WHEN USERS CHANGE THEIR MINDS: MEASURING AND REPAIRING INTENT DRIFT IN LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents often operate over multi-turn interactions in which user intent changes before execution. We study intent drift: the failure mode in which superseded parts of the user's intent continue to influence the final answer or tool action. We introduce INTENTFLUX, an executable benchmark that converts verifiable tasks into dialogues with controlled intent changes while preserving their original graders. In a 627-case calibration, mean task score falls from 0.476 to 0.384 as dialogues contain more superseded and withdrawn information. Across eight models, the rate of fully correct solutions is significantly lower when the same final task must be recovered from an evolving dialogue rather than given directly in a single turn. We further introduce STATEFORGE, which explicitly maintains the active requirements before generation. On GENERAL-TEST, it improves mean task score from 0.367 to 0.467. Providing the ground-truth final state improves performance further but still does not recover single-turn performance, indicating that state-estimation errors explain only part of the gap. These results establish intent drift as a measurable multi-turn failure mode and explicit state maintenance as a partial mitigation.

1 INTRODUCTION

LLM agents often work with users who specify a task incrementally. A user may first request an action, later revise a parameter, withdraw a constraint, and finally ask the agent to act on the resulting specification. The agent must then determine which parts of the user's intent remain active. Intent drift occurs when superseded parts of the user's intent still influence the final answer or tool action.

Existing multi-turn evaluations establish two important but incomplete parts of this problem. Sharded-instruction settings show that models can lose track of a task when its information is disclosed gradually (Laban et al., 2025). More recent evolving-intent evaluations add argument reveal, revision, and task switching while preserving the source verifier (Tack et al., 2026). These settings measure whether an agent ultimately follows an evolving task, but they do not distinguish failures caused by obsolete intent continuing to influence the answer from other multi-turn errors. Studying this failure directly requires a known final intent and stale information whose erroneous use changes the task outcome.

INTENTFLUX addresses this gap by converting executable source tasks into controlled multi-turn interactions in which user intent changes through additions, deletions, and replacements. Because each interaction is constructed from a known source task, the benchmark retains the intended final task while controlling how earlier intent is introduced, revised, or withdrawn. We introduce stale information in two ways: a VARIANT is a plausible alternative value later replaced by the value in the final intent, while a DECOY is a plausible constraint later withdrawn. We retain a decoy only when obeying it changes the source-task grader's verdict.

On a calibration pool of 627 source tasks, mean task score falls from 0.476 in the easy condition to 0.384 in the hard condition, a 19.3% relative reduction. To test whether this effect is specific to the calibration model, we evaluate eight recent LLMs and find lower performance for the evolving-dialogue condition at every difficulty level. A length-matched control without intent revisions scores 0.734, close to the 0.778 single-turn condition and well above the corresponding drift score of 0.469, showing that additional turns alone do not explain the loss. In a live tool-use environment, withhold-

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.

ing an explicit restatement of the final intent lowers partial-credit score by 0.103 relative to the clean condition, extending the effect beyond static final-answer tasks.

We next ask whether compressing dialogue history is sufficient. We compare two external harnesses: Deep Agents uses rolling summarization and OpenHarness uses two-phase compaction. On GENERAL-TEST, our fixed 125-task test set, they score 0.354 and 0.392, respectively, compared with 0.367 for bare multi-turn execution. Thus, compression alone does not reliably recover the lost performance: the generator may still need to determine which retained information remains valid.

Motivated by this observation, we introduce STATEFORGE, which separates intent-state maintenance from downstream task solving. A model-driven tracker removes superseded intent items and conclusions derived from them before supplying the active state to base agent. STATEFORGE raises mean task score on GENERAL-TEST to 0.467. Replacing the estimated state with the ground-truth final intent further raises performance to 0.549, but still falls below the clean single-turn score of 0.778, showing that state-estimation errors are material but explain only part of the remaining gap.

This decomposition motivates two complementary deployment paths: maintaining state with a smaller trainable tracker, or internalizing the state update into the base agent. In the modular path, a 9B tracker is statistically indistinguishable from the 122B reference, and we use on-policy distillation (OPD) to improve a 2B tracker from 0.266 to 0.445. In the second path, thinking distillation internalizes state-folding behavior into a standalone 35B agent, improving its GENERAL-TEST score from 0.296 to 0.461 without an external harness.

Our contributions are:

- We introduce INTENTFLUX, a benchmark for intent drift. Its controlled intent edits make stale intent measurable through its effect on source-task success.
- We establish a controllable intent-drift stress condition and show that the evaluated history-management harnesses leave substantial errors, while STATEFORGE repairs part of the gap by maintaining an explicit active state before generation.
- We show that state estimation is a material component of state folding, while exact-state injection reveals residual error under the retained dialogue history. We further study two complementary transfer paths: distilling state maintenance into a smaller tracker and internalizing state-folding behavior into the base agent.

2 RELATED WORK

Evolving intent and memory updates. Multi-turn benchmarks study instruction retention and conversation-level task completion (Kwan et al., 2024; Sirdeshmukh et al., 2025; Katsis et al., 2025; Laban et al., 2025). EvolIF evolves per-topic constraints through addition, deletion, and modification (Jia et al., 2026); InterruptBench studies revisions and retractions in long-horizon web navigation (Zou et al., 2026); and Tack et al. (2026) converts verifiable tasks into reveal/revision/switch interactions. Long-term-memory work similarly penalizes use of invalidated memories (Uddin et al., 2026) or trains agents to prefer current over superseded values (Patel, 2026). INTENTFLUX complements these settings with heterogeneous executable tasks, controlled revisions and withdrawals, and decoys retained only when obeying them changes the source-task verdict.

Context management and state tracking. Long-context methods compress or retrieve growing histories (Liu et al., 2023; Jiang et al., 2023; Lee et al., 2024; Zhang et al., 2025). Such methods need not explicitly resolve which parts of the prior intent remain active. STATEFORGE instead folds the edit history into an active requirement state before answer generation. This design also relates to entity and belief tracking (Kim & Schuster, 2023; Zhu et al., 2024), but operates over open-ended task requirements and their dependencies in executable settings.

Learning state maintenance. Rationale distillation and deliberative training teach intermediate reasoning behaviors (Hsieh et al., 2023; Mukherjee et al., 2023; Guan et al., 2024); on-policy distillation trains students on their own rollouts against a teacher (Agarwal et al., 2023). We adapt these ideas to two settings: the tracker sub-role and a full agent, using executable stale-intent failures to supervise state-maintenance behavior.

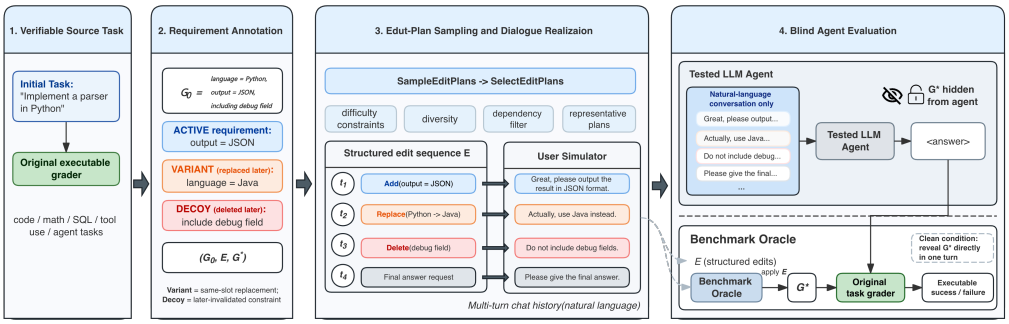

Figure 1: INTENTFLUX turns a verifiable source task into a multi-turn interaction with controlled intent edits. The agent observes only the dialogue; the oracle applies the edit plan to obtain the final active intent $G^\star$, and the original grader evaluates the final output or action.

3 THE INTENTFLUX BENCHMARK

INTENTFLUX measures whether an agent follows the user's *current* intent as that intent changes during a conversation. It turns verifiable source tasks into controlled multi-turn interactions while preserving their original graders (Figure 1).

3.1 EVOLVING INTENT STATES

At the intent-state level, a task is represented as a finite set of atomic natural-language goals, $G = \{g_1, \ldots, g_n\}$. An example consists of an initial intent state G_0, a sequence of edits $E = (e_1, \ldots, e_K)$, and the final active intent $G^\star$. The edits transform the state through

$$\textsc{Add}(g) : G_{t+1} = G_t \cup \{g\}, \qquad \textsc{Delete}(g) : G_{t+1} = G_t \setminus \{g\},$$
$$\textsc{Replace}(g, g') : G_{t+1} = (G_t \setminus \{g\}) \cup \{g'\}.$$

The oracle applies the edit sequence to obtain $G^\star$, against which the source-task grader evaluates the candidate output.

INTENTFLUX creates stale intent in two ways. A VARIANT is a plausible alternative value for an active intent item that is later replaced by the target value, such as an alternative implementation language. A DECOY is a plausible intent item that the user later withdraws. We retain a decoy only when obeying it changes the source grader's verdict. Consequently, obeying a retained decoy is grader-consequential rather than merely a lexical mismatch. This construction makes stale-state use testable through task success, although an aggregate failure need not be attributable to a particular stale item without inspecting the trajectory and output.

3.2 SOURCE TASKS AND CONSTRUCTION

The General track draws on code, math, SQL, data-to-text, summarization, and tool-use tasks (Chen et al., 2021; Jain et al., 2024; Cobbe et al., 2021; Yu et al., 2018; Parikh et al., 2020; Laban et al., 2024; Patil et al., 2025). These tasks provide a range of output spaces while retaining executable or task-native grading. The interactive track uses VitaBench source tasks (He et al., 2025), where the agent acts in delivery, in-store, OTA, and cross-domain environments. The General track supports the stress, paired, harness, and training experiments; VitaBench supplies the interactive-transfer tasks. Full source-set and grader mappings appear in Tables 3–4.

We distinguish a 627-task General-track calibration pool, a separate 502-task training pool, and two fixed evaluation sets. The calibration pool is used only to calibrate the benchmark's difficulty response, with the same 627 source tasks instantiated under each difficulty stratum. The training pool is used for the OPD and thinking-distillation experiments and is disjoint from GENERAL-TEST. GENERAL-TEST is a fixed, stratified General-track test set of 125 source tasks: 10 LiveCodeBench, 21 GSM8K, 10 HumanEval, 21 Spider, 24 ToTTo, 18 SummHay, and 21 BFCL cases.

The same source cases are used at each difficulty level and for the control and harness comparisons. INTERACTIVE-TEST is a fixed set of 100 VitaBench source tasks used only for interactive transfer. Subsequent references use these set names rather than their sample counts.

At the source-task level, we decompose each problem into ordered atomic shards and convert them into target goals that populate $G^\star$. For most task families, source shards map directly to target goals; task-specific loaders may instead construct goals from other source fields as the task format requires. Variants and decoys are additional goals introduced during intent editing and need not correspond to source shards. For standard task families, a shard is treated as atomic when changing it changes the task's required output or action. We annotate behavior-changing variants and plausible decoys, and sample edit plans subject to semantic dependency constraints. Human annotators verify the variant, decoy, and semantic-dependency pools; a downstream audit recovers the annotated final intent for 95% of GENERAL-TEST. Full annotation and quality-control procedures are in the appendix.

3.3 CONTROLLED STALE-INFORMATION LOAD

The General-track calibration study uses a monotone construction budget $b \in \{1, 3, 5\}$ for the easy, medium, and hard strata. At budget b, the sampler exposes up to b variants per goal and up to b eligible decoys, under the same dependency and feasibility rules. The source cases, target intent, source-task grader, sampling procedure, and simulator policy are otherwise shared across strata. Because dependencies and random initialization determine which candidates enter a realized trajectory, the observed variant/decoy counts can be below their budgets; we report both the schedule here and the realized means in Table 5.

Increasing b jointly increases the number of superseded values and withdrawn intent items that the model must discard. It also produces more edit turns. We therefore treat this calibration as a controlled test of *joint stale-information load*, not as a factorial estimate of the separate effects of variants, decoys, or length. We separately test whether dialogue length alone can explain the degradation using a length-matched control in Section 5.1. After the final edit, the simulator adds only a neutral request for the final answer; it does not restate or summarize $G^\star$. Thus, none of the difficulty strata receives an explicit final-intent restatement.

4 EVALUATION PROTOCOL

Each evaluation couples a user simulator, a tested model, and the source task's native evaluator. The simulator realizes (G_0, E) as a multi-turn dialogue, while the final intent $G^\star$ determines the target task scored by the source evaluator. The simulator maintains a queue for each active goal and a separate decoy queue. At each turn, it selects a feasible edit, expresses it as a natural user utterance, and advances only the selected queue. This preserves the order of edits to one goal while allowing edits to multiple goals to be naturally interleaved.

Across all reported experiments, DeepSeek-V4-Flash serves as the user simulator and as the rubric judge whenever an LLM-based judgment is required. The tested agent is kept separate from these roles and varies with the comparison.

For paired evaluations, we also construct a clean condition that reveals $G^\star$ directly. Each task-native evaluator returns a normalized score $s_i \in [0, 1]$. We define full credit as $c_i = \mathbf{1}[s_i = 1]$ and report

$$\begin{aligned} \mathrm{CAR}(\mathrm{drift}) &= \mathbb{E}[c_i \mid \text{multi-turn drift}], \\ \mathrm{CAR}(\mathrm{clean}) &= \mathbb{E}[c_i \mid G^\star \text{ revealed directly}], \\ \mathrm{IDG} &= \mathrm{CAR}(\mathrm{clean}) - \mathrm{CAR}(\mathrm{drift}). \end{aligned}$$

Thus, the current-ground-truth-intent alignment rate (CAR) is the fraction of the entire evaluation set receiving full task-native credit under the current ground-truth intent, and the intent-drift gap (IDG) is its case-paired clean–drift difference. A positive IDG indicates lower task completion when the same final task must be recovered from an evolving interaction rather than presented directly. Clean and drift conditions share the source case, final intent, grader, and tested model, holding task identity and model capability fixed across the pair. The conditions intentionally differ in interaction form, however; IDG captures the total performance loss associated with recovering the final intent from the dialogue rather than isolating a single causal factor. We therefore interpret IDG together with a

separate turn-matched no-drift control, whose narrower purpose is to test whether additional turns alone account for the loss. Key comparisons use case-paired bootstrap confidence intervals.

Code, math, database, and tool-use scores are binary. ToTTo and SummHay produce continuous task-native scores; for CAR, they count as success only at full credit ($s_i = 1$), with no tuned threshold. They remain in the denominator, so CAR on GENERAL-TEST always uses all 125 expected cases, not only the 83 binary-task cases. CAR therefore serves as a strict full-task-completion measure, while mean score captures partial credit on continuous-output tasks. If the tested model fails to produce a scorable output, the case receives $s_i = c_i = 0$ and remains in the denominator.

We separately report the case-micro mean $N^{-1} \sum_i s_i$, denoted *mean score*. Because this average mixes task-native metrics, we use it only for within-benchmark comparisons under the fixed task composition and accompany it with per-family results. The simulator queueing, grader dispatch, model configurations, and reproducibility details are provided in the appendix.

5 MEASURING INTENT DRIFT

We first test whether increasing the construction budget produces a monotonic difficulty response. The calibration study uses `gpt-4.1` for construction, `gemini-3.5-flash` as the tested model, and `deepseek-v4-flash` as the user simulator and rubric judge. The easy, medium, and hard strata use variant/decoy budgets of 1, 3, and 5, respectively. Increasing this joint budget lowers both mean score and CAR (Table 5). Because the additional edits also lengthen the dialogue, this calibration exhibits a monotonic response to increasing joint stale-information load; it does not estimate separate causal effects for variants and decoys.

Table 1: Main benchmark results on GENERAL-TEST ($n = 125$ per condition). IDG is the clean–hard CAR difference; brackets give paired 95% confidence intervals.

	Full-credit CAR				
Tested model	Clean	Easy drift	Medium drift	Hard drift	Hard IDG [95% CI]
Claude Opus 4.7	0.592	0.344	0.240	0.184	0.408 [0.304, 0.512]
DeepSeek V4 Pro	0.584	0.368	0.376	0.200	0.384 [0.296, 0.472]
Gemini 3.5 Flash	0.632	0.384	0.320	0.264	0.368 [0.280, 0.456]
GLM 5.2	0.576	0.368	0.304	0.232	0.344 [0.264, 0.432]
GPT-5.4	0.608	0.376	0.264	0.224	0.384 [0.304, 0.472]
HY3	0.592	0.328	0.304	0.200	0.392 [0.304, 0.480]
Qwen3.6 Plus	0.640	0.448	0.312	0.192	0.448 [0.360, 0.536]
Qwen3.5-122B	0.592	0.288	0.240	0.232	0.360 [0.280, 0.448]

The paired validation shows a positive IDG for every tested model in the main benchmark (Table 1) and at every stratum. The loss generally grows with joint stale-information load; DeepSeek V4 Pro is the only model whose medium point estimate is slightly below its easy estimate. The raw score grid in Table 8 shows the same overall degradation pattern without collapsing continuous task scores into full-credit indicators. Separately, the turn-matched comparison in Table 10 shows that adding turns without changing the final intent has a much smaller effect than the corresponding drift trajectories.

5.1 TASK-FAMILY RESPONSES AND A TURN-MATCHED CONTROL

The aggregate trend is not driven by a single task family (Table 7). LiveCodeBench, GSM8K, and HumanEval show the largest declines, while BFCL is an exception: its score rises under the jointly constructed condition. SummHay remains near a low-score floor. These heterogeneous responses motivate reporting both aggregate and per-family results. Exact family scores appear in Table 7.

The case-micro aggregate gives larger families more weight. As a composition-robust check, an equal-family macro average over the seven task-family rows also decreases monotonically, from .497 (easy) to .433 (medium) and .385 (hard). Thus, the headline trend does not depend on weighting task families by their number of cases. Case-paired IDG by task family is reported in Table 9.

The calibration trajectories also lengthen the dialogue, from 13.2 turns on average in easy to 37.1 in hard. We therefore construct a turn-matched no-drift control on GENERAL-TEST. It presents the final intent in the first turn and uses neutral no-change confirmations thereafter. The control and drift conditions have identical turn counts and comparable user-message length. The control scores .734, close to the .778 single-turn condition, whereas the corresponding drift trajectories score .469. The .265 gap shows that turn count alone does not explain the drift loss.

This control is intentionally narrow: it removes revisions while matching length, since matching competing values and their withdrawal would reintroduce the stale-information mechanism being tested. It therefore does not identify separate effects of wording, edit type, or variant versus decoy; it only rules out generic long-dialogue degradation as the explanation.

5.2 INTERACTIVE TRANSFER

The General track tests a static final answer after a simulated dialogue. We therefore also evaluate INTERACTIVE-TEST, in which the agent acts in a live tool-use environment. Qwen3.5-122B is the tested agent; DeepSeek-V4-Flash is the user simulator and rubric judge. The main condition does not restate the final intent during the closing turns; we retain the original closing protocol, which does restate the final intent, only as a diagnostic.

Table 11 shows that the no-restate condition lowers rubric score by 0.103 relative to clean, while explicitly restating the final intent in the diagnostic condition masks this gap. The binary CAR remains unchanged across the non-oracle conditions, so this evidence concerns partial satisfaction in the interactive protocol.

6 FROM HISTORY COMPRESSION TO EXPLICIT STATE FOLDING

We first ask whether existing history-management strategies are sufficient to handle evolving intent. We compare two external harnesses on the same frozen medium-difficulty GENERAL-TEST trajectories, using Qwen3.5-122B as the tested model and DeepSeek-V4-Flash as the user simulator and rubric judge. Deep Agents applies rolling summarization, while OpenHarness uses two-phase compaction. Under the same evaluator, temperature, and output budget, Deep Agents scores 0.354 and OpenHarness scores 0.392, compared with 0.367 for bare multi-turn execution (Table 2, Panel B). Thus, the evaluated history-management harnesses do not reliably recover the performance lost under evolving intent. These results motivate a closer look at what history compression leaves unresolved. A compressed history can still preserve both a superseded value and its replacement, leaving the generator to determine which one remains active while solving the downstream task.

Stale information can also persist indirectly. An obsolete intent item may already have produced intermediate conclusions, plans, or choices. Even if the original item is later removed, these dependent conclusions can remain and continue to influence generation. A summary or compacted history may preserve them because they appear locally consistent despite depending on outdated intent. This motivates *state folding*: resolving the edit history into an explicit representation of the current intent before downstream task solving. State folding removes not only superseded intent items but also conclusions that depend on them, while preserving information that remains valid after the edit. History management asks what information to retain; state folding additionally asks what information is still valid.

7 STATEFORGE: STATE FOLDING AND TRANSFER

STATEFORGE is a train-free harness that converts an evolving conversation into an explicit active-state estimate. After each user turn, a tracker folds observed intent edits into the current active state. Deleting or replacing an intent item removes it from the state; conclusions derived from that item are invalidated as well. The state distinguishes directly stated intent items from derived conclusions so that dependent conclusions can be removed when their supporting intent changes.

For generation, the updated state is inserted between the cached dialogue history and the current user input, keeping the prior conversation cacheable while making active information recent at generation time. A separate format tracker maintains output conventions, while a final gate checks the generated answer against those conventions and can trigger regeneration. The generator therefore receives an explicit active-state estimate before solving the downstream task (Figure 2).

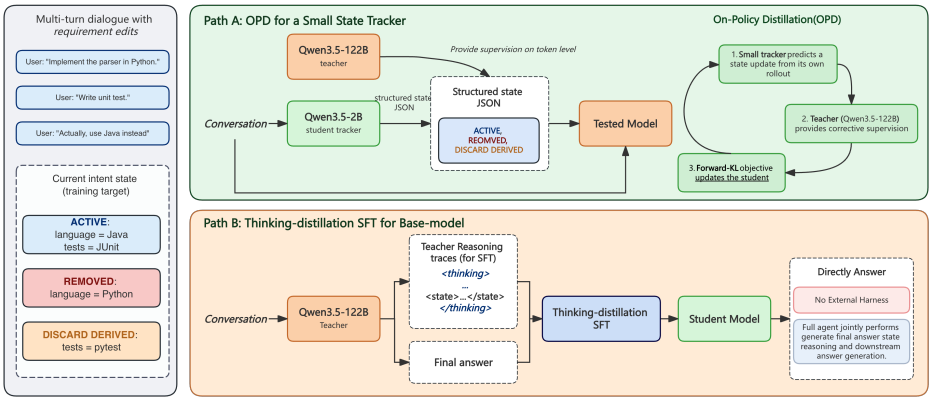

Figure 2: StateForge folds an evolving dialogue into an explicit active state for external generation. The same state-update behavior supports two transfer paths: on-policy distillation to a lightweight tracker and thinking distillation to a standalone agent.

Table 2: State folding on General-Test. StateForge improves over bare multi-turn execution by 0.100 mean score (95% CI $[0.030, 0.171]$).

Setting	Mean score	CAR
Panel A: state-estimation analysis		
Clean single-turn	0.778	–
Bare multi-turn	0.367	0.264
GT-oracle final state	0.549	–
StateForge	**0.467**	**0.368**
Panel B: external-harness comparison		
Deep Agents (rolling summary)	0.354	0.256
OpenHarness (compaction)	0.392	0.296
StateForge	**0.467**	**0.368**

7.1 State Folding Repairs Intent Drift

On General-Test, StateForge raises mean score from 0.367 for bare multi-turn execution to 0.467 (Table 2, Panel A). Supplying the ground-truth final state raises the score further to 0.549, showing that state-estimation error remains material. This oracle supplies exact final-state information at the final generation step while retaining the full dialogue history; it is therefore not equivalent to the clean single-turn condition. Its remaining 0.229 gap to the clean score of 0.778 shows that errors remain even with exact state information under the retained dialogue history. In the external-harness comparison, StateForge also achieves higher scores than the evaluated rolling-summary and compaction harnesses (Panel B). All harness rows use Qwen3.5-122B as the tested model and DeepSeek-V4-Flash as the simulator and rubric judge. Mean score includes the continuous data-to-text and summary metrics; CAR is the full-credit rate over the complete test set.

7.2 Tracker Capacity and Transfer

The ground-truth-state result shows remaining headroom in final-state estimation. We next ask whether this tracking role requires a model as large as the base agent. In the tracker-scale comparison, 2B and 4B trackers are significantly below the 122B reference, whereas the 9B tracker is statistically indistinguishable from it. We therefore use the 9B result as evidence that the modular state-estimation role does not require a tracker at base-agent scale.

We next train the tracker sub-role while keeping the 122B base agent fixed. One OPD iteration raises a 2B tracker from 0.266 to 0.445 on General-Test, while 122B tracker scores 0.428 under the

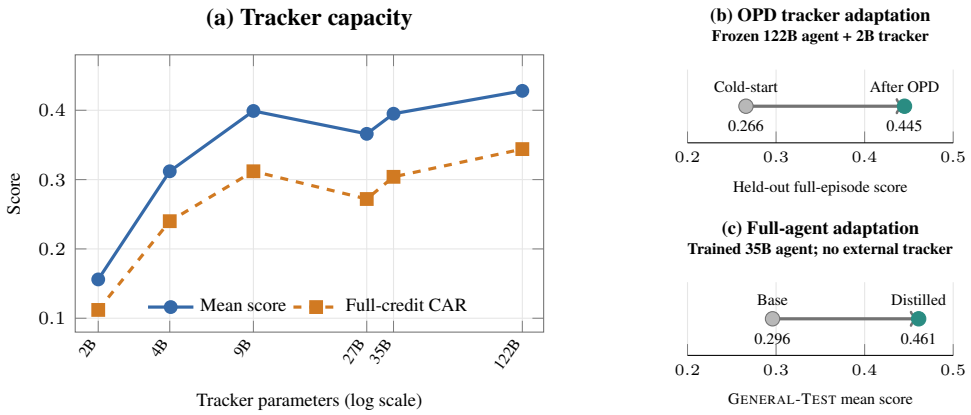

Figure 3: Tracker capacity and two distinct adaptation settings. (a) Tracker results are positioned by parameter count on a logarithmic axis. (b) OPD updates a 2B tracker paired with a frozen 122B base agent; values are held-out full-episode scores. (c) Thinking distillation updates a standalone 35B agent with no external tracker; values are GENERAL-TEST mean scores. Panels (b) and (c) use different system configurations and evaluation protocols, so only the before–after change within each panel should be compared. Tracker-scale confidence intervals appear in Appendix 13.

same evaluation protocol. Training pool is disjoint from GENERAL-TEST. Because the generator remains frozen, this improvement isolates adaptation of the tracker within the modular stack.

Finally, in a separate training setting, we explore whether state-folding behavior can be internalized into the base agent. Thinking distillation raises a standalone 35B agent from 0.296 to 0.461 on the GENERAL-TEST drift condition without an external harness. This is not a direct system comparison with the OPD tracker stack; rather, it tests a different deployment in which state maintenance is absorbed into the agent itself.

7.3 FROM STATE TRACKING TO AGENT BEHAVIOR

The transfer experiments target two distinct deployment choices. In the first, the generator remains unchanged and a smaller tracker supplies the active state at inference time. This separates state maintenance from downstream task solving: the tracker determines which intent items remain active after an edit, while the base agent continues to solve the source task. The scale comparison and OPD result show that this role can be performed by a substantially smaller model and improved through on-policy correction.

In the second deployment, the external tracker is removed. Thinking distillation exposes a 35B agent to teacher rollouts that make state updates explicit in its reasoning, then evaluates the trained agent on the GENERAL-TEST drift condition without a harness. The result does not establish that all state-folding operations have been learned or that the learned behavior generalizes to every update regime. The improvement is also accompanied by a modest decrease on the clean condition: the distilled model scores .718, compared with .745 for the base model.

These two paths answer complementary questions. The tracker path asks whether current intent can be maintained efficiently as a modular operation; the full-agent path asks whether the same behavior can be absorbed into the model itself. Keeping the two settings separate makes their training and deployment assumptions explicit and enables future work to compare modular and learned state maintenance under a shared evaluation protocol.

7.4 WHAT THE ABLATION ESTABLISHES

The system-level comparison above does not determine whether the gain comes specifically from the explicit state representation or from other components of the harness. We therefore compare the complete harness with gate-free state injection and two recap baselines matched to the full harness in model calls and token budget under a separate component-matched re-run. At $n = 125$, the paired differences between each recap baseline and the full harness have 95% confidence intervals that include zero, so the current experiment does not establish representation superiority over a

matched recap (Table 12). The oracle condition, by contrast, shows a positive paired difference from the full harness, demonstrating material headroom from exact state estimation within the same history-preserving interface. This result does not identify the tracker as the dominant source of the clean–drift gap. Detailed tracker-scale results and training progression are reported in the appendix.

8 Analysis, Limitations, and Broader Impact

8.1 What the Benchmark Measures

The variant/decoy calibration provides a controllable stress condition. Its $b \in \{1, 3, 5\}$ schedule jointly introduces more superseded alternatives and more withdrawn constraints, while also lengthening the edit sequence. The simulator does not close by restating the final intent. The resulting score decline therefore characterizes a joint stale-information stress condition rather than the separate causal effects of variants, decoys, or length. The intent-drift claim does not rest on this slope alone: case-paired clean–drift gaps remain positive across models and strata, and the turn-matched control shows substantially less degradation than the drift condition. Per-family results further show heterogeneous sensitivity to the construction: code and math decline sharply, whereas BFCL improves under the same setting.

Trace inspection suggests that intent drift is not limited to explicit retention of deleted content. In drift cases containing a DELETE edit, agents often omit the deleted goal from the final answer. In BFCL, a common REPLACE failure instead binds the new value to the wrong function-call slot. These observations motivate state tracking that represents the identity and dependencies of intent items, rather than relying only on retrieval over prior utterances.

8.2 Simulator and Evaluator Validity

The user simulator is an LLM, so its wording is part of the evaluation protocol. Diversified utterance prompts, explicit final-answer triggers, and per-goal edit queues constrain how scheduled edits are realized. In a sampled audit of 100 edit-phase utterances from the DeepSeek-V4-Flash simulator used throughout our experiments, fidelity reaches 96% and coherence reaches 97%. The audit indicates that invented and contradictory intent changes are uncommon in the sampled dialogues.

The interactive VitaBench study exposes a related protocol issue. The main interactive condition does not restate the final intent. In a separate diagnostic condition, supplying the complete final intent during the closing turns removes the measured partial-credit drift gap. A benchmark can therefore inadvertently supply the state that it intends to test; the restating condition is not used as a difficulty factor.

8.3 Limitations

IntentFlux represents current intent as a finite set of atomic goals and therefore targets tasks with executable or decomposable intent structure. The calibration couples variant and decoy budgets, supporting a joint stale-load effect rather than separate marginal effects; the turn-matched control rules out turn count alone, but message ordering and edit wording remain properties of the drift construction. The external-harness comparison and Interactive-Test each use a single tested model (Qwen3.5-122B), the former on replayed trajectories. The component-matched ablation does not isolate the explicit state representation as the sole source of the system-level gain, and the scaling behavior of both transfer paths remains open.

9 Conclusion

IntentFlux measures whether agents follow current user intent after earlier information is superseded or withdrawn; its construction makes obsolete intent grader-consequential and yields a controlled difficulty response. The evaluated history-management harnesses leave substantial errors, while StateForge, which maintains an explicit active state, improves performance. This tracking role does not require a base-agent-scale model: OPD further improves a small tracker, and thinking distillation begins to internalize the behavior into the agent. Together, these results frame intent drift as a measurable state-maintenance problem that modular and learned approaches can partially mitigate.

AI USE STATEMENT

Generative AI tools assisted language editing of portions of the manuscript and construct dataset, user simulation, and work as judging components. The authors reviewed, executed, and verified AI-assisted code and reviewed every edited passage. The authors take responsibility for the final text, artifacts, and claims.

ETHICS STATEMENT

All tracks are constructed from public research benchmarks under their original terms; no human-subject data or personal information is collected. Acting on withdrawn requirements can cause harm in high-stakes assistant settings. The released benchmark content is synthetic and task-focused.

REPRODUCIBILITY STATEMENT

The appendix specifies benchmark construction, evaluation, graders, and model configurations. The anonymized supplement contains the harness, construction and evaluation pipelines, frozen GENERAL-TEST data, and archived per-case traces for the reported frozen-protocol runs, with aggregation scripts for reported aggregates. Commercial-API traces are not redistributable; the corresponding evaluation pipeline is included.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 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, 2021. URL https://arxiv.org/abs/2110.14168.

Melody Y. Guan, Manas Joglekar, Eric Wallace, Saachi Jain, Boaz Barak, Alec Helyar, Rachel Dias, Andrea Vallone, Hongyu Ren, Jason Wei, Hyung Won Chung, Sam Toyer, Johannes Heidecke, Alex Beutel, and Amelia Glaese. Deliberative alignment: Reasoning enables safer language models, 2024. URL https://arxiv.org/abs/2412.16339.

Wei He, Yueqing Sun, Hongyan Hao, Xueyuan Hao, Zhikang Xia, Qi Gu, Chengcheng Han, Dengchang Zhao, Hui Su, Kefeng Zhang, Man Gao, Xi Su, Xiaodong Cai, Xunliang Cai, Yu Yang, and Yunke Zhao. Vitabench: Benchmarking llm agents with versatile interactive tasks in real-world applications. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics (ACL)*, 2023.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Qi Jia, Ye Shen, Xiujie Song, Kaiwei Zhang, Shibo Wang, Dun Pei, Xiangyang Zhu, and Guangtao Zhai. One battle after another: Probing LLMs' limits on multi-turn instruction following with a benchmark evolving framework. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026. URL https://aclanthology.org/2026.acl-long.433/.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

Yannis Katsis, Sara Rosenthal, Kshitij Fadnis, Chulaka Gunasekara, Young-Suk Lee, Lucian Popa, Vraj Shah, Huaiyu Zhu, Danish Contractor, and Marina Danilevsky. Mtrag: A multi-turn conversational benchmark for evaluating retrieval-augmented generation systems. *Transactions of the Association for Computational Linguistics (TACL)*, 2025.

Najoung Kim and Sebastian Schuster. Entity tracking in language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023. URL https://aclanthology.org/2023.acl-long.558/.

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 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

Philippe Laban, Alexander R. Fabbri, Caiming Xiong, and Chien-Sheng Wu. Summary of a haystack: A challenge to long-context llms and rag systems, 2024. URL https://arxiv.org/abs/2407.01370.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. Llms get lost in multi-turn conversation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. A human-inspired reading agent with gist memory of very long contexts. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics (TACL)*, 2023.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4, 2023. URL https://arxiv.org/abs/2306.02707.

Ankur P. Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. Totto: A controlled table-to-text generation dataset. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Vedant Patel. Supersede: Diagnosing and training the memory-update gap in LLM agents, 2026. URL https://arxiv.org/abs/2606.27472.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

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. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

Jihoon Tack, Philippe Laban, and Jennifer Neville. Llms get lost in evolving user intent, 2026. URL https://arxiv.org/abs/2607.20734.

Md Nayem Uddin, Kumar Shubham, Eduardo Blanco, Chitta Baral, and Gengyu Wang. From recall to forgetting: Benchmarking long-term memory for personalized agents. In *Findings of the Association for Computational Linguistics (ACL)*, 2026. URL https://aclanthology.org/2026.findings-acl.1337/.

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 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Wentao Zhu, Zhining Zhang, and Yizhou Wang. Language models represent beliefs of self and others. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

Henry Peng Zou, Chunyu Miao, Wei-Chieh Huang, Yankai Chen, Yue Zhou, Hanrong Zhang, Yaozu Wu, Liancheng Fang, Zhengyao Gu, Zhen Zhang, Kening Zheng, Fangxin Wang, Yi Nian, Shanghao Li, Wenzhe Fan, Langzhou He, Weizhi Zhang, Xue Liu, and Philip S. Yu. When users change their mind: Evaluating interruptible agents in long-horizon web navigation, 2026. URL https://arxiv.org/abs/2604.00892.

A Benchmark Construction and Evaluation Details

Table 3: IntentFlux source sets and paper strata.

Stratum	Cases	Role
General source tasks	—	code, math, SQL, data-to-text, summary, tool use
General calibration pool	627	variant/decoy stress condition
General-Test	125	cross-model, control, and harness evaluation
Interactive-Test	100	interactive transfer
Training pool	502	OPD and thinking-distillation source pool

Each source problem is decomposed into ordered atomic shards. A shard is atomic when changing it changes the final action. We construct behavior-changing variants and plausible decoys, reject pure rewordings, and enforce dependency constraints when sampling edit plans. Human annotators inspect every variant, decoy, and semantic-dependency triple. A downstream audit of General-Test recovers the annotated $G^\star$ for 95% of cases; further quality-control details appear in Section E.

Table 4: Task-native graders by task family.

Task family	Grader
code	final function against public and hidden tests
math	extracted numeric answer
database	SQL execution against Spider databases
data-to-text	sacreBLEU
summary	SummHay LLM-based grader
actions	BFCL AST check
VitaBench	domain-specific evaluator

The simulator realizes per-goal edit queues and a decoy queue, preserving within-goal order while interleaving feasible edits. Every expected GENERAL-TEST case enters reported aggregates; absent scorable outputs count as zero. The primary tested backbone for the single-model analyses is Qwen3.5-122B-A10B-FP8 served locally with vLLM. DeepSeek-V4-Flash serves as the user simulator and rubric judge throughout; complete model, serving, seed, and bootstrap details are supplied in the anonymized artifact.

B DETAILED BENCHMARK RESULTS

Table 5: Aggregate response in the joint variant/decoy difficulty calibration.

Stratum	Realized variants	Realized decoys	Mean score	CAR
Easy	0.95	0.61	0.476	0.381
Medium	1.71	1.67	0.419	0.327
Hard	2.24	2.50	0.384	0.298
Δ(easy$\rightarrow$hard)	$+1.29$	$+1.89$	-0.092	-0.083

Table 6: Case-paired clean–drift IDG by tested model and difficulty. Brackets give paired 95% confidence intervals.

Tested model	Clean CAR	Easy	Medium	Hard
Claude Opus 4.7	.592	.248 [.160,.336]	.352 [.256,.448]	.408 [.304,.512]
DeepSeek V4 Pro	.584	.216 [.128,.304]	.208 [.128,.288]	.384 [.296,.472]
Gemini 3.5 Flash	.632	.248 [.168,.336]	.312 [.232,.400]	.368 [.280,.456]
GLM 5.2	.576	.208 [.128,.288]	.272 [.192,.352]	.344 [.264,.432]
GPT-5.4	.608	.232 [.160,.312]	.344 [.256,.432]	.384 [.304,.472]
HY3	.592	.264 [.176,.352]	.288 [.200,.384]	.392 [.304,.480]
Qwen3.6 Plus	.640	.192 [.120,.264]	.328 [.248,.416]	.448 [.360,.536]
Qwen3.5-122B	.592	.304 [.224,.384]	.352 [.264,.440]	.360 [.280,.448]

Table 7: Per-family response to increasing joint variant/decoy load.

Task family	n	easy	medium	hard	Δ(easy$\rightarrow$hard)
code (LiveCodeBench)	55	.582	.418	.273	$-.309$
math (GSM8K)	103	.592	.476	.408	$-.184$
code (HumanEval)	45	.689	.622	.511	$-.178$
database (Spider)	107	.430	.327	.308	$-.121$
data-to-text (ToTTo)	120	.367	.352	.325	$-.042$
summary (SummHay)	92	.165	.169	.163	$-.003$
actions (BFCL)	105	.657	.667	.705	$+.048$
ALL	**627**	**.476**	**.419**	**.384**	$-.092$

Table 8: Raw cross-model validation on the GENERAL-TEST stress grid. Each cell is case-micro mean task-native score / full-credit CAR.

Tested model	Easy	Medium	Hard	Avg.
Claude Opus 4.7	.444/.344	.326/.240	.268/.184	.346/.256
DeepSeek V4 Pro	.462/.368	.468/.376	.277/.200	.402/.315
Gemini 3.5 Flash	.482/.384	.416/.320	.360/.264	.420/.323
GLM 5.2	.458/.368	.404/.304	.300/.232	.387/.301
GPT-5.4	.479/.376	.338/.264	.302/.224	.373/.288
HY3	.400/.328	.368/.304	.257/.200	.342/.277
Qwen3.6 Plus	.569/.448	.408/.312	.274/.192	.417/.317
Qwen3.5-122B	.288/.288	.240/.240	.232/.232	.253/.253

Table 9: Case-paired full-credit IDG by task family on GENERAL-TEST (Gemini 3.5 Flash). ToTTo and SummHay have zero IDG because neither condition attains full credit; their partial-credit changes are represented in mean score rather than CAR.

Task family	n	Easy	Medium	Hard
code (LiveCodeBench)	10	.700	.400	.900
code (HumanEval)	10	.400	.500	.500
math (GSM8K)	21	.286	.381	.429
database (Spider)	21	.381	.619	.762
data-to-text (ToTTo)	24	.000	.000	.000
summary (SummHay)	18	.000	.000	.000
actions (BFCL)	21	.286	.429	.333

C CONTROLS AND INTERACTIVE TRANSFER

The difficulty calibration uses variant/decoy budgets of 1, 3, and 5 for the easy, medium, and hard strata. These budgets jointly increase the available same-slot alternatives and eligible withdrawn decoys. They also increase mean dialogue length from 13.2 to 37.1 turns. This coupling is why the main text interprets the calibration as a joint stale-information stress condition and uses a separate control for length. The final-answer trigger does not restate $G^\star$ in any of the three strata.

We additionally construct a turn-matched no-drift control on GENERAL-TEST. It states the final requirements in the first turn and uses no-change confirmations thereafter. The control and drift conditions have identical turn counts and comparable user-message length (13.2k versus 14.6k mean characters). The control is much closer to the single-turn condition than the drift trajectories on most families, whereas GSM8K exhibits a separate premature-commitment failure. This control targets the length hypothesis: it is not designed to reproduce the obsolete competing values, because doing so would reintroduce the drift treatment. It therefore does not separately identify effects of message order, edit wording, variants, and decoys.

Table 10: Turn-matched no-drift control on GENERAL-TEST (Qwen3.5-122B, mean score).

Task family	n	single-turn	matched control	drift	control−drift
code (LiveCodeBench)	10	1.000	1.000	.600	+.400
math (GSM8K)	21	.952	.667	.667	+.000
code (HumanEval)	10	1.000	1.000	.400	+.600
database (Spider)	21	.952	.952	.571	+.381
data-to-text (ToTTo)	24	.499	.497	.219	+.278
summary (SummHay)	18	.294	.321	.188	+.134
actions (BFCL)	21	.952	.952	.667	+.286
ALL	**125**	**.778**	**.734**	**.469**	**+.265**

Table 11: VitaBench interactive-transfer results ($n = 100$ per condition, Qwen3.5-122B as the tested agent and DeepSeek-V4-Flash as the user simulator and rubric judge).

Condition	Score	CAR
clean (no drift)	.697	.190
drift (default close)	.720	.190
drift (no-restate close)	.594	.190
GT-oracle final intent	.764	.310
clean − no-restate drift	$+.103$ [$+.026, +.180$]	
default − no-restate drift	$+.125$ [$+.050, +.202$]	

The aligned no-restate condition yields a lower rubric score than clean, while the default closing restatement masks that difference. All three non-oracle conditions have the same CAR (.190), so this result is specific to partial-credit rubric evaluation in the single-model setup.

D StateForge Diagnostics and Training Details

Table 12: Component ablation under the unified re-run protocol ($n = 125$). Differences are paired 95% CIs against full StateForge.

Configuration	Mean	CAR	Δ vs full
Without gate	.393	.304	$-.035$ [$-.101, +.030$]
Generic recap	.403	.312	$-.025$ [$-.100, +.048$]
Neutral recap	.441	.344	$+.013$ [$-.060, +.087$]
StateForge full	.428	.344	—
Oracle final state	.551	.440	$+.123$ [$+.044, +.202$]

The matched recap and full-harness comparisons are not separated at this sample size. The oracle row nevertheless shows material headroom from exact state estimation within the history-preserving harness; it should not be interpreted as showing that tracking dominates the gap to the clean single-turn condition.

Table 13: Complete tracker-scale comparison on the unified re-run protocol.

Tracker	Mean	CAR	Δ vs 122B
2B	.156	.112	$-.272$ [$-.351, -.196$]
4B	.312	.240	$-.117$ [$-.199, -.036$]
9B	.399	.312	$-.029$ [$-.101, +.042$]
27B	.366	.272	$-.062$ [$-.134, +.005$]
35B	.395	.304	$-.034$ [$-.109, +.038$]
122B	.428	.344	—

For modular adaptation, the 122B base agent is frozen and paired with a 2B tracker; the reported before–after change is therefore attributable to OPD training of the tracker within that stack. These full-episode scores use the tracker-training evaluation protocol rather than the General-Test task-score protocol.

For full-agent internalization, the 35B thinking-distilled v4 model scores .461 on the drift set and .444 under avg@3 decoding; the base model scores .296. The distilled model scores .718 on the clean condition versus .745 for the base model. The v1–v4 progression, SQL-formatting constraint, and per-family results are included in the artifact.

E QUALITY-CONTROL RUBRICS

We audit randomly sampled edit-phase generations rather than full traces. The fidelity rubric checks whether an utterance realizes its scheduled Add, Replace, Delete, or valid pre-retraction decoy without leaking a future edit or contradicting an active requirement. The coherence rubric checks reference resolution, active-state consistency, and dialogue flow. Across 100 sampled utterances from the DeepSeek-V4-Flash simulator used throughout the experiments, fidelity is 96% and coherence is 97%. Residual infidelity is concentrated in redundant restatements rather than invented requirement content.