ENVSIMBENCH: A BENCHMARK FOR EVALUATING AND IMPROVING LLM-BASED ENVIRONMENT SIMULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-based environment simulators offer a flexible substrate for agent training, but their usefulness depends on whether generated feedback and state updates faithfully reflect agent actions. In practice, LLM-simulated environments suffer from hallucinations, logical inconsistencies, and silent state drift—failures that corrupt agent reward signals and compound the construction costs that the paradigm was designed to eliminate. To address this gap, we propose EnvSimBench with four contributions: (1) We provide the first formal definition and operationalization of **Environment Simulation Ability (EnvSim Ability)** as a quantifiable research objective. (2) We construct EnvSimBench, a rigorous benchmark covering 400 samples across 167 diverse environments, equipped with verifiable labels and fine-grained difficulty stratification along three axes. (3) Evaluations of seven frontier language models reveal a pronounced decline in Config Match on state-changing operations relative to state-preserving operations, which we term the state-change cliff. This finding identifies a limitation in predicting the state updates required by the evaluated environments. (4) We design a constraint-driven simulation pipeline that substantially reduces hallucination, boosts environment synthesis yield by 6.8%, and cuts costs by over 90%. Overall, EnvSimBench serves as both a diagnostic framework and a practical optimization path for reliable LLM-based environment simulation, establishing a solid foundation for scalable agent training. Code and data are available at `https://anonymous.4open.science/r/EnvSimBench-D3D2`.

1 INTRODUCTION

Large language models (LLMs) are increasingly required to serve as agents across diverse real-world scenarios (Luo et al., 2025; Yao et al., 2025; Anonymous, 2026). Recent research efforts (Patil et al., 2025; Yao et al., 2025; Lu et al., 2025) have constructed stateful, tool-interactive sandbox environments, delivering improved controllability and stability.

Training autonomous agents at scale requires interactive environments that feature rich diversity, high simulation fidelity, and low deployment costs (Huang et al., 2025; DeepSeek-AI et al., 2025; Froger et al., 2025). However, manually designed environments suffer from prohibitive construction costs, poor extensibility, and inherent limitations in scenario coverage, making them ill-suited for large-scale agent training. A compelling recent direction addresses this challenge by replacing manually crafted, executable environments(Jimenez et al., 2024a; Zhou et al., 2024a; Shridhar et al., 2021) with LLM-simulated counterparts, where a language model generates the feedback for each agent action (Li et al., 2025; Song et al., 2026). This paradigm dramatically lowers the barrier to environment creation and promises seamless scaling across domains. However, it rests on a foundational assumption that has received surprisingly little scrutiny: LLMs can accurately simulate environmental feedback with sufficient fidelity to be trusted as training substrates.

If this assumption fails, agents trained in hallucinated environments optimize against corrupted reward signals, and synthesis pipelines incur compounding failure costs that erode the very cost advantage the paradigm promises, but simulation fidelity has never been systematically characterized.

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.

In practice, when models attempt to act as simulators, their performance is undermined by specific vulnerabilities. Three concrete failure modes underlie this difficulty: hallucination (Ji et al., 2023; Patil et al., 2024; Zhang et al., 2025), whereby a model fabricates plausible but incorrect state transitions; logical inconsistency (Ruan et al., 2024; Elazar et al., 2021), whereby inter-field constraints within a single response are violated; and state drift (Lu et al., 2025; Prabhakar et al., 2025a; Yao et al., 2025), whereby the absence of persistent memory causes earlier state changes to be silently lost. These failures are not incidental engineering deficiencies but reflect a fundamental tension between the generative nature of language models and the deterministic, compositional logic of executable environments. Without a framework to precisely measure these failures, they remain invisible to practitioners and unaddressable by researchers.

To address this gap and establish a rigorous foundation for studying simulation fidelity, we introduce the concept of Environment Simulation Ability (EnvSim Ability) to denote a model's capacity to accurately predict the state transition and observational feedback induced by an agent action, given the current environment state and the action's implementation logic. Without a precise, operationalizable definition, neither systematic evaluation nor principled improvement of this capability is possible. Executable environments provide reference transitions against which simulator predictions can be checked. Their role in EnvSimBench is to make transition errors observable and reproducible. This evaluation quantifies how reliably a model reproduces specified environment behavior; transfer to environments with implicit rules or unavailable execution references requires separate validation.

To address three failure modes and make EnvSim Ability concretely measurable, we construct EnvSimBench. EnvSimBench makes the recorded configuration explicit for independently verifiable transition prediction. In a history-conditioned simulation setting, the current state is not directly supplied, giving a partially observable formulation (Oliehoek et al., 2016) in which the simulator must infer the state from preceding interactions. EnvSimBench separates transition prediction from history-based state estimation. We provide the recorded pre-execution configuration and tool implementation for each sample. This explicit-state protocol isolates errors in predicting the next observation and configuration from errors accumulated while reconstructing earlier states. It enables a controlled diagnosis of transition fidelity, a component of reliable multi-turn simulation. This reframing is the architectural foundation of EnvSimBench.

EnvSimBench makes four contributions, linked by a deliberate logical chain from conceptualization to diagnosis to remedy:

(I) Formalizing EnvSim Ability. We provide the first formal definition and operationalization of EnvSim Ability as a quantifiable research objective, establishing it as a distinct capability from related skills and providing a conceptual foundation for rigorous evaluation and improvement.

(II) A rigorous benchmark. Grounded in the POMDP-to-MDP reframing, we construct a benchmark of 400 samples across 167 diverse tool-interactive environments, with verifiable programmatic labels and fine-grained difficulty stratification along three stratification dimensions: action outcome, state-change complexity, and argument cardinality.

(III) A diagnostic finding: the state-change cliff. Systematic evaluation reveals a universal failure pattern: all models achieve near-perfect accuracy on state-preserving operations yet collapse catastrophically when multiple state variables must update simultaneously. We term this threshold the *state-change cliff*. This gap is orthogonal to model scale and general reasoning ability. More critically, models that generate superficially correct feedback can simultaneously produce incorrect state transitions, silently corrupting training signals with no observable divergence for the agent.

(IV) A constraint-driven remedy. Guided by these findings, we propose a constraint-driven simulation paradigm that makes environment schemas and transition logic explicit at each step. A simulation model trained under this paradigm surpasses all evaluated frontier LLMs on configuration match, boosts synthesis yield by 6.8%, and cuts costs by over 90%. This demonstrates that targeted specialization is a cost-efficient path to reliable environment simulation.

2 Related Work

LLM environment simulation and synthesis. Simia (Li et al., 2025) demonstrated that reasoning models can generate plausible tool feedback for agent training, establishing LLM-simulated tool-

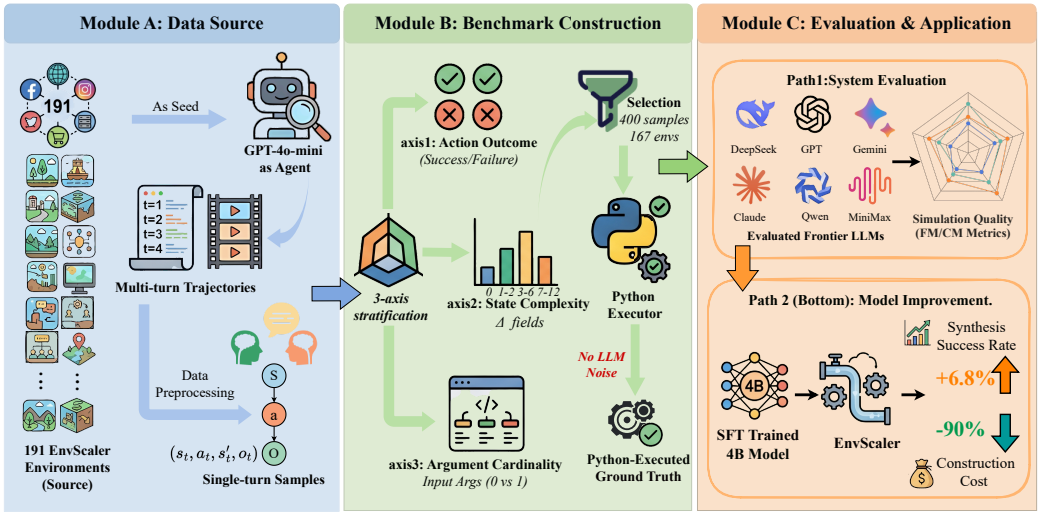


Figure 1: **Overview of EnvSimBench. Module A:** EnvScaler (Song et al., 2026) environments serve as seed data; a GPT-4o-mini agent collects multi-turn execution trajectories, preprocessed into self-contained single-turn state prediction samples (s_t, a_t, s'_t, o_t). Each step is independently verifiable against a programmatic label, decoupling simulation fidelity from state tracking and making EnvSim Ability objectively measurable. **Module B:** Samples undergo three-axis stratification (action outcome, state-change complexity, argument cardinality) and executor-based labeling, yielding 400 benchmark samples across 167 environments. The three axes allow failures to be precisely localized rather than collapsed into a single score. **Module C:** *Path 1* evaluates seven frontier LLMs under identical conditions. *Path 2* trains a specialized 4B simulation model that, when integrated into EnvScaler, improves synthesis yield by 6.8% while cutting costs by over 90%.

interactive environments as a practical substrate for scalable agent training. EnvScaler (Song et al., 2026) extended this direction by demonstrating a fully automated synthesis of tool-interactive environments, substantially reducing construction costs. However, neither work formally defines simulation fidelity as a measurable capability, provides cross-model comparisons of simulation quality, or offers principled evaluation metrics. EnvSimBench addresses all three gaps simultaneously. Relatedly, interactive fiction environments (Hausknecht et al., 2020) and code-execution benchmarks (Jimenez et al., 2024a) use executable environments to evaluate agent behavior; our work inverts this relationship, treating the simulator itself rather than the agent as the object of evaluation.

Agent benchmarks and tool use. Building on a long line of prior work on language agents (Yao et al., 2022; Zhou et al., 2024b; Jimenez et al., 2024b; Liu et al., 2024; Ruan et al., 2024; Deng et al., 2023; Zeng et al., 2024), LLM tool utilization (Patil et al., 2025; Qin et al., 2024; Huang et al., 2024), and task-oriented dialogue systems τ-bench (Yao et al., 2025) and τ^2-bench (Barres et al., 2026) established multi-turn, tool-using evaluation paradigms and introduced the MDP/POMDP distinction that we adopt as the architectural foundation of our diagnostic framework. ToolSandbox (Lu et al., 2025) evaluates stateful tool use with fine-grained state-transition feedback, providing a complementary perspective in which the agent is evaluated against a fixed executable environment; our work evaluates the simulator rather than the agent, and asks whether a language model can *replace* such executors reliably. APIGen-MT (Prabhakar et al., 2025b) generates multi-turn training data via LLM simulation; our pipeline shares structural similarities but replaces LLM-generated labels with programmatically verified ground truth, directly addressing the circular validation problem that motivates our work.

Hallucination and simulation fidelity. Hallucination in language model generation has been extensively studied in open-ended settings (Ji et al., 2023; Zhang et al., 2025). Gorilla (Patil et al., 2024) demonstrated that language models frequently produce incorrect tool arguments even when the correct API schema is provided. This finding also highlights the importance of our research. Prior work on language model program execution (Lu et al., 2025) shows that prediction accuracy degrades as the number of interdependent state updates increases, underscoring the cumulative effect

of state updates across multiple execution rounds. EnvSimBench prevented the cascading dependency identified in this article by using the POMDP-to-MDP pattern.

3 Problem Formulation

Tool interactive environments. Following EnvScaler, we model a tool-interactive environment as $\mathcal{E} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O})$, where $\mathcal{S}$ is the configuration space and $\mathcal{A}$ is the action space. For fixed runtime inputs ξ, the transition and observation functions are deterministic:

$$s'_t = \mathcal{T}(s_t, a_t; \xi_t), \qquad o_t = \mathcal{O}(s_t, a_t; \xi_t) \in \Sigma^*.$$

Here ξ_t denotes runtime inputs such as clock values used during execution. For example, a mutual fund management system stores securities, portfolio holdings, and transaction logs, while a mobile authentication system stores OTP records, session states, and verification timestamps. Reference outputs (o_t, s'_t) are obtained by executing the tool implementation. Runtime inputs need not be recoverable from the stored configuration; their implications for exact matching are discussed in Appendix D.3.

State prediction task. Given the pre-execution configuration s_t, a tool call a_t, and its implementation context $c_t = \texttt{code}(a_t)$, the model predicts a feedback string $\hat{o}_t$ and a list of state-change operations $\widehat{\Delta}_t$. Each operation specifies an addition, modification, or deletion at a configuration path. Let $\Delta_t = \mathrm{Diff}(s_t, s'_t)$ denote the reference changes recorded by the benchmark's configuration-difference procedure, and let $k_t = |\Delta_t|$ denote the state-change count. The predicted list $\widehat{\Delta}_t$ may contain a different number of operations. The predicted after-state is reconstructed as

$$(\hat{o}_t, \widehat{\Delta}_t) = f_\theta(s_t, a_t, c_t), \qquad \hat{s}'_t = \mathrm{Apply}(s_t, \widehat{\Delta}_t). \tag{1}$$

Feedback Match (FM) tests exact equality between $\hat{o}_t$ and o_t, while Config Match (CM) tests equality between the reconstructed $\hat{s}'_t$ and s'_t. Both metrics use executor-produced reference outputs.

POMDP vs. MDP formulation. In history-conditioned simulation, the current configuration is not supplied explicitly. Let $h_t = (a_1, o_1, \ldots, a_{t-1}, o_{t-1})$ denote the preceding interaction history. A belief state is a distribution $b_t(s) = \Pr(S_t = s \mid h_t)$ over possible current configurations. A history-conditioned simulator predicts the next output from (h_t, a_t), combining state reconstruction, transition prediction, and observation generation. When predicted outputs are reused in later interactions, earlier errors can affect subsequent predictions.

Constraint-driven MDP formulation. EnvSimBench supplies the recorded pre-execution configuration for every prediction, together with the current tool call and its implementation context, as in Equation 1. Each sample is scored against its own executor-produced reference output. Consequently, an incorrect prediction on one sample is not propagated as the input state of another. This protocol isolates single-step transition fidelity from history reconstruction while retaining errors in code interpretation, feedback generation, and state updates as measurable outcomes.

4 EnvSimBench and Constraint-Driven Simulation Paradigm

4.1 Benchmark Construction

From trajectories to single-turn samples. The constraint-driven formulation (Section 3) motivates self-contained samples: each supplies (s_t, a_t, c_t) and requires $(\hat{o}_t, \widehat{\Delta}_t)$, from which the after-state is reconstructed. Each prediction is evaluated against the recorded reference transition, without using another prediction as its input state.

As illustrated in Figure 4, we collect execution trajectories $\tau = ((a_t, o_t, s_t, s'_t))_{t=1}^{T}$ from 191 tool-interactive seed environments. We discard the initialization-only record because it does not contain a tool call and a corresponding before/after configuration pair. The configuration produced by initialization serves as the before-state of the first retained tool call. Each retained sample has an executor-produced reference output (o_t, s'_t) and is evaluated separately.

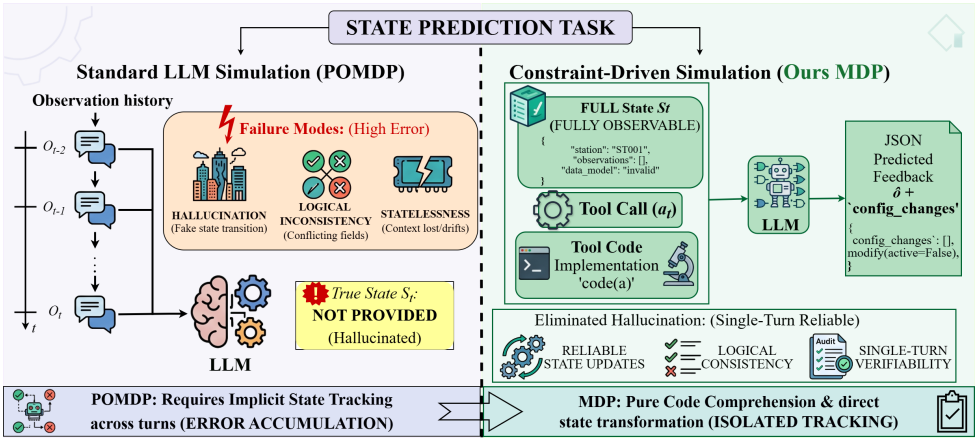


Figure 2: **POMDP vs. MDP formulation.** *Left*: Standard simulation; the LLM infers state from conversation history, causing state drift and hallucination. *Right*: Constraint-driven simulation; the full before-config s_t, tool call a_t, and implementation `code(a)` are provided explicitly, making each step independently verifiable.

Three-axis difficulty stratification. To enable fine-grained diagnosis rather than a single aggregate score, we stratify the sample pool along three dimensions, each targeting a source of simulation difficulty. Figure 1 provides an overview of EnvSimBench, and a detailed construction pipeline is described in Appendix B.

Axis 1 (Action Outcome). Samples are first partitioned by whether the action a_t succeeds or fails under $\mathcal{T}$. The *Failure* group ($|\Delta(s_t, s'_t)| = 0$ with $\mathcal{O}$ returning an error signal) tests whether a model correctly withholds state changes when the transition guard is not satisfied.

Axis 2 (State-Change Complexity). Within the success group, samples are stratified by $|\Delta(s_t, s'_t)|$, the number of changes recorded by the configuration-difference procedure, into four tiers: *No-Change* ($|\Delta| = 0$, 80 samples), *Simple* ($|\Delta| \in \{1, 2\}$, 50 samples), *Medium* ($|\Delta| \in \{3, \ldots, 6\}$, 200 samples), and *Difficult* ($|\Delta| \in \{7, \ldots, 12\}$, 50 samples). The largest allocation is assigned to Medium samples to support analysis of multi-field updates. State-change count is a stratification variable; update structure can also affect prediction difficulty (Appendix A.4).

Axis 3 (Input Argument Cardinality). For Failure and No-Change samples, a third axis further partitions by the number of arguments supplied to a_t (zero vs. one or more). This controls for a qualitatively simpler sub-case: when no arguments are provided, the model need not reason about argument-dependent branching in the transition logic, making correct prediction structurally easier regardless of state-change complexity.

Diversity rule. Within each stratum, samples are selected to maximize the number of distinct environments $|\mathcal{E}_{\text{sub}}|$ represented. This *diversity rule* prevents the benchmark from concentrating in a small number of high-trajectory environments and ensures that evaluation generalizes across the full breadth of domain types present in the source collection. The final benchmark comprises 400 samples drawn from 167 distinct environments (Table 1). All ground-truth labels are produced by the deterministic external executor, making the benchmark entirely LLM-free in its construction.

Table 1: Benchmark sample distribution.

Group	Subgroup	Samples	Constraint
Failure	$\mathcal{O}$ returns error, $\lvert\Delta\rvert = 0$	20	
No-Change	$\lvert\Delta\rvert = 0$, action succeeds	80	40 per argument cardinality
Simple	$\lvert\Delta\rvert \in \{1, 2\}$	50	25 per $\lvert\Delta\rvert$ value
Medium	$\lvert\Delta\rvert \in \{3, \ldots, 6\}$	200	50 per $\lvert\Delta\rvert$ value
Difficult	$\lvert\Delta\rvert \in \{7, \ldots, 12\}$	50	Distributed across $\lvert\Delta\rvert$

Evaluation metrics. For each sample i, we evaluate feedback and configuration correctness separately:

$$\mathrm{FM}_i = \mathbf{1}[\hat{o}_i = o_i], \qquad \mathrm{CM}_i = \mathbf{1}[\mathrm{Apply}(s_i, \widehat{\Delta}_i) = s'_i]. \tag{2}$$

FM requires exact agreement with the reference feedback string. CM requires the reconstructed configuration to match the reference after-state. The two metrics measure complementary aspects of a simulated transition: matching the feedback does not guarantee a correct state update, and a correct state update does not guarantee matching feedback. For a group G, we report the sample mean of each binary metric.

4.2 Constraint-Driven Simulation Paradigm

MDP prompt structure. Each evaluation prompt supplies (s_t, a_t, c_t) in a structured format. The response contains $\hat{o}_t$ and a list $\widehat{\Delta}_t = (\delta_{t,1}, \ldots, \delta_{t,m_t})$ of typed state-change operations. Each operation identifies its type, target path, and a value when required. The evaluator reconstructs $\hat{s}'_t = \mathrm{Apply}(s_t, \widehat{\Delta}_t)$ and compares it with s'_t to compute CM.

Reasoning enhancement. Training data is augmented with explicit intermediate derivations, formatted as structured reasoning traces that walk through the transition logic of $\texttt{code}(a_t)$ step by step before producing $(\hat{o}_t, \widehat{\Delta})$. These traces teach the supervised model to internalize code-execution as a sequential inference process rather than a pattern-matching lookup.

5 Experimental Results and Analysis

5.1 Evaluation of Frontier LLMs

Setup. We evaluate seven frontier language models: DeepSeek-V3.2, Qwen3.5-397B-A17B, GPT-5.4, Gemini-3.1-Pro-Preview, Claude-Sonnet-4.6, MiniMax-M2.7, and GLM-5, all in non-thinking mode via their respective inference APIs, with identical prompts instantiating the MDP triple $(s_t, a_t, \texttt{code}(a_t))$. Ground-truth labels are produced entirely by the deterministic external executor. Full evaluation details are provided in Appendix G. Supplementary reasoning-mode results are reported in Appendix K.

Table 2: Results on Failure and No-Change samples. **Bold**: column-wise maximum; red: column-wise minimum.

	Fail		No-Chg(0)		No-Chg(1)		Avg	
Model	FM	CM	FM	CM	FM	CM	FM	CM
DeepSeek-V3.2	85%	100%	**85%**	**100%**	**82.5%**	100%	**84%**	**100%**
Qwen3.5-397B-A17B	90%	100%	**85%**	**100%**	77.5%	100%	83%	**100%**
GPT-5.4	90%	100%	65%	**100%**	67.5%	100%	71%	**100%**
Gemini-3.1-Pro-Preview	**95%**	100%	77.5%	**100%**	70%	100%	78%	**100%**
Claude-Sonnet-4.6	15%	100%	80%	97.5%	77.5%	100%	66%	99%
MiniMax-M2.7	80%	100%	67.5%	97.5%	62.5%	100%	68%	99%
GLM-5	90%	100%	77.5%	**100%**	70%	100%	77%	**100%**

Failure and No-Change groups. Table 2 shows that CM is near-perfect (97–100%) across all state-preserving operations, confirming that every model correctly withholds predicted state changes when none are warranted. FM exhibits wider variation. Notably, Claude-Sonnet-4.6 achieves only 15% FM on the Failure group despite 100% CM. This divergence reveals a systematic output-format mismatch: the model omits the enclosing structured response schema and returns only the inner message string. Concretely, whereas the ground-truth observation o_t takes the form of a key-value structured response $\langle \textit{status} : \text{false}, \textit{msg} : \varepsilon \rangle$, the model outputs only ε directly. MiniMax-M2.7 exhibits the same unwrapping behavior on success-type responses. Because this mismatch is an artifact of a formatting convention rather than a reasoning failure, CM remains unaffected. We adopt CM as the primary cross-model reasoning metric throughout.

Table 3: Averaged results on state-changing groups. CM degrades sharply with $|\Delta|$. **Bold**: column-wise maximum; red: column-wise minimum.

	Simple (avg)		Medium (avg)		Difficult (avg)	
Model	FM	CM	FM	CM	FM	CM
DeepSeek-V3.2	80%	22%	76.5%	8.5%	26%	4%
Qwen3.5-397B-A17B	84%	46%	65.0%	17.0%	42%	24%
GPT-5.4	80%	40%	75.0%	**17.5%**	**74%**	26%
Gemini-3.1-Pro-Preview	**96%**	48%	68.0%	**17.5%**	68%	18%
Claude-Sonnet-4.6	10%	40%	12.5%	12.0%	12%	16%
MiniMax-M2.7	32%	**50%**	22.0%	16.0%	8%	22%
GLM-5	94%	44%	**81.5%**	14.0%	70%	**28%**

State-changing groups (Simple / Medium / Difficult). Table 3 summarizes CM and FM across groups requiring actual state updates. CM degrades sharply with $|\Delta|$: most models fall below 20% on Medium and approach zero on Difficult, while FM remains substantially higher throughout. This persistent FM–CM gap reveals that models can generate surface-plausible response strings for an action without faithfully tracing the induced state transition $\mathcal{T}(s_t, a_t)$. Detailed per-$|\Delta|$ breakdowns appear in Appendix A.

Table 4: Overall results. CM is robust to output-format variation; FM is depressed for models with format-unwrapping behavior (Claude-Sonnet-4.6, MiniMax-M2.7). **Bold**: column-wise maximum; red: column-wise minimum.

	Fail+No-Chg		State-Change		Overall	
Model	FM	CM	FM	CM	FM	CM
DeepSeek-V3.2	**84%**	**100%**	68.7%	10.0%	72.5%	32.5%
Qwen3.5-397B-A17B	83%	**100%**	64.3%	**23.0%**	69.0%	**42.3%**
GPT-5.4	71%	**100%**	75.7%	22.7%	74.5%	42.0%
Gemini-3.1-Pro-Preview	78%	**100%**	72.7%	22.7%	74.0%	42.0%
Claude-Sonnet-4.6	66%	99%	12.0%	17.3%	25.5%	37.8%
MiniMax-M2.7	68%	99%	21.3%	22.7%	33.0%	41.8%
GLM-5	77%	**100%**	**81.7%**	21.3%	**80.5%**	41.0%

Finding 1: Substantial capability gaps exist across frontier LLMs. Overall CM ranges from 32.5% (DeepSeek-V3.2) to 42.3% (Qwen3.5-397B-A17B), a nearly 10 pp spread that originates *entirely* from state-changing samples, since all models achieve $\geq$99% CM on state-preserving operations. On state-changing samples, DeepSeek-V3.2 achieves only 10.0% CM while the best frontier models reach 22–23%. This gap is not explained by model scale: DeepSeek-V3.2 scores competitively on standard reasoning benchmarks yet exhibits the largest CM deficit here, suggesting that *EnvSim Ability* is a capability orthogonal to general reasoning proficiency (strong reasoning does not imply strong simulation).

Finding 2 (State-change cliff): All models collapse in CM when $|\Delta| \geq 3$. Per-$|\Delta|$ results in Appendix A reveal a universal threshold. At $|\Delta| = 1$, CM spans 36–72%; by $|\Delta| = 5$, every model falls to $\leq$4%. Across all samples with $|\Delta| \geq 5$, CM rates converge near zero: DeepSeek-V3.2 2.7%, Claude-Sonnet-4.6 6.0%, Gemini-3.1-Pro-Preview 9.3%, MiniMax-M2.7 10.7%, GPT-5.4 12.0%, GLM-5 11.3%. This threshold marks a *qualitative* transition rather than a gradual decline. Below it ($|\Delta| \leq 4$), models at least partially track state changes: getting some fields right while missing others. Above it ($|\Delta| \geq 5$), predictions collapse: many models output $\widehat{\Delta} = \emptyset$, reporting no state change at all, even though the corresponding real action actually needs to modify a large number of state fields. A partial CM recovery visible at $|\Delta| = 7$ for several models is not evidence of improved capability but reflects a structural artifact: high-$|\Delta|$ samples that models answer correctly are dominated by *bulk-uniform* operations whose changes instantiate a single repeating template, a qualitatively easier task than the heterogeneous multi-field updates that cause the cliff (see Appendix A.4).

Finding 3: FM and CM capture complementary error types. *Format mismatch* can cause FM failure despite a correct state prediction. For example, Claude-Sonnet-4.6 has 15% FM and 100%

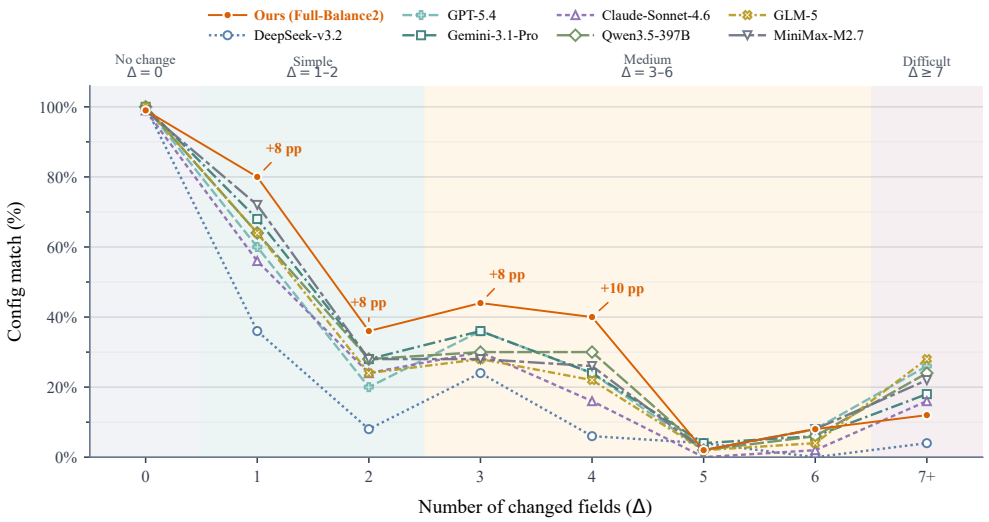


Figure 3: **Config Match by state-change count on the original benchmark.** Full-Balance2 (orange) is compared with the seven non-thinking frontier baselines. At $|\Delta| \in \{1, 2, 3, 4\}$, Full-Balance2 exceeds the strongest frontier baseline at each level by up to 10 pp. The final point pools samples with $|\Delta| = 7$–12; the trend is non-monotonic.

CM on Failure samples, but its FM is 80% and 77.5% on the two No-Change subgroups. Format sensitivity therefore does not imply uniformly low FM across state-preserving operations.

State-tracking failure is universal and $|\Delta|$-dependent, manifesting in three sub-patterns: **(i)** models systematically neglect auxiliary state updates induced as side effects of the primary action; **(ii)** fields whose values are determined only at execution time (such as clock-generated timestamps) cannot be correctly predicted without actually running the transition logic; and **(iii)**, most consequentially, a model may produce $\hat{o}_t = o_t$ while simultaneously computing $\hat{s}'_t \neq s'_t$, giving the agent no divergence signal whatsoever. On $|\Delta| \geq 3$ samples where CM fails, 50–64% of failures also have FM $= 1$ for DeepSeek-V3.2, GLM-5, GPT-5.4, and Gemini-3.1-Pro-Preview. This means the environment state is silently corrupted while the agent's reward signal remains intact. Quantitative breakdowns and cross-model case studies for all three sub-patterns are provided in Appendix D.

5.2 Evaluation of Small Models: Before and After SFT

Setup. We fine-tune Qwen3-4B-Base via full-parameter SFT on 2×A800 (80 GB) GPUs, evaluating on an `env_id` holdout set that excludes all 167 benchmark environments from training. A pre-SFT scaling baseline confirms that specialization is required beyond scale alone (Appendix I). Full hyperparameter details and training method comparisons are provided in Appendix G. Full-parameter SFT substantially improves FM (27% → 72.8%, 2.7×) and corrects output-schema alignment; the state-change cliff persists at the 5K-sample scale, confirming it as a fundamental capability gap. Detailed per-group results and a comparison of training strategies are reported in Appendix J.

Finding 4: Reasoning-augmented traces do not consistently improve CM at 5K samples. As shown in Table 19, full SFT with structured reasoning traces achieves 30.3% CM, compared with 35.0% without structured reasoning, a 4.7 pp regression. Traces hurt on Simple samples (20% vs. 34%), likely adding distributional noise where transition logic is short, while providing a modest gain on Difficult samples (10% vs. 6%). We speculate that consistent benefits require either substantially more data or difficulty-conditioned routing that applies traces selectively to complex samples.

Finding 5 (Data composition): Composition governs generalization more strongly than volume. We train **Balance2** by mirroring the empirical difficulty distribution of $|\Delta|$ across source environments: 1,000 failure + 1,000 no-change + 2,000 simple-change + 2,230 complex-change samples (6,230 total). This balanced coverage prevents the model from developing pathological biases toward any single difficulty tier. A full composition ablation comparing alternative data mixtures is provided in

Table 17 (Appendix H). Balance2 achieves 45.3% overall CM and 79.5% FM, **surpassing all frontier LLMs on CM** (+3.0 pp over Qwen3.5-397B-A17B) and nearly matching the best FM (79.5% vs. 80.5% for GLM-5). The advantage is concentrated in the deployable regime: Balance2 leads by up to +10 pp at $|\Delta| \in \{1, 2, 3, 4\}$, while all approaches converge near zero at $|\Delta| \geq 5$. Figure 3 visualizes Balance2's per-$|\Delta|$ advantage over all frontier models across the Simple and Medium tiers.

Downstream Validation: EnvScaler Pipeline. We integrate Full-Balance2 into the EnvScaler synthesis pipeline in place of its large-model ensemble. Synthesized environments are retained only if their pass accuracy meets a quality threshold of 0.85. The original pipeline yields 191 passing environments; Full-Balance2 yields 204, a 6.8% improvement in synthesis yield, with approximately $59\times$ lower parameter count. This demonstrates that targeted specialization at the 4B scale is Pareto-superior to frontier-model pipelines on both cost and synthesis quality.

6 Discussion and Conclusion

Why is environment simulation hard? Our results distinguish feedback-format errors from state-transition errors. *Format mismatch* generates structurally different feedback strings that fail exact matching despite correct semantics, motivating semantic metrics as future complements to FM. *State tracking failure* affects all models: predicting which nested fields change and to what values requires faithful execution of the tool's logic, a capability closer to program execution than to language modeling. Our SFT approach partially addresses this, but cases with $|\Delta| \geq 5$ remain challenging.

Cost-quality trade-off. Full-Balance2 surpasses all frontier LLMs on CM (45.3% vs. 42.3%) and nearly matches the best FM (79.5% vs. 80.5%), while running at dramatically lower cost, both on the benchmark and in the downstream synthesis pipeline. This demonstrates that EnvSimBench enables principled cost-quality comparison and targeted model improvement.

Limitations. (1) Benchmark sample size and statistical reliability. Fine-grained per-$|\Delta|$ analysis in the Difficult group rests on strata as small as $n = 4$–6, where a single sample shifts CM by 17–25 pp. The Failure group ($n = 20$) shows the same fragility. Deterministic ground-truth labels do not eliminate selection-induced sampling variance, and bootstrap error bounds should accompany all quantitative findings in future work. **(2) Circular downstream evaluation.** The synthesis quality filter overlaps in distribution with SFT training data, introducing circular validation. Scarcity of high-$|\Delta|$ training samples further compounds this; targeted data augmentation for $|\Delta| \geq 5$ remains the primary open challenge.

Conclusion. We introduced EnvSimBench, a benchmark that reframes LLM environment simulation as a fully observable state prediction task, enabling objective, LLM-free evaluation across 400 samples and 167 diverse tool-interactive environments. Our central finding, the state-change cliff, is a pronounced but non-monotonic decline in configuration accuracy on state-changing operations. The constraint-driven formulation makes the recorded state and transition logic explicit, and targeted fine-tuning produces a 4B model that surpasses the evaluated non-thinking frontier baselines on Config Match. We release EnvSimBench as a public diagnostic framework and hope it grounds future research on reliable environment simulation, a prerequisite for trustworthy scalable agent training.

AI Use Statement

Generative AI assisted with prose revision, result-table formatting, visualization code, and consistency checks against the supplied manuscript, implementation, figures, and experiment records.

Ethics Statement

This research adheres to the ICLR Code of Ethics. This work introduces EnvSimBench, a benchmark for evaluating environment simulation capabilities of large language models, and proposes a constraint-driven simulation paradigm to improve simulation reliability. Our research does not involve human subjects, animal testing, or the collection or use of personally identifiable information. All experiments are performed using publicly accessible LLM inference APIs and synthetically constructed tool-interactive environments generated by the EnvScaler pipeline. No sensitive or private data is included in the benchmark or training data.

Reproducibility Statement

We have made substantial efforts to ensure the reproducibility of all results presented in this paper. The complete benchmark construction methodology, evaluation protocols, and metric definitions are described in detail throughout the main text and in Appendix B (Benchmark Construction Pipeline) and Appendix G (Experimental Setup Details). This includes the three-axis difficulty stratification strategy, the diversity rule for sample selection, and the formal specifications of Feedback Match and Config Match. All model identifiers, API endpoints, training hyperparameters, and inference configurations are explicitly documented in Appendix G and Appendix H, allowing exact replication of both frontier model evaluations and SFT training experiments. The full EnvSimBench benchmark code, dataset, and evaluation scripts will be publicly released at `https://anonymous.4open.science/r/EnvSimBench-D3D2`. The release includes all 400 benchmark samples with Python-executed ground truth labels, the constraint-driven simulation pipeline, and all scripts needed to reproduce the figures and quantitative results. Supplementary results, ablation studies, and expanded analyses provided in the appendices can all be reproduced using the released code and data.

References

Anonymous. Userbench: An interactive gym environment for user-centric agents. In *Submitted to ACL Rolling Review - May 2026*, 2026. URL `https://openreview.net/forum?id=nb8n0icMaj`. under review.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik R Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. In *Forty-third International Conference on Machine Learning*, 2026. URL `https://openreview.net/forum?id=OC2z7iSQKa`.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexin Huang, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Liang Zhao, Liangsheng Yin, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M. S. Di, M. Y Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li, Xu Chen, Xuecheng Su, Xuehai Pan, Xuheng Lin, Xuwei Fu, Y. Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao, Bei Feng, Hui Li, J. L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R. J. Chen, R. L. Jin, S. S. Li, Shuang Zhou, Tianyu Sun, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y. X. Zhu, Yanping Huang, Yaohui

Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T. Wang, W. L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, and Zihua Qu. Deepseek-v3.2: Pushing the frontier of open large language models, 2025. URL https://arxiv.org/abs/2512.02556.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 28091–28114. Curran Associates, Inc., 2023. doi: 10.52202/075280-1220. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/5950bf290a1570ea401bf98882128160-Paper-Datasets_and_Benchmarks.pdf.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031, 2021. doi: 10.1162/tacl_a_00410. URL https://aclanthology.org/2021.tacl-1.60/.

Romain Froger, Pierre Andrews, Matteo Bettini, Amar Budhiraja, Ricardo Silveira Cabral, Virginie Do, Emilien Garreau, Jean-Baptiste Gaya, Hugo Laurençon, Maxime Lecanu, Kunal Malkan, Dheeraj Mekala, Pierre Ménard, Gerard Moreno-Torres Bertran, Ulyana Piterbarg, Mikhail Plekhanov, Mathieu Rita, Andrey Rusakov, Vladislav Vorotilov, Mengjue Wang, Ian Yu, Amine Benhalloum, Grégoire Mialon, and Thomas Scialom. Are: Scaling up agent environments and evaluations, 2025. URL https://arxiv.org/abs/2509.17158.

Matthew J. Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. Interactive Fiction Games: A Colossal Adventure. In *AAAI Conference on Artificial Intelligence*, pp. 7903–7910, 2020. doi: 10.1609/AAAI.V34I05.6297. URL https://mlanthology.org/aaai/2020/hausknecht2020aaai-interactive/.

Yuchen Huang, Sijia Li, Zhiyuan Fan, Minghao LIU, Wei Liu, and Yi R. Fung. Scaling environments for LLM agents: Fundamentals, approaches, and future directions. In *Workshop on Scaling Environments for Agents*, 2025. URL https://openreview.net/forum?id=9axZcDTiJm.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Gong, and Lichao Sun. Metatool benchmark for large language models: Deciding whether to use tools and which to use. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 42978–43007, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/bc12914d66b41b6bfc2d3a5decdb498b-Paper-Conference.pdf.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12), March 2023. ISSN 0360-0300. doi: 10.1145/3571730. URL https://doi.org/10.1145/3571730.

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 B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/edac78c3e300629acfe6cbe9ca88fb84-Paper-Conference.pdf.

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 B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/edac78c3e300629acfe6cbe9ca88fb84-Paper-Conference.pdf.

Yuetai Li, Huseyin A Inan, Xiang Yue, Wei-Ning Chen, Lukas Wutschitz, Janardhan Kulkarni, Radha Poovendran, Robert Sim, and Saravan Rajmohan. Simulating environments with reasoning models for agent training, 2025. URL https://arxiv.org/abs/2511.01824.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/e9df36b21ff4ee211a8b71ee8b7e9f57-Paper-Conference.pdf.

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 Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, Rongcheng Tu, Xiao Luo, Wei Ju, Zhiping Xiao, Yifan Wang, Meng Xiao, Chenwu Liu, Jingyang Yuan, Shichang Zhang, Yiqiao Jin, Fan Zhang, Xian Wu, Hanqing Zhao, Dacheng Tao, Philip S. Yu, and Ming Zhang. Large language model agent: A survey on methodology, applications and challenges. *CoRR*, abs/2503.21460, March 2025. URL https://doi.org/10.48550/arXiv.2503.21460.

Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 126544–126565. Curran Associates, Inc., 2024. doi: 10.52202/079017-4020. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/e4c61f578ff07830f5c37378dd3ecb0d-Paper-Conference.pdf.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, 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 *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2GmDdhBdDk.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Manoj Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, Shelby Heinecke, Weiran Yao, Huan Wang, Silvio Savarese, and Caiming Xiong. Apigen-mt: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025a. doi: 10.52202/085713-2184. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/5e3661f7fe4c8ac5652d62eb3d3c96ea-Paper-Datasets_and_Benchmarks_Track.pdf.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Manoj Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, Shelby Heinecke, Weiran Yao, Huan Wang, Silvio Savarese, and Caiming Xiong. Apigen-mt: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025b. doi: 10.52202/085713-2184. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/5e3661f7fe4c8ac5652d62eb3d3c96ea-Paper-Datasets_and_Benchmarks_Track.pdf.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 9695–9717, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/28e50ee5b72e90b50e7196fde8ea260e-Paper-Conference.pdf.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16, 2020. doi: 10.1109/SC41405.2020.00024.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris Maddison, and Tatsunori Hashimoto. Identifying the risks of lm agents with an lm-emulated sandbox. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 27031–27098, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/7274ed909a312d4d869cc328ad1c5f04-Paper-Conference.pdf.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. {ALFW}orld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. EnvScaler: Scaling tool-interactive environments for LLM agent via programmatic synthesis. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 8326–8357, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.407. URL https://aclanthology.org/2026.findings-acl.407/.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757. Curran Associates, Inc., 2022. doi: 10.52202/068431-1508. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/82ad13ec01f9fe44c01cb91814fd7b8c-Paper-Conference.pdf.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R Narasimhan. {τ}-bench: A benchmark for \underline{T}ool-\underline{A}gent-\underline{U}ser interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=roNSXZpUDN.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for LLMs, 2024. URL https://openreview.net/forum?id=OqlmgmS4Wr.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Chen Xu, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. Siren's song in the ai ocean: A survey on hallucination in large language models, 2025. URL https://arxiv.org/abs/2309.01219.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. LlamaFactory: Unified efficient fine-tuning of 100+ language models. In Yixin Cao, Yang Feng, and Deyi Xiong (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 400–410, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-demos.38. URL https://aclanthology.org/2024.acl-demos.38/.

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 B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/4410c0711e9154a7a2d26f9b3816d1ef-Paper-Conference.pdf.

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 B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/4410c0711e9154a7a2d26f9b3816d1ef-Paper-Conference.pdf.

A Detailed Per-Delta Experimental Results

A.1 Simple Group ($\Delta \in \{1, 2\}$)

Table 5: Results on Simple Group. **Bold**: column-wise maximum; red: column-wise minimum.

	$\Delta = 1$		$\Delta = 2$		Average	
Model	FM	CM	FM	CM	FM	CM
DeepSeek-V3.2	80%	36%	80%	8%	80%	22%
Qwen3.5-397B-A17B	84%	64%	84%	28%	84%	46%
GPT-5.4	84%	60%	76%	20%	80%	40%
Gemini-3.1-Pro-Preview	**100%**	68%	**92%**	28%	**96%**	48%
Claude-Sonnet-4.6	8%	56%	12%	24%	10%	40%
MiniMax-M2.7	40%	72%	24%	28%	32%	50%
GLM-5	96%	64%	**92%**	24%	94%	44%
Full-Balance2	96%	**80%**	88%	**36%**	92%	**58%**

A.2 Medium Group ($\Delta \in \{3, 4, 5, 6\}$)

Table 6: Results on Medium Group. **Bold**: column-wise maximum; red: column-wise minimum.

	$\Delta = 3$		$\Delta = 4$		$\Delta = 5$		$\Delta = 6$		Avg	
Model	FM	CM	FM	CM	FM	CM	FM	CM	FM	CM
DeepSeek-V3.2	86%	24%	88%	6%	74%	**4%**	58%	0%	76.5%	8.5%
Qwen3.5-397B-A17B	78%	30%	62%	30%	60%	2%	60%	6%	65%	17%
GPT-5.4	82%	36%	70%	24%	74%	2%	**74%**	**8%**	75%	17.5%
Gemini-3.1-Pro-Preview	90%	36%	74%	24%	80%	**4%**	28%	6%	68%	17.5%
Claude-Sonnet-4.6	0%	30%	0%	16%	22%	0%	28%	2%	12.5%	12%
MiniMax-M2.7	24%	28%	28%	26%	26%	2%	10%	**8%**	22%	16%
GLM-5	90%	28%	86%	22%	90%	2%	60%	4%	81.5%	14%
Full-Balance2	**92%**	**44%**	**98%**	**40%**	**94%**	2%	**74%**	**8%**	**89.5%**	**23.5%**

A.3 Difficult Group ($\Delta \in \{7, \ldots, 12\}$)

Table 7: Results on Difficult Group, per Δ value and averaged. **Bold**: column-wise maximum; red: column-wise minimum. The CM rebound at $\Delta = 7$ reflects bulk-uniform operations (see Appendix A.4)

	$\Delta = 7$		$\Delta = 8$		$\Delta = 9$		$\Delta = 10$		$\Delta = 11$		$\Delta = 12$		**Avg**	
Model	FM	CM	FM	CM	FM	CM	FM	CM	FM	CM	FM	CM	FM	CM
DeepSeek-V3.2	19%	6%	25%	0%	20%	0%	55%	9%	0%	0%	25%	0%	26%	4%
Qwen3.5-397B-A17B	63%	38%	38%	**25%**	0%	0%	45%	**18%**	33%	**17%**	25%	**25%**	42%	24%
GPT-5.4	**88%**	44%	**100%**	**25%**	**60%**	0%	**73%**	**18%**	50%	**17%**	25%	**25%**	**74%**	26%
Gemini-3.1-Pro-Preview	63%	31%	88%	13%	40%	0%	64%	**18%**	**83%**	0%	**75%**	**25%**	68%	18%
Claude-Sonnet-4.6	19%	13%	25%	**25%**	0%	0%	9%	**18%**	0%	**17%**	0%	**25%**	12%	16%
MiniMax-M2.7	13%	**50%**	0%	0%	0%	0%	18%	**18%**	0%	0%	0%	**25%**	8%	22%
GLM-5	75%	44%	63%	**25%**	**60%**	**20%**	**73%**	**18%**	**83%**	**17%**	50%	**25%**	70%	**28%**

A.4 Non-Monotonicity at $|\Delta| \geq 7$: Bulk-Uniform Operations

Per-$|\Delta|$ results show a partial CM recovery at $|\Delta| = 7$, where Gemini 3.1 Pro, GLM-5, MiniMax-M2.7, and GPT-5.4 reach 31–50% CM before collapsing again at $|\Delta| \geq 8$. This rebound is not evidence of improved state-tracking capability. Inspection of the 15 Difficult-group samples on which

at least one model achieves CM = True reveals a consistent structural regularity. Every such sample corresponds to a *bulk-uniform* operation, in which a single tool call produces N structurally identical changes under the same path prefix. Two dominant patterns account for nearly all recoveries.

Pattern 1: Consecutive date-entry additions. A tool call such as `block_dates(property_id, start, end)` adds one availability record per day across a date range, yielding $|\Delta| = N$ changes that all follow an identical template: `availability_calendar.{property_id}.{YYYY-MM-DD}←{status: blocked}`, with keys that are consecutive calendar dates. Sample 376 ($|\Delta| = 10$) exemplifies this: all six models answer correctly, because identifying the pattern "add blocked entries for 2025-05-01 through 2025-05-10" reduces the task to enumerating a date sequence. No independent value derivation is required per entry. Samples 397 ($|\Delta| = 12$) and 352 ($|\Delta| = 7$) follow the same structure and achieve 5/6 and 5/6 correct respectively.

Pattern 2: Multi-field update on a single object. A call that overwrites all fields of one record (e.g., `update_reservation`) produces $|\Delta|$ equal to the number of updatable fields. Still, every change concerns the same object (e.g., `reservations.RES2.*`) and its values are read directly from the call arguments. Sample 377 ($|\Delta| = 10$) achieves 5/6 correct for exactly this reason: no cross-object inference or runtime-value derivation is required.

The contrast with Medium-group failures is instructive. A $|\Delta| = 5$ sample such as `verify_otp` requires five heterogeneous updates across three sub-objects, including a clock-generated timestamp whose value is unknowable without executing the transition logic. Each field demands an independent derivation. Bulk-uniform samples collapse this complexity into a single template instantiated N times, structurally easier despite the larger $|\Delta|$.

This distinction has a direct implication for benchmark design. The Difficult tier currently conflates two qualitatively different difficulty sources: *heterogeneous multi-field* updates, which cause a permanent CM collapse, and *homogeneous bulk* updates, which remain tractable at high $|\Delta|$. Future iterations of EnvSimBench should stratify these two sub-types separately to provide more precise diagnostic resolution at the upper end of the complexity axis.

B Benchmark Construction Pipeline (Detailed)

Figure 4 illustrates the full sample-selection pipeline with three-axis stratification and the diversity rule—the pipeline proceeds in three stages.

Stage 1: trajectory collection. GPT-4o-mini is deployed as an agent across all 191 EnvScaler environments, executing up to 30 steps per environment. At each step $t \geq 1$, the Python executor records the complete quadruple (a, o, s, s'): the tool call, the resulting observation, the before-config, and the after-config, along with the derived change set $\Delta(s, s')$. The initialization-only record is discarded because it has no corresponding tool-call transition. This yields a large pool of candidate single-turn state prediction samples.

Stage 2: three-axis stratification and filtering. Candidate samples are classified simultaneously along the three axes described in Section 4. Axis 1 partitions by action outcome (`success=True/False`); Axis 2 partitions the success group by $|\Delta|$ into No-Change, Simple, Medium, and Difficult; Axis 3 further splits Failure and No-Change subgroups by input argument cardinality (0 vs. 1). Each resulting subgroup is then subsampled to its target count (Table 1).

Stage 3: diversity rule. Within each subgroup, samples are selected greedily to maximize the number of distinct `env_id`s covered. If the target count exceeds the number of available environments in the subgroup, multiple samples per environment are permitted, prioritizing those with different $|\Delta|$ values. This rule ensures that the 400 benchmark samples span 167 distinct environments rather than being concentrated in a few high-trajectory environments, providing broad domain coverage for evaluation.

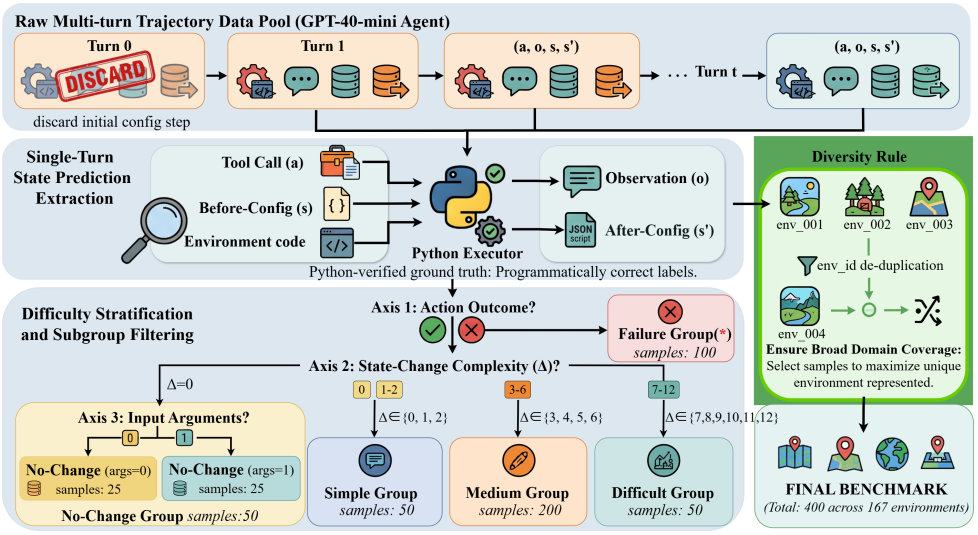


Figure 4: **Benchmark construction pipeline.** *Left*: Multi-turn trajectories from 191 EnvScaler environments are collected by a GPT-4o-mini agent and preprocessed into single-turn (a, o, s, s') samples; Step 0 (no before-config) is discarded. *Center*: Three-axis stratification partitions the pool by action outcome, state-change complexity, and input argument cardinality. *Right*: A diversity rule maximizes distinct `env_ids` per subgroup, yielding 400 samples across 167 environments with Python-executed ground truth.

C Performance Visualizations

C.1 Bar Chart Analysis: Per-Group FM and CM (Figure 5)

Figure 5 renders FM (blue bars) and CM (orange bars) side by side for all seven frontier models across five subgroups and the overall benchmark.

Panels (a) Failure and (b) No-Change: high CM with variable FM. All seven models achieve 100% CM on Failure samples, while CM on the pooled No-Change group ranges from 98.75% to 100%. Feedback accuracy varies more widely. Claude-Sonnet-4.6 achieves 15% FM on Failure samples despite 100% CM, illustrating that feedback errors can occur even when the predicted change set is correct. On the pooled No-Change group, GPT-5.4 achieves 66.25% FM and MiniMax-M2.7 achieves 65%, the lowest score in this group. The examples in Appendix D distinguish output-format mismatches from feedback-content errors.

Panel (c) Simple ($|\Delta| \in \{1, 2\}$): different rankings under FM and CM. FM ranges from 10% to 96%, while CM ranges from 22% to 50%. Gemini-3.1-Pro-Preview achieves the highest FM (96%), whereas MiniMax-M2.7 achieves the highest CM (50%), followed by Gemini at 48%. FM exceeds CM for five of the seven models. The exceptions are Claude-Sonnet-4.6 (10% FM versus 40% CM) and MiniMax-M2.7 (32% versus 50%). DeepSeek-V3.2 exhibits a large gap in the opposite direction, with 80% FM and 22% CM. These differences show that feedback accuracy and configuration accuracy induce different model rankings even on Simple samples.

Panels (d) Medium ($|\Delta| \in \{3, \ldots, 6\}$) and (e) Difficult ($|\Delta| \in \{7, \ldots, 12\}$): low CM with variation across groups. On Medium samples, CM ranges from 8.5% to 17.5%, while FM ranges from 12.5% to 81.5%. DeepSeek-V3.2 and GLM-5 exhibit the largest FM–CM gaps in this group, at 68.0 and 67.5 percentage points, respectively. On Difficult samples, CM ranges from 4% to 28%. GPT-5.4 achieves 74% FM and 26% CM, while GLM-5 achieves 70% FM and 28% CM. CM remains below the Simple-group score for every model, but several models score higher on Difficult than on Medium. The group-level results therefore show low configuration accuracy on complex transitions with a non-monotonic relationship to state-change count.

Panel (f) Overall: model ranking summary. The aggregate view makes the cross-model contrast most accessible. GLM-5 has the tallest FM bar (80.5%) but a moderate CM bar (41.0%). Qwen3.5-

397B-A17B has the tallest CM bar (42.3%). The two format-divergent models, Claude-Sonnet-4.6 and MiniMax-M2.7, have the shortest FM bars despite having CM bars comparable to the other models. Low FM does not imply low CM.

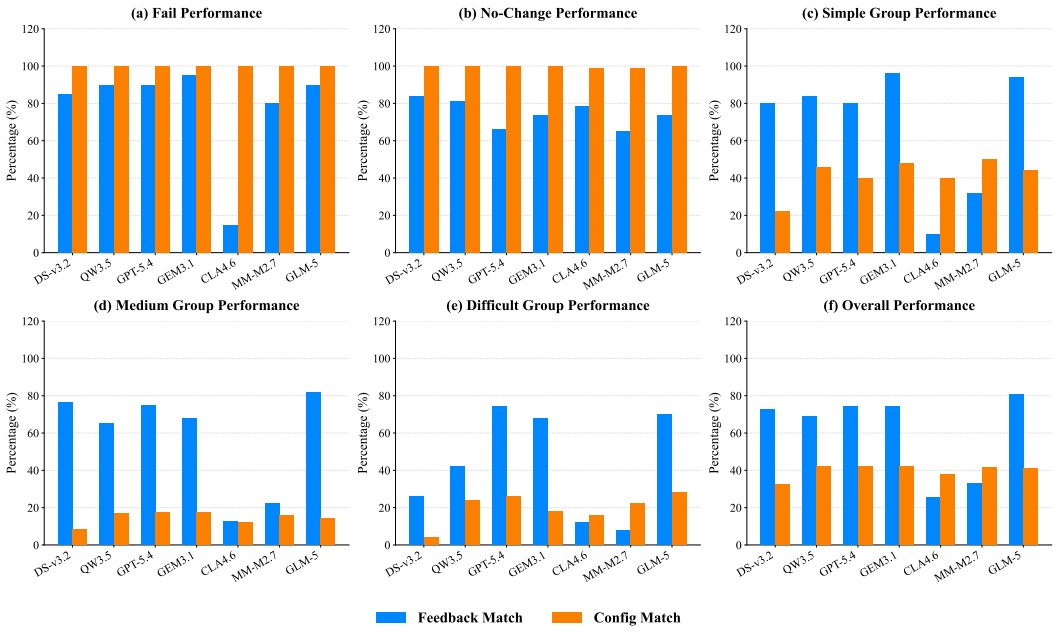


Figure 5: **Frontier LLM performance across difficulty subgroups.** Blue bars show FM and orange bars show CM for seven frontier models under non-thinking inference on the original benchmark. Panels (a)–(b): CM is near-perfect on Failure and No-Change samples, while FM varies across models. Panels (c)–(e): CM ranges from 22–50% on Simple samples, 8.5–17.5% on Medium samples, and 4–28% on Difficult samples. Panel (f): GLM-5 achieves the highest overall FM (80.5%), while Qwen3.5-397B-A17B achieves the highest overall CM (42.3%).

C.2 Heatmap Analysis: FM and CM by State-Change Count (Figure 6)

Figure 6 reports FM on the left and CM on the right for seven frontier models under non-thinking inference. Columns correspond to models, and rows correspond to state-change counts $|\Delta|$. The $|\Delta| = 0$ row pools Failure and No-Change samples, while the final row pools $|\Delta| \in \{7, \ldots, 12\}$. Blue indicates higher match rates and red indicates lower match rates.

Left panel (FM): variation across models and state-change counts. Claude-Sonnet-4.6 and MiniMax-M2.7 achieve 66% and 68% FM, respectively, on the pooled $|\Delta| = 0$ group. Their FM is lower on every state-changing row: Claude ranges from 0% to 28%, and MiniMax ranges from 8% to 40% across the displayed rows. The other five models also vary across change counts. For example, Gemini-3.1-Pro-Preview achieves 100% FM at $|\Delta| = 1$, 28% at $|\Delta| = 6$, and 68% on the pooled $|\Delta| = 7$–12 group. Thus, feedback accuracy varies across both models and sample strata. The format and content errors underlying these differences are examined through individual examples in Appendix D.

Right panel (CM): low accuracy on multi-field transitions. CM is 99–100% on the pooled $|\Delta| = 0$ group and ranges from 36% to 72% at $|\Delta| = 1$. It falls to 8–28% at $|\Delta| = 2$, but rises to 24–36% at $|\Delta| = 3$. The lowest ranges occur at $|\Delta| = 5$ and $|\Delta| = 6$, where CM is 0–4% and 0–8%, respectively. On the pooled $|\Delta| = 7$–12 group, CM ranges from 4% to 28%. These results show a pronounced loss of configuration accuracy relative to state-preserving transitions, with a non-monotonic pattern across state-change counts. MiniMax-M2.7 leads CM on the Simple group (50%), while GPT-5.4 and Gemini-3.1-Pro-Preview tie for the highest Medium-group CM (17.5%).

Comparing the two panels jointly. At $|\Delta| = 5$, GLM-5 achieves 90% FM and 2% CM, while DeepSeek-V3.2 achieves 74% FM and 4% CM. These gaps show that high feedback accuracy can

coexist with low configuration accuracy within the same sample stratum. Feedback accuracy alone therefore does not establish reliable state prediction. Appendix D reports the instance-level analysis of correct feedback paired with incorrect state changes.

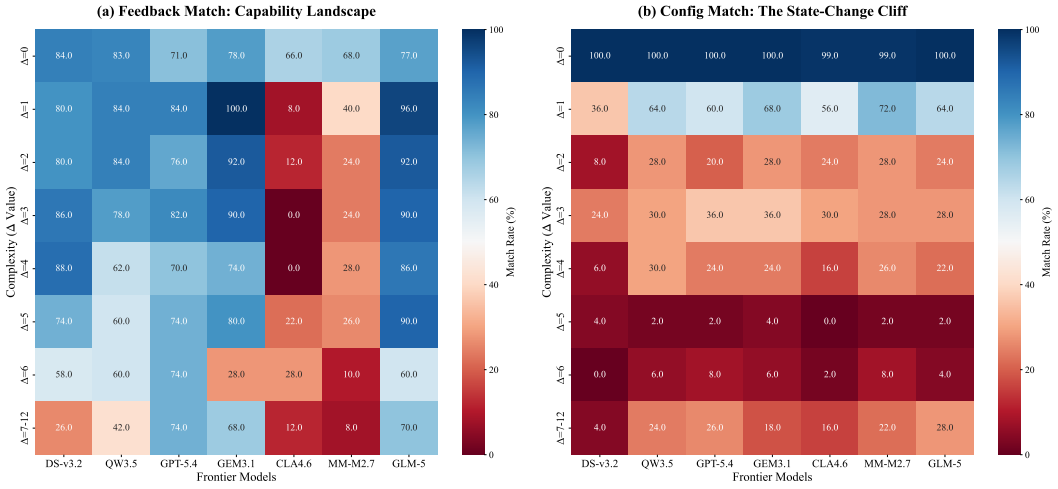


Figure 6: **Feedback and configuration accuracy by state-change count.** The heatmaps report FM (left) and CM (right) for seven frontier models under non-thinking inference. The $|\Delta| = 0$ row pools Failure and No-Change samples, and the final row pools $|\Delta| = 7$–12. CM is 99–100% at $|\Delta| = 0$ and 0–4% at $|\Delta| = 5$. Higher feedback accuracy can coexist with low configuration accuracy at the same change count.

D Extended Failure Mode Analysis

This appendix provides expanded qualitative and quantitative analyses of the three state-tracking failure sub-patterns identified in Finding 3, together with cross-model comparisons on shared benchmark samples.

Table 8: Format divergence examples. CM = 100% in all listed cases; FM = 0%.

Model	Predicted feedback	Ground-truth feedback
Claude	`Security not found`	`{'success': False, 'error': 'Security not found'}`
Claude	`No observations found for station 'ST001' on 2024-04-19.`	`{'success': False, 'error': 'No observations found...'}`
Claude	`Exactly one of 'delta_memory' or 'multiplier' must be provided.`	`{'success': False, 'error': 'Exactly one of...'}`
MiniMax	`User reputation updated successfully.`	`{'success': True, 'message': 'User reputation...'}`
MiniMax	`Movie 'tt0111161' updated successfully.`	`{'success': True, 'message': "Movie 'tt0111161'..."}`

D.1 Format Divergence: Full Example Set

As described in Finding 3, Claude-Sonnet-4.6 and MiniMax-M2.7 systematically unwrap the structured response schema and return only the inner message string. Table 8 enumerates representative cases. In every instance, CM is 100% (state prediction is correct) while FM is 0% (string mismatch). These examples demonstrate format sensitivity in FM; they do not establish a uniform error pattern across all difficulty levels.

One edge case deserves attention: a second Claude failure-group example predicts `"Workout scheduled successfully"` where the ground truth is a failure response. Here, beyond the format mismatch, the model also misclassifies the error condition. The rare case where FM = 0 reflects a genuine semantic error rather than a pure formatting convention difference.

This pattern reveals a counterintuitive deployment tradeoff: Claude-Sonnet-4.6 and MiniMax-M2.7, despite their low absolute FM, may be *safer* choices for simulation pipelines that use feedback mismatches as a reliability signal, since their errors are more likely to be *visible* to the monitoring layer. In contrast, models with near-perfect FM but poor CM produce silent corruptions that accumulate undetected.

D.2 State-Change Cliff: Cross-Model Analysis on Shared Samples

Example 1: `verify_otp` ($|\Delta| = 5$). This call must update five fields across three sub-objects simultaneously. Table 9 shows all seven models' field-level predictions on the same sample. Ground truth requires all five fields to be updated.

Table 9: Cross-model field-level predictions on `verify_otp` ($|\Delta| = 5$). ✓ = correctly predicted; × = missed or wrong value; † = path correctly identified but execution-time value wrong (stale-approximation error, sub-type (b) of Appendix D.3). Ground truth: all five fields must be updated.

Model	is_ver.	timestamp	attempt	is_used	sess.	FM	CM
DeepSeek-V3.2	×	×	✓	✓	✓	✓	×
Qwen3.5-397B-A17B	✓	†×	✓	✓	✓	✓	×
Gemini-3.1-Pro-Preview	×	×	✓	✓	✓	✓	×
GLM-5	×	×	✓	✓	✓	✓	×
MiniMax-M2.7	✓	×	✓	✓	×	✓	×
Claude-Sonnet-4.6	✓	×	×	✓	✓	✓	×
GPT-5.4	×	×	×	×	×	×	×

Three patterns are salient. First, the runtime-generated timestamp field is missed by *every* model: set to the current epoch at call time, it cannot be inferred statically. Second, DeepSeek-V3.2, Gemini-3.1-Pro-Preview, and GLM-5 independently converge on the same three-field prediction, suggesting a shared attentional bias toward the most linguistically salient fields. Third, GPT-5.4 collapses to zero predicted changes despite generating a feedback string, the full cliff failure mode. A fourth pattern distinguishes Qwen3.5-397B-A17B from the other six: it is the only model to identify all five field paths, yet still fails CM because its predicted `last_verification_timestamp` (`1774167900.0`) differs from the executor's runtime value (`1774167865.92`). This is a stale-approximation error rather than a path-identification failure, placing it in sub-type (b) of Appendix D.3.

Example 2: `return_equipment` ($|\Delta| = 5$). The ground-truth transition requires four primary field updates plus one newly created auxiliary log record. Every model fails:

- **DeepSeek-V3.2**: correctly updates three primary fields; omits one primary field and the auxiliary log record. FM = True, CM = False.
- **Claude-Sonnet-4.6**: assigns a domain-plausible but code-absent value to the primary status field; omits the auxiliary record. FM = False, CM = False.
- **Gemini-3.1-Pro-Preview / GLM-5 / MiniMax-M2.7**: same incorrect status value as Claude; omit one primary field and the auxiliary record. FM = False, CM = False.

- **GPT-5.4**: correctly derives all four primary field values, the only model to do so, but fabricates a runtime-generated key for the auxiliary record that mismatches the executor's output. FM = True, CM = False.

Across all $|\Delta| \geq 3$ CM failures, models omit newly *added* components in 68–79% of cases where such additions exist, substantially exceeding the omission rate for *modified* components. Four models independently hallucinate the same domain-plausible but code-absent status value, illustrating how parametric world knowledge overrides code-grounded reasoning when the required value is not directly readable from $\texttt{code}(a_t)$.

D.3 Runtime-Dependent Values: Failure Taxonomy

Timestamp-bearing and runtime-generated fields appear in the ground truth of 23–28% of all $|\Delta| \geq 3$ CM failures across models (Table 10). A model performing static reasoning over $\texttt{code}(a_t)$ cannot infer execution-time values without actually running the transition logic; it must either omit the field entirely or substitute a plausible but incorrect prior value. Three sub-patterns are observed.

Table 10: Presence of runtime-dependent fields in $|\Delta| \geq 3$ CM-failure samples.

Model	**CM failures** ($|\Delta| \geq 3$)	**With runtime field**
DeepSeek-V3.2	231	24%
Gemini-3.1-Pro-Preview	206	23%
GPT-5.4	202	28%
GLM-5	208	27%
Claude-Sonnet-4.6	218	26%
MiniMax-M2.7	207	28%

(a) Complete omission. The model identifies the correct field path but omits it from $\widehat{\Delta}$ entirely. This is the dominant pattern for DeepSeek-V3.2 and GLM-5.

(b) Stale-value substitution. The model includes the field with a plausible but stale value interpolated from the before-state s_t. In a representative case, a model predicts a stale proxy $\hat{v}_\tau$ for a clock-dependent field while the executor writes the true runtime value $v^*_\tau \neq \hat{v}_\tau$; the surface response $\hat{o}_t = o_t$ nonetheless matches (FM = 1, CM = 0).

(c) Transaction-ID key mismatch. For auto-generated record keys encoding a runtime epoch, a model must both create the record *and* predict the correct key. A correct record body with a wrong key still fails CM, a compound challenge of hidden side effect plus runtime dependency, as illustrated by GPT-5.4 in Example 2 above.

D.4 Plausible Feedback Masking Wrong State: Quantitative Breakdown

Table 11 reports the fraction of CM-failure samples that also pass FM at each $|\Delta|$ level. These are the most dangerous cases for simulation-as-training: the agent's reward signal is intact while the environment state is silently wrong.

Table 11: Percentage of CM-failure samples that also pass FM, by $|\Delta|$ level. High values indicate silent state corruption that is invisible to the agent. **Bold**: column-wise maximum; red: column-wise minimum.

Model	$\Delta = 1$	$\Delta = 2$	$\Delta = 3$	$\Delta = 4$	$\Delta = 5$	$\Delta = 6$
DeepSeek-V3.2	81%	78%	82%	**87%**	75%	58%
GLM-5	89%	**89%**	**86%**	85%	**90%**	58%
GPT-5.4	60%	70%	72%	63%	73%	**72%**
Gemini-3.1-Pro-Preview	**100%**	**89%**	84%	66%	79%	23%
Claude-Sonnet-4.6	18%	11%	0%	0%	22%	29%
MiniMax-M2.7	14%	28%	19%	35%	27%	9%

For DeepSeek-V3.2, GLM-5, GPT-5.4, and Gemini-3.1-Pro-Preview, 60–100% of CM failures at every $|\Delta|$ level are accompanied by FM = True: the agent sees no divergence signal. For Claude-Sonnet-4.6 and MiniMax-M2.7, the pattern inverts: their format divergence means errors are *more likely* also to fail FM, making divergence visible. This reveals a counterintuitive deployment tradeoff: models with format divergence, despite low absolute FM, may be safer for simulation pipelines that use feedback mismatches as a reliability signal.

In one representative case at $|\Delta| = 3$, two models independently produce the correct surface response o_t but collapse a nested sub-object to a coarse scalar representation (losing field-level granularity) and substitute a stale value for the clock-dependent timestamp field: $\hat{o}_t = o_t$ but $\hat{s}'_t \neq s'_t$. A third model generates correct feedback and correctly predicts two primary fields but omits the timestamp field entirely: both pass FM and fail CM. In a training pipeline, such silent divergences accumulate and systematically bias agent behavior.

E Hallucination Case Studies

This appendix presents executor-verified examples of fabricated rejection conditions, unsupported entity creation, and failure to use the explicitly provided before-state. These examples illustrate errors that remain possible under single-turn, explicit-state evaluation.

E.1 Failure Mode 1 Hallucination: Fabricating Non-Existent Constraint Violations

In this failure mode, the model invents a plausible business-rule violation to reject a call that the executor accepts. No such constraint exists in the tool code; the model substitutes reasoning grounded in real-world domain conventions for the actual implementation logic.

Sample: `add_observations` ($|\Delta| = 3$, all models fail). The call submits three wildlife observations with IDs `obs_9`, `obs_10`, `obs_11` for distinct observers (`OBSR02`, `OBSR05`, `OBSR04`) at timestamps `1685000000`, `1686000000`, `1687000000`, each approximately 11.6 days apart. The executor accepts all three and creates the corresponding records. The six model outputs detailed below all fail CM, but they differ in whether they reject the call or create records with incorrect values.

Table 12: Hallucination: all six models fabricate different constraint violations to reject three valid observations. Ground truth: 3 records created, `success=True`.

Model	Fabricated rejection reason
DeepSeek-V3.2	`"0 observations added, 3 failed"`; each observation individually flagged as `"Duplicate observation within 1 minute"`, despite timestamps being ∼11.6 days apart.
Gemini-3.1-Pro-Preview	`"0 observations added, 3 failed"`; observations flagged as `"Invalid latitude: 38.75"`, `"Invalid latitude: 69.3"`, `"Invalid latitude: 27.18"`, all of them are valid WGS-84 coordinates.
GPT-5.4	Same fabricated latitude-validity errors as Gemini-3.1-Pro-Preview; predicts zero state changes.
GLM-5	Same fabricated latitude-validity errors; predicts zero state changes.
Claude-Sonnet-4.6	Correctly outputs `"3 observations added successfully, 0 failed"` and creates 3 records, but uses *wrong field values* (hallucinated observer IDs and timestamps), causing CM = False.
MiniMax-M2.7	Same correct feedback as Claude, but with similarly wrong field values.

Among the six models detailed in Table 12, four (DeepSeek-V3.2, Gemini-3.1-Pro-Preview, GPT-5.4, and GLM-5) reject the valid call using constraints absent from the implementation. Claude-Sonnet-4.6 and MiniMax-M2.7 instead report success and create the requested records, but predict incorrect field

values. The example distinguishes fabricated rejection conditions from incorrect values in otherwise correctly identified updates.

E.2 Failure Mode 2 Logical Inconsistency: Self-Contradictory Response

In this case, the predicted feedback and state changes agree with each other but jointly deviate from the tool call: the model introduces an additional entity that the action did not request. The error is unsupported entity creation rather than a contradiction between the two predicted outputs.

Sample: `run_bulk_analysis` ($|\Delta| = 3$, DeepSeek-V3.2). The call requests a trend analysis for one Japanese stock, `7203.T` (Toyota). The executor processes only `7203.T` and updates three fields accordingly. DeepSeek-V3.2 produces a response that is internally self-contradictory in two ways simultaneously.

Table 13: Unsupported entity creation: DeepSeek-V3.2 feedback claims "2 stocks" were analyzed, but the call only specifies 1, and the predicted state invents a full analysis for a second stock (`SONY`) not present in the input.

	Ground truth	**DeepSeek-V3.2 prediction**
Feedback	`"Bulk analysis completed for 1 stocks in Japan."`	`"Bulk analysis completed for 2 stocks in Japan."`
State: `7203.T`	`technical_indicators.7203.T`: updated (SMA values) `trend_analysis.7203.T`: {confidence: 0.8}	Same fields updated correctly (*this part is right*)
State: `SONY`	*Not in request; no changes*	`technical_indicators.SONY`: full SMA table `trend_analysis.SONY`: {confidence: 0.8, trend: up} *(entirely fabricated)*

The inconsistency operates at two levels. First, the feedback count ("2 stocks") contradicts the input (1 stock was requested). Second, the predicted state changes include a complete analysis of `SONY` (with internally coherent SMA values, trend direction, and confidence score) that has no basis in the call arguments. The model correctly processes `7203.T` (the core prediction is right), then spontaneously hallucinates an entire second entity. An agent receiving this response would record a ghost analysis for SONY in the environment state, causing all subsequent reasoning about the SONY data to be grounded in fabricated values.

E.3 Failure Mode 3 Statelessness: Ignoring the Provided Before-State

In this failure mode, the model contradicts the *explicitly provided* before-config. Because our constraint-driven paradigm supplies the full before-state in every prompt, this failure cannot be attributed to implicit state tracking; the model overrides observed facts with a hallucinated prior.

Sample: `complete_task` ($|\Delta| = 2$, DeepSeek-V3.2). The tool call marks task `T-002` as completed. The before-config provided to the model explicitly states:

```
tasks.T-002.status: "in progress"
tasks.T-002.completed_at: null
```

The executor therefore accepts the call: the task transitions from `"in progress"` to `"completed"`, and `completed_at` is set to the current epoch.

The model asserts `"already completed"` while simultaneously having access to the before-config that says the opposite. This is not a state-tracking failure arising from implicit multi-turn inference. It explicitly contradicts the data present in the prompt for a single-turn prediction. The

Table 14: Statelessness: the model asserts the task is "already completed" despite the before-config explicitly showing `status = "in progress"`.

	Ground truth	DeepSeek-V3.2 prediction
Before-state (given in prompt)	`T-002.status: "in progress"` `T-002.completed_at: null`	*(same prompt)*
Feedback	`"Task 'T-002' marked as completed."`	`"error: Task 'T-002' is` **`already completed`**`"`
State changes	`T-002.status: "in progress"` → `"completed"` `T-002.completed_at: null` → `1774080132.66`	*(no changes predicted)*

most likely cause is that the model's parametric knowledge about task-management workflows (where double-completion calls are commonly guarded against) overrides its attention to the provided state. This pattern represents the purest form of hallucination in the environment simulation setting: the model's world model supersedes the ground truth it was given. The same sample was also misclassified by Gemini-3.1-Pro-Preview (which hallucinated a "dependency not met" constraint) and MiniMax-M2.7 (which hallucinated a "task not found" error), each substituting a different invented state for the explicitly provided one.

F Cross-Model Capability Profile Analysis

GPT-5.4 vs. Gemini-3.1-Pro-Preview: same aggregate CM, different per-delta profiles. Both models achieve 42.0% overall CM, but the per-delta breakdown from Appendix A reveals a crossover structure: Gemini-3.1-Pro-Preview leads at $\Delta \in \{1, 2\}$ (68%/28% vs. 60%/20% for GPT-5.4); both are identical at $\Delta \in \{3, 4\}$ (36%/24%); GPT-5.4 leads substantially at $\Delta \in \{7, 8\}$ (44%/25% vs. 31%/12%). This suggests that Gemini-3.1-Pro-Preview has stronger alignment for precise, low-complexity field prediction, while GPT-5.4 has higher resilience against the most complex multi-branch conditional logic. For practitioners choosing a simulation backend, Gemini-3.1-Pro-Preview may be preferable for environments dominated by simple CRUD-style operations, while GPT-5.4 may be preferable for environments with deeply nested state transitions.

DeepSeek-V3.2: strong FM, weak CM. DeepSeek-V3.2 achieves 72.5% overall FM (fourth highest) but only 32.5% CM (lowest), a 40 pp gap that is the largest FM–CM divergence in the evaluation. On state-changing samples, the gap reaches 58.7 pp (68.7% FM vs. 10.0% CM). As Table 11 shows, 75–87% of its CM failures are accompanied by FM = True across delta levels 1–5: the model generates correctly formatted, domain-appropriate response strings while substantially underperforming on code-execution tracing. This makes it a high-risk choice as a simulation backend for generating training data.

FM–CM gap as a model-selection signal. Table 15 ranks all models by the FM–CM gap on state-changing samples. A large positive gap indicates the model generates believable but inaccurate simulations; a near-zero or negative gap indicates visible failures. For use cases where the reliability of the training signal matters more than surface fluency, CM alone is the appropriate criterion for backend selection.

G Experimental Setup Details

G.1 Frontier LLM Evaluation

All seven frontier models are queried via their respective inference APIs in non-thinking mode. Every model receives identical prompts instantiating the MDP triple $(s_t, a_t, \texttt{code}(a_t))$; no model has

Table 15: FM–CM gap on state-changing samples. The gap is the difference between marginal feedback and configuration accuracy; joint error rates are reported separately. **Bold**: column-wise maximum; red: column-wise minimum.

Model	FM (state-chg)	CM (state-chg)	Gap (FM − CM)
GLM-5	**81.7%**	21.3%	**60.4 pp**
DeepSeek-V3.2	68.7%	10.0%	58.7 pp
GPT-5.4	75.7%	22.7%	53.0 pp
Gemini-3.1-Pro-Preview	72.7%	22.7%	50.0 pp
Qwen3.5-397B-A17B	64.3%	**23.0%**	41.3 pp
MiniMax-M2.7	21.3%	22.7%	−1.4 pp
Claude-Sonnet-4.6	12.0%	17.3%	−5.3 pp

access to prior turns or trajectory history. Ground-truth labels (o_t, s'_t) are produced entirely by the deterministic external executor, making evaluation LLM-free and immune to circular validation.

G.2 Model Identifiers and API Endpoints

Table 16 maps each display name used throughout the paper to the exact API model string queried during evaluation, together with the model provider. All models were called via their respective public inference APIs in **non-thinking** (standard) mode with no system-prompt modifications.

Table 16: Display name, exact API model string, and provider for each evaluated frontier model.

Display name (used in paper)	API model string	Provider
DeepSeek-V3.2	`deepseek-v3.2`	DeepSeek
Qwen3.5-397B-A17B	`qwen3.5-397b-a17b`	Alibaba Cloud
GPT-5.4	`gpt-5.4`	OpenAI
Gemini-3.1-Pro-Preview	`gemini-3.1-pro-preview`	Google DeepMind
Claude-Sonnet-4.6	`claude-sonnet-4-6`	Anthropic
MiniMax-M2.7	`MiniMax-M2.7`	MiniMax
GLM-5	`glm-5`	Zhipu AI

G.3 SFT Training Configuration

We fine-tune Qwen3-4B-Base using full-parameter SFT via LLaMA-Factory (Zheng et al., 2024) on 2×A800 (80 GB) GPUs with DeepSpeed ZeRO-3 parallelism (Rajbhandari et al., 2020). Key hyperparameters: learning rate 2×10^{-5} with a cosine schedule and 5% linear warm-up; 3 training epochs; per-device batch size 1 with gradient accumulation over 16 steps (effective batch size 32); bf16 mixed precision with FlashAttention-2. The maximum sequence length is set to 8,192 tokens. Each sample encodes the full transition logic $\texttt{code}(a_t)$ together with the structured before-state s_t, with token counts peaking near 5,000 across the corpus; this window ensures no truncation during training or evaluation. Inference is served via vLLM with a maximum generation length of 16,384 tokens and 90% GPU memory utilization.

All experiments are evaluated under the `env_id` holdout protocol, which excludes all 167 benchmark environments from training, including Findings 4 and 5 and the composition ablation in Appendix H.

H Data Composition Ablation

Table 17 reports the full composition ablation comparing three data mixture strategies, with frontier model performance as a reference.

Change-only (trained exclusively on state-mutating samples) achieves the highest CM on state-changing subsets (60% Simple, 24% Medium, 32% Difficult) but collapses to 0% FM on Failure and No-Change groups, despite maintaining 96% CM on these same samples. The model correctly predicts that no state change should occur in most cases, but has never learned to generate the

corresponding feedback strings: outputting state-change-style success messages where error messages or minimal acknowledgments are expected. In a deployed pipeline, this feedback corruption directly poisons the agent's reward signal: the environment state is preserved, but the agent receives systematically wrong feedback for every failed or no-op action, causing it to misattribute outcomes and learn incorrect action-consequence associations.

Balance adds failure and no-change samples but does not mirror the empirical $|\Delta|$ distribution, resulting in weaker performance on complex-change tiers. **Balance2** additionally upweights complex-change samples to match the source distribution, yielding the best overall CM (45.3%) and FM (79.5%), surpassing all frontier models on CM.

Table 17: Data composition ablation under `env_id` holdout. **Bold**: column-wise maximum; red: column-wise minimum.

	NC/F		Simple		Medium		Difficult		Overall	
Model	FM	CM	FM	CM	FM	CM	FM	CM	FM	CM
Change-only	0%	96%	**96%**	**60%**	**91.5%**	**24%**	52%	**32%**	64.3%	**47.5%**
Balance	73%	99%	**96%**	54%	89.0%	23%	38%	6%	79.5%	43.8%
Balance2 (Ours)	78%	99%	92%	58%	89.5%	23.5%	30%	12%	79.5%	45.3%
Frontier reference:										
GLM-5 (best FM)	77%	**100%**	94%	44%	81.5%	14%	**70%**	28%	**80.5%**	41.0%
Qwen3.5-397B-A17B (best CM)	**83%**	**100%**	84%	46%	65.0%	17%	42%	24%	69.0%	42.3%

I Pre-SFT Scaling Baseline

Table 18 shows Qwen3.5 instruct models (4B, 9B, 27B) without fine-tuning. Clear FM scaling trends emerge (2.3%→62.8%), but even the 27B model is substantially below the best-performing LLMs on the frontier, confirming that specialization is required beyond scale alone.

Table 18: Qwen3.5 non-thinking instruct models before SFT. **Bold**: column-wise maximum; red: column-wise minimum.

	NC/F		Simple		Medium		Difficult		Overall	
Model	FM	CM	FM	CM	FM	CM	FM	CM	FM	CM
Qwen3.5-4B	8%	95%	0%	**42%**	0.5%	10.5%	0%	6%	2.3%	35.0%
Qwen3.5-9B	39%	98%	6%	**42%**	1.5%	**11.6%**	0%	**20%**	11.3%	**38.1%**
Qwen3.5-27B	**82%**	**99%**	**78%**	26%	**58.5%**	2.0%	**26%**	4%	**62.8%**	29.5%

J Training Method Comparison

Table 19 compares LoRA and full-parameter fine-tuning on Qwen3-4B-Base at a 5K-sample budget under the `env_id` holdout. All hyperparameters are held fixed as described in Appendix G.

Full-parameter SFT raises FM from 27% to 72.8% (2.7×) and aligns the model's output schema with the expected response convention, correcting near-zero FM on Failure and No-Change samples (CM rises from 51% to 99% on these groups). For CM, the state-change cliff persists at the 5K-sample scale: Medium-group CM reaches only 10.5%, and Difficult-group CM falls to 6%, confirming that the cliff reflects a fundamental capability gap rather than a formatting artifact.

Full-parameter fine-tuning outperforms LoRA on CM under the same training-data budget (35.0% vs. 31.8%), with the advantage concentrated in the Simple tier (34% vs. 20%). We attribute this to LoRA's limited representational capacity, which overwrites the dense attention patterns that underlie sequential code-execution tracing. The state-change cliff persists under both methods at 5K samples, confirming it as a genuine capability gap rather than a parameter-efficiency artifact.

Table 19: Training method comparison on Qwen3-4B-Base (5K samples, `env_id` holdout). Full fine-tuning outperforms LoRA on CM, particularly in the Simple tier. **Bold**: column-wise maximum; red: column-wise minimum.

	NC/F		Simple		Medium		Difficult		Overall	
Model	FM	CM	FM	CM	FM	CM	FM	CM	FM	CM
Before SFT	18%	51%	34%	**40%**	34%	**22.5%**	10%	4%	27.0%	29.5%
LoRA-5K (w/o reason)	**67%**	**99%**	**90%**	20%	78.9%	5.0%	**46%**	16%	**73.2%**	31.8%
LoRA-5K (w/ reason)	66%	**99%**	**90%**	26%	77.5%	6.0%	34%	**20%**	70.8%	33.5%
Full-5K (w/o reason)	65%	**99%**	86%	34%	**84.0%**	10.5%	30%	6%	72.8%	**35.0%**
Full-5K (w/ reason)	63%	**99%**	88%	20%	80.5%	3.5%	32%	10%	71.0%	30.3%

Table 20: Supplementary reasoning-mode results (%). Failure + No-Change denotes state-preserving samples. Overall results use sample weighting.

	Failure + No-Change		State-Change		Overall	
Model	FM	CM	FM	CM	FM	CM
DeepSeek	79.00	99.00	65.67	19.00	69.00	39.00
Qwen	80.00	100.00	69.67	24.67	72.25	43.50
GPT	67.00	100.00	74.00	22.67	72.25	42.00
Gemini	77.00	100.00	86.33	24.00	84.00	43.00
Claude	79.00	100.00	60.67	19.67	65.25	39.75
MiniMax	73.00	100.00	20.33	22.33	33.50	41.75
GLM	75.00	99.00	80.67	22.00	79.25	41.25

K Supplementary Reasoning-Mode Evaluation

We additionally evaluate the seven frontier models with reasoning enabled. Table 20 reports feedback and configuration accuracy for the state-preserving and state-changing groups, together with overall results. This evaluation concerns inference-time reasoning and is distinct from the use of reasoning traces as SFT targets in Appendix J.

For compact presentation, DeepSeek denotes DeepSeek-V3.2; Qwen denotes Qwen3.5-397B-A17B; GPT denotes GPT-5.4; Gemini denotes Gemini-3.1-Pro-Preview; Claude denotes Claude-Sonnet-4.6; MiniMax denotes MiniMax-M2.7; and GLM denotes GLM-5. These abbreviations are also used in Appendix L.

Reasoning-mode overall CM ranges from 39.00% to 43.50%, with Qwen obtaining the highest score. Gemini achieves the highest overall FM at 84.00%. State-preserving CM remains at 99–100%, whereas state-changing CM ranges from 19.00% to 24.67%. Thus, substantial state-transition errors remain under reasoning-enabled inference. The aggregate results do not determine whether reasoning helps or hurts at each individual change count.

L Expanded Evaluation by State-Change Count

L.1 Expanded sample set and results

We synthesize an expanded set with 100 samples at each state-change count $k = |\Delta| \in \{0, \ldots, 12\}$, for 1,300 samples in total. Tables 21 and 22 report FM and CM separately for all seven models. This evaluation is reported separately from the original 400-sample benchmark because its sampling distribution differs. The $k = 0$ group contains transitions with no recorded configuration change; the results below are stratified by change count rather than action outcome.

CM ranges from 53% to 81% at $k = 1$, falls to 22–39% at $k = 2$, and reaches 1–6% at $k = 5$. The trend is non-monotonic: at $k = 7$, CM ranges from 6% to 24%. These results support a broad decline in configuration accuracy for multi-field transitions. Equalizing the sample count across k reduces

Table 21: Expanded evaluation: FM (%) by state-change count. Each row contains 100 samples. Model abbreviations are defined in Appendix K.

k	DeepSeek	Qwen	GPT	Gemini	Claude	MiniMax	GLM
0	68	69	57	61	64	62	62
1	87	93	78	95	66	27	95
2	88	92	69	90	70	33	93
3	100	84	63	96	80	24	91
4	78	71	64	89	67	27	82
5	85	68	48	87	65	27	85
6	71	69	55	82	76	8	75
7	28	78	55	89	67	11	78
8	25	38	37	64	75	0	75
9	20	0	20	48	60	0	40
10	25	42	50	52	56	17	50
11	0	33	33	48	52	16	50
12	40	42	40	45	50	23	56

Table 22: Expanded evaluation: CM (%) by state-change count. Each row contains 100 samples. Model abbreviations are defined in Appendix K.

k	DeepSeek	Qwen	GPT	Gemini	Claude	MiniMax	GLM
0	100	100	98	100	100	99	100
1	53	73	65	81	65	65	71
2	22	38	37	39	36	35	36
3	0	29	23	30	24	27	27
4	11	19	13	17	18	18	15
5	1	6	6	5	3	2	5
6	3	5	6	6	6	9	5
7	6	19	24	24	16	19	24
8	0	4	4	5	7	0	4
9	0	3	6	3	2	0	0
10	1	3	6	7	7	2	3
11	0	7	6	7	8	0	7
12	0	2	10	8	7	6	2

Table 23: Representative 95% percentile-bootstrap intervals for CM on the expanded set. Each row uses $n = 100$ instances. Values are percentages.

Model	k	CM	95% interval
DeepSeek	1	53.00	[43.00, 63.00]
DeepSeek	2	22.00	[14.00, 31.00]
DeepSeek	5	1.00	[0.00, 4.00]
DeepSeek	12	0.00	[0.00, 0.00]
Gemini	1	81.00	[72.00, 88.00]
Gemini	2	39.00	[30.00, 49.00]
Gemini	5	5.00	[1.80, 10.20]
Gemini	12	8.00	[3.10, 14.50]

the influence of individual examples within each stratum, but does not by itself separate the effects of change count, update pattern, tool logic, and runtime-dependent values.

L.2 Representative bootstrap confidence intervals

For each reported model and change count, we resample instances with replacement within that stratum and compute CM for each resample. The 2.5th and 97.5th percentiles give the reported 95% bootstrap interval. Table 23 presents the representative intervals available for two models and four change counts. These intervals describe instance-level resampling uncertainty within the evaluated strata; they do not measure variation across independently generated datasets or repeated model runs.

For DeepSeek at $k = 12$, all observed outcomes are failures. Resampling this all-zero outcome vector produces the degenerate empirical interval $[0, 0]$. This interval does not establish that the population success probability is zero. The intervals otherwise illustrate uncertainty in the observed CM rates and should be read alongside the point estimates and sample counts.