COGRL: ENHANCING COGNITIVE REASONING OF LLM AGENTS VIA REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in large language models (LLMs) have enhanced agents' reasoning abilities, but deploying them in dynamic environments remains challenging due to misalignment between internal reasoning and environmental dynamics. Existing RL-based methods often couple reasoning with action and rely only on sparse task rewards, leading to ungrounded reasoning and poor sample efficiency—especially in multi-round interactions. We propose CogRL, an integrated RL framework for enhancing two-stage cognitive reasoning. CogRL makes reasoning outcomes verifiable through a Subgoal Reasoning Policy (SRP) that predicts intended next states, supported by an Action Reasoning Policy (ARP) that generates actions to realize these transitions. Its integrated policy optimization combines separately normalized task-return and state-consistency advantages to train the SRP, enabling learning from unsuccessful trajectories even when task rewards are uninformative. The ARP is trained using periodic inverse-dynamics to support stable online learning. Experiments across multiple interactive environments demonstrate improved task success and data efficiency over strong baselines, with further analysis showing reduced state hallucinations. Our code is available at `https://anonymous.4open.science/r/CogRL`.

1 INTRODUCTION

Recent advances in large language models (LLMs) Liu et al. (2024); Achiam et al. (2023) have enabled strong reasoning capabilities across a wide range of tasks Ferrag et al. (2025), including logic Zhang et al. (2025b); Xie et al. (2025), mathematics Shi et al. (2024), and multimodal understanding Dong et al. (2025); Wan et al. (2025); Zhang et al. (2025c). However, deploying an LLM-based reasoning agent in dynamic and interactive environments introduces a fundamental challenge Wang et al. (2024): The reasoning process must strictly align with environmental dynamics to ensure executable and consistent actions. Moreover, given the high interaction cost, the agent needs to quickly understand the environment under few interactions. This gap between agent reasoning and environmental dynamics urges for more stable and efficient agentic RL methods Zhao et al. (2025).

GRPO Shao et al. (2024) and its variants Liu et al. (2025b); Tan et al. (2025) have established effective learning frameworks for optimizing the LLM agent's rewards in single round settings. However, they suffer from poor performance and data inefficiency when extended to multi-round settings, where the environmental dynamics must be considered in the agent's decision-making process Liu et al. (2025a). Enhancing cognitive reasoning through RL in multi-round settings presents three critical challenges. First, the interaction cost between the LLM agent and the environment is usually considerable, especially when the reasoning mode is enabled Cao et al. (2025). This means that the agent has only a limited number of trials to explore the environment. Second, it is hard to judge whether the reasoning process aligns with environmental dynamics due to the lack of explicit reasoning-process rewards Park et al. (2025); She et al. (2025). Consequently, there is no guaranteed improvement in the agent's reasoning capability. Third, existing agent reasoning frameworks often conflate the reasoning process with the final action, ignoring the causal relationship between them Wang et al. (2025). As a result, agents might output actions that totally disobey the environmental constraints.

Early agentic frameworks such as ReAct Yao et al. (2022b) and Reflexion Shinn et al. (2023) enable LLM agents to perceive the environment via in-context learning (ICL) Dong et al. (2024); Wies et al. (2023). Yet, the agents' reasoning capability heavily relies on the

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.

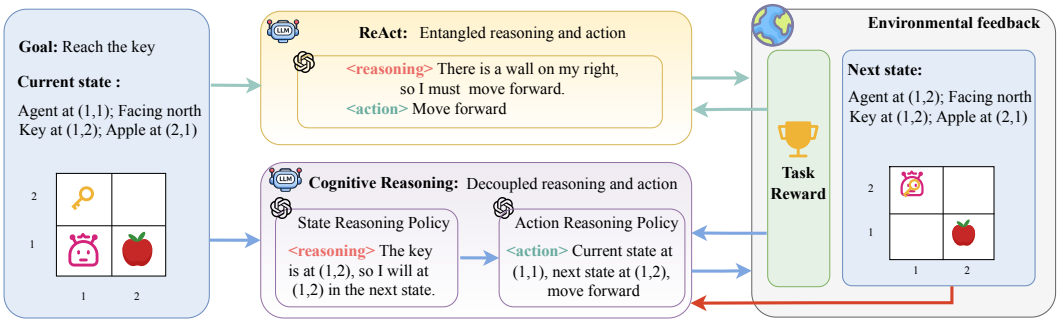


Figure 1: ReAct-style agents entangle reasoning and action, relying only on task rewards and potentially reinforcing hallucinated reasoning when actions succeed. Cognitive reasoning predicts an intended next state that can be verified against observed transitions, enabling CogRL to improve reasoning through RL using both task rewards and environmental feedback.

base model. Recent works employ RL to optimize LLM agents' policies and achieve notable successes Zhang et al. (2025a). However, as illustrated in Figure 1, existing RL-based methods entangle the reasoning process and the final action Wang et al. (2025); Feng et al. (2025). Consequently, the agent can only learn from reward signals but is unaware of environmental dynamics. This reasoning paradigm might cause uncontrollable reasoning hallucinations and low learning efficiency.

To address the above challenges, we propose CogRL, an integrated RL framework to enhance cognitive reasoning for LLM agents. Cognition refers to the agent's ability to sense the environment through continual interactions. Inspired by this, we propose a cognitive reasoning framework which consists of a decoupled **Subgoal Reasoning Policy (SRP)** and an **Action Reasoning Policy (ARP)**. The SRP takes the current state and the task goal as inputs, and reasons about the next subgoal to reach. Note that the quality of reasoned subgoals can be verified by both reward signals and the actual next state returned by the environment. In this way, the agent's reasoning process can be corrected in real-time. The ARP takes the current state and the reasoned subgoal as inputs, and outputs an executable action. This policy can be viewed as an inverse dynamic model Tian et al. (2024), which builds a solid causal relationship among the current state, subgoal and action Tian et al. (2024).

Moreover, we introduce an integrated policy optimization for cognitive reasoning to optimize the agent's two policies simultaneously. To optimize the SRP, we follow the GRPO objective Shao et al. (2024), while grouping rollouts based on the states visited during interaction to enable finer-grained advantage computation. We update the ARP via supervised fine-tuning after every k-step SRP training. Note that the training data of ARP is sampled from a dynamic buffer to ensure the coverage of the state space. This integrated policy optimization enables the agent to learn from both state transitions and reward signals, thereby substantially improving its reasoning capability in dynamic environments. Our main contributions are summarized as follows.

- We propose CogRL, an RL framework to enhance cognitive reasoning for LLM agents that decouples agent decision making into an SRP and an ARP. This new reasoning paradigm enables agent learning from both task rewards and environmental dynamics, leading to significantly lower reasoning hallucination and higher data efficiency.
- We develop an integrated policy optimization method for cognitive reasoning that combines separately normalized task-return and state-consistency advantages to optimize the SRP through RL. Periodic ARP supervised fine-tuning supports stable online learning.
- We conduct extensive experiments across multiple interactive environments to evaluate CogRL in multi-round decision-making scenarios. Experimental results show that CogRL consistently achieves higher task success rates and better data efficiency than strong baselines under limited interaction budgets.

2 Related Works

LLM agent reasoning paradigms. Reasoning is a component of sequential decision making, especially for long-horizon tasks requiring planning and coordination Sun et al. (2025); Plaat et al.

(2024). Early agentic frameworks such as ReAct Yao et al. (2022b) and Reflexion Shinn et al. (2023) introduce reasoning through prompt-based paradigms, where reasoning traces and actions are generated by a single policy. Recent work Wang et al. (2025); Feng et al. (2025) further optimizes such agents with RL, but still entangles reasoning and action generation into a unified policy trained only with task-level rewards. This entanglement can encourage shortcut state-action mappings, causing reasoning to degenerate into ungrounded or hallucinatory rationales when no environment feedback is used to supervise the reasoning process Carta et al. (2023); Tan et al. (2024). In contrast, we decouple reasoning from action generation into two policies, formulating reasoning as an explicit, environment-grounded decision over intended state transitions rather than an implicit byproduct of action optimization.

RL-based post-training for LLM agents. Reinforcement learning (RL) Vincent et al. (2018) has been widely used for post-training LLM agents Cao et al. (2024), and most existing approaches follow the policy-gradient paradigm. While PPO Schulman et al. (2017) and its variants Ouyang et al. (2022); Li et al. (2025) provide dense learning signals through a learned critic in multi-turn decision-making, they often suffer from high computational overhead and training instability. Group-based methods Shao et al. (2024); Yu et al. (2025); Zheng et al. (2025) improve training stability by performing trajectory-level grouping and relative advantage normalization without relying on value functions. These methods compute advantages at the trajectory level based on task reward, which prevents fine-grained credit assignment and fails to learn environment dynamics, leading to degraded performance in multi-turn settings. GiGPO Feng et al. (2025) extends GRPO to multi-turn interaction with step-level credit assignment, but still relies only on task rewards. In contrast, CogRL additionally incorporates environmental transition feedback to assess reasoning consistency. This additional feedback enables the agent to learn from unsuccessful trajectories even when task rewards are uninformative.

3 PRELIMINARIES

Problem Setup. We study sequential decision making with LLM agents interacting with an environment to solve multi-step tasks specified by a goal $g \in \mathcal{G}$. The interaction is modeled as a Markov Decision Process (MDP) Puterman (1990) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $\mathcal{P}(s_{t+1}|s_t, a_t)$ denotes the transition dynamics, and $\mathcal{R}(s_t, a_t)$ denotes the reward function, and $\gamma \in [0, 1]$ is the discount factor.

At each step t, given the task goal g and the current state s_t, an LLM-based agent first generates an explicit internal reasoning trace $z_t \sim \pi_\theta(z_t|s_t)$, and then samples an action $a_t \sim \pi_\theta(a_t|s_t, z_t)$. Unlike traditional agents Mnih et al. (2013) that directly map states to actions, LLM agents introduce this intermediate reasoning process to guide action selection. The action a_t is executed in the environment, which then returns a reward $r_t = \mathcal{R}(s_t, a_t)$ and transitions to the next state $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$. We denote the agent trajectory as $\boldsymbol{\tau} = \{s_0, z_0, a_0, r_0, \ldots, s_T\}$, which records both environment interactions and the internal reasoning decisions generated by the policy. The objective is to learn a policy π that maximizes the expected discounted return $\max_\theta \mathbb{E}_{\boldsymbol{\tau}\sim\pi_\theta}[R(\boldsymbol{\tau})], R(\boldsymbol{\tau}) = \sum_{t=0}^{T-1} \gamma^t r_t$. Since the reasoning trace z_t is internal to the agent and is not directly constrained by the environment, learning reasoning behaviors that reliably guide action selection poses a fundamental challenge.

Group-based RL. Group-based RL optimizes policies Shao et al. (2024) without relying on an explicit critic by comparing multiple trajectories generated under the same task goal. Given a task goal g, the current policy $\pi_{\theta_{\text{old}}}$ samples a group of N agent trajectories $\{\boldsymbol{\tau_1}, \ldots, \boldsymbol{\tau_N}\}$. Each trajectory τ_i is associated with a scalar return $R(\boldsymbol{\tau}_i)$. A group-wise advantage signal is then computed by normalizing returns within the group, for example: $A_i = \frac{R(\boldsymbol{\tau_i}) - \mu_R}{\sigma_R}$, where $\mu_R = \frac{1}{N}\sum_{i=1}^{N} R(\boldsymbol{\tau_i}), \sigma_R^2 = \frac{1}{N}\sum_{i=1}^{N} \big(R(\boldsymbol{\tau_i}) - \mu_R\big)^2$.

4 METHODS

Motivation. In multi-round interaction, LLM agents require environment-grounded reasoning to ensure executable and consistent actions Zeng et al. (2025). However, existing RL-based methods entangle reasoning with action and optimize them jointly using sparse task rewards, which fail to anchor reasoning to the environment. To address this limitation, we introduce cognitive reasoning and

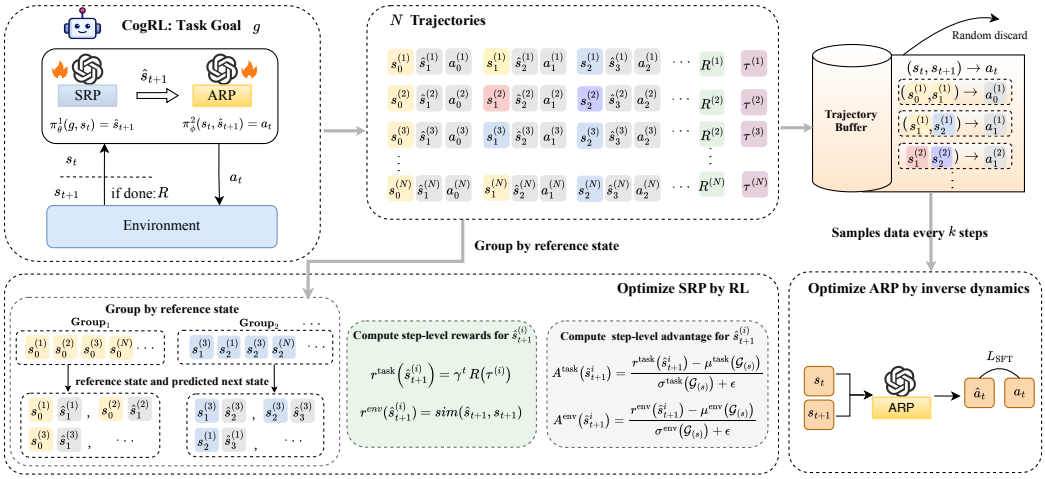


Figure 2: Training overview of CogRL. Given a task goal g, CogRL samples N trajectories through environment interaction. At each step t, the decision process is decoupled into two stages: SRP predicts an intended next state $\hat{s}_{t+1}$ from the current state s_t and goal g, and ARP generates an executable action a_t to realize this transition. SRP is optimized with a GRPO-style objective by grouping step-level samples with the same reference state, enabling fine-grained credit assignment. ARP is periodically updated via inverse-dynamics SFT on (s_t, s_{t+1}, a_t) tuples stored in the buffer.

learning framework for LLM Agent. Concretely, we decouple reasoning and action by first reasoning about *where to go next* as an environment-grounded, verifiable subgoal, and then determining *how to get there* by generating actions to realize it.

Overview. As illustrated in Figure 2, CogRL is an RL framework that enhances cognitive reasoning through task rewards and environmental transition feedback. Given a task goal g, CogRL samples N trajectories through interaction with the environment using a *Subgoal Reasoning Policy (SRP)* and an *Action Reasoning Policy (ARP)*. At each step t, the SRP predicts an intended next state $\hat{s}_{t+1}$ conditioned on g and the current state s_t, while the ARP generates an action a_t to realize this transition. The observed next state s_{t+1} provides feedback on whether the intended transition was realized. CogRL integrates this feedback with task rewards by combining separately normalized task-return and state-consistency advantages to optimize the SRP under a GRPO-style objective. To support stable online learning, the ARP is updated every k SRP training steps through inverse-dynamics SFT on observed transition tuples (s_t, s_{t+1}, a_t) sampled from Trajectory buffer.

4.1 Subgoal Reasoning Policy: Where to Go Next?

We formulate subgoal reasoning as a goal-conditioned next-state prediction problem. In CogRL, a subgoal is defined as a concrete, reachable next environment state that serves as the agent's immediate decision outcome, rather than an abstract symbolic objective or a long-horizon plan. Formally, given the task goal g and the current state s_t, the SRP predicts a target next state $\hat{s}_{t+1} \sim \pi^1_\theta(\cdot \mid g, s_t)$, which specifies the desired environment state after executing a single action. After the agent executes an action, the observed next state s_{t+1} provides immediate feedback on whether the predicted subgoal $\hat{s}_{t+1}$ was successfully realized. As a result, the quality of subgoal reasoning can be assessed by the environment at the step level, rather than only through delayed task rewards.

Supervising next-state prediction in free-form language can be noisy due to diverse surface forms of semantically equivalent states. In our experiments, we use a structured JSON representation Zhou et al. (2024), where each field is parsed from the original observation and corresponds to a well-defined state attribute. This format introduces no extra information, but canonicalizes the same state for reliable field-level matching. JSON is not required by CogRL; the framework only needs a state representation that supports predicted–observed state comparison, such as semantic matching or learned state abstraction. The schema is provided in Appendix A.5.

4.2 Action Reasoning Policy: How to Go There?

While the SRP reasons what environment state to achieve, the ARP is responsible for deciding how to realize the selected state transition through executable actions. Formally, given the current state s_t and the target state $\hat{s}_{t+1}$ predicted by the SRP, the ARP outputs an action $a_t \sim \pi^2_\phi(s_t, \hat{s}_{t+1})$, which aims to realize the intended transition from s_t to $\hat{s}_{t+1}$. In this formulation, the ARP does not reason about task objectives or long-term planning; instead, it conditions on an explicit state-level decision and produces an action that is consistent with the environment dynamics.

By conditioning action generation on both the current state and the predicted target state, the ARP can be viewed as an inverse dynamics model. This design offers two key advantages. First, the ARP is optimized to achieve a one-step state-transition objective rather than to maximize long-horizon task returns, which simplifies the learning problem and stabilizes action execution. Second, inverse-dynamics models capture task-independent environment dynamics, allowing the ARP to be reused across different tasks as long as the environment remains unchanged.

4.3 Integrated Policy Optimization for Cognitive Reasoning

Based on the decoupled decision structure introduced above, we propose an integrated policy optimization that trains the SRP and the ARP through interaction with the environment. At each optimization iteration, we fix the current SRP and ARP and interact with the environment to collect a batch of N trajectories. Each trajectory $\boldsymbol{\tau}^{(i)}$ is recorded as: $\boldsymbol{\tau}^{(i)} = \left\{(s_t^{(i)},\ \hat{s}_{t+1}^{(i)},\ a_t^{(i)})\right\}_{t=0}^{T-1}$, together with its terminal task reward $R^{(i)}$. At the same time, each realized transition $(s_t^{(i)}, \hat{s}_{t+1}^{(i)}, a_t^{(i)})$ is extracted and stored in a replay buffer. The key challenge arises from integrated policy optimization for cognitive reasoning, as the two components operate at different abstraction levels, admit different learning signals, and mutually influence each other during learning.

To address this challenge, we optimize the SRP and ARP using different training strategies. The SRP is optimized via RL to learn next-state predictions that are consistent with environmental dynamics and conducive to long-horizon task progress, guided by environmental feedback and task rewards. In contrast, the ARP does not reason about task objectives or returns, but is trained via inverse-dynamics-based SFT Zhang & Zhang (2024) on ground-truth state transition data, enabling a stable mapping from state transitions to executable actions. Since inverse dynamics are determined by environment mechanics and provide unambiguous supervision, the ARP can be trained with supervised fine-tuning. To stabilize SRP optimization, we update the ARP less frequently, performing updates once every k environment steps. This provides a stable execution model for SRP learning while allowing the ARP to improve as interaction data accumulate. Algorithmic details are provided in Appendix B.

Optimize SRP by State-Grouped Policy Optimization. To optimize the SRP under sparse and delayed task rewards, we follow the GRPO objective Shao et al. (2024) and organize step-level samples according to their reference states. Specifically, samples associated with the same or comparable environment states are grouped together, and step-level advantages are computed within each group based on both task rewards and observed environment transitions.

Reference-State Grouping. To enable meaningful comparison among subgoal reasoning decisions, it is necessary to control for the underlying environment context in which the reasoning is performed. Specifically, we retroactively identify decision points that share the same or comparable environment state representation across trajectories, and refer to such states as *reference states*. In practice, this definition can be instantiated by exact state matching in structured environments and extended to equivalent or similar state representations in more complex environments, keeping the grouping strategy applicable in complex environments. Details of similarity-based state matching and an analysis of reference-state group sizes are provided in Appendix A.2.

These reference states serve as anchors for grouping step-level subgoal predictions under comparable environmental conditions, enabling direct comparison of alternative reasoning outputs and facilitating fine-grained credit assignment. Formally, for each reference state s, we define a group: $\mathcal{G}_s = \{\hat{s}_{t+1}^{(i)} \mid s_t^{(i)} \simeq s,\ 1 \le i \le M,\ 0 \le t < T\}$. Here, $s_t^{(i)} \simeq s$ indicates that the decision state is matched to the reference state by exact matching or state similarity, depending on the environment. This reference-state grouping allows credit assignment to be performed at the step level rather than only at the trajectory level.

Step-level Rewards for $\hat{s}$. The environment provides only sparse terminal task rewards, which offer coarse trajectory-level supervision and are insufficient for optimizing step-level reasoning decisions. To evaluate alternative subgoal predictions within each reference-state group, we introduce two step-level reward signals for each subgoal $\hat{s}_{t+1}^{(i)}$. The first is a step-level task reward propagated from the trajectory-level reward: $r^{\text{task}}\left(\hat{s}_{t+1}^{(i)}\right) = \sum_{k=t}^{T-1} \gamma^{k-t} R^{(i)}$, where $\gamma \in (0,1)$ is a discount factor.

In addition, we introduce a state-consistency reward that evaluates the consistency between the predicted next state and the actual next state observed after action execution: $r^{\text{con}}(\hat{s}_{t+1}^{(i)}) = \text{sim}\left(\hat{s}_{t+1}^{(i)},\, s_{t+1}^{(i)}\right)$. Together, these signals encourage the SRP to propose next states that both advance long-horizon task progress and remain consistent with environment dynamics.

Step-level Advantage for $\hat{s}$. To obtain normalized and comparable learning signals within each reference-state group, we compute group-relative advantages separately for the step-level task reward and state-consistency reward. For a predicted next state $\hat{s}_{t+1}^{(i)} \in \mathcal{G}_s$, we define step-level task advantage and state-consistency advantage

$$A^{\text{task}}(\hat{s}_{t+1}^{(i)}) = \frac{r^{\text{task}}(\hat{s}_{t+1}^{(i)}) - \mu^{\text{task}}(\mathcal{G}_s)}{\sigma^{\text{task}}(\mathcal{G}_s) + \epsilon}, \qquad A^{\text{con}}(\hat{s}_{t+1}^{(i)}) = \frac{r^{\text{con}}(\hat{s}_{t+1}^{(i)}) - \mu^{\text{con}}(\mathcal{G}_s)}{\sigma^{\text{con}}(\mathcal{G}_s) + \epsilon}. \tag{1}$$

We combine the two components at the advantage level:

$$A(\hat{s}_{t+1}^{(i)}) = A^{\text{task}}(\hat{s}_{t+1}^{(i)}) + aw \cdot A^{\text{con}}(\hat{s}_{t+1}^{(i)}), \tag{2}$$

where aw balances long-horizon task progress and environment transition consistency. We combine task-driven and environment-aware signals at the advantage level rather than at the reward level. This allows each signal to be normalized and compared independently within reference-state groups, preserving their relative ordering under different evaluation criteria. Aggregating them in the advantage space avoids entanglement at the reward level and ensures that both task progress and transition consistency contribute to subgoal evaluation.

SRP Optimization Objective. Given the group-relative advantages defined above, we optimize the SRP using a clipped group-relative policy optimization objective:

$$\mathcal{J}(\theta) = \mathbb{E}\left[\frac{1}{MT}\sum_{i=1}^{M}\sum_{t=1}^{T} \min\left(\rho_\theta(\hat{s}_{t+1}^{(i)})\, A(\hat{s}_{t+1}^{(i)}), \text{clip}\Big(\rho_\theta(\hat{s}_{t+1}^{(i)}), 1-\epsilon, 1+\epsilon\Big)\, A(\hat{s}_{t+1}^{(i)})\right)\right], \tag{3}$$

Optimize ARP by Periodic Supervised Fine-Tuning. The ARP is trained to realize a specified environment state transition by producing an executable action. During online interaction, each step yields a transition tuple (s_t, s_{t+1}, a_t), where s_{t+1} is the actual next state returned by the environment after executing action a_t from state s_t. These transition tuples are stored in a replay buffer $\mathcal{B}_{\text{ID}}$ with fixed capacity and serve as direct supervision for inverse-dynamics learning. When the buffer is full, transitions are randomly discarded instead of removed in FIFO order, preserving transition diversity across trajectories and interaction stages.

During training, the ARP conditions on ground-truth transitions (s_t, s_{t+1}), allowing it to learn a stable inverse mapping from state transitions to actions without being affected by inaccurate or unreachable reasoning outputs. At inference time, the ARP instead conditions on $(s_t, \hat{s}_{t+1})$, where $\hat{s}_{t+1}$ is the target next state predicted by the subgoal reasoning policy, and generates an action to realize the intended transition. This asymmetric instantiation is intentional: inverse dynamics are fully determined by environment mechanics and admit unambiguous supervision from transition data, while predicted next states may deviate from observed transitions during early training. As long as predicted states remain within the feasible dynamics support, the ARP can still produce reasonable actions, and execution reliability improves as subgoal reasoning becomes grounded. We analyze this train–inference mismatch and ARP behavior under unreachable targets in Appendix A.3.

The ARP is optimized via supervised fine-tuning by minimizing

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(s_t, s_{t+1}, a_t)\sim\mathcal{B}_{\text{ID}}}\big[\log \pi_\phi(a_t \mid s_t, s_{t+1})\big]. \tag{4}$$

Supervised updates are performed every k environment steps rather than at every step, which improves training stability and prevents frequent executor updates from interfering with SRP optimization.

Table 1: Performance comparison on ALFWorld. Values are success rates (%). Results are averaged over 3 random seeds. Mean±std are computed over multiple runs.

Type	Method	ALFWorld						
		Pick	Look	Clean	Heat	Cool	Pick2	All
Closed-Source Model								
Prompting	GPT-4o	75.3	60.8	31.2	56.7	21.6	49.8	48.0
Prompting	Gemini-2.5-Pro	92.8	63.3	62.1	69.0	26.6	58.7	60.3
Qwen3-4B								
Prompting	Qwen3-4B	30.6	20.8	10.1	5.2	3.1	2.8	12.3
Prompting	ReAct	45.3	31.9	25.5	15.2	8.7	12.2	24.5
Prompting	Reflexion	59.3	40.4	30.9	25.6	30.4	26.8	36.5
RL Training	PPO (with critic)	76.4 ± 9.0	61.0 ± 11.3	64.0 ± 10.1	63.4 ± 14.8	59.3 ± 18.8	20.5 ± 5.2	54.8 ± 3.5
RL Training	RLOO	66.8 ± 24.5	72.7 ± 28.3	87.6 ± 12.7	59.2 ± 28.4	59.5 ± 20.7	58.1 ± 18.4	65.3 ± 7.8
RL Training	GRPO	72.3 ± 19.9	41.5 ± 25.3	37.8 ± 22.4	57.1 ± 26.5	60.2 ± 32.8	46.5 ± 17.9	54.4 ± 7.2
RL Training	GiGPO	84.6 ± 1.3	94.4 ± 3.0	85.8 ± 14.3	84.6 ± 7.4	72.0 ± 4.2	**83.2** ± 2.6	80.0 ± 6.3
RL Training	**CogRL**	**95.6** ± 1.5	**96.4** ± 2.4	**93.4** ± 3.1	**87.5** ± 8.0	**91.6** ± 4.4	78.4 ± 8.4	**85.5** ± 2.4
Qwen2.5-7B-Instruct								
Prompting	Qwen2.5-7B-Instruct	35.8	23.0	21.2	8.2	4.2	4.1	20.1
Prompting	ReAct	54.3	43.5	35.7	20.2	25.2	21.4	37.2
Prompting	Reflexion	62.5	45.2	45.7	32.3	38.6	26.2	45.1
RL Training	PPO (with critic)	76.4 ± 3.4	70.0 ± 12.0	73.9 ± 8.4	82.5 ± 11.0	68.1 ± 2.5	50.0 ± 13.2	70.8 ± 2.5
RL Training	RLOO	87.6 ± 5.3	64.4 ± 3.8	71.7 ± 4.3	61.4 ± 8.0	55.2 ± 2.3	55.0 ± 10.0	68.2 ± 3.5
RL Training	GRPO	71.1 ± 1.9	59.5 ± 13.7	94.5 ± 1.9	81.1 ± 12.3	66.7 ± 4.8	42.0 ± 10.0	71.1 ± 0.8
RL Training	GiGPO	88.8 ± 1.8	**100.0** ± 0.0	**100.0** ± 0.0	94.7 ± 5.3	69.6 ± 4.3	86.7 ± 2.9	89.1 ± 2.1
RL Training	**CogRL**	**100.0** ± 0.0	**100.0** ± 0.0	98.9 ± 1.9	**98.2** ± 3.0	**89.9** ± 2.5	**98.3** ± 2.9	**97.4** ± 0.9

5 ENVIRONMENT

We evaluate CogRL on diverse agentic tasks. Specifically, we aim to demonstrate: **RQ1:** How does CogRL perform compared to closed-source LLMs and agentic RL baselines? **RQ2:** Which components of CogRL contribute to its performance gains? **RQ3:** Does environment-grounded reinforcement learning reduce transition-level reasoning hallucinations?

5.1 EXPERIMENT SETUP

Benchmarks. We evaluate our method on three challenging interactive-agent benchmarks: **ALFWorld** Shridhar et al. (2020), **BabyAI** Chevalier-Boisvert et al. (2018), and **WebShop** Yao et al. (2022a). ALFWorld is a text-based embodied environment that evaluates an agent's ability to perform long-horizon, multi-step decision making. In each episode, the agent receives a natural-language goal and must accomplish it through multi-turn interaction with a household-like environment involving navigation and object manipulation. BabyAI is a grid-world benchmark designed to study instruction following and compositional generalization. The agent receives a concise natural-language instruction and must solve the task by reasoning over spatial relations and sequential subgoals under partial observability and sparse success signals Dwaracherla et al. (2024); Rocamonde et al. (2023). WebShop is a language-mediated web-shopping environment in which the agent must navigate product pages and select items that satisfy natural-language requirements. Compared with ALFWorld and BabyAI, WebShop provides more open-ended textual observations and a larger space of decision-relevant states. Together, these benchmarks cover long-horizon tasks, instruction following, and open-ended web interaction, enabling evaluation across diverse environments.

Baselines. We compare CogRL with baselines from three categories. (1) **Closed-source LLMs.** We report GPT-4o Achiam et al. (2023) and Gemini-2.5-Pro Team et al. (2023) as strong reference models for general reasoning and interactive decision making. (2) **Prompt-based agents.** We include ReAct Yao et al. (2022b) and Reflexion Shinn et al. (2023), which generate multi-step behaviors through prompting without updating model parameters. (3) **RL-based training methods.** We compare with PPO Schulman et al. (2017), a standard actor-critic algorithm with an auxiliary value function, as well as critic-free group-based methods including RLOO Ahmadian et al. (2024), GRPO Shao et al. (2024), and GiGPO Feng et al. (2025). We use the same backbone and comparable training protocols across learned baselines whenever supported by the corresponding environment.

Training details. We use Qwen2.5-7B-Instruct Team et al. (2024) and Qwen3-4B Yang et al. (2025) as backbones. For each backbone, all learned methods share the same initialization and online interaction budget. SRP and ARP use separate LoRA adapters: SRP is optimized through RL, while ARP undergoes SFT every 10 RL training steps on transition tuples (s, s', a) from the replay buffer.

Table 2: Performance comparison on BabyAI and WebShop. Values are success rates (%) for BabyAI and score, success rate (%) for WebShop. Results are averaged over 3 random seeds. Mean±std are computed over multiple runs.

Type	Method	BabyAI					WebShop	
		go to	open and open	pick up	open	All	Score	Succ.
Closed-Source Model								
Prompting	GPT-4o	78.2	0.0	38.9	64.0	38.0	31.8	23.7
Prompting	Gemini-2.5-Pro	81.9	18.8	43.8	75.6	45.1	42.5	35.9
Qwen3-4B								
Prompting	Qwen3-4B	18.1	0.0	6.2	12.4	12.1	16.0	3.1
Prompting	ReAct	25.3	0.0	16.7	20.1	17.5	42.8	15.3
Prompting	Reflexion	27.2	0.0	15.8	22.0	20.3	50.4	22.2
RL Training	PPO (with critic)	62.3 ± 6.5	1.9 ± 3.0	19.2 ± 6.5	60.7 ± 10.5	36.1 ± 3.4	66.0 ± 5.9	55.9 ± 4.7
RL Training	RLOO	55.9 ± 5.8	2.1 ± 2.5	14.3 ± 2.5	55.2 ± 3.3	30.0 ± 5.2	67.4 ± 8.3	56.0 ± 4.2
RL Training	GRPO	66.7 ± 8.5	1.1 ± 2.1	13.5 ± 4.4	54.3 ± 6.2	35.4 ± 13.0	70.4 ± 5.7	55.8 ± 8.3
RL Training	GiGPO	$\mathbf{84.0} \pm 2.1$	15.8 ± 4.4	35.0 ± 3.3	86.7 ± 11.7	55.4 ± 1.6	$\mathbf{78.0} \pm 3.1$	59.4 ± 2.6
RL Training	**CogRL**	82.4 ± 3.8	$\mathbf{17.4} \pm 2.5$	$\mathbf{37.0} \pm 3.0$	$\mathbf{90.1} \pm 6.8$	$\mathbf{60.8} \pm 5.5$	72.0 ± 2.3	$\mathbf{64.8} \pm 1.4$
Qwen2.5-7B-Instruct								
Prompting	Qwen2.5-7B-Instruct	25.4	0.0	7.1	18.1	17.2	28.3	8.2
Prompting	ReAct	33.3	0.0	16.2	27.2	20.5	55.1	21.1
Prompting	Reflexion	36.3	0.0	21.0	32.8	25.7	57.3	30.4
RL Training	PPO (with critic)	67.6 ± 8.8	2.1 ± 3.6	21.7 ± 0.7	61.5 ± 15.6	41.6 ± 6.3	71.7 ± 5.4	62.7 ± 2.3
RL Training	RLOO	57.3 ± 6.2	5.4 ± 5.3	15.5 ± 3.0	58.3 ± 3.1	36.7 ± 2.7	69.5 ± 4.6	58.9 ± 2.1
RL Training	GRPO	65.3 ± 9.8	3.2 ± 2.8	17.5 ± 2.8	53.9 ± 12.0	39.8 ± 7.6	72.3 ± 3.4	60.1 ± 1.4
RL Training	GiGPO	90.1 ± 4.0	16.7 ± 6.0	38.3 ± 5.3	86.3 ± 15.2	61.7 ± 1.3	78.0 ± 3.1	65.3 ± 1.8
RL Training	**CogRL**	$\mathbf{95.8} \pm 2.5$	$\mathbf{32.5} \pm 4.4$	$\mathbf{46.5} \pm 4.5$	$\mathbf{91.5} \pm 10.3$	$\mathbf{72.1} \pm 3.2$	$\mathbf{82.7} \pm 4.1$	$\mathbf{70.2} \pm 3.5$

All RL methods use consistent training protocols and hyperparameters, with rollout group size $N = 8$ for group-based methods. Full settings are provided in Appendix A.1.

5.2 Overall Performance of CogRL (RQ1)

Overall Performance. As shown in Tables 1 and 2, CogRL consistently outperforms prior RL baselines on ALFWorld and BabyAI. With Qwen2.5-7B-Instruct, CogRL achieves **97.4%** on ALFWorld and **72.1%** on BabyAI, surpassing GiGPO by **8.3** and **10.4** percentage points, respectively. It also obtains a substantial gain on the challenging Pick2 task, reaching **98.3%** compared with GiGPO's **86.7%**. With Qwen3-4B, CogRL achieves **85.5%** on ALFWorld and **60.8%** on BabyAI, exceeding GiGPO by **5.5** and **5.4** points. These results demonstrate the benefit of combining decoupled subgoal and action reasoning with environment-grounded RL.

On the more open-ended WebShop benchmark, CogRL achieves a success rate of **70.2%** with Qwen2.5-7B-Instruct and **64.8%** with Qwen3-4B, improving over GiGPO by **4.9** and **5.4** percentage points, respectively. Overall, the consistent gains across environments and backbone models show that CogRL generalizes to both structured and open-ended interactive tasks.

Data Efficiency. Figure 3 compares the training dynamics of CogRL with GRPO and GiGPO on ALFWorld. With the same number of training steps, CogRL converges faster and consistently achieves higher success rates, indicating improved data efficiency. Traditional RL-based LLM agents suffer from low data efficiency due to their reliance on sparse task-level rewards. In contrast, CogRL extracts learning signals not only from task rewards but also from environmental dynamics, enabling the construction of effective, fine-grained, step-level advantages. As a result, the agent can learn from unsuccessful trajectories and obtain meaningful gradients even when all sampled rollouts fail, thereby accelerating policy improvement and convergence.

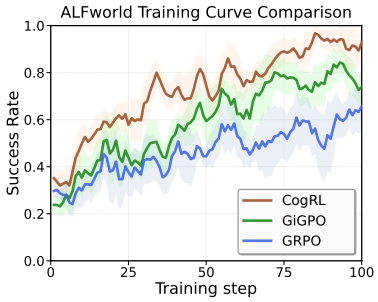
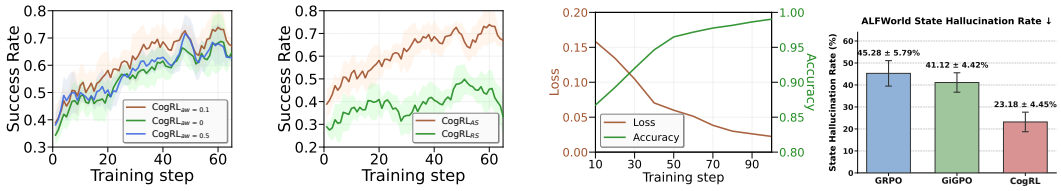


Figure 3: Training curves of CogRL and baselines on ALFWorld.

Convergence of ARP Training. To assess the feasibility and stability of the ARP, we analyze its SFT process on the BabyAI environment. As shown in Figure 4c, the ARP training loss decreases monotonically, while the corresponding accuracy steadily increases and approaches 1.0. These trends indicate that the ARP can reliably learn inverse dynamics from environment transition data and produce accurate low-level actions. This stable convergence demonstrates that the ARP can be trained independently and does not become a bottleneck during training, supporting the feasibility and effectiveness of our decoupled SRP/ARP framework.

5.3 Ablation Studies (RQ2)

Sensitivity to state-consistency Advantage Weight. Figure 4a studies CogRL's sensitivity to the state-consistency advantage weight (aw), which modulates A^{con}. Across values (0, 0.5), CogRL shows stable training behavior and comparable performance, indicating limited sensitivity to this hyperparameter. Among the tested settings, $aw = 0.1$ consistently yields the best overall performance. When $a_w = 0$, CogRL lacks state-consistency feedback and depends exclusively on step-level task reward, leading to notably inferior early-stage learning compared to $a_w = 0.1$. This disparity underscores the importance of state-consistency advantage: while the framework is robust to weight variations, complete ablation ($a_w = 0$) significantly impairs early-stage learning efficiency.



(a) (b) (c) (d)

Figure 4: Training analysis on BabyAI and state hallucination evaluation with Qwen2.5-7B-Instruct. (a) State-consistency advantage weight sensitivity; (b) Advantage vs. Reward Summation; (c) ARP SFT training curves. (d) State hallucination rate comparison

Comparison Between Advantage and Reward Summation. Figure 4b compares the full CogRL (CogRL_{AS}), which uses an advantage formulation for state-consistency feedback, with a variant (CogRL_{RS}) that directly sums state-consistency and step-level task rewards. Direct reward summation leads to noticeably lower success rates. Even with a small state-consistency weight, CogRL_{RS} tends to overemphasize matching predicted and ground-truth next states at the expense of long-horizon task completion. In contrast, CogRL_{AS} preserves the relative contribution of different learning signals, allowing environment feedback to guide learning without distorting the task objective, and consistently achieves higher performance.

5.4 Transition Hallucination Analysis (RQ3)

To assess reasoning quality beyond task success, we measure the state hallucination rate over the final five training steps on ALFWorld. We train GRPO and GiGPO using the same reasoning paradigm, state representation, and transition evaluator as CogRL, while applying their respective RL objectives. Following the strict next-state evaluation protocol, we define the hallucination rate as $H = 1 - \text{Acc}_{\text{strict}}$, where $\text{Acc}_{\text{strict}}$ denotes strict next-state prediction accuracy. A predicted transition is considered correct only when its atomic state change exactly matches the corresponding environment transition. We report the mean hallucination rate over the final five training steps, with error bars indicating the standard deviation across these steps.

As shown in Figure 4d, CogRL achieves the lowest hallucination rate, with $23.18\% \pm 4.45\%$, compared with $41.12\% \pm 4.42\%$ for GiGPO and $45.28\% \pm 5.79\%$ for GRPO. This corresponds to reductions of 17.94 and 22.10 percentage points relative to GiGPO and GRPO, respectively. These results show that environment-grounded RL provides a substantial advantage in reducing environment-unsupported transition predictions, although it does not eliminate transition-level errors.

6 Conclusions

In this work, we proposed CogRL, an RL framework for enhancing cognitive reasoning in LLM agents through task rewards and environmental transition feedback. CogRL integrates separately normalized task-return and state-consistency advantages to optimize the SRP, while periodic inverse-dynamics SFT of the ARP supports stable online learning. We evaluate CogRL on ALFWorld, BabyAI, and WebShop and analyze its overall performance, data efficiency, and state hallucination rate over strong prompting and RL baselines.

AI USE STATEMENT

In this work, we used generative AI tools to assist with research ideation, literature retrieval, and manuscript writing and revision. These tools supported discussions of CogRL, our reinforcement learning framework for enhancing cognitive reasoning in LLM agents; helped identify relevant literature on agentic reinforcement learning and environment-grounded reasoning; and assisted in improving the clarity, organization, and readability of the manuscript.

The authors take full responsibility for the research and its presentation, including the validity of the reported results, the accuracy of citations, and all text, claims, and artifacts developed with AI assistance.

REPRODUCIBILITY STATEMENT

We describe CogRL and its optimization objectives in Section 4. The main experimental section summarizes the benchmarks, backbone models, training protocols, and interaction budgets, while detailed hyperparameters and hardware configurations are provided in Appendix A.1. Prompts for the SRP and ARP are included in Appendix A.6. Main RL results are reported as means and standard deviations over three independent random seeds. Our anonymous implementation is available at `https://anonymous.4open.science/r/CogRL`.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

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. *arXiv preprint arXiv:2402.14740*, 2024.

Shiyi Cao, Dacheng Li, Fangzhou Zhao, Shuo Yuan, Sumanth R Hegde, Connor Chen, Charlie Ruan, Tyler Griggs, Shu Liu, Eric Tang, et al. Skyrl-agent: Efficient rl training for multi-turn llm agent. *arXiv preprint arXiv:2511.16108*, 2025.

Yuji Cao, Huan Zhao, Yuheng Cheng, Ting Shu, Yue Chen, Guolong Liu, Gaoqi Liang, Junhua Zhao, Jinyue Yan, and Yun Li. Survey on large language model-enhanced reinforcement learning: Concept, taxonomy, and methods. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, pp. 3676–3713. PMLR, 2023.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*, 2018.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, et al. A survey on in-context learning. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 1107–1128, 2024.

Yuhao Dong, Zuyan Liu, Hai-Long Sun, Jingkang Yang, Winston Hu, Yongming Rao, and Ziwei Liu. Insight-v: Exploring long-chain visual reasoning with multimodal large language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 9062–9072, 2025.

Vikranth Dwaracherla, Seyed Mohammad Asghari, Botao Hao, and Benjamin Van Roy. Efficient exploration for llms. *arXiv preprint arXiv:2402.00396*, 2024.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*, 2025.

Mohamed Amine Ferrag, Norbert Tihanyi, and Merouane Debbah. From llm reasoning to autonomous ai agents: A comprehensive review. *arXiv preprint arXiv:2504.19678*, 2025.

Junbo Li, Peng Zhou, Rui Meng, Meet P Vadera, Lihong Li, and Yang Li. Turn-ppo: Turn-level advantage estimation with ppo for improved multi-turn rl in agentic llms. *arXiv preprint arXiv:2512.17008*, 2025.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Junnan Liu, Linhao Luo, Thuy-Trang Vu, and Gholamreza Haffari. Situatedthinker: Grounding llm reasoning with real-world through situated thinking. *arXiv preprint arXiv:2505.19300*, 2025a.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025b.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Taekhyun Park, Yongjae Lee, and Hyerim Bae. Reasoning-aware grpo using process mining. *arXiv preprint arXiv:2510.25065*, 2025.

Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Back. Reasoning with large language models, a survey. *arXiv preprint arXiv:2407.11511*, 2024.

Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning. *arXiv preprint arXiv:2310.12921*, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

Shuaijie She, Junxiao Liu, Yifeng Liu, Jiajun Chen, Xin Huang, and Shujian Huang. R-prm: Reasoning-driven process reward modeling. *arXiv preprint arXiv:2503.21295*, 2025.

Wenhao Shi, Zhiqiang Hu, Yi Bin, Junhua Liu, Yang Yang, See Kiong Ng, Lidong Bing, and Roy Ka-Wei Lee. Math-llava: Bootstrapping mathematical reasoning for multimodal large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 4663–4680, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Jiankai Sun, Chuanyang Zheng, Enze Xie, Zhengying Liu, Ruihang Chu, Jianing Qiu, Jiaqi Xu, Mingyu Ding, Hongyang Li, Mengzhe Geng, et al. A survey of reasoning with foundation models: Concepts, methodologies, and outlook. *ACM Computing Surveys*, 57(11):1–43, 2025.

Hongze Tan, Jianfei Pan, Jinghao Lin, Tao Chen, Zhihang Zheng, Zhihao Tang, and Haihua Yang. Gtpo and grpo-s: Token and sequence-level reward shaping with policy entropy. *arXiv preprint arXiv:2508.04349*, 2025.

Weihao Tan, Wentao Zhang, Shanqi Liu, Longtao Zheng, Xinrun Wang, and Bo An. Twosome: An efficient online framework to align llms with embodied environments via reinforcement learning. *International Journal of Artificial Intelligence and Robotics Research*, 1(02):2450004, 2024.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Qwen Team et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2(3), 2024.

Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. *arXiv preprint arXiv:2412.15109*, 2024.

Francois-Lavet Vincent, Henderson Peter, Islam Riashat, G Bellemare Marc, and Pineau Joelle. An introduction to deep reinforcement learning. *Foundations and trends® in machine learning*, 11 (3-4):219–354, 2018.

Zhongwei Wan, Zhihao Dou, Che Liu, Yu Zhang, Dongfei Cui, Qinjian Zhao, Hui Shen, Jing Xiong, Yi Xin, Yifan Jiang, et al. Srpo: Enhancing multimodal llm reasoning via reflection-aware reinforcement learning. *arXiv preprint arXiv:2506.01713*, 2025.

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

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

Noam Wies, Yoav Levine, and Amnon Shashua. The learnability of in-context learning. *Advances in Neural Information Processing Systems*, 36:36637–36651, 2023.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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, 2022a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022b.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Siliang Zeng, Quan Wei, William Brown, Oana Frunza, Yuriy Nevmyvaka, Yang Katie Zhao, and Mingyi Hong. Reinforcing multi-turn reasoning in llm agents via turn-level credit assignment. In *ICML 2025 Workshop on Computer Use Agents*, 2025.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, et al. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*, 2025a.

Jinghan Zhang, Xiting Wang, Weijieying Ren, Lu Jiang, Dongjie Wang, and Kunpeng Liu. Ratt: A thought structure for coherent and correct llm reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 26733–26741, 2025b.

Zhenyu Zhang, Tianyi Chen, Weiran Xu, Alex Pentland, and Jiaxin Pei. Recap: Recursive context-aware reasoning and planning for large language model agents. *arXiv preprint arXiv:2510.23822*, 2025c.

Zhuosheng Zhang and Aston Zhang. You only look at screens: Multimodal chain-of-action agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3132–3149, 2024.

Bingxi Zhao, Lin Geng Foo, Ping Hu, Christian Theobalt, Hossein Rahmani, and Jun Liu. Llm-based agentic reasoning frameworks: A survey from methods to scenarios. *arXiv preprint arXiv:2508.17692*, 2025.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Siyu Zhou, Tianyi Zhou, Yijun Yang, Guodong Long, Deheng Ye, Jing Jiang, and Chengqi Zhang. Wall-e: World alignment by rule learning improves world model-based llm agents. *arXiv preprint arXiv:2410.07484*, 2024.

A Experiment details

A.1 Training details

Hyperparameters for ALFWorld. For all experiments, we adopt Qwen2.5-7B-Instruct and Qwen3-4B as the backbone model and keep the core training hyperparameters identical across all reinforcement learning algorithms to ensure a fair comparison. Specifically, the maximum prompt length is set to 2048 tokens, and the maximum response length is set to 512 tokens. The actor learning rate is fixed to 5×10^{-5}. For all group-based RL methods, we use a group size of 8 and sample 16 different groups per rollout, resulting in a total of 16 × 8 = 128 environments. Each episode allow a maximum of 50 training steps. All model weights are loaded in *safetensors* format. LoRA is applied uniformly across all RL methods. The LoRA rank and scaling factor are set to 64 and 64, respectively, and the same LoRA configuration is used unless otherwise specified. **CogRL-Specific Settings:** In addition to the shared configuration above, CogRL introduces an auxiliary prediction-based advantage with a weight of 0.1. Moreover, CogRL employs a dual-LoRA design within a single *actor_rollout_ref* module, where the *actor* and the *inverse dynamics* components are equipped with separate LoRA adapters. For the inverse dynamics adapter, the LoRA rank and scaling factor are also set to 64 and 64, respectively. If not explicitly specified, the inverse dynamics adapter reuses the same LoRA hyperparameter configuration as the actor adapter. This design enables CogRL to decouple policy learning and inverse dynamics modeling while maintaining a shared backbone representation.

Hyperparameters for BabyAI. For all experiments, we adopt Qwen2.5-7B-Instruct and Qwen3-4B as the backbone model and keep the core training hyperparameters identical across all reinforcement learning algorithms to ensure a fair comparison. Specifically, the maximum prompt length is set to 2048 tokens, and the maximum response length is set to 512 tokens. The actor learning rate is fixed to 5×10^{-5}. For all group-based RL methods, we use a group size of 8 and sample 16 different groups per rollout, resulting in a total of 16 × 8 = 128 environments. Each episode allows a maximum of 30 training steps. All model weights are loaded in *safetensors* format. LoRA is applied uniformly across all RL methods. The LoRA rank and scaling factor are set to 64 and 64, respectively, and the same LoRA configuration is used unless otherwise specified. **CogRL-Specific Settings:** In addition to the shared configuration above, CogRL introduces an auxiliary prediction-based advantage with a weight of 0.1. Moreover, CogRL employs a dual-LoRA design within a single *actor_rollout_ref* module, where the *actor* and the *inverse dynamics* components are equipped with separate LoRA adapters. For the inverse dynamics adapter, the LoRA rank and scaling factor are also set to 64 and 64, respectively. If not explicitly specified, the inverse dynamics adapter reuses the same LoRA hyperparameter configuration as the actor adapter. This design enables CogRL to decouple policy learning and inverse dynamics modeling while maintaining a shared backbone representation.

Hyperparameters for WebShop. For all experiments, we adopt Qwen2.5-7B-Instruct and Qwen3-4B as the backbone model and keep the core training hyperparameters identical across all reinforcement learning algorithms to ensure a fair comparison. Specifically, the maximum prompt length is set to 4096 tokens, and the maximum response length is set to 512 tokens. The actor learning rate is fixed to 5×10^{-5}. For all group-based RL methods, we use a group size of 8 and sample 16 different groups per rollout, resulting in a total of 16 × 8 = 128 environments. Each episode allows a maximum of 15 training steps. All model weights are loaded in *safetensors* format. LoRA is applied uniformly across all RL methods. The LoRA rank and scaling factor are set to 64 and 64, respectively, and the same LoRA configuration is used unless otherwise specified. **CogRL-Specific Settings:** In addition to the shared configuration above, CogRL introduces an auxiliary prediction-based advantage with a weight of 0.1. Moreover, CogRL employs a dual-LoRA design within a single *actor_rollout_ref* module, where the *actor* and the *inverse dynamics* components are equipped with separate LoRA adapters. For the inverse dynamics adapter, the LoRA rank and scaling factor are also set to 64 and 64, respectively. If not explicitly specified, the inverse dynamics adapter reuses the same LoRA hyperparameter configuration as the actor adapter. This design enables CogRL to decouple policy learning and inverse dynamics modeling while maintaining a shared backbone representation.

Computing Details. All experiments on ALFWorld ,BabyAI and WebShop are conducted using Qwen2.5-7B-Instruct and Qwen3-4B with LoRA fine-tuning, and each experiment is trained for a total of 100 training steps. The default experimental setup requires only two NVIDIA A800 GPUs, each equipped with 80 GB of GPU memory, which is sufficient to support both reinforcement learning and supervised fine-tuning stages. To further accelerate training, experiments can be scaled to four or eight GPUs. In such cases, training speed can be significantly improved by increasing the number of CPU cores allocated to each environment instance, enabling more efficient environment parallelism and faster rollout collection.

A.2 Reference-State Grouping: Definition and Empirical Analysis

This appendix defines reference-state grouping in open-ended environments and examines the distribution of group sizes during training. Similarity-based matching extends group construction beyond exact state equality while leaving the SRP optimization objective unchanged.

Decision-relevant state abstraction. In open-ended environments, raw observations may differ in wording, layout, or other details that do not affect the current decision. Exact observation matching can therefore exclude states that support meaningful comparison of subgoal predictions. To account for such variation, we define a goal-conditioned abstraction $\phi_g(s)$ that extracts decision-relevant information from state s under task goal g. Matching is then performed over these abstractions rather than complete raw observations.

Reference-state matching. For a fixed task goal g, a decision state $s_t^{(i)}$ matches a reference state s when

$$s_t^{(i)} \simeq_g s \quad \Longleftrightarrow \quad \mathrm{sim}\big(\phi_g(s_t^{(i)}), \phi_g(s)\big) \geq \eta, \tag{5}$$

where $\mathrm{sim}(\cdot,\cdot)$ measures similarity between the extracted representations and the matching threshold is set to $\eta = 0.8$ in our experiments. The corresponding reference-state group contains the subgoal predictions made from matching decision states:

$$\mathcal{G}_{g,s} = \left\{ \hat{s}_{t+1}^{(i)} \;\middle|\; s_t^{(i)} \simeq_g s,\ 1 \leq i \leq M,\ 0 \leq t < T \right\}. \tag{6}$$

Here, the trajectories are collected under the same goal g, and each prediction occurrence is retained as a separate sample. Each decision state is compared with the reference state; matching does not require equality of the full observations.

Example in WebShop. In WebShop, $\phi_g(s)$ represents the current page using fields relevant to the user instruction, including page type, product category, and applicable constraints such as color, size, price, and brand. These fields are extracted from the page in the context of the task goal. Irrelevant descriptions, promotional text, and formatting differences are excluded from the comparison.

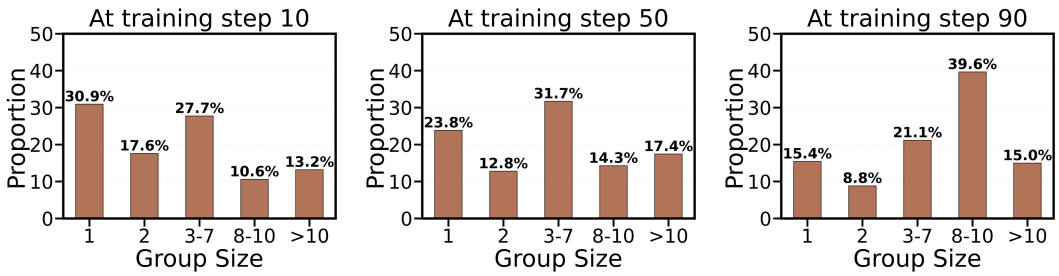
For example, when the task is to purchase a red T-shirt in a specified size, comparable product-page states should agree on the required product attributes and size availability. Differences in unrelated descriptions or page layout need not prevent a match. Conversely, a mismatch in a task-critical

attribute, such as color or the availability of the required size, should prevent grouping even when the overall page text is similar.

This criterion allows subgoal predictions to be compared under similar decision contexts even when identical raw observations rarely recur, supporting step-level credit assignment in open-ended environments.

Distribution of reference-state group sizes. To examine the availability of samples for within-group comparison, Figure 5 reports group-size distributions at training steps 10, 50, and 90. Group size denotes the number of decision samples assigned to a reference state. Across these checkpoints, singleton groups account for no more than 31% of all groups, indicating that at least 69% contain multiple samples for comparison.

From step 10 to step 50, the proportions of size-1 and size-2 groups decrease, while groups of size 3–7 become more prevalent. By step 90, the distribution shifts further toward larger groups, with sizes 8–10 becoming dominant. These observations indicate that sampled decision contexts increasingly fall into reference-state groups containing multiple samples, providing more opportunities to compare subgoal predictions and estimate relative advantages. Such comparisons yield informative learning signals when task returns or state-consistency scores differ within a group.



(a) Training step 10. (b) Training step 50. (c) Training step 90.

Figure 5: Reference-state group-size distributions during training. The x-axis denotes the number of decision samples per group, and the y-axis denotes the percentage of groups with that size.

A.3 Train–Inference Mismatch in the ARP

Source of the mismatch. During supervised training, the ARP learns inverse dynamics from observed state pairs (s_t, s_{t+1}). During online rollouts and inference, it instead conditions on SRP-predicted targets $(s_t, \hat{s}_{t+1})$, which may be inaccurate or unreachable, especially early in training. Observed transitions provide clean supervision for the ARP, but do not eliminate this input mismatch.

Behavior under unreachable targets. We observe two outcomes: an invalid or non-state-changing action leaves the state unchanged, or a valid action leads to a legal state different from the target. Table 3 summarizes these outcomes.

Table 3: ARP behavior under unreachable SRP targets. Unreachable-target percentages are computed over all interactions; percentages in the last two columns are computed over unreachable-target cases.

Checkpoint	Unreachable targets	Valid action to a different state	No state change
Step 2	776/5,248 (14.79%)	305/776 (39.30%)	471/776 (60.70%)
Step 50	94/3,200 (2.94%)	26/94 (27.66%)	68/94 (72.34%)
Step 100	28/1,792 (1.56%)	4/28 (14.29%)	24/28 (85.71%)

Unreachable targets decrease from 14.79% to 1.56% during training. Cases leading to a different legal state decrease from 5.81% to 0.22% of all interactions, while 85.71% of the remaining unreachable-target cases at step 100 leave the state unchanged. These results show that invalid-action rate alone cannot fully capture failures to realize predicted targets.

Table 4: Performance on ALFWorld under the same wall-clock budget using Qwen2.5-7B-Instruct. Final k denotes the mean success rate over the last k updates; Best 20 denotes the highest mean success rate over any consecutive 20-update window. Improvements are reported in percentage points.

Method	Time (h)	Updates	Final 5 (%)	Final 10 (%)	Final 20 (%)	Best 20 (%)
GiGPO	19.46	150	91.42	88.37	88.37	89.18
CogRL	19.46	116	**95.30**	**93.82**	**93.01**	**93.01**
Improvement	–	−34	+3.88	+5.45	+4.64	+3.83

Corrective feedback and replanning. CogRL has no explicit recovery controller, but can continue interacting and replan from the observed state. Predicted–observed mismatches provide state-consistency feedback, discouraging inconsistent SRP predictions when the feedback differentiates samples within a group. Thus, observed transitions provide stable supervision for the ARP, while environmental feedback guides the SRP toward more feasible targets.

A.4 Performance under a Matched Wall-Clock Budget

To evaluate training-time efficiency, we compare CogRL with GiGPO on ALFWorld under the same wall-clock budget. Both methods use Qwen2.5-7B-Instruct, eight NVIDIA H800 GPUs, and identical batch sizes, sampling scales, and interaction horizons. The runtime of CogRL includes all computation for subgoal prediction, action generation, RL updates, and periodic ARP supervised fine-tuning.

As shown in Table 4, CogRL completes 116 updates within 19.46 hours, compared with 150 for GiGPO, reflecting its additional computation. Nevertheless, CogRL achieves higher success rates across all reported averaging windows, improving the final 5-, 10-, and 20-update averages by 3.88, 5.45, and 4.64 percentage points, respectively. Its final 20-update average also exceeds GiGPO's best 20-update average by 3.83 percentage points. These results demonstrate that CogRL achieves better task performance within the same training-time budget, even after accounting for its additional computational costs.

A.5 Environments' State Space

ALFWorld State Space As illustrated in Figure 6, the ALFWorld environment adopts a structured, dictionary-based state representation that encapsulates all critical information required for agent reasoning. Key components of the state space include: `reachable_locations` enumerating all navigable positions (e.g., `countertop 1`, `fridge 1`); `items_in_locations` mapping each location to its contained objects (e.g., `apple 1` and `tomato 1` stored in `fridge 1`); `current_position` specifying the agent's current location and interaction status; `visible_item_in_current_location` listing objects observable at the agent's current position; and `item_in_hand` tracking the object held by the agent along with its state (e.g., a cleaned `plate 2`). This explicit, hierarchical state structure provides precise semantic grounding for environmental dynamics and object interactions, serving as a foundational basis for the agent to execute complex sequential tasks in ALFWorld. The fields highlighted in red correspond to the components that can be modified by the agent: at each step, the agent predicts the next reachable state by updating exactly one of these fields, for example changing `{'location_name':'shelf 6','status':null}` to `{'location_name':'countertop 1','status':null}` to indicate a transition to the next state.

BabyAI State Space As shown in Figure 7, the BabyAI environment employs a compact, task-oriented state representation optimized for navigation and interactive door manipulation tasks. The state space is structured into three core components: `agent_state` tracks the agent's current position (e.g., `[9, 6]`), facing direction (e.g., `'north'`), and held item; `front_cell` describes the type and status of the cell immediately ahead of the agent; and `visible_objects` enumerates observable environmental entities (e.g., `closed green door 1`) with their absolute positions and relative offsets to the agent (e.g., `'ahead': 0, 'side': 'east'`). This state design

Example for ALFWorld's state space

```
State = {
            'reachable_locations': [
                'cabinet 2',
                'cabinet 1',
                'coffeemachine 1',
                'countertop 2',
                'diningtable 1',
                'drawer 1',
                'fridge 1',
                'garbagecan 1',
                'microwave 1',
                'sinkbasin 1',
                'toaster 1'
            ],
            'items_in_locations': {
                'countertop 1': [
                    'bread 1',
                    'butterknife 2',
                    'fork 1',
                    'plate 1',
                    'saltshaker 1',
                    'spoon 3'
                ],
                'fridge 1': [
                    'apple 1',
                    'mug 1',
                    'tomato 1'
                ],
                'sinkbasin 1': [
                    'potato 1',
                    'spoon 2'
                ]
            },
            'current_position' : {
                'location_name' : 'countertop 1',
                'status' : None
            },
            'visible_item_in_current_location' : [
                'bread 1',
                'butterknife 2',
                'fork 1',
                'plate 2',
                'plate 1',
                'saltshaker 1',
                'spoon 3'
            ],
            'item_in_hand' : {
                'item_name' : 'plate 2',
                'status' : 'cleaned'
            }
        }
```

Figure 6: Example for ALFWorld's state space

prioritizes spatial awareness and immediate interaction context, enabling the agent to efficiently reason about movement and object interactions in grid-based BabyAI scenarios. The fields highlighted in red correspond to the components that can be modified by the agent: at each step, the agent predicts the next reachable state by updating exactly one of these fields, for example changing `{'agent_state':'pos':[9,6]}` to `{'agent_state':'pos':[9,7]}` to indicate a transition to the next state.

WebShop State Space As shown in Figure 8, the WebShop environment uses a structured JSON state representation for modeling search, product selection, option configuration, and purchase completion. The state contains three main groups of information. First, the page context records the current webpage through `page_type`, the search query through `query`, and the result page index through `page_number`. Second, the product-selection component records the currently selected product using `selected_item`, together with its configurable attributes, including `option_color`, `option_size`, `option_material`, `option_style`, and the values of additional options stored in `option_extra_1_value` and `option_extra_2_value`. Third, the `item` field stores a compact product summary containing its ASIN, title, and price. The fields highlighted in red correspond to the state fields that the model is allowed to predict: `page_type`, `query`, `page_number`, `selected_item`, the four standard option fields, and the two additional option-value fields. In contrast, `option_extra_1_name`, `option_extra_2_name`, and `item` are maintained by the environment and are not direct prediction targets. At each step, the model predicts exactly one changed state field. For example, an unresolved product color can be updated from `{'option_color': null}` to `{'option_color': 'Black'}`, indicating a transition to the next state. Similarly, after sufficient product evidence has been gathered, the model may predict `{'page_type': 'done'}` to represent the completion of the shopping task.

A.6 Prompts

The prompts used by our LLM agents are shown in Figure 9, Figure 10, and Figure 11. Implemented as Python-style formatted strings, these templates dynamically populate semantic placeholders such as the task description, current state, available actions, and interaction history. Each prompt contains a `<think>` reasoning block followed by a structured output field. For ALFWorld, the agent predicts one state update involving the agent's location, container status, held item, or object status. For BabyAI, it predicts one update to the agent's position, facing direction, carried item, or the status of the front cell. The WebShop prompt contains two complementary policies: the Subgoal Reasoning Policy predicts exactly one changed state field from `query`, `page_number`, `selected_item`, `page_type`, or a product-option value, while the Action Reasoning Policy receives the predicted state delta and selects the single available action that produces it. In all cases, the structured output is constrained to one valid transition, enabling precise parsing and execution in the corresponding environment.

B Pseudo Code

Algorithm 1 outlines the complete training pipeline of CogRL. Compared to vanilla GRPO, our framework introduces key enhancements to address the core challenges of environment-aware LLM agent learning. Specifically, the dual advantage estimation over both task and environment rewards enables fine-grained credit assignment, which directly targets the problem of ungrounded reasoning by aligning policy updates with both task success and environmental dynamics. Building step-level groups by clustering transitions with the same reference states preserves the low-memory, critic-free properties of group-based RL, while introducing decomposed advantage signals that support more precise policy adjustments. Furthermore, the periodic supervised fine-tuning of the inverse dynamics model using a replay buffer ensures that the action generation policy remains anchored to accurate environmental transitions, mitigating the risk of hallucinated actions that violate environmental constraints. As such, CogRL retains the stable convergence characteristics of GRPO, while delivering a modular, data-efficient update pipeline that is uniquely suited for training long-horizon LLM agents in interactive environments.

Example for BabyAI's state space

```
State = {
            'agent_state': {
                  'pos': [9, 6],
                  'facing': 'north',
                  'item_carry': None
            },
            'front_cell': {
                  'type': None,
                  'status': 'null'
            },
            'visible_objects': {
                   'closed green door 1': {
                       'pos': [10, 6],
                       'relation_pos': {
                          'ahead': 0,
                          'side': 'east',
                          'side_steps': 1
                       }
                   },
                   'closed red door 1': {
                       'pos': [10, 1],
                       'relation_pos': {
                          'ahead': 5,
                          'side': 'east',
                          'side_steps': 1
                       }
                   }
               }
            }
```

Figure 7: Example for BabyAI's state space

C REASONING CASE STUDY

ALFWorld Figure 12 and 13 illustrate a reasoning case comparing CogRL with the baseline in the AlFworld environment.

BabyAI Figure 14 and 15 illustrate a reasoning case comparing CogRL with the baseline in the BabyAI environment.

Example for WebShop's state space

```
State = {
            'page_type': 'item',
            'query': 'wireless bluetooth headphones',
            'page_number': null,
            'selected_item': 'B09XYZ1234',
            'option_color': 'Black',
            'option_size': 'Standard',
            'option_material': null,
            'option_style': 'Over-Ear',
            'option_extra_1_name': 'connectivity',
            'option_extra_1_value': 'Bluetooth',
            'option_extra_2_name': 'pack',
            'option_extra_2_value': '1',
            'item': {
               'asin': 'B09XYZ1234',
               'title': 'Wireless Bluetooth Over-Ear Headphones',
               'price': 29.99
            }
        }
```

Figure 8: Example for WebShop's state space

Subgoal Reasoning Policy Prompt Template for ALFWorld

You are a goal-conditioned next-state proposer for the ALFWorld environment.
Your role is to predict the next environment state that moves the task closer to completion, given the current state and the task goal. The proposed next state must be reachable by exactly one valid action.

Task guide:

1. Locations may only change to entries in reachable_locations.
2. Closed containers must be opened before interaction.
3. The task target must be found and held in hand before placement or interaction.
4. Placement or interaction requires being at the target container or required tool.
The current task goal is: {task_goal}.
The locations you can reach are: {reachable_locations}.
You are at step {step_id}.
The history of the last two step is: {action_history}.
The discover items in visited locations are: {items_in_locations}.

The current state is: {current_state}.
Prediction the next state. You may propose a next state corresponding to exactly one of the following transition types:
1. Modify `current_position.location_name` to a reachable location.
2. Modify `current_position.status` to "open" or "close" when interacting with a container at the current location.
3. Modify `item_in_hand.item_name` by taking an object at the current location or putting down the currently held object.
4. Modify `item_in_hand.status` when interacting with an object using a required tool.
  - clean → requires being at a sinkbasin
  - cool → requires being at a fridge
  - heat → requires being at a microwave

Output format (STRICT):
1. Include a short reasoning within <thinking> </thinking> tags, briefly justify the predicted state based on the task goal and the current state.
2. Within <answer> </answer> tags, output ONLY a JSON object containing the single modified field. For example, {{"current_position":{{"location_name":"shelf 6","status":null}}}}
3. Do NOT include unchanged fields. Do NOT output anything outside these tags.

Action Reasoning Policy Prompt Template for ALFWorld

You are an agent in ALF World environment. Your task is to: {task}
Current state:

{current_state}

Available actions (you can only choose from these):

{available_actions}

Next state:

{next_state}

Based on the current state, available actions, and the next state, what action was taken to transition from the current state to the next state? You must choose one of the available actions.Directly output the chosen action and nothing else"""

Figure 9: Prompt templates used in ALFWorld.Top: Subgoal Reasoning Policy prompt.Bottom: Action Reasoning Policy prompt.

Subgoal Reasoning Policy Prompt Template for BabyAI

```
You are a goal-conditioned next-state proposer for the BabyAI environment.
Your role is to predict the next environment state that moves the task closer to completion, given the
current state and the task goal. The proposed next state must be reachable by exactly one available action.

Task guide:
1. Coordinate system: origin at top-left.
   north = up, south = down, west = left, east = right.
2. Available atomic actions: forward, left, right, pick up, drop, toggle.

The current task goal is: {task_goal}.
You are at step {step_id}. The history of the last two steps is: {action_history}.
The current state is: {current_state}.
Predict the next state. You may propose a next state corresponding to exactly one of the following
transition types:

1. Modify `agent_state.pos` to current_position
   - The agent may move forward by ONE cell only.
   - The movement direction MUST match `agent_state.facing`.
   - Position update MUST follow the coordinate system.
2. Modify `agent_state.facing`
   - Allowed facing directions are: `north`, `south`, `west`, `east`.
   - Facing changes by exactly a 90-degree rotation.
3. Modify `agent_state.item_carry`
   - The agent can carry AT MOST ONE object at a time.
   - You can change null to a non-null and non-null to null.
4. Modify `front_cell.status`
   - This transition is ONLY valid when `front_cell.type` contains "door".
   - Allowed status values are: "open"

Output format (STRICT):
1. Include a short reasoning within <thinking> </thinking> tags, briefly justify the predicted state based on
the task goal and the current state.
2. Within <answer> </answer> tags, output ONLY a JSON object containing the single modified field. for
example:
  <answer>
  {{"agent_state":{{"item_carry":"grey key"}}}}
  </answer>
3. Do NOT include unchanged fields. Do NOT output anything outside these tags.
```

Action Reasoning Policy Prompt Template for BabyAI

```
You are an inverse dynamics model for BabyAI environment.
Current state (JSON):
{current_state}

Next state (JSON):
{next_state}

Admissible actions:
- left: Turn 90 degrees counter-clockwise
- right: Turn 90 degrees clockwise
- forward: Move one cell forward
- pickup: Pick up the object in front
- drop: Drop the carried object
- toggle: Interact with object (open/close)

Choose an admissible action that explains this transition. Output only the action:
```

Figure 10: Prompt templates used in BabyAI. Top: Subgoal Reasoning Policy prompt. Bottom: Action Reasoning Policy prompt.

Subgoal Reasoning Policy Prompt Template for WebShop

```
You are a one-step state predictor in the WebShop environment.
Your task is to: {task_description}.
Prior to this step, you have already taken {step_count} step(s).
Below are the most recent {history_length} observations and corresponding actions: {action_history}
You are now at step {current_step}.
Current structured state (JSON): {current_state}
Current available actions are: [ {available_actions} ]

Goal: Predict exactly ONE changed state field for the next best step.

Rules:
1. Output exactly one top-level JSON field.
2. The key must be a real state field, not an action name.
3. Do NOT output an action.
4. Use current_state as the primary source of truth. Use history only as supporting context.
5. Valid state fields are:
   query, page_number, selected_item, page_type,
   option_color, option_size, option_material, option_style,
   option_extra_1_value, option_extra_2_value
6. If predicting page_type, the value must be one of:
   search, results, item, description, features, reviews, done
7. Predict page_number only if:
   - current page_type is results, and
   - click[Next >] or click[< Prev] is available, and
   - the new value is exactly current page number +1 or -1
8. Predict an option field only when exactly one option value changes.
9. On results pages, do not reject an item only because some attributes are still unknown.
10. On item pages:
   - If a required attribute is unresolved and a matching option is available:
   → you MUST change that option.
   - Else if no subpage (description/features/reviews) has been visited:
   → you MUST go to one of them.
   - After a subpage:
   - If the item does not satisfy the task:
     → go back (results or search).
   - If it satisfies the task:
     → predict {{"page_type": "done"}}.
   - Do NOT predict done without evidence.
11. Never output unsupported facts from titles or history.

Output format: <think>brief reason</think> <answer>{{"state_field": value}}</answer>
```

Action Reasoning Policy Prompt Template for WebShop

```
You are an inverse dynamics model for the WebShop environment.
Current state (JSON): {current_state}
Predicted state delta (JSON): {state_delta}
Available actions: [{available_actions}]

Goal: Choose the ONE action that causes the given state delta.

Rules:
1. Use state_delta as the primary signal.
2. Do NOT choose an action only because it matches words in the task or current state.
3. If state_delta changes:
   - query -> choose search[...]
   - page_number -> choose click[Next >] or click[< Prev]
   - selected_item -> choose click[asin]
   - page_type=description/features/reviews -> choose the matching click action
   - page_type=done -> choose click[buy now]
   - page_type=search -> choose click[back to search] or the legal back action
   - an option field -> choose the click action whose label matches the new option value
4. Return exactly one action from the available actions list.
5. Return nothing except the action text.
Action:
```

Figure 11: Prompt templates used in WebShop. Top: Subgoal Reasoning Policy prompt. Bottom: Action Reasoning Policy prompt.

Algorithm 1 CogRL: Environment-Aware Cognitive Reasoning for LLM Agents

Input: Subgoal Reasoning Policy π_θ^1; Action Reasoning Policy π_ϕ^2; Task goal g; Rollout horizon T; Number of trajectories N; Advantage weighting λ_{sim}; Discount factor γ; SFT interval k; SFT batch size B Buffer capacity C
Initialize: Inverse-dynamics buffer $\mathcal{B}_{\text{ID}} \leftarrow \emptyset$
for each training iteration **do**
  Initialize trajectory set $\mathcal{T} \leftarrow \emptyset$
  // Multi-step rollout phase
  for $i = 1$ **to** N **do**
    $s_0^{(i)} \leftarrow Env$.reset()
    Initialize trajectory $\tau^{(i)} \leftarrow [\,]$
    for $t = 0$ **to** $T - 1$ **do**
      Sample $\hat{s}_{t+1}^{(i)} \sim \pi_\theta^1(\cdot \mid s_t^{(i)}, g)$
      Sample $a_t^{(i)} \sim \pi_\phi^2(\cdot \mid s_t^{(i)}, \hat{s}_{t+1}^{(i)})$
      Execute $a_t^{(i)}$, observe $r_t^{(i)}$, $s_{t+1}^{(i)}$
      Store $(s_t^{(i)}, \hat{s}_{t+1}^{(i)}, a_t^{(i)}, r_t^{(i)})$ in $\tau^{(i)}$
      Store $(s_t^{(i)}, s_{t+1}^{(i)}, a_t^{(i)})$ in $\mathcal{B}_{\text{ID}}$
      if $|\mathcal{B}_{\text{ID}}| > C$ **then**
        Randomly remove one transition from $\mathcal{B}_{\text{ID}}$
      end if
    end for
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\tau^{(i)}\}$
  end for
  // Update Subgoal Reasoning Policy π_θ^1
  for each trajectory $\tau \in \mathcal{T}$ **do**
    Compute step-level task reward r^{task}
    Compute state-consistency reward r^{con}
  end for
  Build step-level groups $\mathcal{G}(s)$
  Estimate step-level task advantages A^{task} and Estimate state-consistency advantages A^{con} via Eq. equation 1
  Combine advantages A via Eq. equation 2
  Update SRP parameters θ via Eq. equation 3
  // Update Action Reasoning Policy π_ϕ^2
  if iteration mod $k = 0$ **then**
    Sample mini-batch $\mathcal{B} \subset \mathcal{D}$ of size B
    Update ARP parameters ϕ via Eq. equation 4
  end if
end for

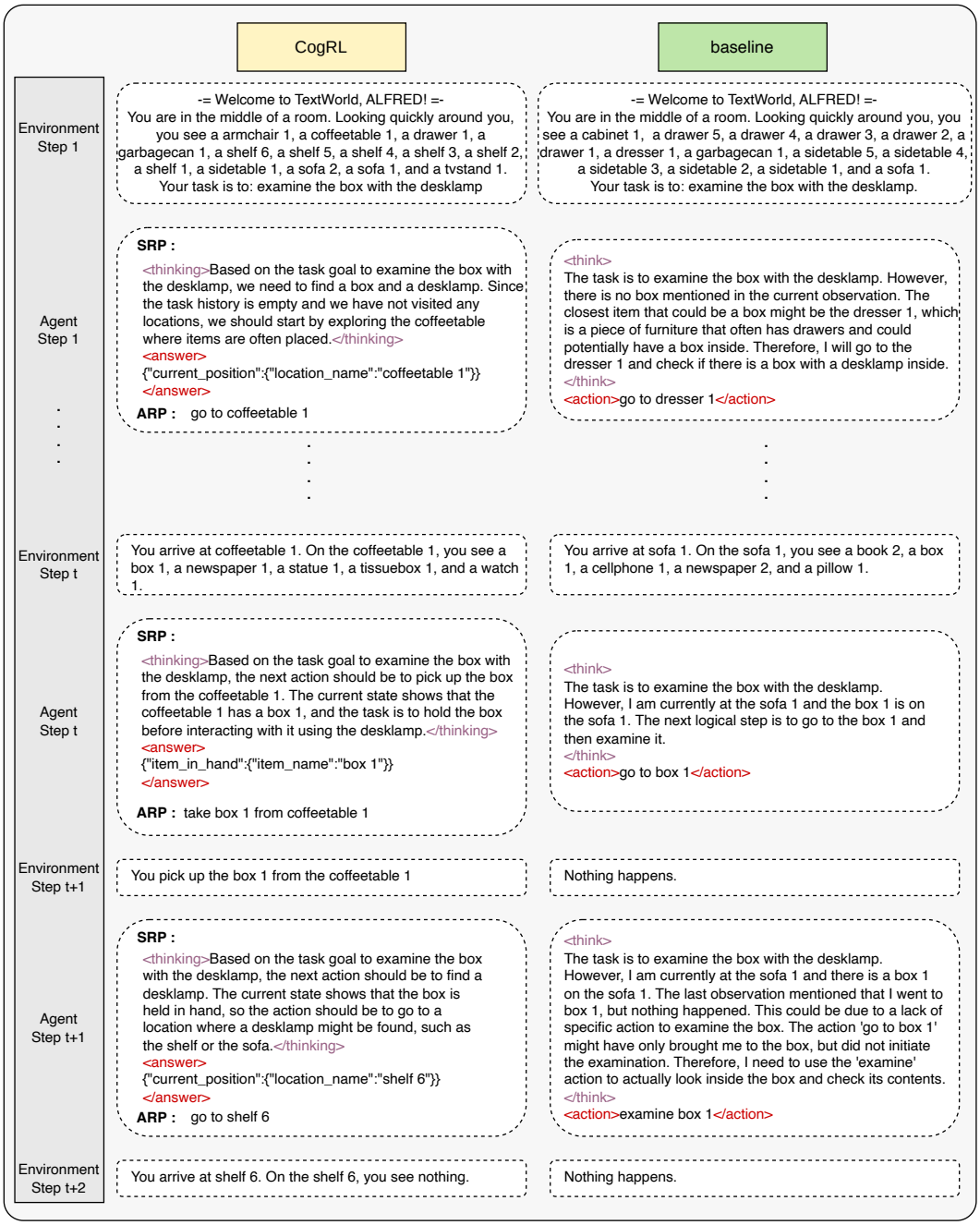


Figure 12: Reasoning case study of ALFworld-1

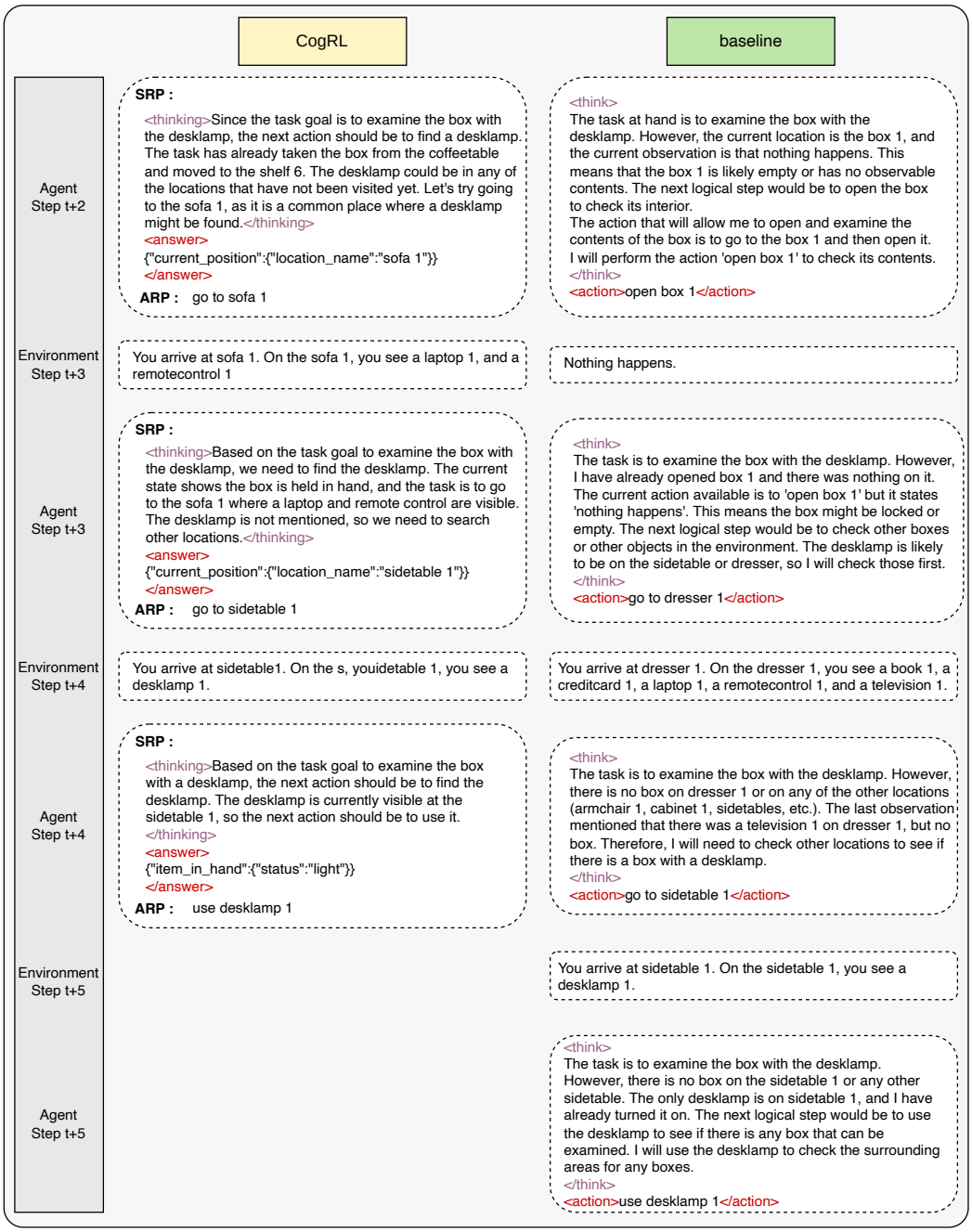


Figure 13: Reasoning case study of ALFworld-2

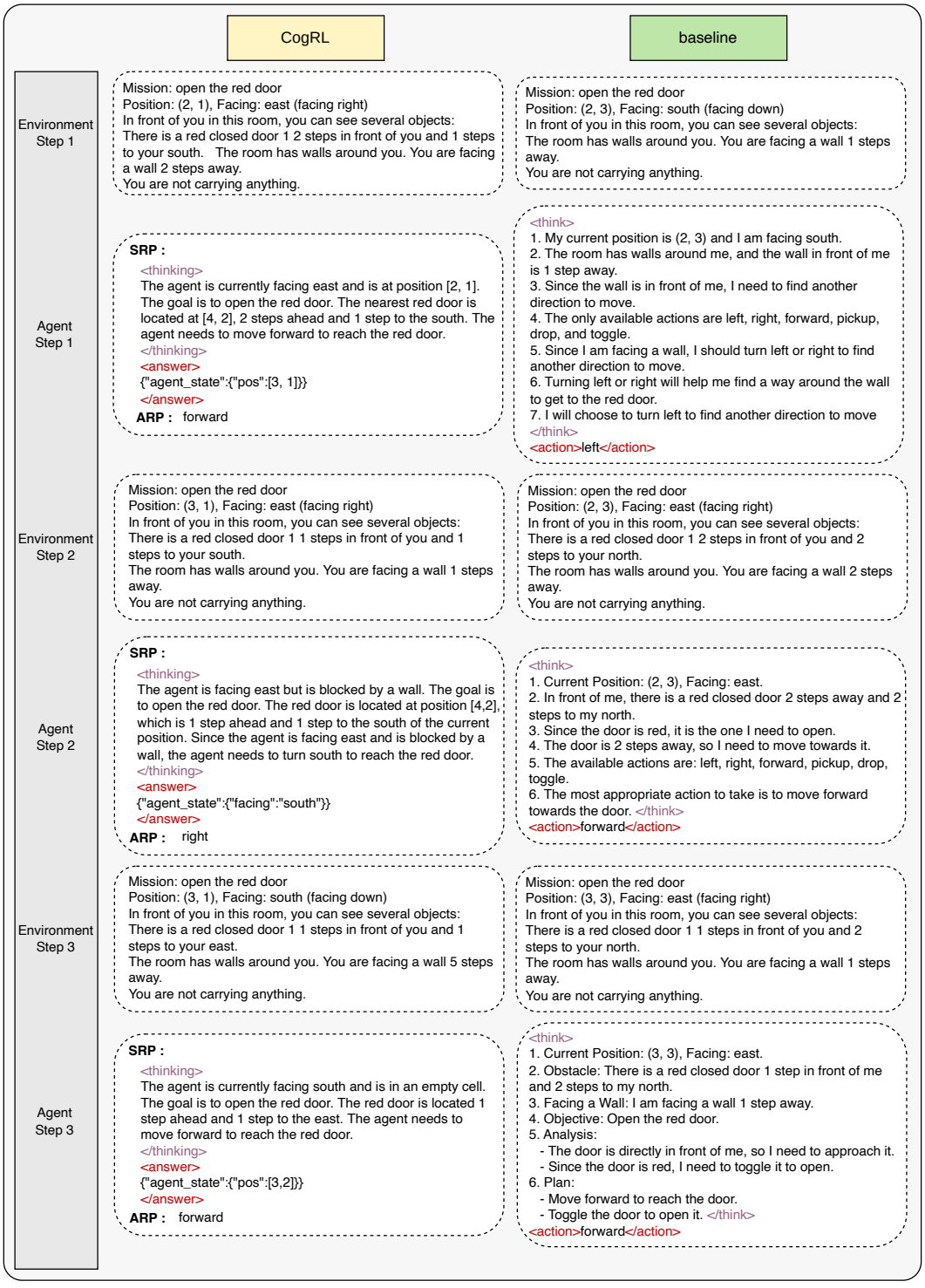


Figure 14: Reasoning case study of BabyAI-1

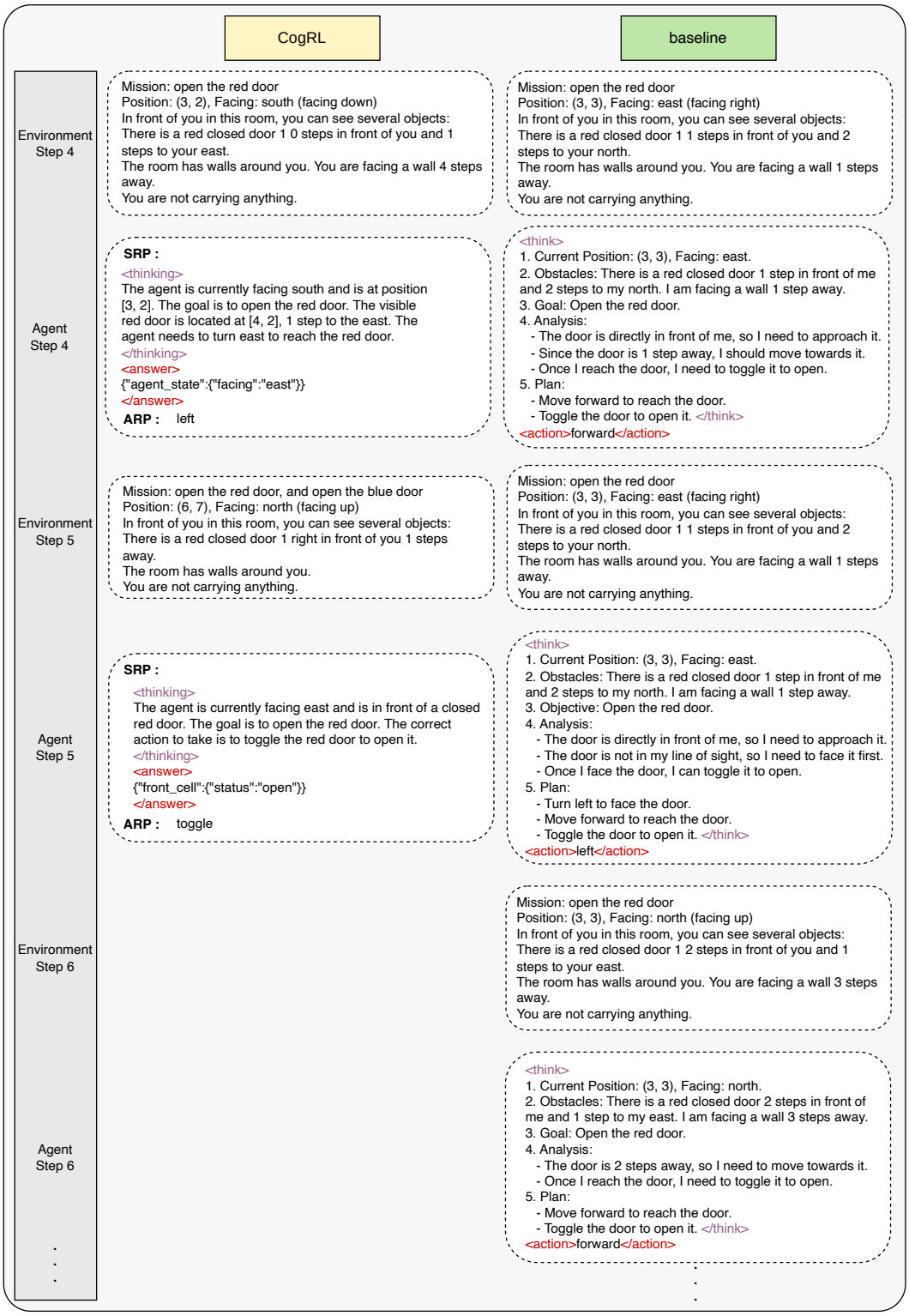


Figure 15: Reasoning case study of BabyAI-2