RETROSPECTIVE PROGRESS-AWARE SELF-REFINEMENT FOR LLM AGENT TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon LLM-based agents receive rich environmental observations during interaction, yet outcome rewards provide limited explicit supervision about how individual actions advance task completion. We investigate whether agents can turn this interaction evidence into useful training signals through retrospective progress assessment. A WebShop pilot shows that direct progress prompting reduces task success, whereas hindsight-annotated demonstrations improve it. We introduce **RePro**, **Re**trospective **Pro**gress-Aware Training, with a *forward-then-reflect* rollout: the agent estimates progress while acting, then reassesses each step using the completed trajectory and outcome. After warmup with externally generated demonstrations, policy optimization combines self-generated progress differences, online-retrospective alignment, and format rewards with environment feedback, requiring neither a separate process reward model nor ongoing teacher annotation. Experiments on WebShop, ALFWorld, and Sokoban show that RePro enhances the Qwen family's performance, with up to 11.57% success rate gains.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has enabled Large language models (LLMs) to learn from experience through outcome rewards, substantially improving the reasoning capabilities DeepSeek (2024); Feng et al. (2026); Dong et al. (2025b). These rewards, however, provide no explicit supervision about progress at intermediate steps. In multi-turn agentic tasks, agents also receive observations after taking actions, accumulating rich environmental feedback about how their interactions affect the environment Xie et al. (2024); Liu et al. (2026); Shridhar et al. (2020); Wang et al. (2026b). Turning this rich feedback into useful training signals requires assessing how each interaction with the environment associates with the task goal.

Human metacognition offers a perspective on deriving training signals from these interactions: when pursuing a goal, people monitor what has been accomplished, what remains, and whether they are on track (Lin et al., 2026; Li et al., 2025b; Han et al., 2025). Progress assessment makes this perspective concrete by evaluating how intermediate interactions advance task completion. Unlike a standard scalar value estimate of expected return, each assessment includes a verbalized rationale that draws on the LLM's semantic knowledge to explain how the observed state relates to the task goal. Such assessments can be generated through the actor policy's own retrospective reflection, providing a source of process-level supervision without a separately trained process reward model. This motivates us to ask whether agents can generate useful progress assessments from their interaction histories and use them as training signals to complement sparse outcome rewards.

In a preliminary study on WebShop, we investigate whether agents can generate progress assessments that improve decisions. We compare two ways of eliciting progress assessments: directly prompting the agent to assess its current state, and teaching the agent with demonstrations of progress annotated retrospectively from completed trajectories with known outcomes. In both settings, the resulting assessments inform subsequent action selection. Relative to a baseline without progress assessments, direct prompting lowers task success by 5.20 on average, whereas hindsight-guided prompting raises it by 3.35 points. This contrast suggests that useful progress assessment does not emerge simply from asking agents to evaluate their current state, but can be supported by retrospective information. These findings motivate leveraging progress assessment from retrospective trajectory outcomes.

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.

This paper proposes **RePro** (**Re**trospective **Pro**gress-Aware Training), a two-phase framework that uses the agent's own retrospective assessments to construct step-wise training signals to shape policy rewards, while an auxiliary alignment objective trains the progress predictions produced during execution toward the retrospective estimates. We augment trajectories with a **forward-then-reflect** paradigm: the agent first *forward* to execute the task while generating online progress estimates; once completing the task and knowing the outcome, the agent retrospectively reassesses its step-wise progress anchored by the final result. Based on this, RePro first applies a stage of **Retrospection Warmup**: a small set of external-LLM demonstrations teaches the agent the forward-then-reflect format for retrospective reflection. After that, **RePro-PO** leverages the retrospective progress signal by a composite reward, comprising retrospective progress shaping, online-retrospective alignment, and format regularization.

We evaluate RePro across LLMs of varying sizes on WebShop, ALFWorld, and Sokoban. On WebShop, RePro improves the absolute task success rate over baseline training by $+8.98\%$, $+11.57\%$, and $+5.82\%$ across different model sizes. It also yields consistent gains on ALFWorld and Sokoban, and outperforms all ablation variants. Progress-quality analysis further provides behavioral evidence for improved progress awareness: compared with the prompting baseline, RePro's intermediate estimates provide substantially stronger discrimination between trajectories that eventually succeed and those that fail. Our contributions are threefold:

1. We show that directly prompting agents to estimate progress can hurt task performance, whereas retrospective demonstrations improve it, motivating the use of hindsight for progress assessment.
2. We introduce RePro, which uses the agent's own retrospective reasoning over completed trajectories and outcomes to generate step-wise progress assessments and incorporate them into policy training.
3. Across three agentic benchmarks and multiple model scales, we demonstrate RePro's effectiveness through performance comparisons and ablations, and characterize its retrospective progress assessments through behavioral analysis.

2 Related Work

2.1 LLM Agent Training

LLM-based agent training has long drawn inspiration from reinforcement learning, showing that agent behavior can be improved by reusing past interaction trajectories as training signals (Mnih et al., 2015; Schulman et al., 2017; Xiong et al., 2024; Song et al., 2024; Cao et al., 2025). Recently, critic-free policy optimization has improved efficiency by estimating advantages from rollout groups, trajectory-level comparisons, or improved sampling strategies (Guo et al., 2025; DeepSeek, 2024; Feng et al., 2026; Dong et al., 2025b;a; Zheng et al., 2025), and have been widely applied to LLM agent training across diverse environments (Qian et al., 2026; Xi et al., 2026a).

Reward design remains a central challenge in long-horizon agent RL. Outcome-based methods suffer from sparse terminal feedback (Guo et al., 2025; DeepSeek, 2024); process supervision introduces expensive annotations, external reward models, or engineered subgoals (Lightman et al., 2024; Zou et al., 2026; Xi et al., 2026b; Zheng et al., 2026; Wang et al., 2026c) or relies on environment-specific signals (Lu et al., 2026). Recent work has also revisited *critics*, combining a critic or value model as compensation (Li et al., 2026; Hou et al., 2026). Alternatively, existing progress-based methods explicitly model task-completion progress through heuristic estimates or supervision from stronger LLMs (Zhang et al., 2025a; 2026b; Chai et al., 2025). Differently, our method uses minimal external supervision and learns progress awareness from the agent's own completed trajectory outcomes, internalizing progress estimation into the agent itself without relying on an additional reward model, and at test time, jointly estimates its progress, helping make decisions.

2.2 Self-Improving LLMs via Reflection

Reflection improves agent behavior by turning past experience into reusable guidance. Failed episodes are converted into verbal feedback or episodic memory (Shinn et al., 2023; Qu et al., 2024), while others ground reflection in execution states or evolving contextual playbooks (Kim et al., 2025; Guo et al., 2026; Cai et al., 2026; Zhang et al., 2025b). Recent work further extends reflection to

uncertainty-aware test-time adaptation, reusable experiential heuristics, and pre-execution plan critique (Acikgoz et al., 2025; Ge et al., 2025; Allard et al., 2026; Wang et al., 2026a). These methods can elicit reflective behaviors already present in the base model, but do not directly internalize new reflection capabilities into the agent.

Training-time methods instead aim to internalize reflection through learning. They improve meta-introspection or reward useful self-reflective tokens (Li et al., 2025a; Bensal et al., 2025), support staged self-correction and consistency (Kumar et al.; Zhang et al., 2026a), and enable self-evolution (Ou et al., 2026). Our work follows this line in internalizing self-improvement signals into the agent itself. Rather than correcting individual actions, our method introduces retrospective, progress-aware reflection, in which the agent uses completed trajectories to assess task-level progress at intermediate steps. This yields self-generated progress targets that train the agent to track *where it stands* in a long-horizon task, without relying on an additional reward model. Detailed comparisons with related methods are provided in Appendix A.

3 PRELIMINARIES

3.1 FORMULATION

Given a natural language goal g, an LLM-based agent interacts with an environment to complete the task. This interaction is modeled as a Partially Observable Markov Decision Process (POMDP) (Kaelbling et al., 1998), defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{O}$, $\mathcal{P}$, and $\mathcal{R}$ denote the state space, action space, observation space, transition kernel, and reward function.

At each step t, the agent receives a partial observation $o_t \in \mathcal{O}$ of the underlying state $s_t \in \mathcal{S}$, and samples an action from its policy conditioned on the interaction history $h_t = (g, o_1, a_1, \ldots, o_t)$:

$$a_t \sim \pi_\theta(\cdot \mid h_t). \tag{1}$$

The environment transitions according to $\mathcal{P}(s_{t+1} \mid s_t, a_t)$, producing observation o_{t+1}. This sequential interaction forms a trajectory

$$\tau = \{g,\, [(o_t, a_t)]_{t=1}^{T}\}, \tag{2}$$

which terminates upon an environment signal or agent decision at step T, yielding a task reward $r = \mathcal{R}(\tau)$. The policy π_θ is optimized via RL to maximize the expected task reward.

Under partial observability, the agent has access only to a bounded window of its interaction history and partial observations of the environment, while its intermediate task progress remains latent and feedback is typically limited to a sparse terminal reward. This motivates our study to infer and verbalize progress awareness and convert it into progress-aware reward signals.

3.2 PILOT STUDY

Under the formulation above, the agent receives an environmental observation after each action. These observations provide step-wise evidence of how the task is unfolding, which the agent may use to estimate its current progress p_t and inform subsequent actions $a_{\geq t}$.

Our first question is: **can progress assessment help an agent act?** An assessment of how far the task has progressed may inform the agent's next decision. We study this question on WebShopYao et al. (2022) by placing a progress-assessment step before action prediction at each interaction turn and measuring the resulting task success rate.

The following question is: **how to elicit the progress assessment.** At each decision-making time, the agent must assess progress from an incomplete trajectory; completed trajectories and their outcomes, however, provide additional context for progress assessment. Therefore, we compare two methods: **(1) direct assessment**, where the agent estimates progress from its current interaction history; and **(2) retrospectively guided assessment**, where we collect completed trajectories and their outcomes from a separate training set. The same agent model retrospectively annotates the progress at each observation using the full traces and outcomes; at test time, these annotated trajectories serve as in-context demonstrations guiding the progress assessment.

In both cases, the agent first estimates its progress upon receiving observations and then appends that assessment to the action-prediction prompt for a separate call without demonstrations. To ablate prompts and added contexts, we include a random baseline by replacing the model's online progress estimates or demonstrations with randomly sampled percentage values and fabricated rationales. Details are provided in Appendix C.

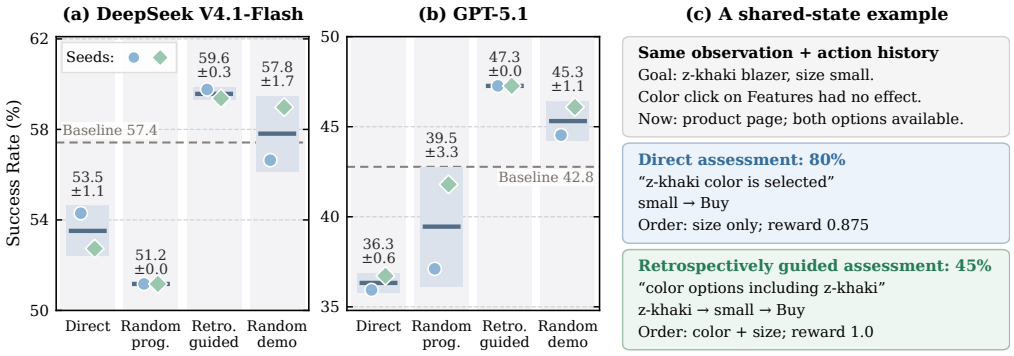


Figure 1: Pilot study results on WebShop. Online progress prompting hurts performance, while retrospective demonstrations for progress improve it.

Figure 1 shows that the utility of progress assessment depends on how it is elicited. (1) Direct assessment lowers mean success rates on both models (DeepSeek: 57.4%→53.5%; GPT-5.1: 42.8%→36.3%), while Random Progress also underperforms the baselines. (2) Retrospectively guided assessment instead achieves 59.6% and 47.3%, improving over the baselines by an average of 3.3 percentage points. Random Demonstration preserves the original observation sequences but replaces progress annotations with random numbers and fabricated rationales. It achieves 57.8% and 45.3%, below retrospectively guided assessment on both models. Although we do not establish the accuracy of the progress estimates, outcomes suggest that informative progress annotations provide benefits beyond trajectory content, while direct prompting does not reliably elicit assessments that improve decision-making.

(3) Figure 1(c) illustrates a concrete difference: given the same action-observation history, direct assessment incorrectly treats a color as already selected with a 80% procress percentage, whereas retrospectively guided assessment describes it as available with a 45% completion. This branch leads to different outcome of subsequent purchases. Together, these observations motivate learning progress assessment from retrospective trajectory outcomes, rather than relying on direct prompting to elicit useful self-assessment.

4 Retrospective Progress-Aware Training

Our preliminaries motivate us to use completed trajectories to generate progress assessments for training. We propose **RePro** (**Re**trospective **Pro**gress-Aware Training), shown in Figure 2, where the agent reassesses its own trajectory after each episode and uses the resulting step-wise progress signals in policy optimization. RePro first uses **Retrospection Warmup** (§4.2) to initialize this reflection behavior, followed by **RePro-PO** (§4.3) to train with the its own retrospective assessments.

4.1 Forward-then-Reflect

By adding retrospective reflection after each episode, RePro turns the agent's working pattern into a forward-then-reflect process.

Forward Execution. At each step t, the agent receives the partially observed interaction history h_t, generates a reasoning trace, and predicts the progress assessment and an action :

$$(\tilde{p}_t, a_t) \sim \pi_\theta(\cdot \mid h_t), \tag{3}$$

where $\tilde{p}_t \in [0, 1]$ denotes the agent's verbalized online estimate of task completion at each decision point t after observing o_t. They are verbalized as percentages and normalized to $[0, 1]$. The reasoning

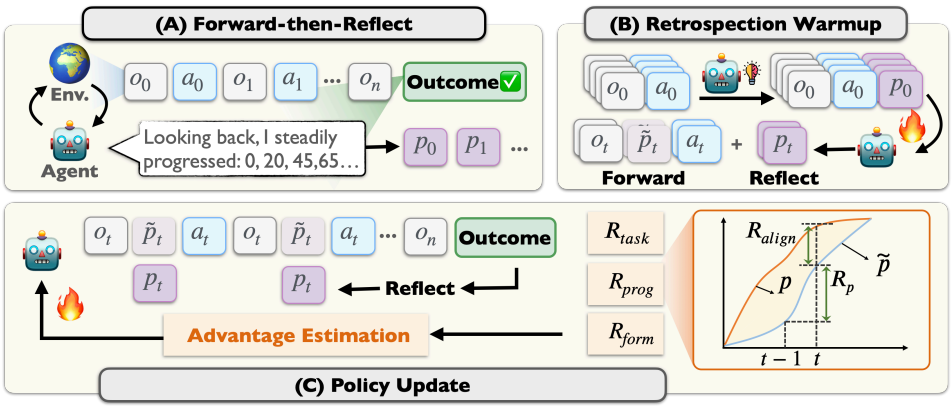


Figure 2: Overview of **RePro**. (A)**Forward-then-reflect**: the agent executes actions with online progress estimates, then retrospectively reassesses progress after observing the final outcome. (B) **Retrospection Warmup** to initialize retrospective reflection behavior. (C) **Policy Update** integrates retrospective progress into RL reward.

trace is omitted from the notation. a_t is the action to execute in the environment, yielding the next observation o_{t+1}. The trajectory continues until the terminal step T, returning task reward r.

Retrospective Reflection. After the episode, the agent reviews the complete trajectory τ and outcome r. Prompted to reason about how the task progressed, it produces retrospective estimates for the previously acted steps. We denote this prompted mode as $\pi_\theta^{\text{retro}}$:

$$(p_1, \ldots, p_T) \sim \pi_\theta^{\text{retro}}(\cdot \mid \tau, r), \tag{4}$$

Here, $p_t \in [0, 1]$ retrospectively assesses task completion after observing o_t and before taking a_t, which are verbalized as percentages and normalized to $[0, 1]$. The trajectory is then re-organized by associating the retrospective estimates p_t with their corresponding steps:

$$\tau_{\text{retro}} = \{g, [(o_t, a_t, p_t, \tilde{p}_t)]_{t=1}^{T}\}. \tag{5}$$

The known outcome anchors the terminal assessment, and the complete trajectory provides context for assessing the intermediate steps.

4.2 Retrospection Warmup

Outcome-based RL trains the agent to act in the environment, but does not by itself establish the behavior of reflecting on a completed trajectory when prompted. We therefore use supervised warmup to initialize retrospective reflection. We collect successful rollout trajectories and use an external model π_{demo} (DeepSeek-V4) to generate each compact record with a retrospective reflection. Then, we replace each online progress estimate in the original trajectory with its retrospective counterpart and append the complete reflection as a final turn, yielding the warmup dataset:

$$\begin{aligned} \mathbf{p}_\tau &= (p_1, \ldots, p_T) \sim \pi_{\text{demo}}^{\text{retro}}(\cdot \mid \tau, r), \quad \tau \in \mathcal{D}_{\text{succ}}, \\ \mathcal{D}_{\text{warmup}} &= \{(\tau, \mathbf{p}_\tau) \mid \tau \in \mathcal{D}_{\text{succ}}\}. \end{aligned} \tag{6}$$

The agent is fine-tuned on this dataset using a cross-entropy loss to generate actions and progress estimates at each step, followed by a retrospective reflection after the episode.

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(\tau, \mathbf{p}_\tau) \sim \mathcal{D}_{\text{warmup}}} \left[\sum_{t=1}^{T} \log \pi_\theta(p_t, a_t \mid h_t) + \log \pi_\theta^{\text{retro}}(\mathbf{p}_\tau \mid \tau, r) \right]. \tag{7}$$

4.3 RePro-PO: Policy Optimization with Retrospective Progress

The reward integrates the outcome task performance with progress-specific objectives.

$$R(\tau) = \sum_{t=1}^{T} r_t + R_{\text{boundary}}(\tau), \tag{8}$$

For each step, progress reflection provides intermediate signals beyond sparse outcome reward:

$$r_t = r_{\text{env}}(t) + r_p(t) + r_{\text{align}}(t) + \mathbb{1}[t = T]\,(R_{\text{corr}}(\tau) + R_{\text{retro-format}}(\tau))\,. \tag{9}$$

$r_{\text{env}}(t)$ is the environment reward, which is sparse, outcomet-based. $r_p(t)$ rewards retrospective progress differences, and $r_{\text{align}}(t)$ encourages pointwise online-retrospective agreement. The trajectory-level correlation and retrospective format rewards are applied at the final step.

$$\begin{aligned} r_p(t) &= \begin{cases} \beta(p_{t+1} - p_t), & t < T, \\ 0, & t = T, \end{cases} \\ r_{\text{align}}(t) &= \frac{\alpha}{T} \max\left(0, 1 - |p_t - \tilde{p}_t|\right), \\ R_{\text{corr}}(\tau) &= \alpha_{\text{corr}}\ \text{PearsonCorr}\left((\tilde{p}_t)_{t=1}^{T}, (p_t)_{t=1}^{T}\right). \end{aligned} \tag{10}$$

Additionally, the retrospective format reward checks for a valid progress estimate. The boundary reward encourages the initial online estimate to be 0 and the last of successful trajectories to be 1:

$$\begin{aligned} R_{\text{retro-format}}(\tau) &= \begin{cases} +0.1, & \text{if the retrospective format is valid,} \\ -0.1, & \text{otherwise,} \end{cases} \\ R_{\text{boundary}}(\tau) &= -w_p\left(|\tilde{p}_1| + \mathbb{1}[\text{success} \wedge T > 1]|\tilde{p}_T - 1|\right). \end{aligned} \tag{11}$$

Following GiGPO (Feng et al., 2026), we compute hierarchical advantages at two granularities. The episode advantage $A^{\text{episode}}(\tau)$ normalizes the total return $R(\tau)$ against trajectories rolled out from the same task. The step advantage A_t^{step} compares discounted returns G_t among actions taken from the same anchor state, i.e., identical observations $o_k = o_t$ that naturally recur across trajectories,

$$G_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k} = r_t + \gamma G_{t+1}. \tag{12}$$

Since r_t incorporates progress shaping, G_t encodes *when* progress increases occur, enabling fine-grained credit assignment. The final advantage combines both levels and is shared by all tokens in the forward response, including reasoning, progress estimates, and actions. We optimize a token-level clipped objective, with $\rho_{t,\ell}(\theta) = \frac{\pi_\theta(y_{t,\ell}|h_t, y_{t,<\ell})}{\pi_{\text{old}}(y_{t,\ell}|h_t, y_{t,<\ell})}$, where $y_{t,\ell}$ denotes the ℓ-th token of the full response at step t:

$$\begin{aligned} A_t &= \underbrace{\frac{R(\tau) - \text{mean}\left(\{R(\tau_j)\}_{j=1}^{N}\right)}{F_{\text{norm}}\left(\{R(\tau_j)\}_{j=1}^{N}\right)}}_{A^{\text{episode}}(\tau)} + \omega \cdot \underbrace{\frac{G_t - \text{mean}\left(\{G_k \mid o_k = o_t\}\right)}{F_{\text{norm}}\left(\{G_k \mid o_k = o_t\}\right)}}_{A_t^{\text{step}}} \\ \mathcal{L}_{\text{PG}}(\theta) &= -\mathbb{E}_{\tau,t,l}\left[\min\left(\rho_{t,\ell}(\theta) A_t,\ \text{clip}\left(\rho_{t,\ell}(\theta), 1-\epsilon, 1+\epsilon\right) A_t\right)\right]. \end{aligned} \tag{13}$$

5 Experiments

5.1 Setup

Benchmarks We evaluate on three agentic benchmarks: **WebShop** for online shopping (Yao et al., 2022), **ALFWorld** for text-based household tasks (Shridhar et al., 2020), and **Sokoban** for box-pushing puzzles (Schrader, 2018). We report task success rate (SR) and environment score on 256 test samples per benchmark.

Baselines Baseline methods for comparison are twofold. **(1) Prompting methods.** Following our pilot study, we evaluate online and retrospective progress prompting, using random progress as a control for the added prompt structure. **(2) Training methods.** We use standard GRPO DeepSeek (2024) and GiGPO Zheng et al. (2025) as our primary RL training baselines. We further include a progress-aware prompting baseline, denoted as *Meta Prompt*, where the agent is explicitly prompted to assess online progress. Since *Meta Prompt* uses the same progress-aware prompting format as our method, we use it as the reference baseline for computing Δ across methods. On the basis of

Table 1: Main results on WebShop across three model sizes. For training methods, we report both the best score (*Best*) and convergence window average (*Avg*). SR: success rate (%). Δ: improvement over the corresponding baseline. The same applies below.

		Qwen2.5-1.5B				Qwen2.5-3B				Qwen2.5-7B			
Method		**SR**	Δ	**Score**	Δ	**SR**	Δ	**Score**	Δ	**SR**	Δ	**Score**	Δ
Prompting Methods													
Base Prompt		0.4	–	5.0	–	12.5	–	36.5	–	14.1	–	50.8	–
Meta Prompt for Progress		1.2	+0.8	3.8	-1.2	16.4	+3.9	41.5	+5.0	10.9	-3.2	41.4	-9.4
Random Progress		0.8	+0.4	13.2	+8.2	16.4	+3.9	52.4	+15.9	10.2	-3.9	44.4	-6.4
Retro Progress		5.1	+4.7	20.7	+15.7	10.6	-1.9	44.5	+8.0	23.8	+9.7	51.5	+0.7
Random Retro Progress		2.0	+1.6	18.5	+13.5	16.0	+3.5	48.4	+11.9	12.5	-1.6	46.5	-4.3
Training Methods													
GRPO	Best	66.41	-7.03	81.22	-8.60	50.00	-21.88	71.02	-13.39	73.29	-5.62	82.18	-3.75
	Avg	62.77	-4.38	77.26	-8.01	44.53	-23.63	63.90	-20.48	64.02	-10.63	76.46	-8.63
GiGPO	Best	67.97	-5.47	81.35	-8.47	71.88	+0.00	83.94	-0.47	74.22	-4.69	83.98	-1.95
	Avg	59.84	-7.31	77.68	-7.59	63.57	-4.59	81.49	-2.89	71.76	-2.89	83.77	-1.32
Meta Prompt	Best	73.44	–	89.82	–	71.88	–	84.41	–	78.91	–	85.93	–
	Avg	67.15	–	85.27	–	68.16	–	84.38	–	74.65	–	85.09	–
L1	Best	68.75	-4.69	84.51	-5.31	76.56	+4.68	88.53	+4.12	80.86	+1.95	**91.74**	+5.81
	Avg	63.20	-3.95	82.17	-3.10	71.68	+3.52	86.21	+1.83	75.62	+0.97	87.81	+2.72
L2	Best	75.00	+1.56	88.01	-1.81	76.56	+4.68	88.60	+4.19	75.39	-3.52	87.54	+1.61
	Avg	70.47	+3.32	85.67	+0.40	70.86	+2.70	86.49	+2.11	70.59	-4.06	83.45	-1.64
RePro (Ours)													
RePro	Best	**81.64**	+8.20	**91.72**	+1.90	**83.59**	+11.71	**92.09**	+7.68	**84.38**	+5.47	91.18	+5.25
	Avg	**76.13**	+8.98	**87.56**	+2.29	**79.73**	+11.57	**90.75**	+6.37	**80.47**	+5.82	**89.13**	+4.04

Table 2: Main results on ALFWorld and Sokoban across three model sizes.

		Qwen2.5-1.5B on ALFWorld				Qwen2.5-7B on ALFWorld				Qwen3-4B on Sokoban			
Method		**SR**	Δ	**Score**	Δ	**SR**	Δ	**Score**	Δ	**SR**	Δ	**Score**	Δ
Training Methods													
GiGPO	Best	89.84	+2.34	5.74	+0.10	95.31	+0.00	7.82	-0.15	86.72	+1.56	6.21	+0.27
	Avg	84.38	+5.90	5.49	+1.02	90.08	-1.29	6.34	-0.15	76.09	+2.03	4.21	+0.29
Meta Prompt	Best	87.50	–	5.64	–	95.31	–	7.97	–	85.16	–	5.94	–
	Avg	78.48	–	4.47	–	91.37	–	6.49	–	74.06	–	3.91	–
L1	Best	95.70	+8.20	7.88	+2.24	99.61	+4.30	9.78	+1.81	**89.84**	+4.68	6.53	+0.60
	Avg	88.05	+9.57	5.81	+1.34	**97.81**	+6.44	8.83	+2.34	79.22	+5.16	4.56	+0.64
L2	Best	74.22	-13.28	4.43	-1.21	98.83	+3.52	9.29	+1.32	88.28	+3.12	6.44	+0.50
	Avg	65.62	-12.86	3.63	-0.84	96.60	+5.23	8.30	+1.81	80.08	+6.02	4.79	+0.88
RePro (Ours)													
RePro	Best	**99.22**	+11.72	**9.42**	+3.78	**100.00**	+4.69	**10.26**	+2.29	88.28	+3.12	**6.61**	+0.67
	Avg	**96.02**	+17.54	**7.94**	+3.47	97.54	+6.17	**8.90**	+2.41	**81.64**	+7.58	**5.10**	+1.18

Meta Prompt, two vanilla implementations of progress augmentation are evaluated. *L1* denotes only adding a format penalty in the reward function to the *Meta Prompt*, while *L2* denotes directly using $\hat{p}_t$ as the step reward r_t in Equation 12. Full details are provided in Appendix B.

5.2 Main Results

Progress assessment does not reliably emerge from prompting or outcome-oriented training. As shown in Table 1, neither improving prompting nor existing training methods reliably elicits effective progress assessment. Simple prompting often degrades performance, while retrospective prompting without training yields only limited gains. Regarding training methods, *Meta Prompt* induces longer reasoning traces and yields performance gains in most settings compared with GRPO and GiGPO, indicating that progress-aware prompting provides a richer context, with additional tokens not necessarily becoming a burden. We further observe that outcome-oriented RL with progress prompt and simple leverage (L1/L2) produces inconsistent improvements across settings.

RePro improves task success across model sizes and environments. Table 1 shows that RePro consistently outperforms all baselines on WebShop across 1.5B, 3B, and 7B models, achieving absolute success rate gains of +8.98%, +11.57%, and +5.82% over the *Meta Prompt* baseline, respectively. Compared to progress-related variants, RePro further improves performance, demonstrating the effectiveness of retrospective progress modeling beyond vanilla reward engineering. Across environments, RePro generalizes to both ALFWorld household planning (+11.72 SR for Qwen2.5-1.5B and +4.69 SR for Qwen2.5-7B) and Sokoban spatial reasoning (+3.12 SR for Qwen3-4B),

Table 3: Results for ablation study and variants of our method, using the Best numbers.

Method	Qwen2.5-1.5B				Qwen2.5-3B				Qwen2.5-7B			
	SR	Δ	Score	Δ	SR	Δ	Score	Δ	SR	Δ	Score	Δ
RePro (Ours)												
RePro	81.64	–	91.72	–	83.59	–	92.09	–	84.38	–	91.18	–
Ablations												
w/ 7B Warmup data	83.59	+1.95	93.14	+1.42	82.81	-0.78	91.28	-0.81	–	–	–	–
Warmup + GiGPO	75.00	-6.64	88.44	-3.28	78.91	-4.68	87.94	-4.15	79.69	-4.69	88.38	-2.80
Warmup Only	0.00	-81.64	0.00	-91.72	28.52	-55.07	32.49	-59.60	61.72	-22.66	71.94	-19.24
w/o Warmup	NA	NA	NA	NA	NA	NA	NA	NA	80.08	-4.30	89.96	-1.22
Variants												
L1+ Reshaping	72.27	-9.37	87.43	-4.29	72.66	-10.93	84.82	-7.27	80.08	-4.30	87.54	-3.64
L1+ Reweighting	71.48	-10.16	83.55	-8.17	75.00	-8.59	87.62	-4.47	77.34	-7.04	88.98	-2.20
L1+ Grouping	75.78	-5.86	89.63	-2.09	73.83	-9.76	86.32	-5.77	78.13	-6.25	89.23	-1.95

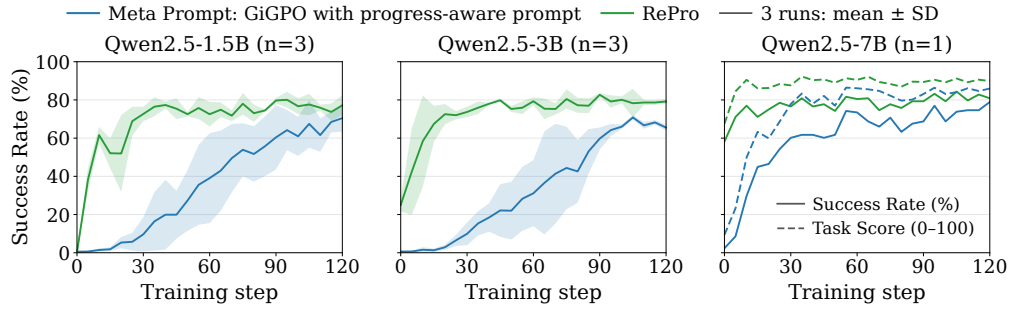


Figure 3: Learning curves on WebShop. Solid lines show evaluation success rate; shaded regions indicate standard deviation across three runs. The dashed curves in the 7B panel show task score.

supporting the hypothesis that retrospective progress awareness enhance long-horizon agent learning. The improvement on Sokoban is relatively smaller, which may be because its reward already provides partial process-oriented feedback.

Across-seed consistency. To assess sensitivity to training randomness, we repeat WebShop training with three independent seeds for Qwen2.5-1.5B and Qwen2.5-3B. Figure 3 and Table 5 report the mean validation performance with standard deviation across runs. RePro reaches a high success rate earlier than the *Meta Prompt* baseline and maintains an advantage throughout training. After step 100, the success-rate gap remains approximately 10 percentage points on average across the plotted checkpoints for both model sizes. The task-score curves show the same qualitative pattern. For Qwen2.5-7B, we report one training run with both success rate and task score, showing a similar trend of faster convergence and an advantage during the later training stages.

6 Analysis

6.1 Ablations and Variants

Ablation settings for RePro include: (1) w/ 7B warmup data: Using the warmup dataset from 7B trajectories. (2) Warmup + GiGPO: Two-phase training with SFT warmup followed by standard GiGPO without retrospective progress mechanism. (3) Warmup Only: Performance after only warmup on retrospective-labeled data without subsequent RL training. (4) w/o Warmup: Direct RL from base model with retrospective mechanism enabled. The following are variants that leverage progress assessments in different ways. Table 3 reports the ablations and variants.

(1) The Role of SFT Warmup. *Warmup Only* achieves substantially lower performance than RePro (crashing on 1.5B and 3B, -22.66% on 7B), despite achieving nearly perfect format accuracy, showing that SFT teaches the *format* of retrospective reflection but does not yield effective progress awareness without task-reward grounding. Removing warmup destabilizes training for 1.5B and 3B models, indicating that warmup is critical for establishing the forward-then-reflect behavior. Using stronger warmup demonstrations from the 7B model further improves the 1.5B model. **(2) The Role**

Table 4: Progress assessment analysis on WebShop. IntermDisc is reported in percentage points, while temporal R^2 is scaled by 100, and all other metrics are percentages (Appendix D).

		Qwen2.5-1.5B		Qwen2.5-3B		Qwen2.5-7B	
Metric	**Outcome**	Baseline	**RePro**	Baseline	**RePro**	Baseline	**RePro**
IntermDisc (pp)	–	1.86	**6.37**	1.11	**31.62**	−0.01	**19.34**
Temporal R^2	Success	96.89	78.14	21.62	94.71	48.74	96.52
	Failure	97.02	75.64	18.69	60.72	46.13	79.24
Monotonicity (%)	Success	99.99	99.60	99.34	100.00	99.98	99.99
	Failure	100.00	99.26	99.32	92.76	99.94	95.72
Decline (%)	Success	0.04	1.28	2.72	0.03	0.15	0.02
	Failure	0.00	2.88	3.01	30.58	0.42	20.07
Plateau (%)	Success	0.47	27.08	78.49	1.37	99.98	0.45
	Failure	0.57	33.03	82.18	12.77	99.86	6.45

of Retrospective Reflection. Comparisons against *Warmup + GiGPO* and *L1 + Reshaping* show that retrospective reflection contributes beyond format initialization or reward shaping alone. Retrospective progress differences correlate positively with step-level advantages in 91.5% of training rollouts, an association that partly reflects the use of progress shaping in advantage computation.

6.2 Progress Predictions

Metrics. Progress lacks objective ground truth in outcome-based agentic tasks, making direct accuracy evaluation infeasible. We therefore examine outcome alignment and behavioral properties. Our primary metric, **Intermediate Discrimination (IntermDisc)**, measures the average stepwise gap in progress estimates between successful and failed trajectories. We additionally report **Temporal R^2**, **Monotonicity**, **Decline Rate**, and **Plateau Rate** to characterize progress dynamics. Detailed definitions and aggregation procedures are provided in Appendix D.

Findings. Table 4 compares the baseline's progress estimates with RePro's retrospective estimates, which serve as reward signals during training. Both are evaluated on their respective test-set trajectories. **(1) RePro's progress signals distinguish task outcomes more clearly.** Baseline IntermDisc remains close to zero across model sizes, ranging from -0.01 to 1.86 percentage points. RePro achieves higher discrimination: 6.37 (1.5B), 31.62 (3B), and 19.34 (7B). The stepwise profiles in Appendix D, Figure 4, show corresponding separation between successful and failed trajectories, particularly for 3B and 7B; the gap is smaller for 1.5B. These results show that RePro's retrospective supervision differentiates progress according to the eventual task outcome. **(2) Progress dynamics also differ across outcomes.** With RePro, successful trajectories exhibit higher temporal R^2 and monotonicity than failed trajectories across all three model sizes. Decline rates on failed trajectories are increased (e.g., 30.58% to 0.03%) while plateau rates decrease (e.g., 82.18% to 12.77%). RePro's retrospective signals exhibit more downward revisions on failed trajectories. Representative progress assessments appear in Table 6 (Appendix E).

7 Conclusion

This work investigates how LLM-based agents can turn interaction histories into training signals that complement sparse outcome rewards. We introduce RePro, which uses the acting policy's own retrospective reasoning over completed trajectories and known outcomes to generate step-wise progress assessments. Following supervised warmup, these assessments provide progress-based reward shaping and alignment targets for online estimates, without requiring a separate process reward model or ongoing external annotation. Experiments on WebShop, ALFWorld, and Sokoban show that this approach improves task success across multiple model settings. These findings support retrospective progress assessment as auxiliary supervision for long-horizon tasks and motivate further research into self-improving agents.

AI USE STATEMENT

LLMs were used solely in an auxiliary capacity, primarily for linguistic refinement, including grammar correction, clarity improvement, phrasing and formulation checks, numerical consistency checks, and the removal of non-academic expressions. We also used DeepSeek to generate retrospective progress labels for the supervised warmup data. LLMs were **not** used to generate research ideas, design experiments, conduct literature reviews, or make methodological decisions. All conceptual contributions, experimental designs, methodological choices, and research conclusions were independently developed and carried out by the authors. We have reviewed all AI-assisted work.

ETHICS STATEMENT

This work studies LLM-based agent training in controlled benchmark environments, including WebShop, ALFWorld, and Sokoban. The experiments do not involve human evaluation, real-world purchases, or physical actions.

REPRODUCIBILITY STATEMENT

Sections 3-5 describe the task formulation, training method, and evaluation setup, with results and ablations in Sections 5-6. Appendix B provides the retrospective reflection prompt, the pilot evaluation protocol and demonstration construction, training configurations, reward implementation, and checkpoint and multi-seed reporting protocols. Appendix D defines the progress-assessment metrics and their aggregation procedures.

REFERENCES

Emre Can Acikgoz, Cheng Qian, Heng Ji, Dilek Hakkani-Tür, and Gokhan Tur. Self-improving llm agents at test-time. *arXiv preprint arXiv:2510.07841*, 2025.

Marc-Antoine Allard, Arnaud Teinturier, Victor Xing, and Gautier Viaud. Experiential reflective learning for self-improving llm agents. *arXiv preprint arXiv:2603.24639*, 2026.

Shelly Bensal, Umar Jamil, Christopher Bryant, Melisa Russak, Kiran Kamble, Dmytro Mozolevskyi, Muayad Ali, and Waseem AlShikh. Reflect, retry, reward: Self-improving llms via reinforcement learning. *arXiv preprint arXiv:2505.24726*, 2025.

Yuxuan Cai, Yipeng Hao, Jie Zhou, Hang Yan, Zhikai Lei, Rui Zheng, Zhenhua Han, Yutao Yang, Junsong Li, Qianjun Pan, Tianyu Huai, Qin Chen, Kai Chen, Bo Zhang, Xipeng Qiu, and Liang He. A benchmark for self-evolving agents via experience-driven lifelong learning, 2026. URL https://openreview.net/forum?id=vznmtmUPmA.

Zouying Cao, Runze Wang, Yifei Yang, Xinbei Ma, Xiaoyong Zhu, Bo Zheng, and Hai Zhao. Pgpo: Enhancing agent reasoning via pseudocode-style planning guided preference optimization. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 14966–14985, 2025.

Huacan Chai, Zijie Cao, Maolin Ran, Yingxuan Yang, Jianghao Lin, Xin Peng, Hairui Wang, Renjie Ding, Ziyu Wan, Muning Wen, et al. Parl-mt: Learning to call functions in multi-turn conversation with progress awareness. *arXiv preprint arXiv:2509.23206*, 2025.

DeepSeek. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Guanting Dong, Licheng Bao, Zhongyuan Wang, Kangzhi Zhao, Xiaoxi Li, Jiajie Jin, Jinghan Yang, Hangyu Mao, Fuzheng Zhang, Kun Gai, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic entropy-balanced policy optimization. *CoRR*, abs/2510.14545, 2025a. doi: 10.48550/ARXIV.2510.14545. URL https://doi.org/10.48550/arXiv.2510.14545.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization. *CoRR*, abs/2507.19849, 2025b. doi: 10.48550/ARXIV.2507.19849. URL https://doi.org/10.48550/arXiv.2507.19849.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=QXEhBMNrCW.

Yubin Ge, Salvatore Romeo, Jason Cai, Monica Sunkara, and Yi Zhang. Samule: Self-learning agents enhanced by multi-level reflection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 16602–16621, 2025.

Dadi Guo, Tianyi Zhou, Dongrui Liu, Chen Qian, Qihan Ren, Shuai Shao, Zhiyuan Fan, Yi R. Fung, Kun Wang, Linfeng Zhang, and Jing Shao. Towards self-evolving agent benchmarks : Validatable agent trajectory via test-time exploration. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=2H03gm4Rq6.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware LLM reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1274. URL https://aclanthology.org/2025.findings-acl.1274/.

Zhenyu Hou, Yujiang Li, Jie Tang, and Yuxiao Dong. Single-rollout asynchronous optimization for agentic reinforcement learning. *arXiv preprint arXiv:2607.07508*, 2026.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Jeonghye Kim, Sojeong Rhee, Minbeom Kim, Dohyung Kim, Sangmook Lee, Youngchul Sung, and Kyomin Jung. Reflact: World-grounded decision making in llm agents via goal-state reflection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33421–33453, 2025.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. In *The Thirteenth International Conference on Learning Representations*.

Jiaqi Li, Xinyi Dong, Yang Liu, Zhizhuo Yang, Quansen Wang, Xiaobo Wang, Song-Chun Zhu, Zixia Jia, and Zilong Zheng. Reflectevo: Improving meta introspection of small llms by learning self-reflection. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 16948–16966, 2025a.

Junbo Li, Peng Zhou, Rui Meng, Meet P Vadera, Lihong Li, and Yang Li. Turn-ppo: Turn-level advantage estimation with ppo for improved multi-turn rl in agentic llms. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 6227–6243, 2026.

Zheng Li, Qingxiu Dong, Jingyuan Ma, Di Zhang, Kai Jia, and Zhifang Sui. Selfbudgeter: Adaptive token allocation for efficient LLM reasoning, 2025b. URL https://openreview.net/forum?id=e7EBzbi8Qd.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Junhong Lin, Xinyue Zeng, Jie Zhu, Song Wang, Julian Shun, Jun Wu, and Dawei Zhou. Plan and budget: Effective and efficient test-time scaling on reasoning large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ctspw4CqbS.

Zhaoyang Liu, JingJing Xie, Zichen Ding, Zehao Li, Bowen Yang, Zhenyu Wu, Xuehui Wang, Qiushi Sun, Shi Liu, Weiyun Wang, Shenglong Ye, Qingyun Li, Zeyue Tian, Gen Luo, Xiangyu Yue, Biqing Qi, Kai Chen, Bowen Zhou, Yu Qiao, Qifeng Chen, and Wenhai Wang. ScaleCUA: Scaling open-source computer use agents with cross-platform data. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=yBFUqdJFZn.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Pengxiang Zhao, Guangyi Liu, et al. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 17608–17616, 2026.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Weixuan Ou, Yanzhao Zheng, Shuoshuo Sun, Wei Zhang, Baohua Dong, Hangcheng Zhu, Ruohui Huang, Gang Yu, Pengwei Yan, and Yifan Qiao. Serl: Self-examining reinforcement learning on open-domain. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 32619–32627, 2026.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru WANG, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=eOLdGbXT6t.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=DRC9pZwBwR.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Max-Philipp B. Schrader. gym-sokoban. https://github.com/mpSchrader/gym-sokoban, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7584–7600, 2024.

Hanyu Wang, Yuanpu Cao, Lu Lin, and Jinghui Chen. Preflect: From retrospective to prospective reflection in large language model agents. *arXiv preprint arXiv:2602.07187*, 2026a.

Qineng Wang, Wenlong Huang, Yu Zhou, Hang Yin, Tianwei Bao, Jianwen Lyu, Weiyu Liu, Ruohan Zhang, Jiajun Wu, Li Fei-Fei, and Manling Li. ENACT: Evaluating embodied cognition with world modeling of egocentric interaction. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=Patx6MRipw.

Taiyi Wang, Sian Gooding, Florian Hartmann, Oriana Riva, and Edward Grefenstette. A subgoal-driven framework for improving long-horizon llm agents. *arXiv preprint arXiv:2603.19685*, 2026c.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Jiaqi Liu, Honglin Guo, yajie yang, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, Wei He, Yiwen Ding, Guanyu Li, Zehui Chen, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Tao Gui, Zuxuan Wu, Qi Zhang, Xuanjing Huang, and Yu-Gang Jiang. Agentgym-RL: An open-source framework to train LLM agents for long-horizon decision making via multi-turn RL. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=ZgCCDwcGwn.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Zhihao Zhang, Wenxiang Chen, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, et al. Agentprm: Process reward models for llm agents via step-wise promise and progress. In *Proceedings of the ACM Web Conference 2026*, pp. 4184–4195, 2026b.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. Watch every step! llm agent learning via iterative step-level process refinement. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1556–1572, 2024.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

Danyang Zhang, Situo Zhang, Ziyue Yang, Zichen Zhu, Zihan Zhao, Ruisheng Cao, Lu Chen, and Kai Yu. Progrm: Build better gui agents with progress rewards. *arXiv preprint arXiv:2505.18121*, 2025a.

Kongcheng Zhang, QI YAO, Shunyu Liu, Yingjie Wang, Baisheng Lai, Jieping Ye, Mingli Song, and Dacheng Tao. Consistent paths lead to truth: Self-rewarding reinforcement learning for LLM reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026a. URL https://openreview.net/forum?id=ckW70ls93V.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*, 2025b.

Xiuhui Zhang, Ning Gao, Xingyu Jiang, Yihui Chen, Yuheng Pan, Mohan Zhang, and Yue Deng. Progress reward model for reinforcement learning via large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026b. URL https://openreview.net/forum?id=TJhHb6CscW.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Congmin Zheng, Xiaoyun Mo, Xinbei Ma, Qiqiang Lin, Yin Zhao, Jiachen Zhu, Xingyu Lou, Jun Wang, Zhaoxiang Wang, Weiwen Liu, et al. Adaptive milestone reward for gui agents. *arXiv preprint arXiv:2602.11524*, 2026.

Jiaru Zou, Ling Yang, Jingwen Gu, Jiahao Qiu, Ke Shen, Jingrui He, and Mengdi Wang. Reasonflux-PRM: Trajectory-aware PRMs for long chain-of-thought reasoning in LLMs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=f3sZjkQbv2.

A Additional Positioning of RePro

Relation to actor-critic learning, process supervision, and reward shaping. A learned critic and RePro's progress-derived rewards can both facilitate credit assignment, but serve different roles. A critic trained on environment rewards usually estimates the expected discounted return:

$$V^{\pi}_{\mathrm{env}}(h_t) = \mathbb{E}_{\pi}\left[\sum_{k=t}^{T} \gamma^{k-t} r_{\mathrm{env}}(k) \;\middle|\; h_t\right]. \tag{14}$$

Although the critic may learn representations correlated with task progress, its training targets do not explicitly describe completed subgoals or explain how individual interactions advance task completion. RePro instead verbalizes such reasoning and augments the reward function using progress assessments generated retrospectively from the complete trajectory and realized outcome:

$$(p_1, \ldots, p_T) \sim \pi_{\theta}^{\mathrm{retro}}(\cdot \mid \tau, r). \tag{15}$$

These assessments combine hindsight context unavailable at the corresponding online step with the semantic knowledge of LLM itself and verbalized reasoning about task completion to infer implicit task structure. For example, in WebShop, it can recognize that a search or filtering action completes an information-gathering subgoal even when the environment provides no immediate reward. Consequently, trajectories with the same terminal outcome may receive different intermediate progress supervision according to how their actions advance task completion. Online-retrospective alignment further trains the policy's online estimates toward these hindsight-derived targets.

RePro is also related to process supervision, but does not require human-labeled intermediate steps or a separately trained process reward model (Lightman et al., 2024; Zou et al., 2026; Xi et al., 2026b). Retrospection Warmup uses a small externally generated set of successful trajectories to initialize the reflection format; subsequent progress assessments in RePro-PO are generated by the agent itself from completed trajectories and task outcomes. RePro further builds on reward shaping. Unlike shaping signals constructed from manually specified subgoals or environment-specific state changes, RePro derives its signal from differences between language-based retrospective progress assessments and combines it with online-retrospective alignment.

Empirically, the GiGPO study provides context for our choice of a critic-free optimizer: it reports stronger WebShop performance than PPO and GRPO under its experimental setup (Feng et al., 2026). Our experiments evaluate the additional benefit of RePro within a GiGPO-based training setup. Table 5 reports published results alongside our comparison between RePro and the GiGPO-based Meta Prompt baseline.

Table 5: Contextual comparison of critic-based, critic-free, and progress-aware optimization on WebShop. Results are from the GiGPO study and our experiments. For our experiments, we report the mean and standard deviation across three seeds, using each run's average success rate over its final five evaluation checkpoints.

Model	Method	WebShop success rate	source
Qwen2.5-1.5B	PPO	51.5 ± 2.9	Feng et al. (2026)
	GRPO	56.8 ± 3.8	Feng et al. (2026)
	GiGPO	67.4 ± 4.5	Feng et al. (2026)
	GiGPO + Meta Prompt	65.78 ± 8.68	Figure 3
	RePro	76.22 ± 1.20	Figure 3

These results do not establish that retrospective progress supervision is fundamentally or information-theoretically superior to a learned critic. Rather, the published results provide context for critic-based and critic-free optimization under a comparable experimental setup and our experiments show that explicit progress supervision provides an effective additional signal within the

critic-free GRPO/GiGPO framework. A critic could in principle be trained using RePro's progress-aware rewards: this would combine the two approaches rather than make retrospective progress supervision redundant. The choice between critic-based and critic-free optimization is therefore orthogonal to RePro's main contribution.

Relation to reflection and self-evolution. Inference-time reflection methods convert past trajectories into verbal feedback, episodic memory, reusable heuristics, or refined plans without updating model parameters (Shinn et al., 2023; Kim et al., 2025; Allard et al., 2026; Wang et al., 2026a). RePro instead uses retrospective reflection as training supervision: the agent first executes the task using online progress estimates and then reassesses the completed trajectory with knowledge of its outcome, allowing the online progress-assessment capability to be internalized through parameter updates. RePro is also related to training-time self-improvement and self-evolution methods (Qu et al., 2024; Li et al., 2025a; Bensal et al., 2025; Kumar et al.; Zhang et al., 2026a; Ou et al., 2026). While these methods broadly optimize self-correction, consistency, or reusable experience, RePro specifically trains outcome-grounded task-progress assessment and integrates the resulting signal into policy optimization.

B Implementation Details

B.1 Prompt

Retrospective Progress Reflection Prompt

```
You have just completed a task on an online shopping website. Now review the trajectory
with the benefit of hindsight.

Your trajectory (4 steps):
Step 1: action="search[wireless mouse under 20 dollars]"   your_estimate=10%
Step 2: action="click[item - Logitech wireless mouse]"   your_estimate=35%
Step 3: action="click[add to cart]"   your_estimate=70%
Step 4: action="click[buy now]"   your_estimate=90%

Final outcome: SUCCEEDED (score=1.00)

Re-assess what fraction of the task was genuinely completed at each step.

Guidelines:
- Step 1 should be near 0 (task just started).
- If SUCCEEDED, the last step should be near 100.
- Values should generally be non-decreasing.

First reason briefly in <think>...</think>, then output exactly 4 integers (0--100) in
<hindsight_progress>v1,v2,v3,v4
</hindsight_progress>.
Example: <think>step 1 started, step 2 found item, step 3 bought it.</think>
<hindsight_progress>0,60,100
</hindsight_progress>
```

C Pilot Study Details

Evaluation setup. We evaluate DeepSeek V4.1-Flash through the official DeepSeek API with thinking disabled, and GPT-5.1 through OpenRouter with Azure as the sole provider. All evaluations use temperature 0 and a maximum of 30 environment actions per task. We fix 256 tasks sampled without replacement from the 500-task WebShop test split using seed 42.

Demonstration construction. For each model, we collect 100 trajectories from the separate training split and retain successful ones for annotation. The same model receives each complete successful trajectory and its terminal reward, and annotates the observation following each action with a percentage and a short rationale. The resulting demonstration pools contain 30 trajectories for

DeepSeek and 32 for GPT-5.1. For each test-time call, we sample 2 trajectories from the corresponding model's pool. Each demonstration contains its instruction and chronologically ordered observation-progress pairs; action fields stored in the trajectory files are not explicitly included in the demonstration prompt to avoid extra hints.

Inference protocol. The baseline predicts actions without an explicit progress-assessment call. Direct assessment receives the current observation and accumulated interaction messages, including previous assessments. Retrospectively guided assessment instead receives the task instruction, current observation, and sampled demonstrations. Both produce a percentage and a short rationale, which are appended to the current observation in a separate action-prediction call. This action call retains the interaction history and previous assessments but does not receive demonstrations.

Random controls. For *Random Progress*, we independently sample a percentage from $\{0, 5, \ldots, 100\}$ at each step and ask the model to generate a task-specific rationale supporting that assigned value. For *Random Demonstration*, we replace demonstration annotations with pre-generated random percentages and fabricated rationales while preserving the instructions, observations, and their temporal order.

Repeated runs and reporting. Figure 1 summarizes runs with 2 seeds. The test tasks and demonstration pools are fixed; the seed initializes demonstration sampling and online random percentages. API and infrastructure failures are retried or resumed rather than counted as task failures. In the qualitative example, the two methods share an action-observation prefix, but their earlier progress messages differ, illustrating a behavioral difference.

C.1 Training Hyper-parameters

(1) SFT Warmup (Phase 1). We construct hindsight-labeled data as described in §4.2:

- **Data source**: Successful trajectories (score ≥ 0.9 for WebShop, success=True for ALFWorld) from pre-trained GiGPO checkpoints
- **Hindsight labeling**: DeepSeek-v4 API (temperature=0) generates hindsight progress sequences
- **Training**: Standard SFT with learning rate 5×10^{-6}, batch size 8, 3 epochs.

(2) RL Training (Phase 2). Following GiGPO (Feng et al., 2026) with modifications for progress reflection:

- **Rollout**: 8 trajectories per task. Episodes are limited to 15 interaction steps for WebShop and Sokoban, and 50 for ALFWorld.
- **PPO hyperparameters**: $\epsilon_{\text{clip}} = 0.2$, $\gamma = 0.95$, and KL coefficient $\beta_{\text{KL}} = 0.01$.
- **Learning rate**: 1×10^{-6}.
- **Training steps**: 120/150 iterations.
- **Progress reward weights**: We use $\beta = 0.1$ for retrospective progress differences, $\alpha = 0.2$ for pointwise online-retrospective alignment, $\alpha_{\text{corr}} = 0.3$ for correlation, and $w_p = 0.2$ for online boundary constraints. The retrospective format reward is $+0.1$ for a valid format and -0.1 otherwise.
- **Advantage estimation**: We use GiGPO with $F_{\text{norm}} = 1$ and step-advantage weight $\omega = 1$. The episode baseline is computed over step samples within each task group, with the episode reward repeated for each valid step.
- **Policy optimization**: All generated tokens in each forward response share the corresponding step advantage. The policy loss is averaged over non-padding response tokens, including reasoning, progress estimates, and actions. We use dual clipping with $c = 3$ for negative advantages.

All RL experiments use full-parameter fine-tuning. We train on PPU using the veRL framework (Sheng et al., 2024).

Reward implementation. Progress percentages are normalized to $[0, 1]$ before reward computation. The correlation reward uses valid online-retrospective pairs and is zero when fewer than three pairs are available. If the retrospective sequence has standard deviation below 0.01, the reward is also zero. Otherwise, an online standard deviation below 0.01 incurs a penalty of -0.3; remaining cases receive α_{corr} times the Pearson correlation.

The retrospective format check requires a `<hindsight_progress>` field containing at least one parseable numeric value. Extra values are truncated, and short sequences are extended by repeating the final value. Parsed percentages are clipped to $[0, 100]$ before normalization. The format check does not require an exact sequence length or validate the reasoning content.

C.2 Setup Details

Checkpoint selection and averaging. To reduce sensitivity to fluctuations in individual evaluations, we report *Avg* over a convergence window comprising ten consecutive evaluation checkpoints, selected using the rule below. *Best* reports the checkpoint with the highest evaluation success rate, together with the task score at that same checkpoint. *Avg* reports equal-weight averages of success rate and task score over a ten-checkpoint reporting window. Starting from the final ten evaluated checkpoints, we shift the window backward by one checkpoint whenever its final success rate is more than 2% below the mean of the preceding nine checkpoints. We stop when this condition no longer holds or the window reaches the beginning of the evaluation sequence.

Multi-seed reporting. Figure 3 presents the learning curves across seeds 0, 1, and 2 for the 1.5B and 3B models. For the 1.5B comparison in Table 5, we first average success rates over the final five evaluation checkpoints of each run, then report the mean and standard deviation across the three seeds. This aggregation is separate from the windows used for the main tables and the progress-metric analysis.

Model size and model-version effects. For every model environment pair, RePro and its baselines use the same backbone and experimental setup. Across tasks, model scale, and version, RePro demonstrates consistent improvement with quantitative variations. The stable training of Qwen2.5-7B without Retrospection Warmup provides further support for our design. It suggests that stronger models can already perform the forward-then-reflect format, whereas warmup primarily bootstraps this behavior for smaller models. Relative to Meta Prompt, RePro improves average success rate by 5.82 percentage points on WebShop and 6.17 points on ALFWorld for Qwen2.5-7B. On WebShop, retaining warmup further improves best success rate by 4.30 points over the variant without warmup, indicating that warmup remains beneficial even when training without it is stable.

We select Qwen2.5 as the principal backbone family because its matched 1.5B, 3B, and 7B checkpoints enable a controlled scaling study (Qwen et al., 2025; Feng et al., 2026). The Qwen3-4B result on Sokoban further provides cross-version evidence.

D Progress Metrics

Evaluation setup. We analyze the baseline's online progress estimates and RePro's retrospective estimates on their respective test-set evaluation trajectories. Let $p_{e,t} \in [0, 100]$ denote the progress estimate at step t of trajectory e, and let T_e denote its length. Successful and failed trajectories are grouped by terminal reward ($r_e \geq 9$ and $r_e < 1$, respectively); intermediate-reward trajectories are excluded from these groups. The definitions below apply at each evaluation checkpoint.

Intermediate Discrimination (IntermDisc). IntermDisc measures the average stepwise progress gap between successful and failed trajectories. We retain the first $H = 15$ steps of each trajectory and pad shorter trajectories with their last valid estimate. Missing estimates within the original sequence remain missing; trajectories without any valid estimate in the retained sequence are excluded. Let $\mu_{\text{s},t}$ and $\mu_{\text{f},t}$ denote the mean available estimates at step t for successful and failed trajectories, respectively. Then

$$\text{IntermDisc} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \left(\mu_{\text{s},t} - \mu_{\text{f},t} \right), \tag{16}$$

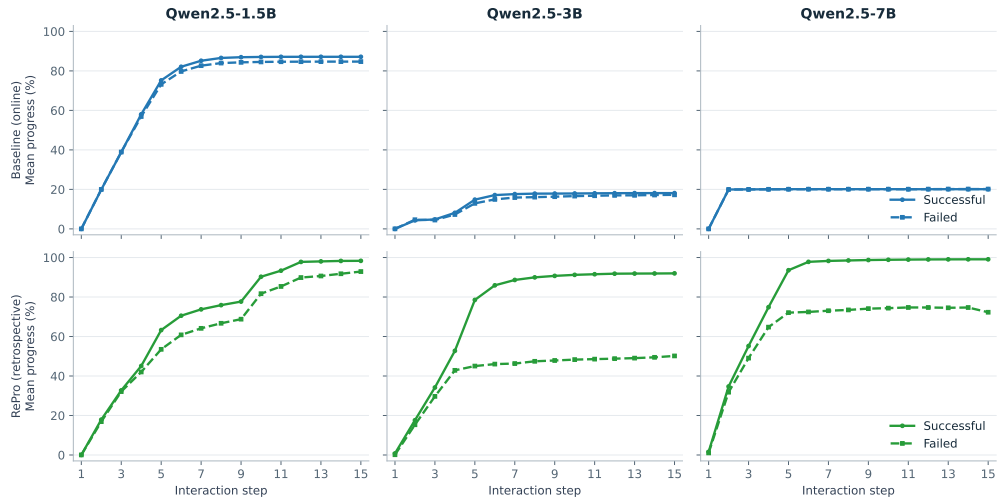


Figure 4: Progress estimates on successful and failed WebShop trajectories. The top row shows online estimates from the Meta Prompt baseline; the bottom row shows retrospective estimates from RePro. Solid and dashed lines denote successful and failed trajectories, respectively. Short trajectories are padded to 15 steps using their last valid estimate. Curves are averaged across checkpoints in the same convergence windows used in Appendix D and Table 4.

where $\mathcal{T} \subseteq \{1, \ldots, H\}$ contains steps with available estimates in both groups. IntermDisc is reported in percentage points. The progress profiles use the same stepwise means and padding procedure.

Temporal R^2. For each trajectory, we fit a linear regression with an intercept from relative step position t/T_e to valid progress estimates. Temporal R^2 is the resulting coefficient of determination, equivalently

$$R_e^2 = \mathrm{corr}^2\left(t/T_e,\, p_{e,t}\right), \tag{17}$$

computed over valid steps. Trajectories with fewer than three valid estimates or an undefined R^2 are omitted from this metric. This statistic measures linear association, not its direction or progress accuracy.

Monotonicity. For the remaining metrics, let $q_{e,1}, \ldots, q_{e,n_e}$ be the valid progress estimates in their original order, and define $d_{e,j} = q_{e,j+1} - q_{e,j}$. We use the original sequences without truncation or padding and exclude trajectories with fewer than two valid estimates. Monotonicity is the fraction of non-decreasing transitions:

$$\mathrm{Mono}_e = \frac{1}{n_e - 1} \sum_{j=1}^{n_e - 1} \mathbf{1}[d_{e,j} \geq 0]\,. \tag{18}$$

Consecutive valid estimates are compared even when missing estimates occur between them.

Decline Rate. A trajectory exhibits a decline if at least one transition decreases by more than 10 percentage points:

$$D_e = \mathbf{1}\left[\min_{1 \leq j < n_e} d_{e,j} < -10\right]. \tag{19}$$

Decline Rate is the fraction of eligible trajectories with $D_e = 1$.

Plateau Rate. A trajectory exhibits a plateau if at least half of its valid transitions change by less than one percentage point:

$$P_e = \mathbf{1}\left[\frac{1}{n_e - 1} \sum_{j=1}^{n_e - 1} \mathbf{1}[|d_{e,j}| < 1] \geq \frac{1}{2}\right]. \tag{20}$$

Table 6: Representative successful and unsuccessful progress assessments on WebShop.

Case	Trajectory and generated progress	Interpretation
Successful progress tracking	Search (0%) → click product (10%) → select color blue/gray (33.33%) → select size 1'9" × 7'2" (50%) → reconfirm color (50%) → reconfirm size (50%) → buy after all constraints are aligned (100%).	Progress increases with task advancement, remains unchanged during repeated confirmation, and reaches 100% only upon completion.
Failure recognition without recovery	Search (0%) → recognize that no result satisfies all constraints (0%) → temporarily overestimate progress (100%) → identify the selected item as unsuitable (0%) → remain in the unresolved failure state (0%).	The estimate recovers from a temporary overestimation and ultimately reflects failure, although the policy does not recover the task.
False-positive progress	Search (0%) → mistake a Henley T-shirt for a dress shirt (100%) → select an incorrect color (0%) → select size 2X and color navy (50%) → treat the Henley T-shirt as satisfying the request (100%).	Misinterpreting a core product attribute produces overconfident progress estimates and an incorrect purchase.

Plateau Rate is the fraction of eligible trajectories with $P_e = 1$.

Aggregation. At each checkpoint, temporal R^2, monotonicity, decline, and plateau are averaged over eligible trajectories separately for successful and failed outcomes. Undefined values are excluded. Table 4 reports temporal R^2 multiplied by 100 and the three remaining metrics as percentages.

We then average checkpoint-level results within each run's convergence window, assigning equal weight to each checkpoint. For a run with K evaluated checkpoints in training order, the reference success rate is the mean over the final $\max(2, \lfloor K/3 \rfloor)$ checkpoints. The convergence window begins at the first checkpoint whose three-checkpoint forward average success rate reaches at least 80% of this reference, and includes all subsequent checkpoints. If no such checkpoint exists, we use the reference tail instead. Unavailable checkpoint-level values are omitted. The plotted progress profiles use the same checkpoint windows and equal-weight averaging.

E Qualitative Progress-Assessment Cases

To complement the aggregate progress metrics, we present three representative WebShop trajectories generated by RePro in Table 6, covering successful progress tracking and two distinct failure patterns. These examples illustrate model behavior rather than quantify the frequency of each pattern.

The successful case shows that progress estimates can reflect meaningful task transitions rather than increase mechanically with trajectory length: repeated confirmation actions retain the same progress value. The second case separates progress assessment from action recovery. The agent eventually recognizes the failed state but cannot find a successful alternative. The third case exposes a key limitation: progress assessment remains conditional on the agent's semantic understanding of the environment, so an incorrect interpretation can lead to confidently incorrect progress.

F Limitations

RePro represents task progress through verbalized assessments, making its estimates and rationales explicit; alternative representations, such as latent states or hidden-vector supervision, remain unexplored. RePro also requires separate supervised fine-tuning and policy optimization stages, adding training and pipeline overhead. Future work could explore joint or interleaved training to reduce this overhead while preserving stable progress assessment. Our evaluation covers WebShop, ALFWorld,

and Sokoban, while resource and infrastructure constraints limit exploration of broader settings such as operating system automation and open-ended embodied interaction. Finally, these benchmarks provide no ground-truth intermediate progress labels. RePro's retrospective estimates use completed trajectories and outcomes but may still misinterpret task states. Future work could assess their calibration using human annotations or stronger-model judgments, accounting for the uncertainty and potential biases of these references.