REWARDING BELIEFS, NOT ONLY ACTIONS: BELIEF-CONSISTENT REINFORCEMENT LEARNING FOR LONG-HORIZON AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning from verifiable rewards (RLVR) is a promising paradigm for improving large language model (LLM) agents on long-horizon interactive tasks. However, in partially observable environments, incomplete observations cause agent beliefs to drift over time, while delayed rewards obscure the causal impact of intermediate decisions, exacerbating temporal credit assignment challenges. To address this, we propose **ReBel** (Reward Belief), a process-level reinforcement learning algorithm that explicitly models structured belief states to summarize interaction history and guide subsequent policy learning. ReBel introduces belief-consistency supervision, converting discrepancies between predicted beliefs and observed feedback into dense self-supervised signals without requiring external step-wise annotations or verifiers. It also employs belief-aware grouping to compare trajectories under similar belief states, yielding more robust and lower-variance advantage estimates. We evaluate ReBel on challenging long-horizon benchmarks, including ALFWorld and WebShop. ReBel improves task success by up to 20.4% over the episode-level baseline GRPO and increases sample efficiency by $2.1\times$. These results suggest that belief-aware self-supervision is a promising direction for reliable long-horizon decision-making under partial observability.

1 INTRODUCTION

Large language models (LLMs) are increasingly being deployed as autonomous agents for long-horizon interactive tasks such as embodied instruction following and web navigation (Xi et al., 2025; Cao et al., 2025; Zhou et al., 2024a). These settings require more than fluent generation: agents must maintain an evolving understanding of the environment, plan over multiple steps, and adapt robustly under uncertainty (Duan et al., 2025; Oh et al., 2026; Qi et al., 2025). Reinforcement learning from verifiable rewards (RLVR) has emerged as a promising paradigm for improving such agents, because it optimizes policies directly against objective and environment-grounded outcomes rather than relying on expensive human annotations or potentially unreliable external judges (Lambert et al., 2025; Guo et al., 2025; Su et al., 2026). Compared with LLM-as-a-judge or manually specified intermediate labels, RLVR provides scalable supervision with clear task-level semantics (Zheng et al., 2023; Gu et al., 2026), making it particularly attractive for training agents that must act reliably in open-ended environments (Qian et al., 2025; Feng et al., 2025).

Despite its promise, RLVR remains difficult in partially observable environments, where agents must continuously infer latent context from incomplete observation histories rather than the full state (Chen et al., 2025). Small inference errors can accumulate over time, inducing belief drift: a growing mismatch between the agent's internal state estimate and the environment (Liu et al., 2025). In Figure 6, an agent may confidently believe that the key is in a box, only to observe after opening it that the box is actually empty. If the policy fails to revise this belief, it may repeatedly inspect the wrong location and drive the trajectory progressively away from the goal (Duan et al., 2025). Delayed terminal rewards further obscure which intermediate belief errors caused the failure, making temporal credit assignment especially challenging. Thus, effective policies must learn not only which actions to take, but also when their latent state estimates should be trusted (Sun et al., 2024).

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.

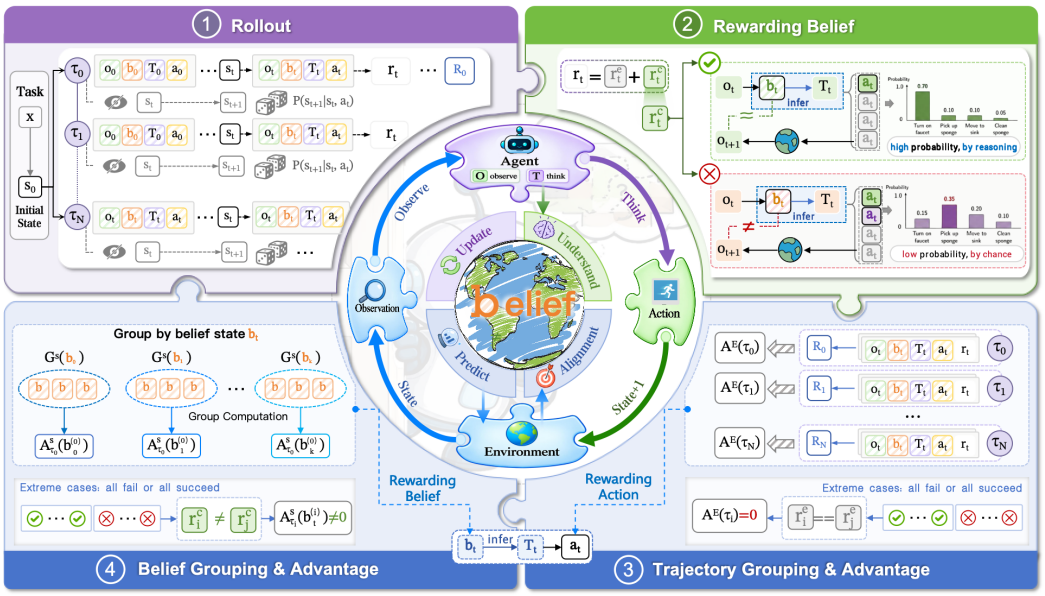

Figure 1: **Overview of ReBel.** ReBel learns belief-aware policies for partially observable long-horizon tasks by making latent belief explicit and decomposing policy generation into belief, think, and action. It turns sparse terminal rewards into step-wise belief consistency feedback and performs belief-anchor grouping to support stable step-level advantage estimation.

Existing RLVR methods are ill-suited to this setting. Episode-level rewards are often too sparse to correct intermediate belief errors, especially when failures are only revealed at the end of a long trajectory (Chen et al., 2026; Fu et al., 2025). Step-wise supervision or external verifiers can provide denser feedback, but they are costly, difficult to scale, and often unavailable in open-ended interactive tasks (Choi et al., 2026). More importantly, most existing approaches optimize trajectories primarily through return-based signals, without explicitly modeling the agent's evolving belief state (Duan et al., 2025; Oh et al., 2026). This makes it difficult to distinguish failures caused by poor action selection from failures caused by incorrect state estimation (Lidayan et al., 2026). In partially observable long-horizon tasks, these two factors are tightly coupled, and ignoring belief dynamics can lead to unstable optimization and poor sample efficiency (Kim et al., 2025; Guo et al., 2025).

To address this challenge, we propose **ReBel** (**Re**ward **Bel**ief), a process-level reinforcement learning framework for partially observable long-horizon tasks. The core insight is that effective credit assignment in partially observable settings requires more than reward signals and demands explicit supervision of belief dynamics that bridge observations and actions. Failures in long-horizon tasks often stem from persistent internal misunderstandings rather than poor action selection. Concretely, ReBel realizes this idea through two complementary design layers. First, it maintains a structured belief representation that summarizes the agent's inferred environment state from past interaction history. Rather than attempting to reconstruct a complete latent world model, this representation captures task-relevant state predicates that are sufficient for decision making. Second, ReBel introduces belief-consistency supervision, which converts mismatches between predicted beliefs and subsequent environment observations into dense self-supervised feedback. This provides process-level correction signals without requiring step-wise annotations or external verifiers. In addition, we design a belief-aware grouping strategy that compares trajectories under similar belief states, enabling more robust advantage estimation and reducing optimization variance. Together, these components form a closed iterative learning loop in which more accurate beliefs induce more informative comparisons, which in turn stabilize policy updates and further improve belief quality.

We evaluate ReBel on two partially observable long-horizon benchmarks: ALFWorld (Shridhar et al., 2021) for embodied task planning under incomplete observations and WebShop (Yao et al., 2022) for goal-driven web interactions under sparse delayed feedback. Using Qwen2.5-1.5B-Instruct (Qwen et al., 2025), ReBel achieves success rates of $93.2 \pm 4.1\%$ on ALFWorld (Shridhar et al., 2021) and $75.1 \pm 2.7\%$ on WebShop (Yao et al., 2022), improving over the strongest episode-level baseline

GRPO (Shao et al., 2024) by 20.4 percentage points and over the strongest step-level method $\text{GiGPO}_{\text{w/ std}}$ (Feng et al., 2025) by 6.5 and 7.7 percentage points on ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022), respectively. ReBel reaches GRPO's (Shao et al., 2024) final performance in approximately 45 training iterations, yielding a $2.1\times$ sample-efficiency gain, all without requiring dense human annotations or external step-wise verifiers. These results validate belief-aware RLVR as an effective framework for optimizing LLM agents in partially observable long-horizon settings. In summary, our main contributions are threefold:

1. We identify belief drift under partial observability as a key failure mode of RLVR in long-horizon interactive tasks, where delayed rewards exacerbate temporal credit assignment.
2. We propose ReBel, a process-level reinforcement learning framework that combines structured belief representation, belief-consistency supervision, and belief-aware grouping to provide dense self-supervised feedback without external step-wise verifiers.
3. We demonstrate substantial gains on ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022), showing that ReBel improves both the task success rate and sample efficiency over strong episode-level and step-level baselines.

2 Related Work

Reinforcement learning for LLM agents. As LLMs are increasingly deployed as autonomous agents in program synthesis, embodied control, device interaction, and web navigation, reinforcement learning (RL) has become a central paradigm for improving their decision-making. Early RLHF methods, such as InstructGPT (Ouyang et al., 2022), demonstrated that preference-based optimization can effectively align model outputs with human intent. Recent work has shifted from single-turn response optimization toward trajectory-level learning in interactive environments such as ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022). To improve exploration and long-horizon reasoning, methods such as RAP (Hao et al., 2023) and Agent Q (Putta et al., 2024) incorporate Monte Carlo tree search (MCTS) into training or inference, while ArCHer (Zhou et al., 2024b) introduces history-aware value estimation for agent trajectories. In parallel, critic-free optimization methods such as GRPO (Shao et al., 2024) and RLOO (Ahmadian et al., 2024) reduce training overhead. Recent systems including DigiRL (Bai et al., 2024), WebRL (Qi et al., 2025), RAGEN (Wang et al., 2025), and LOOP (Chen et al., 2025) further demonstrate the promise of RL for end-to-end device control and web interaction. Despite these advances, most existing methods still rely primarily on terminal rewards or assume near-complete observability, which makes them vulnerable to severe credit assignment challenges in partially observable long-horizon tasks.

Process supervision and belief modeling. Process supervision densifies learning signals by guiding intermediate steps rather than just final outcomes (Uesato et al., 2022; Lightman et al., 2024). Recent methods automate step-level reward construction to reduce reliance on expensive human annotations (Wang et al., 2024; Cui et al., 2025; Xiong et al., 2024). Agent-oriented approaches like GiGPO further refine step-level optimization by grouping trajectories to estimate fine-grained advantages (Feng et al., 2025; Liu et al., 2026; Zhuang et al., 2025). However, these methods typically condition grouping and rewards solely on current observations. In partially observable environments, this ignores the agent's latent state, degrading grouping quality and advantage normalization. Parallel works address partial observability via retrieval and symbolic augmentation (Guu et al., 2020; Mai et al., 2022; Mialon et al., 2023), or intrinsic motivation (Schmidhuber, 1991; Burda et al., 2019). While improving representation or exploration, they do not couple belief evolution with process-level optimization. ReBel addresses this gap by turning belief consistency into a dense self-supervised signal, aligning policy optimization with latent state tracking in long-horizon tasks.

3 Method

Reinforcement learning from verifiable rewards (RLVR) has significantly improved the performance of large language model (LLM) agents on long-horizon tasks. However, extending this paradigm to partially observable environments remains challenging, primarily because the agent's internal reasoning process is not directly transparent. Outcome-based optimization such as vanilla GRPO (Shao et al., 2024) provides only sparse terminal feedback, making it difficult to distinguish a logically

sound trajectory from one that merely succeeds by chance. Action-centered process reward models alleviate this sparsity to some extent, but they often remain insensitive to the underlying belief state and may therefore reinforce actions driven by incorrect internal assumptions. We refer to this phenomenon as **credit drift**, where spurious successes contaminate credit assignment and lead to high-variance updates as well as poor generalization.

To address this issue, we propose **ReBel** (**Re**ward **Bel**ief), a framework that shifts process supervision from what the agent does to why it believes an action should work. ReBel introduces a two-level supervision structure: (i) **consistency-based supervision**, which penalizes mismatches between the agent's predicted belief state and subsequent observations; and (ii) **belief-anchored advantage estimation**, which compares actions only within semantically aligned belief clusters. Together, these components enable process-level optimization that explicitly aligns policy learning with latent-state tracking in partially observable long-horizon tasks; Figure 1 summarizes the pipeline, Algorithm 1 gives the full training procedure, and Appendix E states the corresponding formal analysis.

3.1 Belief-Structured Interaction in POMDPs

We consider the general setting in which large language model (LLM) agents perform multi-step tasks in a partially observable Markov decision process (POMDP). The process is defined by the tuple $(\mathcal{S}, \mathcal{A}, \Omega, \mathcal{T}, \mathcal{O}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ denotes the latent state space, $\mathcal{A}$ the action space, Ω the observation space, $\mathcal{T}$ the transition kernel, $\mathcal{O}$ the observation model, $\mathcal{R}$ the reward function, and γ the discount factor. Given a task instruction $x \in \mathcal{X}$, the agent receives a local observation $o_t \in \Omega$ at each discrete time step $t = 1, \ldots, T$. Since the underlying environment state $s_t \in \mathcal{S}$ is not directly accessible, the agent must reason over a history trace $h_t = (o_1, a_1, o_2, a_2, \ldots, a_{t-1}, o_t)$, which records both observations and prior decisions. At each interaction step, the policy π_θ generates a structured output field $y_t = (b_t, z_t, a_t)$, where b_t denotes a structured belief, z_t an intermediate thought state, and $a_t \in \mathcal{A}$ an executable action. The environment then returns a scalar reward $r_t \in \mathbb{R}$. In realistic long-horizon scenarios, reward signals are typically extremely sparse and delayed, often appearing only at the end of an episode as a terminal indicator of success or failure. When a single trajectory spans thousands of tokens, credit assignment is no longer merely a sparse-reward problem; rather, it becomes an attribution problem across a long generation chain and multiple internal decisions. In particular, it is often difficult to determine whether the final outcome should be attributed to a specific observation interpretation, an intermediate reasoning step, or a particular action choice.

To address this challenge, ReBel does not directly map history to actions in an end-to-end manner. Instead, it explicitly reconstructs each generation step as a Belief-Think-Action process. Specifically, before executing an action, the agent first constructs a structured belief b_t from the history h_t and task instruction x, where b_t explicitly encodes the agent's assessment of the current environment state. The agent then performs internal reasoning conditioned on b_t to produce an intermediate thought state z_t, and finally outputs an executable action a_t based on both b_t and z_t. This process is factorized as

$$\pi_\theta(b_t, z_t, a_t \mid h_t, x) = \pi_\theta(b_t \mid h_t, x)\, \pi_\theta(z_t \mid h_t, x, b_t)\, \pi_\theta(a_t \mid h_t, x, b_t, z_t).$$

Here, $b_t \subseteq P = \{p_1, \ldots, p_K\}$ denotes a task-relevant belief set composed of predicates that are derived automatically from the environment and task specification (Appendix B.1), rather than curated by hand. Note that b_t is not the Bayesian posterior belief in the standard POMDP sense, but rather a symbolic approximation of the current environment state, used to explicitly represent the agent's state assessment. Unlike methods that compress internal reasoning into a single latent variable, we explicitly model belief b_t, thought z_t, and action a_t, thereby making the intermediate cognitive process directly supervisable. Accordingly, training signals can come not only from trajectory-level outcomes, but also from intermediate beliefs, reasoning processes, and action decisions. By organizing the generation process as $b_t \to z_t \to a_t$, ReBel converts the credit assignment problem in long-horizon tasks into local constraints on the consistency between intermediate beliefs and subsequent behaviors, thereby providing a more stable learning signal.

3.2 Belief Consistency Supervision

The effectiveness of explicit belief lies not in its static representation, but in whether it can remain consistent with evidence generated through environmental evolution over time. Based on this insight, ReBel introduces an observation-based dense supervision mechanism that decomposes sparse terminal feedback into step-wise belief consistency evaluations, enabling belief quality to be optimized at

each time step. Let $P = \{p_1, \ldots, p_K\}$ denote the set of task-relevant predicates. We represent the belief at time step t as a binary vector over the predicate space, $b_t \in \{0,1\}^K$, where $b_{t,k} = 1$ indicates that the agent believes predicate p_k to be true at time t, and $b_{t,k} = 0$ indicates it to be false. To characterize the consistency between belief and observations, we define a consistency indicator function $C_k(b_t, a_t, o_{t+1}) \in \{0,1\}$, which determines whether the judgment on predicate p_k encoded in b_t can be supported by the next observation o_{t+1} after executing action a_t. Here, action a_t influences observations indirectly by inducing state transitions in the environment; therefore, C_k measures whether the current belief remains supported by evidence after environmental evolution.

Since not all predicates are verifiable at every time step, we further introduce an observability mask $m_t \in \{0,1\}^K$, where $m_{t,k} = 1$ indicates that predicate p_k can be checked against observation o_{t+1}, and $m_{t,k} = 0$ indicates that the predicate is temporarily unverifiable at the current step. Based on this mask, the belief consistency reward is defined as

$$r_t^{\text{cons}} = \frac{\sum_{k=1}^{K} m_{t,k} C_k(b_t, a_t, o_{t+1})}{\sum_{k=1}^{K} m_{t,k}} \quad \text{if} \quad \sum_{k=1}^{K} m_{t,k} > 0.$$

When $\sum_{k=1}^{K} m_{t,k} = 0$, the consistency reward is undefined. This treatment avoids interpreting an unverifiable state as inconsistent, ensuring that the supervision signal is derived only from observable evidence. In other words, a time step contributes to consistency supervision only when at least one predicate is checkable; for belief items that are temporarily unobservable, no training bias is imposed.

To handle predicates that become verifiable only after a delay, we maintain a pending belief buffer that stores historical belief items for which no validation evidence has yet been obtained. We define

$$\mathcal{U}_t = \left\{ (p_k, t') \,\middle|\, b_{t',k} = 1,\ t' \le t,\ \sum_{\tau=t'}^{t} m_{\tau,k} = 0 \right\},$$

where $\mathcal{U}_t$ denotes the set of belief items that have not yet been validated by observation up to time step t. If a predicate is not observable when it is generated, the corresponding belief is buffered and excluded from reward computation. Once the predicate becomes observable later, its consistency signal is assigned to the original generation step t'. This mechanism enables temporal credit assignment for delayed evidence beyond single-step observability.

3.3 Belief-Anchor Step-wise Advantage

While group-based reinforcement learning reduces variance through relative benchmarking across trajectories, relying solely on terminal feedback in partially observable long-horizon tasks leads to severe **credit drift**, making it difficult to identify the true contribution of specific steps within a sequence. Although introducing step-wise advantage estimation could refine process-level supervision, it encounters the **singleton group problem** in POMDP scenarios: since the environment state s is not directly observable and long-horizon trajectories are highly unique in physical space, different trajectories rarely encounter identical physical states at the same time step. Consequently, state-based grouping often collapses into isolated samples, causing local relative advantage signals to fail due to a lack of comparable benchmarks.

To overcome this bottleneck, ReBel shifts the benchmark for credit assignment from the unobservable physical state s to an explicit **belief anchor** b. The core logic of this shift is that, in partially observable environments, the prerequisite for evaluating action quality is no longer "external consistency" but rather "internal cognitive equivalence." As long as the agent's belief predicate sets b_t are logically equivalent, action choices across different trajectories share a common semantic background for benchmarking. Specifically, let the i-th trajectory be $\tau_i = \{(o_{i,t}, b_{i,t}, z_{i,t}, a_{i,t}, r_{i,t}^{\text{env}}, r_{i,t}^{\text{cons}})\}_{t=1}^{T_i}$. We define the step-wise total reward as $r_{i,t}^{\text{tot}} = r_{i,t}^{\text{env}} + \eta r_{i,t}^{\text{cons}}$, and its corresponding step-wise discounted return as $G_{i,t}^{\text{step}} = \sum_{k=t}^{T_i} \gamma^{k-t} r_{i,k}^{\text{tot}}$. This return not only reflects the action's contribution to the final task goal but also captures its immediate maintenance of cognitive accuracy through $r_{i,t}^{\text{cons}}$.

Next, we leverage the symbolic representation of the predicate space P to aggregate trajectory nodes—which are otherwise isolated in physical state space—into semantically consistent comparison groups. For a belief anchor $\tilde{b} \in \{0,1\}^K$, its corresponding step-wise group is defined as $\mathcal{G}_S(\tilde{b}) = \{(i,t) \mid$

$b_{i,t} = \tilde{b}\}$. Since belief representations are semantic abstractions of complex physical states, this anchoring mechanism significantly increases the coverage of comparable samples, thereby resolving the singleton group problem. On this basis, we define the belief-anchored step-wise advantage as:

$$A_S(i,t) = \begin{cases} 0, & |\mathcal{G}_S(b_{i,t})| = 1 \\ \frac{G_{i,t}^{\text{step}} - \mu(\mathcal{G}_S(b_{i,t}))}{\sigma(\mathcal{G}_S(b_{i,t})) + \epsilon}, & \text{otherwise} \end{cases} \tag{1}$$

where μ and σ denote the mean and standard deviation of returns within the group $\mathcal{G}_S$, respectively. This mechanism achieves the following core advantages through cognitive benchmarking:

- **Explicit Decoupling of Decision and Cognition:** By comparing actions within the same belief anchor, the model can tell whether low return comes from wrong reasoning or from a poor action choice. This helps the agent learn the best action under its current belief.
- **Learning Resilience under Sparse Feedback:** When the environment reward is zero for all samples in a group, the relative advantage still captures small differences in belief maintenance. This lets the model strengthen actions that keep the belief correct and reduce belief drift.

In summary, Belief-Anchor Step-wise Advantage transforms credit assignment from blind competition over outcomes into precise selection under logically equivalent conditions. It provides a more stable process-level supervision signal that aligns with the nature of decision-making for long-horizon agents in partially observable environments.

3.4 Belief-aware Policy Optimization

ReBel optimizes a belief-aware policy by combining trajectory-level outcome signals with belief-anchor step-wise advantages. For the i-th trajectory τ_i and time step $t = 1, \ldots, T_i$, we define the total advantage as $A_{i,t}^{\text{tot}} = A_E(\tau_i) + \omega A_S(i,t)$, where $A_E(\tau_i)$ captures the relative terminal quality of the trajectory within its group, $A_S(i,t)$ measures the relative quality of the current step under the same belief anchor, and $\omega \geq 0$ balances the two terms.

$$\begin{aligned} J(\theta) = \mathbb{E}_{\tau_i \sim \pi_{\theta_{\text{old}}}} \Bigg[\frac{1}{T_i} \sum_{t=1}^{T_i} \Big(& \min\left(\rho_{i,t}(\theta) A_{i,t}^{\text{tot}}, \text{clip}(\rho_{i,t}(\theta), 1-\epsilon, 1+\epsilon) A_{i,t}^{\text{tot}}\right) \\ & - \beta D_{\text{KL}}\big(\pi_\theta(\cdot \mid h_{i,t}) \| \pi_{\text{ref}}(\cdot \mid h_{i,t})\big)\Big) \Bigg]. \end{aligned} \tag{2}$$

Here, ϵ controls the clipping range and β regularizes the deviation from the reference policy π_{ref}. The importance ratio is defined over the joint generation of belief, thought, and action as

$$\rho_{i,t}(\theta) = \frac{\pi_\theta(b_{i,t}, z_{i,t}, a_{i,t} \mid h_{i,t})}{\pi_{\theta_{\text{old}}}(b_{i,t}, z_{i,t}, a_{i,t} \mid h_{i,t})}. \tag{3}$$

Using the Belief-Think-Action factorization, $\pi_\theta(b_{i,t}, z_{i,t}, a_{i,t} \mid h_{i,t}) = \pi_\theta(b_{i,t} \mid h_{i,t}) \pi_\theta(z_{i,t} \mid h_{i,t}, b_{i,t}) \pi_\theta(a_{i,t} \mid h_{i,t}, b_{i,t}, z_{i,t})$, this objective propagates learning signals to belief formation, internal reasoning, and action selection, rather than to the final action alone. As a result, ReBel assigns credit not only to successful outcomes, but also to belief-consistent intermediate decisions, enabling temporally grounded credit assignment in partially observable long-horizon tasks.

4 Experiments

We structure our experiments around three questions: (i) whether ReBel outperforms strong baselines in partially observable environments; (ii) how much each of its main components contributes to the performance; and (iii) whether ReBel improves sample efficiency and mitigates belief drift in long-horizon decision making.

Table 1: **Performance comparison on ALFWorld and WebShop.** ALFWorld reports per-task and overall success rates (%); WebShop reports the average score and task success rate (%). The RL-trained methods use Qwen2.5-1.5B-Instruct (Qwen et al., 2025) as their backbone, and their results are reported as mean ± standard deviation over three random seeds. **Bold** indicates the best result among RL-trained methods. † denotes the belief-annotated SFT cold start used before ReBel's RL stage; the supervised-only controls and incremental ablations are reported in Table 2.

Paradigm	Method	ALFWorld							WebShop	
		Pick	Look	Clean	Heat	Cool	Pick2	Overall	Score	SR
Closed-source frontier models (zero-shot reference)										
Prompt	GPT-4o	75.3	60.8	31.2	56.7	21.6	49.8	48.0	31.8	23.7
Prompt	Gemini-2.5-Pro	92.8	63.3	62.1	69.0	26.6	58.7	60.3	42.5	35.9
Prompt-based agents (Qwen2.5-1.5B-Instruct)										
Prompt	Qwen2.5	5.9	5.5	3.3	9.7	4.2	0.0	4.1	23.1	5.2
Prompt	ReAct	17.4	20.5	15.7	6.2	7.7	2.0	12.8	40.1	11.3
Prompt	Reflexion	35.3	22.2	21.7	13.6	19.4	3.7	21.8	55.8	21.9
RL-trained agents (Qwen2.5-1.5B-Instruct, 3 seeds)										
RL	PPO	$64.8_{\pm 3.5}$	$40.5_{\pm 6.9}$	$57.1_{\pm 4.9}$	$60.6_{\pm 6.6}$	$46.4_{\pm 4.0}$	$47.4_{\pm 1.9}$	$54.4_{\pm 3.1}$	$73.8_{\pm 3.0}$	$51.5_{\pm 2.9}$
RL	RLOO	$88.3_{\pm 3.0}$	$52.8_{\pm 8.6}$	$71.0_{\pm 5.9}$	$62.8_{\pm 8.7}$	$66.4_{\pm 5.5}$	$56.9_{\pm 4.7}$	$69.7_{\pm 2.5}$	$73.9_{\pm 5.6}$	$52.1_{\pm 6.7}$
RL	GRPO	$85.3_{\pm 1.5}$	$53.7_{\pm 8.0}$	$84.5_{\pm 6.8}$	$78.2_{\pm 7.9}$	$59.7_{\pm 5.0}$	$53.5_{\pm 5.6}$	$72.8_{\pm 3.6}$	$75.8_{\pm 3.5}$	$56.8_{\pm 3.8}$
RL	$\text{GiGPO}_{\text{w/std}}$	$94.4_{\pm 5.9}$	$67.5_{\pm 4.6}$	$\mathbf{94.8}_{\pm 3.8}$	$94.4_{\pm 7.8}$	$79.8_{\pm 4.7}$	$76.4_{\pm 5.4}$	$86.7_{\pm 1.7}$	$83.1_{\pm 1.6}$	$65.0_{\pm 3.2}$
RL	$\text{GiGPO}_{\text{w/o std}}$	$\mathbf{96.0}_{\pm 1.4}$	$76.5_{\pm 3.9}$	$91.8_{\pm 5.5}$	$91.3_{\pm 6.3}$	$71.7_{\pm 8.4}$	$79.5_{\pm 7.7}$	$86.1_{\pm 4.7}$	$\mathbf{83.5}_{\pm 1.8}$	$67.4_{\pm 4.5}$
RL	**ReBel (Ours)**†	$91.8_{\pm 2.1}$	$\mathbf{84.2}_{\pm 4.3}$	$91.3_{\pm 3.1}$	$\mathbf{95.8}_{\pm 4.3}$	$\mathbf{84.8}_{\pm 4.2}$	$\mathbf{96.5}_{\pm 3.2}$	$\mathbf{93.2}_{\pm 4.1}$	$79.8_{\pm 2.8}$	$\mathbf{75.1}_{\pm 2.7}$

4.1 Experimental Setup

We evaluate ReBel on ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022), two representative benchmarks for partially observable decision making. We compare against closed-source frontier models, prompt-based open-source agents, and RL-trained baselines across zero-shot prompting, in-context reasoning, and policy optimization settings. All methods use Qwen2.5-1.5B-Instruct (Qwen et al., 2025) as the backbone. ReBel uses an SFT cold start before RL to stabilize the structured `<belief>` output. Unless otherwise specified, results are reported as mean ± standard deviation. Full details, hyperparameters, and prompts are in Appendices A and B (Figures 4, 5).

4.2 Main Results

Table 1 shows that ReBel establishes a new performance frontier, achieving an overall success rate of $93.2 \pm 4.1\%$ on ALFWorld (Shridhar et al., 2021) and $75.1 \pm 2.7\%$ on WebShop (Yao et al., 2022). Compared with the strongest episode-level baseline, GRPO (Shao et al., 2024), this corresponds to improvements of $+20.4$ and $+18.3$ percentage points, respectively. Although closed-source frontier models such as Gemini-2.5-Pro and GPT-4o exhibit strong general reasoning capabilities, their performance on these long-horizon partially observable tasks remains substantially lower than that of RL-trained agents, suggesting that generic zero-shot reasoning is insufficient for maintaining state coherence over long trajectories. Prompt-based agents such as ReAct and Reflexion perform even worse, indicating that without parameter updates, errors accumulate rapidly in these environments.

ReBel consistently outperforms the strongest step-level baseline, $\text{GiGPO}_{\text{w/ std}}$ (Feng et al., 2025), by $+6.5$ points on ALFWorld (Shridhar et al., 2021) and $+7.7$ points on WebShop (Yao et al., 2022). This gain suggests that belief-based credit assignment provides a more semantically stable learning signal than observation-hash grouping, which can conflate distinct latent states under surface-level similarity. In contrast, ReBel propagates supervision through structured belief representations, enabling the model to distinguish temporally separated but semantically related decisions. The benefits of ReBel are further reflected in its sample efficiency and robustness on harder tasks. As shown in Figure 2(a), ReBel matches the final performance of GRPO (Shao et al., 2024) after only 45 training iterations, corresponding to a $2.1\times$ improvement in sample efficiency. Moreover, Figure 2(b) and (c) show that the performance gap between ReBel and GRPO grows with task difficulty and trajectory length. In the most challenging ALFWorld task, **Pick2**, which requires the longest reasoning chain, ReBel achieves a success rate of 96.5%, outperforming the best GiGPO (Feng et al., 2025) variant by $+17.0$ points. Overall, these results indicate that structured belief tracking is a key factor behind ReBel's gains in partially observable long-horizon decision making.

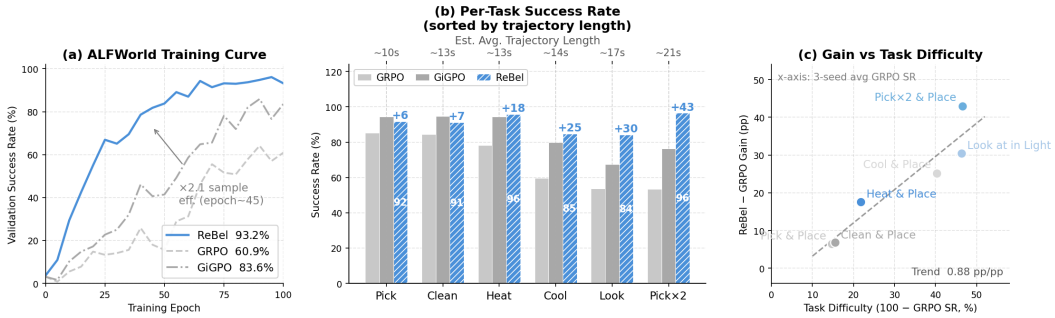

Figure 2: **Training dynamics and per-task performance.** (a) ALFWorld (Shridhar et al., 2021) training curves. REBEL reaches the final GRPO (Shao et al., 2024) performance after roughly 45 iterations, corresponding to an approximate 2.1× improvement in sample efficiency. (b) Per-task success rates on ALFWorld, sorted by estimated trajectory length; Δ denotes the improvement of REBEL over GRPO. (c) The gain of REBEL over GRPO increases with task difficulty, indicating that structured belief tracking is more beneficial in harder partially observable tasks.

4.3 ABLATION STUDY

Table 2 decomposes the performance gains of REBEL into three key factors: explicit representation, fine-grained credit assignment, and consistency supervision. Moving from vanilla GRPO (Shao et al., 2024) (B0) to the belief-prompted variant (B1) yields a 17.2 percentage-point improvement, indicating that making latent state tracking explicit through structured reasoning alleviates the burden of encoding history in internal memory. However, although B1 improves the representational form, it still relies primarily on high-variance trajectory-level rewards; as the task horizon increases, such rewards become increasingly insufficient for providing precise optimization guidance.

Table 2: **Supervised-only controls and incremental ablations on ALFWorld.** The upper section evaluates SFT models without RL optimization. The lower section follows the incremental B0–B3 configurations, successively adding belief prompting, grouping with step-level advantages, and a belief reward.

Method	Belief Prompt	Group	Step Adv.	Belief Reward	ALFWorld (%)
SFT only	×	×	×	×	3.6
Belief SFT only	✓	×	×	×	3.1
B0: GRPO	×	×	×	×	60.9
B1: + Prompt	✓	×	×	×	78.1
B2: + Group	✓	✓	✓	×	93.0
B3: REBEL	✓	✓	✓	✓	**96.9**

The most substantial improvement occurs with B2 (93.0%). This variant mitigates credit diffusion in long-horizon tasks through belief-aligned semantic grouping and step-wise advantage (A_{step}). Specifically, by providing dense optimization signals over belief equivalence classes, the model can more accurately identify critical intermediate subgoals. As shown in Figure 2b, this fine-grained feedback is particularly effective in long-horizon tasks such as **Pick2 & Place**, where trajectory-level rewards are often diluted by noise from lengthy action sequences. This mechanism also explains the 2.1× gain in sample efficiency observed during the early stages of training in Figure 2a.

The full REBEL model (B3, 96.9%) stabilizes the reasoning–action loop through the auxiliary belief reward r^{bel}. This reward serves as an alignment anchor, preventing reasoning drift—a phenomenon in which the agent may produce the correct action based on hallucinated state representations. By continuously constraining the policy to remain grounded in the state space defined by environmental predicates, r^{bel} improves robustness in partially observable settings, with particularly clear benefits in tasks with greater observation sparsity and deeper reasoning requirements (see Figure 2c). Overall, these results show that the performance gains of REBEL arise from the synergy of structured representation (B1), fine-grained credit assignment (B2), and grounded supervision (B3).

4.4 GROUPING QUALITY AND BELIEF-QUALITY DIAGNOSTICS

Grouping quality fundamentally determines the reliability of credit assignment and overall training efficiency. As Figure 3(a) shows, GiGPO (Feng et al., 2025) produces a high fraction of singleton groups, which weakens step-wise normalization and makes advantage estimates susceptible to noise.

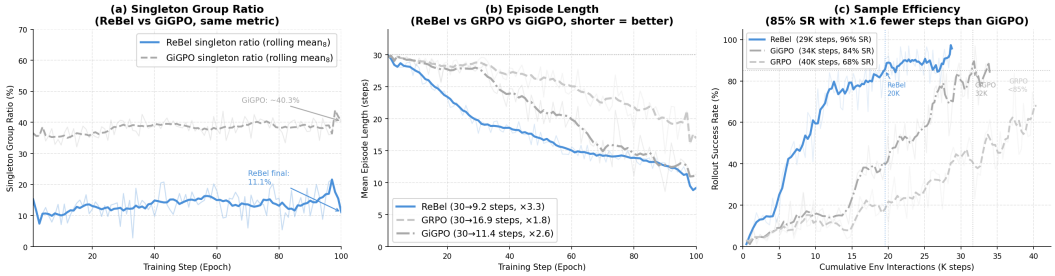

Figure 3: **Grouping quality and training efficiency.** (a) Singleton group ratio for REBEL and GiGPO (Feng et al., 2025) during training. (b) Average episode length on ALFWorld (Shridhar et al., 2021). REBEL reduces it from 29.9 to 9.2 steps, a 3.2× reduction. (c) Success rate versus cumulative environment interactions (environment steps, from the smoothed curves). REBEL reaches the 85% rollout threshold after about 20K interactions, whereas GiGPO requires about 32K (1.6× more); GRPO (Shao et al., 2024) does not reach this threshold within the displayed budget.

In contrast, REBEL maintains a consistently low singleton ratio, continuously forming stable and semantically coherent belief-based groups. This superior grouping directly translates into higher task execution efficiency (Figure 3(b)): REBEL reduces the average ALFWorld (Shridhar et al., 2021) episode length from 29.9 to 9.2 steps (a 3.2× reduction), enabling the policy to complete tasks with much more compact action sequences compared to GiGPO and GRPO (Shao et al., 2024). At the training level, this advantage manifests as higher sample efficiency and faster convergence (Figure 3(c)). REBEL reaches the 85% rollout threshold with roughly 1.6× fewer environment steps than GiGPO (about 20K versus 32K interactions), while exhibiting a smoother convergence curve and a narrower confidence band; GRPO does not reach this threshold within the displayed budget[1]. Together, these results form a clear logical chain: better belief grouping enables reliable credit assignment, which shortens task execution and ultimately yields highly efficient policy optimization.

To verify these efficiency gains stem from accurate latent state tracking, Appendix C.2 evaluates the structural integrity and observational consistency of generated beliefs. Diagnostic evaluations on 128 paired ALFWorld episodes reveal that REBEL substantially outperforms GRPO-Belief. Specifically, during early exploration (steps 1–5), the selected REBEL checkpoint achieves a near-perfect 99.6% belief parse rate, 87.7% state consistency, and 84.3% task-relevant recall, compared to merely 81.8%, 65.1%, and 64.2% for GRPO-Belief. By confirming that REBEL's explicit beliefs are well-formed and grounded in observations, these single-checkpoint diagnostics complement task-level performance (further metric definitions and scope discussions are provided in Appendix C.2, and the underlying diagnostics in Figure 7; Appendix D traces two complete episodes at step-level resolution).

5 CONCLUSION

We introduce REBEL, a process-supervised reinforcement learning framework that opens a new and promising path toward addressing credit assignment in long-horizon agentic tasks by optimizing explicit belief-state consistency, marking a shift from purely outcome-driven optimization to deeper process-level alignment. Rather than relying solely on terminal rewards, REBEL provides dense step-level supervision through the belief-consistency reward to effectively mitigate cascading failures, while employing a belief-grouping mechanism to identify semantically coherent phases in partially observable environments and enable more precise advantage normalization. Experimental results show that, using only a 1.5B-parameter backbone, REBEL achieves success rates of 93.2% on ALFWorld and 75.1% on WebShop, marking substantial improvements of +20.4 and +18.3 percentage points over the strong GRPO baseline, while enhancing sample efficiency by 2.1×. Our ablation studies further confirm that the integration of belief consistency and semantic grouping produces additive performance gains, underscoring that the self-alignment of an agent's internal world model is essential for achieving efficient and robust autonomous intelligence.

[1]Throughout, "environment steps" counts the cumulative number of environment interactions during rollout, scaled by the number of parallel rollouts per training epoch.

AI USE STATEMENT

As described in Appendix A.4, we used GPT-5.4 (OpenAI, 2026) to generate belief annotations for the expert trajectories used in supervised cold-start training. As described in Appendix B.1, an expert LLM was also used to derive the belief predicate vocabulary and the corresponding verification rules; this construction is fully automatic. Additionally, generative AI tools were used to help improve the clarity and language of parts of this manuscript. These tools were not involved in the experimental design, data analysis, or the formulation of research conclusions. The authors have reviewed the generated annotations and manuscript text, verified the reported results against the underlying experiments, and take full responsibility for the final content, claims, and artifacts.

REPRODUCIBILITY STATEMENT

The proposed method and optimization objective are described in detail in Section 3; training configurations, data construction pipelines, prompts, evaluation protocols, and specific training algorithms are provided in the appendix. To ensure reproducibility, an anonymized code repository is provided as supplementary material. It contains the training scripts and configurations required to reproduce the main tables and figures, together with the pipeline scripts that build the belief-annotated SFT dataset and derive the belief predicate vocabulary described in Appendices A.4 and B.1, so that both artifacts can be regenerated end to end from the public expert trajectories. Code is available at: `https://anonymous.4open.science/r/ReBel_RewardingBelief-DF52`.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024.

Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *Advances in Neural Information Processing Systems*, 37:12461–12495, 2024.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=H1lJJnR5Ym`.

Shiyi Cao, Dacheng Li, Fangzhou Zhao, Shuo Yuan, Sumanth R. Hegde, Connor Chen, Charlie Ruan, Tyler Griggs, Shu Liu, Eric Tang, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Skyrl-agent: Efficient rl training for multi-turn llm agent, 2025. URL `https://arxiv.org/abs/2511.16108`.

Bolin Chen, Ru-Ling Liao, Yan Ye, Jie Chen, Shanzhi Yin, Xinrui Ju, Shiqi Wang, and Yibo Fan. Sparse2dense: A keypoint-driven generative framework for human video compression and vertex prediction. In *2026 Data Compression Conference (DCC)*, pp. 3–12. IEEE, 2026.

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. Reinforcement learning for long-horizon interactive llm agents. *arXiv preprint arXiv:2502.01600*, 2025.

Youngbin Choi, Minjong Lee, Saemi Moon, Seunghyuk Cho, Chaehyeon Chung, MoonJeong Park, and Dongwoo Kim. In-place feedback: Reliable refinement for multi-turn expert-LLM collaboration. In *ICML 2026 Workshop on Foundations of Deep Generative Models: Understanding Memorization, Generalization, and Reasoning*, 2026. URL `https://openreview.net/forum?id=KphoyNDT5P`.

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, 2025. URL `https://arxiv.org/abs/2502.01456`.

Jinhao Duan, James Diffenderfer, Sandeep Madireddy, Tianlong Chen, Bhavya Kailkhura, and Kaidi Xu. Uprop: Investigating the uncertainty propagation of llms in multi-step agentic decision-making. *ArXiv*, abs/2506.17419, 2025.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 46375–46408. Curran Associates, Inc., 2025. doi: 10.52202/085713-1544. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/420c9f777c0b4f78d515e53cf74d58b2-Paper-Conference.pdf.

Jiayi Fu, Xuandong Zhao, Chengyuan Yao, Heng Wang, Qi Han, and Yanghua Xiao. Reward shaping to mitigate reward hacking in RLHF. In *ICML 2025 Workshop on Reliable and Responsible Foundation Models*, 2025. URL https://openreview.net/forum?id=62A4d5Mokc.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. A survey on llm-as-a-judge. *The Innovation*, 7(6), 2026.

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.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pp. 3929–3938. PMLR, 2020.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, 2023.

Byunghyun Kim, Minyoung Bae, and Jae-Gil Lee. Sample-efficient multi-round generative data augmentation for long-tail instance segmentation. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 73222–73245. Curran Associates, Inc., 2025. doi: 10.52202/085713-2454. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/69f59f96c3aae0ba9f463e7903715e1e-Paper-Conference.pdf.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=i1uGbfHHpH.

Aly Lidayan, Jakob Bjorner, Satvik Golechha, Kartik Goyal, and Alane Suhr. Abbel: Learning natural-language belief states for memory-efficient interaction, 2026. URL https://arxiv.org/abs/2512.20111.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, Yuheng Cheng, Suyuchen Wang, Xiaoqiang Wang, Yuyu Luo, Haibo Jin, Peiyan Zhang, Ollie Liu, Jiaqi Chen, Huan Zhang, Zhaoyang Yu, Haochen Shi, Boyan Li, Dekun Wu, Fengwei Teng, Xiaojun Jia, Jiawei Xu, Jinyu Xiang, Yizhang Lin, Tianming Liu, Tongliang Liu, Yu Su, Huan Sun, Glen Berseth, Jianyun Nie, Ian Foster, Logan Ward, Qingyun Wu, Yu Gu, Mingchen Zhuge, Xinbing Liang, Xiangru Tang, Haohan Wang, Jiaxuan You, Chi Wang, Jian Pei, Qiang Yang, Xiaoliang Qi, and Chenglin Wu. Advances and challenges in foundation

agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems, 2025. URL https://arxiv.org/abs/2504.01990.

Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Jianbin Jiao, and Junge Zhang. Agentic reinforcement learning with implicit step rewards. In *International Conference on Learning Representations*, volume 2026, pp. 129271–129291, 2026.

Gengchen Mai, Yingjie Hu, Song Gao, Ling Cai, Bruno Martins, Johannes Scholz, Jing Gao, and Krzysztof Janowicz. Symbolic and subsymbolic geoai: Geospatial knowledge graphs and spatially explicit machine learning. *Trans. GIS*, 26(8):3118–3124, 2022.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. Augmented language models: a survey. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=jh7wH2AzKK. Survey Certification.

Changdae Oh, Seongheon Park, To Eun Kim, Jiatong Li, Wendi Li, Samuel Yeh, Sean Du, Hamed Hassani, Paul Bogdan, Dawn Song, and Sharon Li. Uncertainty quantification in LLM agents: Foundations, emerging challenges, and opportunities. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16219–16250, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.738. URL https://aclanthology.org/2026.acl-long.738/.

OpenAI. Introducing GPT-5.4, 2026. URL https://openai.com/index/introducing-gpt-5-4/. Accessed: 2026-09-10.

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. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent q: Advanced reasoning and learning for autonomous ai agents. *arXiv preprint arXiv:2408.07199*, 2024.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, Jie Tang, and Yuxiao Dong. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 79791–79821, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/c66e1fcc9691aae706250638f36f681b-Paper-Conference.pdf.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru WANG, Xiusi Chen, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 105523–105553. Curran Associates, Inc., 2025. doi: 10.52202/085713-3524. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/97c5b2707228e7e3fb67e4ecc2e0e607-Paper-Conference.pdf.

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.

Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pp. 222–227, 1991.

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

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021.

Yi Su, Dian Yu, Linfeng Song, Juntao Li, Haitao Mi, Zhaopeng Tu, Min Zhang, and Dong Yu. Crossing the reward bridge: Expanding reinforcement learning with verifiable rewards across diverse domains. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3872–3892, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.178. URL https://aclanthology.org/2026.acl-long.178/.

Zhouhao Sun, Li Du, Xiao Ding, Yixuan Ma, Yang Zhao, Kaitao Qiu, Ting Liu, and Bing Qin. Causal-guided active learning for debiasing large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14455–14469, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.778. URL https://aclanthology.org/2024.acl-long.778/.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

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

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. *arXiv preprint arXiv:2509.08755*, 2025.

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. Watch every step! llm agent learning via iterative step-level process refinement. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1556–1572, 2024.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024a.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: training language model agents via hierarchical multi-turn rl. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024b.

Zhuoran Zhuang, Ye Chen, Jianghao Su, Chao Luo, Luhui Liu, and Xia Zeng. Enhancing agentic rl with progressive reward shaping and value-based sampling policy optimization, 2025. URL https://arxiv.org/abs/2512.07478.

APPENDIX

A EXPERIMENTAL DETAILS

A.1 COMPUTATIONAL DETAILS

For both ALFWorld and WebShop, we conduct experiments on 4×A800 GPUs using `Qwen2.5-1.5B-Instruct` as the base model. REBEL and all ablation variants are trained for 100 iterations, while the RL baselines are trained for 150 iterations; the two budgets are never mixed within a single comparison. During inference, we use vLLM as the rollout backend with tensor parallelism set to 1 and GPU memory utilization fixed at 0.65. Within each comparison the methods share an identical training budget and environment configuration, so that differences in success rate reflect the optimization method rather than the amount of training.

A.2 DATASET DETAILS

A.2.1 ALFWORLD

We evaluate our method on ALFWorld, a text-based embodied interaction benchmark designed to assess language models in multi-step decision-making scenarios. Unlike static question answering tasks, ALFWorld features a dynamic environment in which the state evolves in response to the agent's actions. At each step, the model must condition on the current observation, historical trajectory, and task goal to decide the next action. As a result, the benchmark not only tests natural language instruction following, but also challenges the model's state tracking, planning, and error correction capabilities in a changing environment.

ALFWorld contains 3,827 task instances spanning six representative household subtasks: Pick & Place, Examine in Light (Look), Clean & Place, Heat & Place, Cool & Place, and Pick Two & Place (Pick2). These subtasks differ substantially in action dependency structure, completion conditions, and state-transition patterns. Following the standard protocol, we evaluate on a fixed set of 128 held-out tasks, with episode lengths ranging from 8 to 30 steps. Owing to the diversity in decision horizon and operational complexity across sub-tasks, ALFWorld provides a comprehensive testbed for evaluating generalization in complex embodied environments. From the perspective of ReBel, the key challenge lies in maintaining a persistent understanding of the environment state across multiple interactions and progressively approaching the goal through a sequence of intermediate actions. This makes ALFWorld particularly suitable for assessing whether the model can establish a stable "belief update–reasoning–action" decision loop.

A.2.2 WEBSHOP

We further evaluate our method on WebShop, a web-based shopping environment designed to test multi-step retrieval, filtering, and decision-making under partial information. In this task, an agent must complete operations such as product search, page navigation, attribute filtering, item comparison, and final purchase within a simulated e-commerce website according to a user instruction.

WebShop contains more than 1.1 million products and over 12,000 user instructions. The maximum episode length is 15 steps, and the reward lies in the interval $[0, 1]$, with high rewards available only when the agent successfully completes the target. Since product information is often incomplete and the set of available actions changes across pages, WebShop imposes strong requirements on the model's retrieval ability, constraint satisfaction, and multi-step decision-making. From the perspective of ReBel, WebShop exhibits a typical "reasoning-driven decision-making" setting: at each step, the model must act based on the current page state and the interaction history, while each intermediate decision directly affects future reachable states and the final return. This makes the benchmark well suited for evaluating whether our method can maintain stable trajectory-level reasoning and policy consistency in complex web environments.

A.3 IMPLEMENTATION DETAILS

Model and cold-start setup. We perform all experiments with `Qwen2.5-1.5B-Instruct`. To ensure that the model acquires basic task-format understanding and structured generation capability

before entering the reinforcement learning stage, we adopt an SFT cold-start procedure prior to RL training. Specifically, we first fine-tune the base model on expert trajectories, using 426 trajectories for ALFWorld and 500 trajectories for WebShop. The SFT stage is trained for 3 epochs with a learning rate of 1×10^{-5}, and the micro-batch size per GPU is 8. The goal of this stage is to teach the model to reliably generate a structured three-part response format consisting of `<belief>`, `<think>`, and `<action>`, thereby providing a format-stable initialization for the subsequent RL stage and reducing invalid exploration caused by formatting errors at the beginning of training.

RL optimization setup. We implement ReBel on top of the veRL-agent framework and introduce the necessary modifications to accommodate partially observable long-horizon tasks. In the RL stage, the learning rate is set to 1×10^{-6}, and AdamW is used as the optimizer. For each task input, we sample $N = 16$ trajectories to form a rollout group, which is used for group-wise normalized advantage estimation. ReBel is trained for 100 iterations, whereas the other RL baselines, including PPO, RLOO, GRPO, and GiGPO, are trained for 150 iterations and averaged over 3 random seeds. For the ablation settings B0–B3, each configuration is trained for 100 iterations. The PPO mini-batch size is set to 256, and the micro-batch size per GPU is 4. The maximum episode length is 30 steps for ALFWorld and 15 steps for WebShop.

Advantage formulation. ReBel uses a linear combination of two-level advantages, namely the trajectory-level episode advantage $A^{\mathrm{ep}}(\tau_i)$ and the belief-anchor step-level advantage $A_S(i,t)$. The former is obtained by normalizing the task success signal within the rollout group, while the latter is computed by normalizing the step-level discounted return within the same belief equivalence class. We set the weight of the step-level component to 0.5, yielding $\hat{A}^{\mathrm{tot}}_{i,t} = A^{\mathrm{ep}}(\tau_i) + 0.5 \cdot A_S(i,t)$.

Pending predicates and grouping key. Predicates that cannot be verified at the step where they are asserted are held in a pending buffer and settled retroactively once a later observation makes them observable; when an episode ends with predicates still unsettled, they are excluded from the consistency totals rather than counted as errors. For the step-level advantage, the grouping key used in our ALFWorld experiments is the environment-derived phase anchor, augmented with structural flags taken from the belief (rather than the raw predicate tuple), so that steps at the same task phase remain comparable across trajectories. We set the minimum group size for normalization to 10 samples with a standard-deviation floor of 0.2, and singleton groups receive a zero step-level advantage rather than a normalized one.

Format regularization. To encourage the model to produce well-formed three-part responses, we apply a reward penalty of -0.1 to outputs that violate the required format. An output is considered valid only if it contains at least one belief-state tag, namely `<belief>...</belief>`, and one action tag, namely `<action>...</action>`.

Regularization and discounting. We set the KL divergence loss coefficient to 0.01 using the low-variance estimator, and the KL penalty coefficient to 0.001. The RL discount factor is set to $\gamma_{\mathrm{RL}} = 0.95$.

A.4 Construction of Belief-Annotated SFT Data

We construct belief-format supervised fine-tuning (SFT) data from a publicly available general-purpose expert trajectory dataset and release the resulting dataset. Specifically, we use GPT-5.4 as the annotation model to annotate expert trajectories step by step in temporal order. At each step, the annotator receives the task instruction, current observation, belief from the preceding step, the two most recent actions, and the expert reasoning and action recorded in the trajectory. It then updates the structured belief and explains the expert action without selecting a new action. The updated belief is carried forward to the next step to maintain continuity throughout the trajectory; when the current observation conflicts with the previous belief, the current observation takes precedence.

The annotations are serialized into step-wise training examples containing `<belief>`, `<think>`, and `<action>` blocks, with the action copied from the original expert trajectory. During offline data processing, subsequent recorded observations are used to construct the training targets, but these observations are not provided to the policy during inference. The dataset contains 426 ALFWorld expert trajectories and 500 WebShop expert trajectories, yielding $5{,}409$ and $2{,}506$ step-wise SFT examples, respectively. These data are used to train the belief-format SFT model before reinforcement learning, providing the initialization model for subsequent policy optimization.

A.5 Evaluation Protocol

We evaluate model performance on the validation set and use task success rate as the primary metric. For ALFWorld, we report both per-subtask success rates and the overall success rate on the fixed set of 128 held-out tasks. For WebShop, we report average score, which reflects partial matching quality, as well as success rate SR. We follow the standard evaluation protocols of each benchmark. During evaluation, sampling is disabled by setting the temperature to 0, or equivalently using greedy decoding, to reflect the model's generalization performance more stably and to distinguish evaluation from the stochastic rollout policy used during training.

B Prompts and Belief Representation

Figure 4 and Figure 5 present the prompt templates used by REBEL in the ALFWorld and WebShop environments, respectively. The templates are implemented with Python-style string formatting, where placeholders enclosed in curly braces are filled at runtime via `.format()`. Each prompt includes the task instruction, a short interaction history, the previous belief state, and the current observation. To keep the prompt concise, we retain only the two most recent actions in the history.

All prompts follow the same three-stage decision protocol: (i) belief update, which integrates the previous belief with the current observation; (ii) belief-grounded reasoning, which identifies the current task phase and the most relevant next step; and (iii) action selection, which outputs one executable action from the admissible action set. The model response is structured into three tagged blocks: `<belief>`, `<think>`, and `<action>`. The `<belief>` block serves as the structured latent state used by REBEL's grouping-based credit assignment, the `<think>` block encourages brief intermediate reasoning, and the `<action>` block contains the final executable action string.

ALFWorld Prompt Template

You are an expert agent operating in the ALFRED embodied environment.
Task: {task_description}
Step count: {step_count} Recent history ({history_length} steps): {action_history}
Current step: {current_step}
Current observation: {current_observation}

Use the previous belief state together with the current observation to update your belief, reason about the current task phase, and select the best next action. Historical observations are omitted because their relevant information is already summarized in the previous belief.

Previous belief:

<belief_prev> {previous_belief} **</belief_prev>**

Respond in the following format:

<belief>

state:

- "objects": {"[obj_id]": "[location | in_hand | unknown]"}
- "states": {"[obj_id]": "[heated | cooled | cleaned | sliced | none]"}
- "visited": ["[receptacle_id]"]
- "unvisited_candidates": ["[receptacle_id]"]

task:

- "phase": "[find | navigate | pickup | transform | place | done]"
- "target": "[current target object]"
- "subgoal": "[one-sentence immediate subgoal]"

"prediction": "If I execute [planned_action], I expect [next-observation consequence]."

</belief>

<think>

[Brief reasoning in 2–3 sentences: what is confirmed, what remains uncertain, which phase the task is in, and why the chosen action is currently optimal.]
</think>

<action>
[Exact action string selected from the admissible action set]
</action>

Figure 4: ALFWorld prompt template.

WebShop Prompt Template

You are an expert autonomous agent operating in the WebShop e-commerce environment.
Task: {task_description}
Step count: {step_count}
Recent history ({history_length} steps): {action_history}
Current step: {current_step}
Current observation: {current_observation}

Use the previous belief state together with the current observation to update the belief, infer the current task phase, identify the main information bottleneck, and select the best next action.

Previous belief:

<belief_prev> {previous_belief} **</belief_prev>**

Respond in the following format:

<belief>
"state":
- "target": {"[attr]": "[value]"}
- "confirmed": {"[attr]": "[verified value]"}
- "unconfirmed": ["[attr]"]
- "product_id": "[ASIN or null]"

"task":
- "phase": "[searching | browsing | viewing | selecting | buying | done]"
- "target": "[product_id or null]"
- "subgoal": "[updated immediate subgoal]"

"prediction":
"If I execute [planned_action], I expect [next-observation consequence]."
</belief>

<think>
[Brief reasoning in 2–3 sentences: what is confirmed, what remains uncertain, which phase the task is in, and why the chosen action is currently optimal.]
</think>

<action>
[Exact action string selected from the admissible action set]
</action>

Figure 5: WebShop prompt template.

B.1 A Unified Prompt for Belief Predicate Construction

We propose a unified prompt for belief predicate construction, formalizing the creation of belief representations into a transferable generative paradigm. This allows the automatic derivation of tailored belief predicates and structured representations based on specific environment and task information. Taking the environment description, action space, representative observations, and task instruction templates as inputs, this prompt automatically outputs a predicate vocabulary, correspond-

ing verification rules, and a structured belief schema. Within this framework, the core instructions of the prompt remain invariant across all environments; only the environment-specific inputs are substituted. This design effectively decouples the abstract construction of the belief representation system from the step-wise belief updates executed by the policy during interaction.

This prompt template inherits the core design philosophy of REBEL: belief states should retain as much decision-relevant information as possible while strictly distinguishing between "observed facts" and "unresolved hypotheses". We require that each generated predicate be equipped with an evidence-based verification rule; predicates for which observational evidence is temporarily unavailable must be maintained in an "unresolved" status, rather than being simply evaluated as "false". The final generated belief schema adopts the three-tier top-level structure of `state`, `task`, and `prediction`, whereas its internal specific fields and task phases are adaptively generated entirely based on the provided environmental inputs.

Construction pipeline. The procedure is fully automatic: it requires no per-task manual engineering and no access to hidden simulator state, and it proceeds in two stages. In the first stage, *predicate proposal*, an expert LLM is prompted with the environment description, the action space and action semantics, representative observations sampled from expert trajectories, and the task instruction templates. It returns an initial predicate vocabulary that fixes each predicate's name, arguments, and value domain, together with the observation or confirmed action outcome that verifies or refutes it and the conditions under which it must remain unresolved rather than be equated with false. In the second stage, *coverage testing*, the initial vocabulary is exercised against environment interaction and its coverage is measured; predicates that are never triggered by an observable cue, or that cannot be verified through the interface, are revised or dropped. The final vocabulary is then assembled into the three-tier `state`/`task`/`prediction` schema, with task phases derived from the action semantics rather than fixed in advance. The prompt is reused unchanged across environments, so only the environment-specific inputs differ between ALFWorld and WebShop, and the resulting vocabulary is never edited by hand.

Derived predicates. Applied to the two benchmarks, the pipeline yields environment-specific vocabularies that share no fields, which is the intended behaviour. For ALFWorld, the state predicates track per-object location (`objects`: a receptacle, `in_hand`, or `unknown`) and transformation status (`states`: heated, cooled, cleaned, sliced, or none), supplemented by receptacle-coverage predicates (`visited` and `unvisited_candidates`); the derived task phases are find, navigate, pickup, transform, place, and done. For WebShop, the predicates instead track attribute satisfaction (`target` and `confirmed` over attribute–value pairs, `unconfirmed` attributes, and `product_id`), with phases searching, browsing, viewing, selecting, buying, and done. Because the two vocabularies are derived independently, the abstract construction of the representation system is decoupled from the step-wise belief updates that the policy performs during interaction.

Unified Belief Predicate Construction Prompt

You are designing a structured belief representation for an agent operating in a partially observable environment. Based on the provided environment information below, construct a compact and highly task-relevant set of predicates, along with a corresponding belief schema. Your core objective is to "define the representation system," rather than to solve a specific individual task or execute actions.

[Inputs]

Environment description: {environment_description}
Action space and action semantics: {action_space}
Representative observations: {observation_examples}
Task instruction templates: {task_templates}

[Construction Requirements]

1. Identify key elements: Extract the entities, attributes, relations, and task progress information essential for distinguishing states that require different actions. Retain only information highly relevant to the given tasks.

2. Define predicate specifications: Specify the name, arguments, allowed value domain, and semantics for each predicate. For facts lacking evidential support, an explicit "unknown" value must be defined. A "missing assertion" must absolutely not be automatically equated to "false".

3. Establish verification rules: Clearly specify which observations or confirmed action outcomes can verify or refute each predicate. Do not assume the agent has access to hidden simulator states, future observations, or information not provided by the interface. If verification is delayed, specify what subsequent evidence is required to resolve the unresolved status. In the consistency scoring mechanism, permanently unverifiable hypotheses should be excluded.

4. Specify update mechanisms: Define how new evidence updates the predicates (when to add, retain, replace, or retract an assertion). In case of conflicts, current evidence takes precedence over past beliefs. Strictly distinguish between "planned actions to be executed" and "confirmed action outcomes."

5. Ensure information isolation: Maintain clear boundaries among task requirements, observed facts, and inferred hypotheses. Strictly bind assertions to their applicable entities and contexts to prevent evidence regarding one entity from being erroneously generalized to others.

6. Derive task phases: Based on the provided task structure and action semantics, derive a compact set of task phases. Describe the supporting evidence that triggers transitions between phases, instead of applying a fixed list of phases shared across all environments.

7. Format output: Construct a JSON belief schema containing three top-level fields:

- state: task-relevant facts (must explicitly distinguish unknown or inferred values);
- task: current phase, overall goal, and immediate subgoal;
- prediction: expected observable outcomes of planned actions (must be strictly separated from confirmed states).

[Final Output Format]

A. Predicate vocabulary
For each predicate, include: name, arguments, value domain, meaning, and its corresponding JSON field mapping.
B. Verification and update rules
For each predicate, include: supporting evidence, refuting evidence, conditions for maintaining an unresolved status, and update/retraction rules. Include evidence-based task phase transition rules.
C. Belief schema
A compact JSON template containing state, task, and prediction. The environment-specific fields within this template must be derived from the inputs above; directly applying fixed domain templates is prohibited.
D. Observability limitations
Identify information that cannot be verified through the current interface, and explain how to exclude it from consistency checks or retain it solely as an unresolved hypothesis.

C Additional Empirical Analysis

C.1 Belief Drift under Partial Observability

In partially observable environments, agents must reason over latent beliefs inferred from incomplete observations. As illustrated in Figure 6, a think-only agent may become overconfident in an incorrect belief and continue selecting actions that are locally consistent with this mistaken assumption, even after receiving contradictory observations. Since sparse terminal rewards provide limited supervision over intermediate reasoning errors, standard outcome-level reinforcement learning often fails to correct such latent belief drift. In contrast, belief-aware policies explicitly revise their internal beliefs according to new observations, progressively reducing uncertainty and enabling recovery from incorrect hypotheses before the trajectory diverges from the task objective.

C.2 Belief Quality and Behavior

Evaluation Setup To investigate the underlying mechanisms of internal state tracking, we conduct a diagnostic evaluation on 128 paired ALFWorld episodes sampled from the *valid-seen* split. All checkpoints are evaluated under a unified environment seed (1042) using greedy decoding, with a maximum of 30 steps per episode. This experiment compares belief quality across the SFT-only model, think-format GRPO, belief-format GRPO (GRPO-Belief), and REBEL. The setup is designed

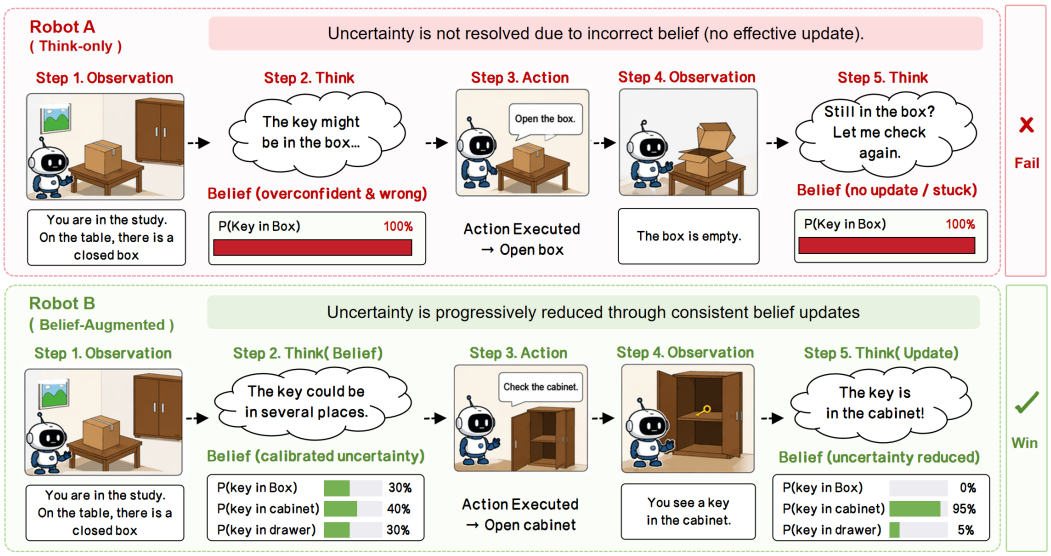

Figure 6: **Belief drift as a failure mode in partially observable environments.** The top row shows a Think-only agent that remains overconfident in an incorrect belief and repeatedly executes invalid actions. The bottom row shows a Belief-Augmented agent that updates its belief from observations, progressively reduces uncertainty, and succeeds in the task. REBEL aims to induce this belief-aware reasoning behavior during RL training.

to distinguish the accuracy of internal state tracking from final task success, thereby examining how different optimization objectives influence the agent's latent state representation.

Metrics We quantify belief quality and its behavioral implications across several dimensions: (i) **Belief Parse Rate**, which measures the model's ability to generate belief blocks that follow the specified syntax; (ii) **Task-Relevant Recall**, which measures the coverage of verified facts concerning task objects and receptacle states; (iii) **Consistency Rate**, which measures the agreement between verifiable belief assertions and the environment state, as tracked from observations, before action execution; and (iv) **Behavioral Robustness Metrics**, including the *stale-unvisited rate*, which measures the frequency of incorrectly marking visited receptacles as unvisited, and the *repeated-action rate*, which measures immediate repetition of the previous action. We report these metrics within episode-step buckets to distinguish early state tracking from subsequent execution behavior. Belief-specific metrics are undefined for think-format GRPO because it does not generate structured belief blocks.
The consistency rate is reported as the mean of per-step agreement fractions, and a step contributes to the numerator and denominator only if it contains at least one predicate that the current observation can verify; steps with no verifiable predicate, and predicates held in the pending buffer, are excluded from both rather than counted as inconsistent. A predicate that an observation contradicts counts once against agreement, while correct assertions over the unvisited-candidate diagnostic are not added to the denominator, so this quantity should be read together with coverage rather than as an exhaustive accuracy over all possible predicates. Because the compared methods converge at different speeds, the later episode-step buckets contain only the episodes that are still running at those steps, and the bucket-wise curves therefore track different subsets of surviving episodes rather than a fixed population observed over time.

Analysis of Results As shown in Figure 7, the diagnostic results reveal a clear association between belief quality, state tracking, and interactive behavior. The SFT-only model illustrates a pronounced gap between structural fidelity and task utility: although it achieves a near-perfect parse rate of 99.5%, its task success rate is only 3.1%. This result shows that syntactically valid belief blocks do not necessarily constitute task-relevant state representations. Structured output is therefore a prerequisite for explicit belief modeling, but it is insufficient to ensure that the generated beliefs are grounded in the environment or support effective downstream decisions.

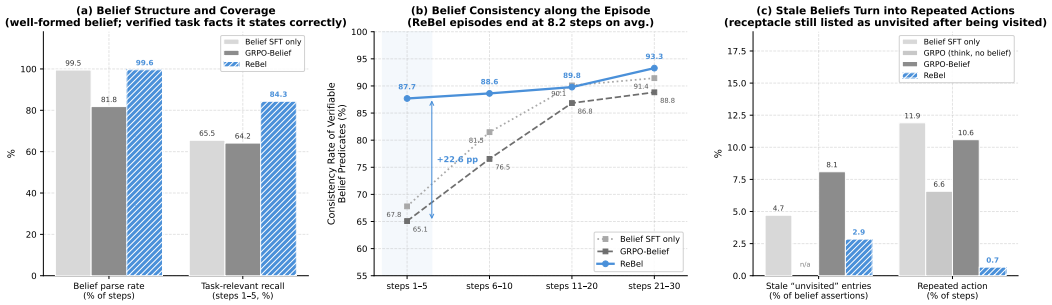

Figure 7: **Belief-quality diagnostics on** 128 **paired ALFWorld episodes.** Results use greedy decoding and single checkpoints; REBEL results refer to the selected seed-123 checkpoint. (a) Structural fidelity, measured by belief parse rate, and semantic coverage, measured by task-relevant recall, during the initial exploration phase (steps 1–5). (b) Consistency of verifiable belief assertions across episode-step buckets. (c) Behavioral diagnostics measuring stale search beliefs and immediate action repetitions across the compared optimization objectives.

REBEL achieves stronger task-relevant coverage and observational consistency while maintaining high structural fidelity. It obtains a 99.6% parse rate and an early task-relevant recall of 84.3%, compared with 81.8% and 64.2%, respectively, for GRPO-Belief. During the first five steps, REBEL also achieves a consistency rate of 87.7%, exceeding GRPO-Belief by 22.6 percentage points. These results indicate that, in this diagnostic setting, REBEL's beliefs are not only well formed but also more closely aligned with the task state supported by available evidence. This finding is consistent with REBEL's objective of integrating belief evolution into process-level optimization, thereby providing more reliable state information for subsequent action selection.

The difference in belief quality is accompanied by more stable interaction behavior. Compared with GRPO-Belief, REBEL reduces the stale-unvisited rate from 8.1% to 2.9% and the repeated-action rate from 10.6% to 0.7%. These results indicate that REBEL is less likely to retain search hypotheses contradicted by subsequent observations or to enter redundant action loops. Taken together, the higher information coverage and observational consistency, along with the lower rates of stale beliefs and repeated actions, support the importance of maintaining high-quality belief states for stable interaction in partially observable, long-horizon tasks.

Scope of the diagnostic comparison The diagnostic contrasts two specific recorded configurations rather than an isolated intervention. The two models share the belief-format initialization and prompt, but they are not trained under identical optimization settings: the recorded GRPO-Belief run uses symmetric clipping of $0.2/0.2$ without entropy protection, whereas REBEL uses $0.2/0.28$ with entropy protection enabled, so the comparison does not by itself separate the contribution of the belief reward from that of the grouping mechanism. Variation across seeds is also present: REBEL seed 2025 has a repeated-action rate of 12.9%, above the 10.6% of GRPO-Belief, and the three-seed aggregate belief-correction rate of $35.2 \pm 19.4\%$ does not exceed the 46.7% of GRPO-Belief. We therefore read the figure as a characterization of the selected run's structure, coverage, and early consistency, and not as evidence of a universal improvement in correction frequency.

D TRAJECTORY CASE STUDIES

The aggregate diagnostics in Appendix C.2 quantify belief quality statistically, but they cannot show how the belief layer behaves at the resolution of individual steps. This section complements those measurements with two complete episodes that illustrate, respectively, the intended operation of the belief–reasoning–action loop on a multi-phase task, and a ground-truth-verified belief correction together with the residual error mode that precedes it. All observations and model outputs are reported verbatim.

D.1 CASE STUDY I: BELIEF-GROUNDED EXECUTION OF A MULTI-PHASE TASK

We first examine a successful episode of the *pick-clean-place* task type, whose summary and full interaction log are given below. The task requires the agent to locate a mug, clean it at a sinkbasin,

and place it in the coffeemachine, so correct execution decomposes into a strict phase sequence: `find`, `pickup`, `transform`, and `place`.

Three aspects of the belief layer are worth noting in the log. First, the `phase` field tracks this decomposition accurately: it changes from `find` to `pickup` exactly when the mug is discovered on countertop 2, and from `transform` to `place` exactly when the cleaning action is confirmed by the environment. Second, the `objects` map and the `visited` list maintain a coherent account of the task-relevant state: the mug is asserted at its observed location, switches to `in_hand` once the pickup is confirmed, and the visited list grows monotonically as receptacles are inspected, which prevents redundant re-examination of the same locations. Third, the `prediction` field commits the policy to an expected observation before each action, providing the substrate that the belief-consistency reward of REBEL verifies against the environment.

The episode also exhibits the recovery pattern that the behavioral diagnostics of Appendix C.2 measure in aggregate. The first placement attempt at Step 7 is not admitted by the environment, and the resulting observation offers no confirmation that the mug was placed. The belief does not destabilize—the mug remains correctly tracked as `in_hand` and the phase remains `place`—and the agent does not repeat the failed action; it rephrases the placement with an alternative action template at Step 8 and completes the task in eight steps. Local action failures are thus treated as isolated events rather than as evidence about the environment state, consistent with the low repeated-action rate reported for REBEL.

ID	`b3_rebel_00082`
Task Type	pick_clean_then_place_in_recep
Status	**Success**
Steps	8
Format Valid Rate:	1.00
Episode ID:	82

TASK

Task Description

put a clean mug in coffeemachine.

INTERACTION

Observation (Step 1)

```
-= Welcome to TextWorld, ALFRED! =-
You are in the middle of a room.  Looking quickly around you, you
see a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet
2, a cabinet 1, a coffeemachine 1, a countertop 3, a countertop
2, a countertop 1, a drawer 3, a drawer 2, a drawer 1, a fridge 1,
a garbagecan 1, a microwave 1, a shelf 3, a shelf 2, a shelf 1, a
sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a
stoveburner 1, and a toaster 1.
Your task is to:  put a clean mug in coffeemachine.
```

Agent (Step 1) (admissible)

Belief State:
Phase: find
Target: mug
Subgoal: Find the mug so it can later be cleaned and placed in the coffeemachine.
Prediction: If I execute go to countertop 1, I expect On the countertop 1, you see a apple 2, a apple 1, a bread 1, a bread 2, a egg 1, a fork 1, a mug 1, a pan 1, and a tomato 1.
Action:

```
go to countertop 1
```

Observation (Step 2)

```
You arrive at countertop 1.  On the countertop 1, you see a apple 3, a
cellphone 1, a creditcard 1, a peppershaker 3, a peppershaker 1, and a
tomato 1.
```

Agent (Step 2) (admissible)

Belief State:
Phase: find
Target: mug
Subgoal: Find the mug so it can later be cleaned and placed in the coffeemachine.
Visited: countertop 1
Prediction: If I execute go to countertop 2, I expect On the countertop 2, you see a apple 2, a bread 1, a bread 2, a egg 2, a fork 2, a mug 2, a pan 2, and a tomato 2.
Action:

```
go to countertop 2
```

Observation (Step 3)

```
You arrive at countertop 2.  On the countertop 2, you see a apple 2, a
bowl 1, a butterknife 1, a fork 1, a mug 1, a spoon 1, and a statue 1.
```

Agent (Step 3) (admissible)

Belief State:
Phase: pickup
Target: mug 1
Subgoal: Pick up mug 1 from countertop 2 so it can be cleaned next.
Objects: mug 1: countertop 2
Visited: countertop 1, countertop 2
Prediction: If I execute take mug 1 from countertop 2, I expect You pick up the mug 1 from the countertop 2.
Action:

```
take mug 1 from countertop 2
```

Observation (Step 4)

```
You pick up the mug 1 from the countertop 2.
```

Agent (Step 4) (admissible)

Belief State:
Phase: transform
Target: mug 1
Subgoal: Bring the mug 1 to a sinkbasin so it can be cleaned.
Objects: mug 1: in_hand
Visited: countertop 1, countertop 2
Prediction: If I execute go to sinkbasin 1, I expect On the sinkbasin 1, you see a apple 2, a apple 1, a bread 1, a fork 2, a fork 1, a knife 1, a mug 2, a potato 1, and a tomato 2.
Action:

```
go to sinkbasin 1
```

Observation (Step 5)

```
You arrive at sinkbasin 1.  On the sinkbasin 1, you see a glassbottle
2, and a spatula 1.
```

Agent (Step 5) (admissible)

Belief State:
Phase: transform
Target: mug 1
Subgoal: Clean the mug 1 using sinkbasin 1 so it can later be placed in the coffeemachine.
Objects: mug 1: in_hand
Visited: countertop 1, countertop 2, sinkbasin 1
Prediction: If I execute clean mug 1 with sinkbasin 1, I expect On the sinkbasin 1, you see a apple 2, a apple 1, a bread 1, a fork 2, a fork 1, a knife 1, a mug 2, a potato 1, and a tomato 2.
Action:

```
clean mug 1 with sinkbasin 1
```

Observation (Step 6)

```
You clean the mug 1 using the sinkbasin 1.
```

Agent (Step 6) (admissible)

Belief State:
Phase: place
Target: mug 1
Subgoal: Carry the cleaned mug 1 to coffeemachine 1 and place it inside.
Objects: mug 1: in_hand
Visited: countertop 1, countertop 2, sinkbasin 1
Prediction: If I execute go to coffeemachine 1, I expect On the coffeemachine 1, you see a apple 1, a bread 2, a bread 1, a butterknife 2, a butterknife 1, a egg 1, a fork 3, a fork 2, a fork 1, a mug 2, a mug 1,...[truncated]
Action:

```
go to coffeemachine 1
```

Observation (Step 7)

```
You arrive at coffeemachine 1.  On the coffeemachine 1, you see
nothing.
```

Agent (Step 7) (inadmissible)

Belief State:
Phase: place
Target: mug 1
Subgoal: Place the cleaned mug 1 into coffeemachine 1.
Objects: mug 1: in_hand
Visited: countertop 1, countertop 2, sinkbasin 1, coffeemachine 1
Prediction: If I execute put mug 1 in/on coffeemachine 1, I expect On the coffeemachine 1, you see a apple 1, a bread 2, a bread 1, a butterknife 2, a butterknife 1, a egg 1, a fork 3, a fork 2, a fork 1, a mug 2...[truncated]
Action:

```
put mug 1 in/on coffeemachine 1
```

Observation (Step 8)

```
On the coffeemachine 1, you see nothing.
```

Agent (Step 8) (admissible)

Belief State:
Phase: place
Target: mug 1
Subgoal: Place the cleaned mug 1 into coffeemachine 1.
Objects: mug 1: in_hand
Visited: countertop 1, countertop 2, sinkbasin 1, coffeemachine 1
Prediction: If I execute move mug 1 to coffeemachine 1, I expect On the coffeemachine 1, you see a apple 1, a bread 2, a bread 1, a butterknife 2, a butterknife 1, a egg 1, a fork 3, a fork 2, a fork 1, a mug 2, ...[truncated]
Action:

```
move mug 1 to coffeemachine 1
```

D.2 Case Study II: Self-Correction of a Post-Placement Belief Regression

The second case study examines belief revision directly, a behavior that the aggregate metrics can count but not display. The episode below is drawn from the full trajectory dumps of the diagnostic evaluation described in Appendix C.2; the agent must find two tissue boxes and place both in the toilet. Because a receptacle name routinely appears in observations simply because the agent stands next to it, revisions in this analysis are identified against an observation-based ground-truth tracker of object locations rather than from the observation text. A *correction* is a transition of an object's asserted location from a value that disagrees with the tracker to the ground-truth value; a *regression* is the converse transition. The model's own commentary in its reasoning blocks is likewise not used as evidence.

The episode exhibits a recurring residual error mode. After the agent moves `tissuebox 1` to the toilet at Step 4 and receives the confirmation at Step 5, the assertion for `tissuebox 1` in the `objects` map becomes unstable: it regresses from the correct `toilet 1` back to the source receptacle `countertop 1` at Step 6, returns to the correct value at Step 7, and regresses again at Step 8 while the agent is searching for the second tissue box. We refer to this pattern as *post-placement regression*: after the agent itself has moved an object to its destination, the belief reverts the object to the receptacle it was taken from. Notably, the regression is silent—the accompanying reasoning blocks continue to describe the plan coherently and never revisit the location of `tissuebox 1`—which is precisely why corrections must be verified against the tracker rather than against the model's self-report.

The correction occurs at Step 9. Arriving at the toilet yields an observation that explicitly lists `tissuebox 1` among its contents, and the belief settles on the ground-truth location `toilet 1`; the same action completes the task. The revision is thus triggered by direct re-observation of the affected receptacle rather than by inference, and it is verified externally rather than declared by the model.

ID	`rebel_seed123_ep45`
Task Type	`find two tissuebox and put them in toilet`
Status	**Success**
Steps	9
Belief CR	0.825
Revision	`tissuebox 1: countertop 1` → `toilet 1`

Task

Task Description

find two tissuebox and put them in toilet..

INTERACTION

Observation (Step 1)

```
-= Welcome to TextWorld, ALFRED! =-
You are in the middle of a room.  Looking quickly around you, you
see a bathtubbasin 1, a countertop 1, a drawer 8, a drawer 7, a
drawer 6, a drawer 5, a drawer 4, a drawer 3, a drawer 2, a drawer
1, a garbagecan 1, a handtowelholder 1, a sinkbasin 1, a toilet 1, a
toiletpaperhanger 1, a towelholder 2, and a towelholder 1.
Your task is to:  find two tissuebox and put them in toilet.
```

Agent (Step 1) (admissible)

Belief State:
Phase: find
Target: tissuebox
Subgoal: Find a tissuebox by inspecting likely receptacles in the bathroom.
Objects: tissuebox: unknown
Visited: —
Prediction: If I execute go to countertop 1, I expect On the countertop 1, you see a candle 1, a creditcard 1...
Action:

```
go to countertop 1
```

Observation (Step 2)

```
You arrive at countertop 1.  On the countertop 1, you see a soapbottle
1, a spraybottle 2, a spraybottle 1, a tissuebox 3, a tissuebox 2, and
a tissuebox 1.
```

Agent (Step 2) (admissible)

Belief State:
Phase: pickup
Target: tissuebox 1
Subgoal: Pick up tissuebox 1 from countertop 1 so it can be placed in the toilet.
Objects: tissuebox 1: countertop 1, tissuebox 2: countertop 1, tissuebox 3: countertop 1
Visited: countertop 1
Prediction: If I execute take tissuebox 1 from countertop 1, I expect You pick up the tissuebox 1 from the countertop 1.
Action:

```
take tissuebox 1 from countertop 1
```

Observation (Step 3)

```
You pick up the tissuebox 1 from the countertop 1.
```

Agent (Step 3) (admissible)

Belief State:
Phase: place
Target: tissuebox 1
Subgoal: Carry tissuebox 1 to toilet 1 and place it there.
Objects: tissuebox 1: in_hand, tissuebox 2: unknown, tissuebox 3: unknown
Visited: countertop 1
Prediction: If I execute go to toilet 1, I expect On the toilet 1, you see a candle 1, a candle 2, a sink 1, a toiletpaper 1, and a tissuebox 3.
Action:

```
go to toilet 1
```

Observation (Step 4)

```
You arrive at toilet 1.  On the toilet 1, you see a candle 1, and a
toiletpaper 1.
```

Agent (Step 4) (admissible)

Belief State:
Phase: place
Target: tissuebox 1
Subgoal: Put tissuebox 1 into toilet 1.
Objects: tissuebox: unknown, tissuebox 1: countertop 1, tissuebox 2: countertop 1, tissuebox 3: countertop 1
Visited: countertop 1, toilet 1
Prediction: If I execute move tissuebox 1 to toilet 1, I expect On the toilet 1, you see a candle 1, and a toiletpaper 1.
Action:

```
move tissuebox 1 to toilet 1
```

Observation (Step 5)

```
You move the tissuebox 1 to the toilet 1.
```

Agent (Step 5) (admissible)

Belief State:
Phase: navigate
Target: tissuebox 2
Subgoal: Find another tissuebox to place at toilet 1.
Objects: tissuebox: unknown, tissuebox 1: toilet 1, tissuebox 2: unknown, tissuebox 3: unknown
Visited: countertop 1, toilet 1
Prediction: If I execute go to garbagecan 1, I expect On the garbagecan 1, you see nothing.
Action:

```
go to garbagecan 1
```

Observation (Step 6)

```
You arrive at garbagecan 1.  On the garbagecan 1, you see nothing.
```

Agent (Step 6) (admissible)

Belief State:
Phase: navigate
Target: tissuebox 2
Subgoal: Go to countertop 1 to continue searching for another tissuebox.
Objects: tissuebox: unknown, tissuebox 1: countertop 1, tissuebox 2: countertop 1, tissuebox 3: countertop 1
Visited: countertop 1, toilet 1, garbagecan 1
Prediction: If I execute go to countertop 1, I expect On the countertop 1, you see a candle 2, a candle 1...
Action:

```
go to countertop 1
```

Observation (Step 7)

```
You arrive at countertop 1.  On the countertop 1, you see a soapbottle
1, a spraybottle 2, a spraybottle 1, a tissuebox 3, and a tissuebox 2.
```

Agent (Step 7) (admissible)

Belief State:
Phase: pickup
Target: tissuebox 2
Subgoal: Pick up tissuebox 2 from countertop 1 so it can be placed at toilet 1.
Objects: tissuebox: unknown, tissuebox 1: toilet 1, tissuebox 2: countertop 1, tissuebox 3: countertop 1
Visited: countertop 1, toilet 1
Prediction: If I execute take tissuebox 2 from countertop 1, I expect You pick up the tissuebox 2 from the countertop 1.
Action:

```
take tissuebox 2 from countertop 1
```

Observation (Step 8)

```
You pick up the tissuebox 2 from the countertop 1.
```

Agent (Step 8) (admissible)

Belief State:
Phase: place
Target: tissuebox 2
Subgoal: Carry tissuebox 2 to toilet 1 and place it there.
Objects: tissuebox: unknown, tissuebox 1: countertop 1, tissuebox 2: countertop 1, tissuebox 3: countertop 1
Visited: countertop 1, toilet 1, garbagecan 1
Prediction: If I execute go to toilet 1, I expect On the toilet 1, you see a candle 1, a candle 2...
Action:

```
go to toilet 1
```

Observation (Step 9)

```
You arrive at toilet 1.  On the toilet 1, you see a candle 1, a
tissuebox 1, and a toiletpaper 1.
```

Belief Revision

`tissuebox 1`: `countertop 1` → `toilet 1` (ground truth: `toilet 1`)

Agent (Step 9) (belief revised)

Belief State:
Phase: place
Target: tissuebox 2
Subgoal: Place tissuebox 2 at toilet 1.
Objects: tissuebox: unknown, tissuebox 1: toilet 1, tissuebox 2: countertop 1, tissuebox 3: countertop 1
Visited: countertop 1, toilet 1, garbagecan 1
Prediction: If I execute move tissuebox 2 to toilet 1, I expect On the toilet 1, you see a candle 1, a tissuebox 1, and a toiletpaper 1.
Action:

```
move tissuebox 2 to toilet 1
```

We report this case with two caveats that bound the interpretation. First, corrections of this kind do not by themselves cause task success: many episodes containing regressions also end successfully, because the regressed assertion typically concerns an object that no longer requires further action. Second, the pattern is residual rather than resolved. Across the 384 episodes in the diagnostic trajectory dumps (three REBEL checkpoints, 128 episodes each), we count 768 ground-truth-verified corrections and 191 regressions, and the majority of both occur in episodes that end successfully. The case therefore illustrates a bounded but genuine capability—the belief layer recovers from drift once decisive evidence arrives, which is the corrective behavior that the belief-consistency reward is designed to reinforce—while the residual instability of post-placement assertions remains an open limitation that we discuss in Appendix **??**.

E TRAINING ALGORITHM AND THEORY

E.1 PSEUDO CODE

Algorithm 1 ReBel Training

Require: Policy π_θ, reference policy π_{ref}, task distribution $p(x)$, rollout size N, horizon T, weights ω, β, clip range ϵ

Initialize $\theta \leftarrow \theta_{\text{SFT}}$
for each training iteration **do**
  Sample task $x \sim p(x)$ and collect N rollouts
  for each rollout $i \in \{1, \dots, N\}$ and step $t \in \{1, \dots, T\}$ **do**
    Generate $b_t^{(i)}, z_t^{(i)}, a_t^{(i)}$ from $\pi_\theta(\cdot \mid h_t^{(i)}, x)$
    Execute $a_t^{(i)}$ and observe $o_{t+1}^{(i)}$
    Compute belief-consistency reward $r_{t,\text{cons}}^{(i)}$ using observability mask and pending buffer
    Update history $h_{t+1}^{(i)}$
  end for
  Compute episode-level advantage $A_E^{(i)}$
  Group step samples by canonicalized belief anchors and compute step-level advantage $A_S^{(i,t)}$
  Combine advantages: $A_{\text{tot}}^{(i,t)} = A_E^{(i)} + \omega A_S^{(i,t)}$
  Update θ by maximizing clipped PPO objective with KL regularization:

$$J(\theta) = \mathbb{E}\Big[\min\Big(r_t(\theta)A_{\text{tot}}^{(i,t)}, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_{\text{tot}}^{(i,t)}\Big)\Big] - \beta\,\text{KL}(\pi_\theta \,\|\, \pi_{\text{ref}})$$

end for

E.2 PROBLEM DIAGNOSIS AND REBEL DESIGN

We consider a partially observable sequential decision process with latent state $s_t \in \mathcal{S}$, observation $o_t \in \mathcal{O}$, action $a_t \in \mathcal{A}$, and history $h_t = (o_1, a_1, \dots, o_t)$. The task input is denoted by x. Since s_t is not directly observable, a policy must maintain an internal belief over the task-relevant aspects of the environment. In what follows, we use $\bar{b}_t$ to denote the implicit belief induced by a conventional end-to-end policy, and b_t to denote the explicit belief used by REBEL.

A conventional end-to-end policy directly maps the history to the next output,

$$\pi_\theta(y_t \mid h_t, x),$$

where y_t may contain reasoning tokens, tool calls, or the final action. Although this policy implicitly encodes a belief $\bar{b}_t$, such belief is neither explicitly exposed nor directly supervised. As a result, if $\bar{b}_t$ deviates from the true task-relevant state abstraction, the agent receives no timely correction signal. Under sparse terminal rewards, this deviation may persist across many steps and propagate through subsequent reasoning and action choices.

We formalize this failure mode as belief drift.

Definition 1 (Belief drift)**.** *Let $D : \mathcal{B} \times \mathcal{B} \to \mathbb{R}_{\geq 0}$ be a discrepancy function. The belief drift at step t is defined as*

$$d_t = D(b_t, \phi(s_t)),$$

where b_t is the belief used by the agent for decision making and $\phi : \mathcal{S} \to \mathcal{B}$ is a task-relevant projection from latent states to belief space.

From an optimization perspective, the main difficulty is credit assignment. With only an episode-level return R_i for trajectory i, all intermediate decisions share the same delayed learning signal:

$$g_i^{\text{trad}} = \sum_{t=1}^{T_i} \nabla_\theta \log \pi_\theta(y_{i,t} \mid h_{i,t}, x_i)\, A_i^E.$$

Therefore, the policy gradient cannot distinguish whether a failure is caused by an incorrect belief update, an unhelpful reasoning step, or a poor final action. In long-horizon partially observable tasks, this ambiguity makes training unstable. Under the usual weak-correlation assumption across steps, the variance of such a trajectory-level estimator grows at least approximately linearly with horizon length, which explains why sparse terminal supervision is often insufficient.

REBEL addresses this issue by making the latent decision structure explicit. Instead of treating each step as a monolithic generation process, REBEL decomposes it into Belief, Think, and Action layers:

$$\pi_\theta(b_t, z_t, a_t \mid h_t, x) = \pi_\theta(b_t \mid h_t, x)\, \pi_\theta(z_t \mid h_t, x, b_t)\, \pi_\theta(a_t \mid h_t, x, b_t, z_t).$$

Here b_t is the explicit belief, z_t is the intermediate reasoning trace, and a_t is the action. This factorization turns the hidden belief into an observable and trainable object, and it separates three sources of error: belief construction, reasoning conditioned on belief, and action selection conditioned on both.

To supervise the belief layer, REBEL introduces belief consistency supervision. Suppose the belief is represented by K task predicates $p_1, \ldots, p_K$. Let $\hat{p}_{t,k}(b_t)$ be the value of predicate p_k implied by belief b_t, and let $v_{t+1,k}$ be the corresponding value verified from the next observation whenever it is observable. Because not every predicate is immediately observable, we define an observability mask $m_{t,k} \in \{0, 1\}$. The consistency indicator is

$$C_{t,k} = \mathbf{1}\{\hat{p}_{t,k}(b_t) = v_{t+1,k}\}.$$

The step-wise consistency reward is

$$r_t^{\text{cons}} = \frac{\sum_{k=1}^K m_{t,k} C_{t,k}}{\max\{1, \sum_{k=1}^K m_{t,k}\}}.$$

This reward provides immediate supervision for the observable part of the belief. For predicates that are not yet observable, REBEL maintains a pending buffer U_t and evaluates them once a later observation makes them verifiable. In this way, consistency supervision supports both immediate correction and delayed verification.

Finally, REBEL introduces a belief-anchor step-wise advantage to avoid the singleton problem in partially observable environments. State-based grouping is ineffective because the same physical state s_t may rarely reappear across sampled trajectories. REBEL instead groups samples by a symbolic belief anchor $\tilde{b}_t$, defined as the observable and pending predicate signature extracted from b_t. The belief-anchor group is

$$\mathcal{G}_B(\tilde{b}) = \{(i, t) : \tilde{b}_{i,t} = \tilde{b}\}.$$

Let $G_{i,t}^S$ be a local step-level return, for example the discounted sum of consistency rewards over a short horizon:

$$G_{i,t}^S = \sum_{\ell=t}^{t+H} \gamma^{\ell-t} r_{i,\ell}^{\text{cons}}.$$

The belief-anchor advantage is then

$$A_{i,t}^S = \frac{G_{i,t}^S - \mu_B(\tilde{b}_{i,t})}{\sigma_B(\tilde{b}_{i,t}) + \varepsilon},$$

where

$$\mu_B(\tilde{b}) = \frac{1}{|\mathcal{G}_B(\tilde{b})|} \sum_{(j,u) \in \mathcal{G}_B(\tilde{b})} G_{j,u}^S,$$

and $\sigma_B(\tilde{b})$ is the corresponding within-anchor standard deviation. The total learning signal combines trajectory-level success with belief-anchor step-level feedback:

$$A_{i,t}^{\text{tot}} = A_i^E + \omega A_{i,t}^S.$$

This design gives REBEL two complementary supervision channels: A_i^E preserves global task optimization, while $A_{i,t}^S$ provides dense belief-level correction under a shared semantic context.

E.3 Discussion of Belief Supervision and Grouping

We close with a bounded, qualitative account of why the two belief-level mechanisms are expected to help. The argument below is a mechanism description rather than a formal guarantee, and we state explicitly the conditions under which it applies.

Belief consistency as a corrective signal. The consistency reward r_t^{cons} compares the predicate values implied by b_t against the values that the subsequent observation confirms, so it penalizes exactly those assertions that the environment later contradicts. Whenever a predicate is observable at step t, a high consistency reward therefore indicates that the belief agreed with the evidence. Repeated over a trajectory, this supplies a dense signal on the observable part of the belief, in contrast to a terminal outcome that is attributed to every step equally. Two limitations bound this reading. First, the signal is informative only for predicates observable at step t: the mask $m_{t,k}$ excludes unverifiable entries, and a step with $\sum_k m_{t,k} = 0$ contributes nothing. Second, predicates resolved only after a delay are settled by the pending buffer, so the correction reaches the originating step retroactively rather than immediately. Correct feedback on one step does not by itself imply a more accurate belief on the next, and we do not claim such a recursion.

Belief-anchor grouping as a comparison set. The step-level advantage compares a step against other steps sharing the same belief anchor $\tilde{b}$, rather than against steps sharing the same latent state. The practical motivation is support size. Grouping by latent state is unreliable in a POMDP because the same physical state is rarely revisited at the same step, so many groups collapse to a single sample and the local baseline becomes unavailable or noisy. Belief anchors are coarse symbolic summaries, so samples with logically equivalent predicates share an anchor even when their underlying states differ, which enlarges the effective comparison set. Whether a larger group also means a lower-variance estimate depends on what the group mixes: if an anchor aggregates steps whose true states differ, the within-group dispersion can grow and the advantage estimate loses contrast. We therefore treat the benefit as conditional on the anchor being a faithful abstraction of the decision-relevant state, and we report the observed singleton behaviour as an empirical property of this formulation rather than a consequence of the analysis above. The total learning signal $A_{i,t}^{\text{tot}} = A_i^E + \omega A_{i,t}^S$ accordingly keeps the trajectory-level objective intact while adding a step-level correction, without a claim that the correction is uniformly lower in variance.