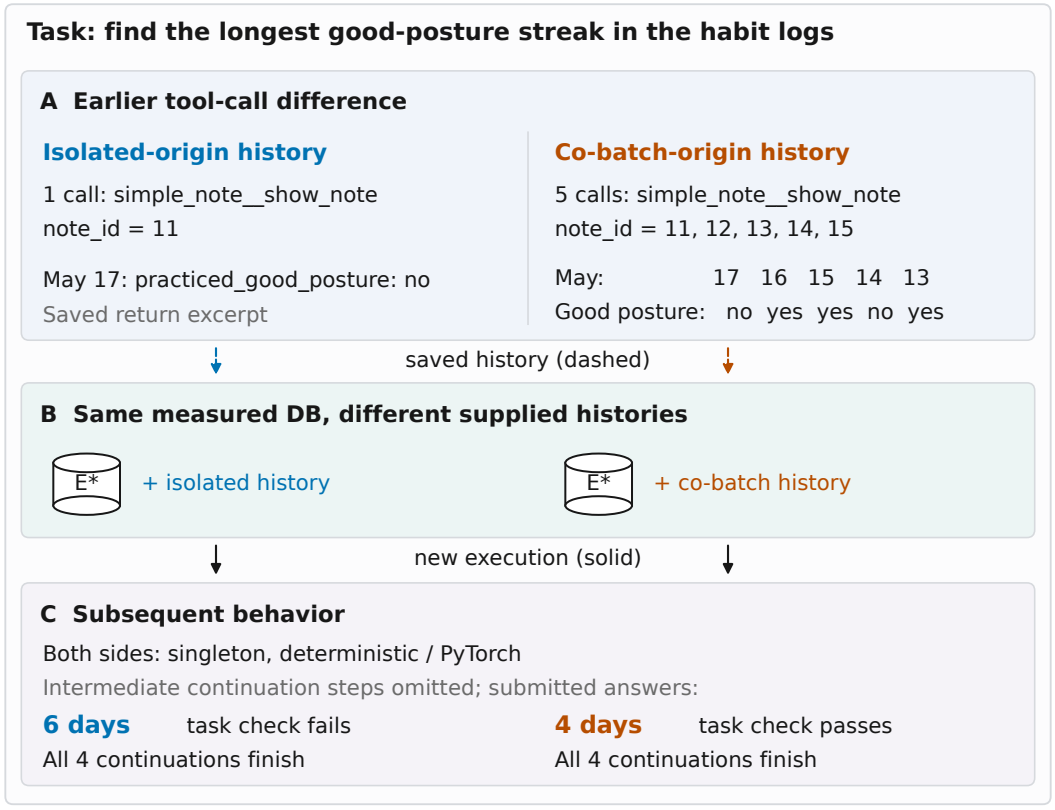

Figure 1: One habit-tracking task yields different answers under two supplied histories. A shows saved call/return excerpts; dashed arrows supply them in a later common-database intervention (B), and solid arrows lead to new continuations (C). Color denotes history origin; both sides continue with singleton deterministic/PyTorch. Omitted steps separate the excerpts from the answers. The benchmark answer is 4. Four runs per history repeat one MiniMax state (F01.1, decision 5) across two blocks. This is not one uninterrupted native episode. Display IDs map to records in Appendix L.

2 Related Work

Serving conditions and agent behavior. System configuration, batch size, and numerical precision can change LLM outputs (Yuan et al., 2025). Batch-conditioned testing finds sparse refusal/capability flips, without a detected aggregate composition effect at its stated sensitivity, and tests invariant kernels with scheduler-verified overlap (Kadadekar, 2026). Patodiya (2026) studies cache settings and quantization in eighty stateful tool episodes at serial batch one, finding trajectory divergence and restored-cache effects; a separate arithmetic task shows bidirectional correctness changes without a detected aggregate accuracy shift. We vary co-batch occupancy and companions, following executable actions through tool returns, database contents, and retained history. τ-bench evaluates task completion against database goals and reliability across repeated trials (Yao et al., 2025). Motivated by such reliability measures and repeated-agent variability (Bjarnason et al., 2026; Yagubyan, 2026; Duesterwald et al., 2026), our native-task contrast subtracts its own within-condition repeat disagreement.

History and common-state intervention. TRACE compares pre/post-compression contexts through closed-loop continuations from the same AppWorld state, using execution regressions to improve compression prompts (Min et al., 2026). Causal Agent Replay distinguishes counterfactual intervention targets (Shah, 2026). We instead supply histories from a fixed-request co-batch fork at common database contents, using restarted blocks, reversed orders, and time-matched same-history controls.

Deterministic inference and local intervention. SGLang provides a runtime for structured model programs (Zheng et al., 2024); its documentation describes deterministic inference (SGLang Project, 2026). LLM-42 and CoRun study deterministic serving (Gond et al., 2026; Zhao et al., 2026). We test how existing configurations affect action agreement at the deployed tool interface. MarginGate verifies tokens and Thought Anchors intervenes on reasoning (Chu et al., 2026; Bogdan et al., 2025); we report unsuccessful prediction and local content interventions.

3 Comparing Serving Conditions, Execution, and History

3.1 A fixed request under two serving conditions

We compare a saved request under two batch conditions in AppWorld, which supplies benchmark APIs, persistent databases, and programmatic task checks (Trivedi et al., 2024). Write $a_b = \mathrm{parse}(G_{\theta,\sigma}(x; b))$, with fixed target input x, batch condition b, model θ, and documented serving settings σ. Input token IDs, tokenizer, tool schema, renderer, parser, precision, tensor parallelism, backend, and greedy decoding are held fixed. The target runs alone (*singleton*) or in a fixed slot with three saved requests from unrelated tasks (*co-batch*). Occupancy and companion composition change together. Figure 2 follows the calls into execution.

The five deployment stacks differ in task support, selection, and repetition budget, precluding a model-capability ranking. Repeats qualify alternatives for execution (Appendix A).

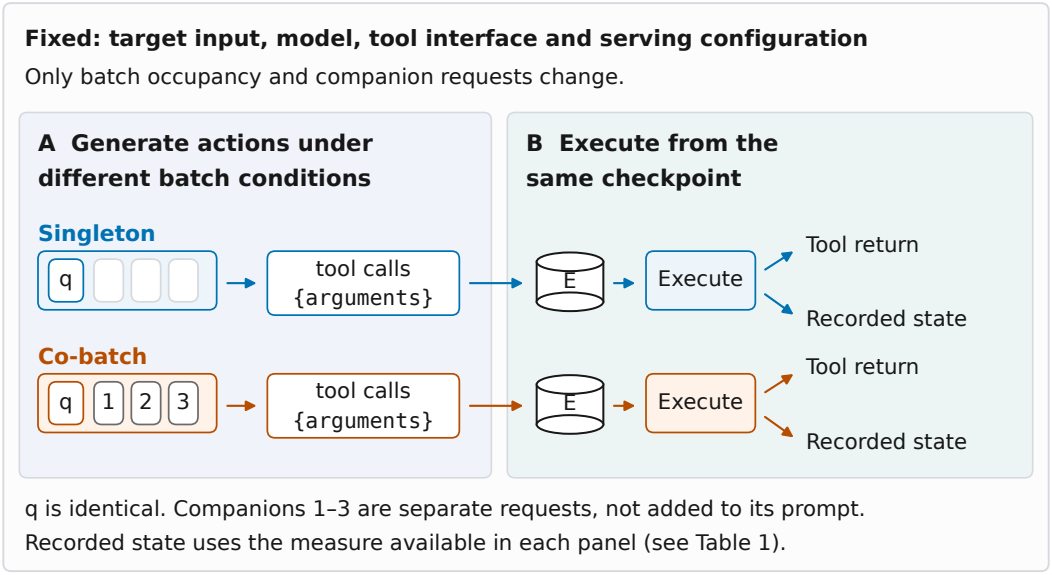

Figure 2: Batch conditions change while the target request stays fixed. Companions remain outside q's prompt. Calls execute in fresh instances restored from E. Tool returns and recorded state are separate measures; state uses rows or the retained proxy (Table 1). Appendix A maps shared states.

3.2 What changes at the tool interface?

The parser produces an action a: ordered tool names and JSON arguments. Canonicalization removes whitespace and object-key order but preserves call and array order; omitted defaults are not inferred. An *admissible* action is nonempty, parser-valid, and schema-compatible, but can still be wrong for the task.

Execution yields observation o and database contents E'. An *operational fork* is a repeat-qualified admissible contrast differing in o or the panel's state projection. *Execution-clean* requires all four initial executions to have the clean class; *error-mediated* contrasts include reproducible API errors. Equal measured transitions are *execution-equivalent*. These classes do not measure correctness (Appendix B).

The official evaluator supplies an endpoint *task-check vector* r, one pass/fail value per requirement. *Official success* differs from harness *completion*: an agent may finish and fail. Natural completion, decision caps, context censoring, and unavailable measurements remain distinct; capped runs have unobserved eventual outcomes.

Row-content reconstruction covers the DeepSeek controlled census, the expanded action-continuation and history panels, and A/B content checks. Other controlled results retain the original mutation-counter state proxy, with targeted content checks for MiniMax exclusions (Table 1; Appendix B). Neither measure includes unlogged interpreter, filesystem, RNG, or cache state; fixed σ does not certify their equality.

3.3 Following an action or substituting its history

Two interventions distinguish action execution from its supplied history. In the *action-continuation panel*, fresh instances restore the same checkpoint, execute either selected execution-clean action, and continue under singleton standard/FlashInfer inference. Assistant messages contain canonical calls with `content=null`; tool messages carry real responses. Branches can differ in both database and history.

The *history panels* instead restore a common isolated-derived database E^* and supply either branch's canonical action–observation history, H_I or H_C, after the shared message prefix. Original free-form reasoning is not transplanted. Both continuation inputs (E^*, H_I, σ_c) and (E^*, H_C, σ_c) use the same singleton deterministic/PyTorch configuration σ_c. The broader discovery panel uses two repeats per history in fixed order C_1, C_2, I_1, I_2.

The *counterbalanced history study* (A) adds cross-history and same-history pairs in two restarted blocks, reversing history order and pair placement. Prespecified wall-clock gaps match the comparisons (Appendix E). Its state-level outcome is first-admissible-action disagreement across histories minus matched same-history disagreement, averaged over both blocks. Invalid actions or unmatched timing leave the primary state value missing, even if one block qualifies. Selected-state continuations use an eight-decision horizon.

The *native-task study* (B) applies either serving condition throughout episodes, retaining original assistant content, reasoning, calls, and returns. It uses a 32-decision horizon and two sessions of two runs per condition. Scheduler logs qualify overlap; Section 6 defines the contrast and Appendix G gives full settings.

3.4 Units and uncertainty

A *state* is a saved task decision. Selected-state panels summarize states; the native study summarizes tasks. Repeats reuse inputs, and service blocks group runs under a restarted server: neither adds independent states. A reuses twenty state/deployment entries from the 77-state history panel, itself drawn from the 103-state action panel. B uses the same sixteen tasks on both deployments. These interventions do not identify a joint mediation effect.

Prespecified A/B intervals use a family-cluster sandwich standard error and Student t_{G-1} limits, requiring complete blocks/sessions without zero imputation. Appendix F reports clustering and missing-value sensitivities. Older endpoint bootstrap intervals describe the selected panels posthoc.

4 Co-Batching Changes Executable Actions

Can a changed batch alter both a repeatable tool action and its executed effect? Operational forks appear in four of five deployment panels (Table 1). Execution narrows the set of structural differences: Qwen has twelve structural candidates but ten operational states.

DeepSeek and MiniMax each have 52 structural candidates and fifty operational states. Some forks execute cleanly, while others act through API errors. This distinction determines which alternatives can support a clean continuation comparison.

In DeepSeek, default `complete_task()` and an explicit failure argument return the same observation and mutation counter but store different task status. Row-content reconstruction detects this

Table 1: Fixed target inputs can yield different executable actions. Counts are selected states; clean states contain an execution-clean operational contrast. DeepSeek uses row contents; MiniMax uses observation-supported positives with content-checked exclusions; Qwen/Llama retain the state proxy. GLM has no candidate execution. Task support and repeat budgets differ.

Deployment	States	Tasks/families	Structural	Operational	Clean
Qwen3-32B	400	60/20	12	10	4
DeepSeek-V3.2	200	30/10	52	50	40
Llama-3.3-70B	200	36/12	1	1	1
MiniMax-M2.5	200	36/12	52	50	26
GLM-5-FP8	200	36/12	0	0	0

difference, while five other equal-observation exclusions remain content-equivalent. Neither action text nor a mutation counter alone captures these executed transitions.

Repeat disagreement and invalid outputs limit qualification, including in GLM's zero-fork panel; the unqualified records remain visible in Appendix A. Zero qualified forks is narrower than general stability.

Previously observed alternative tool actions also recur when requests arrive over time and the server dynamically forms batches; scheduler logs confirm actual co-batching (Appendix I). This tests whether known forks recur under dynamic load in selected cases, rather than estimating their frequency in production.

5 Execution and History Shape Later Behavior

5.1 Do initiating action differences reach task outcomes?

To test downstream consequences, we execute either selected clean action from the same checkpoint and continue for up to eight decisions. Of 103 planned states, 97 are estimable: 57 DeepSeek and forty MiniMax. Task checks differ in 36 states, including completed tasks with different answers or operations. At the window end, 51 differ in database contents or task checks, and seventeen in official success. Ten success-discrepant pairs both complete; seven include an $H = 8$ cap (Figure 3).

These channels give different interpretations. The fifteen database-only endpoints have no established consequence for an official requirement. In a completed dinner-payment case, one branch makes a single \$30.50 transfer and the other makes \$12.25 plus \$18.25: equal totals coexist with a failed single-transaction check. These are simulated benchmark payments, not real expenditure. Appendix H gives both success directions and successful counterexamples; the seventeen discrepancies count disagreement, not net accuracy loss. Thus some initiating forks reach task checks, while database change alone is insufficient to identify task harm.

5.2 Does history matter at common database contents?

We hold measured database contents common and intervene by supplying either canonical action–observation history from the earlier executions. This discovery panel reuses 77 action-panel states: fifty DeepSeek and 27 MiniMax. Row-content reconstruction verifies that both source branches already had equal measured database contents after the initiating actions, matching the common database restored for intervention. Of these states, 71 meet first-action and downstream repeat qualification; six have illegal or empty first actions. Sixty differ at the first admissible action and 69 by $H = 8$. Twenty-two differ at the database-or-task-check endpoint, fifteen in task checks, and five in official success; three success-discrepant pairs both finish.

Of 49 states without an endpoint difference, fifteen take different actions but both finish at the same measured endpoint. Another 32 differ behaviorally while at least one arm reaches the cap; their eventual outcomes are unobserved. Two have no admissible-action difference. Matching database contents therefore leaves a separate information channel through history. The intervention changes action and observation messages together, rather than identifying either component alone or replaying the original reasoning verbatim.

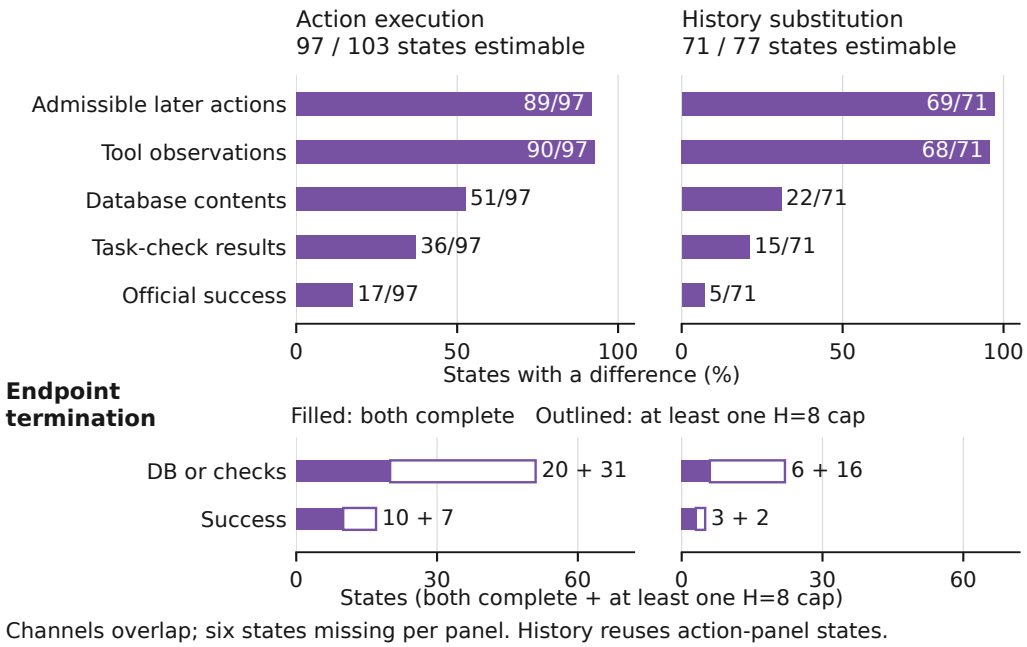

Figure 3: Later behavior differs more often than official success. Upper bars show percentages; n/N labels count differing/estimable selected states. Lower panels separate database-or-task-check and success differences into both-complete (filled) and cap-involved (outlined) states. Channels overlap, rather than forming a funnel. Each panel has six missing states; history reuses action-panel states, so columns are not independent groups. An $H = 8$ cap leaves the eventual outcome unobserved.

5.3 Does the history effect survive order and repeat controls?

The discovery panel's fixed order leaves order and elapsed time as alternative explanations. The counterbalanced study restarts service between two blocks, reverses history order and pair placement, and adds time-matched same-history controls. It uses twenty selected state/deployment entries, seventeen tasks, and thirteen families, with eight continuations per state. Each deployment has eight complete states out of ten planned. In each block, first admissible actions differ on 7/8 DeepSeek and 8/8 MiniMax cross-history pairs; same-history counts are 0/8 for both (Figure 4).

The state-averaged disagreement excess is 0.875 for DeepSeek (95% t interval [0.579,1.171]) and 1.000 for MiniMax ([1,1]). The MiniMax interval reflects zero empirical variance across eight observed states, not population certainty. Appendix F gives the broader bounded-family sensitivity intervals and planned-set missing-value bounds. Invalid actions and the timing exception remain missing from the complete two-block analysis.

Reversed orders and agreeing same-history controls strengthen the evidence for history-dependent behavior beyond measured repeat background. Appendix E reports downstream outcomes, including pairs outside the first-action analysis.

6 New Tasks Show Heterogeneous and Uncertain Effects

On new tasks, we compare task-check disagreement across conditions with the background disagreement between same-condition repeats. This measures excess task-check inconsistency, not every path change. Both deployments use the same sixteen tasks from families unused in retained experiments; "new" does not assert absence from pretraining. Identity hashes select tasks before outputs, without screening for forks, success, or completion. There are 128 trajectories per deployment, including 64 co-batch, and 256 in total.

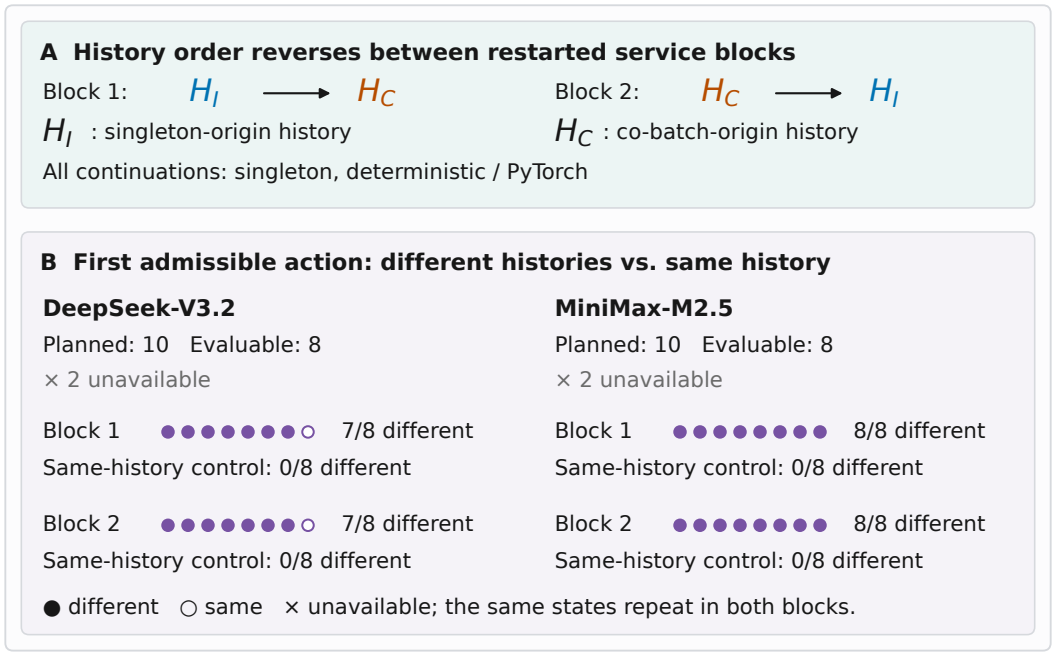

Figure 4: Supplied histories change first actions in both comparison orders. Filled/open circles denote different/same first admissible actions. Eight of ten planned states per deployment are evaluable; invalid actions and the timing exception remain missing (Appendix E). Restarted blocks reuse states, reversing history order and pair placement; the unshown half starts in the opposite order. All continuations use singleton deterministic/PyTorch serving.

We call disagreement beyond this repeat background *requirement excess*. Let X_1, X_2 be the singleton task-check vectors for task t in session s, and Y_1, Y_2 the co-batch vectors. The contrast is

$$\Delta_{t,s} = \frac{1}{4}\sum_{i,j=1}^{2} d(X_i, Y_j) - \frac{1}{2}\big[d(X_1, X_2) + d(Y_1, Y_2)\big], \tag{1}$$

where d is normalized Hamming distance. Each task value averages both sessions; complete tasks receive equal weight. Under iid repeats within condition and independence across conditions, its expectation is the mean squared change in individual requirement-pass probabilities (Appendix F.1), not a signed accuracy change or a test for all joint-distribution changes. Official success uses the same contrast on its binary outcome.

The average contrast is inconclusive on both deployments. DeepSeek's requirement excess is 0.098 (95% interval [-0.036,0.233]); MiniMax's is 0.089 ([-0.035,0.214]). Both intervals include zero and extend well above the prespecified 0.01 material scale. They neither confirm the average contrast nor rule out a meaningful effect. Four DeepSeek task values are positive and twelve zero; MiniMax has three positive, eleven zero, and two missing. The largest tasks contribute 63.5% and 62.2% of the positive sums (Figure 5).

Seven MiniMax co-batch trajectories lack complete required-window overlap evidence, leaving two task estimates and fourteen derived comparisons missing. Their saved outcomes remain visible without filling primary values. Appendix K discloses the post-output evaluator-binding repair; official tests and stored contents were unchanged.

Concrete task outcomes differ in both directions. In AppWorld's simulated Venmo, DeepSeek must send a publicly visible $13 payment request to a specified friend. All four singleton episodes reach $H = 32$ without completing and fail the checks, while all four co-batch episodes finish and pass. The payment amounts in these cases belong to fictional benchmark tasks.

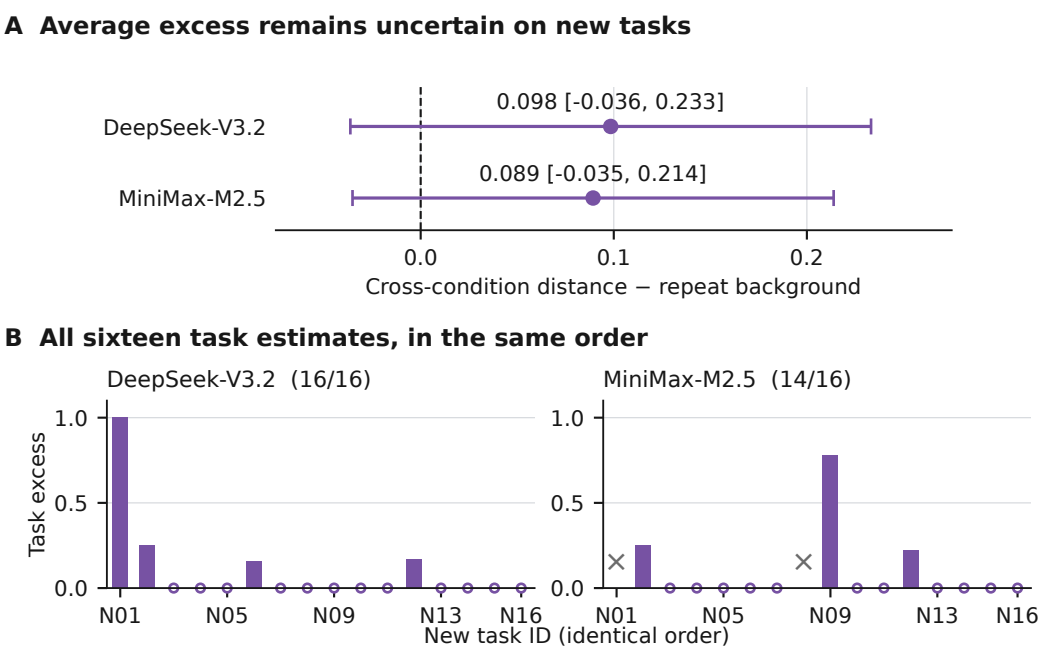

Figure 5: Positive average excess comes with intervals spanning zero and substantial task concentration. Top: deployment means and full family-cluster 95% t intervals. Bottom: all sixteen task estimates on a separate labeled scale; open circles mark zero and crosses missing. DeepSeek has sixteen complete tasks; MiniMax fourteen. Both deployment-level results remain inconclusive. Appendix G retains session values, termination categories, and success directions; Appendix L maps the display IDs.

In MiniMax's rent task, all four singleton episodes finish and pass; co-batch produces one completed wrong-request/amount outcome, one decision cap, and two context censors. The completed co-batch episode selects a lunch request and sends $63.25 instead of $805. The record identifies the wrong request and amount but does not verify a wrong recipient (Appendix H).

Differences also arise between repeats of the same condition. Several DeepSeek tasks have completed success discrepancies but zero excess after that background is subtracted. Across DeepSeek's full panel, both conditions have 48/64 successes even though binary success excess is 0.125. Equal aggregate success rates therefore conceal task-specific differences.

7 Deterministic Configurations Can Prevent Tested Forks

Can an existing deterministic configuration prevent action forks? The tested configurations eliminate all initiating forks in the controlled Qwen and DeepSeek audits. MiniMax gains action agreement in 21 of 38 estimable targets. Figure 6 retains the distinct state/context-pair units; these are separate audits, not a pooled efficacy estimate.

The MiniMax panel retains the limits alongside that benefit. All forty standard targets reproduce their fork; seventeen of the 38 estimable targets remain different, and four of 36 qualified nulls acquire a difference. Two target and four null exclusions remain outside these denominators. The comparison changes the standard/FlashInfer configuration to deterministic/PyTorch; it measures action agreement, not task accuracy or a single-kernel effect.

In dynamic-load checks on each of Qwen and DeepSeek, the previously tracked alternative action no longer recurs in three of four selected cases; the fourth produces invalid output. These checks use the standard-controlled singleton reference. Recorded configurations also increase latency and reduce

throughput (Appendix I). Deterministic serving offers action-stability benefits in tested deployments, with applicability and cost to be checked at the tool interface.

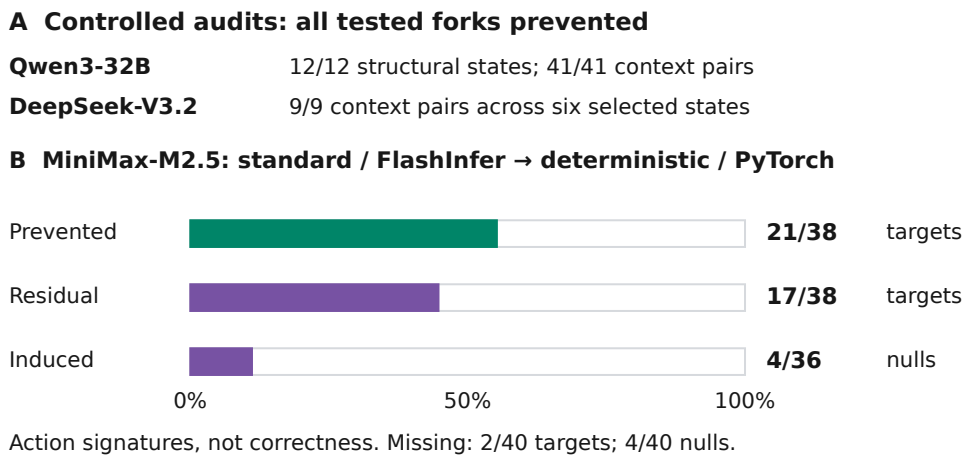

Figure 6: A deterministic configuration can prevent, retain, or introduce action differences. Qwen/DeepSeek use the named state/context-pair units. MiniMax partitions 38 estimable targets; induced uses 36 nulls. Action agreement is not correctness; the comparison changes the standard/FlashInfer versus deterministic/PyTorch configuration bundle.

8 Discussion and Conclusion

Execution determines what a serving difference means: alternative calls can produce equal transitions, database-only changes, or different answers in completed tasks. Observations and task checks distinguish these outcomes.

Application data and action–observation history carry different information. Supplied histories change behavior at common measured database contents, including under reversed orders with agreeing same-history controls.

Equal mean success can conceal opposing task-level differences; an unfinished run can fail finite-window checks. Repeated-condition baselines, actions, task checks, and termination categories reveal these distinctions. Deterministic configurations also require evaluation at the action interface, alongside their systems costs.

The evidence covers one benchmark, heterogeneous deployments, selected states, and finite windows. New tasks broaden selection but leave both average contrasts inconclusive. These panels do not estimate production incident rates.

The tested lightweight predictors and local content interventions have not yielded reliable selective stabilization (Appendix J). Separating action-history from observation effects, and measuring stability against deployment cost, remain useful scientific questions. The present evidence shows that co-batching can change executable actions and that the interaction histories left by those actions can matter beyond matched database contents.

AI USE STATEMENT

Generative AI tools assisted with research design, code development, analysis, literature review, and manuscript preparation. The authors verified all analyses, citations, and claims and take full responsibility for the work.

REPRODUCIBILITY STATEMENT

The appendices describe the experimental protocols, deployment configurations, state-selection and exclusion rules, outcome definitions, and statistical analyses. They also report per-task results, termination categories, sensitivity analyses, and exploratory studies. An anonymous replay package is available at https://anonymous.4open.science/r/agent-runtime-replay-AD9E/.

REFERENCES

Bjarni Haukur Bjarnason, André Silva, and Martin Monperrus. On randomness in agentic evals. In *ICLR 2026 Workshop on Agents in the Wild*, 2026. URL https://openreview.net/forum?id=UFukKhqCpr. Workshop paper; accompanying preprint arXiv:2602.07150v3.

Paul C. Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. Thought anchors: Which LLM reasoning steps matter? In *Mechanistic Interpretability Workshop at NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=VnSlfeRCaU. Workshop paper; accompanying preprint arXiv:2506.19143v4.

Kexin Chu, Yang Zhou, and Wei Zhang. MarginGate: Sparse margin-triggered verification for batch-invariant LLM inference. *arXiv preprint arXiv:2605.30218*, 2026. URL https://arxiv.org/abs/2605.30218.

Evelyn Duesterwald, Benjamin Elder, Lilian Ngweta, Shashanka Ubaru, and Malgorzata Zimon. Closing the consistency gap: Self-evolving agents that learn to stay on course. *arXiv preprint arXiv:2609.08832*, 2026. URL https://arxiv.org/abs/2609.08832.

Raja Gond, Aditya K Kamath, Ramachandran Ramjee, and Ashish Panwar. LLM-42: Enabling determinism in LLM inference with verified speculation. *arXiv preprint arXiv:2601.17768*, 2026. URL https://arxiv.org/abs/2601.17768. Accepted for SOSP 2026; proceedings forthcoming.

Sahil Kadadekar. A paired testing protocol for batch-conditioned refusal robustness in LLM serving. In *The ICML 2026 Workshop on Hypothesis Testing*, 2026. URL https://openreview.net/forum?id=JGUaS2N7G2. Workshop paper; accompanying preprint arXiv:2605.27763.

Guanghui Min, Liang Wu, Mayank Darbari, Chen Chen, and Liangjie Hong. Toward reliable context compression for long-horizon agents: An empirical study of execution instability. *arXiv preprint arXiv:2608.06503*, 2026. URL https://arxiv.org/abs/2608.06503.

Aditi Patodiya. Same request, different answer: Quantization amplifies cache-induced divergence in LLM serving. *arXiv preprint arXiv:2609.04748*, 2026. URL https://arxiv.org/abs/2609.04748. Submitted to IEEE Access.

SGLang Project. Deterministic inference. Official SGLang documentation, 2026. URL https://github.com/sgl-project/sglang/blob/71de97b264b04dcd514cf904003028aefe9775c8/docs/docs/advanced_features/deterministic_inference.mdx. Accessed 2026-09-18; documentation at the experiment's pinned revision.

Jaineet Shah. Causal agent replay: Counterfactual attribution for LLM-agent failures. *arXiv preprint arXiv:2606.08275*, 2026. URL https://arxiv.org/abs/2606.08275.

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, 2024. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Abel Yagubyan. How consistent are LLM agents? measuring behavioral reproducibility in multi-step tool-calling pipelines. *arXiv preprint arXiv:2605.28840*, 2026. URL https://arxiv.org/abs/2605.28840.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, volume 2025, pp. 9965–10017, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/1b126cc38b8638e07bef37e7b2bb72bf-Paper-Conference.pdf.

Jiayi Yuan, Hao Li, Xinheng Ding, Wenya Xie, Yu-Jhe Li, Wentian Zhao, Kun Wan, Jing Shi, Xia Hu, and Zirui Liu. Understanding and mitigating numerical sources of nondeterminism in llm inference. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 169819–169851. Curran Associates, Inc., 2025. doi: 10.52202/085713-5653. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/f80094a824ba5912d4a2de169c404a40-Paper-Conference.pdf.

Shiju Zhao, Jiacheng Yang, Qihang Chen, Junhao Hu, Jiaqi Zheng, Guihai Chen, and Xusheng Chen. CoRun: Padding is simple and efficient for deterministic LLM inference. *arXiv preprint arXiv:2608.14376*, 2026. URL https://arxiv.org/abs/2608.14376.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 62557–62583. Curran Associates, Inc., 2024. doi: 10.52202/079017-2000. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/724be4472168f31ba1c9ac630f15dec8-Paper-Conference.pdf.

A Experiments, Selection, and Shared States

The comparisons below ask different questions. The table names their units and overlap before giving the selection details.

Table 2: Experiments share some state identities but measure different interventions. "Selected" means selected before the continuation outcomes, not a random sample of deployment traffic.

Comparison	What changes / what is measured	Unit and relationship
Fixed request	Co-batch neighbors / action and local execution	State; five separately constructed panels
Action continuation	Executed action / later behavior and endpoint	103 selected states; 97 estimable
Discovery history	Supplied history at common DB / continuation	77 of those states; 71 estimable
Counterbalanced history (A)	History plus matched repeated-history controls / first action	20 reused state/deployment entries; two blocks each
Native tasks (B)	Serving policy during episodes / task-check excess	16 fresh tasks, shared across two deployments
Deterministic settings	Configuration / canonical-action agreement	Selected states or context pairs; units stated per audit

The controlled panels contain 400 Qwen states from sixty tasks/twenty families, 200 DeepSeek states from thirty/ten, and 200 states each for Llama, MiniMax, and GLM from 36/twelve. Development and capability tasks are excluded. Qwen's frozen hash ordering selects 400 of 402 eligible snapshots, first allowing up to fifteen per family and then filling by global hash; source trajectories

have at most ten decisions per task. DeepSeek takes the first eight eligible states per task, then uses salted family/task ordering and round-robin selection to reach 200. The other three deployments likewise allow at most eight states per task, with fixed state and companion salts. The later DeepSeek recovery retains all 248 eligible states, and the continuation repair retains its original 62-state list.

Qwen's initial screen has 1,600 requests across standard/deterministic and singleton/co-batch conditions. Another 852 repeats cover thirteen screening differences, forty fixed null controls, and execution contexts. Its corrected generation test uses two-of-three modal signatures with paired-index support. Each other controlled panel has 1,200 standard requests: two repeats of singleton and of each of two co-batch contexts. Repeats and contexts remain nested within states. Execution order is recorded, but each condition is not independently restarted or randomized.

DeepSeek's 73 executed contrasts cover 52 structurally different states. Row-content reconstruction classifies 57 contrasts as operational-clean, fourteen as operational-error-mediated, and two as execution-equivalent clean. They represent fifty operational states, forty containing a clean contrast. MiniMax's 89 executed candidate contrasts comprise fifty clean and 39 error-mediated contrasts; qualifying contrasts span fifty states, with 26 clean-containing states. These contrast totals are not additional independent states.

Invalid outputs and repeat disagreement. Among 600 standard arm pairs per deployment, DeepSeek has 590 stable-valid pairs, MiniMax 591, and Llama 598; the rest fail parser/schema qualification. GLM has 528 stable-valid and 72 repeat-disagreeing pairs. Its zero qualified operational forks has a descriptive Wilson upper bound of 0.01885, which does not bound all instability. Qwen has 2,452 parser-valid records, of which 2,024 are schema-admissible and 428 are not. Of 800 state-arm summaries per mode, 694 contain one observation, 84 repeated summaries have identical valid signatures, and 22 contain invalid records. These observations describe the background; they do not calibrate a formal false-positive rate for the two-repeat rule.

Llama's one operational state remains a positive observation even though it is below the earlier two-state/two-family continuation threshold. Separate twelve-task capability probes show valid first calls and progress on all Llama and GLM tasks, with two and three official successes, respectively. Later errors and invalid calls remain. The probes establish limited task engagement, not mastery; a panel without qualified forks likewise does not establish general stability.

MiniMax's later 250-state panel is distinct from its original 200-state screen. The expanded 41 states come from the later panel, not the original screen's 26 clean-containing states; forty are estimable. Its 27 history states are a subset of those 41. The deterministic audit also selects from the later 250: its forty targets overlap the expanded panel, including all 27 history states, whereas its forty nulls do not. The original 200 and later 250 share no saved-state identities; this does not imply independent tasks or families.

State reuse connects the selected panels (Table 2). All 77 discovery-history identities belong to the 103 action-continuation entries: fifty of 62 DeepSeek and 27 of 41 MiniMax entries. Eleven task IDs recur across the two history deployments. A reuses twenty of these state/deployment entries, adding controls rather than new state identities. B instead uses sixteen fresh task families, shared by both deployments. Its 256 trajectories are not 32 independent tasks. The study therefore has no single sample size obtained by adding all panels.

B Actions, Observations, and Task Checks

When an action difference is operational. For an action contrast c that passes generation qualification, an operational difference requires agreement within each condition and a difference between conditions:

$$Q(c) = R(c) \wedge [O_I(c) \neq O_C(c) \ \vee \ P_I(c) \neq P_C(c)].$$

R checks within-condition agreement in the checkpoint, injected action, observation, recorded state projection, changed-model list, SQL, completion, and execution class. O is the tool observation and P the panel's state projection. The corrected DeepSeek census also checks reconstructed row contents within condition. A state qualifies when at least one contrast qualifies. The initial operational difference is therefore a selection criterion; it is not counted again as evidence of later propagation.

Table 3: Different channels answer different questions. No row implies a difference in a later row.

Object	Comparison and interpretation
Canonical action	Ordered tool names and JSON arguments; admissibility is checked separately.
Tool observation	Actual ordered responses after execution, including recorded API errors.
Database content	Reconstructed application rows at a checkpoint or window end; coverage specified below.
State proxy	Original mutation-counter projection retained in the named historical panels.
Future action	At least one shared future decision with differing admissible calls.
Task-check vector	Pass/fail for each official task check; excludes evaluator-log formatting.
Official success	The saved aggregate task result, separate from the completion marker.

Execution-clean means that all four initial executions receive the clean execution class. The classifier distinguishes parseable JSON tool responses from non-JSON schema-error patterns, tracebacks, and unresolved responses. Separate API-contract checks examine action–response bindings, documented response schemas, and AppWorld's non-200 exception path. A clean response need not contain a correct answer, while a reproducible API-error contrast can still have an operational effect.

Qwen retains its original modal generation rule and state proxy. Its twelve primary contrasts include four operational-clean, six error-mediated, and two execution-equivalent clean controls; a separate observation is ineligible. The six clean contrasts belong to six states. The two controls are excluded because their local executions agree, not because of duplicate state identities. In control `de329c87`, for example, a payment request omits or supplies `private=false`. Equal observations and state proxies at that checkpoint do not imply equal subsequent behavior. DeepSeek contrast `badd27a6` illustrates the converse: observations agree, but reconstructed task-status contents differ.

Coverage of row-content reconstruction. The DeepSeek controlled census reconstructs all 292 initial executions across 73 contrasts. MiniMax's retained operational contrasts all differ in observation; a targeted row-content check leaves its three equal-observation exclusions unchanged. Qwen and Llama retain the original state proxy, with Llama's qualifying contrasts supported by observation differences; GLM has no structural candidate to execute. The 103-state expanded action-continuation and 77-state discovery-history panels reconstruct both branches and repeats, including submitted answers and task status. A/B also have saved-SQL content checks. The earlier Qwen, DeepSeek, and MiniMax continuations described in Appendix D retain proxy-based state comparisons.

Row-content comparison applies saved cumulative SQL to copies of the recorded base databases and compares ordered rows, excluding the redundant `record_hash` field. It preserves parameters, SQLite types, NULLs, statement order, inserts, and deletions. No-op updates or insert–delete sequences can therefore leave equal contents despite different mutation histories. This projection does not include unlogged interpreter, filesystem, RNG, or cache state.

Continuation eligibility. Eligibility depends on the comparison. Action continuation requires a selected clean-fork state, correct binding, repeat agreement, and an uncensored measurement. Discovery history additionally requires a nonempty, admissible, repeatable first action within each history; it does not require the histories to disagree. All six exclusions in that panel arise from repeated illegal or empty actions. The counterbalanced study (A) requires two complete blocks, while the native-task study (B) requires complete sessions and scheduler qualification. Deterministic audits measure qualified canonical-action differences; they do not re-execute every alternative to establish an operational fork.

C Deployments and Serving Settings

AppWorld provides persistent application databases, benchmark APIs, and official programmatic evaluators for fictional accounts. Table 4 gives the model repositories and serving hardware. Short revisions correspond to the full forty-character model and software identifiers in the accompanying configuration metadata.

Table 4: Controlled-panel deployment identities. Revisions are unambiguous prefixes of the recorded full identifiers. GPU/TP names the accelerator and tensor-parallel degree; these are not matched model-capability comparisons.

Model repository	Revision	Precision	GPU / TP
`Qwen/Qwen3-32B`	`9216db5781`	BF16	H100 / 1
`deepseek-ai/DeepSeek-V3.2`	`a7e62ac04e`	official FP8	H200 / 8
`meta-llama/Llama-3.3-70B-Instruct`	`6f6073b423`	BF16	H200 / 2
`MiniMaxAI/MiniMax-M2.5`	`f710177d93`	official FP8	H200 / 4
`zai-org/GLM-5-FP8`	`4f96cc5eec`	official FP8	H200 / 8

The shared serving baseline is SGLang 0.5.18, revision `71de97b264`; container digests and host settings remain in the records. Qwen uses 16K context for controlled requests and H200 for its dynamic workload. The original DeepSeek controlled panel uses 16K; MiniMax and the later DeepSeek continuation repair use 32K. Rates from these deployments cannot be attributed to model family alone.

Controlled decoding uses temperature zero, top-$p = 1$, top-$k = -1$, seed 100, and fixed stops, with neither speculative decoding nor LoRA. A singleton/co-batch pair has identical target input IDs, schema bytes, parser, renderer, decoding, and deployment. Co-batches contain four requests, with a fixed focal slot and saved companions from other tasks. Occupancy and composition change together. Configuration comparisons change the documented deterministic setting and sampling backend; action and history comparisons instead change the injected messages.

Table 5: Continuation settings in the saved runtime records. All requests are singleton, greedy, with 32K context, $H = 8$, and two repeats per arm. A also uses deterministic/PyTorch singleton continuation, with eight runs per state across two service blocks.

Panel	Deployment	States	Records	Deterministic	Sampler / TP
Expanded clean	DeepSeek	62	248	off	FlashInfer / 8
Expanded clean	MiniMax	41	164	off	FlashInfer / 4
History	DeepSeek	50	200	on	PyTorch / 8
History	MiniMax	27	108	on	PyTorch / 4

The three DeepSeek source panels serve different purposes. The original controlled screen covers 200 states from thirty tasks and ten families at 16K. A 248-state recovery covers 43 tasks and fifteen families, also at 16K. The selected continuation repair replays its entire preselected 62-state list at 32K after a context mismatch was found; the host driver also changes. This is an estimate for the repaired deployment, not a blind replication or a context-only intervention. Separate input locks and actual-stack manifests specify A and B.

What the MiniMax deterministic configuration changes. MiniMax's retained launch requests FA3, TP4, 32K context, and no radix reuse. Adding the deterministic setting also replaces FlashInfer sampling with PyTorch in the saved startup records. Those records show TF32 enabled for the gate, deterministic TP-reduction settings, and the default MoE kernel configuration. Source references support these settings, but do not attest every installed operator. Greedy sampling takes an argmax path, so the sampler name alone does not explain the observed differences. The result applies to this FP8/MoE/TP configuration.

D Following Actions Through a Finite Window

Action continuation follows one fixed primary contrast per selected clean state. DeepSeek retains 57 of 62 states, excluding three context-censored cases and two with evaluator-repeat disagreement. MiniMax retains forty of 41, excluding an action-binding mismatch before outcome inspection. The original action-sequence measure finds differences in 57/57 DeepSeek and 36/40 MiniMax states. Requiring two admissible alternatives at a shared future decision yields 55/57 and 34/40. The re-

maining DeepSeek cases compare valid with invalid outputs; the remaining MiniMax cases include early termination. The main text uses the admissible-pair measure.

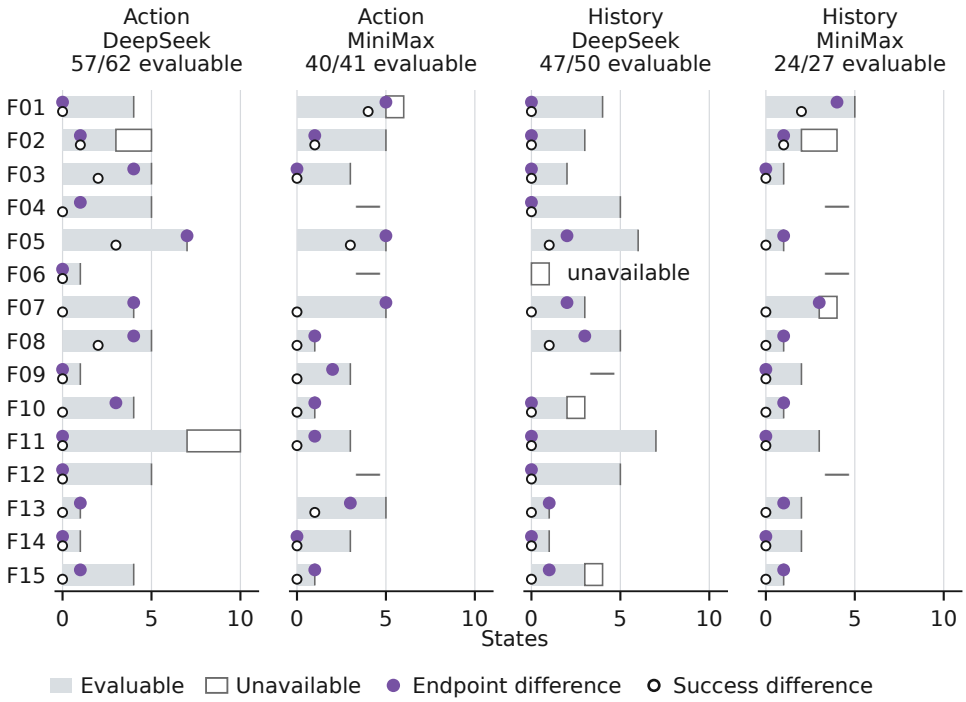

Figure 7: Families contribute different numbers of measured states and consequences. Each column is one panel/deployment; its header gives evaluable/planned states. Gray/outlined bar segments count evaluable/unavailable states. Filled purple and open black circles separately show endpoint and success differences, including zeros; a dash denotes no planned state. Their small vertical offset separates coincident counts. F01–F15 are display aliases (Appendix L); no families are merged. History reuses action-panel states. Table 6 retains completion/cap strata.

Earlier, smaller continuations use historical state proxies: twelve Qwen states, 24 DeepSeek states, and twelve MiniMax states. The latter panels report eighteen and eleven immediate proxy-equal states, respectively; these labels do not certify row-content equality. Twelve separate MiniMax matched-null continuations remain repeat-stable, with no reported future-action, evaluator, or proxy-state difference. They precede the common-database history treatment. Qwen's execution-equivalent default-argument control can still change later behavior, illustrating how representation can matter after an initially equal local transition.

Discovery history restores the isolated-derived database E^*, then supplies the common message prefix followed by either branch's canonical action and actual observations. Original reasoning is omitted. Reconstructing the saved SQL after each initiating action verifies equal source-branch row contents in all 77 states (fifty DeepSeek, 27 MiniMax), before the history intervention. These contents match all 308 restored continuation inputs. The twenty A entries belong to this same set, with matching source histories and common-database fingerprints. This checks database rows, not complete process state. The discovery order is fixed at C_1, C_2, I_1, I_2. Fields containing simulated AppWorld time cannot establish wall-clock order; the recorded completion sequence supports the fixed order, but not a full independent restart chronology. The later counterbalanced study adds explicit ordering and clock controls.

Within each supplied history, all 77 states repeat on the recorded action, observation, database, task-check, and success channels. Six states fail the admissible-first-action rule: two MiniMax states fail in both histories and four states in one. Among the 71 eligible states, first actions differ in 41/47 DeepSeek and 19/24 MiniMax states. Wilson 95% intervals are [0.748, 0.940] and [0.595, 0.908]; the original family-bootstrap intervals are approximately [0.795, 0.943] and [0.560, 1.000]. These intervals concern first actions, not endpoints. No history state is context-censored at 32K, although many reach the eight-decision cap.

Table 6: Endpoint differences can occur with both branches complete or with a decision cap. "Done" means both arms record completion, irrespective of success; "cap" means at least one reaches $H = 8$. Endpoint is database-or-evaluator difference. Success rows are subsets. $I \to C$ means isolated-origin success and co-batch-origin failure, with the reverse in the final column; these are source arms under a common continuation policy.

Panel	Endpoint done / cap	No endpoint done / cap	Success differs done / cap	$I \to C$	$C \to I$
History / DeepSeek	2 / 7	9 / 29	1 / 1	2	0
History / MiniMax	4 / 9	8 / 3	2 / 1	1	2
Expanded clean / DeepSeek	4 / 22	5 / 26	1 / 7	4	4
Expanded clean / MiniMax	16 / 9	12 / 3	9 / 0	3	6

Fifteen of the 51 action-continuation endpoints differ only in database contents, with no established consequence for an official task check. The seventeen success discrepancies span six DeepSeek tasks/four families and seven MiniMax tasks/four families. History's five success discrepancies span five tasks/four families; the three both-complete pairs span three tasks/two families.

Both-complete subsets describe observed outcomes, but conditioning on completion does not define a separate randomized comparison.

Table 7: Posthoc family-cluster bootstrap intervals for the older selected panels. Fractions use eligible states; bracketed 95% intervals are percentages. Tasks/families refer to that eligible set.

Panel	Tasks/families	Endpoint [%]	Evaluator [%]	Success [%]
History / DeepSeek	26/13	9/47 [5.6, 34.8]	7/47 [2.4, 29.4]	2/47 [0.0, 9.4]
History / MiniMax	19/12	13/24 [27.3, 77.3]	8/24 [17.4, 50.0]	3/24 [0.0, 25.8]
Expanded / DeepSeek	30/15	26/57 [21.6, 69.1]	19/57 [15.4, 52.0]	8/57 [3.1, 25.4]
Expanded / MiniMax	28/12	25/40 [37.5, 85.0]	17/40 [25.0, 59.5]	9/40 [3.3, 41.3]

The older-panel intervals use 10,000 percentile-bootstrap draws with seed 70720260904, preserving states within families. The procedure was fixed before its first successful computation, after outcomes were known. Shared families across deployments remain one cluster in combined descriptions; the action and history panels are never pooled. A family with no eligible state contributes no zero value. Combined endpoint intervals are 32.3–71.4% for 51/97 action-continuation states and 15.1–47.5% for 22/71 history states. Leaving out each family gives ranges 45.9–57.5% and 26.2–36.1%. These are descriptive intervals for selected states, not population failure rates.

E Counterbalanced History Comparisons: Full State Results

The counterbalanced study (A) contains twenty state/deployment entries, ten per deployment, representing seventeen distinct tasks and thirteen families. Each deployment covers ten families. Two restarted service blocks provide a cross-history pair and a same-history pair per state, giving eight continuations per state and 160 final records. Assignments were determined from identity metadata before the formal outputs. In each block, five states use either history order and five either cross-history-first/same-history-first placement. Both assignments reverse in block two. All requests are singleton deterministic/PyTorch; isolated-origin and co-batch-origin refer to the supplied histories.

DeepSeek-V3.2

Task / decision	Block 1 order · first	Block 2 order · first
F01.2 / 5	C→I S ● ○	I→C M ● ○
F02.2 / 2	C→I M ● ○	I→C S ● ○
F05.1 / 7	C→I S ● ○	I→C M ● ○
F06.3 / 1	I→C M × ○	C→I S × ×
F08.3 / 7	C→I M ● ○	I→C S ● ○
F10.1 / 7	I→C S ○ ○	C→I M ○ ○
F11.1 / 1	I→C M ● ○	C→I S ● ○
F12.3 / 2	C→I S ● ○	I→C M ● ○
F14.3 / 7	I→C S ● ○	C→I M ● ○
F15.3 / 1	I→C M × ×	C→I S × ○

MiniMax-M2.5

Task / decision	Block 1 order · first	Block 2 order · first
F01.1 / 5	I→C S ● ○	C→I M ● ○
F02.2 / 3	I→C M × ×	C→I S × ×
F03.3 / 3	C→I M ● ○	I→C S ● ○
F05.3 / 1	C→I S **T** ○	I→C M ● ○
F08.3 / 3	I→C S ● ○	C→I M ● ○
F09.1 / 1	C→I M ● ○	I→C S ● ○
F10.2 / 5	C→I S ● ○	I→C M ● ○
F11.3 / 2	I→C M ● ○	C→I S ● ○
F13.2 / 2	C→I S ● ○	I→C M ● ○
F15.3 / 6	I→C M ● ○	C→I S ● ○

Within each block: left marker = cross-history (M); right = same-history (S).

Filled / open: different / equal. × invalid action; T unmatched 71.44 s gap.

I / C: isolated / co-batch origin. M / S after the arrow identifies the pair run first.

Figure 8: Every planned history state remains visible, including incomplete pairs. Columns give the history order and which pair runs first in each service block. Within a cell, the left marker describes the cross-history pair and the right the same-history pair; filled/open means different/equal first admissible actions. Crosses mark invalid actions and T a mismatched wall-clock gap. The blocks reuse the same states. The valid second block for MiniMax task F05.3 does not enter the complete-state primary denominator.

For each state, the primary outcome subtracts first-admissible-action disagreement in the same-history pair from disagreement in the cross-history pair, then averages the two blocks. Paired first-generation wall-clock gaps must be within twelve seconds of 240 for DeepSeek and eighteen seconds of 360 for MiniMax. Records bind service identity, history hashes, starting database contents, and actual timestamps. Both blocks must qualify. An available second block remains visible when the first is missing, but does not supply a complete primary state.

DeepSeek has sixteen qualified state-blocks and eight complete states. F06.3 and F15.3, both at decision 1, have illegal or empty first actions. MiniMax has seventeen qualified state-blocks but eight complete states. Its F02.2, decision 3, has invalid first actions; F05.3, decision 1, has a first-block gap of 71.444739 seconds instead of the required 360, although its second block qualifies. Figure 8 preserves every state and both blocks.

The twenty cross-history pairs per deployment also allow a descriptive continuation summary, including pairs excluded from the first-action analysis. DeepSeek has two database and two task-check differences, no success differences, and six both-complete pairs versus fourteen capped pairs. MiniMax has twelve database, ten task-check, and four success differences, with ten both-complete and ten capped pairs. Eighteen pairs per deployment have an observed admissible-trajectory difference and two are unknown; observations differ in all twenty. These pair counts do not add independent states.

All twenty same-history pairs per deployment agree in observations, database contents, task checks, and success. Full admissible-action sequence agreement is established for sixteen DeepSeek and fourteen MiniMax pairs; the rest contain an illegal action during continuation. Twenty longer-lag same-history comparisons across blocks also agree on the recorded channels, with the same sixteen/fourteen admissible-sequence counts. They supplement the time-matched controls without replacing them.

The posture-streak example. MiniMax task F01.1, decision 5, asks for the longest good-posture streak; the benchmark answer is 4. At the earlier fork, the isolated-origin history reads note 11, while the co-batch-origin history reads notes 11–15. The returned dates and posture flags in Figure 1 are excerpts of those saved responses. Across both blocks' cross-history and same-history variants, four isolated-origin continuations answer 6 and finish at relative decision 8; four co-batch-origin continuations answer 4 and finish at decision 6. All eight finish naturally under the same model configuration. They repeat one selected state. The case records separately identify the earlier fork, common database restoration, substituted history, and new continuation.

F Uncertainty and Missing-Value Sensitivity

For complete primary values z_i with equal row weight, let $\bar{z}$ be their mean and $U_g = \sum_{i \in g}(z_i - \bar{z})$ the family score. The prespecified standard error is

$$\widehat{\mathrm{SE}}(\bar{z}) = \sqrt{\frac{G}{G-1}\frac{\sum_g U_g^2}{n^2}}, \qquad \bar{z} \pm t_{G-1,0.975}\widehat{\mathrm{SE}}(\bar{z}).$$

A contributes complete two-block state entries and B complete two-session tasks, with deployment-specific results in the main text. Shared families remain one cluster in pooled descriptive output. The t limits are left unclipped, including A's upper limit above one. Missing pairs stay missing; an available block or session does not replace a complete primary value.

Table 8: Primary estimates and family-cluster 95% t intervals. Denominators count complete state entries in A and complete tasks in B. MiniMax A's [1,1] is a consequence of zero empirical variance among eight selected states, not population certainty.

Panel / deployment	Complete / planned	Mean excess	95% interval
DeepSeek / A	8/10	0.875000	[0.579422, 1.170578]
DeepSeek / B requirements	16/16	0.098410	[-0.036413, 0.233233]
DeepSeek / B success	16/16	0.125000	[-0.028824, 0.278824]
MiniMax / A	8/10	1.000000	[1.000000, 1.000000]
MiniMax / B requirements	14/16	0.089286	[-0.035294, 0.213865]
MiniMax / B success	14/16	0.107143	[-0.059990, 0.274276]

The bounded-family sensitivity uses half-width $\sqrt{2\log(40)\sum_g w_g^2}$, assuming independent family clusters with weights w_g and outcome range $[-1, 1]$, then intersects the interval with that range. Planned-set bounds assign every missing primary value the full range, without a random-missingness assumption. Leave-one-family-out summaries retain every deletion (Table 9). The prespecified material scales, 0.20 for A and 0.01 for B, aid interpretation rather than authorize sample expansion. B's support rule requires a positive primary lower bound, adequate disclosed coverage, and no identified manipulation or binding defect. An informative-small-effect result requires an upper bound below the material scale. Both deployments remain inconclusive. A's primary intervals support a selected-state history contrast, with broader sensitivity intervals and small family counts limiting population inference.

A combined B description contains thirty complete deployment-task rows from the same sixteen families. Requirement excess is 0.094152, with interval $[-0.000668, 0.188972]$; success excess is 0.116667, with $[-0.002547, 0.235880]$. This description is also inconclusive and does not create thirty independent tasks. The two studies are not pooled, and no mediated proportion is estimated.

Table 9: Sensitivity to clustering and missing values, using the prespecified summaries. B's two missing MiniMax task values remain in the sixteen-task planned bounds.

Panel / deployment	Bounded family	Planned-set bounds	Leave-one-family-out
DeepSeek / A	[-0.085323, 1.000000]	[0.500000, 0.900000]	[0.857143, 1.000000]
DeepSeek / B requirements	[-0.580641, 0.777461]	[0.098410, 0.098410]	[0.038304, 0.104971]
DeepSeek / B success	[-0.554051, 0.804051]	[0.125000, 0.125000]	[0.066667, 0.133333]
MiniMax / A	[0.039677, 1.000000]	[0.600000, 1.000000]	[1.000000, 1.000000]
MiniMax / B requirements	[-0.636650, 0.815222]	[-0.046875, 0.203125]	[0.036325, 0.096154]
MiniMax / B success	[-0.618793, 0.833079]	[-0.031250, 0.218750]	[0.038462, 0.115385]

F.1 What the native-task contrast measures

For one task/session with K binary task checks, let $X_1, X_2 \in \{0,1\}^K$ be singleton outcomes and Y_1, Y_2 co-batch outcomes. Equation 1 uses $d(u,v) = K^{-1}\sum_{k=1}^{K} \mathbf{1}\{u_k \neq v_k\}$. Suppose X_1, X_2 are independent draws from one distribution, Y_1, Y_2 from another, and the two conditions are independent. Write $p_k = \Pr(X_{ik} = 1)$ and $q_k = \Pr(Y_{jk} = 1)$. For check k,

$$\begin{aligned}\mathrm{E}[\mathbf{1}\{X_{ik} \neq Y_{jk}\}] &= p_k + q_k - 2p_kq_k,\\ \mathrm{E}[\mathbf{1}\{X_{1k} \neq X_{2k}\}] &= 2p_k(1-p_k),\\ \mathrm{E}[\mathbf{1}\{Y_{1k} \neq Y_{2k}\}] &= 2q_k(1-q_k).\end{aligned}$$

The four cross-condition distances have the same expectation. Subtracting half the sum of the two within-condition distances therefore gives

$$\mathrm{E}[\Delta] = \frac{1}{K}\sum_{k=1}^{K}\left[p_k + q_k - 2p_kq_k - p_k(1-p_k) - q_k(1-q_k)\right] = \frac{1}{K}\sum_{k=1}^{K}(p_k - q_k)^2.$$

Linearity requires no independence between checks within a trajectory. If check probabilities differ between sessions, each session has this interpretation under its own assumptions, and averaging sessions averages those quantities. The identity explains the metric; it does not certify temporal independence in the observed runs. It measures changes in marginal check-pass probabilities, not a net accuracy loss or every change in their joint distribution. Finite-sample excess may be negative or uncertain.

G New Tasks: Session Values, Termination, and Success

The native-task study (B) selects one task from each of sixteen fresh families by prespecified identity hashes, without conditioning on source-run success or a fork. Each task has two sessions on each deployment, with two singleton and two co-batch episodes per session. The 256 final trajectories use $H = 32$, 32K context, and up to 3,000 generated tokens per request. Subsequent inputs retain the original assistant content and tool responses. Scheduler logs must establish overlap in the required windows; simultaneous client submission alone is insufficient. Task-check vectors follow the frozen benchmark test order.

Seven of MiniMax's 64 co-batch trajectories lack complete overlap qualification: four for N01 and three for N08. One co-batch trajectory of the latter task is observed, but neither task has a complete primary session. Fourteen cross-condition comparisons remain missing. Across all 128 MiniMax trajectories, 108 complete naturally, twelve reach the decision cap, and eight are context-censored. The 121 qualified trajectories contain 101 completions, twelve caps, and eight context censors. DeepSeek has 113 completions, fourteen caps, and one context censor, all qualified. Table 11 includes raw outcomes of unqualified records, without using them to fill primary estimates.

H What the Selected Task Differences Mean

The cases below interpret changed checks in terms of the task's requested operation. All payment amounts belong to fictional AppWorld tasks. Case interpretation does not alter sampling or the primary decision rule. Analysts could see model, condition, and outcome identities; this was not a blinded human evaluation. Table 10 retains the complete task set, including every zero-excess task.

Table 10: All sixteen native tasks, labeled N01–N16 as in Figure 5 and aligned across deployments. s_1, s_2 are session requirement excess; R and S are complete-task requirement and success excess. A dash means missing, not zero. Values are rounded only for display.

	DeepSeek				MiniMax			
Task	s_1	s_2	R	S	s_1	s_2	R	S
N01	1.000	1.000	1.000	1.000	–	–	–	–
N02	0.500	0.000	0.250	0.500	0.000	0.500	0.250	0.500
N03	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
N04	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
N05	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
N06	0.316	0.000	0.158	0.500	0.000	0.000	0.000	0.000
N07	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
N08	0.000	0.000	0.000	0.000	–	–	–	–
N09	0.000	0.000	0.000	0.000	1.000	0.556	0.778	1.000
N10	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
N11	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
N12	0.333	0.000	0.167	0.000	0.000	0.444	0.222	0.000
N13	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
N14	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
N15	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
N16	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 11: B trajectory counts by input condition. "Qualified" refers to native-prompt/overlap evidence. Completion, cap, context, and success columns describe all recorded trajectories, including unqualified MiniMax records. Success is a separate official check and does not partition termination.

Condition	Runs	Qualified	Complete	Cap	Context	Success
DeepSeek / singleton	64	64	56	8	0	48
DeepSeek / co-batch	64	64	57	6	1	48
MiniMax / singleton	64	64	52	8	4	36
MiniMax / co-batch	64	57	56	4	4	36

Table 12: Directions within B's four cross-condition pairs per task-session. These are dependent derived comparisons. S and C denote singleton and co-batch success. The both-complete subset is descriptive, because completion is post-treatment.

Scope	Both pass	Both fail	S fails, C passes	S passes, C fails	Missing
DeepSeek / all cross pairs	82	18	14	14	0
DeepSeek / both complete	82	8	4	6	0
MiniMax / all cross pairs	56	46	4	8	14
MiniMax / both complete	56	26	0	2	14

In DeepSeek's earlier dinner-payment task F05.2, decision 2, both histories finish. One sends a single $30.50 transfer; the other sends $12.25 and $18.25. The request asks for a payment, while the evaluator also requires one transaction of the specified amount. The totals agree, but the checks differ. This supports a transaction-structure difference, not an arithmetic error or economic loss. At another saved state of the same task, both histories succeed despite different behavior; that counterexample remains in the casebook.

DeepSeek task N01 asks the agent to send a publicly visible $13 payment request to a specified friend in AppWorld's simulated Venmo. Four singleton episodes remain incomplete at the decision budget and fail the task checks; four co-batch episodes finish and pass. Its requirement excess is one and contributes 63.5% of the positive sum. The outcome combines a task-check difference with a finite-budget completion difference.

MiniMax task N09 asks the agent to deny request 6097 and pay $805 after a 15% increase. All four singleton episodes finish and pass. One co-batch episode finishes after selecting request 6100 ("ForLunch", $55), receiving an authorization error for the attempted denial, and sending $63.25. The remaining co-batch episodes include one cap and two context censors. This task contributes 62.2% of the positive sum. The record identifies the wrong request and amount, but does not verify a wrong recipient.

In DeepSeek's invoice/receipt task N06, the first session has two completed singleton failures and two completed co-batch successes. The second session does not reproduce this contrast. Tasks N08, N11, and N14 also include completed success discrepancies, but their requirement excess is zero after subtracting repeated-condition background. A differing pair alone therefore does not establish an excess serving-policy contrast.

Other changed checks have narrower meanings. Task N02 requests a changed set of followed artists. Task N12 includes a failed nonempty-updated-list assertion, without evidence of an unrelated-field mutation. In N11, a check requires a header beginning with `#VenmoExpenses`; failure does not establish monetary loss. Task N08 concerns the recipient of an activation notification under a conditional request. Its illustrative MiniMax record cannot replace the missing overlap evidence.

I Configuration Comparisons and Dynamic Workloads

The controlled configuration audits prevent Qwen's twelve selected structural state forks and all 41 context-pair forks, and DeepSeek's nine selected context-pair forks across six states. MiniMax's eighty-state panel instead contains forty targets and forty nulls. Of 38 qualified targets, 21 become equal and seventeen remain different. Four of 36 qualified nulls acquire a difference. Two targets fail deterministic-arm qualification. Four nulls are excluded: two already fork under the standard configuration, one fails standard qualification, and one fails deterministic qualification. They are not stable negative controls.

Dynamic workloads draw from frozen companion pools with Poisson arrivals at four requests per second and maximum concurrency eight. Each deployment contributes 480 standard records: twenty schedule repetitions for twelve candidates and twelve matched nulls. Known candidate alternatives recur in 9/12 Qwen, 4/12 DeepSeek, and 7/12 MiniMax cases. Nonbaseline alternatives recur in 0/12, 1/12, and 3/12 nulls, respectively. These are recurrence checks for selected alternatives, not estimates of their production prevalence.

A further 960 deterministic records compare paired Qwen and DeepSeek schedules. Under deterministic serving, the previously tracked alternative action no longer recurs in three of four selected cases per deployment; the fourth produces invalid output. Recurring nonbaseline signatures appear in 8/12 Qwen and 2/12 DeepSeek cases, where the reference is the standard-controlled singleton output. This reference does not directly test singleton-versus-dynamic disagreement within deterministic mode.

MiniMax throughput is total batch completion tokens divided by sequential-cell wall time. It measures the recorded workload, not a production load test. Neither these systems costs nor action agreement determines answer correctness.

Table 13: Recorded systems costs of the tested deterministic configurations. Workloads differ; rows are not a model ranking. Cold model loading is excluded.

Deployment	Mean latency change	Throughput change	Workload
Qwen3-32B	$+7.7\%$	-17.0%	dynamic arrivals
DeepSeek-V3.2	$+173.8\%$	-62.0%	dynamic arrivals
MiniMax-M2.5	$+78.5\%$	-46.7%	sequential formal cells

J Exploratory Prediction and Local Interventions

Post-fork prediction. The post-fork models use 48 development contrasts and ridge penalty 1.0, with coefficients fixed before prospective propagation labels. M0 uses the inverse within-deployment empirical percentile of the tenth-percentile chosen-minus-best-alternative log-probability gap in the action envelope. This feature is available after generation and before execution; calibration uses unlabeled target margins. M1 uses development-standardized ontology scores for persistent mutation, irreversibility, and action complexity. M2 combines M0 and M1. The constant baseline is development prevalence. Branch distance and downstream observations are excluded.

The prospective evaluation has 97 states and 44 events under the original mutation-counter-or-evaluator target. M2's Brier score is 0.27108, compared with M0's 0.26516. The improvement is -0.005920, with family-bootstrap 5th–95th percentiles $[-0.02109, 0.00953]$ from 10,000 draws, seed 60120260901. M0 also loses to the constant, with constant-minus-M0 Brier -0.002844. M2's point gain is 0.00064 on DeepSeek and -0.01526 on MiniMax; every combined leave-one-family-out improvement is negative. These models retain their original target and have not been refitted or retested on the corrected content endpoint.

Predicting the first action fork. The separate onset analysis has 974 eligible state–context rows from 495 states, with 215 positive rows and 21 invalid/unstable rows label-missing. It compares constant, geometry, margin, margin-plus-geometry, and limited ridge-logistic predictors. Geometry describes batch/focal lengths and log-length dispersion; limited online features add action complexity and schema-size rank. Transforms and coefficients are cross-fitted by family across deployments, keeping a state's contexts together. The family bootstrap uses 10,000 draws with seed 70720260904. Brier gain 0.009986 falls below the exact 0.01 threshold; recall at a 20% budget is 0.330 rather than 0.40, recall gain is 0.088 rather than 0.10, and one family crosses the adverse-family boundary of -0.005. Because these prespecified prediction criteria were not met, held-out confirmation and selective mitigation were not performed. A post-generation margin can inform pre-execution interception, but is unavailable for pre-scheduling routing.

Paired content transplant. The exploratory content-transplant experiment starts from completed singleton and co-batch branches. It substitutes either the last 24 reasoning tokens before a tool envelope or the preceding 24-token span, rebuilds the prefix, and regenerates the call. The isolated branch supplies a consistency reference, not a correct-answer label. All 576 planned records across 24 states are retained. Among twelve qualified positive cases that pass their own sham check, the tail transplant prevents 7/12 forks and the adjacent control 1/12.

Sham fidelity and cost checks prevent interpreting this contrast as a validated method. DeepSeek shams pass 9/12: two actions differ, and one action-faithful generation fails execution. MiniMax passes 9/11, with one additional generation-incomparable state. The deterministic MiniMax reference has twelve planned states, ten comparable and two with forks, including one originally equal-action control. DeepSeek has twelve comparable generations with no forks, but only eleven execution-clean comparisons. Missing comparisons stay missing.

Candidate-to-adjacent generated-suffix token ratios are 1.100 for DeepSeek and 1.231 for MiniMax, above the specified 5% tolerance. Changed content can itself change length, so these ratios do not establish a computation mechanism. They also omit the reference branch, prefix reconstruction, and companions. In the preselected MiniMax notes-search case F01.3, decision 6, the adjacent splice breaks a sentence, while the tail restores a query argument and generates 185 versus 174 suffix tokens. The case reveals a coherence and semantic-completeness confound; it does not show that every adjacent control is defective. The planned sixteen-state confirmation was not performed

because the sham-fidelity and cost conditions were not met. No subsequent $H = 8$ trajectory, evaluator, or success result is available, and no online numerical guard is validated.

K Evaluator Binding and Measurement Corrections

During the native-task study, official evaluation could leave a shared SQLModel class bound to the start database instead of the existing output database. Eight MiniMax records had finished with terminal-only evaluation when the ninth trajectory, N07 singleton repeat 1, triggered a guard at decision 11. The defect concerned a model-collection binding; the records do not show an application-row rollback.

The repair reselects the existing output database after every evaluation, including exception paths. Subsequent reads use the corrected binding; stored rows, official tests, and labels are unchanged. The eight earlier terminal-only records remain because no later action could read the stale binding. The failed attempt is preserved, including its absent final-step payload; one retained retry supplies the planned identity's final record. Later B trajectories on both deployments use the repaired path, while A is unchanged. The repair followed visible outputs and is disclosed as such. Offline validation reads those saved evaluations rather than rerunning them.

Other corrections address separate measurement problems. A continuation runner could alias a secondary contrast by snapshot, so the binding check excluded affected records before outcome inspection. The first DeepSeek continuation used a 16K ceiling despite a 32K specification; its repair preserved the full 62-state list. Discovery-history rescoring corrected empty-action eligibility and excluded evaluator-log formatting without altering the 308 trajectories. A tokenizer correction reconstructed features from saved token IDs without changing generated outputs or target labels.

Row-content reconstruction replaced mutation counters across all 180 expanded-action and discovery-history state entries. The resulting corrections are DeepSeek controlled operational/clean counts 49/39 to 50/40, history endpoints 21 to 22, and action-continuation endpoints 44 to 51. Official evaluator and success labels remain unchanged, as do the historical predictor targets.

L AppWorld Task and Scenario Identifiers

F01–F15 label AppWorld scenarios (task families) in the selected-state panels. Fxx.j identifies a task in that family, preserving the original suffix j; for example, F01.1 is `0a9d82a_1`. N01–N16 label the specific tasks in the native-task study (Figure 5; Table 10). Table 14 maps these paper labels to AppWorld task identifiers or their scenario prefixes.

Table 14: Two independent mappings to AppWorld identifiers. Left: selected-state families. Right: native-study tasks. Entries on the same row do not indicate membership or pairing.

Paper family label	AppWorld scenario/family ID	Paper task label	AppWorld task ID
F01	`0a9d82a`	N01	`024c982_1`
F02	`270f1ff`	N02	`09b0ee6_2`
F03	`29a7b7e`	N03	`0de03ea_1`
F04	`2c544f9`	N04	`13547f5_1`
F05	`2d9f728`	N05	`425a494_3`
F06	`31dc501`	N06	`6b6ca61_1`
F07	`32616b5`	N07	`6f4b9a5_3`
F08	`3b8fb7a`	N08	`9016950_1`
F09	`522e5e5`	N09	`9dabbc9_3`
F10	`634f342`	N10	`a30375d_3`
F11	`83a7951`	N11	`b6d1104_1`
F12	`b9c5c9a`	N12	`bde252e_3`
F13	`c77c005`	N13	`cef9191_2`
F14	`d194965`	N14	`d6ac34d_3`
F15	`f861c32`	N15	`f323bae_1`
		N16	`f3f60f0_1`