RAGEN: Understanding Self-Evolution in LLM Agents via Multi-Turn Reinforcement Learning

Zihan Wang*[1], Kangrui Wang*[1], Qineng Wang*[1], Pingyue Zhang*[1], Linjie Li*[2], Zhengyuan Yang[4], Kefan Yu[1], Minh Nhat Nguyen[6], Licheng Liu[7], Eli Gottlieb[1], Monica Lam[3], Yiping Lu[1], Kyunghyun Cho[5], Jiajun Wu[3], Li Fei-Fei[3], Lijuan Wang[4], Yejin Choi[3], Manling Li[1]

[1]**Northwestern University** [2]**University of Washington** [3]**Stanford University** [4]**Microsoft**
[5]**New York University** [6]**Singapore Management University** [7] **Imperial College London**

Training large language models (LLMs) as interactive agents presents unique challenges including long-horizon decision making and interacting with stochastic environment feedback. While reinforcement learning (RL) has enabled progress in static tasks, multi-turn agent RL training remains underexplored. We propose **StarPO** (**S**tate-**T**hinking-**A**ctions-**R**eward **P**olicy **O**ptimization), a general framework for trajectory-level agent RL, and introduce **RAGEN**, a modular system for training and evaluating LLM agents. Our study on three stylized environments reveals three core findings. First, our agent RL training shows a recurring mode of **Echo Trap** where reward variance cliffs and gradient spikes; we address this with **StarPO-S**, a stabilized variant with trajectory filtering, critic incorporation, and decoupled clipping. Second, we find the shaping of RL rollouts would benefit from **diverse initial states, medium interaction granularity and more frequent sampling**. Third, we show that without **fine-grained, reasoning-aware reward signals**, agent reasoning hardly emerge through multi-turn RL and they may show shallow strategies or hallucinated thoughts.
LLM Agents, Multi-turn RL
Website: https://ragen-ai.github.io/
Code/Environments: https://github.com/RAGEN-AI/RAGEN.

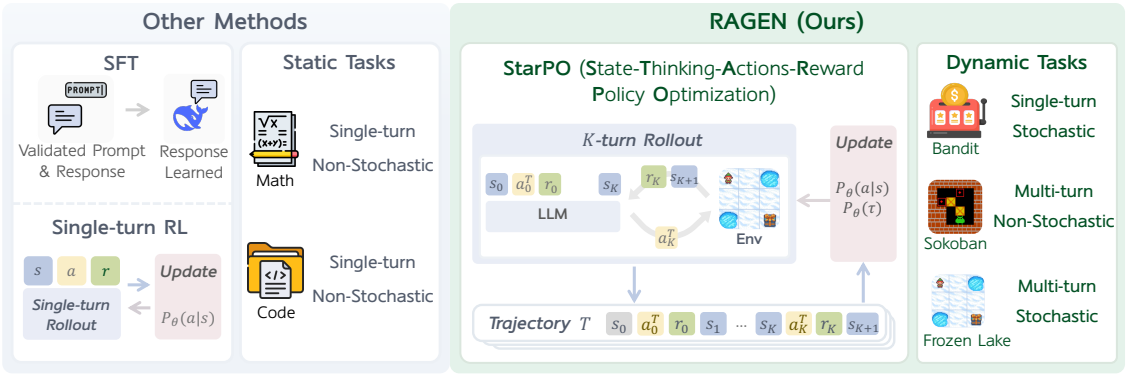


Figure 1 | Previous methods focus on non-interactive tasks such as math or code generation. **RAGEN** implements StarPO, a general agent RL framework that supports multi-turn rollouts, trajectory-level reward assignment, and policy updates, on agent tasks requiring multi-turn stochastic interaction.

1. Introduction

Training large language models (LLMs) to function as autonomous agents in interactive environments presents unique challenges. Unlike static tasks such as math problem solving (Shao et al., 2024) or coding (DeepSeek-AI et al., 2024), agent settings require models to make sequential decisions, maintain memory across turns, and adapt to stochastic feedback from their environment. These settings—central to planning assistants, robotics, and tutoring agents—demand that models not only perform well, but also self-improve through experience.

While recent work has explored reinforcement learning (RL) for LLMs (DeepSeek-AI et al., 2025; Gao et al., 2024; Kumar et al., 2024; OpenAI, 2024; Pan et al., 2025; Zeng et al., 2025) using rule-based reward, it remains largely underexplored to train agents that self-evolve to reason and adapt through rule-based RL. In particular, RL for LLM agents often exhibits training instability, complex reward signals, and limited generalization across prompt variations or environment changes—especially under multi-turn interaction with stochastic feedback. A key open question is: *what design factors make self-evolving LLM agents learn effectively and stably*?

We explore this question through a systematic study of agent learning under a general RL framework **StarPO** (**S**tate-**T**hinking-**A**ctions-**R**eward **P**olicy **O**ptimization). StarPO provides a unified view of **multi-turn, trajectory-level agent training** with flexible control over reasoning, reward assignment, and prompt-rollout structure. Built on top of StarPO, we develop **RAGEN**, a modular agent training and evaluation system designed to support the study of RL-based reasoning in LLMs. RAGEN implements the full training loop—including rollout generation, reward assignment, and trajectory optimization—serving as a research infrastructure for systematic analysis of LLM agent training dynamics under multi-turn and stochastic environments.

Training LLM agents on real-world tasks such as web browsing and embodied manipulation often relies on extensive pretrained priors and task-specific engineering. **To study learning from scratch and independent of these confounding factors**, we evaluate LLMs through RAGEN on three stylized **gaming** environments: **Bandit** (single-turn, stochastic), **Sokoban** (multi-turn, deterministic), and **Frozen Lake** (multi-turn, stochastic). These environments are deliberately minimalistic and fully controllable in difficulty, symbolic variation, and transition dynamics. Crucially, they require agents to learn decision-making policies through environment interaction, relying minimally on pre-existing world knowledge. The shared structure across these tasks (e.g., symbolic grid representations) further enables analysis of cross-task generalization.

Using this setup, we analyze three key dimensions of agent learning, and summarize below findings that **reveal core challenges and design principles** for stable agent RL training:

1. **Gradient Stability in Multi-turn RL is the Key to Stable Training.** We find that **multi-turn RL training** often leads to a recurring instability pattern, **Echo Trap**, where agents overfit to locally rewarded reasoning patterns, marked by reward variance collapse, entropy drop, and gradient spikes. To mitigate this failure mode, we propose **StarPO-S**, a stabilized variant of our framework that improves learning robustness through variance-based trajectory filtering, critic baselining, and decoupled clipping.
2. **Rollout Frequency and Diversity Shape Self-Evolution.** In RL-based agent training, LLM self-generated rollout trajectories are served as core training material. We identify key rollout factors for stable agent RL training: (1) ensuring that rollouts come from a **diverse prompt set** with **multiple responses per prompt**, (2) **implementing multiple actions each turn** to improve interaction horizon within fixed turn limit, (3) maintaining a **high rollout frequency** to ensure online feedback reflects current policies.
3. **Emerging *Agent* Reasoning Requires Meticulous Reward Signal.** We find that simply

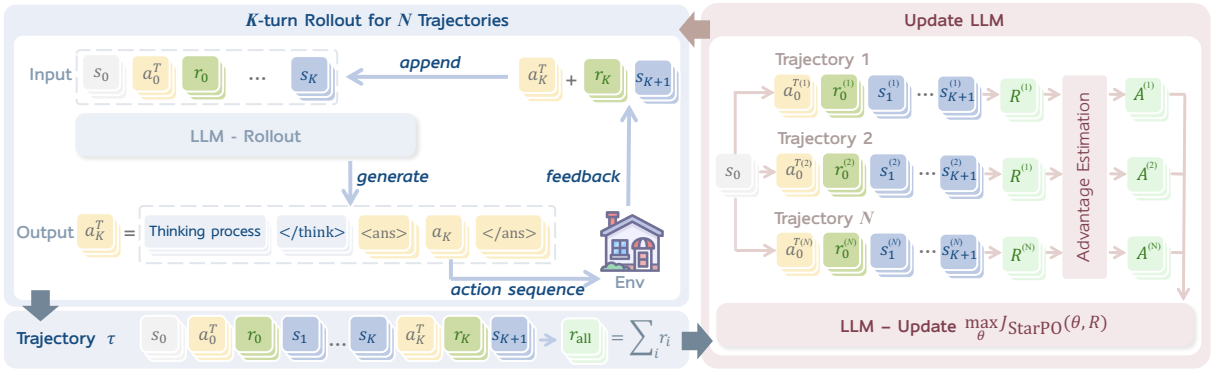


Figure 2 | The State-Thinking-Actions-Reward Policy Optimization (StarPO) framework. LLM generates reasoning-guided actions for multi-turn interactions with environments and accumulates trajectory-level rewards, normalized and used to update the LLM policy.

encouraging reasoning in the action format does not guarantee reasoning behavior. Even when models are prompted to reason (e.g., with '**<think>**' tokens) with trajectory-level optimization via StarPO, they often regress to direct action selection if reasoning offers no distinct reward advantage. We assume this is due to the simple action spaces in MDP where shallow strategies suffice. Moreover, when rewards only reflect task success, models produce **hallucinated reasoning**, revealing a mismatch between thoughts and environment states. These issues underscore the need for **fine-grained, reasoning-aware reward signals** in RL for long-horizon agent training.

Together, our framework and analysis offer insights into the principles behind training reasoning-capable, stable, and generalizable LLM agents. All environments and code are released as part of the RAGEN system.

2. Framework

2.1. The MDP Formulation for Agent Training

Previous reinforcement learning (RL) for language models often assumes a single-turn setting, where the goal is to maximize the expected reward $R(s, a)$ over prompt-response pairs (s, a) sampled from a dataset $\mathcal{D}$:

$$J_{\text{step}}(\theta) = \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi_\theta(\cdot|s)}[R(s, a)]. \tag{1}$$

However, LLM-based agents must operate in interactive environments that unfold over multiple turns and exhibit stochastic feedback. To capture these dynamics, we formulate the problem as a Markov Decision Process (MDP) $\mathcal{M} = \{S, A, P\}$, where S represents states (e.g., observation sequences or interaction histories), A represents actions (often token sequences), and P denotes the transition dynamics and reward generation process. The agent policy π_θ generates an action a_t at each time step t, conditioned on the current state s_t and the interaction history. The environment returns a reward r_t and a new state s_{t+1} given the current transition dynamics:

$$a_t \sim \pi_\theta(\cdot|s_t, \tau_{<t}), \quad (r_t, s_{t+1}) \sim P(\cdot|s_t, a_t),$$

where $\tau_{<t} = \{s_0, a_0, r_0, ..., s_{t-1}, a_{t-1}, r_{t-1}\}$ denotes the interaction history. This interactive process continues for a maximum horizon K, yielding a full trajectory $\tau = \{s_0, a_0, r_0, ..., s_K\}$ that forms the learning material for the agent.

2.2. StarPO: Reinforcing Reasoning via Trajectory-Level Optimization

Trajectory-Level Objective in StarPO vs. Previous Methods

Previous methods (e.g., PPO (Schulman et al., 2017), GRPO (Shao et al., 2024)):

$$J_{\text{step}}(\theta) = \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(\cdot|x)}\left[R(x,y)\right] \quad \text{(optimize single-turn output } y \text{ given input } x\text{)}$$

StarPO (ours):

$$J_{\text{StarPO}}(\theta) = \mathbb{E}_{\mathcal{M},\tau\sim\pi_\theta}\left[R(\tau)\right] \quad \text{(optimize total reward over trajectory } \tau = \{s_0, a_0, r_0 \ldots, s_K\}\text{)}$$

We introduce **StarPO** (State-Thinking-Action-Reward Policy Optimization), a general RL framework designed to optimize entire multi-turn interaction trajectories for LLM agents. Unlike previous methods for static tasks that treat each action independently, StarPO treats the **entire trajectory**—including observations, reasoning traces, actions, and feedback—as a coherent unit for rollout and model optimization. The objective is to maximize expected trajectory reward:

$$J_{\text{StarPO}}(\theta) = \mathbb{E}_{\mathcal{M},\tau\sim\pi_\theta}\left[R(\tau)\right], \tag{2}$$

where $\mathcal{M}$ is the MDP, τ is a full sequence of reasoning-augmented interactions, and $R(\tau)$ denotes the cumulative reward over the entire trajectory. The policy probability $\pi_\theta(\tau)$ is decomposed into token-level likelihoods, making StarPO directly compatible with autoregressive LLMs. Figure 2 illustrates the full StarPO process, and we break them down in detail below.

2.2.2. *Optimization Procedure: Reasoning-Interaction Trajectories*

At each training iteration, the agent begins from an initial state s_0 and generates N trajectories. At each step t, the agent produces a reasoning-guided structured output:

$$a_t^T = \texttt{<think>...</think><answer>}\ a_t\ \texttt{</answer>}, \tag{3}$$

where a_t^T is the full action output including intermediate reasoning, and a_t is the environment-executable sub-action. The environment then returns the next state s_{t+1} and reward r_t. The rollout stage produces complete trajectories $\tau = \{s_0, a_0^T, r_0, s_1, ..., a_{K-1}^T, r_{K-1}, s_K\}$, where *every component is LLM-generated or environment-induced* and will be jointly optimized.

StarPO interleaves rollout and update steps. New rollouts can be generated on-policy using π_θ, or sampled from a replay buffer under π_{old}. Each training loop consists of P initial states s_0, each generating N trajectories, and updates are performed with batch size E for L total loops. This yields $S = \frac{L\cdot P\cdot N}{E}$ total gradient steps. Additional training mechanisms are discussed in §3.

2.2.3. *Modular Optimization Strategies*

StarPO supports a variety of policy optimization algorithms under a unified trajectory-level abstraction. For each rollout trajectory $\tau_i = \{\tau_{i,(1)}, \ldots, \tau_{i,(T)}\}$ of totally $|\tau_i|$ tokens, we instantiate StarPO with the following optimization strategies for token-level updates:

- **PPO (Schulman et al., 2017).** We use the PPO objective (More Details can be found in Appendix A), where a critic is trained to estimate token-level value and advantages $A_{i,t}$:

$$J_{\text{PPO}}(\theta) = \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\tau_i|}\sum_{t=1}^{|\tau_i|}\min\left[\frac{\pi_\theta(\tau_{i,(t)}|\tau_{i,<t})}{\pi_{\text{old}}(\tau_{i,(t)}|\tau_{i,<t})}\cdot A_{i,t},\ \text{clip}\left(\frac{\pi_\theta(\tau_{i,(t)}|\tau_{i,<t})}{\pi_{\text{old}}(\tau_{i,(t)}|\tau_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\cdot A_{i,t}\right], \tag{4}$$

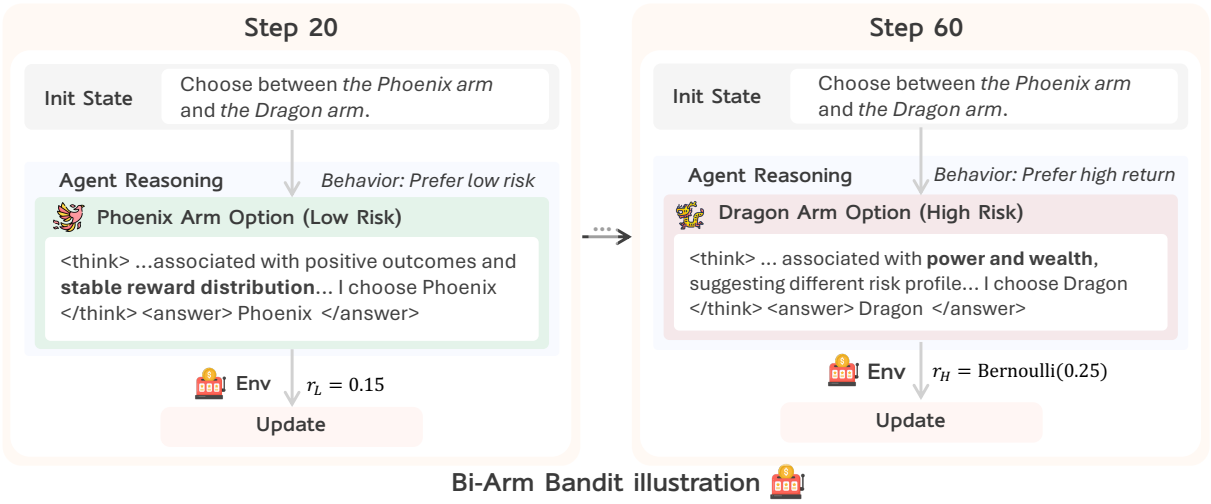


Figure 3 | **Bi-Arm Bandits environment.** The agent chooses between a low-risk arm (Phoenix) and a high-risk yet high-reward arm (Dragon), each linked to symbolic semantics. The agent learns to choose stable reward at early stages and reasons to pursue maximal expected reward and shift toward strategic risk-taking.

where G is the number of trajectories in the batch, $\tau_{i,(t)}$ denotes the t-th token in trajectory τ_i, and $\tau_{i,<t}$ is its prefix.

- **GRPO (Shao et al., 2024).** For critic-free training leveraging GRPO, we assign a scalar reward $R(\tau_i)$ to each trajectory and normalize it across the batch. The normalized reward $\hat{A}_{i,t}$ is shared across all tokens in τ_i:

$$\hat{A}_{i,t} = \frac{R(\tau_i) - \text{mean}(\{R(\tau_1), \ldots, R(\tau_G)\})}{\text{std}(\{R(\tau_1), \ldots, R(\tau_G)\})}. \tag{5}$$

The GRPO objective becomes:

$$J_{\text{GRPO}}(\theta) = \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\tau_i|}\sum_{t=1}^{|\tau_i|}\min\left[\frac{\pi_\theta(\tau_{i,(t)}|\tau_{i,<t})}{\pi_{\text{old}}(\tau_{i,(t)}|\tau_{i,<t})}\cdot\hat{A}_{i,t},\ \text{clip}\left(\frac{\pi_\theta(\tau_{i,(t)}|\tau_{i,<t})}{\pi_{\text{old}}(\tau_{i,(t)}|\tau_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\cdot\hat{A}_{i,t}\right]. \tag{6}$$

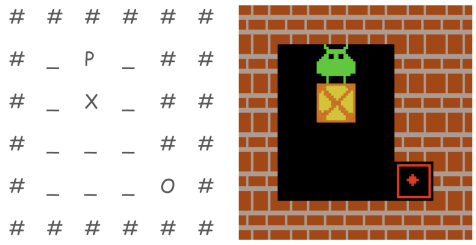
2.3. The RAGEN System

To implement StarPO in practice, we build **RAGEN**, a complete system for LLM agent training in controlled environments. RAGEN supports structured rollouts, customizable reward functions, and integration with multi-turn, stochastic environments. It serves both as the execution backend for StarPO and as a platform for studying stability, generalization, and learning dynamics in training reasoning agents. RAGEN is designed to be modular and extensible: new environments, reward schemes, or rollout strategies can be easily plugged into the training loop, serving as a foundation for analysis of RL-based agent training.

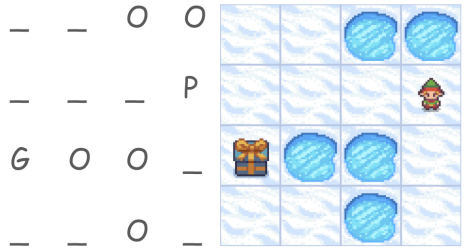
3. Experiment Setup

3.1. Environments and Tasks

We evaluate LLM agents in three **minimal yet comprehensive** symbolic environments designed to isolate core decision-making challenges. These environments are minimal, controllable, and



(a) Sokoban Environment. Actions are irreversible and the boxes cannot be pulled back which needs foresight and careful planning.



(b) FrozenLake Environment. Each agent move may randomly slip to a side direction, requiring the agent to adapt under uncertainty.

Figure 4 | **Sokoban and Frozen Lake environments.** For each environment, the left shows the agent-observed text rendering; the right is a visual illustration. (a) Sokoban is a deterministic multi-turn puzzle where the agent pushes boxes onto targets. (b) Frozen Lake combines multi-turn reasoning and stochasticity where the agent needs to reach the gift to succeed.

stripped of real-world priors, enabling clean analysis of reasoning emergence and learning dynamics. Specifically, **Bandit** (Figure 3) tests risk-sensitive symbolic reasoning under stochastic feedback; **Sokoban** (Figure 4a) requires irreversible multi-step planning in a deterministic setting; and **Frozen Lake** (Figure 4b) combines planning with probabilistic transitions. More details can be found in Appendix B.1.

3.2. Training Settings

We train Qwen-2.5 (0.5B) with StarPO variants on H100 GPUs for 200 rollout–update iterations. Each batch samples P=8 prompts, with N=16 rollouts per prompt, up to 5 turns and 10 actions. Policy updates use GRPO or PPO with GAE (γ=1.0, λ=1.0), Adam optimizer, entropy bonus (β=0.001), and a response-format penalty (−0.1). More details can be found in Appendix B.2.

3.3. Evaluation Metrics

We evaluate on 256 fixed prompts per environment with temperature T=0.5, truncating episodes after 5 turns. Metrics include: **(i)** success rate (task completion), **(ii)** rollout entropy (exploration), **(iii)** in-group reward variance (behavioral diversity), **(iv)** response length (reasoning verbosity), and **(v)** gradient norm (training stability). All are computed over on-policy rollouts and EMA-smoothed. More details can be found in Appendix B.3.

4. Experimental Results and Findings

4.1. Multi-turn agent RL training introduces new instability pattern

We begin by evaluating the baseline performance of StarPO under its default configuration across three agent tasks. As shown in Figure 5, most runs exhibit promising improvements during early-stage training but **ultimately suffer from performance collapse**. This behavior differs from static single-turn tasks, where the collapse issue hardly become the predominant issue. Notably, we observe the PPO variant of StarPO tends to maintain stability longer than the GRPO variant before degradation occurs. For instance, on Bandit and Sokoban, GRPO begins collapsing as early as 20 and 10 steps, respectively, while PPO remains stable until 100 and 50 steps, respectively. These results suggest that while single-turn RL methods like PPO and

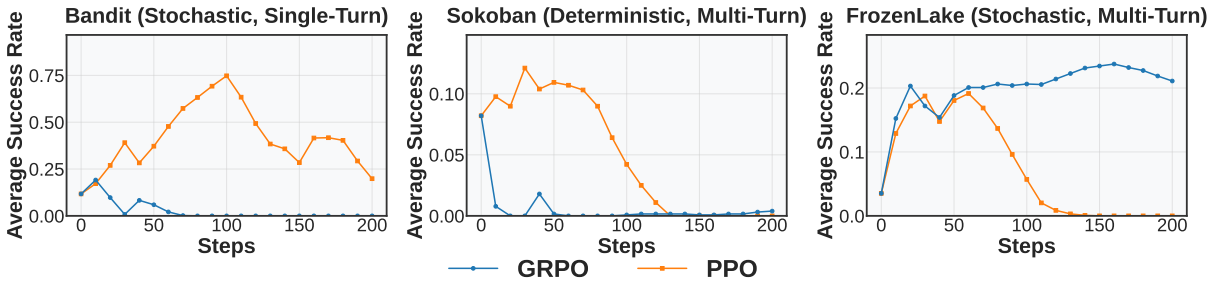


Figure 5 | **Baseline StarPO performance across environments.** All StarPO runs initially improve but eventually collapse in multi-turn agent settings. PPO variant shows better training stability compared to the GRPO variant, especially in Bandit and Sokoban, indicating that critic-based methods better resist early-stage degradation under long-horizon dynamics.

GRPO can initially transfer to multi-turn settings, they may lack robustness for multi-turn agent training where long-horizon interaction is needed; furthermore, the presence of a critic (as in PPO) appears to play a key role in stabilizing the training dynamics. Surprisingly, GRPO variant on Frozen Lake appear to be more stable than the PPO variant. We conjecture this could be due to the inherent feature of the Frozen Lake task where the state value are hard to estimate which may result in reduced stability of PPO, and provide our detailed analyis in Apprendix F.

Finding 1: Single-turn RL may not be directly adapted to Multi-turn agent RL

Vanilla adaptations from single-turn methods like PPO and GRPO achieve early gains in agent settings but often collapse. A critic in PPO may delay instability, but would not prevent reasoning degradation, highlighting the need for specialized stabilization in agent settings.

To understand the causes of collapse, we compare early- and late-stage trajectories. In the Bandit task, early-stage trajectories exhibit diverse reasoning about symbolic meanings and expected rewards, while later-stage responses become repetitive and deterministic. As shown in Table 1, the top rows (Step 0) display varied hypotheses about Dragon and Phoenix, while the bottom rows (Step 150) converge to near-identical phrasing focused on "choosing Dragon" without justification. This suggests that **RL training may have over-amplified inherent reasoning shortcuts**, reinforcing locally rewarded templates while suppressing exploration.

We refer to this failure mode as an "**Echo Trap**", where the model repeatedly reuses memorized reasoning paths under reward feedback, leading to a collapse in trajectory diversity and long-term performance degradation.

Finding 2: Model collapse in agent RL is reflected as "Echo Trap" over training

We find that early-stage agent respond with diverse symbolic reasoning, but collapse into deterministic, repetitive templates after training. Models converge to fixed phrasing, indicating that RL may reinforce superficial patterns instead of general reasoning and forms an "Echo Trap" that hinders long-term generalization.

To identify early signs of collapse, we analyze rollout-level statistical signals that reflect behavioral redundancy and policy instability. Specifically, we examine two early indicators: (1) **Reward Standard Deviation** that measures whether rollouts under the same task instance lead to diverse outcomes or has collapsed into repetitive patterns with similar returns, (2) **Output**

Step	Model Output
0	`Dragon is often associated with patience, strength, and reliability, while Phoenix is associated with speed, agility, and creativity.`
0	`The symbolic meaning of Dragon and Phoenix's names suggests that they may have different reward distributions... Therefore, I think Phoenix is more likely to give higher rewards on average.`
150	`My initial intention is to pick arm Dragon. My first action is to go for arm Dragon.`
150	`My initial intention is to pick arm Dragon, so my first action is to go for arm Dragon.`
150	`My initial intention is to choose Arm Dragon. After analyzing both arms, my intention was to choose Arm Dragon.`

Table 1 | Examples of reasoning patterns in the Bandit task. Top rows show diverse reasoning from model before training, while bottom rows show repetitive and collapsed reasoning after RL training.

Entropy that captures the uncertainty of the model's output, whose sharp drop implies the policy is becoming overconfident and converging to narrow reasoning paths.

To determine whether collapse has already occurred, we further track two metrics: (1) **Average Reward** whose plateau or drop identifies degraded task-solving ability, (2) **Gradient Norm** that measures update magnitude, with its spikes indicate that small updates are triggering large changes, typically a sign of training instability and collapse.

Figure 6 summarizes these dynamics across tasks and optimization methods. From the results, we draw the following conclusions regarding **how model collapse unfolds in multi-turn agent RL**:

- **Reward standard deviation could be a reliable early signal.** For FrozenLake-PPO, the reward mean collapses at step 90, but std drops sharply at step 40—well before performance degrades. In Bandit-PPO, std bottoms out around step 70, just before reward peaks at step 120. In Sokoban-PPO, std and mean collapse almost simultaneously around step 10.
- **Gradient norm spikes indicate irreversible collapse.** Once gradient norm spikes emerge—at step 170 (Bandit), 110 (Sokoban), and 90 (FrozenLake)—even small updates induce drastic parameter shifts, after which recovery becomes unlikely.
- **Entropy typically follows a stable decay trend during effective learning** (e.g., GRPO on FrozenLake). Rapid entropy increases or erratic changes often correlate with collapsed reasoning behavior (e.g. GRPO on Bandit and Sokoban).

Finding 3: Collapse follows similar dynamics and can be anticipated by indicators

Reward standard deviation and **entropy** often fluctuate before performance degrades, while **gradient norm** spikes typically mark the point of irreversible collapse. These metrics provide early indicators and motivate the need for stabilization strategies.

These patterns confirm that multi-turn RL introduces unique challenges that single-turn RL methods fail to handle. In response, we introduce **StarPO-S**, a stabilized variant that targets sampling quality, gradient stability, and exploration regularization to avoid premature collapse.

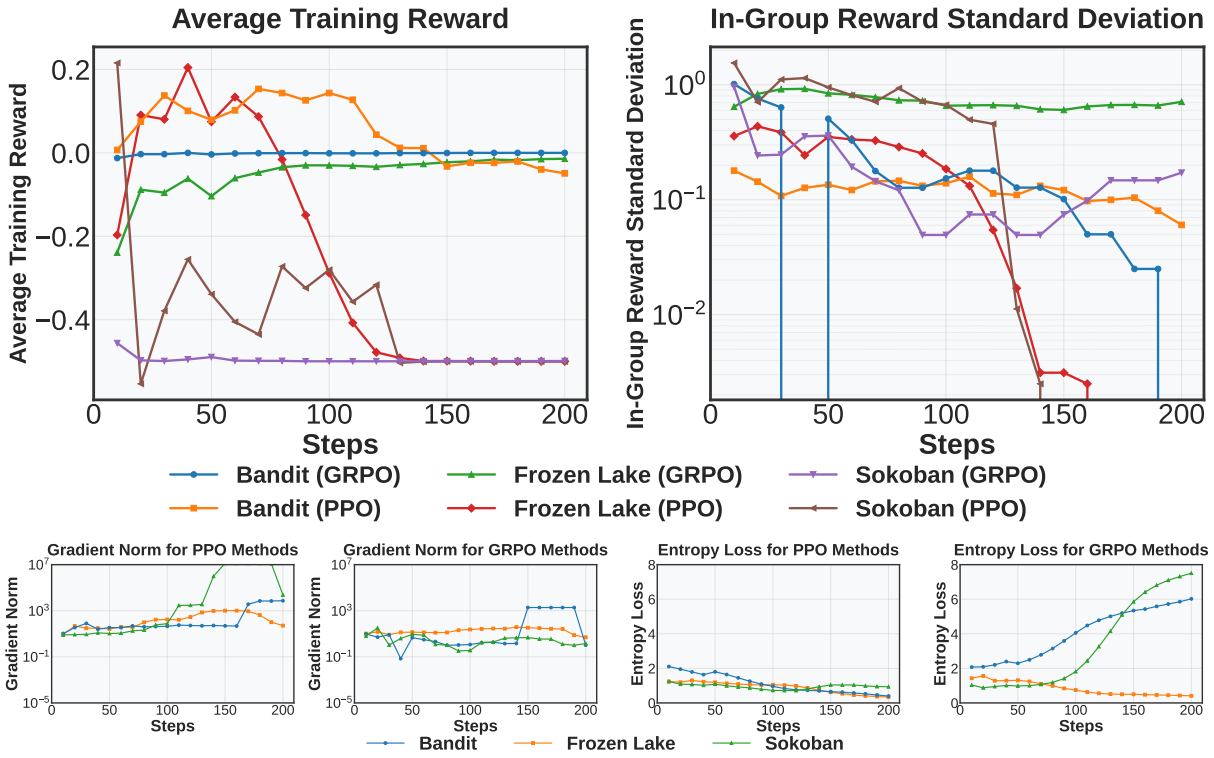


Figure 6 | **Collapse indicators and early warning signals in multi-turn RL.** Reward standard deviation and entropy (right-side plots) drop early, often before reward degrades, serving as early warning signals. Reward mean and gradient norm (left-side plots) reflect collapse directly—plateaus and spikes confirm performance and training instability.

4.2. StarPO-S: Stabilize Multi-turn RL with instance filtering and exploration encouragement

To address the instability of multi-turn reinforcement learning, we introduce **StarPO-S**, a stabilized variant of StarPO that incorporates three key modifications aimed at improving training robustness and efficiency. Building on the insight that declining reward standard deviation often precedes collapse, we investigate the following question: *should agents be trained more intensively on task instances where their behavior is more uncertain with higher reward variance?*

We hypothesize that the most effective training samples are those where the agent **exhibits outcome uncertainty**—avoiding both trivial task instances that lack learning value and overly difficult ones that yield little reward signal. This intuition is rooted in principles of active learning (Settles, 2009), where uncertain examples are the most informative ones models should learn from. We define trajectory-level outcome uncertainty U for policy π_θ on a given agent task instance (initial state s_0 in an MDP $\mathcal{M} = \{S, A, P\}$) as:

$$\mathrm{U}(\pi_\theta, \mathcal{M}, s_0) = \mathrm{Std}_{\tau \sim \pi_\theta(\cdot|s_0)} \left[R(\tau) \right]. \tag{7}$$

During training, we sort prompts based on the standard deviation of reward obtained from repeated rollouts and **retain only the top p% highly-uncertain prompts** at each training step. Figure 7 shows the effect of varying p in PPO and GRPO under StarPO-S.

In PPO runs (Figure 7, left two figures), filtering out low-variance rollouts significantly delays collapse. For example, a 75% retention ratio pushes the collapse point from 100 to 140 steps in FrozenLake, while keeping only 50% of rollouts eliminates collapse within the training

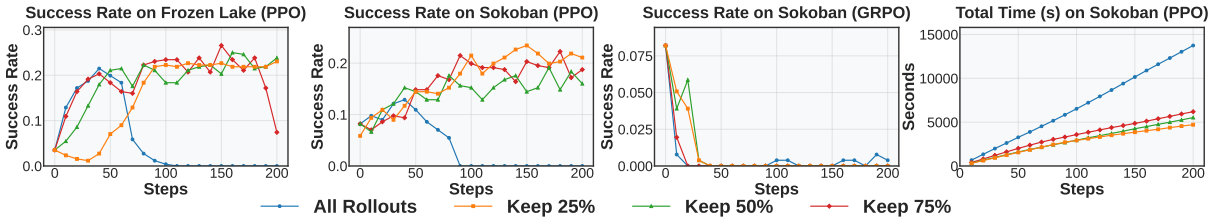


Figure 7 | **Effect of uncertainty-based filtering on multi-turn RL stability.** Filtering out low-variance trajectories reduces collapse risk and improves success rate. On PPO variants, collapse is largely mitigated when more than half of the trajectories are filtered.

horizon entirely. GRPO (third figure from left) is inherently less stable due to its critic-free design, but still benefits modestly from variance-based filtering.

Interestingly, retaining only high-variance samples also improves training efficiency. As shown in the rightmost sub-figure of Fig 7, keeping just 25% of the rollouts reduces total update steps by half, while not sacrificing early learning gains. We adopt 25% as the default filtering ratio for StarPO-S in our experiments. However, we note that this aggressive value may not be optimal for all settings. Tasks like Sokoban and FrozenLake appear to benefit more from aggressive filtering, potentially due to their relatively repetitive reasoning patterns and under-representation in pretraining, which make them tend to collapse when similar trajectories dominate the batch.

Finding 4: Filtering low-variance trajectories improves stability and efficiency

Training on high-variance prompts delays or eliminates collapse in multi-turn RL. StarPO-S improves performance and reduces update steps by discarding low-information rollouts, especially under PPO. This aligns with active learning principles, where uncertain examples offer the most informative learning signals.

In addition to uncertainty-based filtering, we adopt two stabilization techniques in StarPO-S inspired by DAPO (Yu et al., 2025), originally designed for single-turn RL. We extend and evaluate them in the multi-turn agent setting:

- **KL Term Removal:** We eliminate the KL divergence penalty from PPO's objective, relying only on policy loss and entropy bonus for gradient updates. It removes the constraint to stay close to the initial model distribution and encourage the model to explore.
- **Clip-Higher (Asymmetric Clipping):** We decouple the PPO clipping range by using a higher upper bound ($\varepsilon_{\text{high}} = 0.28$) than the lower bound ($\varepsilon_{\text{low}} = 0.2$). It allows the model to learn more aggressively from high-reward rollouts for more effective training.

As shown in Figure 8, both methods boost the success rate and extend stable training phases, showing how multi-turn RL benefits from more flexible gradient shaping—amplifying effective reasoning trajectories while avoiding over-penalization of uncertain ones.

Overall Comparison. We compare StarPO-S with vanilla StarPO across three tasks in Figure 9. StarPO-S consistently delays collapse and enhances final task performance. We attribute these gains to more selective training data (via uncertainty filtering), more balanced optimization signals (via KL removal and decoupled clipping), reducing narrowed reasoning modes.

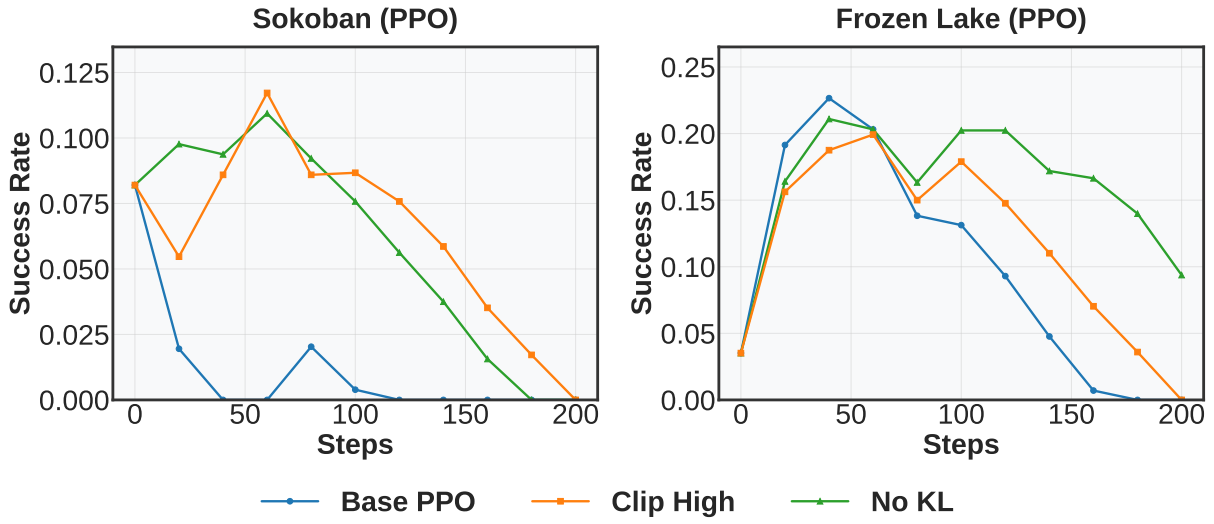


Figure 8 | **Effect of KL removal and asymmetric clipping on PPO stability.** Removing KL constraints and enabling stronger positive gradient flow both improve peak performance and delay collapse in multi-turn RL.

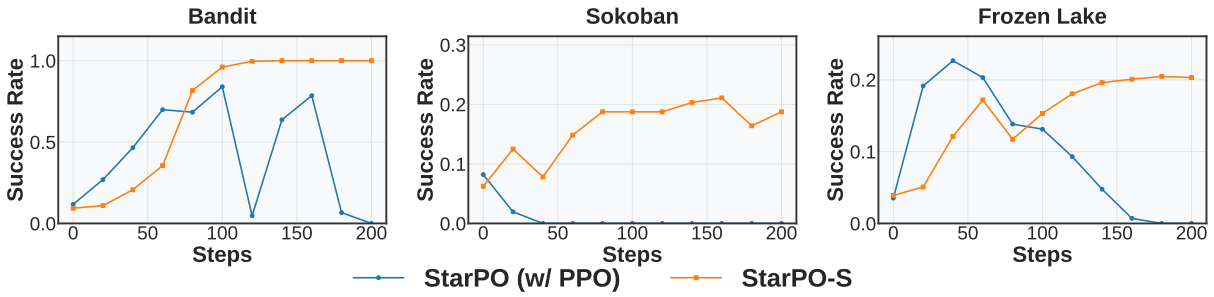


Figure 9 | **StarPO-S improves stability and final performance across tasks.** Compared to vanilla StarPO, StarPO-S achieves higher success rates and reliefs collapse in all three tasks.

4.3. Generating Useful Trajectories for RL Training

Effective RL training depends critically on the quality of trajectory rollouts. We identify three key rollout dimensions that significantly affect learning dynamics and generalization: *task diversity*, *interaction granularity*, and *rollout frequency*. We further analyze how these factors affect generalization by training these models on the vanilla Sokoban task and evaluating them on the SokobanNewVocab, LargeSokoban, and FrozenLake Task, which we detail in Appendix G.

Higher task diversity with response comparison improves generalization. Task diversity refers to the number of distinct initial states used during training. A diverse prompt set exposes the model to broader decision-making contexts, aiding generalization beyond memorized behaviors. Under a fixed batch size, task diversity is inversely related to the number of responses per prompt. In our experiments (Table 2), we vary this trade-off and find that higher task diversity—achieved by fewer responses per prompt (e.g., 4 per prompt)—consistently yields better generalization. However, this only holds when each prompt includes multiple rollouts, enabling the agent to contrast differnet outcomes under similar scenario and refine its policy.

Allowing more action budgets enables planning, while overly long rollouts inject noise. We vary the number of actions allowed per turn in Table 3. Allowing up to 5 or 6 actions per turn

Response Per Prompt	SingleSokoban	SokobanNewVocab	FrozenLake
32	21.09%	20.22%	17.97%
16	20.31%	21.48%	19.53%
8	20.31%	19.53%	17.19%
4	**20.70%**	**25.39%**	**21.48%**
2	19.92%	25.00%	12.50%
1	19.53%	22.27%	12.50%

Table 2 | **Effect of Task Diversity on Generalization Performance (%).** Higher diversity with moderate response comparison (4 responses per prompt) yields the best performance.

Max Actions / Turn	Sokoban	SokobanNewVocab	LargeSokoban	FrozenLake
1	12.11%	13.67%	1.17%	11.72%
2	16.41%	21.09%	3.52%	18.36%
3	19.53%	19.53%	1.95%	20.88%
4	26.95%	26.95%	5.08%	20.70%
5	28.13%	25.78%	6.25%	**21.09%**
6	**33.59%**	**31.64%**	**6.64%**	18.36%
7	22.27%	28.52%	3.91%	19.53%

Table 3 | **Performance across environments under different per-turn action budgets (%).** Allowing 5–6 actions per turn consistently improves success rates, enabling effective multi-step planning while avoiding the noise introduced by overly long rollouts.

yields the best performance, especially on complex environments like SokobanNewVocab and LargeSokoban. This setting provides enough room for planning while avoiding the chaos of overly long rollouts. Increasing the budget to 7 actions degrades performance, likely due to noisy transitions and diluted reward feedback.

Frequent rollout updates ensure alignment between optimization targets and current policy behavior. As shown in Figure 10, agents trained with up-to-date rollouts (online-style collection every 10 updates) achieve faster convergence and higher final success rates compared to agents relying on older rollouts, on both direct evaluation (left) and generalization evaluation (middle and right). We highlight a core design principle in multi-turn RL: learning is most effective when trajectory data reflects the agent's most recent behavior. Frequent sampling mitigates policy-data mismatch and prevents optimization using outdated policy states.

Frequent rollout updates align optimization with the current policy and stabilize learning. To investigate the effect of rollout freshness, we adopt an *Online-k* rollout strategy, where a single set of rollouts is reused for k consecutive policy updates. A smaller k implies more frequent rollout collection. Notably, *Online-1* corresponds to an almost fully online setting, with fresh rollouts collected every update iteration. Importantly, we keep the update batch size fixed across conditions, isolating the effect of rollout frequency from optimization scale.

As shown in Figure 10, agents trained with fresher rollouts (*Online-1*) achieve faster convergence and better generalization across tasks compared to those with delayed updates (e.g., *Online-5* or *Online-10*). This supports a core design principle for multi-turn RL: learning is most effective when trajectories reflect the agent's latest behavior. Frequent rollout refresh reduces policy-data mismatch and improves optimization stability.

	Train on Bandit		Train on Sokoban			
	Bandit	Bandit-Rev	FrozenLake	LargeSokoban	Sokoban	SokobanNewVocab
StarPO-S	**100.00**	**67.58**	**19.92**	2.34	21.48	18.75
NoThinking	81.25	56.25	19.53	**2.73**	**20.73**	**26.17**

Table 4 | Generalization performance (%) with and without reasoning under StarPO-S. Disabling `<think>` tokens significantly reduces generalization in single-turn Bandit task, but has mixed or marginal effects in multi-turn Sokoban task.

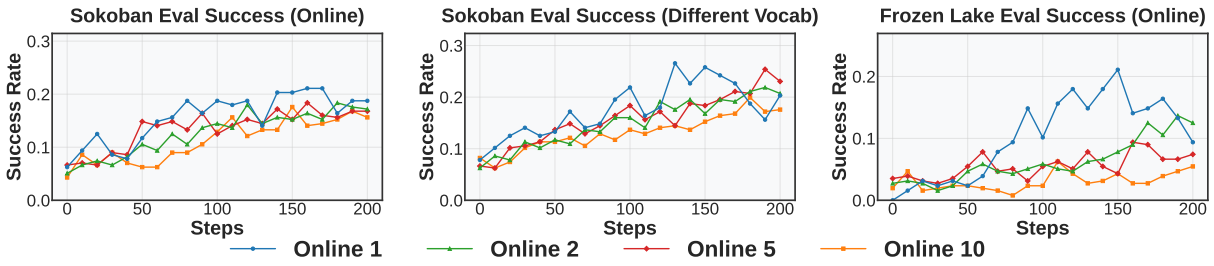


Figure 10 | **Performance under different rollout frequencies (*Online-k*).** We vary the rollout reuse factor k, where each rollout batch is used for k consecutive policy updates while keeping the update batch size fixed. Lower values (e.g., *Online-1*) correspond to more frequent rollout collection. Fresher rollouts lead to faster convergence and stronger generalization across tasks by better aligning data with the current policy.

These findings highlight that rollout quality is multifaceted. Stale or misaligned rollouts can induce collapse, while maintaining freshness—alongside limits on action granularity and sufficient task diversity—enables stable and effective RL training.

Finding 5: Task diversity, action budget, and rollout frequency affect data quality

Diverse task instances enable better policy contrast and generalization across environments. Moderate action budgets provide enough planning space and avoid the noise introduced by overly long sequences. Up-to-date rollouts ensure optimization targets remain aligned with current policy behavior.

4.4. Reasoning Emerges in Single-Turn Tasks but Fails to Grow in Multi-Turn Settings Without Fine-Grained Reward Signals

We investigate the role of symbolic reasoning in agent training by comparing its effect in each environment. We find that reasoning notably improves generalization in single-turn tasks like Bandit, but fails to grow or persist in more complex, multi-step environments like Sokoban. Below, we analyze these effects step-by-step.

Reasoning traces improve symbolic generalization in single-turn Bandit tasks. We design a controlled generalization test in symbolic bandit environments, where each arm is associated with a name and a distinct reward distribution. In the `Bandit` setting, the model is trained on `[Teacher, Engineer]` and evaluated on a disjoint set `[Librarian, Trader]`, while preserving intuitive risk-reward alignments (e.g., more ambitious professions map to higher risk and reward). In contrast, the `BanditRev` setting inverts these associations, assigning

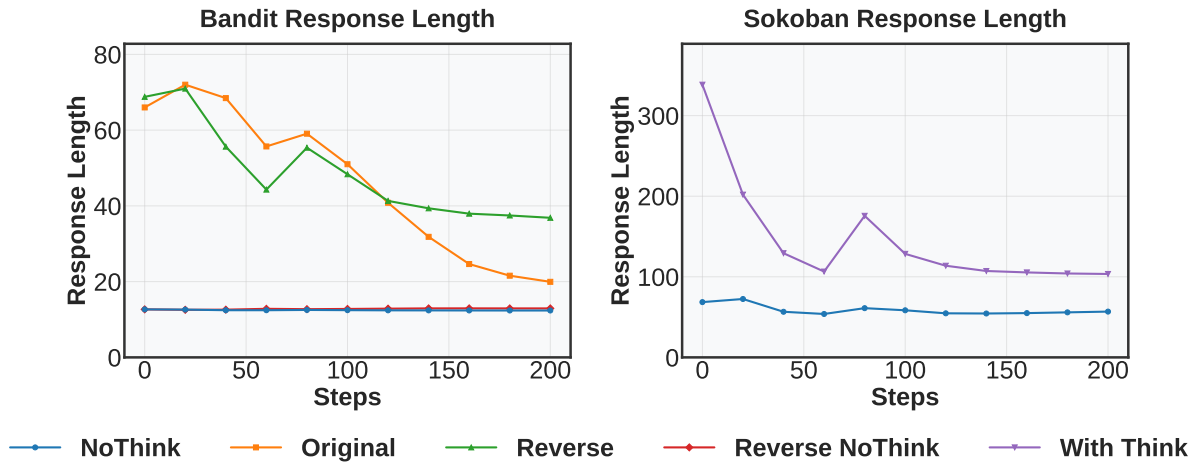


Figure 11 | **Reasoning length over training iterations across different tasks.** We track the average token count of reasoning segments (`<think>` blocks) during RL training. Across all environments, reasoning length declines as training progresses, with `BanditRev` maintaining longer traces—possibly due to greater semantic-reward conflict requiring more deliberation.

counter-intuitive reward profiles (e.g., `Librarian` = high-risk, high-reward), making semantic reasoning more challenging.

Despite the added difficulty in `BanditRev`, models trained with explicit reasoning supervision—via `<think>` tokens—consistently outperform those without, as shown in Table 4. This suggests that reasoning traces help the agent internalize symbolic-reward associations and generalize beyond surface-level memorization, even under semantic-reward misalignment.

In multi-turn tasks, reasoning signals fade as training progresses. In contrast, for tasks like Sokoban and FrozenLake, reasoning provides limited benefit. Even when responses are structured to include `<think>` segments, removing reasoning from prompts (no-think variant) results in comparable or even better performance.

To understand this phenomenon, we analyze the average response length across tasks and settings. As shown in Figure 11, reasoning length consistently declines during training—indicating that the model gradually suppresses its own thought processes. Interestingly, in `BanditRev`, where semantic labels and reward structures are misaligned, reasoning traces tend to be longer—suggesting the agent expends more effort reconciling symbolic cues with observed rewards.

Reward signals in multi-turn tasks may be too noisy to reliably support fine-grained reasoning learning. We hypothesize that reasoning collapse in multi-turn settings stems not only from limited supervision, but also from the structure of the reward landscape. In such environments, reward signals are often sparse, delayed, and outcome-based—making it difficult to distinguish between successful trajectories driven by coherent reasoning and those achieved through trial-and-error. We observe instances where models generate incoherent or hallucinated reasoning yet still arrive at the correct answer.

This raises an important challenge: *how can we consistently reinforce useful reasoning when the reward alone may not reflect its quality*? One possible approach is to decouple action correctness from reasoning quality. Inspired by GRPO, we apply format-based penalties: when the model fails to produce valid `<think>`–`<answer>` structures, we reduce the reward—even if the final

answer is correct. This encourages the model to maintain interpretable reasoning traces and avoid collapsing into shortcut policies.

We believe future work could explore reward designs that directly reinforce intermediate reasoning steps—rewarding partial correctness or trajectory-level explanation quality—rather than relying solely on trajectory-level outcome-based reward feedback.

Finding 6: Reasoning fails to emerge without meticulous reward design

While symbolic reasoning can emerge in simple, single-turn tasks under weak supervision, it fails to persist in multi-turn environments without the reward design explicitly encouraging interpretable intermediate reasoning steps. We observe that even with structured prompts, reasoning gradually decays during training if the reward signal focuses only on final outcomes. This suggests that without meticulous reward shaping, agents may tend to collapse into shortcut behaviors that bypass reasoning altogether.

5. Related Work

Reinforcement Learning for Reasoning in LLMs. Reinforcement learning (RL) on LLMs (Chen et al., 2021; Christiano et al., 2023; Ouyang et al., 2022) has significantly improved LLMs' reasoning capabilities. Notable approaches include the use of Proximal Policy Optimization Algorithms (PPO) (Schulman et al., 2017) which maintains training stability while enhancing performance by clipping policy updates, Group Relative Policy Optimization (GRPO) (DeepSeek-AI et al., 2025) for enhancing the ability of systematic problem-solving, soft actor-critic (SAC) (Haarnoja et al., 2018) leverages an entropy-regularized objective to promote robust exploration and stability, and meta tokens (Goyal et al., 2024; Herel and Mikolov, 2024; Pfau et al., 2024) for structured thinking. Other significant developments include Process Reward Model (PRM) (Lightman et al., 2023; Zhang et al., 2025) and Monte Carlo Tree Search (MCTS) based approaches (Hao et al., 2023a) for systematic problem-solving. On the other hand, recent advances in LLM reasoning have explored techniques to enable models to generate intermediate chain-of-thought rationales. In particular, STaR (Zelikman et al., 2022) iteratively leverages a small set of rationale examples along with a large dataset without rationales. DAPO (Yu et al., 2025), Dr. GRPO (Liu et al., 2025), and Open Reasoner Zero (Hu et al., 2025) all demonstrate that minimalist, reproducible RL techniques—featuring decoupled clipping, unbiased optimization, and simple reward schemes—can significantly enhance LLM reasoning performance.

Existing agent frameworks. LLM-based agent architectures have evolved from early reasoning-action frameworks (Lin et al., 2024a; Shinn et al., 2024; Xu et al., 2023; Yao et al., 2022b) to structured planning approaches (Hao et al., 2023a; Liu et al., 2023). Multi-agent systems (Chen et al., 2023; Du et al., 2023; Li et al., 2023; Wang et al., 2024a) are designed for tasks with more complex interactions. Widely used platforms such as OpenAI Gym (Brockman et al., 2016) and specialized environments including Sokoban (Junghanns and Schaeffer, 2001), FrozenLake (Dell'Aversana, 2021), and Webshop (Yao et al., 2022a) provide diverse testbeds for evaluating these agents. Moreover, general-purpose systems like HuggingGPT (Shen et al., 2024) and other frameworks (Hao et al., 2023b; Wu et al., 2023; Xie et al., 2023; Zhuang et al., 2023) have enabled broad applications ranging from web navigation (Qi et al., 2025), coding copilot (DeepSeek-AI et al., 2024; Jimenez et al., 2024; Wang et al., 2024b) to embodied tasks (Li et al., 2025; Lin et al., 2024b; Xi et al., 2024). Social interaction capabilities have been advanced through Generative Agents and AgentSims (Lin et al., 2023; Park et al., 2023). Challenges persist in architectural

complexity and self-correction (He et al., 2025), especially for diverse, multi-step reasoning tasks (Nguyen et al., 2024; Song et al., 2024; Wang et al., 2025).

6. Conclusions and Broad Impact

In this work, we demonstrate that reinforcement learning, when adapted for complex and stochastic environments, can effectively train language agents to reason and act. It marks a shift from procedure-heavy, human-supervised learning toward reward-driven training based on verifiable outcomes. This opens up a scalable and principled path for building AI systems in domains such as theorem proving, software engineering, scientific discovery, and game playing. Future directions include extending to multi-modal inputs, improving training efficiency, and applying to tasks with complex but checkable objectives.

Limitations

We note RAGEN's limitations:

1. **Model scaling** RAGEN has yet to be evaluated on multimodal models or larger models.
2. **Rewards** RAGEN is not yet optimised for domains without easily verifiable rewards.
3. **Long context:** Long multi-turn context results in large KV-cache, which limits training efficiency on longer more complex tasks.

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 to the TinyZero team for their discoveries that informed our initial exploration. We would like to appreciate insightful discussions with Han Liu, Xinyu Xing, Li Erran Li, John Schulman, Akari Asai, Eiso Kant, Lu Lu, Runxin Xu, Huajian Xin, Zijun Liu, Weiyi Liu, Weimin Wu, Yibo Wen, Jiarui Liu, Lorenzo Xiao, Ishan Mukherjee, Anabella Isaro, Haosen Sun, How-Yeh Wan, Lester Xue, Matthew Khoriaty, Haoxiang Sun, Jiajun Liu.

References

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016. URL https://arxiv.org/abs/1606.01540.

L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. arXiv preprint arXiv:2106.01345, 2021.

W. Chen, Y. Su, J. Zuo, C. Yang, C. Yuan, C. Qian, C.-M. Chan, Y. Qin, Y. Lu, R. Xie, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. arXiv preprint arXiv:2308.10848, 2(4):6, 2023.

P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences, 2023. URL https://arxiv.org/abs/1706.03741.

DeepSeek-AI, Q. Zhu, D. Guo, Z. Shao, D. Yang, P. Wang, R. Xu, Y. Wu, Y. Li, H. Gao, S. Ma, W. Zeng, X. Bi, Z. Gu, H. Xu, D. Dai, K. Dong, L. Zhang, Y. Piao, Z. Gou, Z. Xie, Z. Hao, B. Wang, J. Song, D. Chen, X. Xie, K. Guan, Y. You, A. Liu, Q. Du, W. Gao, X. Lu, Q. Chen, Y. Wang, C. Deng, J. Li, C. Zhao, C. Ruan, F. Luo, and W. Liang. Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence, 2024. URL https://arxiv.org/abs/2406.11931.

DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, X. Zhang, X. Yu, Y. Wu, Z. F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Ding, H. Xin, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Wang, J. Chen, J. Yuan, J. Qiu, J. Li, J. L. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R. J. Chen, R. L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, S. S. Li, S. Zhou, S. Wu, S. Ye, T. Yun, T. Pei, T. Sun, T. Wang, W. Zeng, W. Zhao, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, W. L. Xiao, W. An, X. Liu, X. Wang, X. Chen, X. Nie, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yang, X. Li, X. Su, X. Lin, X. Q. Li, X. Jin, X. Shen, X. Chen, X. Sun, X. Wang, X. Song, X. Zhou, X. Wang, X. Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. Zhang, Y. Xu, Y. Li, Y. Zhao, Y. Sun, Y. Wang, Y. Yu, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Ou, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Xiong, Y. Luo, Y. You, Y. Liu, Y. Zhou, Y. X. Zhu, Y. Xu, Y. Huang, Y. Li, Y. Zheng, Y. Zhu, Y. Ma, Y. Tang, Y. Zha, Y. Yan, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Xie, Z. Zhang, Z. Hao, Z. Ma, Z. Yan, Z. Wu, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Pan, Z. Huang, Z. Xu, Z. Zhang, and Z. Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

P. Dell'Aversana. The frozen lake problem. an example of optimization policy, 12 2021.

Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch. Improving factuality and reasoning in language models through multiagent debate. arXiv preprint arXiv:2305.14325, 2023.

Z. Gao, W. Zhan, J. D. Chang, G. Swamy, K. Brantley, J. D. Lee, and W. Sun. Regressing the relative future: Efficient policy optimization for multi-turn rlhf, 2024. URL https://arxiv.org/abs/2410.04612.

S. Goyal, Z. Ji, A. S. Rawat, A. K. Menon, S. Kumar, and V. Nagarajan. Think before you speak: Training language models with pause tokens, 2024. URL https://arxiv.org/abs/2310.02226.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290.

S. Hao, Y. Gu, H. Ma, J. J. Hong, Z. Wang, D. Z. Wang, and Z. Hu. Reasoning with language model is planning with world model. arXiv preprint arXiv:2305.14992, 2023a.

S. Hao, T. Liu, Z. Wang, and Z. Hu. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. Advances in neural information processing systems, 36: 45870–45894, 2023b.

C. He, B. Zou, X. Li, J. Chen, and H. M. Junliang Xing. Enhancing llm reasoning with multi-path collaborative reactive and reflection agents, 2025. URL https://arxiv.org/abs/2501.00430.

D. Herel and T. Mikolov. Thinking tokens for language modeling, 2024. URL https://arxiv.org/abs/2405.08644.

E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

J. Hu, Y. Zhang, Q. Han, D. Jiang, X. Zhang, and H.-Y. 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.

C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. Narasimhan. SWE-bench: Can language models resolve real-world github issues? arXiv preprint arXiv:2310.06770, 2024.

A. Junghanns and J. Schaeffer. Sokoban: Enhancing general single-agent search methods using domain knowledge. Artificial Intelligence, 129(1):219–251, 2001. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(01)00109-6. URL https://www.sciencedirect.com/science/article/pii/S0004370201001096.

A. Kumar, V. Zhuang, R. Agarwal, Y. Su, J. D. Co-Reyes, A. Singh, K. Baumli, S. Iqbal, C. Bishop, R. Roelofs, L. M. Zhang, K. McKinney, D. Shrivastava, C. Paduraru, G. Tucker, D. Precup, F. Behbahani, and A. Faust. Training language models to self-correct via reinforcement learning, 2024. URL https://arxiv.org/abs/2409.12917.

G. Li, H. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. Advances in Neural Information Processing Systems, 36:51991–52008, 2023.

M. Li, S. Zhao, Q. Wang, K. Wang, Y. Zhou, S. Srivastava, C. Gokmen, T. Lee, L. E. Li, R. Zhang, W. Liu, P. Liang, L. Fei-Fei, J. Mao, and J. Wu. Embodied agent interface: Benchmarking llms for embodied decision making, 2025. URL https://arxiv.org/abs/2410.07166.

H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe. Let's verify step by step. arXiv preprint arXiv:2305.20050, 2023.

B. Y. Lin, Y. Fu, K. Yang, F. Brahman, S. Huang, C. Bhagavatula, P. Ammanabrolu, Y. Choi, and X. Ren. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. Advances in Neural Information Processing Systems, 36, 2024a.

J. Lin, H. Zhao, A. Zhang, Y. Wu, H. Ping, and Q. Chen. Agentsims: An open-source sandbox for large language model evaluation, 2023. URL https://arxiv.org/abs/2308.04026.

J. Lin, H. Gao, X. Feng, R. Xu, C. Wang, M. Zhang, L. Guo, and S. Xu. Advances in embodied navigation using large language models: A survey, 2024b. URL https://arxiv.org/abs/2311.00530.

B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone. Llm+ p: Empowering large language models with optimal planning proficiency. arXiv preprint arXiv:2304.11477, 2023.

Z. Liu, C. Chen, W. Li, P. Qi, C. D. Tianyu Pang, W. S. Lee, and M. Lin. Understanding r1-zero-like training: A critical perspective, 2025. URL https://arxiv.org/abs/2503.20783.

M. Nguyen, A. Baker, C. Neo, A. Roush, A. Kirsch, and R. Shwartz-Ziv. Turning up the heat: Min-p sampling for creative and coherent llm outputs, 2024. URL https://arxiv.org/abs/2407.01082.

OpenAI. Introducing ChatGPT o1, 2024. URL https://openai.com/o1/. Accessed: 2025-02-15.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

J. Pan, J. Zhang, X. Wang, L. Yuan, H. Peng, and A. Suhr. Tinyzero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein. Generative agents: Interactive simulacra of human behavior, 2023. URL https://arxiv.org/abs/2304.03442.

J. Pfau, W. Merrill, and S. R. Bowman. Let's think dot by dot: Hidden computation in transformer language models, 2024. URL https://arxiv.org/abs/2404.15758.

Z. Qi, X. Liu, I. L. Iong, H. Lai, X. Sun, W. Zhao, Y. Yang, X. Yang, J. Sun, S. Yao, T. Zhang, W. Xu, J. Tang, and Y. Dong. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning, 2025. URL https://arxiv.org/abs/2411.02337.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018. URL https://arxiv.org/abs/1506.02438.

B. Settles. Active learning literature survey. 2009.

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. Advances in Neural Information Processing Systems, 36, 2024.

N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: Language agents with verbal reinforcement learning. Advances in Neural Information Processing Systems, 36, 2024.

Y. Song, D. Yin, X. Yue, J. Huang, S. Li, and B. Y. Lin. Trial and error: Exploration-based trajectory optimization for llm agents, 2024.

C. J. Wang, D. Lee, C. Menghini, J. Mols, J. Doughty, A. Khoja, J. Lynch, S. Hendryx, S. Yue, and D. Hendrycks. Enigmaeval: A benchmark of long multimodal reasoning challenges, 2025. URL https://arxiv.org/abs/2502.08859.

Q. Wang, Z. Wang, Y. Su, H. Tong, and Y. Song. Rethinking the bounds of llm reasoning: Are multi-agent discussions the key? arXiv preprint arXiv:2402.18272, 2024a.

X. Wang, Z. Wang, J. Liu, Y. Chen, L. Yuan, H. Peng, and H. Ji. Mint: Evaluating llms in multi-turn interaction with tools and language feedback, 2024b. URL https://arxiv.org/abs/2309.10691.

Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. arXiv preprint arXiv:2308.08155, 2023.

J. Xi, Y. He, J. Yang, Y. Dai, and J. Chai. Teaching embodied reinforcement learning agents: Informativeness and diversity of language use, 2024. URL https://arxiv.org/abs/2410.24218.

T. Xie, F. Zhou, Z. Cheng, P. Shi, L. Weng, Y. Liu, T. J. Hua, J. Zhao, Q. Liu, C. Liu, et al. Openagents: An open platform for language agents in the wild. arXiv preprint arXiv:2310.10634, 2023.

B. Xu, Z. Peng, B. Lei, S. Mukherjee, Y. Liu, and D. Xu. Rewoo: Decoupling reasoning from observations for efficient augmented language models. arXiv preprint arXiv:2305.18323, 2023.

A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, and Z. Qiu. Qwen2.5 technical report. arXiv preprint arXiv:2412.15115, 2024.

S. Yao, H. Chen, J. Yang, and K. Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. Advances in Neural Information Processing Systems, 35:20744–20757, 2022a.

S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. arXiv preprint arXiv:2210.03629, 2022b.

Q. Yu, Z. Zhang, R. Zhu, Y. Yuan, X. Zuo, Y. Yue, T. Fan, G. Liu, L. Liu, X. Liu, H. Lin, Z. Lin, B. Ma, G. Sheng, Y. Tong, C. Zhang, M. Zhang, W. Zhang, H. Zhu, J. Zhu, J. Chen, J. Chen, C. Wang, H. Yu, W. Dai, Y. Song, X. Wei, H. Zhou, J. Liu, W.-Y. Ma, Y.-Q. Zhang, L. Yan, M. Qiao, Y. Wu, and M. Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

E. Zelikman, Y. Wu, J. Mu, and N. D. Goodman. Star: Bootstrapping reasoning with reasoning, 2022. URL https://arxiv.org/abs/2203.14465.

W. Zeng, Y. Huang, W. Liu, K. He, Q. Liu, Z. Ma, and J. He. 7b model and 8k examples: Emerging reasoning with reinforcement learning is both effective and efficient. https://hkust-nlp.notion.site/simplerl-reason, 2025. Notion Blog.

Z. Zhang, C. Zheng, Y. Wu, B. Zhang, R. Lin, B. Yu, D. Liu, J. Zhou, and J. Lin. The lessons of developing process reward models in mathematical reasoning, 2025. URL https://arxiv.org/abs/2501.07301.

Y. Zhuang, X. Chen, T. Yu, S. Mitra, V. Bursztyn, R. A. Rossi, S. Sarkhel, and C. Zhang. Toolchain*: Efficient action space navigation in large language models with a* search. arXiv preprint arXiv:2310.13227, 2023.

A. Background of Reinforcement Learning

Reinforcement learning (RL) enables foundation models to learn through interaction and reward signals. The general RL objective is:

$$J(\theta) = \mathbb{E}_{s\sim\mathcal{D},a\sim\pi_\theta(\cdot|s)}[R(s,a)], \tag{8}$$

where π_θ is the policy, s is the input prompt, a is the response, and $R(s,a)$ is the reward function evaluating response quality.

Common approaches use reward modeling and policy optimization for RL. Proximal Policy Optimization (PPO) stabilizes training through probability ratio clipping and advantage estimation (Schulman et al., 2017). The probability ratio is defined as:

$$\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{9}$$

The PPO objective uses this ratio with clipping:

$$J_{PPO}(\theta) = \mathbb{E}_t[\min(\rho_i A_i, \hat{\rho}_i A_i) - \beta D_{KL}], \tag{10}$$

with probability ratio $\rho_i = \frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}$ and clipped ratio $\hat{\rho}_i = \text{clip}(\rho_i, 1-\varepsilon, 1+\varepsilon)$.

For advantage estimation, Generalized Advantage Estimation (GAE) (Schulman et al., 2018) computes:

$$A_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l} \tag{11}$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ is the TD error, and (γ, λ) control the bias-variance tradeoff.

Recently, DeepSeek-R1-Zero implements this paradigm through Group Relative Policy Optimization (GRPO), sampling G outputs $\{o_i\}$ [consisting of reasoning and actions] for each prompt and optimizes:

$$J_{GRPO}(\theta) = \mathbb{E}_{q,\{o_i\}}[J_{group}(\theta)], \tag{12}$$

where:

$$J_{group}(\theta) = \frac{1}{G}\sum_{i=1}^{G}\min(\rho_i A_i, \hat{\rho}_i A_i) - \beta D_{KL}, \tag{13}$$

while mostly similar to Eq. 3, the GRPO advantage is neural-model free and calculated as:

$$A_i = \frac{r_i - \text{mean}(\{r_j\})}{\text{std}(\{r_j\})}. \tag{14}$$

Using rule-based rewards r_i, this pure RL approach demonstrates emergent reasoning behaviors.

B. Detailed Experimental Settings

B.1. Environments and Tasks

We construct a **minimal yet comprehensive** testbed comprising three symbolic environments to evaluate LLM agents across key axes of decision-making complexity. Crucially, these environments are synthetic, controllable, and symbolically structured, decoupled from real-world priors or task-specific conventions. Current models hardly benefit from general-purpose instruction

fine-tuning or model scale, with larger models up to 32B also perform poorly without training. It enables fair evaluation of RL learning dynamics from scratch, and allows us to systematically study reasoning emergence, training stability, and generalization in agentic LLMs.

Specifically, each environment is designed to stress a different capability: Bandits targets reasoning under uncertainty, Sokoban focuses on irreversible long-horizon planning, and Frozen Lake couples planning with probabilistic transitions.

Bi-Arm Bandits. We design this environment to evaluate whether agents can **form risk-sensitive hypotheses and revise them based on training**. At each step, the agent must choose between two semantically symbolic options—e.g., "Dragon" vs. "Phoenix"—each linked to a fixed reward distribution (Figure 3). The low-risk arm always returns a reward of 0.15, while the high-risk arm samples from $Bernoulli(0.25)$: higher variance, higher expected return.

Importantly, the low-risk arm wins more often per trial, even though the high-risk arm is better in expectation. This designed to test reasoning: without inductive bias, models may prefer the lo-arm due to its more frequent success, but a reasoning agent must learn to associate symbolic cues (e.g., "Dragon") with underlying reward statistics, override misleading short-term signals, and "justify" high-risk choices based on long-term expected return. We further test this by reversing the symbolic labels to probe agent's reasoning under opposed reward systems.

Sokoban. We use the puzzle Sokoban (Figure 4a) to study multi-turn agent interaction. The agent must push a box to the goal in a grid within constrained steps. Unlike standard navigation, Sokoban is irreversible: boxes can only be pushed, not pulled back, which requires the agent to reason ahead to avoid dead-ends. The reward signal encourages efficiency and accuracy: +1 for each box on target, −1 for off-target boxes, +10 upon task completion, and −0.1 per action.

Frozen Lake. This environment (Figure 4b) combines long-horizon decision-making with stochastic transitions. The agent navigates a grid with slippery tiles; each action succeeds with probability 1/3 and deviates perpendicularly with probability 2/3. The agent should reach the goal without falling into holes. Rewards are sparse: successful trials receive a reward of +1, with all others 0.

B.2. Training and Evaluation Settings

We conduct our experiments using Qwen2.5-0.5B-Instruct (Yang et al., 2024), trained via the StarPO variants with a maximum of 200 rollout–update iterations on NVIDIA H100/A100 GPUs leveraging the veRL [*] repository. Each rollout consists of $K = 16$ trajectories per environment group, based on prompt size $P = 8$ and maximum 5 interaction turns per episode. Agents are allowed up to 5 actions per turn and 10 actions per episode. The update batch size is $E = 32$, with mini-batch size 4 per GPU. Policy optimization uses GAE with $(\gamma = 1.0, \lambda = 1.0)$ and Adam with $(\beta_1, \beta_2) = (0.9, 0.999)$. We use entropy regularization ($\beta = 0.001$) For experiments with vanilla StarPO we use a KL coefficient of 0.001, using the k1 estimation[†]. without KL loss term during training, following (Yu et al., 2025), and track KL post-hoc. We impose a format penalty of −0.1 if the agent fails to output valid structured responses (e.g., missing `<think>` or `<answer>` tags), encouraging adherence to response conventions. To accelerate rollout generation, we disable enforce_eager and retain the computation graph across prefill and sampling in vLLM. We utilize Fully Sharded Data Parallel (FSDP) training strategy for multi-GPU experiments.

[*] https://github.com/volcengine/verl
[†] http://joschu.net/blog/kl-approx.html

For distributed training, we employ Ray as the multi-processing backend with XFORMERS attention implementation.

For evaluation, we choose a fixed 256 input prompts per environment and decode using temperature T=0.5, sampling stochastically to better capture robustness in agent behaviors. Episode truncation occurs after 5 turns or 10 total actions.

B.3. Evaluation Metrics

To track agent learning dynamics and detect training instabilities, we monitor the following metrics throughout training. Except for the success rate, which is evaluated on a fixed validation set, all metrics are computed over on-policy rollouts collected during training and smoothed using exponential moving average (EMA).

- **Average Success Rate.** Measures task completion accuracy on a fixed set of validation prompts. An episode is considered successful if the agent solves the task (e.g., pulling the high-reward arm in Bandit, pushing all boxes to targets in Sokoban, or reaching the goal in Frozen Lake).
- **Rollout Entropy.** Computes the average token-level entropy of sampled responses, capturing the exploration level and policy uncertainty. A sharp entropy drop may indicate premature policy convergence or collapse.
- **In-Group Reward Variance.** Measures reward standard deviation across rollouts sampled from the same prompt group. High in-group variance reflects diverse behaviors and learning potential; a sudden collapse indicates reward homogenization and policy stagnation.
- **Total Response Length.** Average number of tokens generated per rollout, measuring the verbosity and reasoning depth of the agent. Fluctuations in length may signal changes in planning style or confidence.
- **Gradient Norm.** ℓ_2 norm of the policy gradient vector, used as a proxy for training stability. Spikes often correlate with phase transitions in policy behavior or unstable reward signals.

These metrics provide complementary views of policy quality, update dynamics, and reasoning behavior, helping diagnose when and why agent training succeeds or fails.

C. Comparing agent RL with Supervised Fine-Tuning

Apart from StarPO for RL training, we also employ Supervised Fine-tuning (SFT) as another agent training approach, evaluating it on the Sokoban and Frozen Lake task. We employ LoRA with a rank of 64 and an alpha value of 32, targeting all linear layers in the model. The SFT process uses a learning rate of 1e-4 with a training batch size of 128. We generate ground-truth trajectory data through breadth-first search (BFS), setting a maximum depth of 100 to create 1,000 training samples and 100 test samples. For SFT, we structure the multi-turn interaction as a conversational format. At each turn, the model must generate the next action from the ground-truth trajectory, encapsulating its response within <answer> </answer> tags to maintain format consistency.

We analyze the comparative performance of SFT against our stable RL baseline StarPO-S. SFT achieves 74.6% and 23% performance on Sokoban and Frozen Lake, respectively, Compared to the 20.3% and 21.8% performance with StarPO-S. The results indicate that SFT demonstrates

superior performance to RL approaches. We draw conclusions from the results that although rule-based RL show promise for agent tasks, there is still a need to build more scalable and effective agent RL algorithms to achieve human-comparable performance with solely model self-evolution.

D. Efficient Training

Motivation. While the main body of the paper reports results obtained by full-parameter fine–tuning, in practice such a setting may be prohibitive when scaling to larger models or longer-horizon tasks. We therefore implement a parameter-efficient variant of RAGEN based on Low-Rank Adaptation (Hu et al., 2021).[‡]

Performance parity. Despite updating only a fraction of the model parameters, LoRA reaches a validation success rate comparable to that achieved by full fine-tuning of the entire network for the SimpleSokoban task, achieving approximately a 0.2% success rate on the validation set.

Resource savings. We compare the hardware footprint of LoRA with full fine-tuning. Across an 80-minute training horizon we measure:

- **GPU memory.** LoRA stabilizes at ≈ **23**% of device memory versus ≈ **48**% for full updates, cutting the peak allocation by >50 %.
- **GPU utilization.** Average GPU utilization drops from ∼34% to ∼14%.
- **Power consumption.** Mean power draw decreases from ∼22% to ∼12%, a ≈ 45% reduction.

Take-aways. Parameter-efficient fine-tuning provides a practically viable alternative for RAGEN: it attains comparable policy quality while more than halving memory, compute, and power demands. Consequently, future work that scales StarPO to larger backbones or longer contexts can adopt LoRA (or other adapter-based methods) as the default optimization strategy without re-engineering the training loop.

E. Prompt Templates

E.1. Bi-Arm Bandit Environment Prompts

The Bi-Arm bandit environment implements a classic reinforcement learning problem where an agent must balance exploration and exploitation. We present the prompt templates below.

Model Templates

```
<|im_start|>[system]:
{prompt}
You're a helpful assistant.  You always respond by giving your answer in <answer>...</answer>.
Max response length:  200 words (tokens).
<|im_end|>
<|im_start|>[user]:
{prompt}
You are playing a bandit game.  Goal:  Maximize your total reward by choosing which arm to
pull.
Game Rules:
1.  There are 2 arms, named name_a and name_b
2.  Each arm has its own reward distribution, related to their names.
3.  Analyze the symbolic meaning of each arm's name to guess how their reward distribution
might behave.
```

[‡]We set rank r=64, α=64, and inject adapters into all linear projections of the transformer blocks. We also increased learning rate by 10× for both actor and critic.

```
4.  Based on the symbolic meaning of their names, which arm do you think is more likely to
give higher rewards on average?  Choose between name_a and name_b, and output like <answer>
name_a </answer> or <answer> name_b </answer>.
<|im_end|>
<|im_start|>assistant
<think>
```

E.2. Sokoban Environment Prompts

The Sokoban environment presents a classic puzzle game where an agent must push boxes to target locations. The following sections detail the prompt structure used to interface with language models.

Model Templates

```
<|im_start|>system
{prompt}
You're a helpful assistant.  You always respond by first wrapping your thoughts in
<think>...</think>, then giving your answer in <answer>...</answer>.  Max response length:
200 words (tokens).
<|im_end|>
<|im_start|>user
{prompt}
You are solving the Sokoban puzzle.  You are the player and you need to push all boxes to
targets.  When you are right next to a box, you can push it by moving in the same direction.
You cannot push a box through a wall, and you cannot pull a box.  The answer should be a
sequence of actions, like <answer>Right || Right || Up</answer>
<|im_end|>
<|im_start|>assistant
<think>
```

The environment uses a grid-based representation with specific symbols for different elements:

Grid Representation

```
 The meaning of each symbol in the state is:
#:  wall, _:  empty, O: target, √:  box on target, X: box, P: player, S: player on target
```

The instruction template only consists of available actions and restrictions:

Instruction Template

```
Your available actions are:
Up, Down, Left, Right
You can make up to 10 actions, separated by the action separator " || "
```

E.3. FrozenLake Environment Prompts

The FrozenLake environment implements a grid-world navigation task where an agent must traverse a slippery frozen surface to reach a goal. Below we detail the prompt structure used for this environment.

Model Templates

```
<|im_start|>system
{prompt}
You're a helpful assistant.  You always respond by first wrapping your thoughts in
```

```
<think>...</think>, then giving your answer in <answer>...</answer>.  Max response length:
200 words (tokens).
<|im_end|>
<|im_start|>user
{prompt}
You are solving the FrozenLake puzzle.  Forbid the whole and go to the target.  You may move
to the unintended direction due to the slippery ice.  Example answer format:  <think>To
forbid the hole and go to the target, I should go left then go up.</think><answer>Left ||
Up</answer>
<|im_end|>
<|im_start|>assistant
<think>
```

The environment uses a grid-based representation with specific symbols for different elements:

Grid Representation

```
 The meaning of each symbol in the state is:
P: player, _:  empty, O: hole, G: goal, X: player in hole, ✓:  player on goal
```

The instruction template only consists of available actions and restrictions:

Instruction Template

```
Your available actions are:
Left, Down, Right, Up
You can make up to 10 actions, separated by the action separator " || "
```

F. PPO Failure Mode in Frozen Lake

Among the three evaluated environments, we observe an interesting divergence on Frozen Lake: PPO tends to collapse earlier or converge less stably than GRPO. This contrasts with the general trend where PPO demonstrates better performance, prompting further analysis.

One possible explanation lies in the environment's long-horizon stochasticity. In Frozen Lake, agent actions always lead to highly non-deterministic transitions, and intermediate states can appear similar while leading to very different outcomes. This makes value estimation challenging. As PPO relies on a learned value function, instability in critic learning may amplify optimization noise and contribute to early collapse. GRPO, by contrast, does not rely on explicit value learning. Its reward-weighted update procedure may be more tolerant to uncertainty in these settings, leading to comparatively more stable training on Frozen Lake—even if it remains less effective in other tasks. Overall, we summarize environments with high stochasticity may pose greater challenges for value-based methods, and that critic-free approaches can serve as a useful baseline in such cases.

Frozen Lake Insight: Critic-free methods may offer robustness under uncertainty

On Frozen Lake, PPO underperforms GRPO, potentially due to value estimation difficulties in a highly stochastic and sparse-reward setting. GRPO's critic-free structure may offer greater tolerance to such conditions.

G. Generalization Evaluation Environments

To evaluate generalization beyond the training distribution, we design two new test environments besides the three training environments that vary along different axes:

- **SokobanDifferentGridVocab** modifies the visual vocabulary used to represent the grid. Instead of using the standard symbols (#, _, O, X, etc.), it maps grid cells to a new vocabulary such as `W`, `G`, `C`, etc. This tests whether the model generalizes across symbol variations while retaining underlying spatial semantics.
- **LargerSokoban** increases the grid size from 6×6 to 8×8 and the number of boxes from 1 to 2, introducing greater spatial complexity and longer-horizon planning demands. This setting evaluates whether the policy trained on small puzzles can scale up to more complex configurations.

These environments are not seen during training and serve to probe the agent's generalization capability under symbol shift, size scaling, and environment shift, respectively.