LANGUAGE AS COMPUTATION: SIMULATED FEEDBACK AND ON-POLICY GUIDANCE FOR INTERACTIVE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

We study the point at which language becomes not only a medium for supervising computation, but a medium for carrying out interaction. For interactive agents, language can serve not only as the environment interface but also as the medium of the learning interface. Because environmental signals and learning feedback are both expressed as text, they can be written directly into the interaction history and participate in subsequent action generation, without requiring access to teacher logits. We instantiate this idea as a language-mediated on-policy learning framework in which a fixed teacher produces action-conditioned environmental feedback, progress signals, and gradually fading hints for training a student policy with GRPO. We evaluate both the fidelity of the simulated feedback and the transfer of the resulting policy to executable environments. Across 5,868 replayed transitions, DeepSeek-V3.2 achieves 75.3% normalized observation exact match, compared with 67.9% for Qwen3-32B, while simulation fidelity decreases with interaction depth. A Qwen3-8B student trained through these interactions reaches 44.13% on BFCL Multi-Turn and 46.88% on Tau-Bench, exceeding matched executable-environment RL by 2.25 and 2.07 points, respectively; without hints, simulated-feedback training remains competitive on BFCL. The same framework transfers to GUI agents on WebArena, matching executable-environment training in three categories and improving CMS.These results suggest that language can unify environmental signals and learning feedback within the policy's computational context, where both directly shape subsequent actions.

1 INTRODUCTION

Training interactive agents is challenged by sparse-feedback exploration. Existing online distillation typically injects the teacher's policy distribution through the loss as an implicit supervision channel. This teacher knowledge remains at the parameter level and does not explicitly enter the student's interaction with the environment. Once the student deviates from the preferred trajectory during interaction, such implicit supervision cannot intervene or correct in time, making long-horizon trajectory learning sample-inefficient.

To overcome this bottleneck, we start from a key observation: **natural language is emerging as a computational medium**. Modern learning systems increasingly use text not only to represent inputs and outputs, but also to specify objectives, convey feedback, assign rewards, retain experience, and condition policy updates. For interactive agents, this trend extends to the environment boundary itself: actions and their consequences can be exchanged through a textual protocol. A tool call, an API response, a web operation, and the resulting state change can all be serialized into the same transcript from which the policy generates its next action. Language thus does more than describe an interaction after the fact; it can carry the intermediate signals that drive the interaction forward. This makes language a natural medium for teacher supervision to enter the student's rollout, rather than remaining confined to parameter updates.

This view suggests a different interface between a teacher and a learning agent. Instead of transferring action labels or token-level distributions only through a distillation objective, the teacher externalizes its knowledge as text and injects it into the student's runtime context during rollout. In this paper, we

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

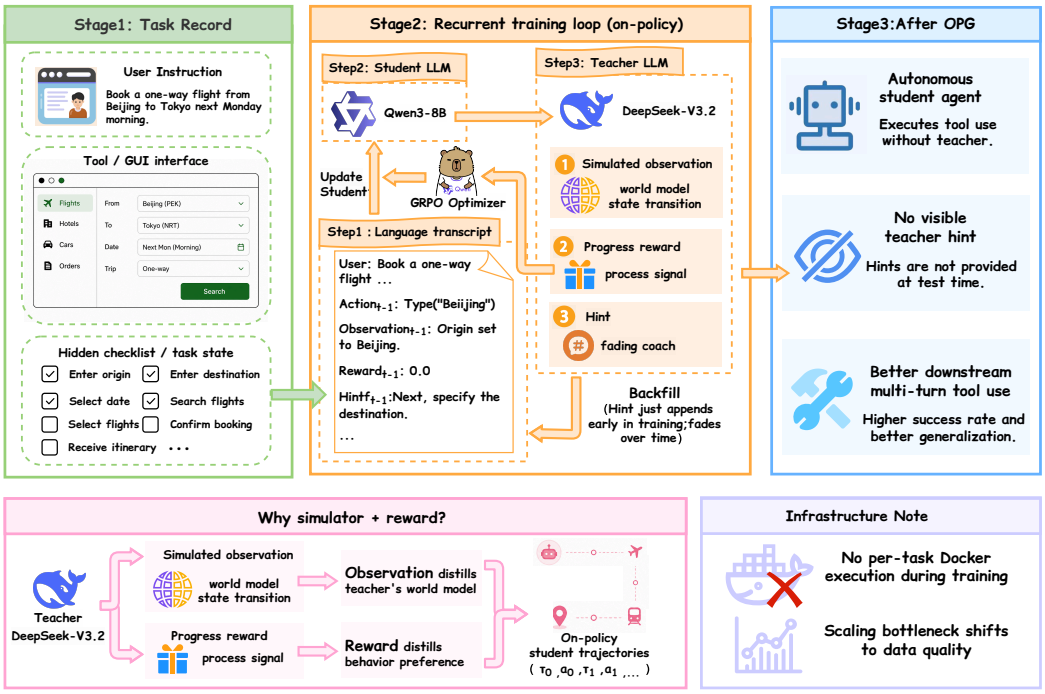

Figure 1: **Overview of Language-Mediate On-Policy Guidance(LM-OPG).** A privileged teacher reads the task, hidden checklist, and the student's action a_t, then writes a simulated observation, a progress reward for GRPO, and a fading natural-language hint into the student's transcript. The student is trained on these teacher-mediated rollouts and deployed without hints.

instantiate this interface with two student-facing signals: **action-conditioned observations**, which report the simulated consequences of the student's actions, and **hints**, which indicate what to attend to and consider next. To keep these signals consistent across turns, the teacher privately maintains a checklist-grounded progress state, whose final assessment is combined with the task-completion signal to construct the reward for reinforcement learning. The student-facing observations and hints can therefore correct deviations within the ongoing trajectory, while the resulting reward updates the policy across trajectories.

The teacher exists only during training. At deployment, its simulated observations are replaced by feedback from the executable environment, its private progress state and completion judgement are no longer needed, and its hints have been annealed to zero. The student therefore acts autonomously from deployment-consistent task and environment inputs.

Our contributions are threefold:

- We introduce a textual interface into the agent learning loop, where a black-box teacher injects observations and hints into the student's runtime context during rollout, while progress rewards remain internal to the teacher.
- We instantiate this interface as a learning guidance that combines action-conditioned observations, annealed on-policy hints, and teacher-internal progress rewards, yielding a unified training loop for tool-use and GUI agents.
- We conduct experiments to evaluate this framework: paired transition replay measures the fidelity, efficiency, and depth-dependent degradation of textual feedback, while BFCL Multi-Turn, Tau-Bench, and WebArena measure transfer to executable environments.

2 LANGUAGE-MEDIATED ON-POLICY GUIDANCE

Having motivated language as a computational medium, we now formalize the interaction setting and present our method. We consider an interactive agent that operates in a partially observable environment and produces actions conditioned on a textual context. Unlike prior online distillation, the teacher does not provide token-level distributions through an auxiliary loss. Instead, the teacher externalizes its knowledge as text injected into the student's context during on-policy rollouts. This shift raises three design questions: what to inject, when to inject, and how to remove it at deployment. This section answers these questions by introducing action-conditioned observations, progress rewards, and hints, together with an annealing schedule that removes teacher guidance before deployment.

2.1 INTERACTION AS TEXTUAL COMPUTATION

We assume a dataset $\mathcal{D}$ of task records (x, z). The public component x contains the user instruction and interface information, such as tool schemas or page descriptions. The privileged component z contains training-only, domain-specific supervision, such as the initial environment state and transition rules for tool use or a task checklist for web interaction. The student never observes z; its only computational input at turn t is the transcript h_t, from which it samples an action

$$a_t \sim \pi_\theta(\cdot \mid h_t), \qquad a_t \in \mathcal{A}, \tag{1}$$

where a_t denotes the action taken by the student at turn t from the action space $\mathcal{A}$.

In a standard executable environment $\mathcal{E}$, the action a_t produces a real observation o_{t+1} and extends the transcript as

$$o_{t+1} \sim \mathcal{E}(\cdot \mid x, h_t, a_t), \qquad h_{t+1} = h_t \oplus a_t \oplus o_{t+1}, \tag{2}$$

where o_{t+1} denotes the observation returned by the executable environment and $\oplus$ denotes concatenation into the interaction transcript.

Conventional online distillation changes θ through an auxiliary loss but, for a fixed rollout, does not add a signal to h_t. Because h_t is the student's sole input for computing a_{t+1}, any supervision intended to affect the next action within the current rollout must itself be written into the transcript.

2.2 TEACHER-WRITTEN TRANSCRIPTS

During training, a fixed teacher $\mathcal{C}_\phi$ replaces the executable transition with a language transition. Conditioned on the task, privileged information, student-visible transcript, current action, teacher-private progress state c_t, and hint strength s_k, it produces

$$(\hat{o}_{t+1}, \eta_t, c_{t+1}) \sim \mathcal{C}_\phi(\cdot \mid x, z, h_t, a_t, c_t; s_k). \tag{3}$$

The interaction therefore has two states. The transcript h_t is visible to the student and serves as its computation state. The progress state c_t is visible only to the teacher, initialized from the privileged record z, and carried across turns. It plays the role of a hidden environment state, but is maintained through the teacher's structured textual output.

The student transcript is updated by

$$h_{t+1} = h_t \oplus a_t \oplus \hat{o}_{t+1} \oplus b_t\, \eta_t, \qquad b_t \sim \mathrm{Bernoulli}(p_k). \tag{4}$$

Here, b_t determines whether the hint is exposed to the student at turn t, with p_k denoting the probability of exposing the hint at training stage k.

The two student-facing texts serve different interfaces. The simulated observation $\hat{o}_{t+1}$ is an environment-side signal whose target is the observation o_{t+1} that $\mathcal{E}$ would return. The hint η_t is a learning-side signal that carries teacher knowledge beyond the environment response by indicating what the student should attend to next. Both affect subsequent actions because both are written into h_{t+1}, whereas c_t remains hidden.

For horizon T, let

$$\tau = (h_0, a_0, \hat{o}_1, b_0, \eta_0, c_1, \ldots, h_T, c_T).$$

Its teacher-mediated rollout distribution is

$$P^{(k)}_{\theta,\phi}(\tau \mid x, z) = \prod_{t=0}^{T-1} \pi_\theta(a_t \mid h_t)\, \mathcal{C}_\phi(\hat{o}_{t+1}, \eta_t, c_{t+1} \mid x, z, h_t, a_t, c_t; s_k)\, p_k^{b_t}(1-p_k)^{1-b_t}. \quad (5)$$

Here *on-policy* refers to the teacher-mediated interaction distribution: the teacher is fixed, but $P^{(k)}_{\theta,\phi}$ changes with θ, so supervision follows states reached by the current student rather than a static demonstration set.

2.3 Removing the Teacher

The teacher is useful for exploration but must not become a deployment-time dependency. During training, we gradually remove the teacher's explicit guidance by decaying both hint strength and visibility:

$$\rho_K(k) = \max\left(0, 1 - \frac{k}{K}\right), \qquad s_k = s_{\min} + (s_{\max} - s_{\min})\rho_{K_s}(k), \quad (6)$$

$$p_k = p_{\min} + (p_{\max} - p_{\min})\rho_{K_p}(k). \quad (7)$$

Early in training, the teacher provides explicit guidance through hints. As training proceeds, both the hint strength s_k and the probability p_k of exposing hints gradually decrease, eventually reaching zero. The policy therefore gradually shifts from relying on explicit teacher guidance to relying only on the task and observation transcript. At deployment, the executable environment replaces the simulated observations, $\hat{o}_{t+1} \to o_{t+1}$. The private progress state c_t and training reward $R(\tau)$ are also no longer used. Thus, training and deployment retain the same action–observation transcript format. The schedule parameters are provided in Appendix B.

2.4 Implementation

We implement the method in `verl` with a custom interaction class exposing the same `start_interaction`/`generate_response` contract as an executable environment. Each parsed non-terminal student action triggers at most one OpenAI-compatible teacher call. The structured JSON response carries an `observation`, a teacher-private progress field (a checklist state or scalar assessment), and a `suggestion` corresponding to η_t. The teacher response directly supplies the textual transition, so RL rollouts require no task-specific backend, browser, sandbox, or Docker execution. Moving to a new interaction domain changes the task record and teacher prompt rather than the training loop; further schema and deployment details appear in Appendix D.

3 Experimental setup

Dataset and baselines. For tool-use experiments, we use EnvScaler (Song et al., 2026), which provides 191 executable tool environments and roughly 7K task scenarios with tool schemas, hidden environment states, interaction rules, and state-checking reward functions. All student training regimes share the same task pool and Qwen3-8B initialization. To compare learning from fixed demonstrations with online interaction, we include two supervised fine-tuning baselines and an executable-environment RL baseline. The SFT baselines use either EnvScaler's provided expert trajectories or trajectories generated offline by DeepSeek-V3.2 through executable-environment rollouts, with matched trajectory budgets.

Evaluation and metrics. DeepSeek-V3.2 serves as the teacher for both teacher-trace SFT and our frozen online simulator. We evaluate on BFCL-v4 (Patil et al., 2025) and τ-bench (Yao et al., 2025), reporting BFCL Multi-Turn accuracy and τ-bench pass-at-1. All downstream evaluations use executable environments without teacher-simulated observations or hints. A second instantiation on GUI agents is summarized in Sec. 5.

4 Results

As shown in Table 1, our LM-OPG achieves 44.13% overall accuracy on BFCL Multi-Turn. Compared with online policy distillation, LM-OPG improves accuracy by 3.50, 0.00, 3.00, and 6.50

Table 1: Direct BFCL Multi-Turn comparison. Accuracy is reported in percent. The five Qwen3-8B rows are matched on student initialization and EnvScaler task pool (Song et al., 2026), isolating expert-trace SFT, off-policy teacher-trace SFT, real-env RL, OPD, and LM-OPG.

Model	Base	Miss-Func	Miss-Param	Long-Context	Overall
Advanced models					
GPT-4.1	46.00	37.50	32.00	44.50	40.00
Qwen3-235B-Thinking-2507	60.00	35.00	34.00	54.00	45.75
Qwen3-235B-Instruct-2507	58.00	33.00	27.50	50.50	42.25
Kimi-K2-Instruct-0905	57.50	35.00	42.00	49.00	45.88
Qwen3-8B (Thinking)	32.00	33.50	22.00	28.00	28.88
+ EnvScaler SFT	47.00	33.00	29.50	38.50	37.00
+ DeepSeek-V3.2 traces (off-policy SFT)	48.00	25.50	26.00	41.00	35.13
+ EnvScaler RL (real env)	55.50	36.00	35.00	41.00	41.88
+ EnvScaler RL (OPD)	50.00	43.50	31.00	39.00	40.88
+ EnvScaler RL (LM-OPG)	53.50	43.50	34.00	45.50	44.13

Table 2: τ-bench pass[1] (%). reported by Song et al. (2026), Table 4 (GPT-4.1 user simulator, mean of 3 trials; 50 airline and 115 retail tasks, with retail values $n/345$). is our legacy one-off evaluation, retained only as context: its task/trial counts, decoding, simulator identity, and item outputs were not retained, and 50.88 is not a multiple of $1/115$. Overall is the unweighted mean of the two domain percentages; no inferential comparison is made.

Model	Airline	Retail	Overall
Qwen3-8B (thinking)	30.00	46.38	38.19
+ EnvScaler SFT	34.00	48.70	41.35
+ EnvScaler RL (real env)	36.00	53.62	44.81
+ LM-OPG	42.00	50.88	46.44

percentage points on Base, Missing-Function, Missing-Parameter, and Long-Context, respectively, corresponding to a 3.25-point gain overall. In contrast, offline policy SFT on the same teacher-generated traces achieves only 35.13%. We further evaluate LM-OPG on τ-bench. As shown in Table 2, LM-OPG achieves a point estimate of 46.44% overall, with 42.00% on Airline and 50.88% on Retail. By comparison, RL with real-environment interaction achieves 44.81% overall, 1.63 percentage points lower than LM-OPG. Overall, these results show that LM-OPG benefits from online teacher supervision.

5 Analysis

5.1 Training dynamics.

Figure 2 shows stable RL training over the first 200 GRPO steps: reward changes from 0.628 in steps 0–30 to 0.636 in steps 170–200, while KL, entropy, length, and turns remain stable. On real-Docker BFCL evaluation, the RL-trained student exceeds the real-env-RL student by 2.25 points despite using teacher-mediated rollouts, indicating that action-conditioned teacher observations carry information that scalar execution rewards miss. We additionally train GUI agents with AgentGym RL on text-based WebShop and ALFWorld tasks, using either executable environments or the teacher simulator, and evaluate transfer on WebArena (Zhou et al., 2024). Figure 3 shows stable training dynamics through GRPO step 500. On WebArena, the student achieves competitive performance with the executable-environment baseline and substantially improves over the zero-shot baseline. Together, these results suggest that teacher-mediated textual feedback can provide effective training signals beyond scalar rewards and transfer across interaction settings.

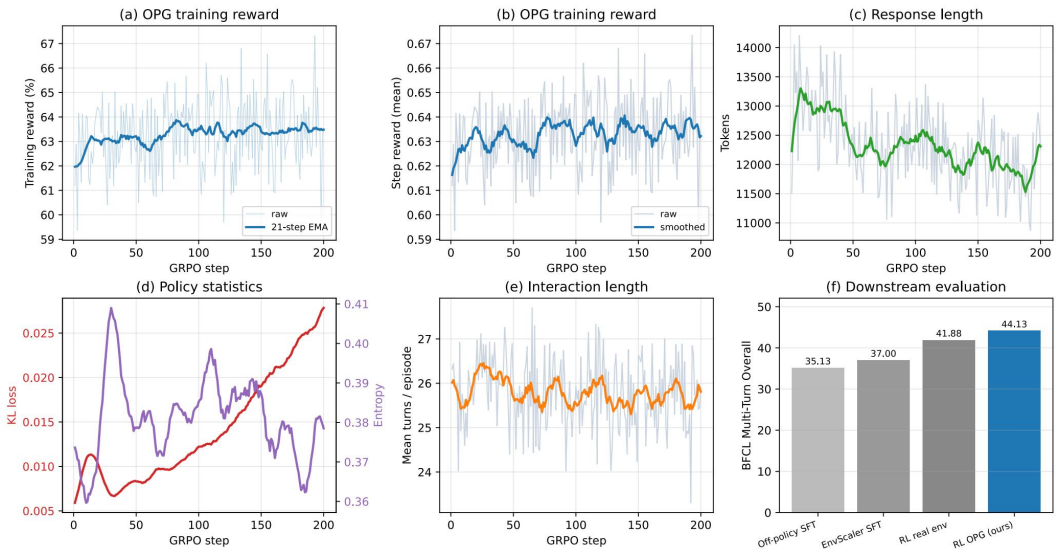

Figure 2: Tool-agent training dynamics and downstream comparison on the EnvScaler task pool. Panels (a)–(e) show reward, length, policy statistics, and turns over the first 200 GRPO steps; panel (f) reports BFCL Multi-Turn Overall for the matched regimes in Table 1.

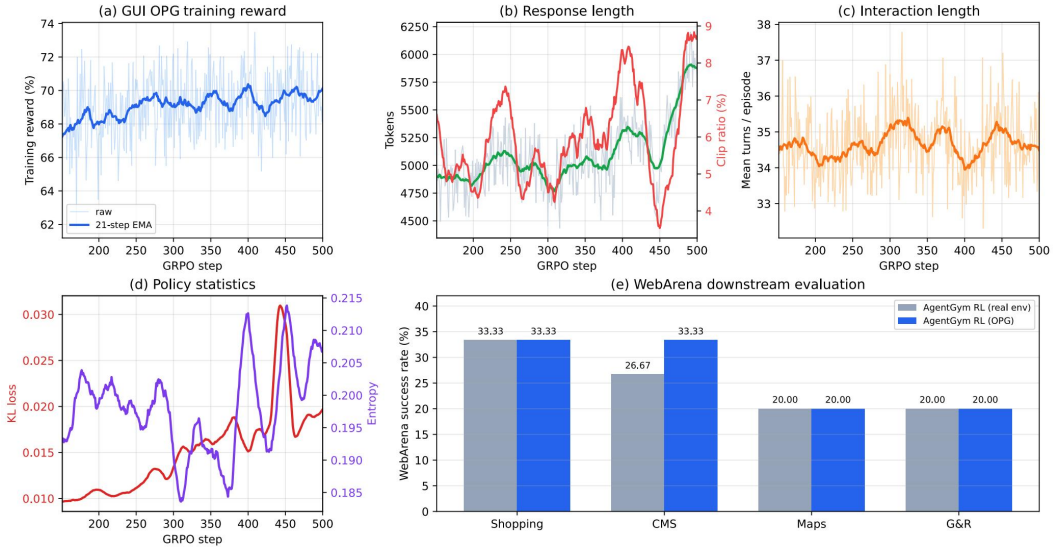

Figure 3: GUI-agent training dynamics up to GRPO step 500 and downstream WebArena comparison against matched real-environment AgentGym RL.

5.2 Similarity of teacher-simulated environments.

Because LM-OPG replaces executable environment transitions with teacher-generated textual observations, a central question is whether the teacher can faithfully simulate the consequences of student actions. We replay 5,868 tool-call transitions from 300 EnvScaler task scenarios and compare teacher-simulated observations with those returned by the executable environment under the same task record, state, tool schema, and interaction history. As shown in Figure 4, the simulated observations remain highly similar to the executable-environment outputs throughout the interaction. In particular, semantic similarity remains consistently high, while surface-form similarity is slightly lower and exact matching exhibits greater variation. The gap between these metrics suggests that most discrepancies arise from differences in wording or formatting rather than changes in the under-

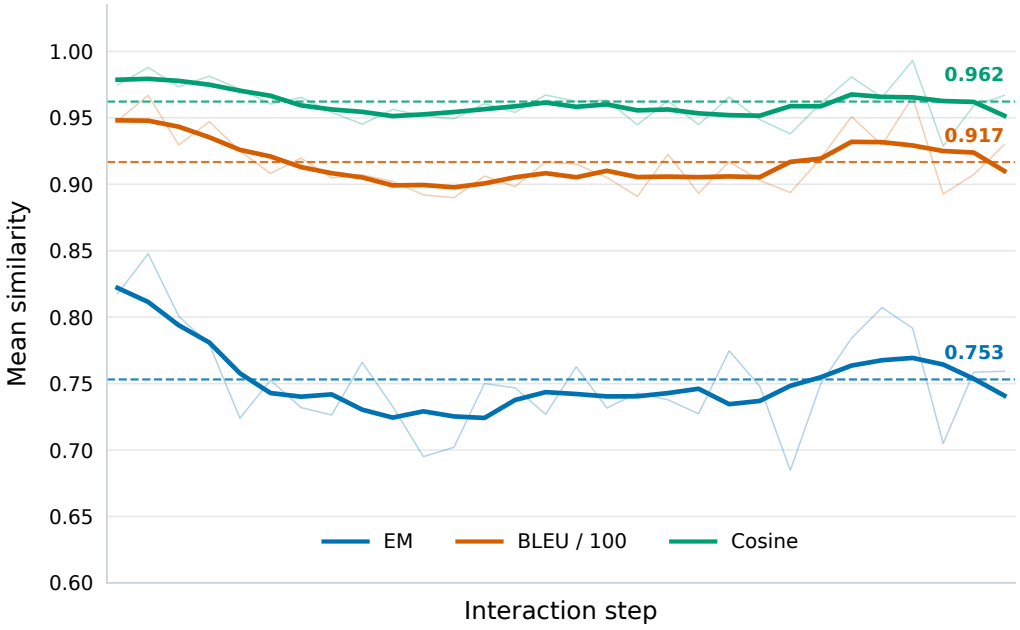

Figure 4: Teacher–environment similarity across interaction depth. The three metrics measure the similarity between teacher-simulated and executable-environment observations. Semantic similarity remains consistently high across interaction steps, while surface-form differences accumulate mildly with depth. The relatively lower exact-match scores reflect local differences in wording or formatting rather than systematic semantic divergence.

lying state semantics. Moreover, semantic similarity shows little degradation with interaction depth, although surface-level differences mildly accumulate over longer interactions. These results indicate that the teacher can preserve task-relevant semantics in environment feedback through a textual interface without executing the original backend. For training, exact reproduction of environment outputs is therefore unnecessary as long as the simulated observations remain sufficiently faithful to state changes relevant to task progress. This supports the premise of simulation-based LM-OPG: a language-model teacher can provide useful, action-conditioned feedback through the same textual interface consumed by the student, while the remaining simulation mismatch represents a limitation in fidelity rather than a semantic failure.

5.3 Ablation of teacher hints.

The teacher observation and reward channels provide persistent process-level supervision, while hints offer additional exploration support early in training. As shown in Figure 5, the full LM-OPG recipe achieves 44.13% on BFCL Multi-Turn Overall. Removing the hint reduces performance to 40.25%, indicating that hints provide additional exploration support beyond the persistent observation and reward channels. Changing the hint schedule also affects performance: both the constant-hint and cosine-schedule variants degrade. These results suggest that how auxiliary guidance is delivered during training matters, while the observation and reward channels remain the primary sources of process-level supervision.

5.4 Effect of progress rewards and observations.

The progress reward provides dense process-level supervision by measuring the student's progress against a task checklist hidden from the policy. Removing this signal reduces BFCL Multi-Turn Overall from 44.13% to 38.38%, the largest degradation among the supervision ablations. This indicates that action-conditioned observations alone are insufficient to fully guide long-horizon

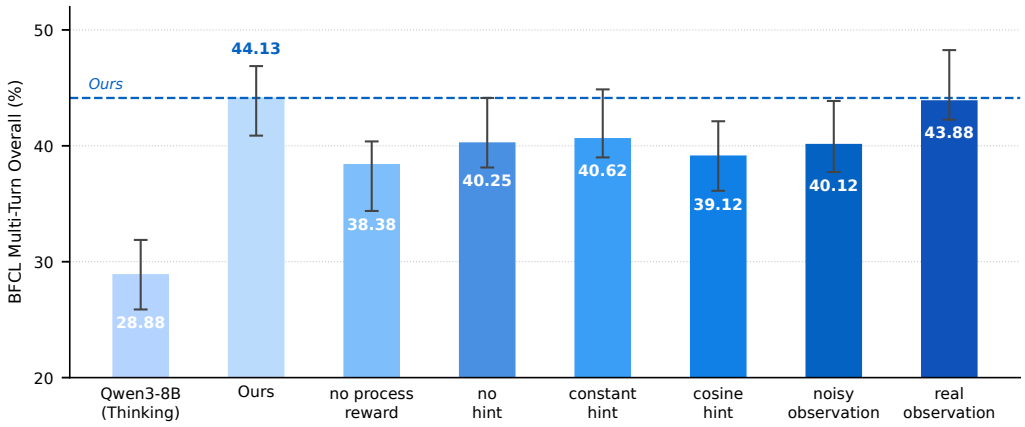

Figure 5: Ablation of supervision components on BFCL Multi-Turn Overall. All variants use the same EnvScaler task pool and Qwen3-8B student initialization.

exploration: the observation describes how the environment changes in response to the student's action, while the progress reward indicates whether that change moves the student closer to the task objective. We further examine the effect of observation quality. Replacing simulated observations with real observations yields 43.88%, close to the full LM-OPG result, while injecting noise into the observations reduces performance to 40.12%. This result suggests that the benefit of the observation channel depends on preserving task-relevant information rather than merely providing additional textual context. Together, these results show that the three supervision components play complementary roles: observations convey action-conditioned state changes, progress rewards provide directional feedback toward the task objective, and hints provide additional exploration guidance.

5.5 Training-Time Supervision Cost.

Unlike teacher-trace SFT, which requires executable-environment rollouts to generate reusable teacher trajectories before optimization, LM-OPG obtains training-time transitions directly from teacher-generated textual simulation. This enables an execution-free training loop: student rollouts no longer require per-step access to task-specific backends such as browsers, sandboxes, or Docker environments, allowing the same training infrastructure to operate across environment families as long as task records specify the interaction interface and supervision needed for the teacher to simulate observations and judge progress. The trade-off is online teacher inference. In our setting, each successfully parsed non-terminal student action triggers at most one teacher query; with 640 trajectories per training step and 5–10 interaction turns, this corresponds to at most 3.2K–6.4K teacher queries, or approximately 11–21M teacher input tokens, per training step. Thus, LM-OPG does not claim universally lower computational cost; rather, it shifts transition-generation cost from executable-environment interaction to online teacher inference. At the same time, because transitions are generated online from the student's own trajectory, supervision remains conditioned on states actually visited by the student, providing action-conditioned observations, progress feedback, and optional hints that are unavailable from a fixed teacher trajectory.

5.6 Broader Impacts.

The positive impact of LM-OPG is practical: it can lower the engineering cost of training interactive agents and make process feedback more inspectable because observations, rewards, and hints are written in natural language. The same property also creates risks. A stronger agent trained through teacher-mediated interaction could be used for harmful automation if connected to sensitive tools, and teacher feedback can inherit bias, reveal private task information, or encourage behavior that satisfies a checklist while missing user intent. These risks argue for conservative deployment boundaries,

benchmark-time separation between privileged supervision and policy inputs, logging of teacher responses during training, and downstream evaluation in real environments before release.

6 Related work

6.1 World models for planning and agent training

Simulation has been used for both inference-time planning and policy training. WebDreamer (Gu et al., 2025) and WMA (Chae et al., 2025) use simulated outcomes to guide action selection, while Simia-SFT/RL (Li et al., 2025), DreamGym (Chen et al., 2026), and CM2 (Zhang et al., 2026) incorporate simulation into policy learning. In contrast, LM-OPG uses the teacher as an online, in-context transition simulator: each student action receives a simulated observation, progress reward, and optional hint conditioned on the student-visited state. Unlike offline trajectory synthesis or inference-time candidate search, these signals directly shape subsequent policy actions during training. Other approaches instead rely on executable environments or offline simulation and cached evaluation (Song et al., 2026; Wang et al., 2026; Lyu et al., 2026; Sun et al., 2026; Guo et al., 2024).

6.2 Interactive agents and evaluation.

Language-agent research studies models that interleave reasoning, actions, observations, and final answers in an interactive transcript. ReAct established this reasoning–acting paradigm (Yao et al., 2023), while Toolformer, Gorilla, API-Bank, and ToolLLM/ToolBench explored learning to invoke and compose external tools (Schick et al., 2023; Patil et al., 2024; Li et al., 2023; Qin et al., 2024). Web and GUI agents extend the same interaction loop to rendered interfaces, with benchmarks such as WebShop, Mind2Web, WebArena, OSWorld, and AgentGym covering shopping, web navigation, and computer-use tasks (Yao et al., 2022; Deng et al., 2023; Zhou et al., 2024; Xie et al., 2024; Xi et al., 2025). Recent evaluations including BFCL and Tau-Bench further emphasize multi-turn dependencies, missing information, stateful environments, and interaction reliability (Patil et al., 2025; Yao et al., 2025). EnvScaler provides executable tool-interaction environments with state-based rewards for SFT and RL (Song et al., 2026). Our LM-OPG retains this unified action–observation interface, but replaces training-time execution in task-specific environments with teacher-generated observations, progress signals, and optional hints, allowing the same supervision mechanism to support both tool-use and GUI interaction.

7 Limitations

LM-OPG deliberately couples the transition model and the reward source: the same teacher that writes the simulated observation also judges checklist progress. This makes training simple, but it also means that systematic teacher mistakes can be reinforced unless the response schema, checklist, and evaluation harness catch them. We reduce this risk with strict checklist rewards, JSON validation, and real-environment evaluation, yet the current draft does not include a full audit of teacher calibration or adversarial prompt failures. The experiments also report single evaluation runs for each benchmark slice, so the reported margins should be read as evidence of a promising training channel rather than a complete statistical characterization. Finally, the GUI study is smaller than the tool-use study and should be expanded to more websites, longer horizons, and multimodal state renderings before making broad claims about computer-use agents.

8 Conclusion

We presented a language-mediated on-policy learning guidance that treats language as both an environment interface and a learning interface. By injecting action-conditioned observations and hints into the student's runtime context, the teacher can provide timely guidance without exposing policy logits. Experiments show that simulated feedback preserves task-relevant semantics and enables effective transfer to executable environments across tool-use and GUI tasks. These results suggest that language can serve as a unified medium through which environment feedback and learning supervision directly shape interactive behavior.

REPRODUCIBILITY STATEMENT

Reproducing the reported results requires implementing the language-mediated on-policy training loop on top of a GRPO framework such as `verl`, with Qwen3-8B as the student policy and DeepSeek-V3.2 as the frozen teacher. For the tool-use experiments, the training setup should support the EnvScaler task records, tool schemas, and executable environments, while replacing environment execution with a teacher-mediated interaction interface that generates action-conditioned observations, progress rewards, and annealed hints. The same GRPO configuration, task pool, student initialization, and evaluation protocol should be used across the matched training regimes. Reproduction of the GUI experiments additionally requires an AgentGym-compatible interaction loop and the corresponding WebShop and ALFWorld task records, with WebArena used for downstream evaluation. The teacher prompts should provide the task, privileged supervision, interaction history, and student action, and return the textual observation and supervision signals specified in Sec. 2. Finally, the reported comparisons require both teacher-simulated training and executable-environment baselines, followed by evaluation in the corresponding real environments without teacher hints.

AI USE STATEMENT

In this work, we used generative AI tools for generating synthetic data sets, designing or providing feedback on research methodology or experiments, assisting with translation, and cleaning and reformatting datasets. We have not used generative AI tools for helping develop theoretical models or conceptual frameworks, providing critical ingredients for proving mathematical claims, proposing or refining hypotheses, assisting in the writing of proofs, implementing methods, or interpreting results, and formulating mathematical claims are not applicable to this work. Additionally, we used generative AI tools for drafting parts of a research paper, summarizing or analysing existing literature, sourcing/searching for information, editing a research paper to improve readability, identifying relevant literature, and formatting references. We have reviewed all AI-assisted work. The information sourced by AI and the sections drafted with AI assistance were verified and tested for correctness and authenticity by at least 3 authors; unreasonable content was corrected and revised, and model-generated data was reviewed and analysed. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Hyungjoo Chae, Namyoung Kim, Kai Tzu-iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. Web agents with world models: Learning and leveraging environment dynamics in web navigation. In *International Conference on Learning Representations (ICLR)*, 2025. URL `https://proceedings.iclr.cc/paper_files/paper/2025/hash/a00548031e4647b13042c97c922fadf1-Abstract-Conference.html`. arXiv:2410.13232.

Zhaorun Chen, Zhuokai Zhao, Kai Zhang, Bo Liu, Qi Qi, Yifan Wu, Tarun Kalluri, Sara Cao, Yuanhao Xiong, Haibo Tong, Huaxiu Yao, Hengduo Li, Jiacheng Zhu, Xian Li, Dawn Song, Bo Li, Jason Weston, and Dat Huynh. Scaling agent learning via experience synthesis. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2511.03773.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023. arXiv:2306.06070.

Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. Is your LLM secretly a world model of the internet? Model-Based planning for web agents. *Transactions on Machine Learning Research*, 2025. arXiv:2411.06559.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. StableToolBench: Towards stable Large-Scale benchmarking on tool learning of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024. doi: 10.18653/v1/2024.findings-acl.664. arXiv:2403.07714.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for Tool-Augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. doi: 10.18653/v1/2023.emnlp-main.187. arXiv:2304.08244.

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. arXiv preprint arXiv:2511.01824, 2025.

Yuanjie Lyu, Chengyu Wang, Lei Shen, Jun Huang, and Tong Xu. Mock worlds, real skills: Building small agentic language models with synthetic tasks, simulated environments, and Rubric-Based rewards. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2026. doi: 10.18653/v1/2026.acl-long.570. arXiv:2601.22511.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2305.15334.

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 *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 2025.

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 *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2307.16789.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2302.04761.

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 *Findings of the Association for Computational Linguistics: ACL 2026*, 2026. doi: 10.18653/v1/2026.findings-acl.407. arXiv:2601.05808.

Shuang Sun, Huatong Song, Lisheng Huang, Jinhao Jiang, Ran Le, Zhihao Lv, Zongchao Chen, Yiwen Hu, Wenyang Luo, Wayne Xin Zhao, Yang Song, Hongteng Xu, Tao Zhang, and Ji-Rong Wen. SWE-World: Building software engineering agents in Docker-Free environments. arXiv preprint arXiv:2602.03419, 2026.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent world model: Infinity synthetic environments for agentic reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2602.10090.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. AgentGym: Evaluating and training large language model-based agents across diverse environments. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025. arXiv:2406.04151.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Jing Hua Toh, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for Open-Ended tasks in real computer environments. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2024. arXiv:2404.07972.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable Real-World web interaction with grounded language agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2207.01206.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.03629.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for Tool-Agent-User interaction in Real-World domains. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2406.12045.

Zhen Zhang, Kaiqiang Song, Xun Wang, Yebowen Hu, Weixiang Yan, Chenyang Zhao, Henry Peng Zou, Haoyun Deng, Sathish Reddy Indurthi, Shujian Liu, Simin Ma, Xiaoyang Wang, Xin Eric Wang, and Song Wang. CM2: Reinforcement learning with checklist rewards for Multi-Turn and Multi-Step agentic tool use. arXiv preprint arXiv:2602.12268, 2026.

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 *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2307.13854.

A NOTATION

Symbol	Meaning
$(x, z) \sim \mathcal{D}$	Task record: public prompt/interface x; privileged supervision z (tool use: initial state s_0 and environment rules; web: checklist G with M items; v3 records additionally carry two demonstrations)
h_t; $h_{t-w:t}$	Transcript before step t; its last w=10 successfully simulated action–observation pairs as shown to the teacher (observations truncated to 500 and arguments to 200 characters)
a_t; $\psi(a_t)$	Student action at step t; JSON schema of the tool it invokes
π_θ, π_{ref}, $\pi_{\theta_{\text{old}}}$	Student policy; its initialization (KL reference); rollout snapshot
$\mathcal{C}_\phi$	Fixed black-box teacher LLM (environment, judge, coach)
s_t, Δ_t	Explicit JSON environment state (tool use); RFC 6902 patch emitted by the teacher
$\hat{o}_t$; o_t; η_t	Teacher-simulated observation; executable-environment observation; teacher hint for action a_t
c_t; χ_T	Teacher-private progress state carried across turns; terminal task-completion signal
k; s_k; p_k; b_t	Trainer step; prompted hint strength; hint visibility probability; per-turn visibility draw
τ; $R(\tau)$	Teacher-mediated trajectory; trajectory-level score constructed from (c_T, χ_T)
N, β, ϵ_{clip}, c	GRPO group size, KL coefficient, clipping range, dual-clip constant

B HINT-SCHEDULE PARAMETERS AND IMPLEMENTATION

The six hyperparameters specify two separate schedules: the requested semantic specificity of a hint, and its probability of entering the student's context. For the requested notation, $s_{\min} \equiv \sigma_{\min}$, $s_{\max} \equiv \sigma_{\max}$, and $K_s \equiv K_\sigma$ in Eq. (7) and Eq. (6). These s-subscripts refer to hint strength, not the environment state s_t. The time index k counts trainer updates, not interaction turns, tokens, or teacher calls.

Table 3: Configured hint-schedule parameters. These are configuration values, not evidence that the legacy runs actually followed the schedule.

Parameter	Meaning / configuration field	Tool use	Web
$s_{\max}$	Initial requested hint strength; `initial_strength`	4	4
$s_{\min}$	Final requested strength; `final_strength`	1	1
$p_{\max}$	Initial inclusion probability; `initial_include_prob`	1	1
$p_{\min}$	Final inclusion probability; `final_include_prob`	0	0
K_s	Updates until strength reaches $s_{\min}$; `decay_steps`	1000	300
K_p	Updates until inclusion reaches $p_{\min}$; `decay_steps_prob`	2000	600

Strength versus visibility. Strength is an integer prompt instruction on the teacher's 1–5 scale: larger values request more specific guidance. It is neither a token budget nor a measured guarantee of the returned text's semantic strength. With positive K_s, the implementation linearly interpolates from $s_{\max}$ to $s_{\min}$, uses Python's round-half-to-even, and enforces a lower bound of 1. Separately, p_k is linearly interpolated between its endpoints over K_p updates and used for a Bernoulli inclusion draw. A hint can enter the transcript only when suggestions are enabled and the teacher supplies a non-empty suggestion on an eligible interaction turn. Thus $p_k = 1$ does not imply that failed or bypassed teacher calls produce a hint; $p_k = 0$ suppresses hints even though the requested strength may remain 1.

Horizon and actual exposure. At web update $k = 200$, the configured schedule gives $(\sigma_k, p_k) = (2, 2/3)$; at $k = 300$ it gives $(1, 1/2)$, and hints disappear only at $k \geq 600$. In tool use, update 250 gives $(3, 0.875)$, not fully faded training. Non-positive decay horizons are special disabling cases

in the implementation, not the endpoint of the above interpolation: strength stays at its initial value when $K_s \leq 0$, while the inclusion check returns true when $K_p \leq 0$. The configurations in Table 3 use positive horizons.

C GRPO OBJECTIVE

For each task (x, z) the rollout worker samples N trajectories $\{\tau_i\}_{i=1}^{N}$ from the current policy and the fixed teacher. With $R_i = R(\tau_i)$, $\bar{R} = N^{-1} \sum_i R_i$, and group standard deviation σ_R, the advantage is

$$A_i = \frac{R_i - \bar{R}}{\sigma_R + \epsilon}, \qquad \epsilon = 10^{-6}. \tag{8}$$

For every student-generated token $y_{i,\ell}$ (teacher-written tokens are masked out), with $h_{i,\ell}$ its context under the stored rollout,

$$q_{i,\ell}(\theta) = \frac{\pi_\theta(y_{i,\ell} \mid h_{i,\ell})}{\pi_{\theta_{\text{old}}}(y_{i,\ell} \mid h_{i,\ell})}, \qquad \tilde{q}_{i,\ell}(\theta) = \text{clip}\big(q_{i,\ell}(\theta), 1 - \epsilon_{\text{clip}}, 1 + \epsilon_{\text{clip}}\big), \tag{9}$$

and the maximized objective is

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}_{i,\ell}\Big[\min\big(q_{i,\ell}(\theta)A_i,\ \tilde{q}_{i,\ell}(\theta)A_i\big) - \beta\, D_{\text{KL}}\big(\pi_\theta(\cdot \mid h_{i,\ell}) \,\|\, \pi_{\text{ref}}(\cdot \mid h_{i,\ell})\big)\Big], \tag{10}$$

where $\mathbb{E}_{i,\ell}$ is the *token mean* over all unmasked tokens of the batch (the `verl` default), the KL term is the low-variance estimator with β=0.001, and π_{ref} is the initialization. The trainer minimizes the negative of Eq. (10). Two implementation details of `verl` are not shown: the loss for negative advantages is additionally bounded by $c\,|A_i|$ with c=3 (dual-clip PPO), and the KL estimator is clamped to $[-10, 10]$. When the PPO mini-batch equals the rollout batch and one epoch is used , there is exactly one optimizer step per rollout batch, $q_{i,\ell} \equiv 1$, and neither clip is active.

D ADDITIONAL IMPLEMENTATION DETAILS

Teacher response schema. Each OPD teacher call is constrained to a compact structured response. For checklist-labeled tasks, the teacher returns an `observation` string, a binary `checklist_satisfied` vector, and a `suggestion` string. For tasks without checklist labels, the vector is replaced by a clipped scalar progress score. The rollout worker rejects malformed responses, retries with the same transcript when parsing fails, and stores the validated response alongside the sampled action so that the GRPO update consumes exactly the transcript that was used during rollout.

Deployment interface. Teacher hints and simulated observations are training-time signals only. During evaluation on BFCL Multi-Turn, Tau-Bench, and WebArena, the student is connected back to the benchmark harness and receives the real observations provided by that harness. This separation is important: it means the reported scores measure whether teacher-mediated rollouts improve the policy, not whether the policy can exploit a teacher at test time.

Checklist aggregation. For checklist tasks, the reward manager treats completion as monotone within an episode: once the teacher marks a checklist item as satisfied, later turns can preserve that progress unless the task semantics explicitly invalidate it. The trajectory score in Eq. **??** is computed from the final validated progress state together with the terminal completion signal. This keeps the reward sparse but stable and avoids rewarding isolated teacher scores that are not grounded in the accumulated transcript.

E TEACHER PROMPT TEMPLATES

TOOL USE (DEFAULT TEMPLATE; SCALAR RUBRIC REWARD; EXPLICIT STATE)

```
You are an environment simulator for training an AI agent. You play
    three roles simultaneously:
```

```
1. **Environment**: Given the agent's action, return a realistic
    observation (what the tool would return).
2. **Reward Model**: Score the agent's action quality (how much it
    progresses toward the goal).
3. **Guide**: Provide a brief directional hint (without revealing the
    answer).

## Task Description
{task_description}

## Available Tools
{available_tools}

## Environment Rules and Summary
{environment_summary}

## Complete Executable Environment State Immediately Before This Action
{complete_environment_state}

## Interaction History (recent {history_window} steps)
{history}

## Agent's Current Action (Step {step_count}/{max_steps})
Tool: {action_name}
Arguments: {action_arguments}

## Your Response
Return a JSON object with exactly these fields:
{
  "observation": <the exact JSON-compatible tool response>,
  "state_patch": <RFC 6902 JSON Patch transforming the supplied complete
    state into the next state; [] if unchanged>,
  "reward": <float between 0.0 and 1.0, where 1.0 = optimal action, 0.5
    = reasonable, 0.0 = useless/harmful>,
  "suggestion": "<1-2 sentence directional hint at strength level
    {suggestion_strength}/5, DO NOT reveal the answer>"
}

Scoring guidelines:
- 1.0: Optimal action directly advancing toward the goal
- 0.7: Good action with clear progress
- 0.5: Reasonable but not the most efficient
- 0.3: Suboptimal, partially useful
- 0.0: Useless or counterproductive (repeated action, wrong direction)
```

WEB (CHECKLIST TEMPLATE; LEGACY AND CONTROLLED REWARD CONTRACTS; NO EXPLICIT STATE)

```
You are simulating a text-based GUI environment for training an AI agent.
The agent interacts with either a web shopping interface (WebShop) or a
    household
simulation (ALFWorld). You play three roles in one response:
  1. Environment Simulator -- given the agent's action, return a
    realistic text observation
  2. Reward Model -- judge which ground-truth checklist items are
    satisfied
  3. Guide -- optional brief hint at strength {suggestion_strength}/5

## Task
{task_description}

## Environment
{environment_summary}
```

```
## Ground-Truth Checklist (INTERNAL -- NEVER reveal to the agent, used
    only for your reward judgement)
{ground_truth_reference}

## Expert Demonstrations (similar tasks -- use as reference for
    realistic state transitions)
{few_shot_demos}

## Interaction History (recent {history_window} steps)
{history}

## Agent's Current Action (Step {step_count}/{max_steps})
Tool: {action_name}
Arguments: {action_arguments}

## Response Rules
- observation: what the REAL environment would return. For WebShop,
    return product
  listings separated by [SEP] with plausible product IDs. For ALFWorld,
    return
  "On the [location], you see [object list]." Be consistent with
    previous observations
  and with the ground-truth checklist (do not contradict the target
    object/location).
- checklist_satisfied: a list of 1-based indices of checklist items that
    are now
  satisfied by the CUMULATIVE trajectory so far (including this action).
    Be strict:
  an item is only satisfied when you can point to concrete evidence in
    the history.
- suggestion: 1-2 sentences directional hint at strength
    {suggestion_strength}/5,
  NEVER reveal the ground truth directly.

Return JSON exactly:
{
  "observation": "<realistic environment observation>",
  "checklist_satisfied": [<list of 1-based int indices>],
  "suggestion": "<brief hint>"
}
```