PROCREDIT: FROM OUTCOME REWARDS TO PROGRESS CREDIT IN AGENTIC REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon agentic tasks require an agent to modify an environment through a sequence of tool calls, with success determined by the final state. The standard recipe assigns a single outcome reward at the end and compares trajectories sampled for the same task. As a result, a group with no successful trajectory yields no training signal, failed attempts cannot be told apart by how close they came to completion, and turns that advance the task receive the same credit as turns that only query the environment. Prior work refines the unit of comparison from the trajectory to the step, or trains a reward model to supply intermediate signal: the former still derives its signal from final success alone, and the latter estimates it with a model. We observe that the acceptance checks that decide success can also be run on intermediate states, so progress is as verifiable as the outcome. We propose ProCredit, which turns this verified progress into credit: it reruns the acceptance checks after each turn, rewards the turn by its change in progress, and uses these rewards to assign credit both across attempts at the same task and across the turns within a trajectory. Starting from Qwen3.5 base models at three scales on AppWorld, ProCredit outperforms outcome-reward baselines and progress-based baselines in task completion rate at every scale on both test sets, exceeding the strongest outcome-reward baseline by 4.1 percentage points at 4B, and results in a second environment show the same direction of improvement. Ablations show that adding the final progress to the trajectory score alone does not improve performance: the gain comes from crediting progress to the turn where it occurs.

1 INTRODUCTION

Language model agents tackle long-horizon tasks through dozens of operations across tools, applications, and services until the environment satisfies a user's request (Trivedi et al., 2024; Yao et al., 2024). We study tasks in which an agent gradually modifies the environment through tool calls and success depends on the final state. Consider a request to move a wake-up alarm earlier and disable the other alarms: acceptance consists of executable checks, each testing a concrete fact about the environment, such as whether the wake-up alarm has the requested time.

A common way to train such agents is reinforcement learning with verifiable rewards (DeepSeek-AI et al., 2025): sample multiple trajectories for the same task, assign each a terminal outcome reward, and compare rewards within the group (Shao et al., 2024; Ahmadian et al., 2024; Yu et al., 2025). It has been applied to search, tool use, and tasks spanning applications (Jin et al., 2025; Qian et al., 2025; Chen et al., 2025). An outcome reward reduces an entire attempt to one number and discards three kinds of information. First, in groups with no successful trajectory, every advantage is zero, so the task contributes no gradient at that update. Second, in mixed groups, a failed trajectory that completes half the task receives the same negative advantage as one that completes none of it. Third, a turn that advances the task shares its advantage with a turn that only queries the environment. The first two losses concern how much progress is made; the third concerns when it is made (Figure 1).

Prior work recovers intermediate signals in two ways. One refines the unit of comparison from the trajectory to the step (Feng et al., 2025; Li et al., 2025a), but the signal still comes from final success alone, and two trajectories can only be compared at a step where they reach the same state. The other

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.

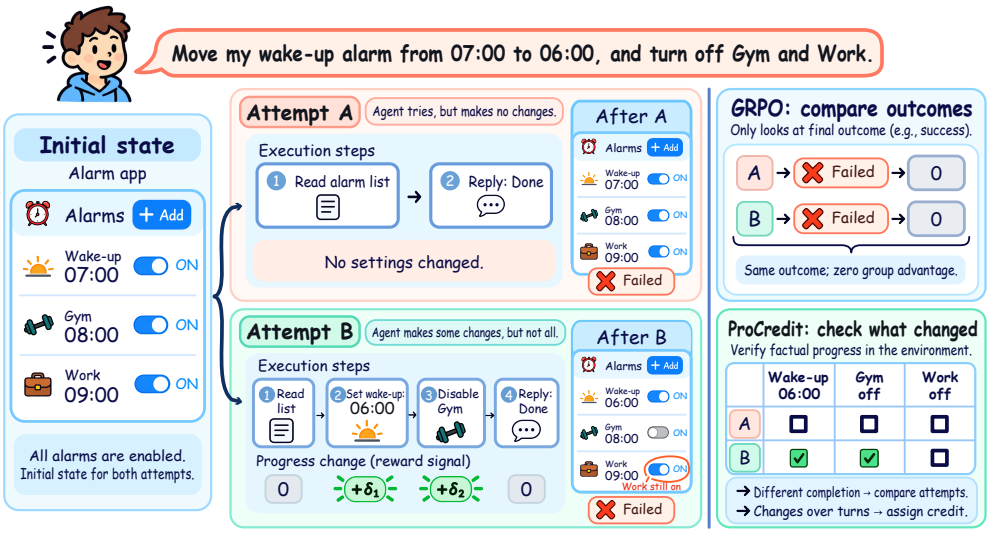

Figure 1: The same failed outcome hides different progress. Attempt A only reads the alarm list and declares completion; Attempt B moves the wake-up alarm earlier and disables Gym but leaves Work enabled. Both receive zero outcome reward and zero GRPO advantage; rerunning the checks after each turn separates them, with rewards δ_1 and δ_2 for changing the time and disabling Gym.

trains a separate model to supply the signal (Xi et al., 2025; Zhou et al., 2026), which is estimated rather than verified in the environment. The acceptance checks that decide success can also run on intermediate states: progress is as verifiable as the outcome, and no separate model is needed.

We propose ProCredit; Figure 2 shows its training procedure. After each turn during training, it reruns the task's acceptance checks, records the fraction passed as progress, and uses the change between consecutive turns as the turn's reward. Credit is assigned at two levels. Across attempts at the same task, trajectories are compared by their total score, so failed attempts are separated by completion; within a trajectory, turns are compared by their turn-level returns, so a turn that advances the task earns more credit than every turn after it. All-fail groups therefore still carry signal, and credit within a trajectory follows its own progress without requiring another trajectory to reach the same state. The procedure uses only the task's existing checks; progress is read only during training.

We train Qwen3.5 base models at three scales on AppWorld and evaluate on its two test sets, one with applications seen in training and one containing applications unseen in training. ProCredit achieves the highest task completion rate at every scale on both test sets; the direction holds on ToolSandbox, and function calling on BFCL does not decline. Two ablations locate the gain. Adding final progress to the trajectory score or spreading it over tokens does not improve at any scale, so the gain comes from crediting progress at the turn where it occurs, not only from how much is completed at the end. Discounting the turn-level return shortens trajectories and loses hard tasks along with them; undiscounted returns are better at all three scales.

2 Related Work

Reinforcement learning for long-horizon agentic tasks. Reinforcement learning allows language model agents to learn directly from environment interactions in search and tool use (Jin et al., 2025; Qian et al., 2025), text games and embodied tasks (Wang et al., 2025c; Feng et al., 2025), and tasks spanning applications in environments such as AppWorld (Chen et al., 2025; Wang et al., 2026; Li et al., 2025a). These methods train with relative advantages across trajectories sampled for the same task, with their main reward provided only at the end. We retain the same environment setting and optimization objective, and change where the reward is read.

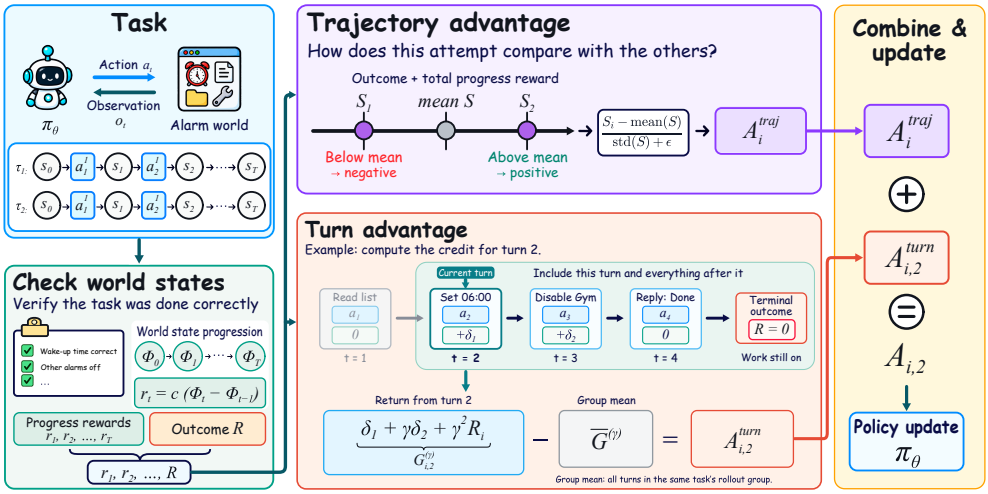

Figure 2: Overview of ProCredit. Each turn's change in progress becomes its reward; the rewards enter a trajectory-level advantage that compares attempts at the same task and a turn-level advantage that centers each turn's onward return. The lower panel traces Attempt B from Figure 1: the turn that changes the time receives exactly δ_1 more credit than the turn that disables Gym.

Sources of intermediate signals. Intermediate signals can come from learned reward models (Xi et al., 2025; Lee et al., 2026; Liu et al., 2025a; Wang et al., 2025b; Kim et al., 2026) or a post-hoc critic (Tan et al., 2026), from the policy's own belief updates, as in the information-gain rewards of IGPO (Wang et al., 2025a), from checklists scored by a language model, as in CM2 (Zhang et al., 2026), from reference trajectories synthesized by language models, as in ToolVerse/TARA (Zhou et al., 2026), or from verifiers: VPR (Yuan et al., 2026) verifies individual intermediate actions. ProCredit belongs to the last category but reads the verifier differently: it reruns the full acceptance checks after every turn, defines progress as the fraction passed, and derives credit from the observed change between consecutive turns, without annotations, reward models, or reference trajectories. Because the signal is read from the environment state, a query that changes nothing earns nothing, whereas an information-gain reward pays for it.

Credit assignment in group-based reinforcement learning. GRPO (Shao et al., 2024), RLOO (Ahmadian et al., 2024), and DAPO (Yu et al., 2025) assign the same advantage to all turns of a trajectory. Advantages are zero in all-fail groups; DAPO addresses this through resampling, while ProGPO (Yang et al., 2026) uses first-visit coverage as a fallback. Turn-level credit can come from hand-designed turn rewards, as in MT-GRPO (Wei et al., 2025), which must be redesigned for each environment, or from a value function, as in Turn-PPO (Li et al., 2025b), which like a reward model is a learned estimate. Another route refines the unit of comparison: GiGPO (Feng et al., 2025) and HGPO (He et al., 2026) group identical environment states across trajectories to compute step-level advantages, and G2PO and SALT propagate credit through state-transition graphs. These methods redistribute outcome rewards, and their step comparisons depend on states recurring across trajectories. ProCredit obtains turn rewards from the task's acceptance checks, which order the turns within a trajectory without coincident states across trajectories, and the same signal makes trajectories in all-fail groups comparable by completion.

3 Method: ProCredit

ProCredit checks the environment state after each tool-execution turn during training, converts changes in progress into turn rewards, and uses these rewards at two levels: the total trajectory reward compares attempts at the same task, and returns from individual turns onward assign credit within each trajectory. The policy update uses the sum of the two advantages, as shown in Figure 2.

3.1 PRELIMINARIES

A task consists of repeated turns of action generation, tool execution, and observation. Let g denote the task and h_t the history visible to the model before turn t. The policy samples an action $a_t \sim \pi_\theta(\cdot \mid h_t, g)$, after which the environment enters state s_t and returns observation o_t. The model receives observations rather than the full state, so these tasks are typically modeled as partially observable Markov decision processes (Chen et al., 2025; Zhang et al., 2025); the state s_t is used for verification only during training. An attempt ends when the model declares completion or reaches a limit of h turns; its actual length is T, and the environment then assigns an outcome reward $R \in \{0, 1\}$. We denote the group of trajectories sampled for task g by $\mathcal{G}_g$, index trajectories by i and other members of the group by j, and omit i when discussing a single trajectory.

Group relative policy optimization (GRPO) standardizes outcome rewards within the group to obtain a trajectory advantage:

$$A_i = \frac{R_i - \mathrm{mean}_{j \in \mathcal{G}_g} R_j}{\mathrm{std}_{j \in \mathcal{G}_g} R_j + \epsilon}. \tag{1}$$

Let $a_{i,t,k}$ be token k of the action at turn t in trajectory i, with importance ratio $\rho_{i,t,k} = \pi_\theta(a_{i,t,k} \mid h_{i,t}, a_{i,t,<k}, g)/\pi_{\theta_{\mathrm{old}}}(a_{i,t,k} \mid h_{i,t}, a_{i,t,<k}, g)$. The policy is updated with a clipped objective (Schulman et al., 2017), using advantage $A_{i,t}$ at turn t:

$$\mathcal{J}(\theta) = \mathbb{E}\Big[\tfrac{1}{\sum_{i,t}|a_{i,t}|} \textstyle\sum_{i,t}\sum_{k=1}^{|a_{i,t}|} \min\big(\rho_{i,t,k}A_{i,t},\ \mathrm{clip}(\rho_{i,t,k}, 1-\varepsilon, 1+\varepsilon)A_{i,t}\big)\Big]. \tag{2}$$

GRPO uses the same advantage for every turn of a trajectory, $A_{i,t} = A_i$. Following LOOP (Chen et al., 2025), we omit a KL term; the loss is averaged over all generated tokens and excludes tool responses.

3.2 VERIFIABLE TASK PROGRESS

Task acceptance consists of executable checks, each testing a concrete fact about the environment. Let $\mathcal{C}_g$ be the check set, with size $K_g > 0$, and let $C_{g,m}(s) \in \{0, 1\}$ equal one if check m passes in state s. We define task progress as the fraction of checks passed:

$$\Phi_g(s) = \frac{1}{K_g} \sum_{m=1}^{K_g} C_{g,m}(s). \tag{3}$$

Progress after turn t is $\Phi_t = \Phi_g(s_t) \in [0, 1]$, and Φ_0 is progress before any action. In our environments, we include only checks that fail in the initial state, so $\Phi_0 = 0$.

Progress depends on the environment state, rather than whether a tool call returns successfully: a query leaves progress unchanged, a modification raises it only if more acceptance checks pass, and an incorrect modification can invalidate previously passed checks and reduce it. In Figure 1, reading the list leaves progress at zero, changing the time and disabling Gym each pass one check, and Work remains enabled, so the outcome reward R, recorded separately from progress, is zero. Progress is read only during training.

3.3 PROGRESS REWARDS AND TWO LEVELS OF CREDIT

Progress rewards. The reward at turn t is the change in progress between consecutive states, scaled by $c > 0$:

$$r_t = c\,(\Phi_t - \Phi_{t-1}). \tag{4}$$

Turns that advance the task receive positive reward, turns that undo progress negative reward, and turns that leave the checks unchanged, including queries, zero. The trajectory score combines the outcome reward with the sum of progress rewards:

$$S = R + \sum_{t=1}^{T} r_t = R + c\,(\Phi_T - \Phi_0). \tag{5}$$

The score depends only on initial and final progress, as in potential-based reward shaping (Ng et al., 1999) with potential $c\,\Phi_g(s)$. Unlike Ng et al. (1999), we do not set terminal potentials to zero, so

the score retains final completion. Successful trajectories still score strictly higher than any failed trajectory, and the set of optimal policies is unchanged when the task is solvable (Appendix G), so progress refines the ranking only among attempts with the same outcome; the turn-level term below uses when that progress was made.

Trajectory-level advantage. We replace R_i in GRPO's within-group standardization with the trajectory score S_i:

$$A_i^{\text{traj}} = \frac{S_i - \text{mean}_{j \in \mathcal{G}_g} S_j}{\text{std}_{j \in \mathcal{G}_g} S_j + \epsilon}. \tag{6}$$

This term treats all turns of a trajectory alike and expresses how the attempt compares with the other attempts at the same task.

Turn-level advantage. Credit at turn t uses the discounted return from that turn onward:

$$G_{i,t}^{(\gamma)} = \gamma^{T_i - t} R_i + \sum_{u=t}^{T_i} \gamma^{u-t} r_{i,u}. \tag{7}$$

ProCredit uses $\gamma = 1$. An attempt has at most h turns and bounded rewards, so its undiscounted return is finite and does not require discounting to converge. The return has the closed form

$$G_{i,t} = R_i + c\,(\Phi_{i,T_i} - \Phi_{i,t-1}), \tag{8}$$

which is the outcome reward plus the progress still to be earned from this turn on. We examine the effect of discounting in Section 4.3.2. Let $\mathcal{U}_g$ contain all turns of all trajectories for task g. We center turn returns by their mean over $\mathcal{U}_g$, without dividing by a standard deviation:

$$A_{i,t}^{\text{turn}} = G_{i,t} - \text{mean}_{(j,u) \in \mathcal{U}_g} G_{j,u}. \tag{9}$$

Appendix H explains why turns with different histories can share this reference value. The final advantage is the sum of the two terms:

$$A_{i,t} = A_i^{\text{traj}} + A_{i,t}^{\text{turn}}. \tag{10}$$

Roles of the two terms. The trajectory-level term determines whether an attempt is reinforced or suppressed as a whole; the turn-level term adds a tilt within the trajectory. When $\gamma = 1$, the difference in credit between adjacent turns is exactly the progress reward of the earlier turn:

$$A_{i,t} - A_{i,t+1} = r_{i,t}. \tag{11}$$

A turn that advances the task receives more credit than the next turn by exactly the reward for its own progress, and a query that leaves the state unchanged receives the same credit as the next turn, as in the lower panel of Figure 2. This tilt depends only on progress differences, not on success or failure, so turns can still receive different credit in groups where every trajectory succeeds (Appendix B gives a complete numerical example). The policy is then updated with the objective in Equation (2), using $A_{i,t}$ in place of the shared A_i; all tokens generated within a turn share that turn's advantage.

4 Experiments

4.1 Experimental setup

Environments. AppWorld requires agents to complete tasks across applications through multiple turns of tool use, with executable unit tests determining task completion. We train on 90 tasks, use a development set of 57 tasks, and evaluate on test-normal (168 tasks) and test-challenge (417 tasks); the latter includes applications unseen during training. We report task goal completion (TGC) and scenario goal completion (SGC); a scenario is complete only when all its tasks are complete. ToolSandbox (Lu et al., 2025) is a conversational tool environment with state-modifying tools and milestone-based evaluation. Of its 101 tasks, we use 52 for training and 21 for evaluation; the remaining tasks are excluded because the base model completes all of them or they have fewer than two milestones.

Table 1: TGC and SGC (%) on AppWorld Test-Normal and Test-Challenge.

	Test-Normal (N)					
	4B		9B		35B	
Method	**TGC ↑**	**SGC ↑**	**TGC ↑**	**SGC ↑**	**TGC ↑**	**SGC ↑**
Base model	17.78	4.24	19.49	4.91	37.28	12.28
SFT	19.87	6.47	31.70	6.70	49.78	21.43
Outcome rewards only						
GRPO	$66.32_{\pm 1.27}$	$42.34_{\pm 3.17}$	$72.79_{\pm 0.96}$	$51.64_{\pm 1.95}$	$75.00_{\pm 0.61}$	$57.37_{\pm 0.85}$
RLOO	$65.87_{\pm 3.76}$	$42.04_{\pm 5.36}$	$72.72_{\pm 3.10}$	$51.26_{\pm 4.22}$	$\underline{78.45}_{\pm 0.76}$	$58.53_{\pm 2.59}$
DAPO	$\underline{67.46}_{\pm 5.62}$	$\underline{43.38}_{\pm 6.19}$	$\underline{76.96}_{\pm 4.53}$	$\underline{56.70}_{\pm 3.98}$	$77.90_{\pm 1.24}$	$\mathbf{61.23}_{\pm 2.46}$
SALT	$65.60_{\pm 3.92}$	$40.25_{\pm 6.26}$	$74.68_{\pm 2.42}$	$54.61_{\pm 4.22}$	$78.27_{\pm 2.45}$	$58.31_{\pm 3.38}$
Outcome rewards + progress						
GRPO-Φ	$65.40_{\pm 4.12}$	$41.67_{\pm 6.25}$	$74.78_{\pm 2.54}$	$54.24_{\pm 4.68}$	$76.41_{\pm 1.40}$	$58.48_{\pm 2.70}$
GiGPO	$64.53_{\pm 3.78}$	$42.26_{\pm 6.56}$	$69.77_{\pm 2.21}$	$47.25_{\pm 3.21}$	$56.99_{\pm 3.28}$	$35.49_{\pm 2.88}$
ProCredit	$\mathbf{71.53}_{\pm 0.48}$	$\mathbf{48.88}_{\pm 0.45}$	$\mathbf{78.15}_{\pm 2.61}$	$\mathbf{59.82}_{\pm 3.95}$	$\mathbf{78.65}_{\pm 0.85}$	$\underline{59.60}_{\pm 1.35}$
	Test-Challenge (C)					
	4B		9B		35B	
Method	**TGC ↑**	**SGC ↑**	**TGC ↑**	**SGC ↑**	**TGC ↑**	**SGC ↑**
Base model	10.67	1.80	12.59	3.78	27.82	8.81
SFT	9.95	2.52	20.14	5.76	31.59	8.63
Outcome rewards only						
GRPO	$46.50_{\pm 0.66}$	$24.46_{\pm 0.78}$	$60.77_{\pm 2.35}$	$37.23_{\pm 2.03}$	$\underline{66.73}_{\pm 3.66}$	$\underline{43.35}_{\pm 2.96}$
RLOO	$48.28_{\pm 4.55}$	$26.20_{\pm 3.06}$	$60.13_{\pm 4.87}$	$35.61_{\pm 6.08}$	$63.09_{\pm 0.52}$	$41.49_{\pm 2.37}$
DAPO	$\underline{49.06}_{\pm 7.43}$	$\underline{27.94}_{\pm 7.11}$	$\underline{63.13}_{\pm 2.39}$	$\underline{39.93}_{\pm 2.12}$	$64.31_{\pm 1.58}$	$42.63_{\pm 2.59}$
SALT	$48.50_{\pm 3.28}$	$25.66_{\pm 2.22}$	$58.73_{\pm 2.52}$	$35.73_{\pm 3.40}$	$65.21_{\pm 4.31}$	$43.11_{\pm 4.28}$
Outcome rewards + progress						
GRPO-Φ	$47.82_{\pm 2.38}$	$25.48_{\pm 2.44}$	$62.15_{\pm 2.99}$	$38.85_{\pm 2.77}$	$65.05_{\pm 1.03}$	$\underline{43.35}_{\pm 1.65}$
GiGPO	$43.69_{\pm 4.33}$	$22.96_{\pm 3.54}$	$57.89_{\pm 2.43}$	$32.67_{\pm 3.10}$	$41.43_{\pm 3.27}$	$17.99_{\pm 3.53}$
ProCredit	$\mathbf{52.96}_{\pm 0.50}$	$\mathbf{29.26}_{\pm 0.45}$	$\mathbf{63.85}_{\pm 0.45}$	$\mathbf{41.07}_{\pm 0.73}$	$\mathbf{68.11}_{\pm 3.95}$	$\mathbf{45.68}_{\pm 4.40}$

Models and training. We use Qwen3.5 (Qwen Team, 2026) models: Qwen3.5-4B, Qwen3.5-9B, and Qwen3.5-35B-A3B (abbreviated as 35B). Reinforcement learning starts from the base models, and comparisons are made within each model size. ProCredit uses $c = 0.5$ and $\gamma = 1$. For each reinforcement learning method, we report the mean and standard deviation over three independent training runs.

Comparison methods. SFT uses trajectories generated on AppWorld. Outcome-only methods include GRPO (Shao et al., 2024), RLOO (Ahmadian et al., 2024), DAPO (Yu et al., 2025), and SALT (Li et al., 2025a). Methods that use progress include GRPO-Φ, GiGPO (Feng et al., 2025), and ProCredit. GRPO-Φ adds final progress to the trajectory score, giving the same score as ProCredit, but uses only trajectory-level advantages. GiGPO uses the same progress rewards as ProCredit and groups states by the set of passed checks; the main table uses its default discount factor, $\gamma = 0.95$. SALT also defines state equivalence classes by the set of passed checks.

4.2 Main results

Overall performance. Table 1 groups methods by their reward source and reports results on both test sets. ProCredit achieves the highest TGC at all three model sizes on both test sets, and the highest SGC in every column except 35B test-normal, where DAPO is slightly higher. Its margin over the strongest outcome-only method is largest at 4B, 4.1 and 3.9 TGC points on test-normal

and test-challenge with the smallest standard deviations in those columns, and narrows to 1 to 3 points at 9B; at 35B, RLOO, DAPO, SALT, and ProCredit all score around 78 in test-normal TGC, and ProCredit leads by 1 to 2 points on test-challenge. GiGPO uses the same progress rewards yet scores below GRPO at 9B and 35B; Section 4.3.2 traces this to its default discount factor.

Same progress, different credit. GRPO-Φ and ProCredit assign the same score to a given trajectory and differ only in the turn-level term. ProCredit scores higher at all three model sizes on both test sets: its test-normal TGC is higher by 6.1, 3.4, and 2.2 points at 4B, 9B, and 35B, and its test-challenge TGC by 5.1, 1.7, and 3.1 points. GRPO-Φ itself does not consistently outperform GRPO. Adding progress to the trajectory score alone is insufficient; the difference comes from when progress is credited. Appendix E plots the training and development curves of the three methods.

Table 2: Success rates (%) on ToolSandbox.

Model	Base model	GRPO	ProCredit
4B	63.1	$81.0_{\pm 1.0}$	$\mathbf{84.5}_{\pm 0.5}$
9B	72.0	$85.7_{\pm 1.8}$	$\mathbf{89.3}_{\pm 1.9}$
35B	79.8	$89.9_{\pm 2.6}$	$\mathbf{92.3}_{\pm 2.9}$

A second environment. Table 2 reports success rates on ToolSandbox. ProCredit outperforms GRPO at each model size. On the single-turn and multi-turn function-calling categories evaluated in BFCL (Patil et al., 2025), ProCredit's scores after training remain close to those of the base models; Appendix F reports the full results.

4.3 Core ablations

Table 3 reports TGC and SGC on test-normal: the upper block varies the credit unit with undiscounted returns, and the lower block discounts returns with $\gamma = 0.95$. Rows with the same configuration as the main table reuse its results.

4.3.1 Rewards and credit assignment

Credit granularity. The upper block of Table 3 assigns the same progress signal at different granularities. GRPO and GRPO-Φ credit entire trajectories, without and with final completion in the score. GiGPO with $\gamma = 1$ groups turns by state anchors and assigns credit at milestones. ProCredit (turn) assigns credit per turn. ProCredit (token) distributes each turn's progress reward linearly across token positions: for token k among the turn's n tokens, the return includes only the remaining portion of that turn's progress reward, giving $G_{i,t} - r_{i,t}\, k/n$, with grouping and centering unchanged. This variant introduces no token-level environmental feedback; it only dilutes the same reward.

Turn-level credit works best. ProCredit (turn) exceeds ProCredit (token) in TGC by 8.3 points at 4B and 3.6 points at 9B, with larger gaps in SGC. The token variant falls below GRPO at 4B, exceeds it at 9B with much larger variation across runs, and matches the turn variant in TGC at 35B, where its slightly higher mean SGC lies within its own variability. GiGPO with $\gamma = 1$ uses the same progress rewards but scores below ProCredit (turn) at all three model sizes; their turn-level advantages differ only by an anchor-group mean term, as shown in Appendix H.

4.3.2 Discount factor

The discount factor has the largest individual effect. GiGPO adopts $\gamma = 0.95$ without an ablation, whereas GEM (Liu et al., 2025b) and a practical guide to multi-turn agent reinforcement learning (Wang & Ammanabrolu, 2025) argue that $\gamma < 1$ is incompatible with group-based multi-turn training. In turn-level credit assignment, γ sets how much credit for early turns decays with each subsequent turn, and hence the model's preference over trajectory length. The lower block of Table 3 discounts both ProCredit and GiGPO. Changing ProCredit from $\gamma = 0.95$ to $\gamma = 1$ increases TGC by approximately 3, 10, and 13 points at 4B, 9B, and 35B; the gap grows with model size. GiGPO also scores higher without discounting at 9B and 35B, which accounts for most of

Table 3: Ablations on Test-Normal (%): the same progress signal credited at different granularities, with undiscounted returns (upper block) and with $\gamma = 0.95$ (lower block).

Method	Credit unit	4B		9B		35B	
		TGC ↑	SGC ↑	TGC ↑	SGC ↑	TGC ↑	SGC ↑
Credit granularity, undiscounted returns							
GRPO	Trajectory	$66.32_{\pm 1.27}$	$42.34_{\pm 3.17}$	$72.79_{\pm 0.96}$	$51.64_{\pm 1.95}$	$75.00_{\pm 0.61}$	$57.37_{\pm 0.85}$
GRPO-Φ	Trajectory	$65.40_{\pm 4.12}$	$41.67_{\pm 6.25}$	$74.78_{\pm 2.54}$	$54.24_{\pm 4.68}$	$76.41_{\pm 1.40}$	$58.48_{\pm 2.70}$
GiGPO	Milestone	$64.66_{\pm 5.09}$	$39.73_{\pm 6.61}$	$\underline{75.30}_{\pm 1.95}$	$\underline{54.84}_{\pm 3.23}$	$74.40_{\pm 5.17}$	$56.25_{\pm 6.22}$
ProCredit (token)	Token	$63.19_{\pm 3.69}$	$38.54_{\pm 4.72}$	$74.60_{\pm 4.84}$	$51.86_{\pm 8.97}$	$\underline{78.47}_{\pm 3.59}$	$\mathbf{61.61}_{\pm 5.03}$
ProCredit (turn)	Turn	$\mathbf{71.53}_{\pm 0.48}$	$\mathbf{48.88}_{\pm 0.45}$	$\mathbf{78.15}_{\pm 2.61}$	$\mathbf{59.82}_{\pm 3.95}$	$\mathbf{78.65}_{\pm 0.85}$	$\underline{59.60}_{\pm 1.35}$
Discounted returns, $\gamma = 0.95$							
GiGPO	Milestone	$64.53_{\pm 3.78}$	$42.26_{\pm 6.56}$	$69.77_{\pm 2.21}$	$47.25_{\pm 3.21}$	$56.99_{\pm 3.28}$	$35.49_{\pm 2.88}$
ProCredit (turn)	Turn	$\underline{68.48}_{\pm 2.53}$	$\underline{45.24}_{\pm 5.61}$	$68.20_{\pm 5.66}$	$45.98_{\pm 8.65}$	$65.13_{\pm 7.37}$	$44.64_{\pm 8.12}$

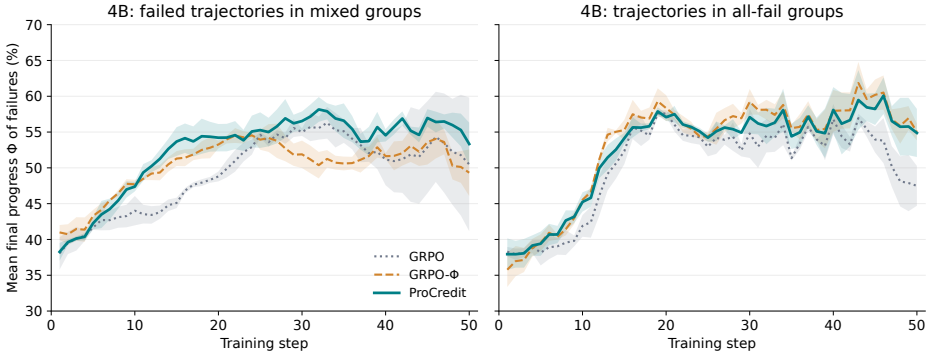

Figure 3: Final progress Φ_T of failed trajectories during 4B training, in mixed groups (left) and all-fail groups (right).

its deficit in Table 1, although it remains below ProCredit at all three model sizes. Appendix D examines discounting only selected types of sampled groups.

Why discounting reduces performance. With $\gamma < 1$, the outcome-reward component of a turn's credit is $\gamma^{T-t}R$: a turn at the same position receives strictly less credit in a longer trajectory, independently of the task. Within an all-success group for the same task, the shortest trajectory therefore almost always receives the highest credit, and the model learns to use fewer turns. After training, ProCredit with $\gamma = 0.95$ produces trajectories that are 9% and 20% shorter on average than GRPO at 4B and 9B, whereas $\gamma = 1$ yields lengths close to GRPO. The shorter the trajectories are pushed, the more accuracy is lost, and the losses concentrate on medium and hard tasks, whose required steps discounting removes. Discounting thus adds a *length tax*, a reduction in earlier credit caused by additional turns, and ProCredit uses $\gamma = 1$. Appendix C states this tax as a proposition and reports credit statistics from actual training trajectories and results by task difficulty.

4.4 Analysis

Progress differences among failed attempts. We call a sampled group all-success, mixed, or all-fail according to whether all, some, or none of its trajectories succeed. In ProCredit's training trajectories, 54.9% and 38.6% of all-fail groups at 4B and 9B, respectively, contain distinguishable progress differences. These groups provide no gradient under outcome-only rewards but still provide learning signals under ProCredit. Figure 3 compares the final progress of failed trajectories under GRPO, GRPO-Φ, and ProCredit in two group types. In mixed groups, both progress-based methods produce failed trajectories with higher completion than GRPO early in training; GRPO-Φ declines during the middle and later stages, while ProCredit remains higher. In all-fail groups, both progress-based methods exceed GRPO later in training and differ little from each other.

Output entropy after tool feedback. The trained models respond more distinctly to progress and error feedback than GRPO: output entropy is lower after turns that advance the task, while the entropy increase after errors is preserved. Appendix A provides the entropy curves, computation details, and results at all three model sizes.

5 Discussion

ProCredit requires task progress to be measurable at intermediate states, which holds whenever the environment state is accessible and the acceptance checks can be rerun. AppWorld's unit tests and ToolSandbox's milestones provide examples; test cases for coding tasks and tasks evaluated by their final database state also fit this setting, and other environments can use an intermediate-state verifier. The checks also set the granularity of progress: more checks passed across more turns let turn-level credit distinguish more turns, whereas with one or two checks, or all checks passed on the final turn, the turn-level term is constant within a trajectory and the method reduces to a GRPO-style comparison of trajectory scores.

Progress must come from the environment rather than the policy's own predictions: when a model predicts its own dense signal and is rewarded for it, group normalization lets the policy improve that prediction without increasing task success (Wang, 2026). ProCredit's progress is determined by the environment state, so the policy can obtain it only by changing that state.

The gains narrow at larger model sizes, consistent with the information supplied by progress: larger models produce fewer all-fail groups and failed attempts, leaving less missing information for progress to recover. In harder environments, where mixed and all-fail groups are as common for a larger model as they are for the 4B model here, progress has correspondingly more to supply. Our current progress measure is the fraction of passed checks, treating checks as equally weighted and unordered; future work will extend ProCredit to larger models and harder environments, and adapt progress measurement to those where a verifier decomposes final acceptance into intermediate checks.

6 Conclusion

Outcome rewards reduce a long-horizon attempt to one number and discard how much progress was made and when. ProCredit runs the acceptance checks on intermediate states during training, reads progress from the environment state, and uses its change to assign trajectory-level and turn-level credit. On AppWorld it achieves the highest task completion rate at three model sizes on both test sets, including applications unseen in training. The benefit comes from crediting progress to the turn where it occurs: adding progress only to trajectory scores or moving to token-level credit does not improve performance, while discounting credit for early turns reduces it.

Reproducibility statement

Code and configurations for training and evaluation are available at `https://anonymous.4open.science/r/ProCredit-8886`. All experiments use publicly available base models and environments: Qwen3.5 base models, AppWorld, ToolSandbox, and BFCL. Section 4.1 describes the task splits, the comparison methods, and the number of independent runs behind each reported mean and standard deviation; Appendix I lists the training settings, and Appendix B works through the credit computation on a complete example.

AI use statement

We used generative AI tools to edit the text of this manuscript and to assist with writing experiment and analysis code. We did not use them for research ideation, experimental design, or the analysis and interpretation of results. The authors reviewed all edited text and verified and tested all AI-assisted code before use. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs. *arXiv preprint arXiv:2402.14740*, 2024. URL https://arxiv.org/abs/2402.14740.

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. Reinforcement Learning for Long-Horizon Interactive LLM Agents. *arXiv preprint arXiv:2502.01600*, 2025. URL https://arxiv.org/abs/2502.01600.

DeepSeek-AI et al. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-Group Policy Optimization for LLM Agent Training. *arXiv preprint arXiv:2505.10978*, 2025. URL https://arxiv.org/abs/2505.10978.

Shuo He, Lang Feng, Qi Wei, Xin Cheng, Lei Feng, and Bo An. Hierarchy-of-Groups Policy Optimization for Long-Horizon Agentic Tasks. *arXiv preprint arXiv:2602.22817*, 2026. URL https://arxiv.org/abs/2602.22817.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. *arXiv preprint arXiv:2503.09516*, 2025. URL https://arxiv.org/abs/2503.09516.

Wonjoong Kim, Yeonjun In, Sangwu Park, Dongha Lee, and Chanyoung Park. PAIR: Prefix-Aware Internal Reward Model for Multi-Turn Agent Optimization. *arXiv preprint arXiv:2605.17877*, 2026. URL https://arxiv.org/abs/2605.17877.

Jaewoo Lee, Archiki Prasad, Justin Chih-Yao Chen, Zaid Khan, Elias Stengel-Eskin, and Mohit Bansal. PRInTS: Reward Modeling for Long-Horizon Information Seeking. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026. URL https://arxiv.org/abs/2511.19314.

Jiazheng Li, Yawei Wang, David Yan, Yijun Tian, Zhichao Xu, Huan Song, Panpan Xu, and Lin Lee Cheong. SALT: Step-level Advantage Assignment for Long-horizon Agents via Trajectory Graph. *arXiv preprint arXiv:2510.20022*, 2025a. URL https://arxiv.org/abs/2510.20022.

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. *arXiv preprint arXiv:2512.17008*, 2025b. URL https://arxiv.org/abs/2512.17008.

Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Junge Zhang, and Jianbin Jiao. Agentic Reinforcement Learning with Implicit Step Rewards. *arXiv preprint arXiv:2509.19199*, 2025a. URL https://arxiv.org/abs/2509.19199.

Zichen Liu, Anya Sims, Keyu Duan, Changyu Chen, Simon Yu, Xiangxin Zhou, Haotian Xu, Shaopan Xiong, Bo Liu, Chenmien Tan, Chuen Yang Beh, Weixun Wang, Hao Zhu, Weiyan Shi, Diyi Yang, Michael Shieh, Yee Whye Teh, Wee Sun Lee, and Min Lin. GEM: A Gym for Agentic LLMs. *arXiv preprint arXiv:2510.01051*, 2025b. URL https://arxiv.org/abs/2510.01051.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, 1999. URL https://ai.stanford.edu/~ang/papers/shaping-icml99.pdf.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

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. *arXiv preprint arXiv:2504.13958*, 2025. URL https://arxiv.org/abs/2504.13958.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. Hindsight Credit Assignment for Long-Horizon LLM Agents. *arXiv preprint arXiv:2603.08754*, 2026. URL https://arxiv.org/abs/2603.08754.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents. *arXiv preprint arXiv:2407.18901*, 2024. URL https://arxiv.org/abs/2407.18901.

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, and Zhenzhe Ying. Information Gain-based Policy Optimization: A Simple and Effective Approach for Multi-Turn Search Agents. *arXiv preprint arXiv:2510.14967*, 2025a. URL https://arxiv.org/abs/2510.14967.

Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. SPA-RL: Reinforcing LLM Agents via Stepwise Progress Attribution. *arXiv preprint arXiv:2505.20732*, 2025b. URL https://arxiv.org/abs/2505.20732.

Ruiyi Wang and Prithviraj Ammanabrolu. A Practitioner's Guide to Multi-turn Agentic Reinforcement Learning. *arXiv preprint arXiv:2510.01132*, 2025. URL https://arxiv.org/abs/2510.01132.

Yu Wang. The Dark Room in the Reward Channel: Dense Prediction Rewards Collapse GRPO-Trained LLM Agents – and The Channel, Not the Content, Decides What Works. *arXiv preprint arXiv:2607.21273*, 2026. URL https://arxiv.org/abs/2607.21273.

Yunan Wang, Minghui Song, Zihan Zhang, Shaohan Huang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. Group-Graph Policy Optimization for Long-Horizon Agentic Reinforcement Learning. *arXiv preprint arXiv:2606.22995*, 2026. URL https://arxiv.org/abs/2606.22995.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding Self-Evolution in LLM Agents via Multi-Turn Reinforcement Learning. *arXiv preprint arXiv:2504.20073*, 2025c. URL https://arxiv.org/abs/2504.20073.

Quan Wei, Siliang Zeng, Chenliang Li, Zhongruo Wang, William Brown, Oana Frunza, Wei Deng, Anderson Schneider, Yuriy Nevmyvaka, Yang Katie Zhao, Alfredo Garcia, and Mingyi Hong. Reinforcing Multi-Turn Reasoning in LLM Agents via Fine-Grained Reward Structure and Credit Assignment. *arXiv preprint arXiv:2505.11821*, 2025. URL https://arxiv.org/abs/2505.11821.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Yajie Yang, Wenxiang Chen, Zhihao Zhang, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. AgentPRM: Process Reward Models for LLM Agents via Step-Wise Promise and Progress. *arXiv preprint arXiv:2511.08325*, 2025. URL https://arxiv.org/abs/2511.08325.

Kaibing Yang, Guangfeng Cai, Shengtian Yang, Shuo He, Yu Li, Mengyi Liu, Pengwei Chen, Jun Xu, and Lei Feng. Progress-conditioned Group Policy Optimization for Long-Horizon Agentic Tasks. *arXiv preprint arXiv:2607.22724*, 2026. URL https://arxiv.org/abs/2607.22724.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. *arXiv preprint arXiv:2406.12045*, 2024. URL https://arxiv.org/abs/2406.12045.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476.

Huining Yuan, Zelai Xu, Huaijie Wang, Xiangmin Yi, Jiaxuan Gao, Xiao-Ping Zhang, Yu Wang, Chao Yu, and Yi Wu. Verifiable Process Rewards for Agentic Reasoning. *arXiv preprint arXiv:2605.10325*, 2026. URL https://arxiv.org/abs/2605.10325.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Francisco Piedrahita-Velez, Yue Liao, Hongru Wang, Mengyue Yang, Heng Ji, Jun Wang, Shuicheng Yan, Philip Torr, and Lei Bai. The Landscape of Agentic Reinforcement Learning for LLMs: A Survey. *arXiv preprint arXiv:2509.02547*, 2025. URL https://arxiv.org/abs/2509.02547.

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. URL https://arxiv.org/abs/2602.12268.

Shuaiyu Zhou, Fengpeng Yue, Zengjie Hu, Yuanzhe Shen, Chenyang Zhang, feng hong, Cao Liu, and Ke Zeng. ToolVerse: Unlocking Massive Environments and Long-Horizon Tasks for Agentic Reinforcement Learning. *arXiv preprint arXiv:2607.15660*, 2026. URL https://arxiv.org/abs/2607.15660.

A OUTPUT ENTROPY AFTER TOOL FEEDBACK

Computation and comparison. We compute full-vocabulary Shannon entropy of the model's output after tool feedback at temperature 1, in nats, over the first 120 tokens. We classify feedback into tool errors, progress without errors, and unchanged progress without errors; we call the last category ordinary turns. Errors take precedence in classification. First turns, progress regressions, and records with unknown status are excluded from ordinary turns. We retain only untruncated turns with at least 120 output tokens. We average curves within trajectories, within tasks, and across tasks, then give equal weight to the three seeds. Shading shows the sample standard deviation across seeds. For paired differences, we first subtract ordinary-turn entropy from feedback-turn entropy within the same trajectory, then aggregate over the beginning (tokens 1 to 20), middle (21 to 60), and end (61 to 120). This statistic differs from subtracting the two overall mean curves directly.

Overall entropy and feedback differences at 4B. In Figure A1, both training methods reduce ordinary-turn entropy in the middle segment, from 0.57 before training to 0.24 for GRPO and 0.37 for ProCredit. Meanwhile, the middle-segment entropy difference between progress and ordinary turns is -0.093 ± 0.014 for ProCredit and -0.025 ± 0.019 for GRPO. All three ProCredit seeds have lower values than all three GRPO seeds. The beginning-segment difference between error and ordinary turns is $+0.235 \pm 0.034$ before training, $+0.245 \pm 0.027$ for ProCredit, and $+0.129 \pm 0.068$ for GRPO. ProCredit preserves the entropy increase after errors and widens the middle-segment difference between progress and ordinary turns.

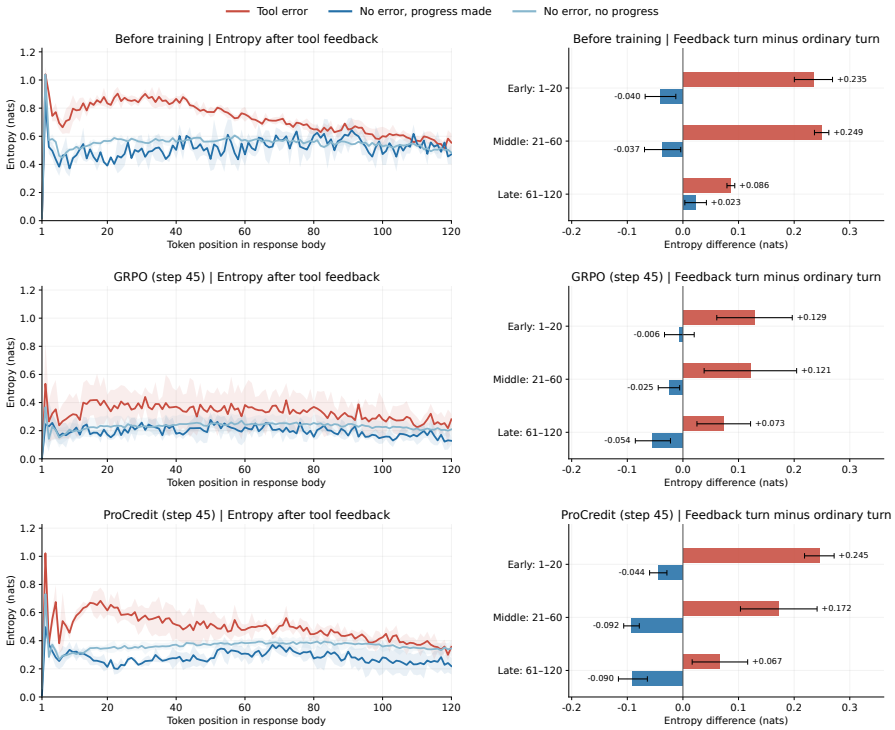

Figure A1: Output entropy after tool feedback for 4B before training, after GRPO, and after ProCredit.

Feedback differences at 9B depend on output position. In Figure A2, ProCredit's error curve starts above the ordinary-turn curve and gradually approaches it; its progress curve is lower in the middle segment. Within-trajectory paired results show the same change. The middle-segment progress-minus-ordinary difference is -0.062 ± 0.011 for ProCredit and -0.037 ± 0.041 for GRPO. The end-segment error-minus-ordinary differences are -0.008 ± 0.021 and $+0.065 \pm 0.040$, respectively. At 9B, the entropy increase after errors is therefore concentrated at earlier positions and approaches ordinary-turn levels toward the end.

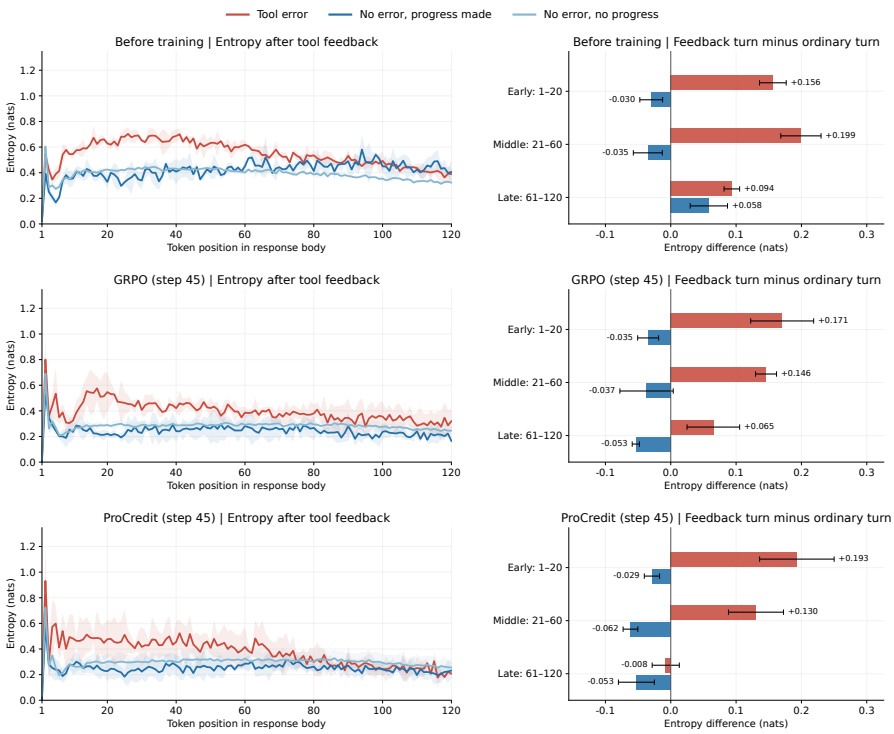

Figure A2: Output entropy after tool feedback for 9B before training, after GRPO, and after ProCredit.

35B also distinguishes feedback types. In Figure A3, ProCredit's progress curve lies below the ordinary-turn curve in the middle segment; entropy after errors rises early and then falls. The middle-segment progress-minus-ordinary difference changes from -0.029 ± 0.022 before training to -0.080 ± 0.019 under ProCredit. The end-segment error-minus-ordinary difference changes from $+0.080 \pm 0.021$ to $+0.017 \pm 0.010$. Output distributions continue to differ across feedback types, with the magnitude and duration of these differences depending on model size and training method.

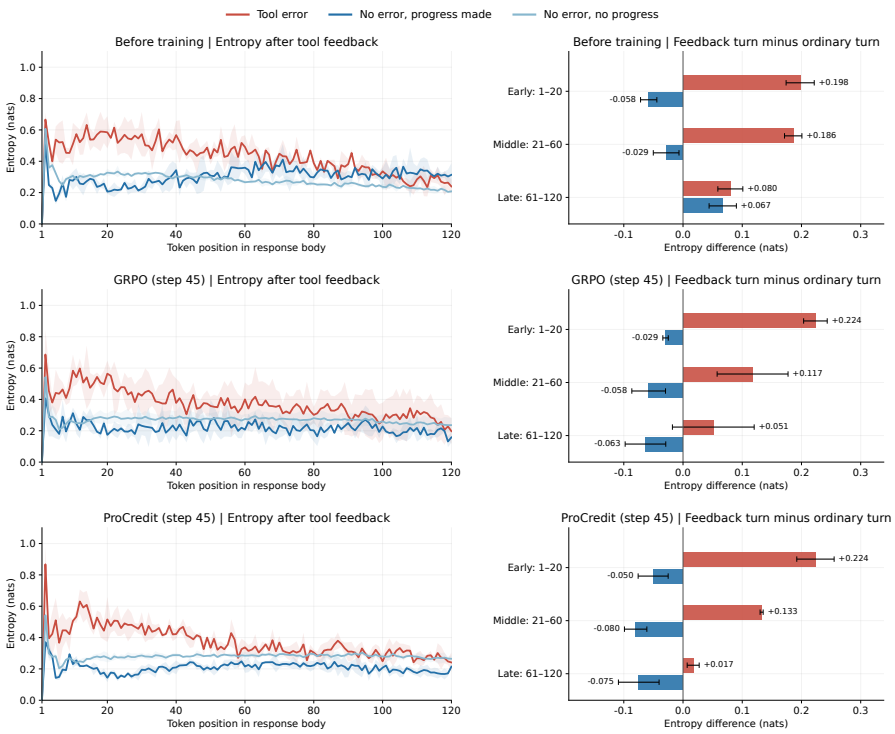

Figure A3: Output entropy after tool feedback for 35B before training, after GRPO, and after ProCredit.

B A WORKED CREDIT EXAMPLE FOR THE ALARM TASK

We apply the computation in Section 3.3 to the task in Figure 1. The three checks require setting the wake-up alarm to 06:00, disabling Gym, and disabling Work; $K_g = 3$, $c = 0.5$, and $\gamma = 1$. The illustrative group contains three trajectories: A reads the alarm list and declares completion; B changes the time and disables Gym but misses Work; C completes all three requirements. Table B1 lists each turn's progress Φ_t, progress reward r_t, return G_t, and two levels of advantage.

The three trajectory scores S are 0, 0.333, and 1.5, and their group-standardized trajectory-level advantages are -0.776, -0.353, and $+1.129$. Completion distinguishes A and B despite both failing. The mean return over all turns for the task is 0.667; each turn-level advantage is its return minus this mean.

Table B1: Per-turn progress, reward, return, and advantages for three illustrative trajectories on the task in Figure 1.

Trajectory	Turn	Action	Φ_t	r_t	G_t	Turn-level advantage	Total advantage
A (failed)	1	Read alarm list	0	0	0	-0.667	-1.443
	2	Declare completion	0	0	0	-0.667	-1.443
B (failed)	1	Read alarm list	0	0	0.333	-0.333	-0.686
	2	Set wake-up alarm to 06:00	$1/3$	$+0.167$	0.333	-0.333	-0.686
	3	Disable Gym	$2/3$	$+0.167$	0.167	-0.500	-0.853
	4	Declare completion	$2/3$	0	0	-0.667	-1.019
C (successful)	1	Read alarm list	0	0	1.5	$+0.833$	$+1.962$
	2	Set wake-up alarm to 06:00	$1/3$	$+0.167$	1.5	$+0.833$	$+1.962$
	3	Disable Gym	$2/3$	$+0.167$	1.333	$+0.667$	$+1.795$
	4	Disable Work	1	$+0.167$	1.167	$+0.500$	$+1.629$
	5	Declare completion	1	0	1.0	$+0.333$	$+1.462$

The table illustrates three properties from Section 3.3. Reading the list and the immediately following progress turn receive equal credit because they have the same remaining progress to earn. After each progress turn, the next turn's credit decreases by exactly the progress reward just earned, 0.167. In B, declaring completion after making progress has the lowest turn-level credit, tied with both turns in A. Across trajectories, every turn in C has higher total credit than any turn in B, and every turn in B has credit at least as high as those in A.

C How discounting penalizes long trajectories

Section 4.3.2 shows that discounting leads models to use fewer turns, with accuracy losses concentrated on medium and hard tasks. We first state where discounting reduces credit, then use illustrative and actual training trajectories to quantify the reduction, and finally examine the trained models by task difficulty.

Proposition 1 (Length tax)**.** *Using the notation of Section 3.3, consider turn p of a trajectory. For any $t \leq p$, let the discounted value at turn t of rewards received after p be*

$$D_t^{(p)} = \sum_{u=p+1}^{T} \gamma^{u-t} r_u + \gamma^{T-t} R. \tag{12}$$

Inserting a state-preserving turn after turn p delays every subsequent reward, including the outcome reward, by one turn. The return $G_t^{(\gamma)}$ then decreases by $(1-\gamma)D_t^{(p)}$ for every $t \leq p$.

Proof. Before insertion, $G_t^{(\gamma)} = \sum_{u=t}^{p} \gamma^{u-t} r_u + D_t^{(p)}$. After insertion, each term after p receives one additional factor of γ, giving $\sum_{u=t}^{p} \gamma^{u-t} r_u + \gamma D_t^{(p)}$. Subtracting gives the result. □

Proposition 1 shows that each additional turn reduces earlier turns' credit by $(1-\gamma)$ times the discounted value of rewards yet to arrive: 5% per turn at $\gamma = 0.95$, and no reduction at $\gamma = 1$. The reduction depends only on the number of turns, regardless of what the inserted turn does. This is the length tax described in Section 4.3.2.

Discounting also gives the two levels of credit different objectives. The trajectory-level term uses the undiscounted score $S_i = G_{i,1}^{(1)}$, whereas the turn-level term uses discounted returns when $\gamma < 1$. Both correspond to the same objective $\mathbb{E}[G_1^{(1)}]$ only when $\gamma = 1$.

Credit differences between equally successful trajectories. In an all-success group for the same task, every trajectory has $R = 1$ and $\Phi_T = 1$. Their scores are equal, so their trajectory-level advantages are zero and the turn-level term supplies the only within-group signal. Figure C1 plots the return G_t at each turn for two illustrative successful trajectories. One takes 8 turns and passes a check at turns 2, 5, and 8; the other takes 16 turns and passes a check at turns 4, 10, and 16, with each progress event occurring at twice the turn index. At $\gamma = 1$, a turn's return depends only on the remaining progress. Both trajectories have the same three-level staircase, with each level lasting twice as long in the longer trajectory. At $\gamma = 0.95$, returns also decay with distance to the next reward. The longer trajectory receives less return at each progress stage, with the gap given by Proposition 1. Statistics from actual training trajectories agree. We collect 119, 189, and 148 all-success groups from training the $\gamma = 0.95$ variant at 4B, 9B, and 35B, respectively. Using the return formula in Section 3.3, we average turn-level credit within each trajectory and compare trajectories within each group. At $\gamma = 0.95$, the shortest trajectory has the highest average credit in 94.1%, 94.2%, and 91.9% of groups, respectively. Recomputing credit on the same trajectories with $\gamma = 1$ reduces these proportions to 5.0%, 6.3%, and 15.5%.

Accuracy losses concentrate on medium and hard tasks. We divide test-normal tasks into easy, medium, and hard groups using GRPO's per-task success rates, independently of the methods being compared. Table C1 compares mean turn counts and success rates for $\gamma = 1$, $\gamma = 0.95$, and a variant that discounts only all-success groups (V4 in Appendix D). Discounting shortens trajectories in all three difficulty groups. Easy tasks already require few turns, leaving little room for shortening and almost no change in success. Medium and hard tasks need more turns; removing these turns lowers

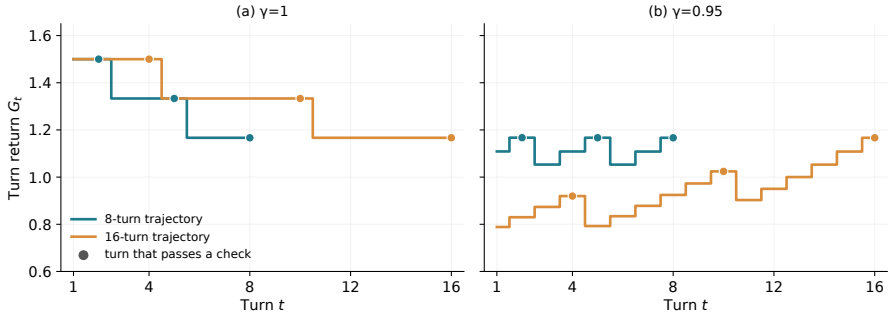

Figure C1: Per-turn return G_t for two illustrative successful trajectories with three checks and $c = 0.5$.

success rates. Discounting only all-success groups produces intermediate results at 9B, while at 4B both trajectory length and success rate remain nearly unchanged.

Table C1: Mean turn counts and success rates (%) by task difficulty on Test-Normal.

		Easy		Medium		Hard	
Model	**Configuration**	**Turns**	**Success**	**Turns**	**Success**	**Turns**	**Success**
4B	ProCredit $\gamma = 1$	18.4	93.3	23.9	73.8	33.7	20.2
4B	ProCredit $\gamma = 0.95$	15.5	92.6	20.2	69.0	29.3	17.9
4B	V4	17.9	93.9	22.9	73.9	33.0	19.8
9B	ProCredit $\gamma = 1$	16.4	95.2	21.5	78.7	31.2	19.4
9B	ProCredit $\gamma = 0.95$	14.0	88.6	18.0	64.8	24.1	9.8
9B	V4	15.7	93.2	20.1	71.1	28.1	15.4

D Complete comparisons of group-dependent variants

Section 4.3.2 uses the same γ for all groups. Here we apply discounting, or disable the turn-level term, only for selected types of sampled groups. The four configurations modify only the turn-level term and retain the trajectory-level term.

Table D1: Group-dependent discounting or removal of turn-level credit on Test-Normal (%).

	Turn-level configurations	
Variant	**All-success groups**	**Mixed and all-fail groups**
V1	$\gamma = 0.95$	$\gamma = 0.95$
ProCredit	$\gamma = 1$	$\gamma = 1$
V3	Disabled	$\gamma = 0.95$
V4	$\gamma = 0.95$	$\gamma = 1$

	4B		**9B**		**35B**	
Variant	**TGC ↑**	**SGC ↑**	**TGC ↑**	**SGC ↑**	**TGC ↑**	**SGC ↑**
V1	$68.48_{\pm 2.53}$	$45.24_{\pm 5.61}$	$68.20_{\pm 5.66}$	$45.98_{\pm 8.65}$	$65.13_{\pm 7.37}$	$44.64_{\pm 8.12}$
ProCredit	$\underline{71.53}_{\pm 0.48}$	$\underline{48.88}_{\pm 0.45}$	$\mathbf{78.15}_{\pm 2.61}$	$\mathbf{59.82}_{\pm 3.95}$	$\mathbf{78.65}_{\pm 0.85}$	$\mathbf{59.60}_{\pm 1.35}$
V3	$69.84_{\pm 1.62}$	$47.10_{\pm 2.35}$	$68.90_{\pm 5.01}$	$44.57_{\pm 7.91}$	$71.50_{\pm 6.37}$	$51.34_{\pm 7.77}$
V4	$\mathbf{71.70}_{\pm 3.76}$	$\mathbf{49.63}_{\pm 7.03}$	$\underline{73.64}_{\pm 1.98}$	$\underline{52.68}_{\pm 3.33}$	$\underline{74.31}_{\pm 2.99}$	$\underline{57.44}_{\pm 3.53}$

V1 and V4 differ only in whether mixed and all-fail groups are discounted. Removing discounting for these groups improves TGC and SGC at all three model sizes. V4 and ProCredit differ only in

whether all-success groups are discounted. ProCredit scores higher at 9B and 35B; they perform similarly at 4B, where V4 varies more across training runs. V3 disables turn-level credit in all-success groups but retains discounting elsewhere, and scores below ProCredit at all three model sizes. The main method retains the turn-level term and uses $\gamma = 1$ for every group.

E LEARNING DYNAMICS

To examine how progress affects learning, Figure E1 compares training and development success rates for GRPO, GRPO-Φ, and ProCredit over the first 50 training steps; all three methods sample the same number of trajectories per step, so steps measure training compute. Training success improves for all three methods and remains similar across them. Differences emerge on the development set, where ProCredit is generally higher during the middle and later stages at 4B and 9B.

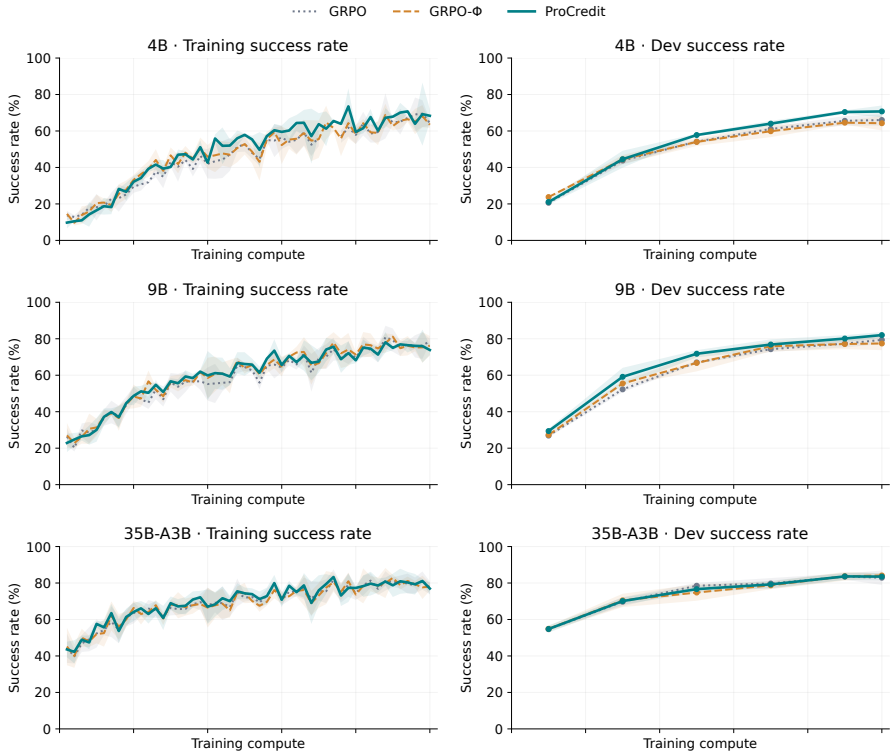

Figure E1: Training and development success rates over the first 50 training steps for GRPO, GRPO-Φ, and ProCredit at three model sizes.

Compared with GRPO-Φ, ProCredit's gains appear on held-out tasks. By the later stages of training, development success rises from 64.6% to 70.4% at 4B and from 77.1% to 80.1% at 9B. GRPO-Φ matches or exceeds ProCredit's training success at some steps, but these gains do not carry over to the development set.

Comparison by training time. Figure E2 plots the same 4B development checkpoints against cumulative training time. Each run uses four GPUs, and cumulative time excludes initialization, downtime, and development evaluation. Under a six-hour budget, using each run's latest evaluated checkpoint available at that time, ProCredit achieves a mean success rate of 70.39%, compared with 61.99% for GRPO and 64.55% for GRPO-Φ.

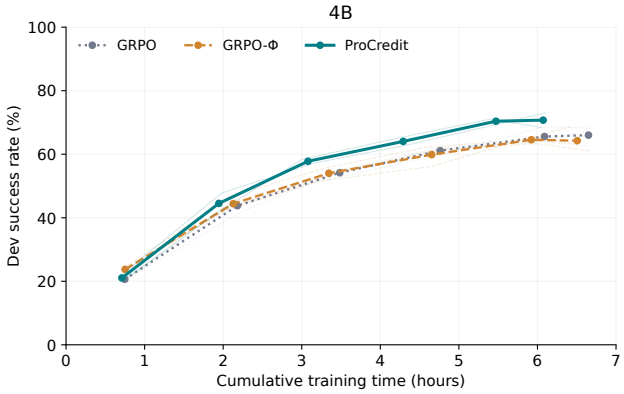

Figure E2: 4B development success against cumulative training time. Thick curves show seed means at the median time of each checkpoint; faint lines show individual seeds.

F Retention of BFCL capabilities

Evaluation scope. We use BFCL to assess changes in existing function-calling capabilities after AppWorld training, covering non-live single-turn AST evaluation and four multi-turn subcategories. Categories outside this scope do not enter the aggregate scores. BFCL is not used for training or for selecting checkpoints, seeds, or methods. We evaluate the same checkpoints as in the main AppWorld table.

Table F1: BFCL scores (%) before and after AppWorld training.

	4B		9B	
Method	**Single-turn AST ↑**	**Multi-turn ↑**	**Single-turn AST ↑**	**Multi-turn ↑**
Base model	85.73	52.38	87.19	52.25
GRPO	$86.52_{\pm 0.30}$	$53.54_{\pm 1.23}$	$86.30_{\pm 0.22}$	$52.21_{\pm 1.60}$
GRPO-Φ	$85.28_{\pm 0.06}$	$52.50_{\pm 1.25}$	$86.47_{\pm 0.30}$	$53.21_{\pm 0.69}$
ProCredit	$85.94_{\pm 0.10}$	$53.04_{\pm 1.01}$	$86.73_{\pm 0.17}$	$53.83_{\pm 1.94}$

General function-calling capabilities remain stable. All three trained methods score close to the base models: single-turn AST scores differ by less than 1 percentage point and multi-turn scores by less than 2 points, comparable to the variation across three training runs. AppWorld training does not weaken the function-calling capabilities measured by BFCL. The ordering of methods varies by model size, and we do not rank them here.

G Two properties of the trajectory score

We use the notation from Sections 3.2 and 3.3.

Proposition 2. *For any $c > 0$, every successful trajectory for a task has a strictly higher score than every failed trajectory for that task.*

Proposition 3. *If a policy can complete the task with probability 1, the set of policies maximizing expected trajectory score is identical to the set maximizing success probability.*

The proofs use three facts: all trajectories for a task share the same Φ_0; $\Phi_T \in [0, 1]$; and $R = 1$ implies $\Phi_T = 1$, because the outcome reward requires all acceptance checks to pass and $\mathcal{C}_g$ is a subset of those checks.

Proof of Proposition 2. A successful trajectory has score $S = 1 + c\,(1 - \Phi_0)$. A failed trajectory has $R = 0$ and $\Phi_T \leq 1$, so its score satisfies $S \leq c\,(1 - \Phi_0)$. The difference is at least 1, independently of c. □

Proof of Proposition 3. For any policy π,

$$\mathbb{E}_\pi[S] = \mathbb{E}_\pi[R] + c\,(\mathbb{E}_\pi[\Phi_T] - \Phi_0) \le 1 + c\,(1 - \Phi_0) =: S_{\max}. \tag{13}$$

Equality holds if and only if $\mathbb{E}_\pi[R] = 1$. Necessity follows because $\mathbb{E}_\pi[R]$ and $\mathbb{E}_\pi[\Phi_T]$ are each bounded above by 1, and equality requires both to reach that bound. Sufficiency follows because $R = 1$ implies $\Phi_T = 1$. When the task is solvable, $S_{\max}$ is attainable. Thus, the policies maximizing $\mathbb{E}[S]$ are exactly those that succeed with probability 1, which are also exactly the policies maximizing $\mathbb{E}[R]$. □

Two points qualify these results. First, Proposition 3 does not hold when the task is unsolvable: expected score is success probability plus c times expected completion, so optimizing it trades success probability for partial completion at a rate set by c. This trade-off is the purpose of scoring partial completion. Second, the policy-invariance theorem of Ng et al. (1999) requires zero potential at terminal states, which corresponds to subtracting $c\,\Phi_T$ at the final turn. The trajectory score then becomes $R - c\,\Phi_0$: optimal policies remain unchanged, but trajectory-level progress information disappears, leaving all-fail groups with zero advantages again. We retain Φ_T to preserve this information. AppWorld has deterministic environments and a reference solution for every task, so the premise of Proposition 3 holds for the training tasks.

H Why turns with different histories can share one reference value

GRPO compares trajectories sampled for the same task, all starting from the same initial state. ProCredit's turn-level term pools all turns for the task and subtracts their mean, even though different trajectories have different histories by turn t. This section explains why this comparison remains valid and how it relates to state-based grouping.

Both terms subtract a reference computed from the sampled group. For the trajectory-level term, including a trajectory's own score in the group mean scales the expected policy gradient by $(N - 1)/N$, as in GRPO. The turn-level term subtracts the mean return over all turns of all trajectories for the same task. A turn's reference need not come from turns with the same history: the turns of the other trajectories are sampled independently of the action taken at this turn, so their mean b_{-i} is a valid baseline for every turn of trajectory i. Equation (9) uses the pooled mean, which differs from b_{-i} by $w_i(\bar{G}_i - b_{-i})$, where $w_i = T_i/|\mathcal{U}_g|$ is the share of turns contributed by trajectory i, about $1/N$, and $\bar{G}_i$ is its mean turn return. This term varies with the trajectory's outcome, progress, and length, and is a bounded correction of order $1/N$. Equation (11) is unaffected, since it requires only that every turn for a task subtract the same value. The resulting turn-level advantage measures how much the return from that turn exceeds the mean over all turns for the task.

State-based grouping, as in GiGPO with $\gamma = 1$ in Table 3, changes the reference value. It groups turns at the same progress state for the same task and subtracts the group's mean, with the aim of evaluating arrival at a state separately from the current action. The two turn-level advantages differ by one term:

$$A^{\text{anchor}}_{i,t} = A^{\text{turn}}_{i,t} - \big(\mu_{\mathcal{A}(i,t)} - \mu_{\mathcal{U}_g}\big), \tag{14}$$

where $\mu_{\mathcal{A}}$ is the mean for the turn's anchor group and $\mu_{\mathcal{U}}$ is the mean over all turns for the task. The relative ordering of turns within the same state is identical. On AppWorld, anchor groups have a median size of 6.6 turns and fewer than 1% are singletons, so group size is not the issue. Yet this method does not outperform ProCredit at any of the three model sizes. There are two reasons. Once turns are grouped by state, credit for reaching a good state is no longer assigned to the current turn; earlier turns must recover that credit through comparisons within their own groups, which requires different continuations within those groups. Also, the anchors record only passed checks, omitting database contents and recent actions. Turns in the same group may therefore face different decisions, and a reference value estimated from just a few turns varies more than the mean over all turns for the task.

I TRAINING SETTINGS

Table I1 lists the training settings of ProCredit behind Tables 1 and 2. Length limits and the number of GPUs grow with model size; the ToolSandbox settings differ from AppWorld only in tasks per step, training steps, and the turn limit. Baselines use the hyperparameters of their original papers.

Table I1: Training settings of ProCredit.

Setting	**AppWorld**	**ToolSandbox**
Rollouts per task	8	8
Tasks per step	25	16
Training steps	50	35
Learning rate	1e-6	1e-6
PPO clip range	0.2	0.2
KL term / entropy term	none	none
Loss aggregation	token-mean	token-mean
Maximum turns per trajectory	50	30
Maximum tokens per turn	2048	2048
Maximum prompt length	4096	8192
Maximum response length	28672–32768	28672
Context length	32768–36864	32768
Sampling temperature	1.0	1.0
Progress coefficient c / discount γ	0.5 / 1	0.5 / 1
GPUs per run	4–8	2