DYNAEVAL: LARGE LANGUAGE MODELS STRUGGLE TO TRACK EVOLVING TASK STATES

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model evaluation has largely focused on static, single-turn tasks. Real-world interactions, however, often involve multi-turn conversations in which task requirements evolve. We evaluate model performance in this setting and find that accuracy declines across dynamic multi-turn conversations. We attribute this decline to limitations in models' ability to reconstruct the current task state from information dispersed across the conversation history. To evaluate this ability, we introduce DYNAEVAL, a benchmark that simulates dynamic, long-horizon task environments. DYNAEVAL comprises 543 eleven-turn sequences across mathematics, code, and databases, and provides two aligned versions of each turn: one presents an incremental update to the task conditions, while the other provides a complete description of the conditions valid after that update. Both versions share the same target question and reference answer. DYNAEVAL comprises 543 eleven-turn sequences across mathematics, code, and databases. Under the full-history protocol with incremental updates, all ten models exhibit lower final-turn than first-turn accuracy, with the mean falling from 67.9% to 32.9%. Further diagnostic experiments attribute this cross-turn accuracy decline primarily to models' difficulty in recovering the currently valid task conditions from information dispersed across prior interactions, revealing a bottleneck in multi-turn conversations and long-horizon tasks.

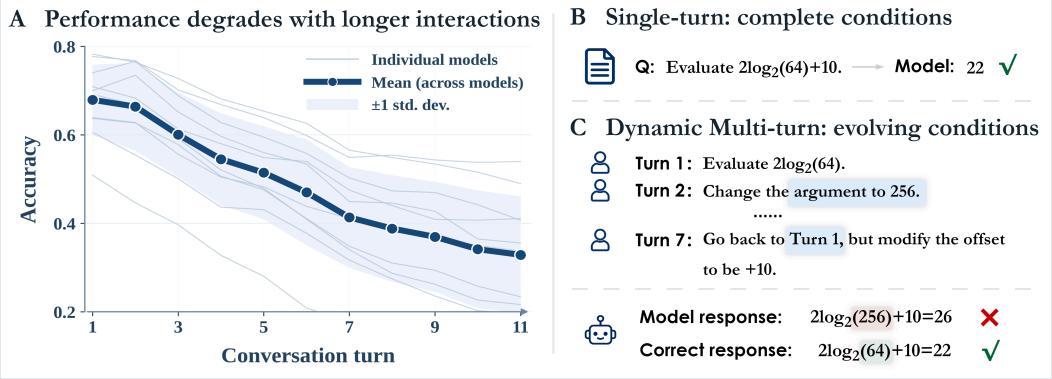

Figure 1: Accuracy declines across all evaluated models under full-history protocol (left). Single-turn tasks provide complete conditions, and dynamic multi-turn tasks revise the effective state (right). The rollback example shows a model retaining an exponent from a wrong state, producing an incorrect answer despite correct arithmetic.

1 INTRODUCTION

Recent advances in large language models (LLMs) have led to the emergence of a wide range of conversational models (OpenAI, 2023; Anthropic, 2024; DeepSeek-AI, 2024; Yang et al., 2024; Grattafiori et al., 2024). In practice, their use often involves extended multi-turn interactions, including sustained conversations with users and repeated exchanges with environments during task

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.

execution. As shown in Figure 1, Solving these long-horizon tasks often involves integrating new information and task updates with prior context, maintaining an up-to-date understanding of the task, and responding according to the currently valid conditions. Yet prevailing LLM evaluation practices remain centered on single-turn tasks with fixed, fully specified conditions.

Recent work has begun to evaluate models in long-horizon tasks and multi-turn conversations. Laban et al. (2026) reveal sharded instructions across turns, requiring models to integrate progressively revealed information under a fixed task specification. But the underlying task specification remains fixed. SR-Eval (Zhan et al., 2025) examines evolving requirements in code generation, focusing on models' ability to iteratively revise code as requirements change. MemOps (Hao et al., 2026) evaluates memory operations and state trajectories across interactions, examining how models maintain and update memory over time. While these evaluations extend across multiple turns, the causes of model performance degradation in long-horizon tasks and multi-turn conversations remain insufficiently explored. We suggest that this degradation may stem from models' difficulty reconstructing the currently valid task state from information dispersed across the interaction history, particularly when frequent changes produce easily confused versions of task conditions.

To investigate this, we introduce DYNAEVAL, a benchmark of 543 eleven-turn sequences across mathematics, code, and databases, with aligned incremental and self-contained presentations of each evolving task. Across ten models, mean accuracy under incremental updates falls from 67.9% at the first turn to 32.9% at the final turn. We investigate this degradation by tracing the parameter versions identifiable in model responses and intervening on the state information available at each turn. Supplying the currently effective state restores performance close to the Oracle reference across seven models while preserving the original conversation history. Additional controls on two models show substantially smaller gains from reconstruction reminders or unresolved historical recaps. Together, these findings identify effective-state reconstruction as a major source of degradation in dynamic multi-turn interactions.

Our main contributions are as follows:

- We introduce DYNAEVAL, a multi-turn benchmark with two aligned versions of each turn. Its paired incremental and self-contained test sets enable fine-grained evaluation of state reconstruction and task solving.
- We evaluate ten leading proprietary and open-weight models on DYNAEVAL, revealing substantial first-to-final accuracy declines across all evaluated models in dynamic multi-turn conversations.
- We identify difficulty recovering currently valid task conditions from dispersed interaction history as a primary source of this degradation, exposing a state reconstruction bottleneck in multi-turn conversations and long-horizon tasks.

2 RELATED WORK

Earlier language models, including BART, GPT-2, and T5, were developed and evaluated primarily through static, single-turn tasks (Lewis et al., 2020; Radford et al., 2019; Raffel et al., 2020). The same premise underlies the dominant evaluation practice for LLMs, from holistic multi-scenario suites (Liang et al., 2023) to reasoning, code, and text-to-SQL benchmarks whose inputs are fully specified and whose answers are independently verifiable (Cobbe et al., 2021; Hendrycks et al., 2021; Chen et al., 2021; Yu et al., 2018; Li et al., 2023). As LLMs became conversational assistants, MT-Bench and MT-Eval extended evaluation to multi-turn interactions (Zheng et al., 2023; Kwan et al., 2024). Later benchmarks broadened the range of conversational abilities under test, including instruction retention, inference memory, versioned editing, and multi-turn tool use (Bai et al., 2024; Sirdeshmukh et al., 2025; Wang et al., 2024). We view these benchmarks as largely episodic: each turn defines a locally self-contained subtask whose response can be evaluated in isolation. Laban et al. (2026) instead split a complete task across multiple turns, exposing the difficulty of integrating dispersed information. Nevertheless, the underlying task specification remains fixed: it is progressively disclosed rather than changed. This leaves a mismatch with real interactions, where the state of a user's initially proposed task is not fixed, but may be revised, overturned, or withdrawn over time (Herlihy et al., 2024; Baumann et al., 2026). DynaEval centers this dynamic regime by evolving the operative task state throughout the dialogue.

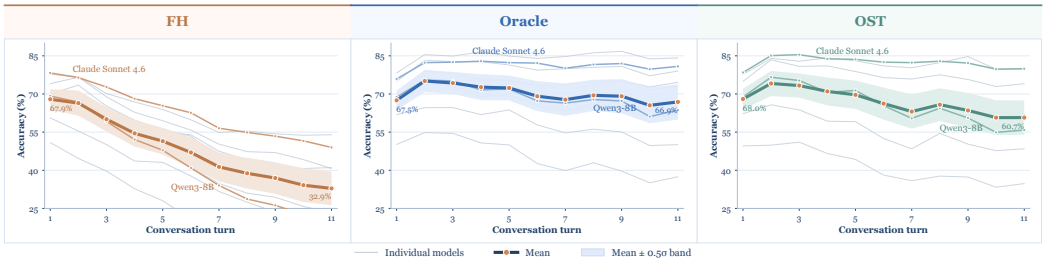

Figure 2: FH, Oracle, and OST accuracy across ten models. Thin lines denote individual models; thick lines denote model means.

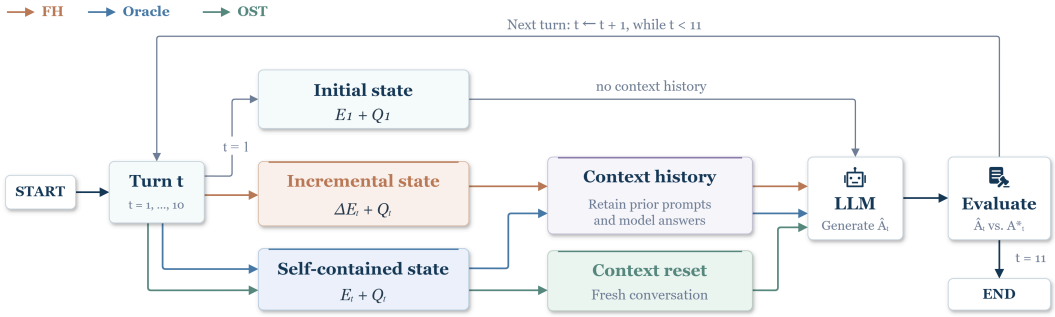

Figure 3: Three evaluation protocols in DYNAEVAL.

State maintenance has long been studied in task-oriented dialogue, where models predict schema-bound belief states (Budzianowski et al., 2018; Wu et al., 2019; Rastogi et al., 2020); in long-term conversational memory, where models retrieve and update user information (Packer et al., 2023; Park et al., 2023; Maharana et al., 2024; Wu et al., 2025); and in tool-using agents, where actions modify an external environment (Yao et al., 2023; 2025; Lu et al., 2025), and where long-horizon evaluation requires tracking an environment that changes as a result of the agent's own actions (Shridhar et al., 2021; Jimenez et al., 2024; Zhou et al., 2024; Xie et al., 2024). These lines center state prediction, retrieval, or action execution rather than the versioned task specification that determines a reasoning answer. In dynamic multi-turn interaction, however, state maintenance and task solving occur simultaneously: answering each target question requires reconstructing the currently valid state and then solving the task defined by that state. End-to-end accuracy alone therefore conflates state-reconstruction failures with task-solving failures. DynaEval separates these dimensions through aligned history-conditioned and self-contained instances that share the same operative state and turn-level answer. The self-contained score estimates task-solving ability, while its gap from the history-conditioned score quantifies the additional loss caused by state maintenance.

3 DEGRADATION IN DYNAMIC MULTI-TURN CONVERSATIONS

To evaluate whether language models can reconstruct currently valid task conditions from an evolving conversation and solve the resulting task, we introduce DynaEval (Section 4) and evaluate ten models under three protocols: Full History (FH) presents incremental updates alongside prior user messages and model responses; Oracle explicitly supplies the complete current state while retaining conversation history; and Oracle Single-Turn (OST) presents the same complete state in a fresh session. These protocols share the underlying tasks and turn-level evaluation targets, enabling comparisons between incremental and fully specified task presentations. We examine performance across turns, trace the response-applied state through parameter versions identifiable in model responses, and use controlled state injection to assess the role of state reconstruction in performance degradation.

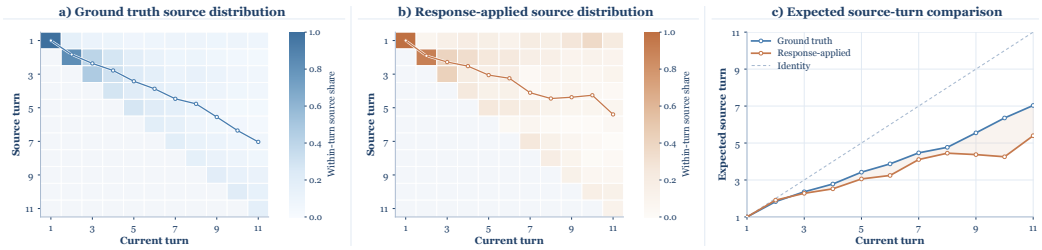

Figure 4: Historical sources of GT and response-applied parameter states under FH. (a,b) Heatmaps show within-turn source shares for GT and response-applied states, respectively; overlaid lines indicate the corresponding mean source turn. Future source turns are gray. (c) Comparison of the mean source turns, with shading marking their difference. The dashed identity line denotes the current turn.

3.1 Performance Degradation across Models

Figure 2 shows that performance deteriorates as conversations progress under FH. All ten models achieve lower accuracy at the final turn than at the first, with mean accuracy falling from 67.9% to 32.9%. This pattern spans models with different initial performance levels, revealing a shared difficulty in sustaining task performance as requirements are repeatedly revised.

Explicitly providing the current state substantially attenuates this degradation. Under Oracle, mean accuracy ends close to its initial level, and its advantage over FH widens as the conversation progresses. OST exhibits a smaller first-to-final decline than FH, indicating that changes in the difficulty of the underlying tasks do not fully account for the observed deterioration. Moreover, Oracle outperforms OST on average despite retaining earlier task versions and model responses: conversation history can support task solving when the current conditions are explicit. Together, these comparisons point to reconstructing the effective state from incremental updates as a potential source of the additional degradation under FH.

3.2 Tracing State Reconstruction in Model Responses

To investigate the source of FH degradation, we identify task-parameter values applied in model responses and trace their historical origins. For traceable numerical parameters, we identify values used in calculations and match them to the corresponding parameter's recorded historical values. Each matched value is assigned to the earliest turn at which it was established; retaining or restoring that value preserves its source. Values without a historical match and ambiguous observations receive no source assignment. We reconstruct the currently effective parameter values and their sources from the task specifications to obtain the ground-truth state (GT). The parameter values identifiable as applied in a response, together with their source assignments, form the response-applied state (RAS). Comparing their source distributions reveals which parts of the task history are represented in the required and response-applied states. Appendix D details the extraction, provenance rules, reliability checks, and aggregation procedures.

Figure 4(a) shows that GT draws on both recent updates and still-valid conditions from earlier turns. Its expected source turn rises steadily but remains below the current turn. The RAS trajectory initially follows GT closely, then falls increasingly behind in later turns (Figure 4(b and c)). Across turns 8 and 10, the RAS mean stays around 4.3 and 4.5 while the GT mean rises from 4.8 to 6.4. At turn 11, the corresponding means are 5.4 and 7.0. These means summarize assignments with identifiable historical sources: in later turns, the response-applied distribution places greater relative weight on earlier sources than the effective state does.

Examining the underlying responses reveals concrete version-selection errors: the model restores an incorrect parameter value and retains a superseded value after an update. These errors show how historical information can be incorporated into a response without resolving which conditions are currently valid. Difficulty integrating dispersed updates into the effective task state is therefore a plausible contributor to FH degradation. The GPT-5.4-mini case study in Appendix D.1 provides

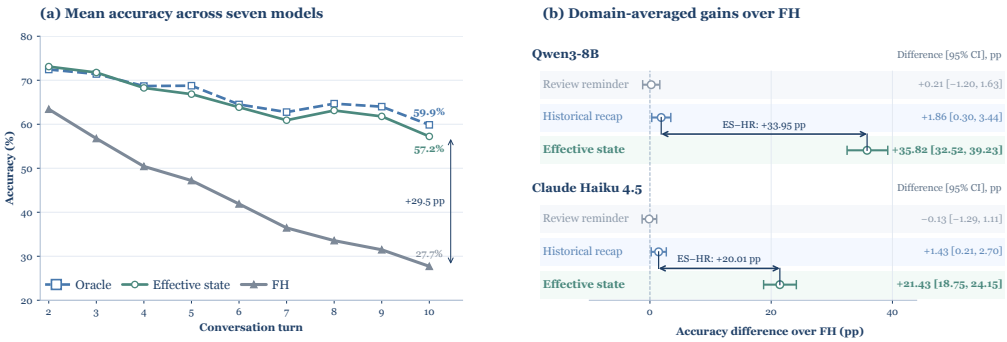

Figure 5: (a) Mean accuracy across seven models. (b) Domain-averaged gains over FH; error bars show 95% bootstrap confidence intervals. Arrows indicate ES-HR differences.

complementary evidence of how such an error affects task solving: when restoring the turn 5 state, the model retains an exponent introduced at turn 6 and obtains an incorrect answer through correct arithmetic on the resulting expression.

3.3 Degradation Diagnostics

We intervene on the current-turn input by replacing the incremental update with the currently effective task state while retaining the preceding conversation history. We evaluate turns 2 to 10 across seven models. Each intervention branches independently from the original FH trajectory, so injected responses never enter subsequent turns. Appendix E provides the experimental settings, prompt construction, and detailed results.

Figure 5(a) shows that supplying the effective state restores accuracy close to the Oracle reference. Substantial recovery occurs even in later turns, with accumulated user messages, earlier model errors, and superseded conditions retained. Thus, much of the growing FH deficit is recoverable without shortening the preceding context: long FH histories remain compatible with substantially higher accuracy when the effective task conditions are explicit.

To examine which information supports this recovery, we evaluate Qwen3-8B and Claude Haiku 4.5 at turns 8 to 10 under three appended-information controls. Review reminder (RR) prompts the model to reconstruct the current state; Historical recap (HR) repeats all preceding user requests in chronological order; Effective state (ES) supplies the complete current task specification. Each condition retains the FH prefix and current incremental instruction, then appends its respective block. This design compares a reconstruction reminder, concentrated historical updates, and an explicitly resolved state within the current message.

Figure 5(b) shows that RR provides no clear improvement, while HR yields gains of 1.86 and 1.43 percentage points for Qwen3-8B and Haiku, respectively. Repeating the user history near the current request therefore offers a small benefit while leaving the accumulated revisions unresolved. ES yields substantially larger gains of 35.82 and 21.43 points. The direct ES–HR advantages are 33.95 points (95% CI [30.85, 37.30]) and 20.01 points [17.43, 22.58], showing a substantial benefit from supplying the resolved conditions beyond concentrating the historical updates. These contrasts support difficulty determining which conditions survive revisions, cancellations, and rollbacks as a major source of FH degradation. Together with the recovery under retained histories in Figure 5(a), these results support difficulty reconstructing currently valid conditions from evolving interaction history as a major source of FH degradation.

4 DynaEval: Benchmark Design

4.1 Dynamic Task State Formalization

DynaEval formalizes a multi-turn interaction as a sequence of versioned task states. The task state E_t records the currently operative environment at turn t, including task variables, active constraints,

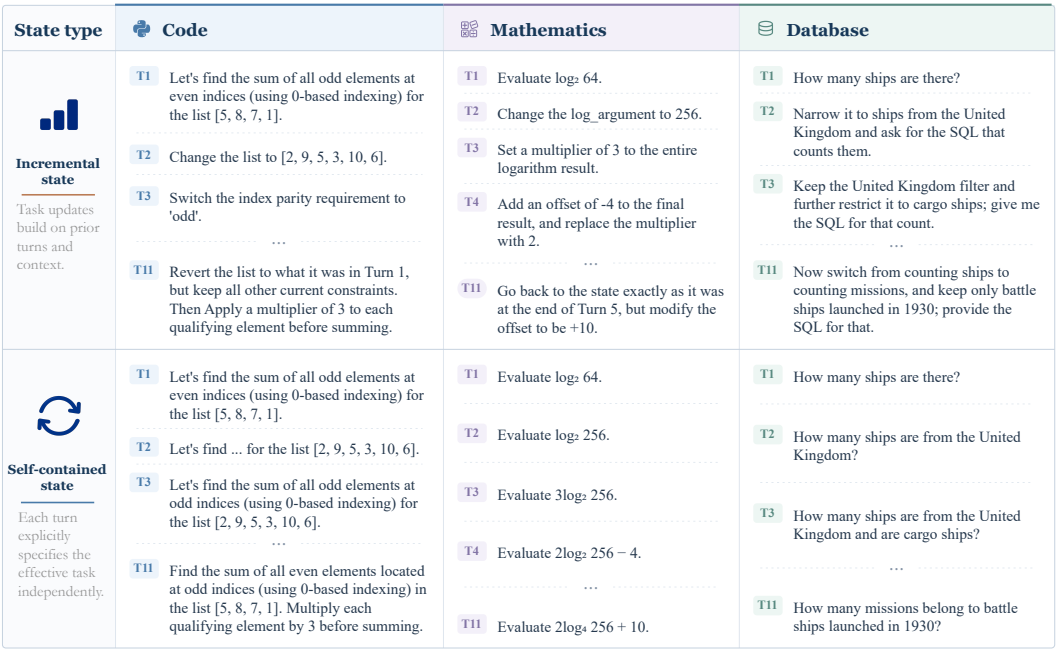

Figure 6: Paired examples from DYNAEVAL. Incremental-state prompts (top) present task updates; self-contained-state prompts (bottom) specify the complete current task. Each pair shares the same target question and reference answer.

domain-specific specifications, and output requirements. Each user instruction supplies a state update ΔE_t and a task Q_t. The update follows explicit precedence semantics: unaffected components persist, targeted components change, and superseded constraints become inactive. This process establishes the effective task specification for the current turn. The reference answer A_t^* is defined by E_t and Q_t, giving every turn a state-bound evaluation target.

DynaEval includes seven state-transition operators: ADD introduces a new requirement; OVERRIDE replaces an active value or rule; DELETE deactivates an existing constraint; REVERT restores an earlier task version; BRANCH applies an update under a specified condition; REFERENCE retrieves a previously established state or constraint; and MERGE combines selected components from multiple versions. Appendix A provides the complete state representation and transition semantics.

4.2 PAIRED BENCHMARK CONSTRUCTION

Figure 6 illustrates the paired incremental-state and self-contained-state prompts across the three domains. Benchmark construction begins with seed tasks sampled from established, fully specified single-turn benchmarks. We select tasks with explicit inputs, verifiable outputs, and task specifications that support controlled modification. Each selected task is normalized into a structured triple (E_1, Q_1, A_1^*). The initial state E_1 records the complete task specification, Q_1 defines the requested output, and A_1^* provides the verified reference answer.

Starting from each structured seed task, we define a state update ΔE_t and a target question Q_t for every turn $t > 1$. An automated state machine applies each update sequentially:

$$E_t = E_{t-1} \oplus \Delta E_t, \quad t \geq 2,$$

where $\oplus$ denotes applying the environment update to the preceding state. After each update, the state machine records the resulting state E_t and derives the corresponding ground-truth answer A_t^*. Each A_t^* is bound to the state active at turn t. An LLM then converts each structured record $(E_t, \Delta E_t, Q_t, A_t^*)$ into two aligned natural-language versions. The incremental-state version presents the current update ΔE_t and target question Q_t, leaving the effective state to be reconstructed from the interaction history. The self-contained-state version explicitly presents the currently effective state E_t and the same target question Q_t. Both versions share E_t, Q_t, and the

Table 1: Turn-level accuracy (%). Darker red indicates lower FH accuracy on a shared 0–100 scale; Mean averages models equally. Small annotations in all protocol rows show percentage-point changes from Turn 1 within each row, computed from the displayed values; ↓ denotes a decrease and ↑ an increase.

Model	Protocol	Turn										
		1	2	3	4	5	6	7	8	9	10	11
Claude Sonnet 4.6	FH	78.3	76.4↓1.9	72.7↓5.6	68.1↓10.2	65.4↓12.9	62.6↓15.7	56.5↓21.8	54.9↓23.4	53.4↓24.9	51.6↓26.7	49.0↓29.3
	Oracle	75.9	82.3↑6.4	82.5↑6.6	82.9↑7.0	82.3↑6.4	82.1↑6.2	80.1↑4.2	81.6↑5.7	82.0↑6.1	79.7↑3.8	80.8↑4.9
	OST	78.5	85.1↑6.6	85.5↑7.0	83.8↑5.3	83.6↑5.1	82.5↑4.0	82.3↑3.8	82.9↑4.4	82.1↑3.6	79.7↑1.2	79.9↑1.4
Claude Haiku 4.5	FH	74.0	76.6↑2.6	68.9↓5.1	62.8↓11.2	59.5↓14.5	55.8↓18.2	50.3↓23.7	47.3↓26.7	47.0↓27.0	44.2↓29.8	40.7↓33.3
	Oracle	75.0	83.2↑8.2	82.7↑7.7	82.7↑7.7	81.4↑6.4	79.4↑4.4	79.9↑4.9	80.5↑5.5	80.8↑5.8	77.2↑2.2	79.0↑4.0
	OST	75.0	83.4↑8.4	80.8↑5.8	81.0↑6.0	78.8↑3.8	76.4↑1.4	75.9↑0.9	77.5↑2.5	75.7↑0.7	72.9↓2.1	74.0↓1.0
GPT-5.4	FH	77.7	76.8↓0.9	70.2↓7.5	66.9↓10.8	63.9↓13.8	59.9↓17.8	54.9↓22.8	55.4↓22.3	54.3↓23.4	53.8↓23.9	54.0↓23.7
	Oracle	78.3	85.5↑7.2	84.9↑6.6	86.6↑8.3	84.9↑6.6	84.0↑5.7	84.7↑6.4	86.2↑7.9	86.7↑8.4	83.8↑5.5	84.2↑5.9
	OST	77.7	84.0↑6.3	82.9↑5.2	84.2↑6.5	83.1↑5.4	81.0↑3.3	80.3↑2.6	82.3↑4.6	84.7↑7.0	79.7↑2.0	79.9↑2.2
GPT-5.4-mini	FH	70.9	68.3↓2.6	61.3↓9.6	58.0↓12.9	54.9↓16.0	54.0↓16.9	47.5↓23.4	44.0↓26.9	40.9↓30.0	40.7↓30.2	41.1↓29.8
	Oracle	68.5	76.2↑7.7	75.0↑6.5	75.9↑7.4	75.3↑6.8	73.3↑4.8	73.5↑5.0	74.8↑6.3	75.3↑6.8	72.7↑4.2	74.6↑6.1
	OST	69.2	76.4↑7.2	74.6↑5.4	74.2↑5.0	70.9↑1.7	68.0↓1.2	61.3↓7.9	63.9↓5.3	59.1↓10.1	57.8↓11.4	55.4↓13.8
Qwen3-32B	FH	70.0	73.5↑3.5	65.2↓4.8	59.7↓10.3	56.0↓14.0	53.4↓16.6	44.9↓25.1	44.6↓25.4	42.7↓27.3	36.5↓33.5	35.5↓34.5
	Oracle	67.8	79.0↑11.2	78.1↑10.3	75.7↑7.9	75.3↑7.5	73.5↑5.7	71.8↑4.0	73.5↑5.7	75.1↑7.3	71.6↑3.8	72.9↑5.1
	OST	68.7	74.6↑5.9	75.3↑6.6	72.4↑3.7	70.7↑2.0	68.5↓0.2	62.2↓6.5	67.4↓1.3	60.6↓8.1	63.0↓5.7	60.8↓7.9
Qwen3-8B	FH	69.2	66.8↓2.4	59.2↓10.0	52.1↓17.1	47.9↓21.3	40.9↓28.3	34.0↓35.2	28.7↓40.5	26.2↓43.0	22.7↓46.5	21.6↓47.6
	Oracle	68.8	75.5↑6.7	75.1↑6.3	71.6↑2.8	72.0↑3.2	67.3↓1.5	66.4↓2.4	67.8↓1.0	67.3↓1.5	61.1↓7.7	63.7↓5.1
	OST	68.8	76.6↑7.8	75.3↑6.5	70.6↑1.8	71.4↑2.6	65.5↓3.3	60.4↓8.4	64.4↓4.4	60.6↓8.2	54.9↓13.9	55.8↓13.0
Grok 4.5	FH	63.9	62.8↓1.1	55.6↓8.3	50.5↓13.4	48.3↓15.6	43.8↓20.1	41.1↓22.8	38.1↓25.8	36.8↓27.1	35.0↓28.9	33.7↓30.2
	Oracle	64.8	74.5↑9.7	72.9↑8.1	69.9↑5.1	69.0↑4.2	66.1↑1.3	63.7↓1.1	64.9↑0.1	64.9↑0.1	63.8↓1.0	63.8↓1.0
	OST	66.4	74.2↑7.8	73.4↑7.0	72.0↑5.6	69.4↑3.0	66.1↓0.3	65.3↓1.1	67.0↑0.6	67.0↑0.6	62.9↓3.5	62.2↓4.2
Ministral-3-8B	FH	63.7	62.7↓1.0	57.8↓5.9	50.6↓13.1	47.6↓16.1	41.1↓22.6	34.8↓28.9	31.0↓32.7	29.4↓34.3	25.8↓37.9	23.4↓40.3
	Oracle	63.9	75.4↑11.5	72.8↑8.9	69.0↑5.1	69.7↑5.8	65.2↑1.3	63.2↓0.7	66.2↑2.3	64.2↑0.3	60.4↓3.5	62.2↓1.7
	OST	64.0	71.3↑7.3	70.5↑6.5	65.9↑1.9	65.5↑1.5	63.7↓0.3	58.8↓5.2	60.4↓3.6	57.8↓6.2	55.0↓9.0	55.9↓8.1
Llama-3.1-70B	FH	60.6	55.4↓5.2	50.1↓10.5	43.6↓17.0	43.1↓17.5	37.8↓22.8	31.7↓28.9	27.4↓33.2	23.6↓37.0	20.3↓40.3	19.5↓41.1
	Oracle	61.9	64.6↑2.7	64.6↑2.7	61.9=0.0	63.7↑1.8	57.8↓4.1	54.7↓7.2	56.2↓5.7	55.1↓6.8	49.7↓12.2	50.1↓11.8
	OST	62.2	65.7↑3.5	63.7↑1.5	59.3↓2.9	59.1↓3.1	52.5↓9.7	48.4↓13.8	54.5↓7.7	50.3↓11.9	47.7↓14.5	48.4↓13.8
Llama-3.1-8B	FH	50.9	44.6↓6.3	39.7↓11.2	32.7↓18.2	28.0↓22.9	20.8↓30.1	17.6↓33.3	16.7↓34.2	15.2↓35.7	11.0↓39.9	10.2↓40.7
	Oracle	50.1	54.8↑4.7	54.5↑4.4	50.8↑0.7	50.0↓0.1	42.7↓7.4	39.9↓10.2	42.9↓7.2	39.7↓10.4	35.1↓15.0	37.5↓12.6
	OST	49.6	49.9↑0.3	51.0↑1.4	46.6↓3.0	44.3↓5.3	38.1↓11.5	35.9↓13.7	37.7↓11.9	37.3↓12.3	33.3↓16.3	34.8↓14.8
Mean	FH	67.9	66.4↓1.5	60.1↓7.8	54.5↓13.4	51.4↓16.5	47.0↓20.9	41.3↓26.6	38.8↓29.1	37.0↓30.9	34.1↓33.8	32.9↓35.0
	Oracle	67.5	75.1↑7.6	74.3↑6.8	72.7↑5.2	72.4↑4.9	69.1↑1.6	67.8↑0.3	69.5↑2.0	69.1↑1.6	65.5↓2.0	66.9↓0.6
	OST	68.0	74.1↑6.1	73.3↑5.3	71.0↑3.0	69.7↑1.7	66.2↓1.8	63.1↓4.9	65.8↓2.2	63.5↓4.5	60.7↓7.3	60.7↓7.3

reference answer A_t^*. This pairing holds the underlying task fixed while varying whether its effective conditions are supplied explicitly, enabling controlled comparisons of task performance under the two presentations.

Automatic checks verify transition consistency, reference-answer correctness, and semantic alignment between the paired prompts. All 543 sequences underwent independent review by two reviewers, with disagreements adjudicated by a third. Identified issues were corrected and revalidated before finalization. Appendix B details the construction and validation procedures.

5 Experiments

5.1 Experimental Setup

We evaluate the ten models under the three protocols described in Section 3, with a shared deterministic scoring pipeline. Mathematical responses are scored by numerical or symbolic equivalence of the final answer. Code implementations are executed in a restricted sandbox and checked against reference outputs and task constraints. SQL responses are executed on the corresponding databases and scored by agreement with reference results. We report accuracy by conversation turn and task domain, averaging repeated runs within each model and weighting models equally in aggregate results.

5.2 Overall Results

Table 1 summarizes performance across models and protocols. All ten models exhibit lower final-turn than first-turn accuracy under FH, with the mean falling from 67.9% to 32.9%. In contrast,

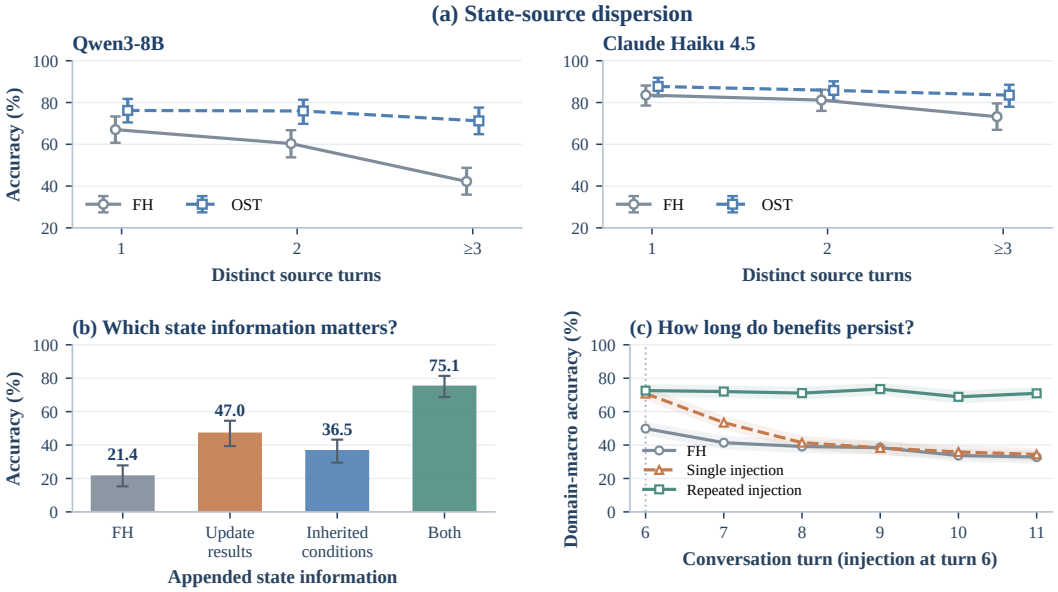

Figure 7: (a) FH and OST accuracy by state-source count. (b) Partial versus complete state support. (c) Single versus repeated state injection, starting at turn 6 (three-domain average). Panels (b,c) use Qwen3-8B; error bars and bands show 95% bootstrap confidence intervals.

Oracle accuracy remains comparatively stable, moving from 67.5% to 66.9%, while OST shows a smaller decline. This contrast indicates that underlying task difficulty and interaction history do not fully explain the deterioration under FH, consistent with the state-reconstruction diagnostics in Section 3.

Initial accuracy also provides an incomplete picture of sustained performance. Qwen3-8B and GPT-5.4-mini begin within two percentage points of each other but finish nearly twenty points apart. Model rankings likewise depend on the evaluation horizon: Claude Sonnet 4.6 achieves the highest mean FH accuracy, whereas GPT-5.4 achieves the highest final-turn accuracy. Evaluating evolving task sequences therefore reveals differences in maintaining performance that initial-turn scores alone do not capture. Table 1 reports the full per-turn results; domain breakdowns are provided in the Appendix C.

5.3 State Source Dispersion and Task Performance

We explore whether the distribution of effective parameters across historical turns is associated with the FH and OST performance gap. On a mathematics subset comprising 184 sequences and 1,321 turn instances, we group states by the number of distinct source turns among their tracked effective parameters: one, two, or at least three. A source turn identifies where a parameter version was established; restoring an earlier version preserves its original source. We compare Qwen3-8B and Claude Haiku 4.5 on the same subset.

Figure 7(a) shows larger OST and FH gap estimates for states spanning more source turns. Between the one-source and at-least-three-source groups, the gap increases from 9.2 to 29.0 percentage points for Qwen3-8B and from 4.1 to 10.3 points for Claude Haiku 4.5. FH accuracy declines more sharply across these groups than OST accuracy. This descriptive pattern associates broader state-source dispersion with a larger protocol gap.

5.4 Which State Information Matters?

We examine which parts of the effective state support recovery by separating facts established, modified, restored, or removed by the current update from inherited conditions that remain active. We evaluate Qwen3-8B on 313 target turns (turns 8–10) from 121 database sequences whose states ad-

mit this partition. Each target independently branches from its original FH history. We retain the incremental prompt and append the update results, the inherited conditions, or their union, alongside an FH control without appended information. Facts are rendered as task-level conditions with identical wording across the partial and combined cards.

Figure 7(b) shows that both components improve performance. Accuracy rises from 21.4% under FH to 47.0% with update results and 36.5% with inherited conditions. Update-result support exceeds inherited-condition support by 10.5 percentage points (95% CI [2.3, 18.6]). Combining both components raises accuracy to 75.1%, improving on update-result support alone by a further 28.1 points [20.9, 35.4]. On the additive accuracy scale, the combined gain exceeds the sum of the separate gains by 13.0 points [5.5, 20.4].

Update-result support makes the outcome of the current operation explicit, including the values selected by historical references or reversions. Inherited-condition support supplies requirements that remain active across updates. Their combined benefit indicates that both resolving the current update and retaining valid historical conditions contribute to recovery. The positive interaction further shows that providing both components yields a larger accuracy gain than the sum of their separate gains.

5.5 How Long Do the Benefits of State Injection Persist?

The state-injection diagnostics in Section 3 establish an immediate performance benefit. We now examine whether that benefit persists through subsequent updates. Using Qwen3-8B, we branch three continuations from each original five-turn FH history: FH continuation, a single state injection at turn 6 followed by incremental updates, and repeated state injection at every turn from 6 to 11. Each branch retains its own generated responses in subsequent history. We report domain-macro accuracy on 530 sequences with complete results across all three branches.

Figure 7(c) shows that a single injection improves accuracy over FH by 20.9 percentage points at turn 6 (95% CI [17.0, 24.9]). The gain narrows to 12.0 points at turn 7 and ranges from -0.2 to 2.3 points at turns 8–11. Averaged over turns 7–11, the benefit is 3.6 points [1.0, 6.1], with the clearest persistence in database tasks. Repeated injection maintains accuracy between 68.9% and 73.5%, exceeding single injection by 30.6 points [27.7, 33.6] over turns 7–11, with positive gains in all three domains.

The injected state and the response it produces remain in the single-injection branch's subsequent history. The smaller later gains therefore occur even though the corrected snapshot remains available: subsequent turns still require integrating new updates with conditions that continue to hold. Repeated injection supplies the resolved state after each update and sustains substantially higher accuracy. These trajectories distinguish immediate recovery from sustained performance, showing that a single state correction does not remove the continuing demands of maintaining an evolving task state.

6 Conclusion

We introduced DynaEval, a benchmark of 543 eleven-turn sequences across mathematics, code, and databases, pairing incremental updates with complete descriptions of evolving task states. Across ten models, mean full-history accuracy falls from 67.9% at the first turn to 32.9% at the final turn. Response-applied state tracing on Qwen3-8B shows that identifiable parameter sources increasingly lag behind those of the effective state, while response-level analysis reveals incorrect restorations and the continued use of superseded values. Supplying the effective state restores near-Oracle performance across seven models while retaining the preceding conversation, supporting state reconstruction as a major source of the observed degradation. Further experiments with Qwen3-8B show that recovery depends on both resolved updates and still-valid inherited conditions, and that repeated state injection sustains substantially higher accuracy than a single correction. These findings highlight the importance of continually integrating new updates with conditions that remain valid as tasks evolve. DynaEval provides a controlled testbed for evaluating and improving this ability.

AI USE STATEMENT

We used generative AI tools to assist manuscript polishing and revision, and literature discovery. For benchmark construction, Gemini-3.1-Pro converted structured task records into paired natural-language instructions, and DeepSeek-R1 supported semantic consistency checks. The generated data underwent automatic validation and human review as described in Appendix B. Reference answers were derived from the structured task states and checked using domain-specific procedures. The authors take responsibility for the final manuscript, dataset, analyses, and conclusions.

ETHICS STATEMENT

DYNAEVAL uses synthetic task trajectories derived from the publicly available MATH, HumanEval, and Spider benchmarks. Dialogue instructions are constructed from structured task specifications, without collecting private user conversations. Generated code is evaluated in a restricted sandbox, and SQL evaluation uses the associated benchmark databases. The benchmark is intended to support research on model reliability under evolving task requirements. The reported findings concern the controlled tasks and interaction protocols studied in this work. Source attribution and licensing information will accompany the planned data release.

REPRODUCIBILITY STATEMENT

The main text describes the benchmark construction and evaluation protocols, while the appendices document data construction and validation, experimental settings, prompt templates, state-tracing procedures, and statistical aggregation. We plan to publicly release the code used for multi-turn task simulation and the associated data upon acceptance, together with evaluation and analysis scripts and recorded model responses. Repeating experiments with proprietary models depends on the continued availability of the evaluated versions, which may be updated or retired and become inaccessible through their APIs. The recorded responses will support recomputing the reported scores and statistics when the original model versions are no longer available.

REFERENCES

Anthropic. The Claude 3 model family: Opus, Sonnet, Haiku. Technical report, Anthropic, 2024. URL https://www-cdn.anthropic.com/c6a80a657af445f40e31afac050f3bf76d3b1404.pdf.

Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, and Wanli Ouyang. MT-Bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7421–7454, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.401. URL https://aclanthology.org/2024.acl-long.401/.

Joachim Baumann, Vishakh Padmakumar, Xiang Li, John Yang, Diyi Yang, and Sanmi Koyejo. SWE-chat: Coding agent interactions from real users in the wild. *arXiv preprint arXiv:2604.20779*, 2026. URL https://arxiv.org/abs/2604.20779.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. MultiWOZ: A large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 5016–5026, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1547. URL https://aclanthology.org/D18-1547/.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI. DeepSeek-V3 technical report, 2024. URL https://arxiv.org/abs/2412.19437.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Xixuan Hao, Zeyu Zhang, Zehao Lin, Yihang Sun, Ziliang Guo, Xichong Zhang, Yuxuan Liang, Feiyu Xiong, and Zhiyu Li. MemOps: Benchmarking lifecycle memory operations in long-horizon conversations. *arXiv preprint arXiv:2607.12893*, 2026. doi: 10.48550/arXiv.2607.12893. URL https://arxiv.org/abs/2607.12893.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://arxiv.org/abs/2103.03874.

Christine Herlihy, Jennifer Neville, Tobias Schnabel, and Adith Swaminathan. On overcoming miscalibrated conversational priors in LLM-based ChatBots. In *Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence*, volume 244 of *Proceedings of Machine Learning Research*, pp. 1599–1620. PMLR, 2024. URL https://proceedings.mlr.press/v244/herlihy24a.html.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Wai-Chung Kwan, Xingshan Zeng, Yuxin Jiang, Yufei Wang, Liangyou Li, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. MT-Eval: A multi-turn capabilities evaluation benchmark for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 20153–20177, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1124. URL https://aclanthology.org/2024.emnlp-main.1124/.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs get lost in multi-turn conversation. In *International Conference on Learning Representations*, volume 2026, pp. 54738–54778, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/59f6421e64707225fdf5b28840679a07-Abstract-Conference.html.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703/.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can LLM already serve as a database interface? a BIg bench for large-scale database grounded text-to-SQLs. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2305.03111.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language

models. *Transactions on Machine Learning Research*, 2023. URL https://arxiv.org/abs/2211.09110.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, Albuquerque, New Mexico, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. *arXiv preprint arXiv:2402.17753*, 2024. URL https://arxiv.org/abs/2402.17753.

OpenAI. GPT-4 technical report, 2023. URL https://arxiv.org/abs/2303.08774.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. URL https://arxiv.org/abs/2310.08560.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*, San Francisco, CA, USA, 2023. ACM. doi: 10.1145/3586183.3606763. URL https://arxiv.org/abs/2304.03442.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL https://jmlr.org/papers/v21/20-074.html.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards scalable multi-domain conversational agents: The Schema-Guided Dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8689–8696, 2020. doi: 10.1609/aaai.v34i05.6394. URL https://ojs.aaai.org/index.php/AAAI/article/view/6394.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03768.

Ved Sirdeshmukh, Kaustubh Deshpande, Johannes Mols, Lifeng Jin, Ed-Yeremai Cardona, Dean Lee, Jeremy Kritz, Willow Primack, Summer Yue, and Chen Xing. MultiChallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier LLMs. *arXiv preprint arXiv:2501.17399*, 2025. URL https://arxiv.org/abs/2501.17399.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. MINT: Evaluating LLMs in multi-turn interaction with tools and language feedback. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.10691.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, 2019. Association for Computational Linguistics. URL https://arxiv.org/abs/1905.08743.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*, volume 2025, pp. 86809–86836, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/d813d324dbf0598bbdc9c8e79740ed01-Abstract-Conference.html.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*, 2024. URL https://arxiv.org/abs/2404.07972.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Dayiheng Liu, Fei Huang, Guanting Dong, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024. URL https://arxiv.org/abs/2407.10671.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

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/hash/1b126cc38b8638e07bef37e7b2bb72bf-Abstract-Conference.html.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018. Association for Computational Linguistics. URL https://arxiv.org/abs/1809.08887.

Zexun Zhan, Shuzheng Gao, Ruida Hu, and Cuiyun Gao. SR-Eval: Evaluating LLMs on code generation under stepwise requirement refinement. *arXiv preprint arXiv:2509.18808*, 2025. doi: 10.48550/arXiv.2509.18808. URL https://arxiv.org/abs/2509.18808.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623. Curran Associates, Inc., 2023. doi: 10.52202/075280-2020. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

A Task State Representation and Transition Semantics

This appendix makes the update semantics in Section 4.1 explicit. The incremental and self-contained views share the same effective task, question, and reference answer; their difference is how the task conditions are presented.

A.1 Task State Representation

Let $E_t = (V_t, C_t, S_t, O_t)$ contain inputs and variable assignments, active constraints, domain-specific specifications, and output requirements. Each component contains named fields or conditions: a slot can be an input list, a logarithm base, a filtering predicate, or a required function interface. Thus an update can address a field inside a component, rather than replacing the entire component. The target question Q_t specifies what to compute under E_t, and

$$A_t^* = \mathrm{Solve}(E_t, Q_t).$$

Here A_t^* denotes a verified reference answer. Equivalent mathematical forms, implementations, or SQL queries can also receive credit under the domain-specific scorer.

State, question, and history. The conversation contains user instructions and model responses. The benchmark's reference state follows the task instructions; a model's incorrect answer does not automatically become a new state assignment. A historical state reference and a historical result reference are distinct. To resolve both, retain the canonical archive

$$\mathcal{A}_{t-1} = \big((E_j, Q_j, A_j^*)\big)_{j=1}^{t-1}.$$

Historical result references in the benchmark's reference trajectory resolve to A_j^*. This also covers questions that combine the current computation with a previous result. Restoring E_j alone need not restore Q_j.

A.2 Applying State Updates

For $t > 1$, resolve the instruction into an ordered update ΔE_t and the current question Q_t:

$$E_t = \mathrm{Apply}(E_{t-1}, \Delta E_t; \mathcal{A}_{t-1}).$$

This expands $E_t = E_{t-1} \oplus \Delta E_t$ in the main text. Unaffected fields persist. A replaced or deleted condition is inactive, even though its text remains in the conversation. Historical references select the requested version and scope. For a compound instruction, operations follow its stated dependencies; simultaneous assignments read the same pre-assignment state. For example, swapping a base and multiplier exchanges their old values, whereas "restore turn 5, then set the offset to 10" applies two operations in sequence. An instruction without a unique, valid interpretation requires correction during dataset validation.

A.3 State-Transition Operators

For a partial reversion to turn $j < t$,

$$E_t[J] = E_j[J], \qquad E_t[\overline{J}] = E_{t-1}[\overline{J}],$$

before subsequent operations in the instruction are applied. Copying an absent historical field restores its absence. A full state reversion uses all state fields for J. This distinguishes "restore the earlier list" from "restore the entire earlier task state."

A.4 Illustrative Update Composition

Consider $f(x; b, m, p, c) = m \log_b(x^p) + c$, with $x > 0$, $b > 0$, and $b \neq 1$. In sequence 298, the turn-5 state is $(x, b, m, p, c) = (256, 4, 2, 1, -4)$. Turn 6 changes p to 3. Restoring the complete turn-5 state and then setting $c = 10$ yields

$$2 \log_4(256^1) + 10 = 18.$$

The restoration includes the argument power; retaining $p = 3$ instead gives 34. Table 3 shows all eleven turns, including a later question that uses the canonical turn-7 result.

Table 2: Operator semantics. J denotes the addressed fields or conditions; all fields outside the instructed scope persist.

Operator	Effect
ADD	Introduce a new field or active condition. Existing applicable conditions remain in force.
OVERRIDE	Replace the value or rule at the addressed slot. The old and new values are not composed unless explicitly requested.
DELETE	Deactivate the addressed condition or remove the addressed field. Earlier values are not automatically restored.
REVERT	Copy the requested historical state or fields, preserving unaffected current fields for a partial reversion.
BRANCH	Evaluate the stated predicate against its specified state or historical result, then apply the selected alternative.
REFERENCE	Retrieve a named historical field or result for the current update or question; retrieval alone does not restore other fields.
MERGE	Copy specified fields from their designated historical versions. Conflicts follow the instruction's precedence; no unrestricted union of historical constraints is taken.

Table 3: Complete mathematical trajectory (sequence 298). The state is (x, b, m, p, c) for $m \log_b(x^p) + c$.

Turn	Update / requested computation	State	Answer
1	Evaluate the initial logarithm	$(64, 2, 1, 1, 0)$	6
2	Set argument to 256	$(256, 2, 1, 1, 0)$	8
3	Set multiplier to 3	$(256, 2, 3, 1, 0)$	24
4	Add offset -4; multiplier $\leftarrow 2$	$(256, 2, 2, 1, -4)$	12
5	Base $\leftarrow 4$	$(256, 4, 2, 1, -4)$	4
6	If $x > 100$, power $\leftarrow 3$; else 2	$(256, 4, 2, 3, -4)$	20
7	Restore turn-1 base; remove offset	$(256, 2, 2, 3, 0)$	48
8	Restore turn 5; offset $\leftarrow 10$	$(256, 4, 2, 1, 10)$	18
9	If multiplier is even, swap base and multiplier	$(256, 2, 4, 1, 10)$	42
10	Power $\leftarrow b$; argument $\leftarrow 16$	$(16, 2, 4, 2, 10)$	42
11	Offset $\leftarrow 0$; evaluate and add A_7^*	$(16, 2, 4, 2, 0)$	80

A.5 Alignment of the Two Task Presentations

The incremental view supplies the current update and question; the self-contained view states their resolved conditions and the same question. Their alignment requirement is

$$\mathrm{Resolve}(E_1, Q_1, \Delta E_2, Q_2, \ldots, \Delta E_t, Q_t) \equiv (E_t, Q_t),$$

where resolution maintains $\mathcal{A}$ and $\equiv$ means equality of operative task requirements. Equal final answers alone do not establish alignment: different states can coincidentally yield the same answer.

B Dataset Construction and Quality Validation

DynaEval contains 543 sequences of eleven turns, or 5,973 paired instances. Each pair shares its domain, identifier, turn, difficulty label, and reference answer. The incremental and self-contained prompt fields differ.

B.1 Source Tasks and Structured Construction

Seed tasks originate from MATH (Hendrycks et al., 2021), HumanEval (Chen et al., 2021), and Spider (Yu et al., 2018). Mathematics starts from 200 candidates sampled by source difficulty; inspection of the original problem statements excludes seven tasks unsuitable for multi-turn or long-horizon expansion, leaving 193. Code starts from 150 sampled HumanEval tasks; a model assigns

Table 4: Retained sequences and exported turn-1 difficulty labels. Each sequence contributes eleven paired turns.

Domain	Sequences	Turns	Easy	Medium	Hard	Extra
Mathematics	193	2,123	101	81	11	—
Code	150	1,650	131	19	0	—
Database	200	2,200	55	53	48	44

difficulty labels after selection. Those code labels describe the selected tasks and are not sampling strata.

For databases, the retained sampling plan uses Spider's `train_spider.json` (7,000 source rows), random seed 42, and SQL hardness strata. Prefiltering records 3,023 duplicate-SQL removals, 132 empty-result queries, and two execution failures. The candidate plan contains 840 tasks across 140 databases, with 210 easy, 211 medium, 210 hard, and 209 extra-hard candidates. The final 200 sequences cover 127 databases, with at most two sequences per database. Candidate-plan counts describe the construction pool; Table 4 reports the retained benchmark.

Difficulty labels in the exported benchmark are descriptive categories; the mathematical source levels, code model labels, and SQL hardness categories are not a shared cross-domain scale.

Each seed is normalized into (E_1, Q_1, A_1^*). Subsequent structured updates produce $(E_t, \Delta E_t, Q_t, A_t^*)$, retaining the historical versions needed for restoration and references. Both natural-language presentations derive from this common specification.

B.2 Paired Natural-Language Rendering

Gemini-3.1-Pro renders the structured tasks into paired instructions. The incremental prompt expresses the update and current question; the self-contained prompt includes the complete effective requirements. Rendering preserves the attached reference answer and must not add constraints or solution information. DeepSeek-R1 provides a separate semantic comparison: it interprets the incremental user history or the current self-contained prompt, without receiving the canonical state or reference answer, and the recovered requirements are compared with the canonical record. Uncertain cases proceed to human review.

B.3 Automatic Validation

Validation checks complementary properties:

- **Structure and transitions:** eleven consecutive turns, one counterpart per instance, aligned metadata and answers, available historical references, and update replay against the stored state.
- **References:** recomputation or equivalence checks for mathematics; executable tests and required contracts for code; execution on the associated SQLite database for SQL, together with checks of the requested task. Execution success alone does not certify semantic fidelity.
- **Rendering:** compare recovered inputs, constraints, rules, output requirements, questions, and reference scopes with the canonical specification. Agreement between the two renderings alone is insufficient.

The offline release check confirms all 5,973 keys in each view are unique and aligned, all 543 sequences contain turns 1–11, and paired domain, difficulty, and reference-answer fields agree. These checks establish structural consistency; human review supplies the semantic judgments below.

B.4 Full-Sequence Human Review

Five reviewers cover all 543 sequences. Two reviewers independently inspect each complete trajectory, both prompt views, and reference answers. A third reviewer adjudicates disagreements and does not adjudicate a sequence for which they provided an initial assessment. Initial judgments precede consultation of the other reviewer's decisions and automated semantic flags.

Table 5: Human review before correction: adjudicated judgments over 5,973 paired turns. Percentages are rounded author-provided aggregates.

Criterion	Statistic	Percentage
State-transition correctness	Pass rate	86.0
Incremental update fidelity	Pass rate	85.3
Self-contained completeness	Pass rate	98.2
Pair semantic equivalence	Pass rate	89.4
Reference-answer correctness	Pass rate	95.2
Direct answer leakage	Incidence	1.8

Table 5 reports the adjudicated initial judgments before correction. Each rate uses the same 5,973 paired-turn denominator; categories may overlap. They measure task quality at initial review, not agreement between reviewers or the accuracy of the automated verifier.

B.5 Direct Answer Leakage and Correction

The reported leakage criterion is direct appearance of the reference answer in a prompt. A paired turn is counted once when either view is affected. The 1.8% rate concerns the initial version and does not quantify disclosure of intermediate steps or other solution hints. Issues identified during initial review were corrected, and affected turns and dependent subsequent turns passed reinspection.

The twelve sequence defects listed in the historical diagnostic exclusion manifests were also corrected in the final dataset. The reported diagnostic experiments retain their frozen cohorts and denominators; their historical exclusion lists do not describe defects remaining in the final release.

Presentation corrections retain the intended task and reference; corrections to the canonical task require updating its dependent records. The offline audit also retains literal answer-overlap candidates for inspection. Numeric inputs and legitimate examples can coincide with an answer, so string matches are not automatically counted as leakage incidents. No incidence estimate is inferred from that screening list.

B.6 Task Complexity and Historical References

The mathematical state-source analysis in Figure 7 covers 184 sequences and 1,321 eligible turns with 3,230 tracked slot occurrences. The states contain 1–7 tracked slots (mean 2.45); 388 turns have one distinct source turn, 410 have two, and 523 have at least three. The corresponding sequence counts are 177, 169, and 135; a sequence can enter multiple groups. These counts describe tracked state components, not all natural-language constraints. Source count, turn depth, and tracked-slot count covary.

An additional reproducible text scan finds explicit numbered references to earlier turns in 297/1,930 mathematics updates and 175/1,500 code updates; 64 and 33 updates, respectively, name multiple earlier turns. Among these updates, the median maximum reference distance is six turns (maximum ten). The 2,000 database updates contain no matching numbered-turn references. The scan counts literal `Turn j` references; implicit references and operations expressed without numbered turns remain outside this statistic.

B.7 Source-Task Familiarity

The source tasks are public benchmarks. Pairing fixes the target state and reference within each incremental/self-contained comparison, and the injection controls retain the original history. These comparisons measure performance under the specified presentations. New trajectories and wording do not establish that a model was unfamiliar with the source problem; training-data membership is not inferred from task or answer overlap.

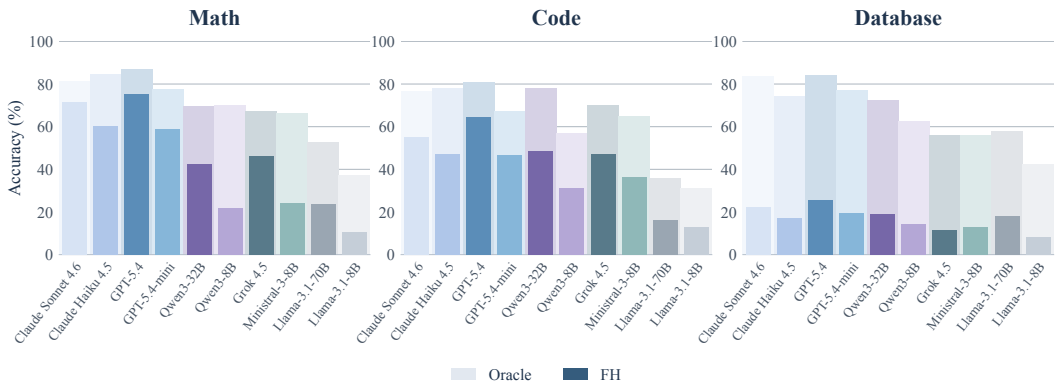

Figure 8: Domain-level accuracy at Turn 11 under FH and Oracle, pooled across repeated runs where available. Dark inner bars represent FH, and light outer bars represent Oracle. The difference between their endpoints shows the accuracy gap associated with providing complete current task conditions.

C Performance across Task Domains

Figure 8 compares FH and Oracle accuracy at Turn 11 across mathematics, code, and databases. Averaged equally across models, the FH and Oracle gaps are 25.9, 23.4, and 49.8 percentage points, respectively. Databases show the largest gap: accuracy rises from 16.8% to 66.6%, approaching Oracle mathematics accuracy (69.4%) despite having the lowest FH accuracy. This contrast suggests that task-solving difficulty alone does not explain the low database performance under FH.

The database gap remains large among the strongest models. Claude Sonnet 4.6 improves from 22.5% to 83.5%, a 61.0-point gap, compared with 9.8 points in mathematics. GPT-5.4 similarly exhibits a 58.5-point database gap, versus 11.9 points in mathematics and 16.0 points in code. Both models therefore depend more strongly on explicit current conditions in database tasks.

GPT-5.4 leads all three domains under both protocols, while Qwen3-32B shows competitive coding performance: its FH accuracy is 48.7%, and its Oracle accuracy of 78.0% ties Claude Haiku 4.5 for second place. Within the Qwen family, the 32B model substantially exceeds the 8B model in FH mathematics accuracy (42.5% versus 21.7%), despite similar Oracle accuracies (69.4% versus 69.9%). Their FH difference thus suggests an additional disparity in recovering current requirements.

For Llama-3.1-70B and Llama-3.1-8B, code accuracy increases from 16.0% and 13.1% under FH to 36.0% and 31.1% under Oracle. Explicit conditions help both models, but their Oracle performance remains below that of the leading models. These results illustrate how task-solving difficulty and the demands of recovering current conditions jointly affect final-turn accuracy.

D Response-Applied State Tracing

This appendix specifies the response-applied state (RAS) analysis in Section 3.2 and Figure 4. The figure uses Qwen3-8B's first full-history (FH) run on 105 mathematical sequences. The frozen inclusion ledger selects 105 of the 193 mathematics sequences using a stored measurable-GT share of at least 0.95. The included set is fixed across turns. The threshold code uses the stored GT-side shares, without filtering on answer correctness.

Slots, versions, and sources. A tracked parameter is a named, value-bearing component of the task, such as a logarithm base, multiplier, or coefficient. The analysis tracks numerical parameters for which a historical value and source turn can be identified. For each sequence i, turn t, and slot k, the ground-truth (GT) record contains the effective value v_{itk}^{GT} and its source turn s_{itk}^{GT}. Source denotes the earliest turn at which that value content was established for the slot. A value that remains active retains its source; reverting to an earlier value restores its original source. When a slot takes

the same value in several versions, this convention assigns their shared value to the earliest recorded establishment turn. The source is therefore a property of the tracked value under this convention. Rule-like conditions without a traceable value are outside the figure's parameter denominator.

GT is reconstructed from the task data. For supported expression families, the complete-state (Oracle) prompt at each turn supplies the current components; changes between successive states establish their value provenance. For other tracked mathematical slots, a replay of the user prompts supplies candidate values and sources, with the complete-state prompts used to check or resolve identifiable cases. Slots whose source remains unresolved have no historical assignment in the corresponding turn. All provenance records are constructed without consulting the model response.

Evidence in model responses. The RAS procedure analyzes the first FH response at each turn. Its slot-level evidence ledger records the parameter slot, value, response span, evidence type, and matched historical source when available. The slot-level path identifies values bound to a named parameter in an applied calculation. It matches numeric tokens with value and token boundaries, associates them with the relevant parameter name or symbol, and checks their use in a calculation. A statement of a value without observable use is recorded separately as a declaration. Historical recollections, removed conditions, section headings, and isolated final answers do not provide applied-value evidence. A value used for a slot that matches an earlier recorded version receives that version's source; an adopted value with no historical match is recorded as unmatched. Unsupported expressions, ambiguous bindings, and insufficient response evidence retain unresolved status.

A second path handles the supported sum-of-terms expression family. The procedure parses the response's final stated sum into aggregate coefficient and constant totals. A match to the current GT totals assigns the tracked coefficient and constant components to their GT sources. A match to an earlier complete-state total assigns the sources from the latest matching earlier state. Otherwise, coefficient and constant totals are compared separately: components in a matching class receive their current GT sources, and components in a nonmatching class remain outside the historical-source matrix. These assignments use aggregate-expression evidence; several component configurations can have the same total. Aggregate totals and their historical comparisons are recorded separately from slot-level bindings.

Illustrative version assignments. In one mathematical sequence, the turn-8 instruction restores the logarithm base from the turn-5 state, where its effective value is 4. The response applies a base of 2, which has no matching established version through turn 8 and is recorded as an adopted value without a historical source. At turn 9, the response continues to use a multiplier of 2 from turn 4 after the update makes the effective multiplier 4; this response value is assigned to the turn-4 source. The two observations therefore enter different evidence categories: unmatched adoption and an identifiable earlier version.

Source mass and Figure 4. Let $N = 105$. The archived GT and RAS panels use their respective slot ledgers, denoted $\mathcal{K}_{it}^X$ with size K_{it}^X for $X \in \{\mathrm{GT}, \mathrm{RAS}\}$. The GT panel uses canonical components where supported and replay slots otherwise; the RAS panel uses its evidence ledger or aggregate-expression components. Each panel assigns $1/K_{it}^X$ of a sequence–turn's mass to each tracked slot. With $a_{itk}^X(s) \in \{0, 1\}$ indicating a historical-source assignment,

$$H_{s,t}^X = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{K_{it}^X} \sum_{k \in \mathcal{K}_{it}^X} a_{itk}^X(s), \qquad 1 \le s \le t, \tag{1}$$

where a sequence–turn with $K_{it}^X = 0$ contributes zero and remains in the denominator. Unmatched and unresolved entries receive no historical source. The panel-specific slot sets need not coincide, so the figure is a comparison of assigned-source distributions rather than a paired slot-level error rate. A common-slot check is reported below.

Figure 4 displays the distribution of the assigned historical-source mass at each current turn. For $C_t^X = \sum_{r=1}^{t} H_{r,t}^X > 0$, each heatmap cell is

$$P_{s|t}^X = \frac{H_{s,t}^X}{C_t^X}, \qquad \mu_t^X = \sum_{s=1}^{t} s P_{s|t}^X. \tag{2}$$

Table 6: State restoration in GPT-5.4-mini (sequence 298). At turn 8, the response retains the argument power introduced at turn 6.

Turn	State update	Effective expression	Model expression
5	Base $\leftarrow 4$	$2\log_4 256 - 4 = 4$	$2\log_4 256 - 4 = 4$
6	Power $\leftarrow 3$	$2\log_4(256^3) - 4 = 20$	$2\log_4(256^3) - 4 = 20$
7	Base $\leftarrow$ turn 1; remove offset	$2\log_2(256^3) = 48$	$2\log_2(256^3) = 48$
8	Restore turn 5; offset $\leftarrow +10$	$2\log_4 256 + 10 = 18$	$2\log_4(256^3) + 10 = 34$

The overlaid lines and panel (c) plot μ_t^X, the expected source turn among assigned historical-source mass. Panel (c) also shows the identity line $\mu_t = t$. Future source turns $s > t$ are masked. The normalization in Equation 2 is performed after the fixed-denominator aggregation in Equation 1. At turn 11, historical sources account for 0.99 of the raw GT mass and 0.11 of the raw RAS mass; both conditional distributions sum to one in the displayed heatmaps.

D.1 Case Study: Restoring a Historical State

Table 6 follows sequence 298 from its initial question, "Evaluate $\log_2 64$," through a later rollback. The table uses the original user instructions, complete-state reference prompts, and GPT-5.4-mini FH run-1 responses. The recorded prompts match the inputs in the evaluation log at all 11 turns.

At turn 5, the logarithm argument is 256 without exponentiation. Turn 6 adds the power of 3, and turn 7 changes the base while retaining that power. Turn 8 explicitly returns to the complete turn-5 state before changing the offset. GPT-5.4-mini's response begins, "At the end of Turn 5: $2\log_4(256^3) - 4$," and computes $2\log_4(256^3) + 10 = 34$. The expression applies the turn-6 power inside a turn-5 restoration; the effective turn-8 expression gives $2\log_4 256 + 10 = 18$. The incorrect historical component is visible in the calculation itself, and the calculation explains the recorded answer of 34. This GPT-5.4-mini case supplies a response-level illustration alongside the Qwen3-8B cohort summarized in Figure 4.

E State-Injection Diagnostics

This appendix specifies the interventions in Section 3.3 and Figure 5. Panel (a) replaces the current incremental instruction with its paired complete-state prompt. Panel (b) retains the incremental instruction and appends one of three information blocks. Both experiments branch from stored FH trajectories and evaluate the response at the intervention turn. Their cohorts and aggregation procedures are specified separately below.

E.1 Independent Replacement of the Current Instruction

Input construction. For a target turn t, we construct the preceding conversation from the incremental dataset's user prompts and the corresponding model's stored FH run-1 responses at turns $1, \ldots, t-1$. The current user message is the paired complete-state prompt, specifying the effective task after the turn-t update. The FH system prompt is retained. Each target is evaluated independently: a response generated after injection is never used as the history for another target. The FH comparator uses the incremental current instruction; the Oracle comparator comes from a separate trajectory with complete-state instructions at every turn. Thus, injection supplies complete current conditions within a history generated under incremental instructions.

Let S_{FH} denote the FH system prompt, $\mathcal{H}_{m,i,<t}$ the constructed history for model m and sequence i, and u_{it}^{inc} and u_{it}^{full} the paired incremental and complete-state prompts. The inputs are

$$\mathcal{M}_{mit}^{\mathrm{FH}} = [S_{\mathrm{FH}}; \mathcal{H}_{m,i,<t}; u_{it}^{\mathrm{inc}}], \tag{3}$$

$$\mathcal{M}_{mit}^{\mathrm{Inj}} = [S_{\mathrm{FH}}; \mathcal{H}_{m,i,<t}; u_{it}^{\mathrm{full}}]. \tag{4}$$

Table 7: Replacement intervention by model, averaged over turns 2–10. N is the paired sequence count per turn. Accuracy is reported in percent; gain is injection minus FH in percentage points.

Model	N	FH	Injection	Oracle	Gain
Claude Haiku 4.5	543	56.93	76.41	80.87	19.48
Grok 4.5	542	45.86	63.86	67.75	18.00
Qwen3-32B	543	52.94	75.83	74.85	22.90
Qwen3-8B	543	41.97	73.03	68.96	31.06
Ministral-3-8B	543	42.52	66.93	67.55	24.41
Llama-3.1-70B	543	37.00	56.78	58.71	19.79
Llama-3.1-8B	543	25.27	43.65	45.75	18.38
Equal-model mean	—	43.21	65.21	66.35	22.00

Table 8: Turn-level results for the replacement intervention. Accuracy (%) averages the seven models equally; gain is in percentage points. No intervention is performed at turn 1 or turn 11.

Turn	FH	Injection	Oracle	Gain
2	63.42	73.11	72.45	9.68
3	56.76	71.76	71.42	15.00
4	50.42	68.26	68.68	17.84
5	47.21	66.84	68.76	19.63
6	41.90	63.87	64.50	21.97
7	36.45	60.92	62.76	24.47
8	33.55	63.13	64.68	29.58
9	31.48	61.79	64.00	30.31
10	27.71	57.24	59.87	29.52

The brackets denote the ordered chat messages, including their roles. The complete-state prompt contains the task specification and question; the reference answer is supplied to the evaluator.

Models and paired cohort. We evaluate Claude Haiku 4.5, Grok 4.5, Qwen3-32B, Qwen3-8B, Ministral-3-8B, Llama-3.1-70B, and Llama-3.1-8B at turns 2–10. Figure 5(a) uses run 1 for each model and pairs injection, FH, and Oracle records by sequence and turn within that run. The intersection contains 543 sequences at every evaluated turn for six models and 542 for Grok 4.5, yielding 34,200 paired model–sequence–turn observations.

Generation and scoring. The execution logs record a requested output limit of 4,096 tokens for all seven models. The generation runner uses non-streaming requests with no client-side early stop, then extracts the final-answer content for scoring. The shared response-format instructions request a final `<FINAL>` block containing the answer, complete code, or a read-only SQLite query, as appropriate to the domain. Execution commands and generation options are supplied with the runners.

The figure aggregates the archived Boolean correctness verdicts. Injection scoring contexts are constructed from the paired complete-state tasks using the evaluation pipeline's mathematical equivalence, code evaluation, and database execution procedures.

Aggregation. Let $y^p_{mit} \in \{0, 1\}$ denote correctness for model m, sequence i, turn t, and presentation $p \in \{\mathrm{FH}, \mathrm{Inj}, \mathrm{Oracle}\}$. With n_m paired sequences for model m, the plotted accuracy is

$$A^p_t = \frac{1}{7} \sum_{m=1}^{7} \frac{1}{n_m} \sum_{i=1}^{n_m} y^p_{mit}. \tag{5}$$

Sequences receive equal weight within each model, and models receive equal weight in the plotted mean. Table 7 averages these accuracies over turns 2–10; Table 8 gives the complete turn-level curves. These are descriptive point estimates on the recorded runs; run-1 comparisons do not average the additional runs in the main evaluation.

E.2 APPENDED-INFORMATION CONTROLS

Cohort and baselines. We evaluate turns 8 to 10 for sequences with complete original FH run 1 records through turn 10. The evaluated Qwen cohort contains 529 sequences: 187 mathematics, 148 code, and 194 database sequences, giving 1,587 targets. The Haiku cohort contains 531 sequences: 187 mathematics, 148 code, and 196 database sequences, giving 1,593 targets. Every target has a baseline response and responses under all three controls.

Qwen's baseline consists of its original FH run-1 responses rescored with the same pipeline as the controls. Haiku's baseline is generated alongside its controls from the same frozen original FH prefixes, with no appended information block. This baseline response also does not enter subsequent history. All comparisons are paired within model.

Messages and information blocks. Each input contains the FH system prompt, the frozen preceding user messages and model responses, and the current incremental instruction. The selected block is appended to that instruction after two newline characters. The preceding conversation and current instruction are identical across RR, HR, and ES for a given target. Writing b_{it}^{c} for the block under condition $c \in \{\mathrm{RR}, \mathrm{HR}, \mathrm{ES}\}$, the input is

$$\mathcal{M}_{mit}^{c} = [S_{\mathrm{FH}}; \mathcal{H}_{m,i,<t}; u_{it}^{\mathrm{inc}} \parallel b_{it}^{c}], \tag{6}$$

where $\parallel$ concatenates text within the final user message using the two-newline separator. The blocks are constructed as follows.

- **Review reminder (RR):** a fixed, instance-independent instruction to re-examine the conversation, account for changed or removed conditions, and follow references to earlier turns.
- **Historical recap (HR):** the complete prior user prompts from turns 1 through $t-1$, copied verbatim in chronological order with labels of the form `[Turn j]`. The block contains no assistant responses and performs no resolution of updates.
- **Effective state (ES):** the paired complete-state prompt for turn t, copied verbatim after a header identifying it as the current effective task. It supplies the resolved task specification; the reference answer is used by the scorer.

Generation settings. The control runs request temperature 0, with output limits of 3,072 tokens for Qwen3-8B and 4,096 for Haiku. The runner sets its thinking option to disabled and streams responses with a client-side stop at `</FINAL>`. Each target receives one recorded response per condition. Unfinished requests can be retried; completed records are reused when resuming a run. The materials preserve message and appended-block hashes for checking input identity.

Scoring and uncertainty. All four conditions within each model are scored with the same evaluation pipeline, with a shared scoring context for each target across conditions. The full planned denominator is retained; truncated and malformed outputs are processed by the same scorer. For condition c, domain-macro accuracy is

$$A_{m}^{c} = \frac{1}{3}\sum_{d=1}^{3} \frac{1}{n_{md}} \sum_{i \in \mathcal{I}_{md}} \frac{1}{3}\sum_{t=8}^{10} y_{mit}^{c}, \tag{7}$$

where $\mathcal{I}_{md}$ contains the n_{md} sequences in domain d for model m. Gains are paired differences of this statistic. Specifically, the gain over FH is $100(A_{m}^{c} - A_{m}^{\mathrm{FH}})$, and the ES–HR contrast is $100(A_{m}^{\mathrm{ES}} - A_{m}^{\mathrm{HR}})$, both in percentage points. We compute 95% percentile confidence intervals from 5,000 bootstrap replicates, resampling whole sequences within each domain. Each sampled sequence carries all three turns and all conditions, preserving the pairing. Intervals describe sequence-sampling uncertainty conditional on the recorded responses.

E.3 CONTROL RESULTS AND INTERPRETATION

Table 9 reports absolute accuracy, gains over the model-specific FH baseline, and the direct ES–HR contrast. RR intervals include zero for both models. HR produces small positive gains, and ES

Table 9: Appended-information controls at turns 8–10. Accuracy is domain-macro accuracy (%). Gains and 95% confidence intervals are in percentage points. ES–HR reports the direct paired contrast. FH uses the model-specific baseline described in the text.

Model	Condition	Accuracy	Gain	95% CI
Qwen3-8B	FH	26.41	—	—
	RR	26.63	+0.21	[−1.20, 1.63]
	HR	28.28	+1.86	[0.30, 3.44]
	ES	62.23	+35.82	[32.52, 39.23]
	ES–HR	—	+33.95	[30.85, 37.30]
Claude Haiku 4.5	FH	47.58	—	—
	RR	47.45	−0.13	[−1.29, 1.11]
	HR	49.00	+1.43	[0.21, 2.70]
	ES	69.01	+21.43	[18.75, 24.15]
	ES–HR	—	+20.01	[17.43, 22.58]

Table 10: Accuracy (%) by domain for the appended controls, averaged over turns 8–10. The final column reports the ES gain over the model-specific FH baseline in percentage points.

Model	Domain	FH	RR	HR	ES	ES–FH
Qwen3-8B	Mathematics	27.27	26.56	31.19	69.70	42.42
	Code	32.21	33.56	34.91	60.81	28.60
	Database	19.76	19.76	18.73	56.19	36.43
Claude Haiku 4.5	Mathematics	70.05	69.16	72.19	83.42	13.37
	Code	50.23	50.90	52.03	63.74	13.51
	Database	22.45	22.28	22.79	59.86	37.41

exceeds HR by 33.95 points for Qwen and 20.01 points for Haiku. ES gains span all three domains: 42.42, 28.60, and 36.43 points in mathematics, code, and databases for Qwen, and 13.37, 13.51, and 37.41 points for Haiku. Across the individual targets, ES changes 603 Qwen answers from incorrect to correct and 26 from correct to incorrect; the corresponding Haiku counts are 378 and 23. These counts give each target equal weight, whereas the reported accuracy gains weight domains equally. Table 10 reports all four conditions separately for each domain.

The replacement experiment measures recovery with the accumulated FH history retained. The appended controls place the reminder, prior user updates, or resolved task specification in the current message. HR makes earlier instructions locally available while preserving their revisions and conflicts; ES supplies the resulting valid conditions. Their difference measures the benefit of these two information presentations under a fixed preceding history. The blocks differ in content and length; input token counts are not matched. The recovery therefore demonstrates successful task solving within retained long histories when current conditions are explicit, with the direct effect of token count remaining a separate quantity.

E.4 Additional Checks on the Appended Controls

RR, HR, and ES share the frozen preceding messages, the current update, and within-model scoring context. RR controls for a generic review instruction; HR repeats prior user instructions in chronological order; ES supplies their resolved current requirements. Table 11 adds two checks from the existing records.

First, we count Unicode characters in each complete appended block, including its header. ES is shorter than HR for 1,581/1,587 Qwen targets and 1,587/1,593 Haiku targets. This records the size difference explicitly; it is not token matching or an independent manipulation of input length. Second, we recompute ES–HR after excluding pairs where either response has `finish_reason=length`. This conditions on observed completion and is reported as a sensitivity check; the full-cohort comparison remains the primary result. Intervals use 5,000 sequence-

Table 11: Checks using frozen appended-control records. Lengths are median [interquartile range] characters. ES–HR gains and 95% intervals are percentage points.

Statistic	Qwen3-8B	Claude Haiku 4.5
RR block length	610 [610, 610]	610 [610, 610]
HR block length	1,560 [1,181, 2,000]	1,564 [1,183, 2,002]
ES block length	836 [462, 1,199]	837 [462, 1,209]
HR / ES length-stop counts	23 / 14	0 / 2
Pairs without either length stop	1,563	1,591
ES–HR on these pairs	34.07 [30.84, 37.24]	20.10 [17.53, 22.72]

bootstrap replicates stratified by domain (seed 20260926). Within each domain, remaining targets receive equal weight; the three domain means are then averaged equally.

Scope of the comparison. HR contains user messages within its appended block, while the preceding FH context still includes assistant responses. Thus it does not implement a separate user-history-only conversation. The recorded controls also do not independently match length, layout, or the authority signaled by the ES header. The supported comparison is between the complete specified prompt conditions with a fixed preceding history. These distinctions define the intervention measured by the reported gains.

E.5 Prompt Templates and Reproducibility

Shared system prompt. The replacement intervention and appended controls use the following FH system prompt:

> You are a helpful assistant. Answer each user request normally and provide any explanation or reasoning that you consider appropriate.
>
> At the very end of every response, emit exactly one final-answer block in this exact form: `<FINAL>\boxed{...}</FINAL>`
>
> The content inside `\boxed{...}` is the answer that will be evaluated. It must be complete and self-contained. End the response immediately after `</FINAL>`.
>
> For programming tasks, place the complete final code inside `\boxed{...}` as plain code, without Markdown code fences.
>
> For database tasks, place exactly one complete read-only SQLite query inside `\boxed{...}` as plain SQL, without Markdown code fences or any additional SQL statement. The query must begin with `SELECT` or `WITH`; this final block is the sole database answer that will be evaluated.
>
> For mathematical tasks, place the final number, expression, equation, formula, set, or other mathematical answer inside `\boxed{...}`.
>
> For other tasks, place the concise final answer that should be evaluated inside `\boxed{...}`.
>
> Do not use `<FINAL>`, `</FINAL>`, or `\boxed{...}` anywhere except for the single final-answer block at the end of the response.

Review reminder. The RR block is identical for every target:

> Before answering, re-examine the entire conversation history. Keep in mind that:
>
> - conditions stated in earlier turns may have been modified, overridden, superseded, or explicitly removed by later turns;
> - some earlier conditions may have been reverted or cancelled, so an earlier value or rule may no longer apply;
> - if the current request refers back to an earlier turn, follow that reference exactly as it was stated at that turn;
> - your answer must reflect the requirements that are currently in effect at the latest turn.
>
> Answer the current request as stated above. Do not repeat or restate this reminder.

Historical recap and effective state. HR begins with the header "Prior user requests in this conversation (Turns $1..t-1$)" and the note: "The current request above is the latest update in this conversation; the earlier requests are listed for reference in chronological order." The turn-labelled user prompts follow. ES begins with "Current effective task (complete, self-contained)" and the note: "The block below restates the complete, currently effective task specification after all updates so far. Use it together with the current request above." The paired complete-state prompt follows. Both headers use the Markdown prefix ##. The frozen materials preserve the exact Markdown headers, whitespace, and prompt text.