
Let's Try Again: Eliciting Multi-Turn Reasoning in Language Models via Simplistic Feedback

Licheng Liu*

Imperial College London
licheng.liu22@imperial.ac.uk

Zihan Wang*

Northwestern University
zihanw@u.northwestern.edu

Linjie Li

University of Washington
linjli@cs.washington.edu

Chenwei Xu

Northwestern University
cxu@u.northwestern.edu

Yiping Lu

Northwestern University
yiping.lu@northwestern.edu

Han Liu

Northwestern University
hanliu@northwestern.edu

Avirup Sil

IBM Research AI
avi@us.ibm.com

Manling Li

Northwestern University
manling.li@northwestern.edu

Abstract

Large Language Models (LLMs) are increasingly deployed as agents that solve problems through multi-turn interaction, receiving feedback and refining their reasoning based on users' feedback. However, existing reinforcement learning with verifiable reward (RLVR) methods train them under a single-turn paradigm. As a result, we discovered that models often fail to explore alternative reasoning paths or reflect on prior mistakes, producing repetitive and unadapted responses to feedback. To address this gap, we propose Unary Feedback as Observation (UFO), a framework that conditions policy updates on minimal unary feedback (e.g., "Let's try again") after incorrect answers. UFO is simple, compatible with existing single-turn RL setups, and incentivizes self-reflection. To further promote efficient and adaptive reasoning, we design reward structures that encourage *minimality* (solving in fewer turns) and *diversity* (exploring alternatives under failure). Experiments show that UFO preserves single-turn performance while improving multi-turn reasoning accuracy by about 14%. Crucially, UFO-trained models also generalize beyond their training domain, transferring effectively to out-of-domain tasks across mathematics, STEM, QA, and general knowledge, showing that UFO teaches models self-reflective reasoning that carry over across domains. Beyond these empirical gains, UFO points toward a broader paradigm for building adaptive reasoning agents: one that scales supervision from static datasets, reduces dependence on costly domain-specific feedback, and lays the foundation for more general, self-improving AI systems in open-ended real-world settings.

1 Introduction

Large language and reasoning models (LLMs/LRMs) [1, 2, 3, 4] have demonstrated strong capabilities in solving complex tasks like math and code generation. Reinforcement learning with verifiable rewards (RLVR) [1, 5, 6, 7] further enhances their reasoning ability. However, real-world applications such as chatbots and programming assistants [8, 9, 10, 11, 12] demand multi-turn problem solving, a setting where current single-turn RL training falls short.

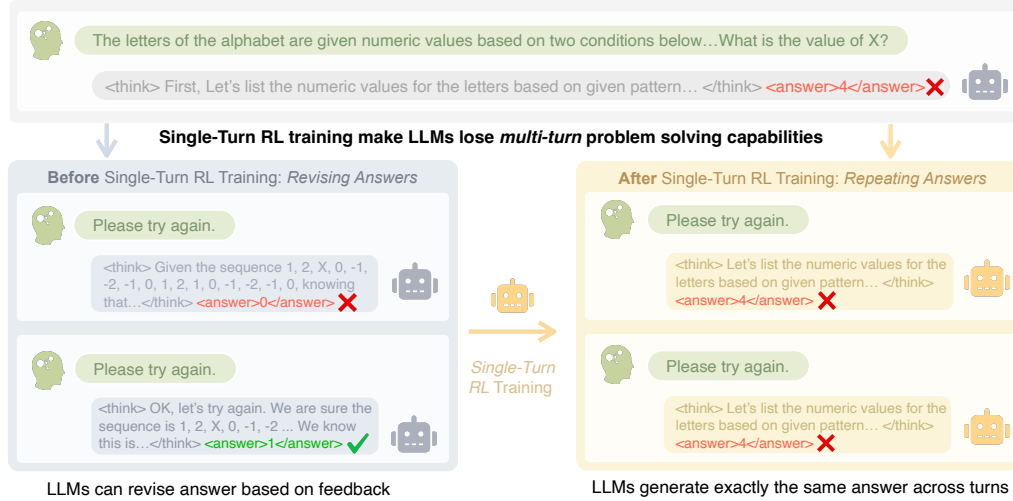


Figure 2: Single-turn RL causes LLMs to repeat answer across turns instead of revising based on feedback.

In this work, we first observe that single-turn RL can hinder a model’s ability to engage in interactive multi-turn reasoning. Specifically, models trained with single-turn RL often **repeat the same answer across turns** (Figure 2). One particular reason for that is that most existing datasets are inherently single-turn and lack signals for iterative exploration. Without such signals, models have limited opportunities to explore diverse reasoning paths or adapt their strategies based on feedback. Collecting real-world multi-turn user feedback is expensive and challenging. Existing multi-turn RL framework rely on automated signals from code interpreters [8, 9, 12] or simulators [11, 13], which are costly to design and limited in scope [14]. To address this, we propose a simple yet effective method that enables multi-turn RL using static datasets.

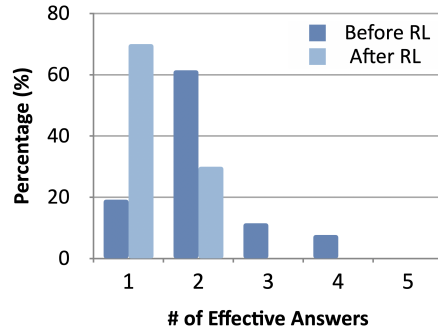


Figure 1: After single-turn RL training, the model gives the same answer across five turns in 70% of the cases.

By inserting minimal verbal feedback (e.g., “try again”) into the context, we prompt models to revise their reasoning. We call this approach **Unary Feedback as Observation (UFO)**, which treats interactive problem solving as an MDP where unary feedback serves as the observation signal.

Experiments show that multi-turn RL with UFO encourages interactive reasoning and exploration of alternative solutions, reducing repeated answers. Models trained with UFO improve multi-turn success rates by 14% over single-turn RL and generalize well to diverse out-of-domain tasks by better adapting their reasoning.

To better align with real-world multi-turn reasoning, we propose two principles: **minimality**—solving problems in fewer turns, and **diversity**—exploring varied strategies after failure. We implement these via turn-wise reward decay and repetition penalties, promoting efficient and systematic reasoning.

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.

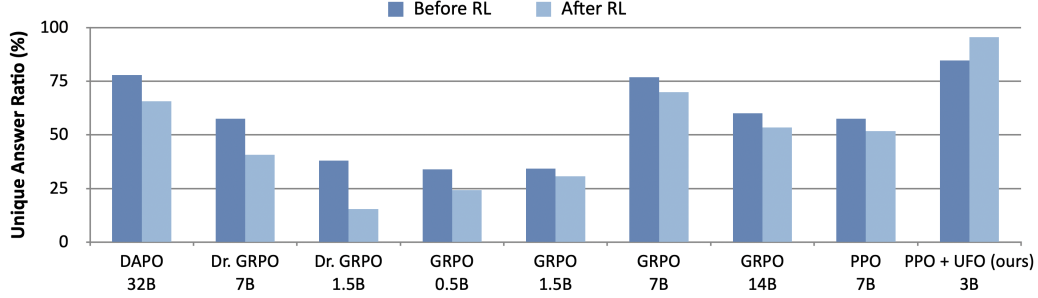


Figure 3: **Comparison of unique answer ratio (%) before and after RL training.** Across single-turn RL methods, the unique answer ratio consistently drops after training across multiple model scales.

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 LLMs by maximizing 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} , is the LLM policy parameterized by θ , and $R(x, y)$ is the reward for response y . Algorithms such as PPO [5, 15] and GRPO [1, 16] apply this objective to static datasets, yielding strong single-turn gains in math and code generation.

Multi-Turn Extensions. While existing methods optimize for single-shot correctness, real-world tasks like tutoring, coding, and embodied agents require *multi-turn* and iterative refinement under feedback. In some domains (e.g., programming), automated feedback is available, but in others (e.g., math), feedback is often sparse.

Recent work addresses the challenge of sparse feedback by optimizing full interaction trajectories. For example, CollabLLM [17] introduces multi-turn fine-tuning with collaborative simulation and sparse rewards, while RAGEN [7] frames reasoning as an MDP to enable delayed credit assignment. However, these methods often depend on custom environments or simulated rewards, limiting scalability. Since most real-world math and code datasets are single-turn and collecting turn-level human feedback is costly, some efforts synthesize proxy signals [9, 8] or build tool-augmented environments [18, 19, 12]. Still, multi-turn training remains insufficiently studied.

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

To understand multi-turn behavior, we examine how single-turn RL models respond to minimal feedback (e.g., “try again”). While these models are **effective solvers**, they are **poor revisers**, often repeating initial answers without incorporating feedback (Figure 2).

We quantify this using the *effective answer* metric. As shown in Figures 3, single-turn RL training reduces the diversity of responses across turns. Off-the-shelf models fine-tuned with PPO [5], GRPO [16], DAPO [20], and Dr. GRPO [21] all exhibit decreased effective answer ratios. For instance, DAPO reduces the 32B model’s ratio from 78.0% to 65.7%, and Dr. GRPO drops the 1.5B model from 38.0% to 15.4%. PPO and GRPO show milder declines.

We theoretically analyze why single-turn RL models repeat mistakes in multi-turn settings. This behavior is a predictable consequence of the training process itself, which creates a **peaked, low-entropy output distribution** [22, 23].

To quantify this, we use the *collision probability*, the likelihood that two independent samples are identical, which is formally defined as

$$\text{Coll}(q) := \sum_y q(y | x)^2.$$

This probability is fundamentally lower-bounded by the distribution’s Shannon Entropy,

$$\text{Coll}(q) \geq \exp(-\mathcal{H}(q)).$$

We made a rigorous formalization in Appendix B, showing that as a policy becomes more deterministic (lower entropy), repetition probability increases. Thus, for static policies that can’t incorporate feedback, repetition becomes an inevitable failure mode.

To overcome this limitation, a policy must learn from interaction history. We formalize this via a Markov Decision Process (MDP) and define two policy classes. **Parallel Policies** (Π_{par}), used in single-turn RL, generate k independent answers from the initial input; **Sequential Policies** (Π_{seq}) adapt based on the full interaction history. Since

$$\Pi_{\text{par}} \subseteq \Pi_{\text{seq}},$$

the latter provably achieves higher expected success rates:

$$\max_{\pi \in \Pi_{\text{seq}}} \mathbb{E}[\text{Succ}@k] \geq \max_{\pi \in \Pi_{\text{par}}} \mathbb{E}[\text{Succ}@k].$$

This guarantee follows from a sequential policy’s ability to avoid past errors, boosting its conditional success. By Blackwell dominance [24], it achieves success rates at least as high as parallel policies. See Appendix C for details.

This highlights a critical gap: single-turn RL falls short for multi-turn reasoning, yet fine-grained supervision is often impractical. This leads to our central question:

Can minimal feedback like “try again” enable multi-turn reasoning on static datasets?

3 Training Multi-Turn Reasoning Models with Unary Feedback

3.1 Problem Formulation

We model multi-turn problem solving from static single-turn datasets as a finite-horizon MDP:

$$(S, \mathcal{A}, \mathcal{P}, R, T_{\text{max}}).$$

Here, 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 $s_t \in S$, that encodes the original question q and the history of past attempts and feedbacks:

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

where a_k denotes the k -th answer, and f_k is a feedback token returned by the environment. The agent samples an answer

$$a_t \sim \pi_{\theta}(\cdot \mid s_t),$$

and receives reward

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

The episode ends upon success or after T_{max} turns. This frames multi-turn learning within standard RL.

3.2 Unary Feedback as Observation (UFO)

To realize the above MDP on static datasets, we introduce **Unary Feedback as Observation (UFO)** (Figure 4), a simple mechanism for constructing state histories. The key idea is to restrict f_k in the observation to negative signals only. UFO restricts feedback f_k to a generic negative signal (e.g., TryAgain) when answers are incorrect; no explicit positive feedback is given. The episode ends upon a correct answer, and the agent must learn to revise based solely on failed attempts.

In practice, prompts are natural-language sequences listing prior answers and feedback, e.g.:

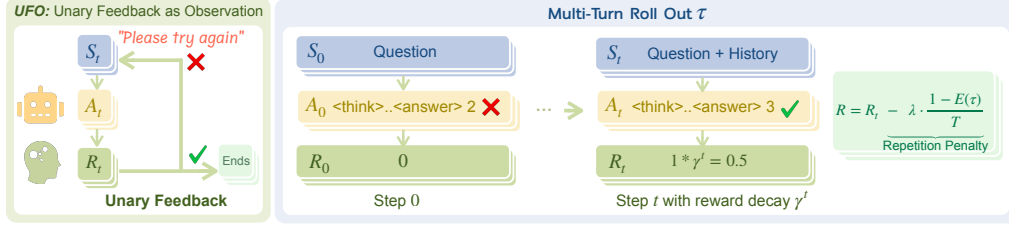


Figure 4: **The UFO framework for multi-turn training.** At each step t , the model observes the full interaction history and generates a response. Correct responses receive discounted rewards γ^t , while incorrect ones receive none. A repetition penalty based on the uniqueness of trajectory τ is applied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

```

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

```

UFO converts static single-turn data into multi-turn episodes with no need for expert labels, structural changes, or external tools—enabling multi-turn RL with minimal supervision.

3.3 Reinforcement Learning with Unary Feedback

With the MDP and UFO design, we train the agent via RL to learn revision-aware, multi-turn policies. Since datasets only provide final-answer correctness without reasoning traces, supervised finetuning is infeasible. RL instead enables exploration under sparse, delayed supervision.

We adopt Proximal Policy Optimization (PPO) to train the policy π_θ , following prior work [25, 7] showing that a learned critic stabilizes optimization. In each episode, the agent interacts with a problem over multiple turns, observing x_t , generating a_t , and receiving binary reward $r_t \in \{0, 1\}$. The resulting trajectory is defined as

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

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

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

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:

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

UFO enables the policy to condition on past failures, allowing context-sensitive behaviors like error correction, elimination, and refinement—skills hard to learn from static supervision alone.

3.4 Reward Design for Adaptive Reasoning

Binary correctness offers minimal supervision but may lead to blind trial-and-error behavior. To encourage efficiency and reasoning, we introduce *reward decay* for **minimality**, favoring shorter trajectories, and *repetition penalty* for **diversity**, discouraging repeated answers and promoting exploration.

Reward decay encourages early success by assigning exponentially lower rewards to correct answers at later turns:

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

where $\gamma \in (0, 1)$ is a decay factor.

We define the repetition penalty using the number of *effective answers* $E(\tau)$ in a trajectory τ of length T :

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

where $\lambda > 0$ controls the penalty strength and $E(\tau)/T$ reflects diversity.

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

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

To improve stability, we apply a small penalty $\eta < 0$ for malformed or missing outputs. Overall, the reward combines final-turn correctness, answer diversity, and output validity across turns.

4 Experiments

4.1 Setup

Dataset We conduct major experiments on the MATH subset of MetaMathQA [26] dataset (MMQ-Math), where data are augmented from the MATH [27] dataset. We also select eight other widely-used datasets in four different domains to evaluate the generalization ability of UFO: TheoremQA[28], GSM8K[29], GPQA[30], MMLU-STEM[31], HotPotQA[32], ConcurrentQA[33], MMLU [31] and MMLU-Pro[34].

Training Settings We train Qwen-2.5-3B-Instruct using PPO for 200 steps on A100 GPUs. Each batch samples $P=8$ prompts with $N=16$ rollouts per prompt. We vary the maximum episode length $T_{\max} \in 1, 5, 10$ during training and fix $T_{\max}=5$ for validation. PPO uses GAE $(\gamma, \lambda) = (1.0, 1.0)$, Adam with $\beta = (0.9, 0.999)$, and entropy coefficient 10^{-3} .

Baseline and Metrics We compare our method **UFO** against a single-turn PPO model trained with parallel sampling. The baseline uses Pass@k, selecting the best from k parallel responses. Our multi-turn model generates responses sequentially with feedback and is evaluated using Succ@k and AvgTurns. We also perform ablations on $T_{\max}$ to study the effect of multi-turn training.

- **Pass@k (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)** This metric measures the percentage of problems solved within a fixed number of turns. Let τ_j be the number of turns the agent takes to solve problem q_j , or ∞ if it fails. $\text{Succ@k} = \frac{1}{N} \sum_{j=1}^N \mathbb{1}[\tau_j \leq k]$.
- **Average Number of Turns (Multi-turn)** 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 .

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:

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 multi-turn RL models to single-turn PPO baselines using Succ@5 over 21 checkpoints on a held-out set (Figure 5). Multi-turn training achieves up to **14% higher success** with similar inference cost, highlighting the value of iterative revision under sparse feedback.

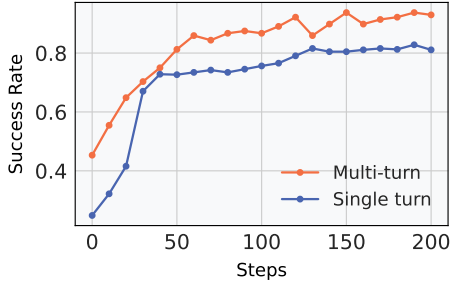


Figure 5: 5-turn UFO significantly outperforms single-turn RL baseline with similar inference cost.

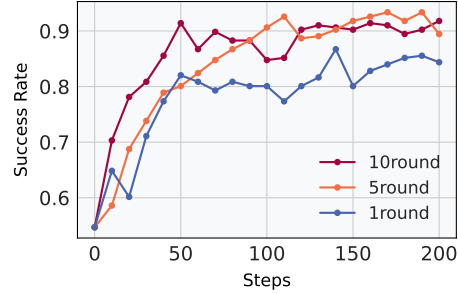


Figure 6: Performance with 5 evaluation turns shows that training with 5 turns performs best, while increasing to 10 offers no clear benefit.

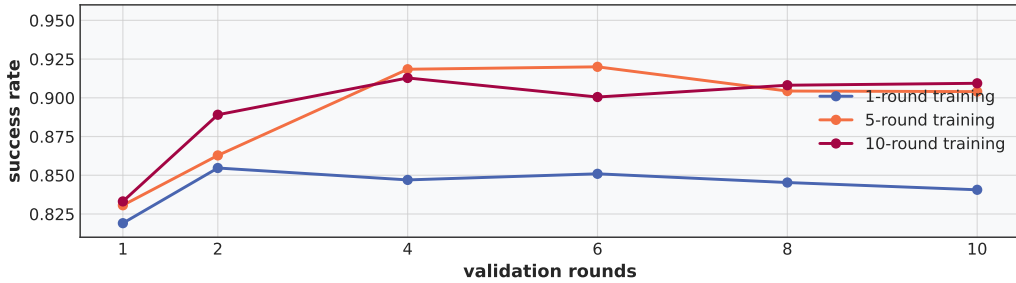


Figure 7: **Validation performance (Succ@k) under different training and inference turn settings.** Multi-turn UFO consistently improves success rates across all k , including $k = 1$.

We further evaluate models trained with $T_{\max} \in \{1, 5, 10\}$ using a fixed 5-turn validation. As shown in Figure 6, both 5- and 10-turn training yield **over 6% gains** relative to single-turn training, confirming the benefits of longer training horizons.

To test generalization, we assess models under various inference budgets $k \in \{1, 2, 4, 6, 8, 10\}$ (Figure 7). Multi-turn-trained models consistently outperform baselines across all k , including $k=1$, indicating improved single-turn performance from multi-turn training.

We evaluate the transferability of multi-turn reasoning using 5-turn UFO across nine benchmarks, covering math (MMQ-Math, TheoremQA, GSM8K), STEM (GPQA, MMLU-STEM), QA (HotPotQA, ConcurrentQA), and general knowledge (MMLU, MMLU-Pro), as shown in Table 1. **Applying 5-turn UFO consistently improves performance over task-specific RL across all domains.**

UFO enhances both in-domain and out-of-domain reasoning by promoting **exploration** and **self-reflection**. For example, on MMQ-Math, 5-turn UFO improves the 3B model from 52.3% to 88.5%. Gains generalize across tasks: models trained on MMQ-Math show 7–10% improvement on STEM, QA, and general knowledge datasets. Similarly, training on HotPotQA yields boosts on GPQA (+9.4%), MMLU-STEM (+7%), and MMLU (+5.1%). Notably, UFO recovers and surpasses accuracy drops from RL on MMLU (66.8% $\rightarrow$ 85.2%) and improves MMLU-Pro from 48.3% to 60.9%. These results show that UFO enables robust multi-turn and cross-task generalization by guiding models to revise failed attempts using minimal feedback.

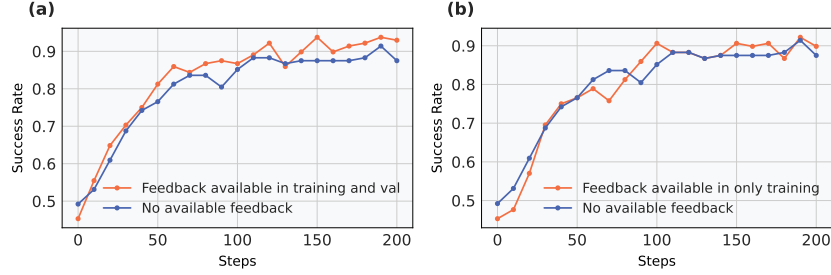
4.2.2 Multi-turn Setting Enables LRMs to Revise From Feedback

The multi-turn setting enables agents to engage repeatedly with each prompt, thereby constructing more informative interaction trajectories from the same training data. This enhanced utilization of feedback is hypothesized to extract more meaningful signals, potentially improving solution quality and accelerating convergence, especially in data-limited contexts.

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

Table 1: 5-turn success rate across different tasks and training settings.

Model	Math			STEM		QA		General	
	MMQ-Math	TheoremQA	GSM8k	GPQA	MMLU-STEM	HotpotQA	ConcurrentQA	MMLU	MMLU-Pro
Qwen2.5-1.5B-Instruct									
Base Model w/o RL	10.9	11.7	26.6	21.9	62.5	2.4	3.1	52.3	35.2
RL on MMQ-Math	74.8	20.1	84.7	22.7	65.5	19.2	9.5	43.8	34.8
+5turn UFO	83.6	26.8	88.1	27.3	64.8	22.6	9.5	60.9	34.8
Qwen2.5-3B-Instruct									
Base Model w/o RL	52.3	28.3	68.0	51.6	75.8	7.8	3.9	75.2	42.2
RL on MMQ-Math	79.7	32.0	93.0	50.1	77.6	19.5	12.9	66.8	48.3
+5turn UFO	88.5	40.8	95.3	52.3	87.5	26.6	15.2	85.2	60.9
RL on HotQA	72.4	31.8	89.1	48.4	81.3	38.3	16.8	71.5	49.3
+5turn UFO	72.7	29.2	85.0	57.8	88.3	44.2	16.8	76.6	48.9
Qwen2.5-7B-Instruct									
Base Model w/o RL	56.4	32.1	56.3	62.5	83.6	13.3	4.7	72.3	64.1
RL on MMQ-Math	85.1	33.6	95.2	50.8	84.8	26.3	14.1	73.4	52.3
+5turn UFO	93.0	42.1	96.8	56.9	84.8	28.6	16.4	80.5	58.8
Llama3.2-1B-Instruct									
Base Model w/o RL*	2.3	2.3	1.6	1.6	4.6	0.8	0.8	3.9	2.3
RL on MMQ-Math	53.9	21.1	52.3	20.3	57.0	19.5	0.8	57.8	32.8
+5turn UFO	64.8	26.8	56.3	26.6	60.2	21.1	1.6	66.4	32.8
Llama3.2-3B-Instruct									
Base Model w/o RL	50.8	20.3	48.4	47.7	77.3	29.7	6.0	65.6	49.2
RL on MMQ-Math	86.7	24.2	92.2	46.9	78.1	44.5	13.3	71.1	60.9
+5turn UFO	92.2	32.0	93.8	50.8	82.0	39.8	14.8	82.8	66.4

Figure 8: **Multi-turn success rate comparison:** Feedback prompt used in both training and validation vs. blank prompt; (b) Feedback prompt used only in training vs. blank prompt.

when explicit feedback is provided. Even when used only during training (Figure 8(b)), feedback improves reasoning, suggesting intrinsic benefits of multi-turn learning.

Robustness analysis (Figure 9) further shows consistent gains across prompt variations, highlighting the generality and applicability of our approach.

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 suggests that the exponential decay schedule encourages the model to engage in more profound self-reflection and systematic thinking before generating a response.

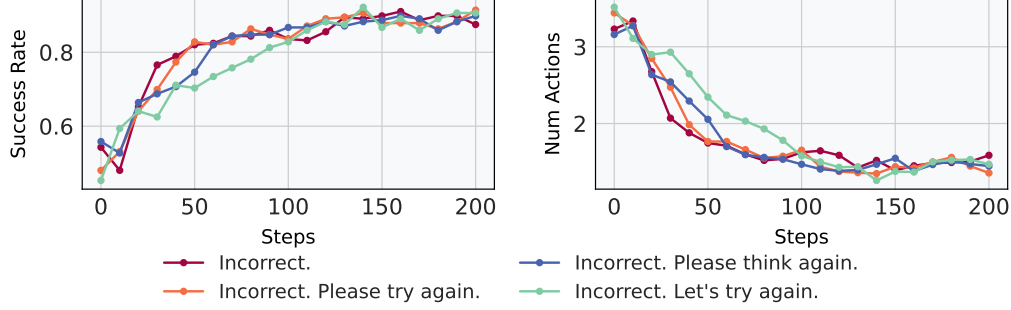


Figure 9: **Validation under different verbal feedback prompts.** Success rates and action counts remain consistent across all variants, demonstrating UFO’s robustness to various prompts.

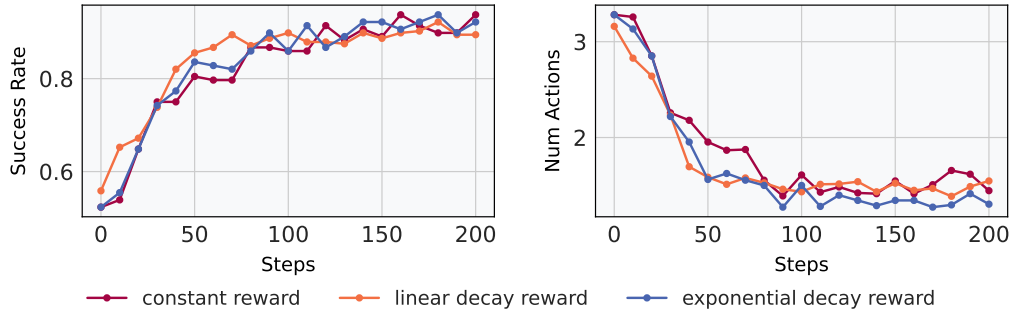


Figure 10: **Comparison of reward shaping strategies.** While constant, linear decay, and exponential decay schedules achieve similar success rates (left), exponential decay consistently leads to fewer actions per episode (right), indicating more efficient problem solving with less external supervision.

5 Related Work

Enhancing LLM Reasoning with Test-Time Search and Parameter-Efficient Training. **Test-time reasoning frameworks** keep model weights frozen yet boost performance: Graph-of-Thought [35], Reflexion [36], Monte Carlo Tree Self-Refine [37], Self-Refine [38], CRITIC [39] and memory-augmented agents such as POEM [40] and Larimar [41] rely on search, self-feedback or episodic memory *without* updating model parameters.

Training-time optimisation methods, in contrast, adjust the policy itself. RLHF [42, 15] and its low-cost variant RLAIIF [43] align models to preference data; scalable-oversight *debate* protocols explore alignment with weak judges [44]. Lightweight objectives such as Direct Preference Optimisation (DPO) [45], Parameter-Efficient RLHF (PERLHF) [46] and Self-Play Fine-Tuning (SPIN) [47] further cut roll-out cost, while hierarchical ArCHer [6] tackles long-horizon credit assignment. Benchmarks like UNO Arena [48] expose the strengths and weaknesses of both families in stateful, multi-turn settings.

Multiturn training for LLMs. Multiturn training for LLMs has been explored through diverse benchmarks, optimization methods, and architectural designs. Evaluation suites like LMRL-Gym [49] and MT-Eval [50] assess consistency, instruction-following, and planning across dialogue turns. On the training side, multiturn RLHF methods include regression-based value estimation [51], hierarchical actor-critic [6], and trajectory-level preference modeling [52]. Extensions integrate execution feedback [53], optimize full-dialogue rewards [54], and leverage multiturn-aware frameworks like CollabLLM [17] and RAGEN [7]. Beyond RL, parameter-efficient methods such as Baize [55] achieve strong multiturn performance via LoRA and self-chat. A recent survey [56] offers a comprehensive taxonomy of these strategies, covering optimization, memory, and evaluation.

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. By explicitly incorporating prior interaction history, UFO enables models to engage in exploration and self-reflection across multiple attempts, leading to deeper reasoning and improved adaptability. 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 diverse reasoning, self-correction and more thoughtful response patterns. Our approach is lightweight, generalizable, and easily applicable to existing datasets. A limitation of our work is its primary focus on relatively small models, leaving its generalizability to larger scales for future investigation.

References

- [1] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.
- [2] OpenAI. Gpt-4 technical report, 2024.
- [3] 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.
- [4] Gemini Team. Gemini: A family of highly capable multimodal models, 2025.
- [5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.
- [6] Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn rl, 2024.
- [7] 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.
- [8] 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. Osvorld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024.
- [9] 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.
- [10] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents, 2023.
- [11] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning, 2021.
- [12] 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, 2024.
- [13] 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.

- [14] 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.
- [15] 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.
- [16] 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.
- [17] Shirley Wu, Michel Galley, Baolin Peng, Hao Cheng, Gavin Li, Yao Dou, Weixin Cai, James Zou, Jure Leskovec, and Jianfeng Gao. Collabllm: From passive responders to active collaborators. In *International Conference on Machine Learning (ICML)*, 2025. Outstanding Paper Award (oral).
- [18] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Serkan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning, 2025.
- [19] 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.
- [20] 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.
- [21] 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.
- [22] Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language models, 2025.
- [23] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?, 2025.
- [24] David Blackwell. Comparison of experiments. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 93–102. University of California Press, 1951.
- [25] 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.
- [26] 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.
- [27] 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.

- [28] Wenhui 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, 2023.
- [29] 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. *arXiv preprint arXiv:2110.14168*, 2021.
- [30] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof qa benchmark. *arXiv preprint arXiv:2311.12022*, 2023. Accessed: 2025-08.
- [31] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2020.
- [32] 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.
- [33] Simran Arora, Patrick Lewis, Angela Fan, Jacob Kahn, and Christopher Ré. Reasoning over public and private data in retrieval-based systems, 2022.
- [34] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhramil 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, 2024.
- [35] 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.
- [36] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023.
- [37] 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.
- [38] 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.
- [39] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujia Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-augmented critiquing, 2023.
- [40] Dai Do, Quan Tran, Svetha Venkatesh, and Hung Le. Large language models prompting with episodic memory, 2024.
- [41] Payel Das, Subhjit 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.
- [42] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2017.
- [43] 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.
- [44] 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.
- [45] 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.

- [46] 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.
- [47] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models, 2024.
- [48] 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.
- [49] Marwa Abdulhai, Isadora White, Charlie Snell, Charles Sun, Joey Hong, Yuexiang Zhai, Kelvin Xu, and Sergey Levine. LMRL Gym: Benchmarks for multi-turn reinforcement learning with language models. *arXiv preprint arXiv:2311.18232*, 2023.
- [50] Wai-Chung Kwan, Xingshan Zeng, Yuxin Jiang, et al. MT-Eval: A multi-turn capabilities evaluation benchmark for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024.
- [51] Zhaolin Gao, Wenhao Zhan, Jonathan D. Chang, Gokul Swamy, Kianté Brantley, Jason D. Lee, and Wen Sun. Regressing the relative future: Efficient policy optimization for multi-turn rlhf. In *International Conference on Learning Representations*, 2025.
- [52] Wentao Shi, Mengqi Yuan, Junkang Wu, Qifan Wang, and Fuli Feng. Direct multi-turn preference optimization for language agents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.
- [53] Wei Xiong, Chengshuai Shi, Jiaming Shen, Aviv Rosenberg, Zhen Qin, Daniele Calandriello, Misha Khalman, Rishabh Joshi, Bilal Piot, Mohammad Saleh, Chi Jin, Tong Zhang, and Tianqi Liu. Building math agents with multi-turn iterative preference learning, 2025.
- [54] Lior Shani, Aviv Rosenberg, Asaf Cassel, Oran Lang, Daniele Calandriello, Avital Zipori, Hila Noga, Orgad Keller, Bilal Piot, Idan Szpektor, Avinatan Hassidim, Yossi Matias, and Rémi Munos. Multi-turn reinforcement learning from preference human feedback. In *Advances in Neural Information Processing Systems*, 2024.
- [55] Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.
- [56] Chen Zhang, Xinyi Dai, Yaxiong Wu, Qu Yang, Yasheng Wang, Ruiming Tang, and Yong Liu. A survey on multi-turn interaction capabilities of large language models. *arXiv preprint arXiv:2501.09959*, 2025.
- [57] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2015.

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:

$$J(\theta) = \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(\cdot|x)} [R(x, y)], \quad (2)$$

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) [5], which stabilizes training by clipping the likelihood ratio between the new and old policies. The ratio is defined as:

$$\rho_t(\theta) = \frac{\pi_\theta(y_t | x_t)}{\pi_{\theta_{\text{old}}}(y_t | x_t)}. \quad (3)$$

The PPO objective minimizes over the clipped surrogate advantage:

$$J_{\text{PPO}}(\theta) = \mathbb{E}_t [\min(\rho_t A_t, \hat{\rho}_t A_t) - \beta D_{\text{KL}}], \quad (4)$$

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) [57] is often used:

$$A_t^{\text{GAE}(\gamma, \lambda)} = \sum_{l=0}^{\infty} (\gamma \lambda)^l \delta_{t+l}, \quad \text{with} \quad \delta_t = r_t + \gamma V(x_{t+1}) - V(x_t), \quad (5)$$

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

More recently, DeepSeekMath [16] and DeepSeek-R1 [1] 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:

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{x, \{y_i\}} [J_{\text{group}}(\theta)], \quad (6)$$

with

$$J_{\text{group}}(\theta) = \frac{1}{G} \sum_{i=1}^G \min(\rho_i A_i, \hat{\rho}_i A_i) - \beta D_{\text{KL}}, \quad (7)$$

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

$$A_i = \frac{r_i - \text{mean}(\{r_j\})}{\text{std}(\{r_j\})}. \quad (8)$$

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 Details on the Theoretical Analysis on Repetition Pattern

We provide a detailed proof on how peaked, low-entropy output distribution from RL training [22, 23] can lead to high repetition in model multi-turn behavior.

Preliminaries. Let $q(y | x)$ denote the model’s output distribution given input x . We introduce the following definitions:

Definition 1 (Collision Probability). The collision probability of $q(y | x)$ is defined as:

$$\text{Coll}(q) := \sum_y q(y | x)^2. \quad (9)$$

This is the probability that two i.i.d. samples from q yield the same answer: $\Pr[A_i = A_j] = \text{Coll}(q)$.

Definition 2 (Entropy). The Shannon entropy of $q(y \mid x)$ is:

$$\mathcal{H}(q) := - \sum_y q(y \mid x) \log q(y \mid x).$$

Lower entropy corresponds to a more peaked distribution.

Definition 3 (Expected Number of Duplicate Pairs). Given k i.i.d. samples $\{A_1, \dots, A_k\} \sim q(\cdot \mid x)$, the expected number of duplicate pairs is:

$$\mathbb{E}[\text{DupPairs}] = \binom{k}{2} \cdot \text{Coll}(q).$$

Repetition Under Sequential Sampling.

Proposition 1. *Let $A_1, \dots, A_k \sim q(\cdot \mid x)$ be sampled sequentially. Since single-turn RL does not guarantee any multi-turn capability, we simplify the assumption that the policy is static and does not update based on prior turns, i.e., the answer at each turn t is sampled from the same fixed distribution $q(\cdot \mid x)$. The probability of generating a duplicate answer is the collision probability, which is lower-bounded by:*

$$\Pr[A_i = A_j] \geq \exp(-\mathcal{H}(q)),$$

where $\mathcal{H}(q)$ denotes the Shannon entropy of the base distribution $q(\cdot \mid x)$.

Proof. Let $Y \sim q(\cdot \mid x)$, and define the collision probability as:

$$\text{Coll}(q) = \mathbb{P}[A_i = A_j] = \sum_y q(y)^2 = \mathbb{E}_{Y \sim q}[q(Y)].$$

By Jensen’s inequality applied to the concave function $\log$, we have:

$$\log \mathbb{E}_Y[q(Y)] \geq \mathbb{E}_Y[\log q(Y)] = -\mathcal{H}(q),$$

which implies:

$$\text{Coll}(q) \geq \exp(-\mathcal{H}(q)).$$

□

Remark 1 (On the Tightness of the Bound). The lower bound $\text{Coll}(q) \geq \exp(-\mathcal{H}(q))$ is tight in the following cases:

- When q is uniform on a support of size n , i.e., $q(y) = 1/n$, then $\mathcal{H}(q) = \log n$ and $\text{Coll}(q) = 1/n$, achieving equality.
- When q is a delta distribution (i.e., concentrated on one point), then $\mathcal{H}(q) = 0$ and $\text{Coll}(q) = 1$.

We assume $\log q(y)$ is only computed where $q(y) > 0$, so the result still holds for distributions with zero-probability points.

A tighter bound can be obtained using the Rényi–Shannon inequality:

$$\text{Coll}(q) = e^{-\mathcal{H}(q) - D_{\text{KL}}(q \parallel u)},$$

where u is the uniform distribution on the support of q . Our stated bound omits the KL divergence for simplicity and interpretability.

This demonstrates that a low-entropy model is mathematically guaranteed to have a higher floor for its repetition rate. Thus, for a static agent that does not learn from feedback, repetition is not an accidental bug but a predictable outcome of the low-entropy distributions created by standard RL.

C Theoretical Analysis on Advantages of Sequential Policies

We model multi-turn reasoning as a finite-horizon Markov Decision Process (MDP), where the state at step t is given by

$$s_t = (q, a_1, f_1, \dots, a_{t-1}, f_{t-1}),$$

with input question q , previous answers a_i , and corresponding feedback f_i .

We represent policies as functions $\pi_t(a \mid s_t)$ that produce an action a based on the interaction history s_t . In our setting, the *single-turn RL* paradigm corresponds to a **parallel policy**, where the model samples multiple answers independently from a fixed distribution $\pi(a \mid q)$ without conditioning on feedback. In contrast, our *UFO* (Unary Feedback as Observation) operates as a **sequential policy**, where the action at each step is conditioned on the full history and thus can adapt dynamically.

We formalize the distinction as follows:

- **Parallel policies** $\pi \in \Pi_{\text{par}}$: sample answers i.i.d. from a fixed policy $\pi(a \mid q)$, without using feedback.
- **Sequential policies** $\pi \in \Pi_{\text{seq}}$: choose actions based on the full state s_t , enabling feedback-driven refinement.

Every parallel policy is a special case of a sequential policy that ignores interaction history. Formally, for any $\pi(a \mid q)$, we can construct $\pi_t(a \mid s_t) = \pi(a \mid q)$ for all t , implying

$$\Pi_{\text{par}} \subseteq \Pi_{\text{seq}}.$$

Since $\Pi_{\text{par}} \subseteq \Pi_{\text{seq}}$, we immediately have:

$$\max_{\pi \in \Pi_{\text{seq}}} \mathbb{E}[\text{Succ}@k] \geq \max_{\pi \in \Pi_{\text{par}}} \mathbb{E}[\text{Succ}@k],$$

where $\text{Succ}@k$ denotes the probability of producing a correct answer within k attempts.

Let p denote the success probability under a parallel policy. Then:

$$\mathbb{P}_{\text{par}}[\text{success in } k \text{ turns}] = 1 - (1 - p)^k.$$

For a sequential policy, let p'_t be the conditional success probability at step t , which may depend on the state s_t . The success probability is:

$$\mathbb{P}_{\text{seq}}[\text{success in } k \text{ turns}] = 1 - \prod_{t=1}^k (1 - p'_t), \quad \text{where } p'_t \geq p.$$

Sequential policies can eliminate previously failed answers by maintaining a rejection set $\mathcal{H}_t \subset \mathcal{A}$ and enforcing:

$$\pi_t(a \mid s_t) = 0 \quad \text{for all } a \in \mathcal{H}_t.$$

This behavior approximates sampling without replacement and yields increasing conditional success rates:

$$p'_t = \frac{p}{1 - \sum_{i=1}^{t-1} p_i} > p.$$

Therefore, sequential policies such as UFO improve $\text{Succ}@k$ by adaptively avoiding prior failure modes. This formalizes the theoretical advantage of feedback-aware reasoning strategies over static single-turn RL.

D Detailed Evaluation under Multi-round Settings

We illustrate a detailed analysis of how multi-round training improves generalization on long-horizon interactive reasoning. Figure 11 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

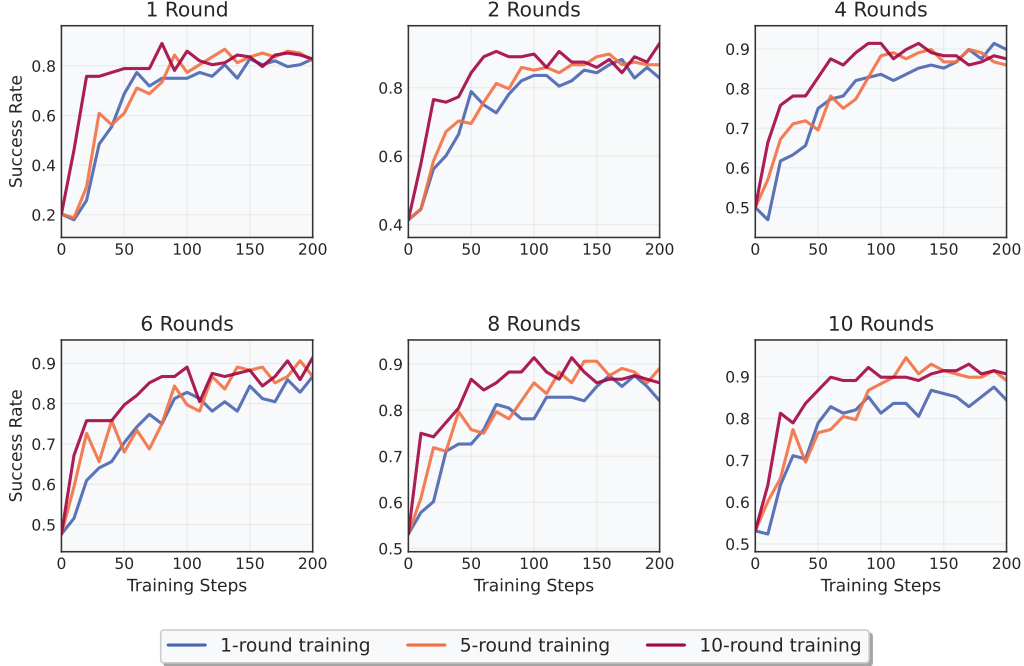


Figure 11: **Performance across different evaluation round settings.** Each subplot shows the success rate evaluated at r rounds. While all methods perform similarly under 1-round evaluation, models trained with multi-round feedback (UFO) generalize significantly better to longer evaluation horizons.

single-round interactions struggle to adapt to multi-turn dynamics, leading to degraded performance as the task horizon increases.

E Prompt Settings

E.1 Problem Solving Model Prompt

We adopt a simple and structured prompt format for mathematical problem solving, following prior designs from [16, 3], 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 , we present the prompt template used during training and evaluation.

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|>

```

E.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|>

```

E.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 , 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
...
<|im_end|>

```

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

F Model Evaluation Details

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

G 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 –) highlight the evolving dynamics of exploration, convergence, and reasoning quality throughout training.

G.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). 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 suggest that reinforcement learning has the potential to further stabilize and guide emergent reasoning process toward convergence.

G.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). 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

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

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.

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

This example illustrates the effectiveness of multi-turn reinforcement learning (Box). 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.

G.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 , 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 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, ...

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, ...

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 $X = 2$.</think>

<answer>2</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>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 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 $X = 2$.</think>

<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.</think>

<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,

$7X + 3 = 17$.</think>

<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