Toward Scalable Verifiable Reward: Proxy State-Based Evaluation for Multi-turn Tool-Calling LLM Agents

Anonymous ACL submission

Abstract

Interactive large language model (LLM) agents operating via multi-turn dialogue and multi-step tool calling are increasingly used in production. Benchmarks for these agents must both *reliably compare models* and *yield on-policy training data*. Prior agentic benchmarks (e.g., τ-bench, τ^2 bench, AppWorld) rely on fully deterministic backends, which are costly to build and iterate. We propose *Proxy State-Based Evaluation*, an LLM-driven simulation framework that preserves final state-based evaluation without a deterministic database. Specifically, a *scenario* specifies the user goal, user/system facts, expected final state, and expected agent behavior, and an LLM *state tracker* infers a structured proxy state from the full interaction trace. LLM *judges* then verify goal completion and detect tool/user hallucinations against scenario constraints. Empirically, our benchmark produces stable, model-differentiating rankings across families and inference-time reasoning efforts, and its on-/off-policy rollouts provide supervision that transfers to *unseen* scenarios. Careful scenario specification yields near-zero simulator hallucination rates as supported by ablation studies. The framework also supports sensitivity analyses over user personas. Human–LLM judge agreement exceeds 90%, indicating reliable automated evaluation. Overall, proxy state-based evaluation offers a practical, scalable alternative to deterministic agentic benchmarks for industrial LLM agents.

1 Introduction

LLM-based agents are increasingly deployed to solve *multi-turn*, *multi-step*, *tool-calling* tasks in industrial workflows (e.g., commerce, account management, customer operations). Building these agents requires two ingredients: (1) **stable evaluation** that reflects whether the agent truly accomplished user goals, and (2) **on-policy data generation** so the agent can learn from interaction, exploration, and environment feedback. Recent agentic benchmarks embrace a benchmark-as-environment paradigm (e.g., τ-bench, τ^2 bench, and AppWorld) (Yao et al., 2025; Barres et al., 2025; Trivedi et al., 2024) with multi-turn user↔agent dialogue, multi-step agent↔tool interaction over deterministic backends, and *final state-based evaluation* that scores terminal database state and user-facing responses rather than trajectory matching. While state-based evaluation accommodates the fact that there can exist multiple correct tool-calling paths, it relies on fully deterministic backends, which demands substantial engineering (schema design, deterministic tools, assertions), slowing iteration; for example, AppWorld reports ∼60K LOC for the engine and ∼40K LOC for the benchmark, plus ∼1.8K unit tests over 14 months (Trivedi et al., 2024).

To this end, we ask: *Can we retain the benefits of final state-based evaluation without building a heavy deterministic backend?* We answer with **Proxy State-Based Evaluation**, which judges success against an *LLM-inferred proxy final state* extracted from the complete interaction trace (conversation + tool calls/outputs). Concretely, we introduce: (i) a *scenario* object encoding the user goal, user/system facts, the expected final *proxy* backend state, and the expected final agent reply; (ii) an *LLM proxy state tracker* that infers state transitions and the proxy final state from multi-turn, multi-step traces; and (iii) an *LLM judge* that compares the proxy final state and agent responses against the scenario specification to decide goal completion. In production settings, this proxy, benchmark-as-environment design can be stood up quickly and evolve with product roadmaps. It generates multi-turn, multi-step rollouts suitable for model training, provides stable, model-differentiating metrics that guide iteration, and also specifies tool schemas that inform how tools should behave even while they are still under development.

Our benchmark yields *consistent capability ordering* across model families: goal completion (GC) scales with model strength and with inference-time reasoning effort. Training the RA (SFT, RFT) within the environment improves open-weight RAs using both on-policy and off-policy data. Ablation studies confirm the robustness of the proxy state tracker and to scenario completeness, and user persona variability is captured while keeping user-induced error low. Human–LLM judge agreement rate exceeds 90%, and the user and tool simulator hallucination rates are close to zero, supporting reliable evaluation.

Contributions. (1) We formalize *Proxy State-Based Evaluation* and instantiate a practical benchmark that preserves state-based evaluation without a deterministic backend. (2) We propose a scenario schema and five cooperating LLM components (reasoning agent, user simulator, tool simulators, state tracker, judge). (3) We define reliability criteria and diagnostics (bootstrapped SE, hallucination rates, human–judge agreement) and execute targeted ablations (persona sensitivity; system/user fact ablations; state-tracker strength). (4) We demonstrate how the environmeny yields scalable on-policy data and off-policy data for post-training (SFT/RFT) and supports an interactive evaluation for model comparison.

2 Related Work

State-based evaluation of interactive agents. Benchmarks such as τ bench, τ^2 bench, and AppWorld advance *final state-based evaluation*, checking terminal database state and final user response rather than trajectory matching because multiple distinct tool-calling paths call all correctly satisfy the same user goal. (Barres et al., 2025; Yao et al., 2025; Trivedi et al., 2024). Our work keeps this principle but replaces the heavy deterministic backend with an LLM-inferred *proxy* state and an *LLM judge*. Related work in LLM-based dialogue state tracking infers *dialog state* (e.g., user intents) from the conversation history (Carranza and Rojas, 2025; Hu et al., 2022b); however, it does not infer the *backend database* state that tools read/write and that is required for state-based evaluation. In contrast, we infer a verifiable proxy database state.

LLMs as simulators and judges. LLMs have been used as user simulators, environment simulators, and automatic judges for open-ended tasks (Wang et al., 2024; Zheng et al., 2023). We extends this idea to *state-based evaluation* by: (1) extracting a structured *proxy state* from the full trace and (2) checking outcome conditions against a scenario specification.

On-policy data generation from simulation. Interactive environments support both on-policy and off-policy data generation for model training (e.g., RFT, DPO, GRPO, expert iteration) (Trivedi et al., 2024; Chen et al., 2025). Our benchmark is designed to produce *rewarded, on-policy* traces centered on tool calling and end-state verification.

3 Preliminaries

3.1 Task Overview

We study the task where a *reasoning agent* (RA) must achieve a user goal via *multi-turn* dialogue and *multi-step* tool calling.

Turns (user $\leftrightarrow$ RA). A *turn* $t \in \{1, \dots, T\}$ begins with a user query U_t and ends when the RA emits a user-facing text message Y_t (e.g., answer, clarification, or follow-up). A task is *multi-turn* if $T \geq 2$.

Steps (RA $\leftrightarrow$ tools). Before the RA emits Y_t to the user in turn t, the RA will execute K_t tool steps $S_t = ((a_{t,k}, r_{t,k}))_{k=1}^{K_t}$, where $a_{t,k} = \textsc{ToolCall}(\text{tool}_{t,k}, q_{t,k})$ for $\text{tool}_{t,k} \in \mathcal{T}$ and $r_{t,k}$ is the tool's structured return. We say turn t is *multi-step* if $K_t \geq 2$. Tools in $\mathcal{T}$ take natural-language subqueries $q_{t,k}$ (LLM subagents or NL-native services, e.g., search); concrete tools are in Sec. 4.

Interactive Trajectory Simulation. A *trajectory* $\tau = (U_1, S_1, Y_1, \dots, U_T, S_T, Y_T)$ records user utterances U_t, intra-turn tool-step sequences S_t, and agent messages Y_t. A *user simulator* f_{user} generates U_t conditioned on RA's message Y_t and the scenario z; a *tool simulator* f_{tool} returns a structured output $r_{t,k}$ conditioned on the RA's subquery $q_{t,k}$ and the scenario z; a *state tracker* f_{state} updates the proxy state $\tilde{s}_{t,k}$ after each step. The scenario z supplies the information these components condition on (Sec. 4). See Appendix D for a concrete, step-by-step example trajectory.

State Tracking and State-based Evaluation. We maintain a structured *proxy state* $\tilde{s}_{t,k}$ that approximates the latent database state at step (t, k). It is "proxy" because it is *inferred* by an LLM state tracker f_{state} from tool calls rather than read from a deterministic database. The initial proxy

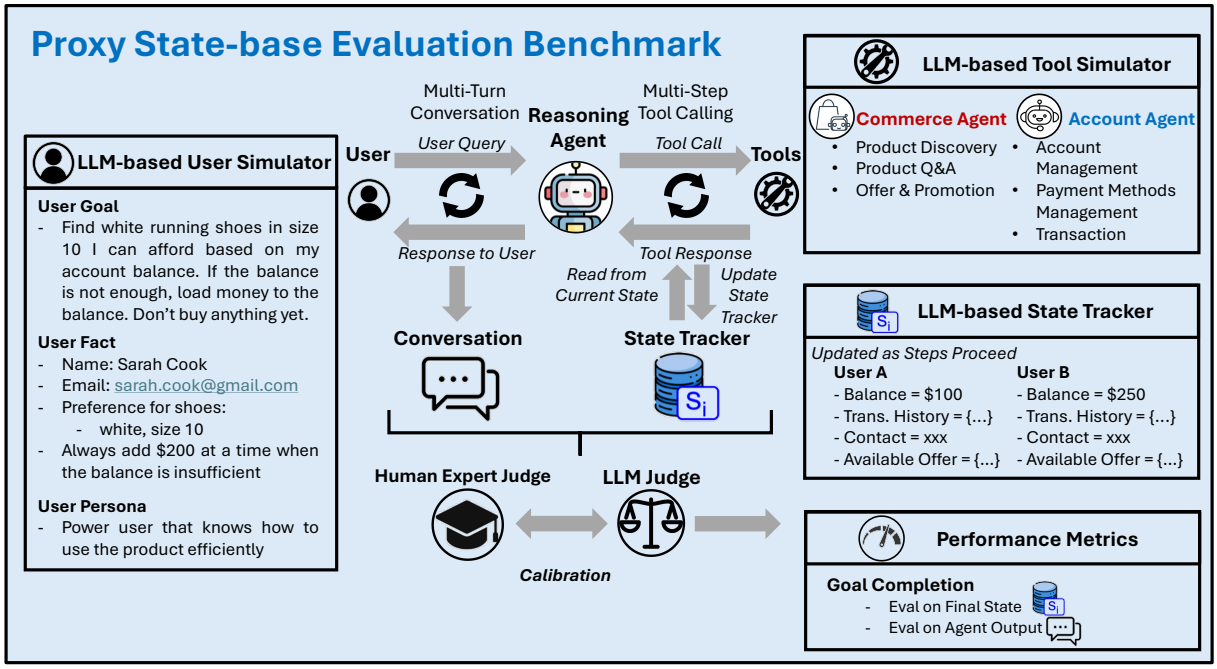

Figure 1: Overview of the proxy state-based evaluation benchmark. In a *multi-turn* interaction, an LLM-based user simulator converses with a reasoning agent that plans and executes *multi-step* tool calls to LLM-based tool simulators. An LLM judge, calibrated with human experts, determines goal completion by checking the final proxy state. The benchmark 1) evaluates the reasoning agent's ability to achieve goals via multi-turn dialogue and tool-calling, and also 2) yields conversation data with rewards and supporting a leaderboard for comparing reasoning agents.

state is initialized by the scenario ($\tilde{s}_{1,0} = s_0(z)$). After each tool step, the state tracker computes $\tilde{s}_{t,k} = f_{\text{state}}(\tau_{\leq(t,k)})$, where $\tau_{\leq(t,k)}$ denotes the interaction history up to step (t,k).[1] The tool simulator f_{tool} "reads from" and "writes to" the current proxy state $\tilde{s}_{t,k}$. After the entire conversation finishes, state-based evaluation uses an LLM judge J to check whether the final state $(\tilde{s}_T, y_T)$ satisfies the scenario's *expected final state* $s^*(z)$ and *expected agent behavior* $b^*(z)$ to determine if the goal is successfully completed (details in Sec. 4).

4 Methods

Scenario We follow the paradigm of recent agentic benchmarks (Barres et al., 2025; Yao et al., 2025; Trivedi et al., 2024), which avoid labeling ground-truth trajectories for each task. Such labeled trajectory ignores the fact that multiple distinct tool-calling paths may correctly satisfy the same user goal. Instead, we define a *scenario* z that specifies outcome-level constraints rather than trajectory-level matching.

[1] Rather than maintaining a purely step-by-step evolving state $\tilde{s}_{t,k} = f(\tilde{s}_{t,k-1}, \cdot)$, which may accumulate errors over long trajectories, we condition the state tracker on the full trajectory prefix to ensure robustness.

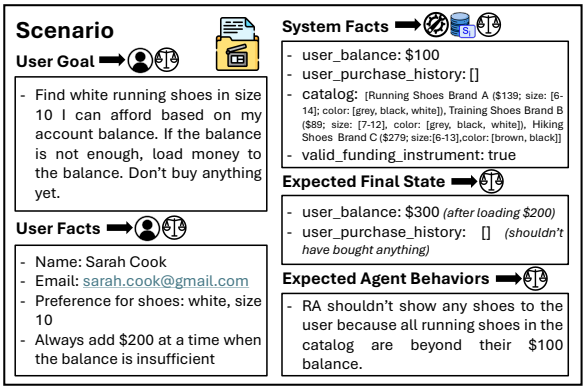

Figure 2: A scenario z specifies user goal $g(z)$ and user facts $u(z)$ (both used by user simulator and LLM judge), system facts $s_0(z)$ (used by tool simulators, state tracker, and LLM judge), expected final state $s^*(z)$, and expected agent behavior (both used by LLM judge). Arrows denote inputs. These fields drive the interactive simulation and proxy state-based evaluation.

As illustrated in Fig. 2, a scenario z provides: the *user goal* $g(z)$ and *user facts* $u(z)$ for the user simulator f_{user}; the *system facts* $s_0(z)$ (initial database state) for the tool simulator f_{tool}, the state tracker f_{state}, and the *expected final state* and *expected agent behavior* $(s^*(z), b^*(z))$ for the LLM judge. Evaluation is therefore based on whether the

proxy final state and user-facing message satisfy $(s^*(z), b^*(z))$, treating all correct paths equally.

We ensure internal consistency within each scenario. The expected final state $s^*(z)$ must logically follow from the user goal $g(z)$ and system facts $s_0(z)$. For example, if the goal is to add $100 to the balance and $s_0(z)$ specifies a valid funding instrument, then $s^*(z)$ reflects the corresponding balance increase of $100. We ensure that all information required by the user simulator and tool simulator is fully specified in z. Empirically, we ensure near-zero user and tool hallucination rates in simulation (Sec. 5). All scenarios are synthetic but designed to cover diverse and realistic workflows in e-commerce and account management. Our benchmark contains $|\mathcal{Z}| = 208$ scenarios, partitioned into a training set $\mathcal{Z}_{\text{train}}$ (size = 157) and a testing set $\mathcal{Z}_{\text{test}}$ (size = 51).

Reasoning Agent (RA) and Subagents The RA is the model under evaluation and also the primary training optimization target. It follows a ReAct-style loop (Yao et al., 2023): *reason* $\rightarrow$ *act* (tool call or show messages to the user) $\rightarrow$ observe tool/user response $\rightarrow$ next step.

The RA's action space comprises three calls: `call_account`(q), `call_commerce`(q), and `show_to_user`(q), where commerce and account capabilities are served by two LLM-powered subagents that parse the subquery q and return JSON outputs, and `show_to_user` is a special action that emits the user-facing message Y_t for turn t and concludes the turn t.

User simulator (f_{user}) The user simulator generates the next user utterance U_t conditioned on the scenario's *user goal* $g(z)$, *user facts* $u(z)$, the selected persona $p \in \{\text{power}, \text{ambiguous}, \text{confused}\}$ (see Sec. 5), and the RA's previous user-facing messages Y. It emits a special <done> token when it believes its user goal has been fulfilled, or when the maximun turn $T_{\max} = 10$ is exhausted.

Tool simulators (f_{tool}) Tool simulators implement `call_account` and `call_commerce` (Fig. 1). Each tool call is generated conditioned on: (1) the *system facts* $s_0(z)$ (initial database state), (2) the current *proxy state* $\tilde{s}_{t,k-1}$, and (3) the RA's subquery $q_{t,k}$. Formally, tool outputs are produced as $r_{t,k} = f_{\text{tool}}(s_0(z), \tilde{s}_{t,k-1}, q_{t,k})$, ensuring that tools "read from" the current proxy state. Tool simulators are constrained not to fabricate information beyond the state $(s_0(z), \tilde{s}_{t,k-1})$ and the RA's subquery content q.

Proxy State Tracker The state tracker f_{state} implements the proxy state mechanism defined in Sec. 3. At each step (t, k), it infers the current proxy state $\tilde{s}_{t,k} = f_{\text{state}}(\tau_{\leq(t,k)})$. Tool calls are categorized as *read* or *write* operations. While read-only calls (e.g., product search or transaction lookup) do not modify state, write operations (e.g., "add $200 to balance" or "create dispute") modify state fields only if the corresponding tool output $r_{t,k}$ indicates success.

LLM Judges Given the *proxy final state* $\tilde{s}_T$ and the entire trajectory τ (which includes all the user-facing messages), the judge evaluates them against the scenario specification $(s^*(z), b^*(z))$. Concretely, we use two LLM judges as follows.

(1) Goal-completion Judge. The primary judge classifies $(\tilde{s}_T, \tau$ into one of three outcomes: (i) *goal completed*, (ii) *goal not completed due to user error*, or (iii) *goal not completed due to agent error*. Formally, it produces $J_{\text{goal}}(\tilde{s}_T, y_T, \tau, z) \rightarrow \{c, e_{\text{user}}, e_{\text{agent}}\}$, where $c \in \{0, 1\}$ indicates goal completion.

(2) Hallucination Detection Judge. A separate judge detects hallucinations and returns two binary indicators: $J_{\text{hall}}(\tau, z) \rightarrow \{h_{\text{tool}}, h_{\text{user}}\}$, where $h_{\text{tool}} \in \{0, 1\}$ indicates *tool hallucination* and $h_{\text{user}} \in \{0, 1\}$ indicates *user hallucination*. Tool hallucination is defined as the tool simulator f_{tool} producing information that is not supported by the scenario's system facts $s_0(z)$ or RA's subqeury q. User hallucination is defined as the user simulator f_{user} generating information that is inconsistent with the scenario's user facts $u(z)$, the user goal $g(z)$, and RA's response Y (Ji et al., 2023).

To validate the LLM judges, we compare the goal-completion judge and the detection judge against two independent human domain expert annotations. The human–LLM judge agreement rate exceeds 90% (Appendix A).

Evaluation Metrics. For a set of scenarios $\mathcal{Z}$ and any binary indicator $x(z)$, we define its rate as $\text{Rate}(x) = \frac{1}{|\mathcal{Z}|} \sum_{z \in \mathcal{Z}} x(z)$. We report goal completion rate $\text{GC} = \text{Rate}(c)$, user-error rate $\text{ER}_{\text{user}} = \text{Rate}(e_{\text{user}})$, agent-error rate $\text{ER}_{\text{agent}} = \text{Rate}(e_{\text{agent}})$, tool hallucination rate $\text{HR}_{\text{tool}} = \text{Rate}(h_{\text{tool}})$, and user hallucination rate $\text{HR}_{\text{user}} = \text{Rate}(h_{\text{user}})$. All reported evaluation metrics are computed on $\mathcal{Z}_{\text{test}}$.

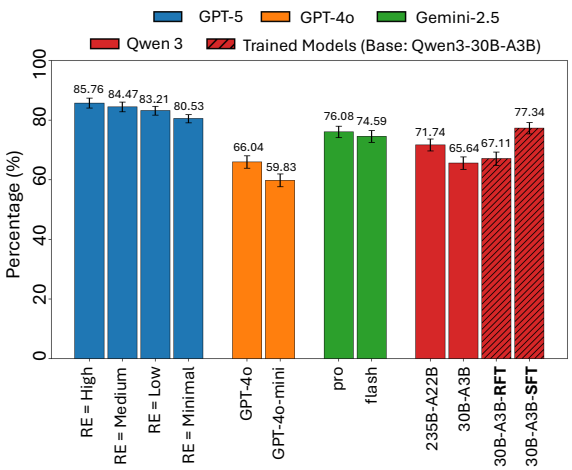

Figure 3: Goal completion rate (GC) on testing scenarios $\mathcal{Z}_{\text{test}}$ across baseline reasoning agents and trained models. Error bars show the bootstrap standard error. Fine-tuning substantially improves the base Qwen3-30B-A3B-Thinking-2507 model. RE: reasoning effort.

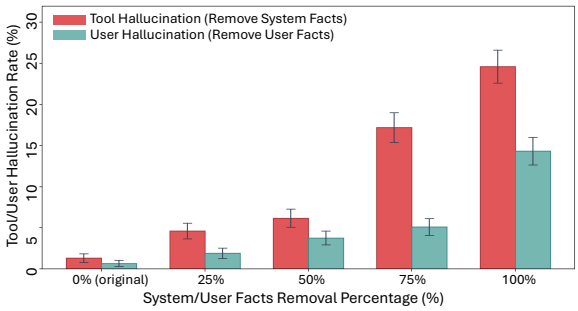

Figure 4: **Ablations on scenario facts increase hallucinations.** We randomly remove a fraction of system facts $s_0(z)$ or user facts $u(z)$. Tool hallucination rate and user hallucination rate rise steadily with more facts being removed. Error bars show the bootstrap standard error.

Training We investigate two training paradigms for the LLM underlying the reasoning agent (RA), using trajectories from $\mathcal{Z}_{\text{train}}$. Only the RA's LLM parameters are updated; other LLMs (f_{user},f_{tool}, f_{state}, J) remain fixed. **Training with On-policy Data.** The current RA interacts with the simulator to generate trajectories τ, each scored by J_{goal}; we retain only c=1 rollouts and use them as supervised targets for rejection-sampling fine-tuning (RFT; Anil et al., 2025) of the RA. **Training with Off-policy Data.** We replace the RA with a stronger teacher to generate trajectories on the same $\mathcal{Z}_{\text{train}}$; again, only c=1 rollouts are retained and used for supervised fine-tuning (SFT; Ouyang et al., 2022) of the base RA.

5 Experimental Settings

Domains and Tools. We expose two tool families $\mathcal{T}$: *Commerce* (Product Discovery, Checkout, Cart Management, Product Q&A, Offers & Promotions) and *Account* (Account Management, Wallet & Funding, Payment & Transfer, Dispute & Refund, Security & Fraud, Transaction Inquiry). The reasoning agent (RA) interacts via `call_commerce`(q), `call_account`(q), and `show_to_user`(q) as defined in Sec. 4.

Reasoning Agent (RA) Models. We evaluate a diverse set of LLMs as the RA, including: GPT-5 (reasoning effort $\in$ {*minimal*, *low*, *medium*, *high*}) (Singh et al., 2025), GPT-4o, GPT-4o-mini, Gemini-2.5-pro, Gemini-2.5-flash (Comanici et al., 2025), Qwen3-235B-A22B, and Qwen3-30B-A3B-Thinking-2507 (Yang et al., 2025). All models use temperature $= 1$ during trajectory rollout.

Simulators and Judge Models. Unless otherwise specified, the user simulator f_{user}, tool simulator f_{tool}, state tracker f_{state}, and LLM judges J are instantiated using GPT-5o with medium reasoning effort. This configuration empirically yields near-zero tool hallucination rate (1.33%) and user hallucination rate (0.67%).

User Personas. The user simulator f_{user} is instantiated with a persona variable $p \in \{\text{power}, \text{ambiguous}, \text{confused}\}$. Unless otherwise specified, we evaluate with $p =$ power to ensure that failure cases are primarily attributable to the RA rather than user behaviors. Detailed persona definitions are provided in Appendix C.

Training Configuration. For training experiments, we use Qwen3-30B-A3B-Thinking-2507 as the base RA model and fine-tune it on $\mathcal{Z}_{\text{train}}$ as described in Sec. 4. Detailed hyperparameters and training data statistics are provided in Appendix B.

6 Results

6.1 Goal Completion Across Models

Baseline model comparison. Across model families (Fig. 3), goal completion rate (GC) scales with model strength and inference-time reasoning effort. Larger variants consistently outperform their smaller counterparts (e.g., GPT-4o $>$ GPT-4o-mini; Gemini-2.5-Pro $>$ Flash; Qwen3-235B $>$ 30B), and within the GPT-5 family, increasing reasoning effort yields a monotonic

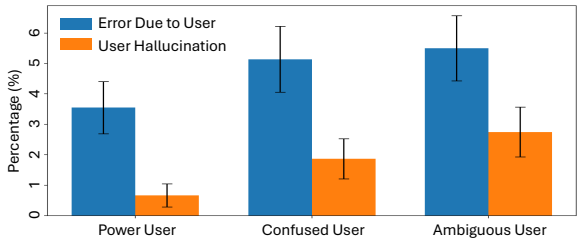

Figure 5: User persona sensitivity analysis. Error due to user ($\mathrm{ER}_{\mathrm{user}}$) and user hallucination rate ($\mathrm{HR}_{\mathrm{user}}$) across three personas p. More challenging personas increase user-induced errors and user hallucination rates. Error bars denote bootstrap standard error.

GC gain (high $>$ medium $>$ low $>$ minimal; e.g., $85.76 > 80.53\%$). The alignment of these trends with expected capability ordering indicates that our proxy state-based evaluation is suitable for *ranking and separability* of RA's performance.

Training improvements. We further evaluate training effects on Qwen3-30B-A3B-Thinking-2507. The base model achieves 65.64% GC. Rejection-sampling fine-tuning (RFT) yields a modest improvement to 67.11%, while supervised fine-tuning (SFT) using filtered successful trajectories substantially improves GC to 77.34%.

6.2 Ablations

Proxy State Tracker Model. To assess the impact of the state tracker f_{state}, we replace the default configuration (GPT-5o, medium reasoning effort) with the weaker model GPT-4o while keeping all other components fixed. The tool hallucination rate increases from $1.33\% \pm 0.53$ to $3.61\% \pm 0.88$. This suggests that weaker state inference degrades consistency of the proxy state $\tilde{s}_{t,k}$, which in turn propagates errors to tool outputs since f_{tool} reads from the current proxy state. The results highlights the importance of accurate state tracking for stable state-based evaluation.

System-Fact and User-Fact Ablations. We further ablate scenario completeness by randomly removing a fraction of the specified *system facts* $s_0(z)$ or *user facts* $u(z)$ while keeping the evaluated RA fixed. Fig. 4 shows a monotonic degradation: removing system facts substantially increases tool hallucination. Similarly, removing user facts increases user hallucination. These results validate that hallucination rates are sensitive to the underlying scenario specification: incomplete $s_0(z)$ induces tool-side fabrication, while incomplete $u(z)$ induces user-side fabrication. Overall, these results highlight that the meticulous effort we invest in curating scenario files is essential. It specifies the complete information required by each scenario keeps simulation grounded and minimized both tool- and user-side hallucinations.

6.3 User Persona Sensitivity

We evaluate the impact of user personas $p \in \{\text{power}, \text{confused}, \text{ambiguous}\}$ on error patterns. Fig. 5) reports error due to user ($\mathrm{ER}_{\mathrm{user}}$) and user hallucination rate ($\mathrm{HR}_{\mathrm{user}}$) across personas. When evaluated with the default *power* user, the user-error rate is 3.55% and user hallucination rate is 0.67%. In contrast, the *confused* and *ambiguous* personas exhibit higher user-error rates (5.14% and 5.50%, respectively) and higher user hallucination rates (1.87% and 2.75%). These results demonstrate that the benchmark meaningfully captures variation in user behavior. Importantly, we use power-user as our default setting (similar to Yao et al., 2025) so user-induced errors remain low. This ensures that model comparisons primarily reflect RA performance rather than user-side errors.

7 Conclusion

We introduced *Proxy State-Based Evaluation*, a benchmark-as-environment that preserves the benefits of final state evaluation without the engineering burden of a fully deterministic backend. Our scenario schema and cooperating LLM components (RA, f_{user}, f_{tool}, f_{state}, J) yield stable, interpretable metrics and *model-differentiating rankings* across families and reasoning-effort settings. Reliability is supported by human–LLM agreement (>90%) and near-zero simulator hallucination under the default configuration. Ablations studies show the importance of accurate state inference and scenario completeness. Beyond evaluation, the environment produces on-/off-policy rollouts that improve an open-weight RA via SFT/RFT and *transfer to unseen scenarios*. Persona studies confirm meaningful sensitivity to user. Taken together, our industrial evaluation indicates that this proxy environment is a practical, scalable alternative to deterministic suites. It supports faster training iteration for LLM agents while retaining rigorous state-based evaluation.

References

Gautham Govind Anil, Dheeraj Mysore Nagaraj, Karthikeyan Shanmugam, and Sanjay Shakkottai. 2025. Rejection sampling based fine tuning secretly performs ppo. In *Second Workshop on Test-Time Adaptation: Putting Updates to the Test! at ICML 2025*.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. τ^2-bench: Evaluating conversational agents in a dual-control environment. *Preprint*, arXiv:2506.07982.

Rafael Carranza and Mateo Alejandro Rojas. 2025. Interpretable and robust dialogue state tracking via natural language summarization with llms. *arXiv preprint arXiv:2503.08857*.

Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. 2025. Retaining by doing: The role of on-policy data in mitigating forgetting. *arXiv preprint arXiv:2510.18874*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022a. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A Smith, and Mari Ostendorf. 2022b. In-context learning for few-shot dialogue state tracking. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2627–2643.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, and 1 others. 2025. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. 2024. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16022–16076, Bangkok, Thailand. Association for Computational Linguistics.

Ruoyao Wang, Graham Todd, Ziang Xiao, Xingdi Yuan, Marc-Alexandre Côté, Peter Clark, and Peter Jansen. 2024. Can language models serve as text-based world simulators? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–17, Bangkok, Thailand. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R Narasimhan. 2025. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

A Human Evaluation and Inter-Rater Agreement

Protocol. Two domain experts independently annotated n=50 randomly sampled conversations from $\mathcal{Z}_{\text{test}}$ across three dimensions: goal completion c, tool hallucination h_{tool}, and user hallucination h_{user}. We compare their labels to the outputs of the LLM judges J_{goal} and J_{hall} defined in the main paper.

Results

Table 1 reports the *three-way agreement* rate, i.e., the percentage of examples where *both* human annotators and the LLM judge fully agree on the label for a given dimension.

Dimension	Three-way Agreement (%)
Goal completion (c)	82.7
Tool hallucination (h_{tool})	94.7
User hallucination (h_{user})	94.7

Table 1: Three-way agreement among two human experts and the LLM judge on n=50 conversations.

Takeaways

The LLM judges align closely with human experts, supporting the reliability of our evaluation setup.

B Training Data and Hyperparameters

Training data. We roll out each training scenario $z \in \mathcal{Z}_{\text{train}}$ ($|\mathcal{Z}_{\text{train}}|$=157) for 10 trajectories, yielding $\mathcal{D}_{\text{raw}} = \{\tau_i\}_{i=1}^{1570}$. Each trajectory τ is scored by the goal-completion judge J_{goal} with indicator $c(\tau) \in \{0, 1\}$. For rejection-sampling fine-tuning (RFT) and supervised fine-tuning (SFT), we success-filter as

$$\mathcal{D}_{\text{rft}}^{\text{succ}} = \{\tau \in \mathcal{D}_{\text{raw}} : c(\tau)=1\}, \quad |\mathcal{D}_{\text{rft}}^{\text{succ}}| = 1078; \qquad \mathcal{D}_{\text{sft}}^{\text{succ}} = \{\tau \in \mathcal{D}_{\text{raw}} : c(\tau)=1\}, \quad |\mathcal{D}_{\text{sft}}^{\text{succ}}| = 1209.$$

Each successful trajectory is decomposed into stepwise supervision pairs at RA emission points (tool calls or user-facing messages): $\mathcal{S}_{\text{rft}} = \{(x_s, y_s)\}$ with $|\mathcal{S}_{\text{rft}}| = 5017$, and $\mathcal{S}_{\text{sft}} = \{(x_s, y_s)\}$ with $|\mathcal{S}_{\text{sft}}| = 8057$, where x_s is the trajectory prefix immediately before emission (the history $\tau_{<s}$) and $y_s \in \{a_{t,k}, Y_t\}$ is the RA's next response.

Hyperparameters. We fine-tune the base RA model (Qwen3-30B-A3B-Thinking-2507) with LoRA (Hu et al., 2022a):

- LoRA rank/alpha: 32 / 32
- LoRA targets: all linear projections in self-attention (Q, K, V, O)
- MoE router auxiliary loss coefficient: 1×10^{-3}
- Learning rate: 1×10^{-5} (constant; no scheduler)
- Training epochs: 2

C User Persona Definitions

The user simulator f_{user} supports three persona p configurations:

- **Power user:** Provides complete constraints upfront and interacts efficiently.
- **Ambiguous user:** Initially omits key information and requires clarification.
- **Confused user:** Seeks guidance about the process and may misunderstand system responses.

D Example Trajectory

Conversation Step 1 — **User**

```
Hi, this is James Wilson. Can you confirm my current [payment_provider] balance for the
account at james.wilson@email.com? I'm seeing $450 on my end. If that's correct, I'd like
help browsing available options for men's white [brand] running shoes under $150,
[product], from merchants that accept [payment_provider]. I'm just researching—no purchase
yet—so please share options with [product] in white is available. Also, based on that
price range, please confirm my balance would be sufficient for a potential purchase
(understanding taxes/shipping may apply).
```

Conversation Step 2 — **Reasoning Agent**

```
Tool call(s):
call_account({"subquery": "Confirm current [payment_provider] balance for the account
associated with james.wilson@email.com. Provide exact amount and currency."})
```

Conversation Step 3 — **Tool Response**

```
{
  "success": true,
  "message": "Your current [payment_provider] balance is $450.00",
  "data": {
    "current_balance": 450.0,
    "currency": "USD"
  }
}
```

Conversation Step 4 — **Reasoning Agent**

```
Tool call(s):
call_commerce({"subquery": "Search for men's white [brand] running shoes priced under $150
from merchants that accept [payment_provider]. Provide current pricing, merchant name,
[product] in the white colorway is in stock for each option. Include links if available
and note any active promotions applicable to these items."})
```

Conversation Step 5 — **Tool Response**

```
{
  "success": true,
  "message": "[product] men's white [brand] running shoes under $150 that accept
  [payment_provider]. Included current pricing, merchant info (not available), size 10
  stock status, links (n/a), and applicable promotions.",
  "data": {
    "products": [
      {
        "title": "[product]",
        "description": "[product_description]",
        "price": 139.99,
...
```

Figure 6: Conversation flow across sequential steps, showing user turns, reasoning-agent tool calls, and tool responses. Brand names and trademarks have been redacted.

Conversation Step 5 (cont.) **Tool Response**


```
        "brand": "[brand]",
        "category": "Men's Running",
        "colorway": "White/Black",
        "rating": 4.6,
        "availability": "In Stock",
        "size_10_in_stock": true,
        "size_10_stock_count": 5,
        "merchant": {
          "name": "n/a",
          "payment_provider_accepted": true
        },
        "product_url": "n/a",
        "shipping": {
          "standard": "Free (3-5 business days) on orders over $50",
          "express": "$15.99 (1-2 business days)",
          "next_day": "$25.99 (order by 2 PM PT)"
        },
        "promotions": [
          "Free standard shipping on orders over $50",
          "[product] interest-free payments available for purchases $30-$1,500"
        ]
      },
      {
        "title": "[product]",
        "description": "[product_description]",
        "price": 69.99,
        "brand": "[brand]",
        "category": "Men's Running",
        "colorway": "White/Grey",
        "rating": 4.3,
        "availability": "In Stock",
        "size_10_in_stock": true,
        "size_10_stock_count": 7,
        "merchant": {
          "name": "n/a",
          "payment_provider_accepted": true
        },
        "product_url": "n/a",
        "shipping": {
          "standard": "Free (3-5 business days) on orders over $50",
          "express": "$15.99 (1-2 business days)",
          "next_day": "$25.99 (order by 2 PM PT)"
        },
        "promotions": [
          "Free standard shipping on orders over $50",
          "[product] interest-free payments available for purchases $30-$1,500"
        ]
      },
      {
        "title": "[product]",
        "description": "[product_description]",
        "price": 129.99,
...
```


Figure 6: Conversation flow across sequential steps, showing user turns, reasoning-agent tool calls, and tool responses. Brand names and trademarks have been redacted.

Conversation Step 5 (cont.) — **Tool Response**


```
        "brand": "[brand]",
        "category": "Men's Lifestyle/Running",
        "colorway": "White",
        "rating": 4.7,
        "availability": "In Stock",
        "size_10_in_stock": true,
        "size_10_stock_count": 4,
        "merchant": {
          "name": "n/a",
          "payment_provider_accepted": true
        },
        "product_url": "n/a",
        "shipping": {
          "standard": "Free (3-5 business days) on orders over $50",
          "express": "$15.99 (1-2 business days)",
          "next_day": "$25.99 (order by 2 PM PT)"
        },
        "promotions": [
          "Free standard shipping on orders over $50",
          "[product] interest-free payments available for purchases $30-$1,500"
        ]
      }
    ],
    "result_count": 3,
    "search_query": "Search for men's white [brand] running shoes priced under $150 from
    merchants that accept [payment_provider]. Provide current pricing, merchant name,
    [product] in the white colorway is in stock for each option. Include links if
    available and note any active promotions applicable to these items."
  }
}
```


Conversation Step 6 — **Reasoning Agent**


```
Tool call(s):
show_to_user({"subquery": "Hi James \u2014 here\u2019s what I found:\n\n1) Balance
confirmation\n\u2022 Your current [payment_provider] balance is $450.00 USD.\n\n2)
Men\u2019s white [brand] running shoes under $150 ([payment_provider] accepted; size 10 in
white available)\n\u2022 [product] (White/Black) \u2014 $139.99\n  - Size 10: In stock (5
left)\n  - Shipping (from listing): Free standard over $50; Express $15.99; Next-day
$25.99\n  - Promo noted: [product] available (for eligible purchases)\n\n\u2022 [product]
(White/Grey) \u2014 $69.99\n  - Size 10: In stock (7 left)\n  - Shipping (from listing):
Free standard over $50; Express $15.99; Next-day $25.99\n  - Promo noted: [product]
available (for eligible purchases)\n\n\u2022 [product] (White) \u2014 $129.99
(Lifestyle/Running)\n  - Size 10: In stock (4 left)\n  - Shipping (from listing): Free
standard over $50; Express $15.99; Next-day $25.99\n  - Promo noted: [product] available
(for eligible purchases)\n\nNotes:\n\u2022 The listings indicate that the merchants accept
[payment_provider]; specific merchant names and direct product links were not provided in
the results.\n\u2022 Stock and pricing can change quickly; let me know if you want me to
refresh availability or pull more options.\n\nBalance sufficiency\n\u2022 Based on these
prices, your $450.00 balance would be sufficient to cover any of the listed items before
taxes and shipping.\n\nWould you like me to refine the search (e.g., specific
...
```


Figure 6: Conversation flow across sequential steps, showing user turns, reasoning-agent tool calls, and tool responses. Brand names and trademarks have been redacted.

Conversation Step 6 (cont.) — **Reasoning Agent**

```
model/cushioning like [product] vs. [product], road vs. trail), check additional
retailers, or set up an [product] in white under $150?"})
```

Conversation Step 7 — **User**

```
<done>
```

Figure 6: Conversation flow across sequential steps, showing user turns, reasoning-agent tool calls, and tool responses. Brand names and trademarks have been redacted.