

A Simple "Try Again" Can Elicit Multi-Turn LLM Reasoning

Licheng Liu^{1*}, Zihan Wang^{2*}, Linjie Li³, Chenwei Xu², Yiping Lu², Han Liu², Avirup Sil⁴,
Manling Li²

¹Imperial College London ²Northwestern University ³University of Washington ⁴IBM Research AI
[unary-feedback.github.io](https://github.com/unary-feedback)

Multi-turn problem solving is critical yet challenging for Large Reasoning Models (LRMs) to reflect on their reasoning and revise from feedback. Existing Reinforcement Learning (RL) methods train large reasoning models on a single-turn paradigm with verifiable rewards. However, we observe that models trained with existing RL paradigms often **lose their ability to solve problems across multiple turns** and struggle to revise answers based on contextual feedback, leading to repetitive responses. We ask: can LRMs learn to reflect their answers in a multi-turn context? In this work, we find that training models with multi-turn RL using only unary feedback (for example, "Let's try again") after wrong answers can improve both single-turn performance and multi-turn reasoning. We introduce **Unary Feedback as Observation (UFO)** for reinforcement learning, which uses minimal yet common unary user feedback during iterative problem solving. It can be easily applied to existing single-turn RL training setups. Experimental results show that RL training with UFO keeps single-turn performance and improves multi-turn reasoning accuracy by up to 14%, enabling language models to better react to feedback in multi-turn problem solving. To further minimize the number of turns needed for a correct answer while encouraging diverse reasoning when mistakes occur, we design reward structures that guide models to produce careful and deliberate answers in each turn. Our code and models are open source <https://github.com/lichengliu03/unary-feedback>.

1. Introduction

Large language and reasoning models (LLMs/LRMs) (DeepSeek-AI, 2025; OpenAI, 2024; Yang et al., 2024; Team, 2025) can solve complex tasks such as math and code generation, especially with the current progress of reinforcement Learning (RL) (DeepSeek-AI, 2025; Schulman et al., 2017; Zhou et al., 2024; Wang et al., 2025) that enhances LLM reasoning capabilities with verifiable rewards. On the other hand, many real-world applications like chatbots, programming assistants, and educational tools (Xie et al., 2024; Pan et al., 2024; Yao et al., 2023a; Shridhar et al., 2021; Wang et al., 2024a) require models to engage in multi-turn problem solving and adapting their reasoning with feedback. However, it remains underexplored how models trained with single-turn RL generalizes to multi-turn interactive problem-solving settings.

In this work, we first observe that single-turn RL may hinder a model's ability to engage in interactive, multi-turn reasoning. Specifically, such models often fail to incorporate in-context

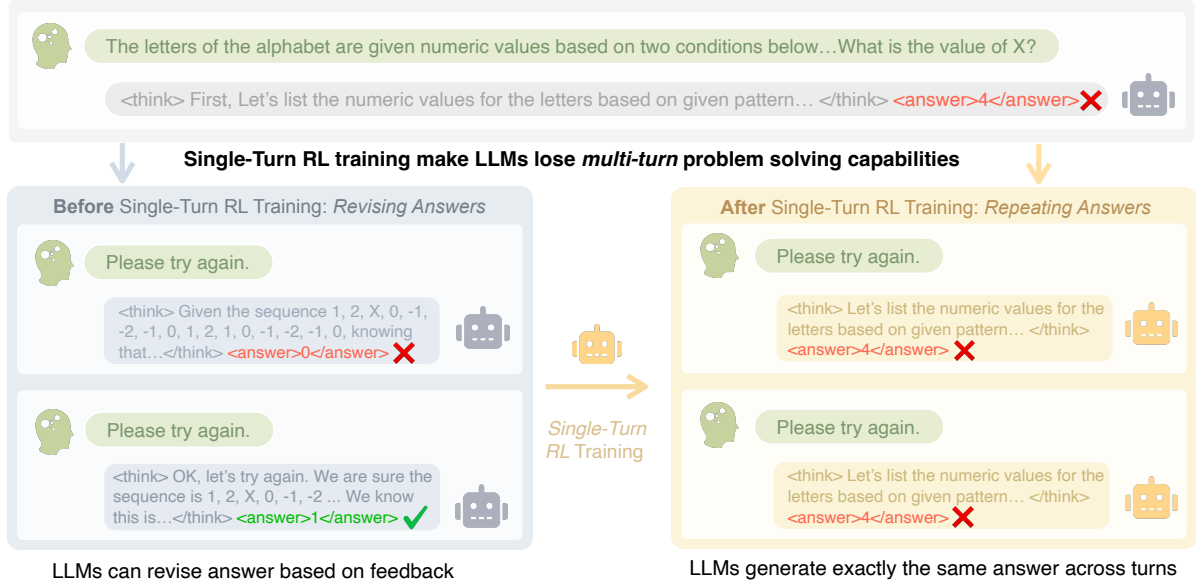


Figure 1 | Single-turn RL causes LLMs to repeat the same answer across turns instead of revising based on feedback.

feedback and instead **persist with their initial answers across multiple turns** (Figure 1). In Figure 2, we define *effective answer* as a new answer that has never been given by the LLM in multi-turn interaction. It turns out that in 70% of failed cases, single-turn-trained models generate exactly the same answer across five interaction turns (i.e. only 1 effective answer). Though these models excel at single turns, enabling them to effectively leverage in-context feedback and improve over multiple turns remains a challenge. One particular reason is that most existing reasoning datasets are inherently single-turn and do not incorporate multi-turn feedback. This challenge motivates our research question: *How can we train language models that not only generate correct solutions but also improve iteratively from sparse, minimal feedback?*

Real-world multi-turn user feedback is very expensive and hard to obtain. Constrained by this data bottleneck, existing multi-turn framework has been training on automatic feedback such as code interpreter messages (Xie et al., 2024; Pan et al., 2024; Wang et al., 2024a) and embodied simulator signals (Shridhar et al., 2021; Zhuang et al., 2025). However, those inherently single-turn static dataset (e.g. QA, Math) are originally built for single-turn RL training. Moreover, code interpreter and embodied environment still costs large amounts of resource to build (Cao et al., 2025). In light of these limitations, in this work we explore a surprisingly simple yet effective framework that can leverage static dataset for multi-turn RL training, by simply adding verbal unary feedback and encouraging the model to *try again* when the model is wrong. We call this **Unary Feedback as Observation (UFO)**, framing interactive problem-solving as Markov Decision Processes (MDP) where the model (agent) only receives unary feedback (e.g., think again) as its environmental observation.

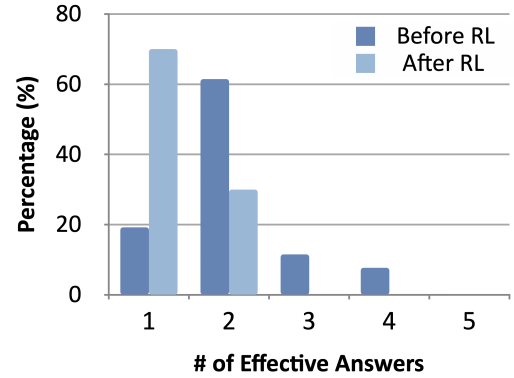


Figure 2 | After single-turn RL training, the model gives exactly the same answer across five interaction turns in 70% of the cases.

Through experiments, we show that applying UFO in multi-turn RL settings can effectively

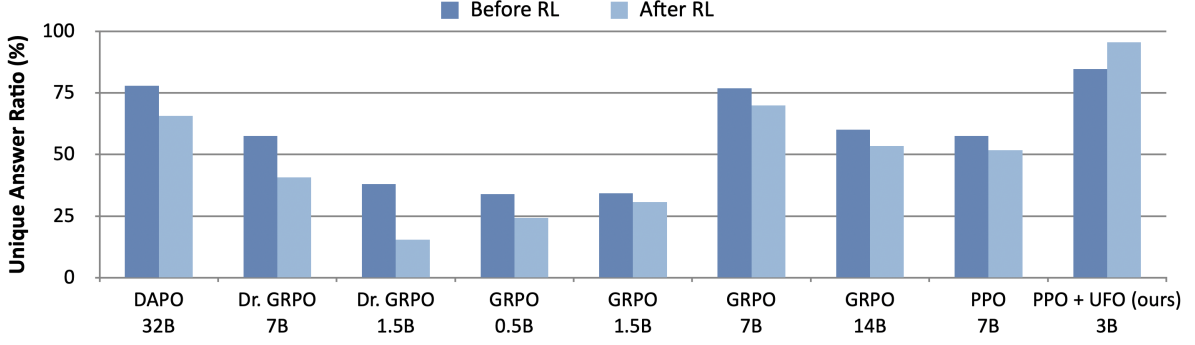


Figure 3 | Comparison of effective (unique) answer ratio (%) before and after RL training. Across single-turn RL methods, the unique answer ratio consistently drops after training across multiple model scales.

stimulate interactive reasoning and allow the model to effectively revise its reasoning across turns. Models trained with UFO inherently learn to try to adopt different approaches when the current answer is wrong, achieving a 14% improvement in success rates under multi-turn evaluation compared to previous single-turn RL approaches. Furthermore, we show that such capability to self-reflect and revise is generalizable to various out-of-domain tasks. To align with real-world multi-turn reasoning objectives, we point out two principles, **minimality** and **diversity**, where the models should minimize interaction turns needed to arrive at a correct answer and explore diverse strategies upon failure. We design turn-wise reward decay and an answer repetition penalty to encourage systematically planning, improve reasoning efficiency.

To summarize, our contributions are as follows:

- We identify that while current single-turn RL training improves reasoning, they can lead to repetitive and degraded outputs in multi-turn, interactive reasoning scenarios.
- We explore a simple yet effective framework, **Unary Feedback as Observation (UFO)**, to enable multi-turn RL training on existing static single-turn reasoning datasets.
- We show that turn-wise reward decay and answer repetition penalty could effectively improve multi-turn reasoning minimality and diversity.

2. Reinforcement Learning for LLM Reasoning

2.1. Background

Single-Turn Reinforcement Learning. Reinforcement Learning (RL) is a general framework to steer the behavior of Large Language Models (LLMs) by maximizing an expected reward:

$$\mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\theta}(\cdot|x)} [R(x, y)],$$

where \mathcal{D} is a prompt distribution, π_{θ} , the policy, and $R(x, y)$, the reward for response y . Algorithms such as PPO (Schulman et al., 2017; Ouyang et al., 2022) and GRPO (DeepSeek-AI, 2025; Shao et al., 2024) apply this objective to static datasets, yielding strong single-turn gains in math and code generation.

Multi-Turn Extensions. Reinforcement Learning (RL) applied to static, single-turn datasets has proven highly effective. Algorithms such as PPO and GRPO have significantly enhanced the reasoning capabilities of LLMs, achieving outstanding results in domains like mathematics and code generation. These methods excel at optimizing a policy to produce a correct answer in a single attempt, leveraging large-scale datasets of problems and their answers.

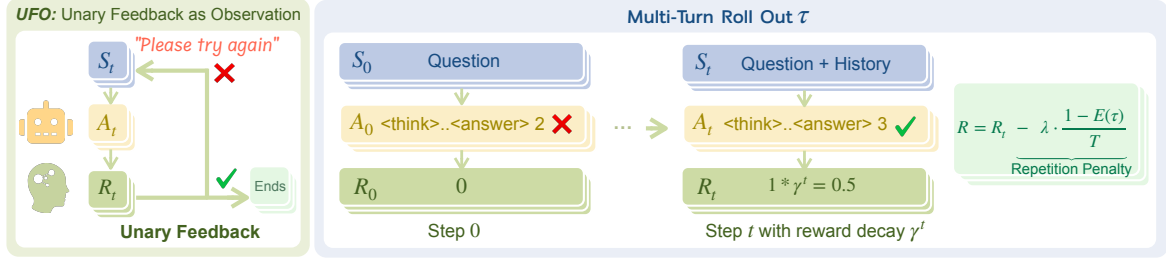


Figure 4 | The UFO framework for multi-turn training. At each step t , the model observes the full interaction history and generates a response. Correct responses receive discounted rewards γ^t , while incorrect ones receive none. A repetition penalty based on the uniqueness of trajectory τ is applied after success or when the turn limit is reached.

However, real-world applications like tutoring, coding assistants, embodied agents demand *multi-turn* interaction, where a model refines answers across steps under sparse feedback. In domains like programming, automated feedback is readily available from compilers or unit tests, enabling iterative correction. But for tasks like mathematical reasoning, obtaining such dense, intermediate feedback is inherently difficult; often, only a final signal of correctness is available. Recent work like RAGEN (Wang et al., 2025) addresses this by framing reasoning as an MDP and optimizing whole trajectories, supporting delayed credit assignment in tasks like symbolic logic and interactive programming. Yet widely used math and code datasets remain single-turn, and collecting turn-by-turn human signals is costly. Most prior work synthesizes feedback (Xie et al., 2024; Pan et al., 2024) or uses tool-augmented environments (Wang et al., 2024a; Jin et al., 2025; Feng et al., 2025), which leads to the key question we study here: *Can models trained only with single-turn RL generalize to multi-turn reasoning?*

2.2. Single-Turn RL Leads to Collapsed Multi-Turn Reasoning

To answer the question posed above, we examine how models trained with single-turn RL perform in multi-turn interaction settings. Specifically, in practical use cases such as tutoring or mathematical assistance, users typically offer minimal feedback (e.g., “try again”) and expect the model to adjust its reasoning accordingly. However, we find that single-turn RL models are **effective solvers but poor revisers**, consistently failing to incorporate feedback.

This phenomenon is illustrated in Figure 1: a pre-trained model refines its answer across turns, while a single-turn RL model fails to revise, repeating its initial output. To quantify this behavior, we use *effective answer* as our metric, which has been defined as a new answer that has not been given in the previous turns. We present the results in Figure 3 for off-the-shelf LLMs and Figure 2 for our trained models with single-turn RL, and show that after single-turn RL training, LLMs tend to give less effective answers across multiple turns.

Specifically, for off-the-shelf LLMs, we select models fine-tuned with various RL algorithms including PPO (Schulman et al., 2017), GRPO (Shao et al., 2024), DAPO (Yu et al., 2025), and Dr. GRPO (Liu et al., 2025). Details of the models used can be found in Appendix D. As shown in Figure 3, all models exhibit a noticeable decline in the unique answer ratio after RL training, and the extent varies by method and model size. For example, under DAPO the 32B model’s effective answer ratio falls from 78.0% to 65.7%, and under Dr. GRPO the 1.5B model drop from 38.0% to just 15.4%. GRPO shows more moderate losses (e.g., the 0.5B model declined from 34.0% to 24.3%), and PPO’s impact is also mild (the 7B model went from 57.6% to 51.7%). We also measure how many distinct answers our single-turn RL model provides for questions it ultimately fails to answer correctly (more details in 4.1). As shown in Figure 2, under nearly

70% cases the model provide identical wrong answers across multiple interaction turns.

This highlights a critical gap: single-turn RL is insufficient for multi-turn reasoning, yet acquiring the necessary step-by-step supervision is often infeasible. In light of this, we ask the following question: **Can we leverage only the simplest form of supervision, such as “try again”, to simulate multi-turn interaction on static datasets and train models to learn adaptive revision behaviors?**

Can minimal feedback alone unlock multi-turn reasoning on static datasets?

3. Training Multi-Turn Reasoning Models with Unary Feedback

3.1. Problem Formulation

We model the process of multi-turn problem solving based on static single-turn datasets as a finite-horizon Markov Decision Process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, T_{\max})$. Here, \mathcal{S} is the state space, \mathcal{A} is the action space consisting of all possible answers, \mathcal{P} is the transition function defined by the agent–environment interaction, R is the reward function, and T_{\max} is the maximum number of interaction steps per episode. At each turn t , the agent observes a state $s_t \in \mathcal{S}$ that encodes the original question q and the history of past attempts and feedbacks:

$$s_t = \text{Concat}(q, \{(a_k, f_k)\}_{k=1}^{t-1}), \quad (1)$$

where a_k denotes the k -th answer, and f_k is a feedback token returned by the environment. The agent then generates an answer $a_t \sim \pi_\theta(\cdot | s_t)$ and receives a scalar reward:

$$r_t = \begin{cases} 1, & \text{if } a_t \text{ is correct,} \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

The episode ends when the agent provides a correct answer or reaches the maximum number of steps T_{\max} . This formulation grounds the multi-turn learning problem in a standard RL framework.

3.2. Unary Feedback as Observation (UFO)

To implement the MDP described above on static datasets, we propose a simple yet general mechanism called **Unary Feedback as Observation (UFO)** (Figure 4). This mechanism defines how the state history is constructed and presented to the agent. The key idea is to restrict f_k in the observation to negative signals only. Specifically, when an answer a_k is incorrect, the feedback is a generic signal such as TryAgain. When the agent produces a correct answer, the episode terminates immediately. Consequently, no explicit positive confirmation (e.g., Correct) is ever added to the state history. The agent thus only receives unary feedback and must learn to revise its answers based solely on a history of failed attempts.

In practice, the prompt is constructed as a natural-language sequence concatenating all previous attempts and their feedback. For example:

```
Question: What is the value of ...?
Attempt 1: [wrong answer]
Feedback: Try Again.
...
Attempt K: [correct answer]
```

This UFO mechanism enables us to transform static single-turn datasets into multi-turn interaction episodes without requiring structural changes, expert annotations, or execution environments. Thus, UFO allows multi-turn reinforcement learning on LLMs with minimal supervision. We describe this training setup as follows.

3.3. Reinforcement Learning with Unary Feedback

Given the MDP formulation and the UFO-based observation design, we optimize the agent using reinforcement learning to learn revision-aware and multi-turn policies. Since the dataset only contains the final-answer accuracy and lacks ground-truth reasoning traces, supervised finetuning is not applicable. Reinforcement learning, in contrast, enables exploration of diverse reasoning strategies under sparse and delayed supervision.

We adopt Proximal Policy Optimization (PPO) to train the policy π_θ , following prior work (Wang et al., 2025; Hu et al., 2025) which shows that a learned critic enables fine-grained value estimates and stabilizes optimization. At each episode, the agent interacts with a problem over multiple rounds. At each turn t , it observes input x_t , generates an answer a_t , and receives a binary reward $r_t \in \{0, 1\}$. The resulting trajectory is defined as:

$$\tau = \{(x_1, a_1, r_1), (x_2, a_2, r_2), \dots, (x_T, a_T, r_T)\}, \quad (3)$$

where $T \leq T_{\max}$ is the number of turns before success or termination. The objective is to maximize the expected return:

$$\mathcal{J}^{\text{RL}}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=1}^T r_t \right]. \quad (4)$$

We apply PPO with a clipped surrogate objective. For each training batch, we estimate the advantage \hat{A}_t using a baseline value function and update the policy as:

$$\mathcal{L}^{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min \left(\frac{\pi_\theta(a_t | x_t)}{\pi_{\theta_{\text{old}}}(a_t | x_t)} \hat{A}_t, \text{clip} \left(\frac{\pi_\theta(a_t | x_t)}{\pi_{\theta_{\text{old}}}(a_t | x_t)} \hat{A}_t, 1 - \epsilon, 1 + \epsilon \right) \right) \right]. \quad (5)$$

Crucially, the UFO design enables the policy to condition on the full history of failure signals, giving rise to context-sensitive behaviors such as error correction, elimination, and hypothesis refinement—capabilities that are difficult to elicit through static supervision alone.

3.4. Reward Design for Adaptive Reasoning

Binary correctness signals offer a minimal form of supervision, but they could induce suboptimal behavior such as blind trial-and-error or repeated guesses. To encourage more efficient and reflective reasoning, we introduce a trajectory-level *reward decay* with *repetition penalty*. Reward decay encourages **minimality** by favoring trajectories that reach correct answers in fewer turns, thereby promoting concise and purposeful reasoning, while the repetition penalty promotes **diversity** by discouraging repetitive generations and encouraging the model to explore alternative strategies upon failure.

Formally, reward decay promotes early success by assigning exponentially diminishing rewards to correct answers produced at later turns:

$$R_t = \begin{cases} \gamma^t, & \text{if } a_t \text{ is correct,} \\ 0, & \text{otherwise,} \end{cases} \quad (6)$$

Table 1 | 5-turn success rate (%) across different tasks and training settings.

	Math		QA		General	
	MMQ-Math	TheoremQA	HotPotQA	ConcurrentQA	MMLU	MMLU-Pro
Base Model w/o RL	52.3	28.3	7.8	3.9	75.2	42.2
RL on MMQ-Math	79.7	32.0	19.5	12.9	66.8	48.3
+ 5-turn UFO	88.5	40.8	26.6	15.2	85.2	60.9
RL on HotPotQA	72.4	31.8	38.3	16.6	71.5	49.3
+ 5-turn UFO	72.7	29.2	44.2	16.8	76.6	48.9

where $\gamma \in (0, 1)$ is a decay factor that favors solving the problem in fewer turns.

We define repetition penalty based on the number of *effective answers*—i.e., responses that have not been submitted previously in the same episode. Let T denote the number of turns in the episode, and $E(\tau)$ be the number of effective answers in the trajectory τ . We define a normalized penalty term:

$$\text{Penalty}(\tau) = \lambda \cdot \left(1 - \frac{E(\tau)}{T}\right), \quad (7)$$

where $\lambda > 0$ is a tunable penalty weight, and $E(\tau)/T$ measures answer diversity. The penalty is maximized when all answers are identical, and vanishes when all are distinct.

Combining above components, the trajectory-level reward for RL training is defined as:

$$R = R_t - \text{Penalty}(\tau). \quad (8)$$

To ensure syntactic validity and improve training stability, we apply a small penalty $\eta < 0$ for each invalid output across turns, i.e., when the model produces malformed or missing answers. To summarize, the reward is determined by the reasoning correctness at the last turn, answer diversity and format correctness of model answers across turns.

4. Experiments

4.1. Setup

Dataset We conduct major experiments on the MATH subset of MetaMathQA (Yu et al., 2024) dataset (MMQ-Math), where data are augmented from the MATH training sets. This environment provides math questions with adequate difficulty, enabling us to observe and analyze its reasoning emergence. We also select six widely-used datasets to evaluate training generalization: MMQ-Math requires multi-step symbolic and numerical reasoning to solve complex math word problems. TheoremQA (Chen et al., 2023a) evaluates formal mathematics understanding through questions about theorem statements and proofs. HotPotQA (Yang et al., 2018) tests multi-hop factual reasoning across Wikipedia passages, while ConcurrentQA (Arora et al., 2022) focuses on temporal and causal reasoning in concurrent event settings. MMLU (Hendrycks et al., 2020) assesses general-knowledge proficiency across 57 academic subjects, and its extension, MMLU-Pro (Wang et al., 2024b) covers more specialized expert domains.

Training Settings We train Qwen-2.5-3B-Instruct with PPO for 200 optimization steps on A100 GPUs. Each batch samples $P=8$ prompts, with $N=16$ rollouts per prompt. During training, we experiment with three distinct configurations for the maximum number of turns per episode, setting T_{\max} to 1, 5, and 10, respectively. For the validation phase, T_{\max} is fixed at 5 turns. In both

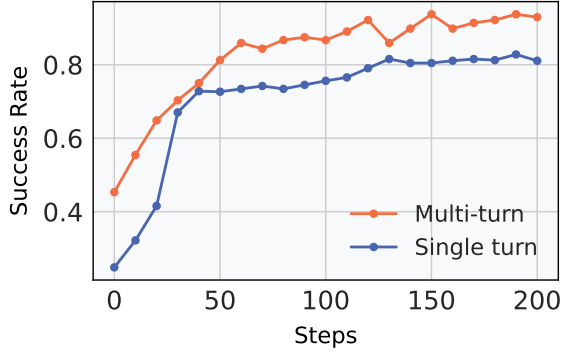


Figure 5 | Multi-turn (5-turn) RL significantly outperforms single-turn baseline, achieving higher success rates (Pass @ 5) with similar inference cost.

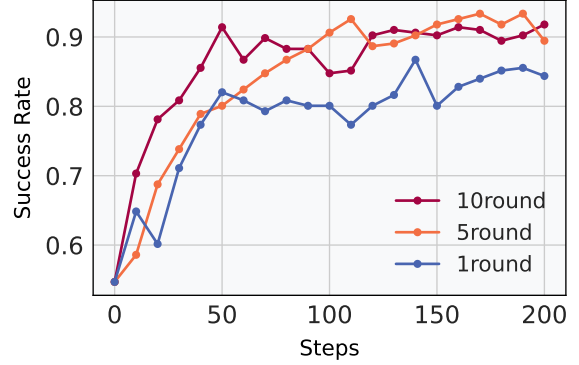


Figure 6 | Performance comparison when evaluating with 5 turns after training with different maximum turns (1, 5, and 10). Training with 5 turns yields the best performance, while increasing to 10 turns offers no significant gain.

training and validation, episodes are limited to a maximum of 10 actions in total. Policy updates use PPO with GAE parameters $(\gamma, \lambda) = (1.0, 1.0)$, Adam with $\beta = (0.9, 0.999)$, entropy coefficient 10^{-3} .

Baseline and Metrics We compare our method **UFO** against a single-turn PPO-trained model using parallel sampling. For each problem, the baseline generates k independent responses in parallel and is evaluated using standard Pass@ k metric. In contrast, our multi-turn model generates responses sequentially with unary feedback after each attempt, and is evaluated using both Succ@ k and AvgTurns. Success is recorded if any of the 5 responses is correct. We also conduct ablation studies with different maximum interaction turns (T_{\max}) to further analyze the effect of multi-turn training.

We report the following complementary metrics to assess both effectiveness and efficiency.

- **Pass@ k (Single-turn baseline).** The proportion of problems for which at least one of the k parallel completions is correct. This metric reflects performance when no feedback is used during generation.
- **Succ@ k (Multi-turn model)** This metric measures the percentage of problems solved within a fixed number of interaction turns. Let τ_j be the number of turns the agent takes to solve problem q_j , or ∞ if it fails. We have:

$$\text{Succ@}k = \frac{1}{N} \sum_{j=1}^N \mathbb{1}[\tau_j \leq k]. \quad (9)$$

We report Succ@1 for single-turn performance, and Succ@5/10 to reflect multi-turn capability.

- **Average Number of Turns (Multi-turn model)** To evaluate interaction efficiency, we report the average number of turns the agent takes to solve each problem: $\text{AvgTurns} = \frac{1}{N} \sum_{j=1}^N T_j$. T_j denotes the number of interactive turns taken for problem q_j . This metric reflects how efficiently the agent reaches a solution, accounting for retries and step-wise refinement across multi-turn episodes.

4.2. Experimental Results and Findings

In this section, we present empirical findings that address three central questions in our study of multi-turn reinforcement learning with unary feedback:

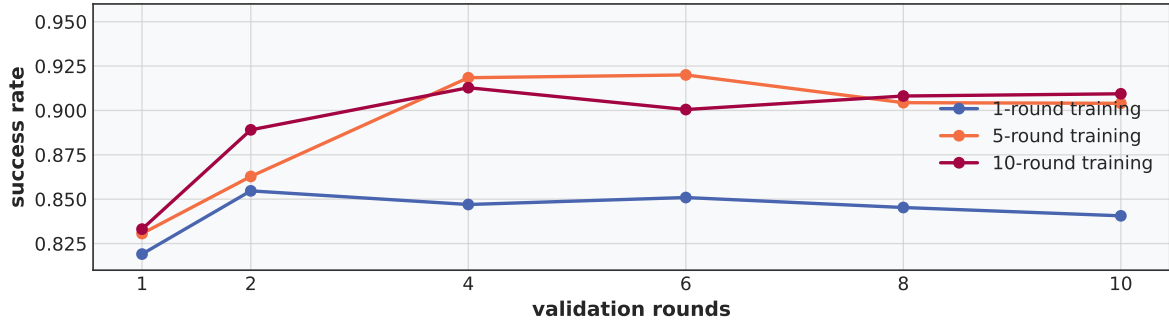


Figure 7 | Validation performance (Succ@k) of models trained with different roll-out turns under varying inference-time turn budgets. Multi-turn training (5 or 10 turns) consistently yields higher success rates across all inference turn budgets, including $k = 1$, indicating better generalization even to single-turn reasoning.

1. Section 4.2.1: Does multi-turn RL unlock stronger reasoning than single-turn training?
2. Section 4.2.2: Can models effectively revise their answers from sparse feedback alone?
3. Section 4.2.3: How do reward shaping strategies impact reasoning efficiency and diversity?

We explore each question in the following subsections, with quantitative analyses and ablation studies. Additional qualitative examples and robustness checks are included in the Appendix.

4.2.1. Multi-turn RL Unlocks Higher Upper Bound of LLM Reasoning

We compare models trained with multi-turn RL against single-turn PPO baselines, using Succ@5 on a held-out validation set evaluated at 21 checkpoints across 200 training steps. During validation, each agent is allowed up to 5 interaction turns per problem ($k = 5$).

As shown in Figure 5, multi-turn training consistently outperforms the single-turn baseline, **achieving up to 14% higher success rate** with comparable inference cost. This highlights the benefit of iterative revision under sparse feedback.

Furthermore, we conduct additional experiments comparing various multi-turn training budgets ($T_{\max} = 1, 5, 10$) while consistently using a 5-turn validation setup. Findings presented in Figure 6 demonstrate that larger training budgets yield enhanced performance relative to the single-turn baseline. Notably, both the $T_{\max} = 10$ and $T_{\max} = 5$ configurations deliver **more than a 6% relative improvement** over single-turn training at their peak, clearly emphasizing the benefits of multi-turn training.

To validate the robustness of these improvements, we expand our analysis by evaluating peak-performing models trained with $T_{\max} \in 1, 5, 10$ across varied inference-time interaction budgets ($k \in 1, 2, 4, 6, 8, 10$). Results illustrated in Figure 7 reinforce previous observations, consistently showing superior Succ@ k performance by models trained under multi-turn conditions. Intriguingly, these improvements are observable even at the lowest inference budget ($k = 1$), suggesting that multi-turn training enhances not only iterative performance but also generalizes well to single-shot scenarios.

We further evaluate multi-turn reasoning capability transfer. Table 1 reports 5-turn success rates across six benchmarks spanning mathematical reasoning (MMQ-Math, TheoremQA), question answering (HotPotQA, ConcurrentQA), and general knowledge (MMLU, MMLU-Pro). **We observe that applying 5-turn UFO on top of task-specific reinforcement learning (RL) consistently improves performance across all domains.** Note that since MMLU is a four-way multiple-choice benchmark, and we allow the model to attempt up to five independent

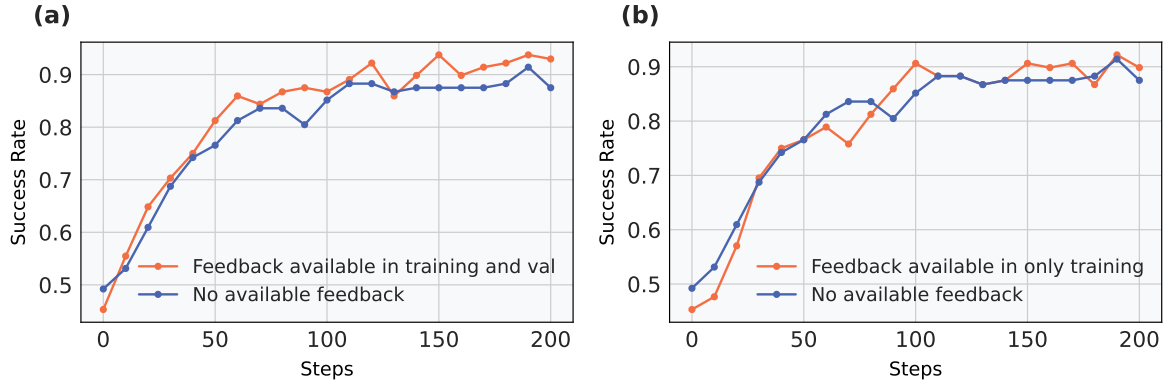


Figure 8 | Comparison of success rate with multi-turn setting. (a) with feedback prompt in both training and validation compared to blank prompt; (b) with feedback prompt only in training compared to blank prompt.

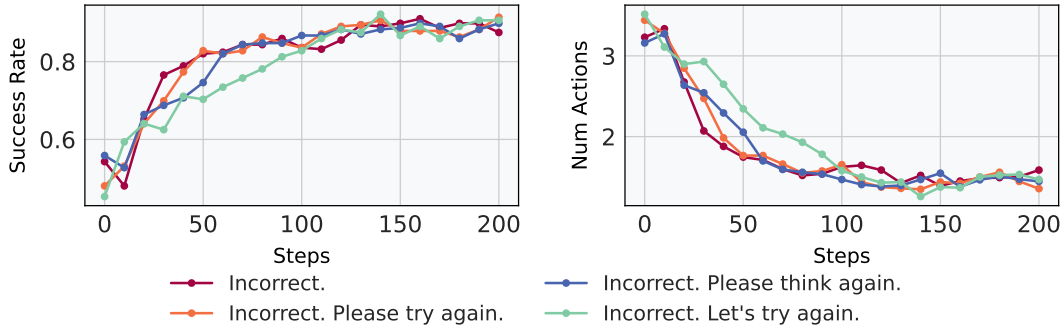


Figure 9 | Validation under different verbal feedback prompts. Success rates and action counts remain consistent across all variants, demonstrating UFO’s robustness to various prompts.

completions. Under this setting, the model is theoretically guaranteed to produce a correct answer within the five attempts. Nevertheless, we adopt this setting to ensure consistency with prior work and to allow fair comparison across different datasets.

On the in-domain MMQ-Math dataset, 5-turn UFO raises performance from 52.3% to 88.5%. Notably, the benefit of UFO extends beyond the source task: when models trained on MMQ-Math are evaluated on TheoremQA, HotPotQA, and ConcurrentQA, UFO provides gains of +8.8, +7.1, and +2.3 points respectively over RL alone, highlighting strong cross-task generalization. Similar trends are observed when training on HotPotQA, where UFO improves downstream TheoremQA (+0.9), MMLU (+5.1), and ConcurrentQA (+0.2). The most striking results occur in the general knowledge domain: although RL on MMQ-Math degrades MMLU performance from 75.2% to 66.8%, applying UFO not only recovers but surpasses the original accuracy, reaching 85.2%. On MMLU-Pro, UFO also boosts performance from 48.3% to 60.9%. These results collectively demonstrate that UFO enhances multi-turn reasoning while preserving and even improving generalization to out-of-domain tasks.

4.2.2. Multi-turn Setting Enables LRMs to Revise From Feedback

The multi-turn setting enables agents to engage repeatedly with each prompt (up to T_{\max} turns), thereby constructing richer and more informative interaction trajectories from the same training data. This enhanced utilization of feedback is hypothesized to extract more meaningful learning signals per problem, potentially improving solution quality and accelerating convergence, especially in data-limited contexts.

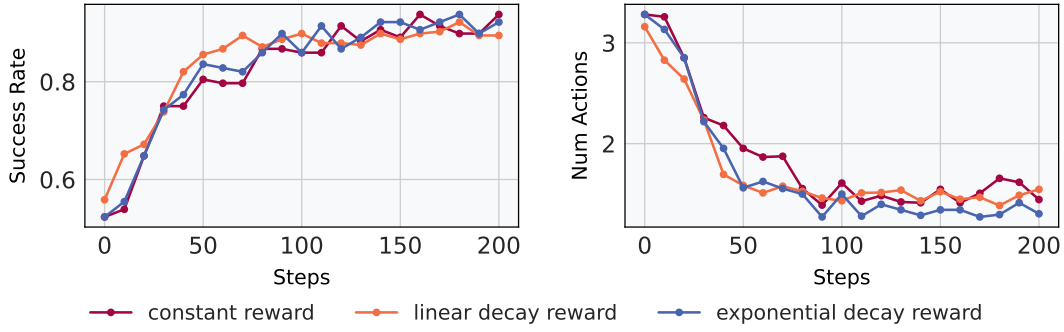


Figure 10 | Comparison of reward shaping strategies. While constant, linear decay, and exponential decay schedules achieve similar success rates (left), exponential decay consistently leads to fewer actions per episode (right), indicating more efficient problem solving with less external supervision.

To empirically validate that LRMs can be improved effectively utilizing conversational feedback for revision, we compared 5-turn training scenarios with and without explicit feedback prompts. Results presented in Figure 8(a) support this hypothesis, **demonstrating an over 8% peak performance improvement when explicit feedback is provided.**

An additional analysis with feedback prompt only in training (Figure 8(b)) reveals performance improvement as well. This suggests that multi-turn training can even intrinsically enhance model reasoning capabilities.

Finally, our robustness analysis in the Figure 9 shows that the effectiveness of this approach is preserved across a range of prompt formulations, underscoring its practical applicability in real-world scenarios.

4.2.3. Reward Shaping Encourages Efficient Problem Solving

We investigate how different reward schedules influence the agent’s learning behavior, particularly in encouraging early success versus allowing extended exploration. All schedules define a reward $r(n)$ based on the turn index n when the first correct answer is produced, with $n \in \{1, \dots, T_{\max}\}$.

We define and evaluate three distinct reward schedules. Following the formulas proposed in Section 3.4, we compare three approaches: (1) **Exponential Decay:** $r_{\text{exp}}(n) = \gamma^n$ (with $\gamma = 0.5$), (2) **Linear Decay:** $r_{\text{lin}}(n) = \max(0, 1 - 0.2(n - 1))$, (3) **Constant Reward:** $r_{\text{const}}(n) = 1$. All schedules operate for $n \in 1, \dots, T_{\max}$. The agent’s objective remains to maximize the expected cumulative reward.

Experimental validation (Figure 10) confirms that **exponential reward decay notably reduces the mean number of actions by roughly 10%**, without sacrificing overall success rates. This reduction in action count suggests that the exponential decay schedule encourages the model to engage in more profound self-reflection and systematic thinking before generating a response. By compelling the model to find solutions in fewer turns, it learns to be more deliberate and efficient, thus minimizing redundant interactions.

Considering the normalized penalty term in our experiment (Equation 7), we count the number of non-repetitive answer for each validation round, as shown in Figure 11. The percentage **increases from 80% to 90%**, suggesting that the model performs better in the later stages of training as the model learned to generate different responses better, reducing duplicate answers. This is an important measure of model performance, as high repetition rates lead to higher penalties and thus lower overall rewards. The chart shows that the model improved in this area

during training.

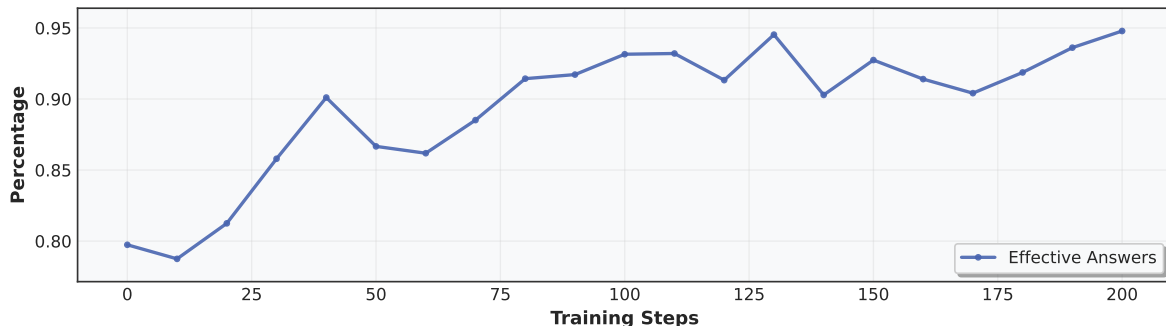


Figure 11 | Proportion of effective answers over training. The upward trend suggests improved diversity across turns, which reduces penalty from repeated responses and contributes to higher overall rewards.

5. Related Work

Enhancing LLM Reasoning and Multi-Turn Interactions. Test-time reasoning frameworks keep model weights frozen yet boost performance: Graph-of-Thought (Besta et al., 2023), Reflexion (Shinn et al., 2023), Monte Carlo Tree Self-Refine (Zhang et al., 2024), Self-Refine (Madaan et al., 2023), CRITIC (Gou et al., 2023) and memory-augmented agents such as POEM (Do et al., 2024) and Larimar (Das et al., 2024) rely on search, self-feedback or episodic memory *without* updating model parameters.

Training-time optimisation methods, in contrast, adjust the policy itself. RLHF (Ouyang et al., 2022; Christiano et al., 2017) and its low-cost variant RLAI (Lee et al., 2023) align models to preference data; scalable-oversight *debate* protocols explore alignment with weak judges (Kenton et al., 2024). Lightweight objectives such as Direct Preference Optimisation (DPO) (Rafailov et al., 2023), Parameter-Efficient RLHF (PERLHF) (Sidahmed et al., 2024) and Self-Play Fine-Tuning (SPIN) (Chen et al., 2024) further cut roll-out cost, while hierarchical ArCHer (Zhou et al., 2024) tackles long-horizon credit assignment. Benchmarks like UNO Arena (Qin et al., 2024) expose the strengths and weaknesses of both families in stateful, multi-turn settings.

Advancements in Mathematical Reasoning with Large Language Models. LLMs have advanced rapidly on mathematical benchmarks, from grade-school GSM8K (Cobbe et al., 2021) to Olympiad-level MATH (Hendrycks et al., 2021). Prompting innovations, like Chain-of-Thought (Wei et al., 2022), its self-consistent variant (Wang et al., 2022), and Tree/Graph-of-Thought (Besta et al., 2023; Yao et al., 2023b), made intermediate reasoning explicit. ReAct interleaves reasoning with environment actions (Yao et al., 2022), while tool-coupled approaches such as PAL (Gao et al., 2022), PoT (Chen et al., 2023b) and Toolformer (Schick et al., 2023) off-load heavy computation. Verifier pipelines boost reliability: from Let’s Verify Step-by-Step (Lightman et al., 2023) to AutoPSV (Lu et al., 2024), MATH-Shepherd (Wang et al., 2023) and progress-aware verifiers (Setlur et al., 2024). Binary-search debuggers like URSA locate first-error steps (Luo et al., 2025). Nevertheless, intrinsic self-correction remains limited (Huang et al., 2023).

6. Conclusions and Limitations

In this work, we highlight a critical limitation of current single-turn RL training: its tendency to impair multi-turn reasoning by promoting repetitive and shallow responses. To address this, we propose Unary Feedback as Observation (UFO), a simple yet effective method that

integrates minimal feedback into existing RL pipelines. UFO enables models to recover and improve both single-turn and multi-turn reasoning performance. Our experiments show a 14% gain in multi-turn accuracy while preserving single-turn quality. Additionally, we demonstrate that incorporating reward decay and repetitive penalty encourages deeper reasoning, self-correction and generating different responses. Our approach is lightweight, generalizable, and easily applicable to existing datasets. A limitation of our work is its primary focus on mathematical reasoning tasks, leaving its generalizability to broader reasoning domains for future investigation.

7. Acknowledgements

We thank the DeepSeek team for providing the DeepSeek-R1 model and early conceptual inspirations. We are grateful to the veRL team for their infrastructure support and the RAGEN team for their multi-turn RL framework. We thank Anna Zhao for their valuable contribution in helping revising manuscripts.

References

- DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL <https://arxiv.org/abs/2501.12948>.
- OpenAI. Gpt-4 technical report, 2024. URL <https://arxiv.org/abs/2303.08774>.
- 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, 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. *arXiv preprint arXiv:2412.15115*, 2024.
- Gemini Team. Gemini: A family of highly capable multimodal models, 2025. URL <https://arxiv.org/abs/2312.11805>.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL <https://arxiv.org/abs/1707.06347>.
- Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn rl, 2024. URL <https://arxiv.org/abs/2402.19446>.
- Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Monica Lam, 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, 2025. URL <https://arxiv.org/abs/2504.20073>.
- Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024. URL <https://arxiv.org/abs/2404.07972>.
- Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym, 2024. URL <https://arxiv.org/abs/2412.21139>.
- Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents, 2023a. URL <https://arxiv.org/abs/2207.01206>.

- Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning, 2021. URL <https://arxiv.org/abs/2010.03768>.
- Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. Mint: Evaluating llms in multi-turn interaction with tools and language feedback, 2024a. URL <https://arxiv.org/abs/2309.10691>.
- Yan Zhuang, Jiawei Ren, Xiaokang Ye, Xuhong He, Zijun Gao, Ryan Wu, Mrinaal Dogra, Cassie Zhang, Kai Kim, Bertt Wolfinger, Ziqiao Ma, Tianmin Shu, Zhiting Hu, and Lianhui Qin. Simworld: A world simulator for scaling photorealistic multi-agent interactions, 2025.
- Shiyi Cao, Sumanth Hegde, Dacheng Li, Tyler Griggs, Shu Liu, Eric Tang, Jiayi Pan, Xingyao Wang, Akshay Malik, Graham Neubig, Kourosh Hakhmaneshi, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Skyrl-v0: Train real-world long-horizon agents via reinforcement learning, 2025.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL <https://arxiv.org/abs/2203.02155>.
- 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, 2024. URL <https://arxiv.org/abs/2402.03300>.
- 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, 2025. URL <https://arxiv.org/abs/2503.09516>.
- Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjuan Zhong. Retool: Reinforcement learning for strategic tool use in llms, 2025. URL <https://arxiv.org/abs/2504.11536>.
- Qiyang 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, 2025. URL <https://arxiv.org/abs/2503.14476>.
- Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.
- Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model, 2025. URL <https://arxiv.org/abs/2503.24290>.
- Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamathqa: A dataset for mathematical reasoning with large language models, 2024. URL <https://arxiv.org/abs/2405.17633>.
- Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. Theoremqa: A theorem-driven question answering dataset, 2023a. URL <https://arxiv.org/abs/2305.12524>.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering, 2018. URL <https://arxiv.org/abs/1809.09600>.

- Simran Arora, Patrick Lewis, Angela Fan, Jacob Kahn, and Christopher Ré. Reasoning over public and private data in retrieval-based systems, 2022. URL <https://arxiv.org/abs/2203.11027>.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2020. URL <https://arxiv.org/abs/2009.03300>.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhui Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024b. URL <https://arxiv.org/abs/2406.01574>.
- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefer. Graph of thoughts: Solving elaborate problems with large language models, 2023. URL <https://arxiv.org/abs/2308.09687>.
- Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL <https://arxiv.org/abs/2303.11366>.
- Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree search self-refinement, 2024. URL <https://arxiv.org/abs/2406.07394>.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL <https://arxiv.org/abs/2303.17651>.
- Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-augmented critiquing, 2023. URL <https://arxiv.org/abs/2305.11738>.
- Dai Do, Quan Tran, Svetha Venkatesh, and Hung Le. Large language models prompting with episodic memory, 2024. URL <https://arxiv.org/abs/2408.07465>.
- Payel Das, Subhajit Chaudhury, Elliot Nelson, Igor Melnyk, Sarath Swaminathan, Sihui Dai, Aurélie Lozano, Georgios Kollias, Vijil Chenthamarakshan, Jiří Navrátila, Soham Dan, and Pin-Yu Chen. Larimar: Large language models with episodic memory control, 2024. URL <https://arxiv.org/abs/2403.11901>.
- Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2017. URL <https://arxiv.org/abs/1706.03741>.
- Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback, 2023. URL <https://arxiv.org/abs/2309.00267>.
- Zachary Kenton, Noah Y. Siegel, János Kramár, Jonah Brown-Cohen, Samuel Albanie, Jannis Bulian, Rishabh Agarwal, David Lindner, Yunhao Tang, Noah D. Goodman, and Rohin Shah. On scalable oversight with weak llms judging strong llms, 2024. URL <https://arxiv.org/abs/2407.04622>.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2023. URL <https://arxiv.org/abs/2305.18290>.
- Hakim Sidahmed, Samrat Phatale, Alex Hutcheson, Zhuonan Lin, Zhang Chen, Zac Yu, Jarvis Jin, Simral Chaudhary, Roman Komarytsia, Christiane Ahlheim, Yonghao Zhu, Bowen Li, Saravanan Ganesh, Bill Byrne, Jessica Hoffmann, Hassan Mansoor, Wei Li, Abhinav Rastogi, and Lucas Dixon. Parameter efficient reinforcement learning from human feedback, 2024. URL <https://arxiv.org/abs/2403.10704>.

- Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models, 2024. URL <https://arxiv.org/abs/2401.01335>.
- Zhanyue Qin, Haochuan Wang, Deyuan Liu, Ziyang Song, Cunhang Fan, Zhao Lv, Jinlin Wu, Zhen Lei, Zhiying Tu, Dianhui Chu, Xiaoyan Yu, and Dianbo Sui. Uno arena for evaluating sequential decision-making capability of large language models, 2024. URL <https://arxiv.org/abs/2406.16382>.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL <https://arxiv.org/abs/2110.14168>.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL <https://arxiv.org/abs/2103.03874>.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2022. URL <https://arxiv.org/abs/2201.11903>.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain-of-thought reasoning in language models, 2022. URL <https://arxiv.org/abs/2203.11171>.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023b. URL <https://arxiv.org/abs/2305.10601>.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2022. URL <https://arxiv.org/abs/2210.03629>.
- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models, 2022. URL <https://arxiv.org/abs/2211.10435>.
- Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning, 2023b. URL <https://arxiv.org/abs/2211.12588>.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023. URL <https://arxiv.org/abs/2302.04761>.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, and Ilya Sutskever and Karl Cobbe. Let’s verify step by step, 2023. URL <https://arxiv.org/abs/2305.20050>.
- Jianqiao Lu, Zhiyang Dou, Hongru Wang, Zeyu Cao, Jianbo Dai, Yingjia Wan, and Zhijiang Guo. Autopsv: Automated process-supervised verifier, 2024. URL <https://arxiv.org/abs/2405.16802>.
- Peiyi Wang, Lei Li, Zhihong Shao, R.X. Xu, Damai Dai, Yifei Li, Deli Chen, Y.Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step by step, 2023. URL <https://arxiv.org/abs/2312.08935>.
- Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning, 2024. URL <https://arxiv.org/abs/2410.08146>.
- Ruilin Luo, Zhuofan Zheng, Yifan Wang, Xinzhe Ni, Zicheng Lin, Songtao Jiang, Yiyao Yu, Chufan Shi, Ruihang Chu, Jin Zeng, and Yujiu Yang. Ursa: Understanding and verifying chain-of-thought reasoning in large language models, 2025. URL <https://arxiv.org/abs/2501.04686>.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet, 2023. URL <https://arxiv.org/abs/2310.01798>.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2015. URL <https://arxiv.org/abs/1506.02438>.

Appendix

Table of Contents

A	Extended Background of Reinforcement Learning in LLMs	18
B	Detailed Evaluation under Multi-round Settings	19
C	Prompt Settings	19
C.1	Problem Solving Model Prompt	19
C.2	Feedback Provider Model Prompt	21
C.3	TheoremQA Prompt Format	21
D	Model Evaluation Details	22
E	Case Analyses	22
E.1	Case 1: Pre-training Behavior	22
E.2	Case 2: Post Single-turn RL	23
E.3	Case 3: Success Adaptation to Feedback Through Multi-turn RL with UFO	23
E.4	Case 4: Reasoning Drift of Multi-turn RL with UFO	23

A. Extended Background of Reinforcement Learning in LLMs

Reinforcement Learning (RL) enables large language models to improve through interaction and reward feedback. The general RL objective maximizes the expected reward over sampled responses:

$$J(\theta) = \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(\cdot | x)} [R(x, y)], \quad (10)$$

where π_θ is the model policy, x is the input prompt, y is the generated output, and $R(x, y)$ is a scalar reward assessing response quality.

A widely adopted method for RL fine-tuning is Proximal Policy Optimization (PPO) (Schulman et al., 2017), which stabilizes training by clipping the likelihood ratio between the new and old policies. The ratio is defined as:

$$\rho_t(\theta) = \frac{\pi_\theta(y_t | x_t)}{\pi_{\theta_{\text{old}}}(y_t | x_t)}. \quad (11)$$

The PPO objective minimizes over the clipped surrogate advantage:

$$J_{\text{PPO}}(\theta) = \mathbb{E}_t [\min(\rho_t A_t, \hat{\rho}_t A_t) - \beta D_{\text{KL}}], \quad (12)$$

where $\hat{\rho}_t = \text{clip}(\rho_t, 1 - \epsilon, 1 + \epsilon)$, and A_t is the advantage function estimating how much better y_t is than the baseline under prompt x_t .

For advantage estimation, Generalized Advantage Estimation (GAE) (Schulman et al., 2015) is often used:

$$A_t^{\text{GAE}(\gamma, \lambda)} = \sum_{l=0}^{\infty} (\gamma \lambda)^l \delta_{t+l}, \quad \text{with} \quad \delta_t = r_t + \gamma V(x_{t+1}) - V(x_t), \quad (13)$$

where (γ, λ) trade off bias and variance.

More recently, DeepSeekMath (Shao et al., 2024) and DeepSeek-R1 (DeepSeek-AI, 2025) adopts Group Relative Policy Optimization (GRPO), a RL method that samples a set of outputs $\{y_i\}_{i=1}^G$ for each prompt x , and optimizes:

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{x, \{y_i\}} [J_{\text{group}}(\theta)], \quad (14)$$

with

$$J_{\text{group}}(\theta) = \frac{1}{G} \sum_{i=1}^G \min(\rho_i A_i, \hat{\rho}_i A_i) - \beta D_{\text{KL}}, \quad (15)$$

where the advantage A_i is computed using a reward-normalized baseline:

$$A_i = \frac{r_i - \text{mean}(\{r_j\})}{\text{std}(\{r_j\})}. \quad (16)$$

This avoids dependency on value networks and uses rule-based or environment-specific rewards r_i , making it well-suited for reasoning tasks where explicit heuristics can guide learning. GRPO has shown to induce emergent multi-step reasoning behavior across domains.

B. Detailed Evaluation under Multi-round Settings

We illustrate a detailed analysis of how multi-round training improves generalization on long-horizon interactive reasoning. Figure 12 provides a comprehensive view of validation performance across all checkpoints, comparing models trained under 1-round, 5-round, and 10-round settings. Each curve represents evaluation success rates under a fixed number of evaluation rounds.

We observe that under 1-round evaluation (top-left), all training strategies achieve similar performance, suggesting that even single-turn training can suffice in this limited setting. However, as evaluation round count increases, the gap between single-round training and multi-round training becomes increasingly significant. In particular, models trained with 10-round UFO feedback consistently outperform the others under 6, 8, and 10-round evaluation, demonstrating more stable and generalizable behavior across turns.

These results support our core hypothesis: **unary feedback, when used as structured observation during training, enables better long-horizon generalization.** In contrast, models trained only with single-round interactions struggle to adapt to multi-turn dynamics, leading to degraded performance as the task horizon increases.

C. Prompt Settings

C.1. Problem Solving Model Prompt

We adopt a simple and structured prompt format for mathematical problem solving, following prior designs from Yang et al. (2024); Shao et al. (2024), with an extension to support multi-turn interactions. A key element of our prompt is the explicit `<think>` and `<answer>` separation, paired with an action budget (Y) and max length (Z). This guides the model to reason step-by-step while planning within a fixed turn horizon, improving controllability and alignment in multi-turn settings. As shown in Box 1, we present the prompt template used during training and evaluation.

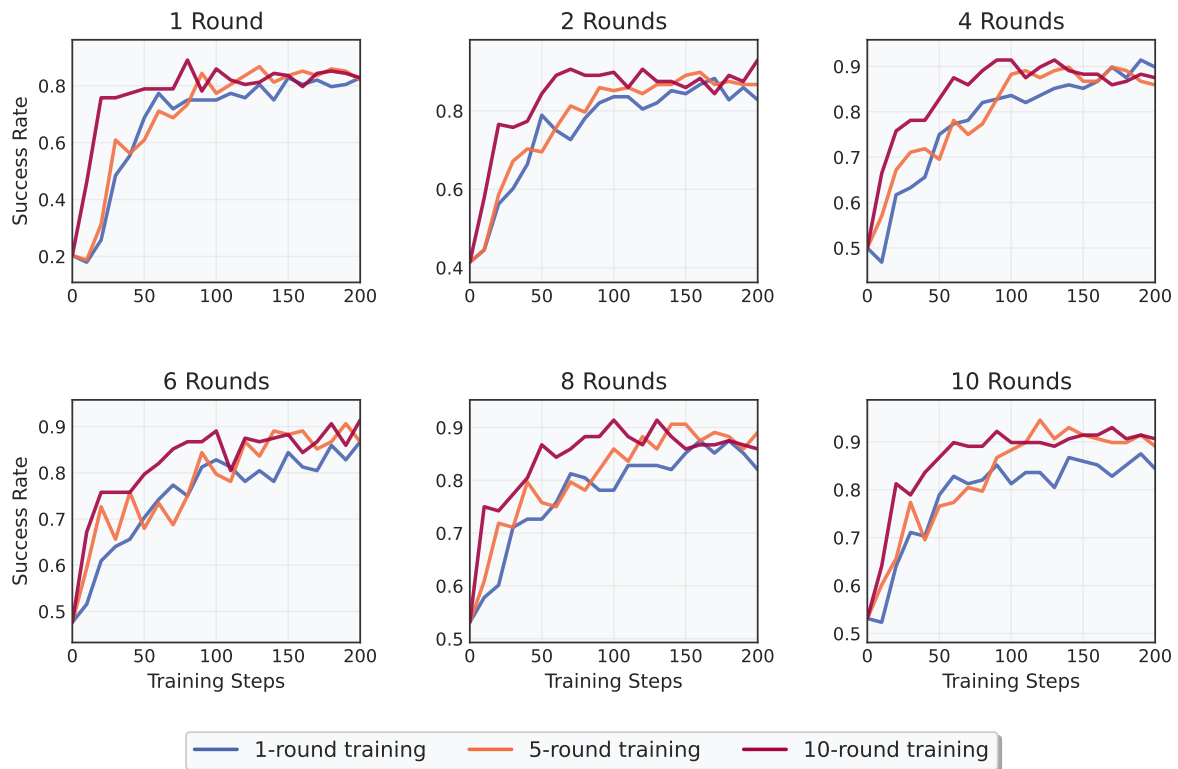


Figure 12 | Performance across different evaluation round settings. Each subplot shows the success rate evaluated at r rounds. While all methods perform similarly under 1-round evaluation, models trained with multi-round feedback (UFO) generalize significantly better to longer evaluation horizons.

Box 1: Model Prompt Template

```

<|im_start|>system
{prompt}
You're a helpful assistant.
<|im_end|>
<|im_start|>user
{prompt}
You are solving Math problems.
Turn X:
State:
(Question)
You have Y actions left. Always output: <think> [Your thoughts]
</think> <answer> [your answer] </answer> with no extra text.
Strictly follow this format. Max response length: Z words
(tokens).
<|im_end|>
<|im_start|>assistant
... (This conversation pattern repeats for up to K turns)
<|im_end|>

```


C.2. Feedback Provider Model Prompt

We also present the prompt for the feedback provider that gives the problem-solving model more detailed feedback as follows.

Box 2: Tutor Prompt Template

```
<|im_start|>system
You are a helpful math tutor.
<|im_end|>
<|im_start|>user
Problem: {question}
Student's answer: {wrong_answer}
This answer is incorrect. Give a brief, encouraging hint (1-2
sentences) that helps the student reconsider their approach without
revealing the correct answer. Focus on guiding them to check their
work or think about the problem differently.
Response format: Just the hint, no extra formatting.
<|im_end|>
<|im_start|>assistant
...
<|im_end|>
```

C.3. TheoremQA Prompt Format

The TheoremQA environment follows a similar prompting structure as the MMQ-Math, with an additional image token placed at the beginning of the question when a picture is present. This enables compatibility with multimodal pipelines, where the image is processed separately while the text prompt includes a placeholder token to signal its presence.

As shown in Box 3, the image token `<image>` is placed on a new line above the question if an image is available.

Box 3: TheoremQA Prompt Template

```
<|im_start|>system
You're a helpful assistant.
<|im_end|>
<|im_start|>user
{prompt}
You are solving Math problems.
Turn X:
State:
<image>
+ (Question)
You have Y actions left. Always output: <think>[Your
thoughts]</think><answer>[your answer]</answer> with no extra
text. Strictly follow this format. Max response length: Z words
(tokens).
<|im_end|>
<|im_start|>assistant
...
```

Table 2 | Hugging Face model names used in the unique answer ratio evaluation.

Method	Model (Hugging Face name)
DAPO	Qwen/Qwen2.5-32B BytedTsinghua-SIA/DAPO-Qwen-32B
Dr. GRPO	Qwen/Qwen2.5-Math-7B sail/Qwen2.5-Math-7B-Oat-Zero Qwen/Qwen2.5-Math-1.5B sail/Qwen2.5-Math-1.5B-Oat-Zero
GRPO	Qwen/Qwen2.5-0.5B hkust-nlp/Qwen-2.5-0.5B-SimpleRL-Zoo Qwen/Qwen2.5-1.5B hkust-nlp/Qwen-2.5-1.5B-SimpleRL-Zoo Qwen/Qwen2.5-7B hkust-nlp/Qwen-2.5-7B-SimpleRL-Zoo Qwen/Qwen2.5-14B hkust-nlp/Qwen-2.5-14B-SimpleRL-Zoo
PPO	Qwen/Qwen2.5-Math-7B RLHFlow/Qwen2.5-7B-PPO-Zero Qwen/Qwen2.5-3B-Instruct LichengLiu03/Qwen2.5-3B-UFO

<|im_end|>

D. Model Evaluation Details

We present the model used to evaluate answer repetition in Table 2.

E. Case Analyses

We investigate the impact of multi-turn reinforcement learning (RL) on large language models (LLMs) through a series of curated examples across distinct training stages. These case studies (shown in Boxes 4–7) highlight the evolving dynamics of exploration, convergence, and reasoning quality throughout training.

E.1. Case 1: Pre-training Behavior

Before any reinforcement learning, we observe the model’s default multi-turn reasoning behavior in a symbolic pattern-matching task (Box 4). The model is asked to recover a missing variable X from a repeating alphabetic value pattern, given partial information and a constraint on the sum of values in a specific word. In Turn 1, the model identifies the relevant positions in the word “numeric” and proposes an initial guess for X . As feedback indicates the answer is incorrect, the model progressively refines its understanding: it attempts to align characters in the input word with their positions in the pattern and adjusts its value for X .

Despite making several wrong guesses, the model demonstrates **adaptive behavior across turns**: it updates its assumptions, introduces new hypotheses, and makes meaningful structural

progress (e.g., recognizing the 8-length cycle). However, it ultimately fails to reach the correct solution within the available steps. The case shows that **pretrained models already possess multi-step reflective capabilities and can utilize external feedback to revise their reasoning**, even without explicit training for multi-turn alignment. It suggests that reinforcement learning has the potential to further stabilize and guide emergent reasoning process toward convergence.

E.2. Case 2: Post Single-turn RL

After reinforcement learning with single-step reward feedback, the model demonstrates drastically different behavior from its pretrained counterpart (Box 5). When tasked with identifying the variable X in a cyclic pattern-based word problem, the model immediately commits to a single interpretation. In Turn 1, it attempts a symbolic derivation by aligning the letter values of “numeric” with a fixed periodic pattern, then solving $X - 3 = -1$. However, this derivation mistakenly assumes that the letters in “numeric” correspond to the first 7 elements of the pattern without justifying the mapping. More notably, this exact sequence of logic and answer is **repeated identically** in Turns 2 through 5.

The case reveals that single-turn RL induces brittle, overconfident behavior: once the model settles on a trajectory during initial inference, it does not reconsider alternative hypotheses or respond meaningfully to corrective feedback. The reward optimization has led to collapse in exploration, as each turn simply replays the same incorrect reasoning with no adaptation. In contrast to the pretraining stage, where the model at least attempts different strategies, this behavior illustrates a major drawback of single-step reward supervision: it teaches the model what to say once, but not how to revise when it’s wrong.

E.3. Case 3: Success Adaptation to Feedback Through Multi-turn RL with UFO

This example illustrates the effectiveness of multi-turn reinforcement learning (Box 6). The model is prompted to determine the sum of all positive integers n for which $\frac{n+18}{n}$ is an integer. In Turn 1, it begins by simplifying the expression to $1 + \frac{18}{n}$, and attempts a partial answer without listing all divisors. Upon receiving feedback, the model updates its understanding in Turn 2 by enumerating all positive divisors of 18. By Turn 3, it completes the reasoning process by summing those divisors correctly, arriving at the correct final answer of 39.

The case shows a successful case of multi-turn self-correction, where the model refines its reasoning incrementally in response to feedback. Each turn builds on the previous one: the model first identifies the mathematical form, then retrieves the correct domain knowledge (divisors), and finally executes a complete and valid computation. Unlike single-turn RL, **the multi-turn reward structure encourages adaptive reasoning and gradual convergence**. The example shows that properly trained multi-turn agents can both reason correctly and update their behavior mid-trajectory in pursuit of a verifiable answer.

E.4. Case 4: Reasoning Drift of Multi-turn RL with UFO

We examine a failure trajectory under multi-turn reinforcement learning where the model arrives at the correct final answer through incorrect or misaligned reasoning. As shown in Box 7, the task is to find the value of X such that the matrix $\begin{pmatrix} X & 3 \\ -1 & 7 \end{pmatrix}$ has an inverse consistent with the given answer “17.” In the initial turn, the model misinterprets the problem structure by assuming that the matrix inverse itself is the scalar 17, and raises a structural objection. In the next turn, it shifts to a more plausible interpretation where 17 is assumed to be the determinant. This leads

Table 3 | Comparison of multi-turn reasoning behaviors across training stages.

Case	Stage	Exploration	Convergence	Reasoning Quality	Failure Mode
1	Pre-RL	High	No	Incomplete	Early guessing
2	Single-turn RL	None	No	Repetitive	Overfitting
3	Multi-turn RL	Moderate	Yes	Correct	aligned
4	Multi-turn RL	Moderate	Yes	Incorrect	Reasoning drift

to the formulation of the correct equation $7X + 3 = 17$, though the value of X is still miscalculated at that stage.

By the third turn, the model correctly solves for $X = 2$, but then presents a flawed expression for the inverse matrix, misapplying the standard inversion formula for 2×2 matrices. Despite producing the right numerical solution, the trajectory contains conceptual errors related to matrix algebra, including an incorrect cofactor layout. This case highlights a form of reasoning drift: **while the final output aligns with the ground truth, the intermediate steps do not constitute a logically valid derivation.** Such behavior reveals a potential failure of reward design in multi-turn RL, where correctness of the answer is reinforced even in the absence of faithful reasoning.

To summarize (as in Table 3), multi-turn RL with UFO improves convergence and stabilizes reasoning, but naive reward design can still lead to misalignment between thought and outcome. It highlights the need for evaluation frameworks to go beyond final answer correctness and explicitly assess reasoning integrity throughout the trajectory.

Box 4: Multi-Turn Rollout before training

Turn 1:

State: The letters of the alphabet are given numeric values based on the two conditions below.

- Only the numeric values of $-2, -1, 0, 1$ and 2 are used.
- Starting with A and going through Z, a numeric value is assigned to each letter according to the following pattern:

$$1, 2, X, 0, -1, -2, -1, 0, 1, 2, 1, 0, -1, -2, -1, 0, \dots$$

Two complete cycles of the pattern are shown above. The letter A has a value of 1, B has a value of 2, F has a value of -2 and Z has a value of 2. The sum of the numeric values of the letters in the word “numeric” is -1 . What is the value of unknown variable X ? You have 10 actions left. Always output: `<think>` [Your thoughts] `<think>` `<answer>` [your answer] `<answer>` with no extra text. Strictly follow this format. Max response length: 100 words (tokens).

Output:

`<think>`First, let’s identify the positions of each letter in the word “numeric” based on the pattern provided. We know the values of A, B, F, and Z. We need to determine the value of X and then find the values of N, M, E, and C to sum up to -1 .`</think>`
`<answer>` $X = 2$ `</answer>`

Reward: 0.0

Turn 2:

State: Incorrect. Please try again. You have 3 actions left. Always output: `<think>` [Your thoughts] `<think>` `<answer>` [your answer] `<answer>` with no extra text. Strictly follow

this format. Max response length: 100 words (tokens).

Output:

<think>Given the incorrect answer, let's re-examine the pattern and the values assigned to each letter. The word "numeric" has the letters N, U, M, B, E, I, C. From the provided pattern, we know the values of A, B, F, and Z. To find the value of X, we need to identify the value for N and then determine the pattern for subsequent letters.</think>

<answer>X = 0</answer>

Reward: 0.0

Turn 3:

State: Incorrect. Please try again. You have 2 actions left. Always output: <think> [Your thoughts] <think> <answer> [your answer] <answer> with no extra text. Strictly follow this format. Max response length: 100 words (tokens).

Output:

<think>Given the incorrect answer, let's re-evaluate the pattern and the values assigned to each letter in the word "numeric". The values for A, B, F, and Z are known. The pattern seems to repeat every 8 letters, and the values cycle through 1, 2, X, 0, -1, -2, -1, 0. We need to determine the value of X such that the sum of the values of the letters in "numeric" equals -1.</think>

<answer>X = -1</answer>

Reward: 0.0

Turn 4:

State: Incorrect. Please try again. You have 1 actions left. Always output: <think> [Your thoughts] <think> <answer> [your answer] <answer> with no extra text. Strictly follow this format. Max response length: 100 words (tokens).

Output:

<think>Given the incorrect answer, let's re-examine the pattern and the values assigned to each letter in the word "numeric". The values for A, B, F, and Z are known, and the pattern seems to repeat every 8 letters with values 1, 2, X, 0, -1, -2, -1, 0. To find the value of X such that the sum of the values of the letters in "numeric" equals -1, we need to assign values to N, U, M, B, E, I, C and ensure their sum is -1</think>

<answer>X = 1</answer>

Reward: 0.0

Turn 5:

State: Incorrect. Please try again. You have 0 actions left. Always output: <think> [Your thoughts] <think> <answer> [your answer] <answer> with no extra text. Strictly follow this format. Max response length: 100 words (tokens).

Output:

<think>Given the incorrect answer, let's re-examine the pattern and the values assigned to each letter in the word "numeric". The values for A, B, F, and Z are known, and the pattern seems to repeat every 8 letters with values 1, 2, X, 0, -1, -2, -1, 0. To find the value of X such that the sum of the values of the letters in "numeric" equals -1, we need to assign values to N, U, M, B, E, I, C and ensure their sum is -1.</think>

<answer>X = -1</answer>

Reward: 0.0

Box 5: Model rollout after single-turn RL training

Turn 1:

State: The letters of the alphabet are given numeric values based on the two conditions below.

- Only the numeric values of $-2, -1, 0, 1$ and 2 are used.
- Starting with A and going through Z, a numeric value is assigned to each letter according to the following pattern:

$1, 2, X, 0, -1, -2, -1, 0, 1, 2, 1, 0, -1, -2, -1, 0, \dots$

Two complete cycles of the pattern are shown above. The letter A has a value of 1, B has a value of 2, F has a value of -2 and Z has a value of 2. The sum of the numeric values of the letters in the word “numeric” is -1 . What is the value of unknown variable X? You have 10 actions left. Always output: `<think> [Your thoughts] <think> <answer> [your answer] <answer>` with no extra text. Strictly follow this format. Max response length: 100 words (tokens).

Output:

`<think>`For the word “numeric”, the values of the letters are as follows: N has a value of 1, U has a value of 2, M has a value of 0, E has a value of -1 , R has a value of -2 , I has a value of -1 , C has a value of 1. The sum of these values is $1 + 2 + 0 - 1 - 2 - 1 + 1 = 0$, which contradicts the given sum of -1 . However, we can re-evaluate the pattern and the values. The pattern repeats every 16 letters, and the values repeat as well. The word “numeric” consists of 7 letters, so we need to find the sum of the first 7 values in the pattern: $1, 2, X, 0, -1, -2, -1$. We know the sum of these values is -1 . So, we have $1 + 2 + X + 0 - 1 - 2 - 1 = -1$, which simplifies to $X - 3 = -1$, and thus $X = 2$.`</think>`
`<answer>2</answer>`

Reward: 0.0

Turn 2:

State: Incorrect. Please try again. You have 3 actions left. Always output: `<think> [Your thoughts] <think> <answer> [your answer] <answer>` with no extra text. Strictly follow this format. Max response length: 100 words (tokens).

Output:

`<think>`For the word “numeric”, the values of the letters are as follows: N has a value of 1, U has a value of 2, M has a value of 0, E has a value of -1 , R has a value of -2 , I has a value of -1 , C has a value of 1. The sum of these values is $1 + 2 + 0 - 1 - 2 - 1 + 1 = 0$, which contradicts the given sum of -1 . The pattern repeats every 16 letters, and the values repeat as $1, 2, X, 0, -1, -2, -1, 0, 1, 2, 1, 0, -1, -2, -1, 0$. For the word “numeric”, the values are $1, 2, X, 0, -1, -2, -1$. The sum of these values is $1 + 2 + X + 0 - 1 - 2 - 1 = -1$, which simplifies to $X - 3 = -1$, and thus $X = 2$.`</think>`
`<answer>2</answer>`

Reward: 0.0

Turn 3:

State: Incorrect. Please try again. You have 2 actions left. Always output: `<think> [Your thoughts] <think> <answer> [your answer] <answer>` with no extra text. Strictly follow this format. Max response length: 100 words (tokens).

Output:

`<think>`For the word “numeric”, the values of the letters are as follows: N has a value of 1, U has a value of 2, M has a value of 0, E has a value of -1 , R has a value of -2 , I has a value of -1 , C has a value of 1. The sum of these values is $1 + 2 + 0 - 1 - 2 - 1 + 1 = 0$,

which contradicts the given sum of -1. The pattern repeats every 16 letters, and the values repeat as 1, 2, X, 0, -1, -2, -1, 0, 1, 2, 1, 0, -1, -2, -1, 0. For the word "numeric", the values are 1, 2, X, 0, -1, -2, -1. The sum of these values is $1 + 2 + X + 0 - 1 - 2 - 1 = -1$, which simplifies to $X - 3 = -1$, and thus $X = 2$.

<answer>2</answer>

Reward: 0.0

Turn 4:

State: Incorrect. Please try again. You have 1 action left. Always output: <think> [Your thoughts] <think> <answer> [your answer] <answer> with no extra text. Strictly follow this format. Max response length: 100 words (tokens).

Output:

<think>For the word "numeric", the values of the letters are as follows: N has a value of 1, U has a value of 2, M has a value of 0, E has a value of -1, R has a value of -2, I has a value of -1, C has a value of 1. The sum of these values is $1 + 2 + 0 - 1 - 2 - 1 + 1 = 0$, which contradicts the given sum of -1. The pattern repeats every 16 letters, and the values repeat as 1, 2, X, 0, -1, -2, -1, 0, 1, 2, 1, 0, -1, -2, -1, 0. For the word "numeric", the values are 1, 2, X, 0, -1, -2, -1. The sum of these values is $1 + 2 + X + 0 - 1 - 2 - 1 = -1$, which simplifies to $X - 3 = -1$, and thus $X = 2$.

<answer>2</answer>

Reward: 0.0

Turn 5:

State: Incorrect. Please try again. You have 0 actions left. Always output: <think> [Your thoughts] <think> <answer> [your answer] <answer> with no extra text. Strictly follow this format. Max response length: 100 words (tokens).

Output:

<think>For the word "numeric", the values of the letters are as follows: N has a value of 1, U has a value of 2, M has a value of 0, E has a value of -1, R has a value of -2, I has a value of -1, C has a value of 1. The sum of these values is $1 + 2 + 0 - 1 - 2 - 1 + 1 = 0$, which contradicts the given sum of -1. The pattern repeats every 16 letters, and the values repeat as 1, 2, X, 0, -1, -2, -1, 0, 1, 2, 1, 0, -1, -2, -1, 0. For the word "numeric", the values are 1, 2, X, 0, -1, -2, -1. The sum of these values is $1 + 2 + X + 0 - 1 - 2 - 1 = -1$, which simplifies to $X - 3 = -1$, and thus $X = 2$.

<answer>2</answer>

Reward: 0.0

Box 6: Model rollout after multi-turn RL training with UFO (Success Case)

Turn 1:

State:

Determine the sum of all positive integer values of n such that $\frac{n+18}{n}$ is an integer.

You have 10 actions left. Always output: <think> [Your thoughts] </think> <answer> [your answer] </answer> with no extra text. Strictly follow this format. Max response length: 100 words.

Output:

<think>First, I need to understand when $\frac{n+18}{n}$ is an integer. This fraction can be simplified to $1 + \frac{18}{n}$. For this to be an integer, n must be a divisor of 18.

<answer>18 and 1 (since $1 + \frac{18}{18} = 2$ and $1 + \frac{18}{1} = 19$ are integers)</answer>

Reward: 0.0

Turn 2:

State:

Incorrect. Please think again.

You have 9 actions left.

Output:

<think>Given $\frac{n+18}{n}$ is an integer, we have $1 + \frac{18}{n}$ as an integer. This means n must be a divisor of 18. The positive divisors of 18 are 1, 2, 3, 6, 9, and 18. I need to verify which of these values make the expression an integer.</think>

<answer>1, 2, 3, 6, 9, 18</answer>

Reward: 0.0

Turn 3:

State:

Incorrect. Please think again.

You have 8 actions left.

Output:

<think>Upon reviewing, I realize the divisors of 18 are 1, 2, 3, 6, 9, and 18. For each of these values, $1 + \frac{18}{n}$ is indeed an integer. Therefore, the sum of all positive integer values of n is $1 + 2 + 3 + 6 + 9 + 18 = 39$.</think>

<answer>39</answer>

Reward: 0.25

Box 7: Model rollout after multi-turn RL training with UFO (Failure Case)

Turn 1:

State:

Find the inverse of the matrix

$$\begin{pmatrix} X & 3 \\ -1 & 7 \end{pmatrix}.$$

If the inverse does not exist, then enter the zero matrix. The answer is 17. What is the value of unknown variable X ?

You have 10 actions left. Always output: <think> [Your thoughts] </think> <answer> [your answer] </answer> with no extra text. Strictly follow this format. Max response length: 100 words.

Output:

<think>Since the inverse of a matrix is given as 17, and the matrix provided is 2x2, the inverse should be a 2x2 matrix. However, the given inverse of 17 does not align with the structure of a 2x2 matrix. The problem might have a mistake in the provided inverse value or the matrix structure.</think>

<answer>There is an inconsistency in the problem statement. The given inverse of 17 does not match the structure of a 2x2 matrix inverse.</answer>

Reward: 0.0

Turn 2:

State:

Incorrect. Please think again.

You have 9 actions left.

Output:

<think>Given the inverse of the matrix is provided as 17, which does not fit the expected format of a 2x2 matrix inverse (which should be a 2x2 matrix), it seems there might be a misunderstanding. However, since the problem statement specifies the inverse as 17, and considering the structure of the matrix, it implies the inverse is a scalar value.

For a 2x2 matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$, the inverse is $\frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$. If the inverse is simply 17, it suggests $ad - bc = 17$. Given the matrix $\begin{pmatrix} X & 3 \\ -1 & 7 \end{pmatrix}$, $ad - bc = X \cdot 7 - 3 \cdot (-1) = 7X + 3$. Thus,

$7X + 3 = 17$.

<answer>7</answer>

Reward: 0.0

Turn 3:

State:

Incorrect. Please think again.

You have 8 actions left.

Output:

<think>Given the matrix $\begin{pmatrix} X & 3 \\ -1 & 7 \end{pmatrix}$, the condition for the inverse to be a scalar 17 is $7X + 3 = 17$. Solving for X gives $7X = 14$, thus $X = 2$. The inverse of the matrix should be $\frac{1}{17} \begin{pmatrix} 7 & -3 \\ 1 & X \end{pmatrix}$, which simplifies to $\begin{pmatrix} \frac{7}{17} & -\frac{3}{17} \\ \frac{1}{17} & 2 \end{pmatrix}$, not just 17.**</think>**

<answer>2</answer>

Reward: 0.25