RESOLVING THE ACTION BOTTLENECK: AGENTIC REINFORCEMENT LEARNING INFORMED BY TOKEN-LEVEL ENERGY

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic reinforcement learning trains large language models using multi-turn trajectories that interleave long reasoning traces with short environment-facing actions. Common policy-gradient methods, such as PPO and GRPO, treat each token in a trajectory equally, leading to uniform credit assignment. In this paper, we critically demonstrate that such uniform credit assignment largely misallocates token-level training signals. From an energy-based modeling perspective, we show that token-level training signals, quantified by their correlations with reward variance of different rollouts sampled from a given prompt, concentrate sharply on action tokens rather than reasoning tokens, even though action tokens account for only a small fraction of the trajectory. We refer to this phenomenon as the Action Bottleneck. Motivated by this observation, we propose an embarrassingly simple token reweighting approach, ACTFOCUS, that downweights gradients on reasoning tokens, along with an additional energy-based redistribution mechanism that further increases the weights on action tokens with higher uncertainty. Across four environments and different model sizes, ACTFOCUS improves all reported PPO final-step results and four of five reported GRPO final-step results, yielding final-step gains of up to 21.5 and 63.7 percentage points, respectively, with negligible additional runtime or memory cost.

1 INTRODUCTION

As large language models (LLMs) are increasingly deployed as autonomous agents that combine reasoning (Guo et al., 2025; Jaech et al., 2024), planning (Zhou et al., 2023), and tool use (Wölflein et al., 2025) to solve multi-turn, long-horizon tasks, reinforcement learning (RL) has emerged as the natural post-training framework for sequential decision-making settings (Wang et al., 2025; 2024; Xi et al., 2025; Abdulhai et al., 2023). Despite rapid progress with different policy optimization methods (Schulman et al., 2017; Shao et al., 2024), training LLM agents with RL remains difficult. One of the core challenges is credit assignment (Pignatelli et al., 2024; Kazemnejad et al., 2025), where, in multi-turn trajectories, a sparse episodic reward must be propagated back through thousands of generated tokens, making it difficult to determine which parts of a trajectory should drive learning. The problem is compounded by optimization instability (Cui et al., 2025b), as delayed feedback, long generated traces, and noisy policy updates often lead to degraded late-stage performance or collapse.

Recent evidence shows that multi-turn, underspecified conversations substantially increase outcome variability for LLMs, sharpening the need for reliable credit assignment in agentic RL (Laban et al., 2025). In a typical agentic RL rollout, such outcome variability often hinges on a few critical environment-facing action tokens that drive state transitions, yet these tokens are buried within massive amounts of reasoning tokens in the trajectory (Yao et al., 2023b; Schick et al., 2023; Wen et al., 2024). Even though these action tokens are the only parts of the output that directly interact with the environment, standard policy optimization methods treat all generated tokens uniformly (Schulman et al., 2017; Liu et al., 2025b). Consequently, gradient updates are spread equally across the whole trajectory, diluting training signals for the critical action tokens that trigger the decision-making (Chan et al., 2024; Vassoyan et al., 2025). In multi-turn settings, this mismatch compounds with each additional turn: reasoning tokens keep accumulating while the environment is driven by only a few informative actions.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

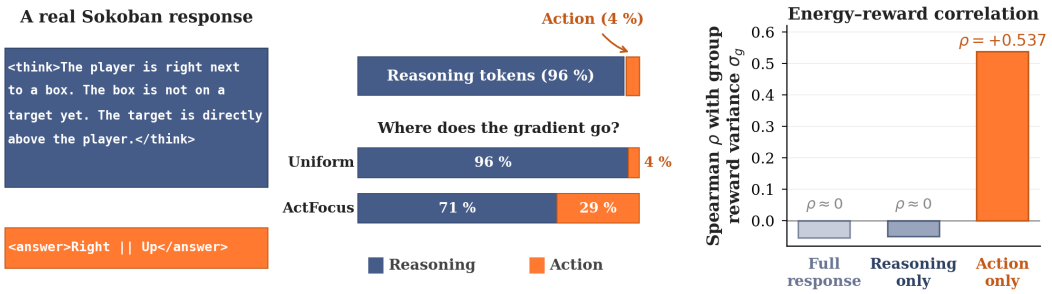


Figure 1: **Action Bottleneck in agentic reinforcement learning.** *Left:* A real response from a trajectory, where reasoning tokens far outnumber action tokens. *Middle:* Action tokens constitute only 4% of model-generated tokens; our token-level reweighting redirects gradient mass towards them. *Right:* Training signal concentrates in action spans. Results are from Sokoban 3B.

Figure 1 shows that this structural mismatch manifests as token-level signal sparsity in agentic RL. In a representative Sokoban trajectory, action tokens constitute only 4% of the generated output, while reasoning tokens account for the remaining 96%. Under uniform token averaging, the mean gradients of the reasoning and action spans are weighted in proportion to their token counts, giving sparse action spans a much smaller coefficient. However, in this paper, we demonstrate that the training signals follow the opposite pattern. We quantify each token's training signal as its predictive uncertainty under a frozen reference model, measured by the free energy of its next-token predictive distribution.[1] To further evaluate how informative the signal is for training, we hinge it on outcome variability: for each prompt, we calculate the variance of episodic rewards σ_g across multiple rollouts (sampled from the same prompt), which captures how uncertain the current policy is about the prompt's outcome. We find that the mean energy of action tokens is strongly correlated with reward variance (with $\rho = +0.537$), while the mean energy of reasoning tokens or full trajectories, which average over both action and reasoning tokens, is nearly uninformative (with $\rho \approx 0$). **This suggests that informative training signals are concentrated in a small number of action tokens rather than being distributed uniformly across the trajectory.** We refer to this phenomenon as the *Action Bottleneck*.

Motivated by this observation, we propose an energy-based approach to token-level credit assignment in agentic RL. The method is conceptually intuitive and can be decomposed into two aspects: 1) We downweight gradients on reasoning tokens, which is equivalent to increasing the weights on action tokens and helps to redirect gradient mass toward the sparse action span that determines environmental outcomes; 2) We utilize token-level energy to further redistribute the weights across action tokens, which prioritizes action tokens with high energy (i.e., more predictive uncertainty). In this way, we transform our empirical observations into an actionable approach that systematically guides the model to better leverage the informative training signals underlying a few critical tokens in the long trajectory.

We evaluate our approach, ACTFOCUS, on four representative multi-turn agentic environments using Qwen2.5-Instruct (Qwen et al., 2025) models at multiple scales, and train each configuration with both PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024). Through these experiments, we find that ACTFOCUS improves all reported PPO final-step results and four of five reported larger-scale GRPO results, with gains of up to 21.5 and 63.7 percentage points, respectively. Beyond absolute performance metrics, we observe that ACTFOCUS effectively improves GRPO's training stability by substantially reducing peak-to-final degradation. Our main contributions are summarized as follows:

(i) We analyze token-level attributions in agentic RL training through an energy-based modeling perspective and reveal a counter-intuitive phenomenon: while reasoning tokens dominate in long-horizon multi-turn trajectories, infrequent action tokens contribute the majority of informative training signals. In particular, the uncertainty in action tokens, as measured by the free energy of next-token predictive distributions, is highly correlated with outcome variability.

[1] An ideal indicator for quantifying training signals should reveal where outcome uncertainty concentrates, stay stable across training, and be computationally inexpensive. Energy, used in this work, satisfies all of these criteria; alternatives such as entropy and log-likelihood are compared empirically in Sec. 5.6.

(ii) We propose a token reweighting mechanism that downweights gradients on reasoning tokens and additionally redistributes the weights on action tokens according to their uncertainty levels as characterized by the free energy. This simple scheme shifts gradient mass towards the most informative action tokens while remaining fully compatible with common policy-gradient algorithms.

(iii) Through extensive experiments across four environments and multiple model scales, ACTFOCUS improves all reported PPO final-step results and four of five reported larger-scale GRPO results, with average gains of 11.6 and 33.8 percentage points and final-step gains of up to 21.5 and 63.7 percentage points, respectively. In particular, ACTFOCUS can significantly enhance the training stability of GRPO by mitigating the peak-to-final degradation observed under uniform credit assignment.

2 RELATED WORK

Agentic Reinforcement Learning for LLMs. Reinforcement learning has become a central paradigm for aligning large language models with sequential decision-making objectives beyond single-turn text generation (Ouyang et al., 2022). Recent work extends RL to multi-turn agent settings from different angles. For instance, LOOP (Chen et al., 2025) formulates interactive digital agents as RL in stateful environments, while RAGEN (Wang et al., 2025) presents a general framework for multi-turn agent training under stochastic environments. ARLArena (Wang et al., 2026) systematizes the study of training stability by decomposing policy-gradient design into core dimensions. However, despite these advances, credit assignment in existing agentic RL methods remains relatively under-explored and does not explicitly distinguish between different parts of an agent trajectory, particularly reasoning and action tokens.

Credit Assignment in Agentic RL. A growing line of work argues that uniform credit assignment largely under-represents the token-level training signals of long-horizon agent trajectories. Most existing efforts operate at the step or trajectory level. CARL (Shen et al., 2026) uses policy entropy as a proxy for action importance and concentrates optimization on critical actions across turns. GiGPO (Feng et al., 2025) introduces a two-level advantage estimator based on anchor states that reappear across trajectories, enabling step-level relative credit assignment. iStar (Liu et al., 2025a) learns implicit step-wise rewards through trajectory-level preference optimization and combines step-level and trajectory-level signals in policy updates. These methods recognize that credit should vary across an agent trajectory, but they remain coarse in granularity and do not distinguish reasoning tokens from action tokens within each step. Several works in reasoning RL have explored finer-grained reward redistribution and intermediate supervision (Tan et al., 2026; Cui et al., 2025a; Ma et al., 2026), but these approaches are designed for single-turn reasoning without diving into the reasoning-action disentanglement in multi-turn agent trajectories.

3 PRELIMINARIES

3.1 MULTI-TURN AGENTIC RL

We formulate multi-turn agentic RL as a Markov Decision Process $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R \rangle$ (Sutton et al., 1998), where states $s \in \mathcal{S}$ are interaction histories, actions $a \in \mathcal{A}$ are environment-facing operations, the transition $P(s_{k+1} \mid s_k, a_k)$ specifies how the environment responds to an action, and the reward $R(\tau)$ assigns a scalar episodic return based on the terminal state of the trajectory. At each turn $k \in \{1, \ldots, K\}$, the policy π_θ observes the current state s_k and generates a structured response:

$$o_k = \texttt{<think>}\ldots\texttt{</think>}\ \texttt{<answer>}\ a_k\ \texttt{</answer>}, \tag{1}$$

where the `<think>` block contains the agent's reasoning and the `<answer>` block contains the action a_k. The environment extracts a_k from o_k, transitions to the next state s_{k+1}, and returns a reward $R(\tau)$ at episode termination. The full trajectory is $\tau = (s_1, o_1, s_2, o_2, \ldots, s_K, o_K)$, and the RL objective is to maximize expected episodic reward: $J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$.

Token-level policy optimization. Each response o_k consists of tokens $(y_{k,1}, \ldots, y_{k,T_k})$ from both the reasoning and action segments. For notational simplicity, we drop the turn index and use y_t for a generic response token. Both PPO and GRPO optimize this objective via a clipped token-level

Table 1: **Token composition before RL training.** Each row reports the two model scales shown in the second column.

Environment	Scale	Tokens	Think (%)	Action (%)
Sokoban	0.5B / 3B	225 / 202	90.5 / 96.1	9.5 / 3.9
FrozenLake	0.5B / 3B	284 / 240	95.9 / 97.4	4.1 / 2.6
Sudoku	0.5B / 1.5B	400 / 397	84.7 / 85.8	15.3 / 14.2
WebShop (Yao et al., 2023a)	1.5B / 3B	1,951 / 2,133	89.2 / 92.7	10.8 / 7.3

surrogate loss that shares the same general form:

$$\mathcal{L}(\theta) = -\frac{1}{|\mathcal{D}|} \sum_{(y_t, \hat{A}_t) \in \mathcal{D}} w_t \cdot \min\Big(\rho_t(\theta)\hat{A}_t,\ \mathrm{clip}\big(\rho_t(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_t\Big), \tag{2}$$

where $\rho_t(\theta) = \pi_\theta(y_t \mid c_t)/\pi_{\theta_{\text{old}}}(y_t \mid c_t)$ is the importance ratio between the current and old policy, ϵ is the clipping threshold, and $\hat{A}_t$ is the per-token advantage. Here, c_t denotes all tokens preceding y_t, to distinguish it from the environment state s_k, and w_t is a per-token weight.

PPO estimates $\hat{A}_t$ via Generalized Advantage Estimation (GAE) (Schulman et al., 2018) with a learned value function, providing a distinct advantage for each token position. GRPO eliminates the critic and instead computes a single trajectory-level advantage:

$$\hat{A}_\tau = \frac{R(\tau) - \mathrm{mean}(R(\tau))}{\mathrm{std}(R(\tau))}, \tag{3}$$

where the mean and standard deviation are computed over all trajectories sampled from the same prompt. Under GRPO, every token in a trajectory shares the same advantage: $\hat{A}_t = \hat{A}_\tau$ for all t in τ. In both algorithms, the common practice is to set $w_t = 1$ for all response tokens, which corresponds to the uniform credit assignment that we critically challenge in this paper. Complete descriptions for algorithm-specific objectives and implementation details are deferred to Appendix B.

3.2 Frozen-Reference Token Energy

Recent work (Grathwohl et al., 2020; Xie et al., 2016) establishes an equivalence between neural classifiers and energy-based models (EBMs). Given a classifier that maps inputs to C-dimensional logits $h(x) \in \mathbb{R}^C$, the classifier-EBM equivalence is established by setting the energy of an input-label pair as $E(x, y) = -h(x)[y]$. Marginalizing over labels yields a free energy $E(x) = -\log \sum_c \exp(h(x)[c])$, a scalar score derived from the model's unnormalized logits (Liu et al., 2020).

We extend this idea to autoregressive language models, which predict a distribution over the vocabulary at each position. At each position t, a frozen reference model produces next-token logits $f_t^{\text{ref}} \in \mathbb{R}^{|V|}$ over vocabulary V, given the preceding context. We define the **token-level energy** as:

$$E_t = -\log \sum_{v \in V} \exp\big(f_{t,v}^{\text{ref}}\big). \tag{4}$$

A lower E_t corresponds to higher confidence under the reference model, while a higher E_t indicates greater uncertainty about the next token. This signal suits agentic RL diagnosis for two reasons. First, because the reference model is frozen, E_t remains stable throughout training, unlike policy-derived quantities such as entropy or confidence that drift as the policy evolves. Second, computing E_t requires only a single forward pass, making it cheaper than Monte Carlo estimates that require repeated rollouts (Shen et al., 2026). We use token-level energy to locate where the reward-predictive signal concentrates within agent trajectories; this analysis motivates the intervention in Sec. 4.

4 Token Credit Assignment

We now turn the Action Bottleneck diagnosis into a token-level weighting mechanism that aims to address two issues: 1) reasoning tokens dominate the trajectory, and 2) training signals are not

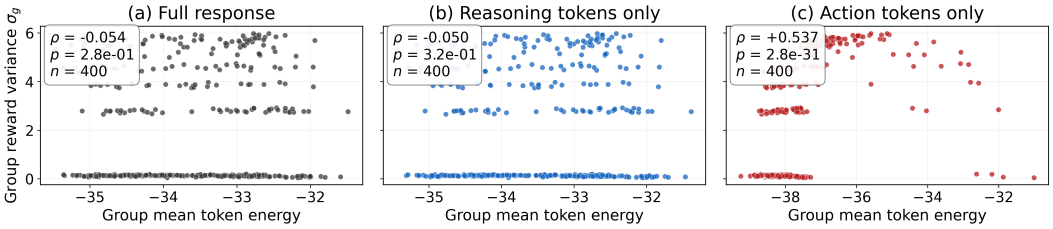


Figure 2: **Detailed energy–reward diagnostic behind Figure 1.** We measure the Spearman correlation between mean frozen-reference token energy and group reward variance σ_g on Sokoban 3B over three token subsets. Only the action-token energy shows a significant positive correlation ($\rho = +0.537$, $p < 10^{-30}$); full-response and reasoning-only energy are indistinguishable from noise.

uniformly distributed across action tokens. Table 1 reveals the imbalance: across environments and model scales, action tokens account for less than 16% of generated tokens in a trajectory. Given this, uniform credit assignment would concentrate most gradient mass on reasoning spans.

To further validate the results in Figure 1 across different cases, we examine where the training signals live within a trajectory. For each prompt in the training set, we run G independent rollouts under the current policy, each producing a complete multi-turn trajectory and its episodic reward $R(\tau)$. We refer to the G trajectories sampled from the same prompt as a *trajectory group*, with group reward variance $\sigma_g = \mathrm{Var}(R(\tau^{(1)}), \ldots, R(\tau^{(G)}))$. A high σ_g means the policy sometimes succeeds and sometimes fails on this prompt, so the prompt provides useful learning signals. For each trajectory group, we then compute the mean frozen-reference token energy over three token subsets: the full response, reasoning tokens only, and action tokens only. We measure the Spearman correlation between the mean energy of each subset and σ_g across all trajectory groups. Figure 2 shows the result on Sokoban 3B. We find that action-only energy correlates with σ_g, with $\rho = +0.537$ and $p < 10^{-30}$, while full-response and reasoning-only energy are indistinguishable from random noise. This shows that the training signals captured by frozen-reference energy largely stem from action tokens instead of reasoning tokens that dominate in the trajectory.

The observation motivates a token-level reweighting scheme that proceeds in two steps. Specifically, we first use the model-generated `<think>`/`<answer>` boundary to reallocate gradient mass from reasoning tokens towards environment-facing action tokens. We then use frozen-reference energy within the action span to prioritize positions where the reference model is more uncertain.

4.1 Reasoning Downweighting and Energy-Based Action Prioritization

For each response token belonging to either a reasoning span $\mathcal{T}_{\text{think}}$ or an action span $\mathcal{T}_{\text{action}}$, as identified by the `<think>` and `<answer>` tags, we assign a weight:

$$w_t = \begin{cases} \alpha, & t \in \mathcal{T}_{\text{think}}, \\ 1 + \beta \ \mathrm{sigmoid}\left(\dfrac{E_t - \mu_E}{\sigma_E}\right), & t \in \mathcal{T}_{\text{action}}, \end{cases} \tag{5}$$

where $\alpha \in [0, 1]$ controls how much gradient is retained on reasoning tokens, $\beta \geq 0$ controls how strongly energy modulates the weights on action tokens, and E_t is the frozen-reference token energy from Eq. 4. μ_E and σ_E denote the mean and standard deviation of E_t over all action tokens in the current training batch, respectively. The sigmoid function is applied to a batch-normalized action-token energy score, making energy modulation depend on relative uncertainty within the action-token distribution rather than on the raw energy scale.

When setting $\alpha = 1$ and $\beta = 0$, which gives $w_t = 1$ for all tokens, the reweighting mechanism described by Eq. 5 reduces to standard uniform credit assignment. Our proposed approach uses $\alpha < 1$ and $\beta > 0$, which simultaneously downweights reasoning tokens and applies energy-based redistribution within the action span. The former simply shifts the gradient mass towards informative action tokens, while the latter additionally prioritizes the action tokens for which the model is more uncertain. In our experiments, we compare standard UNIFORM credit assignment against our proposed token reweighting approach, which we refer to as ACTFOCUS.

Table 2: PPO final-step results across four environments at two capability levels. Success rate (%) is reported at step 200 for Sokoban, FrozenLake, and Sudoku, and at step 100 for WebShop. WebShop reports both purchase rate and strict success. Peak values are reported in Table 5. Best results are **bolded**. Results are reported as mean ± sample standard deviation over five random seeds.

Method	Sokoban	FrozenLake	Sudoku	$\text{WebShop}_{\text{purchase}}$	$\text{WebShop}_{\text{strict}}$
	3B	3B	1.5B	3B	3B
Uniform	26.2 ± 2.8	16.6 ± 9.9	81.2 ± 3.3	96.9 ± 2.9	22.7 ± 0.2
ActFocus	**37.5** ± 1.9 (+11.3)	**27.0** ± 6.4 (+10.4)	**95.7** ± 1.8 (+14.5)	**99.2** ± 0.6 (+2.3)	**36.7** ± 3.8 (+14.0)
	0.5B	0.5B	0.5B	1.5B	1.5B
Uniform	12.5 ± 3.9	5.1 ± 1.7	68.9 ± 9.9	94.9 ± 10.9	9.8 ± 0.8
ActFocus	**34.0** ± 2.6 (+21.5)	**15.6** ± 4.0 (+10.5)	**80.1** ± 5.3 (+11.2)	**98.2** ± 1.4 (+3.3)	**26.6** ± 2.1 (+16.8)

4.2 Modified Objective

We incorporate token-level credits by replacing the average in Eq. 2 with a weighted sum:

$$\mathcal{L}(\theta) = -\frac{1}{\sum_t w_t} \sum_t w_t \, \min\Big(\rho_t(\theta)\,\hat{A}_t,\ \text{clip}\big(\rho_t(\theta),\, 1-\epsilon,\, 1+\epsilon\big)\,\hat{A}_t\Big), \tag{6}$$

where w_t is given by Eq. 5. Tokens with larger w_t contribute more to the policy-gradient update, while tokens with smaller w_t contribute less. Importantly, this modification does not change how the advantage is computed. For PPO, $\hat{A}_t$ still comes from the original critic-based estimator; for GRPO, it still comes from the group-normalized trajectory return. Thus, the method only changes how the existing advantage signal is distributed across tokens.

We normalize by $\sum_t w_t$ so that changing α or β does not simply make the whole loss larger or smaller. Instead, α and β control where the gradient mass goes: decreasing α reduces the contribution of reasoning tokens, while increasing β gives more weight to high-energy action tokens.

Ignoring PPO clipping for clarity, the weighted objective separates the update into a reasoning component and an action component:

$$\nabla_\theta \mathcal{L} \;=\; -\frac{1}{\sum_t w_t}\left[\underbrace{\alpha \sum_{t\in\mathcal{T}_{\text{think}}} \hat{A}_t \nabla_\theta \log \pi_\theta(y_t \mid c_t)}_{\text{reasoning component}} + \underbrace{\sum_{t\in\mathcal{T}_{\text{action}}} w_t \hat{A}_t \nabla_\theta \log \pi_\theta(y_t \mid c_t)}_{\text{action component}}\right]. \tag{7}$$

The two terms correspond directly to the two steps introduced at the start of this section. Reducing α reallocates gradient mass from reasoning spans to action spans. Positive β then redistributes mass within the action span, giving larger weight to high-energy tokens where the frozen reference model is more uncertain.

Detailed derivations of the gradient-allocation mechanism are provided in Appendix B.1.

5 Experiments

We evaluate our proposed method on four multi-turn environments spanning distinct reasoning tasks: **Sokoban**, a spatial planning task where the agent pushes boxes onto targets on a grid; **FrozenLake**, a navigation task on a stochastic slippery surface; **Sudoku 4×4**, a constraint-satisfaction task requiring logical deduction; and **WebShop** (Yao et al., 2023a), an e-commerce task involving product search, evaluation, and purchase. For WebShop, we report both purchase rate and strict success: purchase rate measures whether the agent completes a purchase, while strict success additionally requires the purchased item to match the user instruction. We use the **Qwen2.5-Instruct family** (Qwen et al., 2025) throughout all experiments. For each environment, we use two model scales to test whether the Action Bottleneck and reweighting benefits persist across different base-model capabilities. We train each model with both PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024), built on veRL

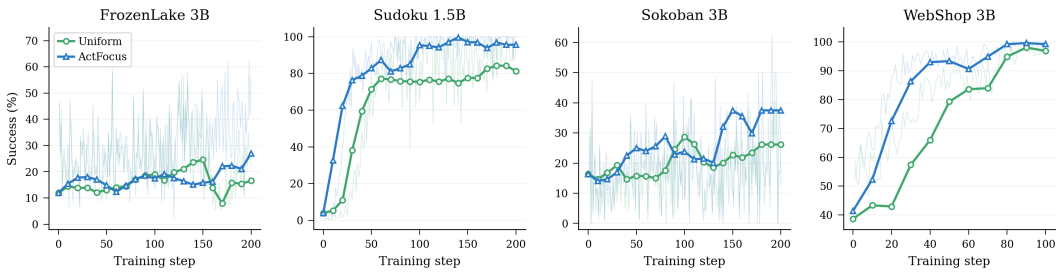


Figure 3: PPO success rate across training steps on FrozenLake 3B, Sudoku 1.5B, Sokoban 3B, and WebShop 3B. Shaded curves denote per-step training success.

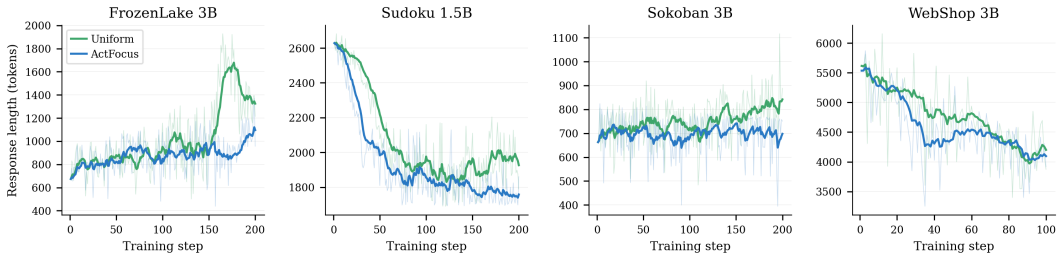


Figure 4: PPO response length across training steps on FrozenLake 3B, Sudoku 1.5B, Sokoban 3B, and WebShop 3B. Shaded curves denote per-step response length.

following StarPO-S configuration (Wang et al., 2025). We compare two token-weighting schemes: UNIFORM applies standard equal credit and ACTFOCUS applies the full structure-aware weighting rule in Eq. 5. Detailed dataset specifications, rollout settings, and training hyperparameters are provided in Appendix C. We further evaluate Qwen2.5-7B-Instruct and test alternative action-weight assignments in Appendix D.

5.1 MAIN RESULTS

All main results in Table 2 use a single fixed hyperparameter setting across all four environments and both model scales. We deliberately avoid per-environment tuning to test whether the Action Bottleneck correction generalizes without configuration-specific calibration. Ablations on the choice of α and β are deferred to Sec. 5.4 and Sec. 5.5.

ActFocus improves across all PPO settings. Table 2 shows that ACTFOCUS outperforms UNIFORM in every tested configuration, across four diverse environments and two model scales. The consistent improvement suggests that the gains stem from correcting a shared structural imbalance in agent trajectories rather than fitting to any specific task or model scale.

ActFocus helps regardless of whether Uniform is learning. On Sokoban 3B, ACTFOCUS raises success from 26.2% to 37.5%, and the gain is similar in size on FrozenLake 3B (16.6% to 27.0%) and Sudoku 1.5B (81.2% to 95.7%), where UNIFORM already learns nontrivial policies. When UNIFORM is weak, the benefit persists: ACTFOCUS raises success from 12.5% to 34.0% on Sokoban 0.5B and from 5.1% to 15.6% on FrozenLake 0.5B. These results suggest that uniform credit suffers from the same structural problem across regimes: it spreads the learning signal over many reasoning tokens, leaving too little credit for the action tokens that determine the outcome.

ActFocus improves decision quality beyond task completion. On WebShop, UNIFORM already achieves a high purchase rate, so there is limited room to improve the purchase-rate metric. The more informative signal is strict success, where ACTFOCUS improves from 22.7% to 36.7% on WebShop 3B and from 9.8% to 26.6% on WebShop 1.5B. The strict success gain shows that the method improves the quality of the selected product, not merely the ability to finish an episode. This matches the role of frozen-reference energy in Sec. 4: energy identifies uncertain action positions, and ACTFOCUS assigns them larger credit during training.

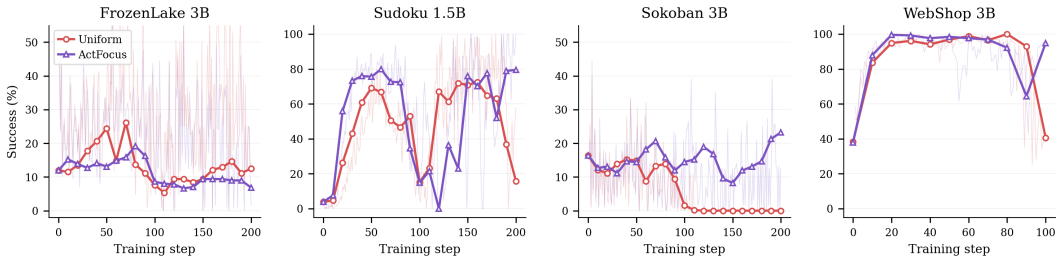


Figure 5: GRPO success rate across training steps on FrozenLake 3B, Sudoku 1.5B, Sokoban 3B, and WebShop 3B. Shaded curves denote per-step training success.

5.2 Training Dynamics and Response Efficiency

ActFocus also improves training dynamics. Figure 3 shows that ActFocus improves PPO training dynamics across all four configurations, but the form of improvement differs by environment. On Sudoku 1.5B and WebShop 3B, ActFocus reaches high success earlier than Uniform, indicating faster learning. On Sokoban 3B, the two methods are closer in the early stages, but ActFocus continues to improve in later steps, while Uniform improves more slowly, so the performance gap widens over training. FrozenLake 3B is noisier because of stochastic transitions, but ActFocus still reaches a better final result. Overall, these curves show that the gains in Table 2 are reflected in the training process, rather than appearing only at the final checkpoint.

Gains are not caused by longer responses. Figure 4 shows that ActFocus produces shorter or more stable responses than Uniform across all configurations. On FrozenLake 3B and Sokoban 3B, Uniform response length grows throughout training while ActFocus keeps it controlled. On Sudoku 1.5B and WebShop 3B, both methods shorten responses over training, but ActFocus does so earlier and converges to shorter final lengths. Thus, ActFocus improves success without relying on longer reasoning traces, as better credit assignment directs the policy toward action tokens without requiring excessive verbalization.

5.3 Extension to GRPO

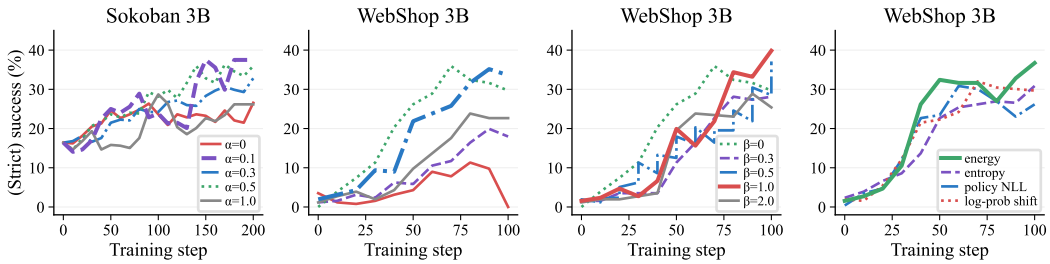
We also evaluate whether the same token-weighting rule transfers to GRPO. Under GRPO, the critic is removed, and a single trajectory-level advantage is assigned to every token in the same rollout, making token weights the primary source of within-trajectory credit differentiation. This makes GRPO especially sensitive to the Action Bottleneck: under uniform credit, reasoning tokens dominate the update even though outcomes are mostly determined by sparse action decisions.

Figure 5 shows that this mismatch often appears as peak-to-final degradation. Across Sokoban 3B, Sudoku 1.5B, and WebShop 3B, Uniform often reaches a useful intermediate policy but fails to preserve it, leading to sharp peak-to-final degradation. ActFocus either prevents the late-stage drop or helps the policy recover from it. Final-step and peak GRPO results are given in Tables 6 and 5 (Appendix D). Response-length curves under GRPO, which track stability rather than efficiency, are shown in Appendix D.3.

5.4 How Much Reasoning Credit Should Be Retained?

Although ActFocus improves over Uniform across all main PPO experiments, the optimal α varies by configuration. The intuition is simple: when the base model already reasons well enough for the task, a smaller α removes redundant reasoning updates and concentrates RL on action decisions; when reasoning still needs refinement, a larger α preserves that gradient signal.

Figure 6a illustrates this trade-off on two representative environments. On Sokoban 3B, the best result comes from the aggressive setting $\alpha = 0.1$, consistent with a model that already has sufficient spatial reasoning and benefits from focusing updates on action tokens. On WebShop 3B, the optimum shifts to $\alpha = 0.3$, and the extreme $\alpha = 0$ eventually collapses. Unlike Sokoban, WebShop requires ongoing reasoning refinement, such as query reformulation, evidence aggregation, and product comparison, making full reasoning suppression too aggressive.



(a) Effect of α (b) Effect of β (c) Signal comparison

Figure 6: **Ablations on ACTFOCUS (PPO).** (a) Effect of α on Sokoban 3B and WebShop 3B; the legend in the first panel applies to both. (b) Sweeping β on WebShop 3B: strict success peaks at β=1.0 (39.8%) and degrades at both extremes. (c) Replacing energy with alternative signals on WebShop 3B: energy outperforms entropy, log-probability shift, and policy NLL.

The two extremes illustrate why α requires calibration. At $\alpha = 1.0$, reasoning tokens retain full credit, so energy-based amplification alone cannot overcome the dominance of the reasoning majority. At $\alpha = 0$, reasoning is suppressed entirely, discarding the gradient signal needed for coherent reasoning. The best α sits between these extremes, balancing action focus with reasoning retention.

5.5 How Strongly Should Energy Modulate Action Credit?

With the reasoning-retention parameter α fixed, β controls how strongly energy reshapes credit within the action span. Setting $\beta = 0$ recovers uniform weighting across action tokens, so the action component gain depends only on α, while a larger β progressively concentrates updates on high-energy positions. We sweep $\beta \in \{0, 0.3, 0.5, 1.0, 2.0\}$ on WebShop 3B and train each setting with PPO for 100 steps. As shown in Figure 6b, performance is non-monotonic. Strict success improves from 29.7% at $\beta = 0$ to 39.8% at $\beta = 1.0$, but drops to 25.4% at $\beta = 2.0$, falling below the $\beta = 0$ baseline. This indicates that energy modulation must be strong enough to differentiate uncertain action tokens from confident ones, since treating all action tokens equally fails to exploit the signal energy carries about which decisions are most informative. At the same time, β cannot be too large: when the sigmoid-based weighting becomes too peaked, gradient mass collapses onto a small number of high-energy outliers and leaves the remaining action tokens under-trained.

5.6 Why Energy? A Comparison Against Alternative Signals

A natural question is whether the gains from ACTFOCUS are specific to energy or an artifact of the reweighting architecture. To test this, we replace energy E_t with three alternative signals while keeping all other parameters fixed on WebShop 3B: entropy, policy NLL, and the log-probability shift. Figure 6c shows that energy reaches 36.7% strict success, outperforming entropy (30.9%), log-probability shift (29.4%), and policy NLL (26.2%). The gap traces back to the properties motivated in Sec. 4: policy-derived signals such as policy NLL and log-probability shift change as the policy updates, making the weighting target a moving one during training, while frozen-reference energy remains stable and retains magnitude information that the frozen-reference entropy discards after normalization. A detailed comparison is included in Appendix C.3.

6 Conclusion

In this work, we identify the *Action Bottleneck* as a critical failure mode in agentic reinforcement learning. To address this, we propose a simple structure-aware token reweighting scheme with energy-based redistribution, which redirects gradient mass toward action-relevant decisions. Results across four environments show that this intervention improves learning effectiveness and training stability. Our findings highlight the Action Bottleneck as an important token-weighting issue in the evaluated agentic RL settings and show that correcting it provides a principled pathway toward more effective training. Since ACTFOCUS delivers these gains at comparable computational cost to standard PPO and GRPO, it serves as a practical replacement for the uniform token-averaged objective in existing agentic RL pipelines. Limitations and broader impact are discussed in Appendix E.

Ethics Statement

This work studies how gradient credit is allocated across tokens during agentic RL training. ActFocus does not change the reward function or the behaviors that are reinforced; it only redistributes a fixed advantage signal across tokens within a trajectory, and therefore inherits the safety properties of the surrounding RL pipeline. All experiments use synthetic environments (Sokoban, FrozenLake, Sudoku) and the public WebShop benchmark; no human subjects or personal data are involved. A broader discussion of potential positive and negative impacts is given in Appendix E.1.

Reproducibility Statement

The method is fully specified by the token weight in Eq. 5 and the weighted objective in Eq. 6; it changes neither the advantage estimator nor the auxiliary regularizers (Appendix B). Environment definitions, prompt templates, rollout settings, training hyperparameters, and the evaluation protocol are given in Appendix C, and the exact system prompts are reproduced in Appendix F. The alternative signals used in the ablation of Sec. 5.6 are defined in Appendix C.3. Tables 2, 5, and 6 report mean $\pm$ sample standard deviation over five random seeds. Supplementary studies are reported in Appendix D.

The Use of Large Language Models (LLMs)

In this work, we used generative AI tools to edit the paper for readability, to draft parts of the text, and to write plotting scripts for the figures; all such text and code were reviewed and verified by the authors. We have not used generative AI tools to generate synthetic data, to develop the theoretical framing or formulate mathematical claims, to design the methodology or experiments, to implement the method, or to interpret the results. Assisting with proofs, translation, dataset cleaning, and qualitative data analysis are not applicable to this work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

References

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, 2023. URL https://arxiv.org/abs/2311.18232.

Alex J. Chan, Hao Sun, Samuel Holt, and Mihaela van der Schaar. Dense reward for free in reinforcement learning from human feedback, 2024. URL https://arxiv.org/abs/2402.00782.

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. Reinforcement learning for long-horizon interactive llm agents, 2025. URL https://arxiv.org/abs/2502.01600.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards, 2025a. URL https://arxiv.org/abs/2502.01456.

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, 2025b. URL https://arxiv.org/abs/2505.22617.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training, 2025. URL https://arxiv.org/abs/2505.10978.

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one, 2020. URL https://arxiv.org/abs/1912.03263.

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

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. Vineppo: Refining credit assignment in rl training of llms, 2025. URL https://arxiv.org/abs/2410.01679.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. Llms get lost in multi-turn conversation, 2025. URL https://arxiv.org/abs/2505.06120.

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in neural information processing systems*, 33:21464–21475, 2020.

Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Junge Zhang, and Jianbin Jiao. Agentic reinforcement learning with implicit step rewards, 2025a. URL https://arxiv.org/abs/2509.19199.

Zheng Liu, Mengjie Liu, Siwei Wen, Mengzhang Cai, Bin Cui, Conghui He, and Wentao Zhang. From uniform to heterogeneous: Tailoring policy optimization to every token's nature, 2025b. URL https://arxiv.org/abs/2509.16591.

Chiyu Ma, Shuo Yang, Kexin Huang, Jinda Lu, Haoming Meng, Shangshang Wang, Bolin Ding, Soroush Vosoughi, Guoyin Wang, and Jingren Zhou. Fipo: Eliciting deep reasoning with future-kl influenced policy optimization, 2026. URL https://arxiv.org/abs/2603.19835.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Eduardo Pignatelli, Johan Ferret, Matthieu Geist, Thomas Mesnard, Hado van Hasselt, Olivier Pietquin, and Laura Toni. A survey of temporal credit assignment in deep reinforcement learning, 2024. URL https://arxiv.org/abs/2312.01072.

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

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023. URL https://arxiv.org/abs/2302.04761.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018. URL https://arxiv.org/abs/1506.02438.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Leyang Shen, Yang Zhang, Chun Kai Ling, Xiaoyan Zhao, and Tat-Seng Chua. Carl: Focusing agentic reinforcement learning on critical actions, 2026. URL https://arxiv.org/abs/2512.04949.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Hongze Tan, Zihan Wang, Jianfei Pan, Jinghao Lin, Hao Wang, Yifan Wu, Tao Chen, Zhihang Zheng, Zhihao Tang, and Haihua Yang. Gtpo and grpo-s: Token and sequence-level reward shaping with policy entropy, 2026. URL https://arxiv.org/abs/2508.04349.

Jean Vassoyan, Nathanaël Beau, and Roman Plaud. Ignore the KL penalty! boosting exploration on critical tokens to enhance RL fine-tuning. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 6123–6133, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.340. URL https://aclanthology.org/2025.findings-naacl.340/.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

Xiaoxuan Wang, Han Zhang, Haixin Wang, Yidan Shi, Ruoyan Li, Kaiqiao Han, Chenyi Tong, Haoran Deng, Renliang Sun, Alexander Taylor, Yanqiao Zhu, Jason Cong, Yizhou Sun, and Wei Wang. Arlarena: A unified framework for stable agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2602.21534.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2504.20073.

Muning Wen, Ziyu Wan, Weinan Zhang, Jun Wang, and Ying Wen. Reinforcing language agents via policy optimization with action decomposition, 2024. URL https://arxiv.org/abs/2405.15821.

Georg Wölflein, Dyke Ferber, Daniel Truhn, Ognjen Arandjelović, and Jakob Nikolas Kather. Llm agents making agent tools, 2025. URL https://arxiv.org/abs/2502.11705.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, Wei He, Yiwen Ding, Guanyu Li, Zehui Chen, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Tao Gui, Zuxuan Wu, Qi Zhang, Xuanjing Huang, and Yu-Gang Jiang. Agentgym-rl: Training llm agents for long-horizon decision making through multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2509.08755.

Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *International conference on machine learning*, pp. 2635–2644. PMLR, 2016.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents, 2023a. URL https://arxiv.org/abs/2207.01206.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023b. URL https://arxiv.org/abs/2210.03629.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

A NOTATION

Table 3 summarizes the notation used throughout the paper. Symbols are grouped by category for ease of reference.

Table 3: Summary of notation used in the paper.

Symbol	Meaning
Environment and trajectory	
$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R \rangle$	Multi-turn agentic MDP
$s_k \in \mathcal{S}$	State at turn k (interaction history)
$a_k \in \mathcal{A}$	Environment-facing action at turn k
o_k	Policy response at turn k (reasoning + action)
$\tau = (s_1, o_1, \ldots, s_K, o_K)$	Multi-turn trajectory
K	Total number of turns in trajectory τ
T	Total number of response tokens in trajectory τ
$R(\tau)$	Terminal episodic reward
r_t	Per-token reward; $r_t = R(\tau)$ if $t = T$, else 0
Policy and tokens	
π_θ	Trained policy with parameters θ
$\pi_{\theta_{\text{old}}}$	Rollout policy (snapshot used for sampling)
π_{ref}	Frozen reference policy (KL anchor and energy source)
y_t	Generic response token at position t
c_t	Context preceding y_t (state, prior turns, prior tokens)
$\mathcal{T}_{\text{think}}$	Set of token positions inside `<think>` spans
$\mathcal{T}_{\text{action}}$	Set of token positions inside `<answer>` spans
Policy optimization	
$J(\theta)$	Expected episodic return objective
$\rho_t(\theta)$	Token-level importance ratio $\pi_\theta(y_t \mid c_t)/\pi_{\theta_{\text{old}}}(y_t \mid c_t)$
$\hat{A}_t$	Per-token advantage estimate
$\hat{A}_\tau$	Trajectory-level (GRPO) advantage
V_ϕ	Token-level value function used by PPO+GAE
ϵ	PPO clipping threshold
(γ, λ)	Discount factor and GAE coefficient
G	Number of trajectories per trajectory group (GRPO)
$\beta_{\text{KL}}, \beta_{\text{ent}}$	KL and entropy regularization coefficients
Token energy and reweighting	
f_t^{ref}	Frozen-reference next-token logits at position t
E_t	Frozen-reference token energy at position t
μ_E, σ_E	Mean and std of E_t over action tokens in a batch
w_t	Per-token weight in the structure-aware loss
α	Reasoning-retention coefficient (w_t on `<think>` tokens)
β	Action component energy modulation strength
σ_g	Reward variance within a trajectory group

B REINFORCEMENT LEARNING BACKGROUND

We follow the notation of Sec. 3: y_t denotes a generic response token, c_t denotes the context consisting of all tokens preceding y_t, including state observations and prior turns within the trajectory, and the token-level importance ratio between the current policy and the rollout policy is:

$$\rho_t(\theta) = \frac{\pi_\theta(y_t \mid c_t)}{\pi_{\theta_{\text{old}}}(y_t \mid c_t)}. \tag{8}$$

Throughout this appendix, T denotes the total number of response tokens in a trajectory τ.

General objective. The objective maximizes expected episodic return over trajectories:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\big[R(\tau)\big], \tag{9}$$

where $\tau = (s_1, o_1, \ldots, s_K, o_K)$ is a K-turn trajectory and $R(\tau)$ is the terminal episodic reward defined in Sec. 3. The corresponding policy gradient is:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=1}^{T} \hat{A}_t \, \nabla_\theta \log \pi_\theta(y_t \mid c_t)\right], \tag{10}$$

which distributes credit across all response tokens y_t in τ via a per-token advantage estimate $\hat{A}_t$. PPO and GRPO differ primarily in how $\hat{A}_t$ is constructed.

Proximal Policy Optimization (PPO). PPO (Schulman et al., 2017) stabilizes the policy update through a clipped surrogate objective combined with a learned token-level value function V_ϕ, which estimates the expected return-to-go from context c_t. The token-level surrogate loss is:

$$\mathcal{L}_{\text{PPO}}(\theta) = -\mathbb{E}_t\Big[\min\Big(\rho_t(\theta)\hat{A}_t,\ \text{clip}\big(\rho_t(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_t\Big)\Big], \tag{11}$$

where ϵ is the clipping threshold. The advantage $\hat{A}_t$ is computed via Generalized Advantage Estimation (GAE) (Schulman et al., 2018):

$$\hat{A}_t^{\text{GAE}(\gamma,\lambda)} = \sum_{l=0}^{T-t} (\gamma\lambda)^l \, \delta_{t+l}, \qquad \delta_t = r_t + \gamma V_\phi(c_{t+1}) - V_\phi(c_t), \tag{12}$$

where γ and λ control the bias and variance of the advantage estimate. Following standard practice in LLM RL, the episodic reward $R(\tau)$ is delivered at the final response token of the trajectory: $r_t = R(\tau)$ if $t = T$, and $r_t = 0$ otherwise. The value function V_ϕ is trained jointly with the policy by regressing toward the empirical return-to-go.

Group Relative Policy Optimization (GRPO). GRPO (Shao et al., 2024) removes the value function by sampling a group of G trajectories $\{\tau^{(i)}\}_{i=1}^{G}$ from the same prompt and computing a group-normalized trajectory advantage:

$$\hat{A}_{\tau^{(i)}} = \frac{R(\tau^{(i)}) - \text{mean}\big(\{R(\tau^{(j)})\}_{j=1}^{G}\big)}{\text{std}\big(\{R(\tau^{(j)})\}_{j=1}^{G}\big)}, \tag{13}$$

which is the explicit group-level form of $\hat{A}_\tau$ in Eq. 3. Because this advantage is defined at the trajectory level, it is broadcast to every token within the trajectory: $\hat{A}_t = \hat{A}_{\tau^{(i)}}$ for all tokens y_t in $\tau^{(i)}$. The clipped surrogate then takes the form:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{G}\sum_{i=1}^{G} \mathbb{E}_t\Big[\min\Big(\rho_t(\theta)\hat{A}_{\tau^{(i)}},\ \text{clip}\big(\rho_t(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_{\tau^{(i)}}\Big)\Big]. \tag{14}$$

GRPO reduces memory and computation by removing the critic, but its uniform broadcasting of $\hat{A}_\tau$ across all tokens makes it particularly sensitive to the Action Bottleneck studied in this work.

Auxiliary regularizers. Both PPO and GRPO are typically augmented with two regularizers. A KL term against a reference policy π_{ref} controls how far the trained policy drifts from its initialization:

$$\mathcal{L}_{\text{KL}}(\theta) = \beta_{\text{KL}}\, \mathbb{E}_t\big[D_{\text{KL}}\big(\pi_\theta(\cdot \mid c_t) \,\|\, \pi_{\text{ref}}(\cdot \mid c_t)\big)\big], \tag{15}$$

and an entropy bonus $\mathcal{L}_{\text{ent}}(\theta) = -\beta_{\text{ent}}\, \mathbb{E}_t\big[H\big(\pi_\theta(\cdot \mid c_t)\big)\big]$ encourages exploration during early training. The total objective is $\mathcal{L}_{\text{total}} = \mathcal{L}(\theta) + \mathcal{L}_{\text{KL}}(\theta) + \mathcal{L}_{\text{ent}}(\theta)$, where $\mathcal{L}(\theta)$ is the surrogate loss of either PPO or GRPO. In practice, the same frozen base model serves as the reference policy π_{ref} for KL regularization and as the source of the logits used to compute token energy in Eq. 4. The values of β_{KL} and β_{ent} used in our experiments are listed in Appendix C.

Relation to our weighting scheme. The structure-aware reweighting in Eq. 6 is orthogonal to both algorithms' choice of $\hat{A}_t$. Under PPO, $\hat{A}_t$ remains the GAE estimate; under GRPO, it remains the group-normalized trajectory advantage. ActFocus only changes the per-token weight w_t that multiplies the surrogate, so it composes with PPO and GRPO without modifying their advantage estimators or auxiliary regularizers.

B.1 Gradient Allocation under Structure-Aware Token Weighting

We derive how the weights in Eq. 5 change the policy-surrogate gradient in Eq. 6. The analysis separates allocation between reasoning and action spans from energy-dependent weighting within action spans.

Setup. Consider the valid response-token positions contributing to one normalized surrogate objective, partitioned into $\mathcal{T}_{\text{think}}$ and $\mathcal{T}_{\text{action}}$. Write $N_r = |\mathcal{T}_{\text{think}}| > 0$, $N_a = |\mathcal{T}_{\text{action}}| > 0$, and $N = N_r + N_a$. All comparisons below are evaluated at the same policy parameters and on the same sampled batch. The sampled tokens, span masks, advantage estimates, old policy, and reference-derived scores are held fixed during differentiation. Context tokens are fixed, but differentiation through the current policy is otherwise unchanged.

Define the token surrogate loss and its parameter gradient as

$$\ell_t(\theta) = -\min\Big(\rho_t(\theta)\hat{A}_t,\ \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\Big), \tag{16}$$

$$\mathbf{g}_t(\theta) = \nabla_\theta \ell_t(\theta). \tag{17}$$

This definition retains both importance ratios and clipping. On an unclipped branch,

$$\mathbf{g}_t(\theta) = -\hat{A}_t \nabla_\theta \rho_t(\theta) = -\rho_t(\theta)\hat{A}_t \nabla_\theta \log \pi_\theta(y_t \mid c_t). \tag{18}$$

At $\theta = \theta_{\text{old}}$, the ratio equals one and clipping is inactive, giving $\mathbf{g}_t = -\hat{A}_t \nabla_\theta \log \pi_\theta(y_t \mid c_t)$. Away from this point, the ratio and clipping must be retained, as in Eqs. 16–17. The identities below apply wherever the surrogate is differentiable, or with a consistent choice of subgradient at clipping boundaries. They also apply to asymmetric clipping. We analyze the policy surrogate before optimizer preconditioning and auxiliary regularization.

For action tokens, abbreviate the normalized energy score by

$$u_t = \text{sigmoid}\left(\frac{E_t - \mu_E}{\sigma_E}\right), \qquad \bar{u} = \frac{1}{N_a}\sum_{t\in\mathcal{T}_{\text{action}}} u_t, \qquad \kappa = 1 + \beta\bar{u}, \tag{19}$$

where $\sigma_E > 0$, using the numerical stabilizer in Appendix C.3 when necessary. For $\alpha \in [0,1]$ and $\beta \geq 0$, the normalizer is

$$Z = \sum_t w_t = \alpha N_r + N_a \kappa > 0. \tag{20}$$

Since the reference-derived weights do not depend on θ for this fixed batch, differentiation gives the exact identity

$$\nabla_\theta \mathcal{L}_{\alpha,\beta} = \frac{1}{Z}\left[\alpha \sum_{t\in\mathcal{T}_{\text{think}}} \mathbf{g}_t + \sum_{t\in\mathcal{T}_{\text{action}}} (1+\beta u_t)\mathbf{g}_t\right]. \tag{21}$$

Here $\mathcal{L}_{\alpha,\beta}$ is Eq. 6 with the indicated weights. No independence or equal-norm assumption on token gradients is required.

Proposition 1 (Token-count weighting under uniform averaging). Let

$$\bar{\mathbf{g}}_r = \frac{1}{N_r}\sum_{t\in\mathcal{T}_{\text{think}}} \mathbf{g}_t, \qquad \bar{\mathbf{g}}_a = \frac{1}{N_a}\sum_{t\in\mathcal{T}_{\text{action}}} \mathbf{g}_t. \tag{22}$$

The uniform objective $\mathcal{L}_{\text{U}} = \mathcal{L}_{1,0}$ satisfies

$$\nabla_\theta \mathcal{L}_{\text{U}} = \frac{N_r}{N}\bar{\mathbf{g}}_r + \frac{N_a}{N}\bar{\mathbf{g}}_a. \tag{23}$$

Proof. Set $\alpha = 1$ and $\beta = 0$ in Eq. 21, and group the terms by span. □

Thus, token counts determine the coefficients on the two mean gradients. They do not by themselves determine the norms of the resulting vector components, which also depend on $\bar{\mathbf{g}}_r$ and $\bar{\mathbf{g}}_a$.

Proposition 2 (Action emphasis under structure-aware weighting). Define the weighted action mean and its mixing coefficient by

$$\bar{\mathbf{g}}_a^{\,E} = \frac{\sum_{t\in\mathcal{T}_{\text{action}}}(1+\beta u_t)\mathbf{g}_t}{N_a\kappa}, \qquad \lambda_a = \frac{N_a\kappa}{\alpha N_r + N_a\kappa}. \tag{24}$$

Then

$$\nabla_\theta \mathcal{L}_{\alpha,\beta} = (1-\lambda_a)\bar{\mathbf{g}}_r + \lambda_a \bar{\mathbf{g}}_a^{\,E}, \qquad \lambda_a \geq \frac{N_a}{N}. \tag{25}$$

For fixed batch scores, λ_a decreases with α and is nondecreasing with β.

Proof. The decomposition follows from Eq. 21. Since $\kappa \geq 1 \geq \alpha$,

$$\lambda_a - \frac{N_a}{N} = \frac{N_a N_r(\kappa-\alpha)}{NZ} \geq 0, \tag{26}$$

$$\frac{\partial\lambda_a}{\partial\alpha} = -\frac{N_a\kappa N_r}{Z^2} < 0, \qquad \frac{\partial\lambda_a}{\partial\beta} = \frac{\alpha N_r N_a \bar{u}}{Z^2} \geq 0. \tag{27}$$

□

Exact relative amplification from reasoning downweighting. For $\beta = 0$, let $\mathbf{G}_r^\alpha$ and $\mathbf{G}_a^\alpha$ denote the two additive gradient components of $\nabla_\theta\mathcal{L}_{\alpha,0}$. With $Z_\alpha = \alpha N_r + N_a$, direct substitution gives

$$\mathbf{G}_r^\alpha = \frac{\alpha N}{Z_\alpha}\mathbf{G}_r^{\mathrm{U}}, \qquad \mathbf{G}_a^\alpha = \frac{N}{Z_\alpha}\mathbf{G}_a^{\mathrm{U}}, \tag{28}$$

where $\mathbf{G}_r^{\mathrm{U}}$ and $\mathbf{G}_a^{\mathrm{U}}$ are the uniform components in Eq. 23. Consequently, for $0 < \alpha \leq 1$ and nonzero components,

$$\frac{\|\mathbf{G}_a^\alpha\|}{\|\mathbf{G}_r^\alpha\|} = \frac{1}{\alpha}\frac{\|\mathbf{G}_a^{\mathrm{U}}\|}{\|\mathbf{G}_r^{\mathrm{U}}\|}. \tag{29}$$

This is a statement about actual span-component norms on the same batch, not only token counts. For example, with a 4% action fraction, $\alpha = 0.1$ and $\beta = 0$ increase the action mixing coefficient from 0.04 to $0.04/(0.1\times 0.96 + 0.04) \approx 0.294$ and multiply the action-to-reasoning component-norm ratio by ten. The mixing coefficient is not itself a gradient-norm percentage. More generally, for any fixed β, changing α from one to a positive value multiplies the action-to-reasoning component-norm ratio by $1/\alpha$, provided both components are nonzero.

Proposition 3 (Within-action energy weighting). Define the empirical score–gradient covariance over action positions as

$$\mathbf{C}_a = \frac{1}{N_a}\sum_{t\in\mathcal{T}_{\text{action}}}(u_t - \bar{u})(\mathbf{g}_t - \bar{\mathbf{g}}_a). \tag{30}$$

The weighted action mean and the full gradient satisfy

$$\bar{\mathbf{g}}_a^{\,E} = \bar{\mathbf{g}}_a + \frac{\beta}{\kappa}\mathbf{C}_a, \tag{31}$$

$$\nabla_\theta\mathcal{L}_{\alpha,\beta} = (1-\lambda_a)\bar{\mathbf{g}}_r + \lambda_a\bar{\mathbf{g}}_a + \frac{\beta N_a}{Z}\mathbf{C}_a. \tag{32}$$

Proof. Expanding the covariance gives $\mathbf{C}_a = N_a^{-1}\sum_{t\in\mathcal{T}_{\text{action}}} u_t\mathbf{g}_t - \bar{u}\bar{\mathbf{g}}_a$. Hence

$$\sum_{t\in\mathcal{T}_{\text{action}}}(1+\beta u_t)\mathbf{g}_t = N_a\left(\kappa\bar{\mathbf{g}}_a + \beta\mathbf{C}_a\right). \tag{33}$$

Substitute this expression into Eqs. 24 and 21. □

Subtracting Eq. 23 also gives

$$\nabla_\theta\mathcal{L}_{\alpha,\beta} - \nabla_\theta\mathcal{L}_{\mathrm{U}} = \left(\lambda_a - \frac{N_a}{N}\right)(\bar{\mathbf{g}}_a - \bar{\mathbf{g}}_r) + \frac{\beta N_a}{Z}\mathbf{C}_a. \tag{34}$$

This separates the change from uniform weighting into a shift between span means and an adjustment within the action span.

Equation 32 separates two effects of energy modulation: it changes the total action coefficient through λ_a and changes the mean action gradient through $\mathbf{C}_a$. For an analytical control, assign weight α to reasoning tokens and the constant weight κ to every action token, and call the resulting objective $\mathcal{L}_{\text{const}}$. This control has the same total reasoning and action weights and the same normalizer Z. Therefore,

$$\nabla_\theta \mathcal{L}_{\alpha,\beta} - \nabla_\theta \mathcal{L}_{\text{const}} = \frac{\beta N_a}{Z} \mathbf{C}_a. \tag{35}$$

The control is defined for this analysis and should not be confused with the all-token uniform baseline $\mathcal{L}_{\text{U}}$.

Random assignment. Let ϖ be a uniformly random permutation of the action positions, conditional on the fixed batch, and assign action weight $1+\beta u_{\varpi(t)}$. Permutation preserves Z, while $\mathbb{E}_\varpi[u_{\varpi(t)}] = \bar{u}$. Thus,

$$\mathbb{E}_\varpi\big[\nabla_\theta \mathcal{L}_{\text{perm}}\big] = \nabla_\theta \mathcal{L}_{\text{const}}. \tag{36}$$

Permutation removes the covariance term in expectation and preserves the total span weights. If shuffling is restricted to individual spans or other blocks, the identity applies separately within each shuffled block, using its own mean action weight. This is a fixed-batch identity, not a claim that randomly weighted and constant-weight training have the same trajectories or final performance.

When can the covariance adjustment help? Suppose the expected return J is differentiable at the current parameters, and define each action token's return-aligned update score $h_t = -\langle \nabla_\theta J, \mathbf{g}_t \rangle$. Writing $\bar{h} = N_a^{-1} \sum_{t \in \mathcal{T}_{\text{action}}} h_t$ and $\text{Cov}_a(u,h) = N_a^{-1} \sum_{t \in \mathcal{T}_{\text{action}}} (u_t - \bar{u})(h_t - \bar{h})$, a first-order expansion gives

$$\begin{aligned} &J(\theta - \eta \nabla_\theta \mathcal{L}_{\alpha,\beta}) - J(\theta - \eta \nabla_\theta \mathcal{L}_{\text{const}}) \\ &\quad = \eta \frac{\beta N_a}{Z} \text{Cov}_a(u,h) + o(\eta). \end{aligned} \tag{37}$$

Indeed, the first-order difference is $-\eta \langle \nabla_\theta J, \nabla_\theta \mathcal{L}_{\alpha,\beta} - \nabla_\theta \mathcal{L}_{\text{const}} \rangle$, and Eq. 35 yields the result. If $\beta > 0$ and this covariance is positive, the energy-weighted step has higher return than the matched constant-weight step for sufficiently small η. This is a conditional comparison of plain gradient steps, not an unconditional improvement or convergence guarantee. The covariance condition concerns token-gradient alignment and is not established by the group-level energy–reward-variance correlation alone.

Energy and token gradients. For logits $\mathbf{f}$, let $\mathbf{p} = \text{softmax}(\mathbf{f})$ and $E(\mathbf{f}) = -\log \sum_v \exp(f_v)$. Direct differentiation gives

$$\nabla_{\mathbf{f}} E = -\mathbf{p}, \qquad \nabla_{\mathbf{f}} \log p_y = \mathbf{e}_y - \mathbf{p} = \mathbf{e}_y + \nabla_{\mathbf{f}} E, \tag{38}$$

where $\mathbf{e}_y$ is the one-hot vector for token y. This relates the derivatives, but does not equate scalar energy with uncertainty or gradient magnitude. In particular, $\text{softmax}(\mathbf{f}+b\mathbf{1}) = \mathbf{p}$ while $E(\mathbf{f}+b\mathbf{1}) = E(\mathbf{f}) - b$. In ActFocus, energy is computed by the frozen reference, so it controls the coefficients multiplying current-policy gradients rather than contributing a derivative through the reference model. The preceding results establish the gradient-allocation mechanism; the usefulness of this reference score is evaluated empirically.

C Experimental Details

C.1 Task Environment Details

We evaluate our method on four multi-turn environments that span a broad range of reasoning demands and interaction structures. Sokoban, FrozenLake, and Sudoku are compact symbolic environments with fully specified dynamics, making them well-suited for controlled analysis of reinforcement learning behavior and token-level credit assignment. Although all three are synthetic, they stress different forms of difficulty: irreversible planning in Sokoban, stochastic transitions in FrozenLake, and structured constraint reasoning in Sudoku. To complement these controlled settings, we also include WebShop (Yao et al., 2023a), a more realistic environment in which the agent must interpret natural-language instructions, navigate a semi-structured interface, and complete a purchase through a sequence of grounded actions. Together, these environments let us study the Action Bottleneck under both tightly controlled symbolic tasks and a more open-ended web setting.

Sokoban. Sokoban is a grid-based planning task in which the agent must push boxes onto target cells within a limited action budget. Unlike standard navigation problems, Sokoban is irreversible: boxes can be pushed but not pulled back, so an early mistake can permanently destroy a valid solution path. This makes the environment a useful testbed for long-horizon planning and action precision in multi-turn agent training.

FrozenLake. FrozenLake combines goal-directed planning with stochastic transitions. The agent must navigate a grid world to reach the goal while avoiding holes, but actions are executed on slippery tiles and may lead to unintended outcomes. As a result, good performance depends not only on moving toward the goal but also on choosing actions that remain robust under transition uncertainty.

Sudoku. Sudoku evaluates structured reasoning in a sequential decision-making setting. The agent fills a 4×4 grid one move at a time, subject to row, column, and subgrid constraints. Because each valid action depends on a set of coupled logical conditions, the environment places more emphasis on inference and constraint satisfaction than on exploration or long-horizon control.

WebShop. WebShop (Yao et al., 2023a) is a realistic multi-turn web interaction environment in which the agent must satisfy a natural-language shopping request through search, navigation, option selection, and purchase. Compared with the symbolic environments, WebShop introduces richer language grounding, a semi-structured interface, and a less cleanly defined action space. It therefore serves as a complementary setting for studying whether token-level credit assignment mechanisms continue to help when reasoning and action unfold in a more realistic decision-making process.

C.2 Training and Evaluation Settings

We conduct our main experiments using the **Qwen2.5-Instruct** family (Qwen et al., 2025) at three scales: 0.5B, 1.5B, and 3B. For each configuration, we train with both PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024) under two token-weighting variants: Uniform and ActFocus. These main experiments are built on the RAGEN framework (Wang et al., 2025), which is implemented on top of veRL[2], and are run on a single node with $4\times$ NVIDIA H100 GPUs. We train for 200 rollout–update iterations on Sokoban, FrozenLake, and Sudoku 4×4, and for 100 iterations on WebShop (Yao et al., 2023a) due to its long-context nature.

Each on-policy update uses grouped rollouts. On Sokoban, FrozenLake, and Sudoku, we use prompt-group size $P = 8$ and sample $G = 16$ trajectories per prompt; on WebShop, we use $P = 16$ and $G = 8$. This yields 128 trajectories per training step before filtering. Agents are allowed at most 2 actions per turn and 10 actions per episode on Sokoban and FrozenLake (5-turn horizon), an 8-turn horizon with a 16-step cap on Sudoku, and a 9-turn horizon with a single action per turn on WebShop. Following StarPO-S (Wang et al., 2025), we apply reward-variance trajectory selection with filter ratio 0.25 on Sokoban and Sudoku. On WebShop and GRPO-based FrozenLake, we use a filter ratio of 1.0 (no filtering), since environmental stochasticity contaminates within-prompt reward variance as a selection signal. PPO uses a mini-batch size of $E = 32$, with per-GPU micro-batch sizes of 1–8 depending on model scale and environment, under FSDP and Ray.

Policy optimization uses GAE with $(\gamma, \lambda) = (1.0, 1.0)$, Adam with $(\beta_1, \beta_2) = (0.9, 0.999)$, actor learning rate 1×10^{-6}, and critic learning rate 1×10^{-5}. We use asymmetric clipping $(\epsilon_{\text{low}}, \epsilon_{\text{high}}) = (0.2, 0.28)$ following Yu et al. (2025), together with entropy regularization of coefficient 0.001. For Sokoban, FrozenLake, and Sudoku, we drop the KL loss term during training and track KL post hoc, following Yu et al. (2025). For WebShop under GRPO, we add a KL loss with coefficient 0.001 and the k_1 estimator to reduce late-stage collapse, following the vanilla StarPO setting in Wang et al. (2025). We impose a format penalty of -0.1 when the agent fails to produce a valid `<think>`/`<answer>` structure. Our two token-weighting variants instantiate w_t in Eq. 5: Uniform (α=1.0, β=0) and ActFocus (α=0.1, β=0.5). Rollouts are generated with vLLM using `tensor_parallel_size`=1 and `enforce_eager`, with a maximum model length of 3600 tokens and 400-token responses for Sokoban, FrozenLake, and Sudoku, and an 8192-token context window for WebShop.

[2] https://github.com/volcengine/verl

Evaluation. We evaluate every 10 training steps and also before training begins (`val_before_train`). For Sokoban, FrozenLake, and Sudoku, we use a fixed evaluation set of 32 distinct prompts rolled out 16 times each under stochastic decoding, for a total of 512 trajectories per evaluation. For WebShop, we use 256 distinct prompts with one rollout each (256 trajectories). Decoding uses a temperature of 0.5 to measure robustness under sampling. Evaluation episodes are truncated using the same per-environment turn and action budgets as in training. We report evaluation success rate on this fixed evaluation set at step 200 for Sokoban, FrozenLake, and Sudoku, and at step 100 for WebShop. Supplementary evaluations with Qwen2.5-7B-Instruct and action-weight controls are described in Appendices D.4 and D.5.

C.3 Signal Variants for the Energy Ablation

This appendix defines the four action-token signals compared in Sec. 5.6. The goal is to separate the weighting architecture from the choice of signal. All variants use the same weighting pipeline. They differ only in the raw score s_t assigned to each action token.

Common pipeline. For a batch of action-token positions $\mathcal{T}_{\text{action}}$, we first normalize the raw signal s_t within the action tokens:

$$\mu_s = \frac{1}{|\mathcal{T}_{\text{action}}|} \sum_{t \in \mathcal{T}_{\text{action}}} s_t, \qquad \sigma_s = \sqrt{\frac{1}{|\mathcal{T}_{\text{action}}|} \sum_{t \in \mathcal{T}_{\text{action}}} (s_t - \mu_s)^2 + \varepsilon}. \tag{39}$$

We then map the normalized score to $(0, 1)$:

$$\tilde{s}_t = \text{sigmoid}\left(\frac{s_t - \mu_s}{\sigma_s} \right). \tag{40}$$

The final token weight is

$$w_t = \begin{cases} \alpha, & t \in \mathcal{T}_{\text{think}}, \\ 1 + \beta\, \tilde{s}_t, & t \in \mathcal{T}_{\text{action}}. \end{cases} \tag{41}$$

Thus, all four variants use the same α, β, normalization, and sigmoid mapping. Only the raw action-token signal s_t changes.

(1) Frozen-reference energy. Energy is the signal used by ActFocus. It measures uncertainty from the frozen reference model before applying any policy update:

$$s_t^{\text{energy}} = E_t = -\log \sum_{v \in V} \exp\big(f_{t,v}^{\text{ref}}\big), \tag{42}$$

where $f_{t,v}^{\text{ref}}$ is the frozen-reference logit for vocabulary token v. This matches the definition in Eq. 4. Because the reference model is frozen, this score does not change during training.

(2) Frozen-reference entropy. Entropy is another fixed-reference uncertainty signal. It measures how spread out the frozen reference distribution is:

$$s_t^{\text{entropy}} = H\big(\pi_{\text{ref}}(\cdot \mid c_t)\big) = -\sum_{v \in V} p_{t,v}^{\text{ref}} \log p_{t,v}^{\text{ref}}, \qquad p_{t,v}^{\text{ref}} = \frac{\exp(f_{t,v}^{\text{ref}})}{\sum_{u \in V} \exp(f_{t,u}^{\text{ref}})}. \tag{43}$$

Entropy uses the same frozen-reference forward pass as energy. The difference is that entropy is computed after softmax normalization, and therefore captures only the dispersion of the distribution, discarding the absolute logit scale.

(3) Policy NLL. Policy NLL measures how surprising the sampled token is under the rollout policy:

$$s_t^{\text{nll}} = -\log \pi_{\theta_{\text{old}}}(y_t \mid c_t). \tag{44}$$

This signal is cheap because PPO already stores rollout log-probabilities. However, it is policy-dependent: as the policy changes during training, the meaning and scale of this signal also change.

Table 4: Stability of the four token-level signals during training. Frozen-reference signals provide a fixed weighting target, while policy-derived signals move as the policy changes.

Signal	Source	Drifts during training?
Energy	Frozen reference π_{ref}	No
Entropy	Frozen reference π_{ref}	No
Policy NLL	Rollout policy $\pi_{\theta_{\text{old}}}$	Yes
Log-probability shift	Rollout policy and frozen reference	Partly

(4) Log-probability shift. Log-probability shift measures how much the rollout policy has moved away from the frozen reference at the sampled token:

$$s_t^{\text{shift}} = \log \pi_{\theta_{\text{old}}}(y_t \mid c_t) - \log \pi_{\text{ref}}(y_t \mid c_t). \tag{45}$$

A large positive value indicates that the rollout policy assigns a higher probability to y_t than the frozen reference does. The log-probabilities are computed from logits without additional temperature scaling. Unlike energy and entropy, this signal partly moves with the policy because it contains $\pi_{\theta_{\text{old}}}$.

Signal stability. Table 4 summarizes whether each raw signal changes as the rollout policy updates. The empirical ranking in Sec. 5.6 matches this stability distinction. Energy and entropy are computed from a fixed model, so they provide stable weighting targets throughout training. Policy NLL and log-probability shift depend on the rollout policy, so their values change as the policy is updated. Among the two frozen-reference signals, energy performs better in our experiments. A useful interpretation is that energy keeps scale information from the reference logits, while entropy keeps only the normalized distribution shape. This makes frozen-reference energy a natural fit for ACTFOCUS: it is stable during training and still preserves confidence information useful for prioritizing uncertain action tokens.

D ADDITIONAL EXPERIMENTAL RESULTS

This appendix complements the main benchmark with peak performance, GRPO results, response-length analysis, and targeted checks of model scale and action-weight assignment.

D.1 PEAK PERFORMANCE

Table 5 reports the peak evaluation success reached during training for both PPO and GRPO, complementing the final-step results in Table 2 and Table 6.

Table 5: PPO and GRPO peak performance across four environments. Values are success/purchase rate (%). WebShop reports both purchase rate and strict success. Best results are **bolded**. Results are reported as mean ± sample standard deviation over five random seeds.

Alg.	Method	Sokoban	FrozenLake	Sudoku	WebShop$_{\text{purchase}}$	WebShop$_{\text{strict}}$
		3B	3B	1.5B	3B	3B
PPO	Uniform	28.7 ± 3.4	24.6 ± 12.2	84.2 ± 4.1	98.0 ± 1.0	23.8 ± 0.6
	ActFocus	**37.5** ± 1.9 (+8.8)	**27.0** ± 6.4 (+2.4)	**97.6** ± 1.5 (+13.4)	**99.5** ± 0.3 (+1.5)	**36.7** ± 3.8 (+12.9)
GRPO	Uniform	17.8 ± 3.1	**26.2** ± 2.6	70.9 ± 1.0	**99.8** ± 0.2	44.9 ± 0.3
	ActFocus	**23.2** ± 5.9 (+5.4)	19.1 ± 1.0 (-7.1)	**80.9** ± 0.4 (+10.0)	99.6 ± 0.3 (-0.2)	**51.6** ± 6.1 (+6.7)
		0.5B	0.5B	0.5B	1.5B	1.5B
PPO	Uniform	**43.6** ± 4.4	16.2 ± 3.1	75.4 ± 3.3	94.9 ± 10.9	10.9 ± 0.8
	ActFocus	40.4 ± 1.2 (-3.2)	**20.9** ± 2.7 (+4.7)	**82.0** ± 3.3 (+6.6)	**98.8** ± 1.4 (+3.9)	**27.0** ± 1.2 (+16.1)
GRPO	Uniform	20.7 ± 3.6	23.8 ± 1.8	0.0 (no learning)	**99.3** ± 0.3	49.2 ± 0.4
	ActFocus	**23.4** ± 2.4 (+2.7)	**25.2** ± 3.6 (+1.4)	0.0 (no learning)	98.8 ± 0.9 (-0.5)	**64.5** ± 1.3 (+15.3)

D.2 GRPO FINAL-STEP RESULTS

Table 6 reports final-step GRPO results at the model scales listed in the table, including a GiGPO baseline. The UNIFORM and ACTFOCUS rows provide the GRPO results discussed in Sec. 5.3.

Table 6: GRPO final-step results across four environments at the larger of the two model scales tested per environment. For a controlled comparison, we evaluate GiGPO (Feng et al., 2025) using its official implementation under the same model, environment, interaction horizon, training budget, and evaluation protocol as ActFocus. Success rate (%) is reported at step 200 for Sokoban, FrozenLake, and Sudoku, and at step 100 for WebShop. WebShop reports both purchase rate and strict success. Deltas are relative to Uniform. Uniform values here reflect the peak-to-final degradation under critic-free training (see Sec. 5.3). Best results are **bolded**. Results are reported as mean ± sample standard deviation over five random seeds.

Method	**Sokoban**	**FrozenLake**	**Sudoku**	**WebShop$_{\text{purchase}}$**	**WebShop$_{\text{strict}}$**
	3B	3B	1.5B	3B	3B
GiGPO	15.5 ± 4.4	**19.8** ± 7.4	68.3 ± 10.4	61.3 ± 3.1	43.2 ± 2.9
Uniform	3.9 ± 3.5	12.5 ± 1.6	15.8 ± 3.1	40.6 ± 0.7	14.1 ± 2.2
ActFocus	**23.2** ± 5.9 (+19.3)	6.8 ± 3.1 (-5.7)	**79.5** ± 2.7 (+63.7)	**94.9** ± 8.2 (+54.3)	**51.6** ± 6.1 (+37.5)

D.3 Response Length under GRPO

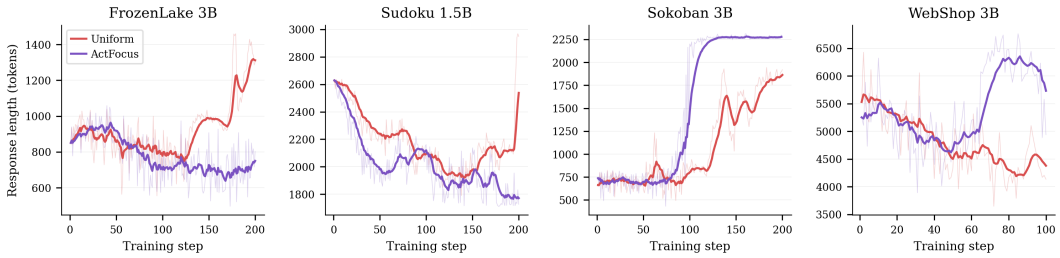


Figure 7: GRPO response length across training steps on FrozenLake 3B, Sudoku 1.5B, Sokoban 3B, and WebShop 3B. Shaded curves denote per-step response length.

Figure 7 provides a complementary view of the stabilization effect discussed in Sec. 5.3. Under GRPO, response length tracks training stability rather than response efficiency. Noisy length curves coincide with unstable policy behavior, while smooth curves indicate that training has settled into a consistent regime, regardless of whether length is rising, falling, or stable. Stability here does not require shorter responses.

D.4 Additional Evaluation at 7B

We extend the evaluation to Qwen2.5-7B-Instruct through a pre-RL token-composition diagnostic on all four environments and an end-to-end PPO comparison on WebShop. These studies examine trajectory structure and training performance separately.

Token composition. We repeat the diagnostic in Table 1, counting only model-generated tokens and excluding environment observations through the loss mask. Table 7 shows that action tokens account for 4.9% to 9.6% of generated tokens, while reasoning tokens account for more than 90% in every environment. Thus, the reasoning–action imbalance remains present at 7B. This diagnostic measures token composition, not the energy–reward correlation at this scale.

Table 7: Pre-RL token composition of Qwen2.5-7B-Instruct. Fractions are computed over model-generated tokens, excluding environment observations.

Environment	**Think (%)**	**Action (%)**
Sokoban	93.4	6.6
FrozenLake	95.1	4.9
Sudoku 4×4	90.5	9.5
WebShop	90.4	9.6

PPO training on WebShop. We train Qwen2.5-7B-Instruct for 100 steps using the same PPO recipe and the default ACTFOCUS setting, $\alpha = 0.1$ and $\beta = 0.5$, without 7B-specific hyperparameter tuning. Table 8 reports peak and final strict success separately. ACTFOCUS improves peak strict success from 43.4% to 55.5% and final strict success from 41.8% to 46.1%, corresponding to gains of 12.1 and 4.3 percentage points. The final-step comparison supports a benefit at 7B without relying only on the best intermediate checkpoint.

Table 8: WebShop strict success (%) with Qwen2.5-7B-Instruct under PPO. Parentheses give the step of the reported peak. Final performance is measured at step 100. Point estimates are shown separately from the five-seed main benchmark.

Method	Peak (step)	Final at step 100
Uniform	43.4 (66)	41.8
ACTFOCUS	**55.5** (80)	**46.1**

D.5 CONTROLLED ACTION-WEIGHT ASSIGNMENT

To test whether performance depends on the association between energy-derived weights and action-token positions, we compare three action-weighting variants on WebShop 3B with PPO for 100 steps. The reasoning-token weight and the remaining training configuration are held fixed within this comparison. *Energy assignment* uses the weights produced by ACTFOCUS. *Uniform action weighting* assigns equal weights within the action span. *Random assignment* permutes the energy-derived weights across action-token positions, preserving the set of weight values while breaking their original correspondence with tokens. This control differs from selecting a random subset of tokens.

Table 9: Controlled action-weight assignment on WebShop 3B with PPO. Values are peak success rates (%), with the corresponding training steps in parentheses. Each metric has its own peak checkpoint. These supplementary point estimates are not the default results in Table 2.

Action-weight assignment	Strict peak (step)	Purchase peak (step)
Energy (ACTFOCUS)	**25.4** (100)	**94.1** (100)
Uniform within action	18.8 (70)	87.5 (60)
Random assignment	3.1 (50)	69.9 (50)

Table 9 shows that energy assignment achieves higher peak strict success than both controls and reaches its peak at the final training step. The randomized control has a much lower peak despite preserving the weight values under permutation. These results support the importance of weight–token correspondence in this experiment. They do not establish that energy measures each token's causal contribution to reward, nor do they separate all possible state-level and token-level explanations.

E LIMITATIONS AND BROADER IMPACT

Our study has several limitations. First, most training experiments use Qwen2.5-Instruct at 0.5B, 1.5B, and 3B. Additional 7B results in Appendix D.4 cover token composition on all four environments and PPO training on WebShop, but not GRPO training at 7B, other model families, or larger scales. Second, α and β require calibration, and an automatic adaptation method during training would be more practical at scale. Finally, frozen-reference energy keeps the credit signal stable and cheap, but its uncertainty landscape does not co-evolve with the policy and may become progressively misaligned with the current decision boundary in late training. A natural extension is to refresh the reference periodically and make the credit signal adaptive, for example by reshaping action weights with the policy's evolving log-probability shift or by coupling energy with token-level importance estimates that track the current rollout distribution.

Our experiments use explicitly separated reasoning and action spans. Applying the same weighting rule through a different interface requires a reliable mask of the model-generated tokens submitted to the environment. Performance on less structured interfaces and broader search or coding workloads

has not been established here. The energy–reward diagnostic and the action-weight controls support an empirical weighting strategy, but do not identify token-level causal credit.

E.1 Broader Impact

This work studies how to allocate gradient credit across tokens in agentic RL training. The contribution is methodological: ActFocus does not change the reward function or what behaviors are reinforced, but only redistributes a fixed advantage signal across tokens within a trajectory. It therefore inherits the safety properties of the surrounding RL pipeline rather than introducing new ones. On the positive side, more sample-efficient and stable agentic RL lowers the cost of training capable agents for beneficial applications such as scientific assistants, accessibility tools, and software engineering automation, and stabilizing GRPO against late-stage degradation is particularly valuable for academic and resource-constrained settings where running many seeds is infeasible. On the negative side, any technique that improves the efficacy of LLM agents could, in principle, accelerate the development of agents whose objectives are misaligned with user or societal interests. We view careful reward design, sandboxed evaluation, and human oversight of deployed agents as appropriate mitigations and consider them complementary to the efficiency improvements we study here. The additional cost of ActFocus is one forward pass through a frozen reference model per rollout, which keeps the method's energy and compute footprint small relative to the underlying RL training.

F Environment Prompts and Example Rollouts

F.1 Prompt Templates

Across environments, we use the same three-role chat template: a fixed `system` message that describes the environment, a per-turn `user` message containing the current state, and an `assistant` message produced by the policy. The system message is assembled from the per-environment instruction together with an optional grid-symbol vocabulary and, when applicable, an enumeration of admissible actions. The user message follows the same general structure across environments: `(Turn` n`):\nState:\n{state}\nYou have {actions_left} actions left. Always output: <think> [Your thoughts] </think> <answer> [your answer] </answer> with no extra text. Strictly follow this format. Max response length: {max_tokens} words (tokens).`

Below, we show, for each of the four environments used in our experiments, the exact `system` prompt emitted by the training pipeline.

F.1.1 Sokoban (SimpleSokoban)

Sokoban System Prompt

```
You're a helpful assistant. 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>

The meaning of each symbol in the state is:
#: wall, _: empty, O: target, √: box on target, X: box, P: player,
S: player on target
Your available actions are:
Up, Down, Left, Right
You can make up to 10 actions, separated by the action separator " || "
```

F.1.2 SUDOKU (SIMPLESUDOKU, 4×4)

Sudoku System Prompt

```
You're a helpful assistant. You are solving a 4x4 Sudoku puzzle.
Fill in the grid so that every row, column, and 2x2 box contains
the numbers 1-4 without repetition.
Initial cells are shown in [brackets] and cannot be modified.
Empty cells are shown as dots (.).
Place numbers one at a time using the format:
<answer>place 3 at row 1 col 2</answer> or <answer>1,2,3</answer>
The environment will provide feedback on valid/invalid moves and
show conflicts if any occur.
```

F.1.3 FROZENLAKE (COORDFROZENLAKE, 4×4, SLIPPERY)

FrozenLake System Prompt

```
You're a helpful assistant. You are solving the FrozenLake puzzle.
The observation includes both a symbol grid and zero-indexed
coordinates for the start, goal, player, and any holes.
Beware that the ice is slippery, so the agent might slide and end
up in an unintended tile.
Respond with a sequence of actions such as
<answer>Left || Up || Up</answer>.

The meaning of each symbol in the state is:
P: player, _: empty, O: hole, G: goal, X: player in hole,
√: player on goal
Your available actions are:
Left, Down, Right, Up
You can make up to 10 actions, separated by the action separator " || "
```

F.1.4 WEBSHOP

The WebShop system prompt contains the task description only; the admissible-action list is inserted into each turn's state string rather than the system prompt itself.

WebShop System Prompt

```
You're a helpful assistant. You are browsing an online shop. Based
on the instruction, buy a product that close to the production
description. You need to search, read the search results, pick a
product, choose the size and color and buy. You should only choose
action from the available actions list provided later. Example
process: I need a gingko light and 20x20 pillow cover that is hand
painted. First search[gingko light 20x20 pillow cover hand painted],
answer format: <answer>search[blanket with fleece throw]</answer>.
Valid answer is search[<keywords>] or click[<clickable>].
```

F.2 MULTI-TURN STATE–ACTION TEMPLATES

Each example below is a verbatim excerpt from an evaluation-time rollout. We preserve the exact `system / user / assistant` role markers, state formatting, and model outputs.

F.2.1 SOKOBAN

Sokoban Rollout Example (Turns 1–2)

Input:

```
system
You're a helpful assistant. 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>

The meaning of each symbol in the state is:
#: wall, _: empty, O: target, √: box on target, X: box, P: player,
S: player on target
Your available actions are:
Up, Down, Left, Right
You can make up to 10 actions, separated by the action separator " || "

user

Turn 1:
State:
######
###P_#
###X_#
###__#
###O_#
######
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 (Turn 1):

```
<think>The player is right next to a box. The box is not on a target
yet. The target is directly above the player.</think>
<answer>Right || Up</answer>
```

Next User message (Turn 2):

```
user
Reward:
-0.2

Turn 2:
State:
######
###_P#
###X_#
###__#
###O_#
######
You have 8 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 (Turn 2):

```
<think>After pushing the box right, the player is now right next to
the target. The box is now on the target.</think>
<answer>Up</answer>
```

F.2.2 SUDOKU

Sudoku Rollout Example (Turn 1)

Input:

```
system
You're a helpful assistant. You are solving a 4x4 Sudoku puzzle.
Fill in the grid so that every row, column, and 2x2 box contains
the numbers 1-4 without repetition.
Initial cells are shown in [brackets] and cannot be modified.
Empty cells are shown as dots (.).
Place numbers one at a time using the format:
<answer>place 3 at row 1 col 2</answer> or <answer>1,2,3</answer>
The environment will provide feedback on valid/invalid moves and
show conflicts if any occur.

user

Turn 1:
State:
==================================================
SUDOKU PUZZLE
==================================================
```

```
[3][1]| [2][4]
[4] . |  .  .
---------
 . [3]| [4] .
 . [4]| [3][2]

Legend: [N]=initial cell, N=user-placed, *N*=conflict, .=empty

VALID NUMBERS FOR EMPTY CELLS:
  - (2,2): [2]
  - (2,3): [1]
  - (2,4): [1, 3]
  - (3,1): [1, 2]
  - (3,4): [1]
  - (4,1): [1]

Progress: 10/16 cells filled (10 initial, 0 placed)
Steps: 0/16
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: 150 words (tokens).
```

Output:

```
<think>Considering the initial placement, the only number that can go
in (2,2) is 2.</think>
<answer>2,2,2</answer>
```

F.2.3 FROZENLAKE

FrozenLake Rollout Example (Turns 1–2)

Input:

```
system
You're a helpful assistant. You are solving the FrozenLake puzzle.
The observation includes both a symbol grid and zero-indexed
coordinates for the start, goal, player, and any holes.
Beware that the ice is slippery, so the agent might slide and end
up in an unintended tile.
Respond with a sequence of actions such as
<answer>Left || Up || Up</answer>.

The meaning of each symbol in the state is:
P: player, _: empty, O: hole, G: goal, X: player in hole,
√: player on goal
Your available actions are:
Left, Down, Right, Up
You can make up to 10 actions, separated by the action separator " || "

user

Turn 1:
State:
Coordinates:
Board size: 4 rows x 4 cols (zero-indexed).
Start: (2, 3)
Goal: (3, 0)
Player: (2, 3)
Holes: (0, 3), (1, 1)
Grid Map:
___O
_O__
___P
G___
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: 120 words (tokens).
```

Output (Turn 1):

```
<think>The player is at position (2, 3) and needs to move towards
the goal at (3, 0). Moving left is the first logical step.</think>
<answer>Left || Left</answer>
```

Next User message (Turn 2):

```
user
Reward:
```

```
0

Turn 2:
State:
Coordinates:
Board size: 4 rows x 4 cols (zero-indexed).
Start: (2, 3)
Goal: (3, 0)
Player: (2, 1)
Holes: (0, 3), (1, 1)
Grid Map:
___O
_O__
_P__
G___
You have 8 actions left. Always output: <think> [Your thoughts]
</think> <answer> [your answer] </answer> with no extra text.
Strictly follow this format. Max response length: 120 words (tokens).
```

Output (Turn 2):

```
<think>After moving left, the player is now at (2, 1). Moving left
again would lead the player into a hole. The next best move is to
move up.</think>
<answer>Up</answer>
```

F.2.4 WEBSHOP

WebShop Rollout Example (Turn 1)

Input:

```
system
You're a helpful assistant. You are browsing an online shop. Based
on the instruction, buy a product that close to the production
description. You need to search, read the search results, pick a
product, choose the size and color and buy. You should only choose
action from the available actions list provided later. Example
process: I need a gingko light and 20x20 pillow cover that is hand
painted. First search[gingko light 20x20 pillow cover hand painted],
answer format: <answer>search[blanket with fleece throw]</answer>.
Valid answer is search[<keywords>] or click[<clickable>].

user

Turn 1:
State:
Instruction: Find me machine wash, wash cold women's fashion hoodies
& sweatshirts for dry clean, tumble dry with color: vintage camo,
and size: large, and price lower than 70.00 dollars.We must buy a
product within 10 actions. It doesn't have to match perfectly with
description.
Search term should not include details like size, color.
Never search for more than 2 times.
Do not be too strict about the description, it's more important to
buy one that is close enough within action limit.
Prioritize click a product in the current page over going to next
page.
Almost never click[next >] for more than 2 times.Almost never
click[< prev] unless you are sure the product is on one of the
previous pages.
If you have less than 3 actions left, just buy the first product you
see in the current page.
If an matching option exists, make sure to click[size] then
click[color], one at a time, before click[buy now], but don't have
to if only 1 action left, in that case you just click[buy now].
Never click description.
You must choose from these actions:search[<content>].
You have 9 actions left. Always output: <think> [Your thoughts]
</think> <answer> [your answer] </answer> with no extra text.
Strictly follow this format. Max response length: 200 words (tokens).
```

Output:

```
<think>Based on the instruction, I need to find a machine washable,
wash cold women's hooded sweatshirt that is vintage camo pattern and
in large size, with a price lower than 70.00 dollars. The search
term should not include size or color.</think>
```

```
<answer>search[women hooded sweatshirt vintage camo large machine
wash cold]</answer>
```