Enhancing Decision-Making of Large Language Models via Actor-Critic

Heng Dong [*1] **Kefei Duan** [*2] **Chongjie Zhang** [2]

Abstract

Large Language Models (LLMs) have achieved remarkable advancements in natural language processing tasks, yet they encounter challenges in complex decision-making scenarios that require long-term reasoning and alignment with high-level objectives. Existing methods either rely on short-term auto-regressive action generation or face limitations in accurately simulating rollouts and assessing outcomes, leading to sub-optimal decisions. This paper introduces a novel LLM-based Actor-Critic framework, termed LAC, that effectively improves LLM policies with long-term action evaluations in a principled and scalable way. Our approach addresses two key challenges: (1) extracting robust action evaluations by computing Q-values via token logits associated with positive/negative outcomes, enhanced by future trajectory rollouts and reasoning; and (2) enabling efficient policy improvement through a gradient-free mechanism. Experiments across diverse environments – including high-level decision-making (ALFWorld), low-level action spaces (BabyAI-Text), and large action spaces (WebShop) – demonstrate the framework's generality and superiority over state-of-the-art methods. Notably, our approach achieves competitive performance using 7B/8B parameter LLMs, even outperforming baseline methods employing GPT-4 in complex tasks. These results underscore the potential of integrating structured policy optimization with LLMs' intrinsic knowledge to advance decision-making capabilities in multi-step environments.

[*]Equal contribution [1]IIIS, Tsinghua University [2]Washington University in St. Louis. Correspondence to: Heng Dong <drd-hxi@gmail.com>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

1. Introduction

Large Language Models (LLMs) (Touvron et al., 2023; Jiang et al., 2023; Team et al., 2024) have demonstrated impressive capabilities across various natural language processing tasks, including text generation (Li et al., 2024a), question answering (Li et al., 2024b), and summarization (Jin et al., 2024). The successful application of LLMs in these areas, coupled with their extensive internal knowledge, has generated significant interest in leveraging LLMs to tackle complex decision-making problems, particularly in data-scarce environments.

Early works (ichter et al., 2023; Huang et al., 2022a) have employed LLMs as policies, generating actions in an auto-regressive way that directly utilizes the models' prior knowledge for decision-making. While these methods are simple and effective for short-term action generation, they often lack the capacity for long-term planning. Although reasoning techniques such as Chain-of-Thought have been introduced to enhance the reasoning capabilities of LLMs (Yao et al., 2023; Shinn et al., 2024) and improve action selection, the ability to engage in comprehensive long-term planning remains underdeveloped. Consequently, decisions may appear locally optimal but fail to meet the overall objectives in more complex, multi-step environments.

Other approaches (Hao et al., 2023; Liu et al., 2023; Fu et al., 2024; Brooks et al., 2024) incorporate planning, either through interaction with real environments or by leveraging the LLMs' internal imaginative capabilities, followed by action evaluation based on the planning outcomes. These evaluations are then used to select the actions that yield the best results. However, such methods heavily depend on the model's accuracy in performing rollouts and evaluating outcomes, leading to sub-optimal action selection when the model's rollouts diverge from reality or when action evaluations are inaccurate.

The fundamental limitation of these two lines of research lies in the decoupling of LLM's prior policy and action evaluations. They either (1) rigidly follow LLM priors without sufficient planning or (2) over-rely on potentially flawed simulated rollouts, both fail to reconcile the LLM's inherent knowledge with error-corrected planning insights.

To address the above problems, we propose a novel LLM-

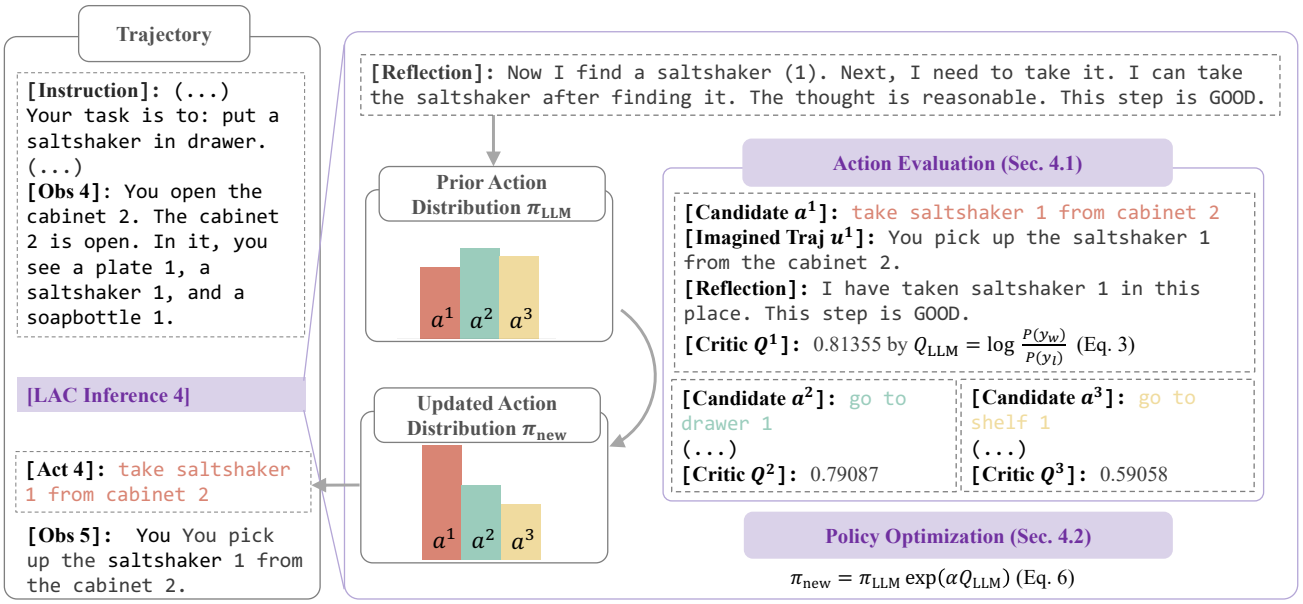

Figure 1. Framework of our LAC. At each time step, LAC optimizes the policy via two steps: (1) the critic Q_{LLM} evaluates each candidate action sampled from the policy π_{LLM}; (2) the policy π_{LLM} is optimized according to the action evaluations using a gradient-free method.

based Actor-Critic (LAC) approach that leverages action evaluations to optimize the LLM's prior policy under constraints, facilitating the effective integration of decision-making information inherent in the LLM's prior knowledge with insights derived from robust action evaluations that incorporate long-term planning. While there have been attempts to simultaneously utilize both types of information (the LLM's prior knowledge and action evaluations) for decision-making, these efforts have typically resulted in simplistic combinations (Zhang et al., 2023).

Implementing this idea involves two major challenges. First, how to extract action evaluation information from the LLM. A simple approach is to directly prompt the LLM to output evaluation values that indicate the quality of action; however, this direct prompting method can be unstable and easily influenced by the instructions and examples provided to LLMs. To address this issue, we use the logits of specific tokens to calculate Q values (action values). Specifically, we employ tokens that convey positive or negative meanings as the LLM's internal beliefs about the action's quality or its likelihood of successfully completing the task. We derived a simple yet effective formula to use these logits for Q value computation. Additionally, to allow long-term planning ability and enhance the accuracy of this calculation, we perform rollouts for each candidate action to predict future trajectories, conduct a brief reasoning process based on these trajectories, and then collect positive/negative token logits to compute Q values.

The second challenge is how to utilize these action evaluation insights to optimize the policy. A straightforward method might involve direct gradient-based actor-critic updates; however, this approach is inefficient for LLMs due to high computational demands, leading to increased decision latency. To solve this challenge, we formulate the policy improvement problem as a KL-divergence optimization problem and derive a closed-form solution, allowing us to optimize the policy in a gradient-free manner and improve the accuracy of decision-making.

Empirically, we demonstrate the effectiveness and generality of LAC across diverse environments, including high-level decision-making tasks (ALFWorld, (Shridhar et al., 2021)), low-level action space (BabyAI-Text, (Carta et al., 2023b)) and potentially infinite action space (WebShop, (Yao et al., 2022)). The results show that our approach consistently outperforms state-of-the-art methods, *e.g.*, RAP (Hao et al., 2023), LATS (Zhou et al., 2024a). Notably, in several challenging tasks, LAC using 7B/8B LLMs significantly surpasses ReAct (Yao et al., 2023) with GPT-4 (Achiam et al., 2023).

Our contributions are twofold: (1) We propose a novel Q-function estimation approach to extract action evaluation information from LLMs that leverages LLMs' internal belief about success or failure of the current task; (2) We formulate the policy improvement problem as a KL-divergence constrained optimization and derive a closed-form solution, allowing us to optimize the policy in a gradient-free manner using the action evaluation.

2. Related Work

Large Language Models for Sequential Decision-Making Sequential decision-making is a key ability of intelligent agents, involving generating actions to achieve goals (Barto et al., 1989; Littman, 1996; McCarthy et al., 1963; Bylander, 1994). Recently, LLM-based agents have gained popularity in decision-making across various fields, requiring only instructions or few-shot examples to adapt to new tasks (Huang et al., 2022b; Singh et al., 2023; Ding et al., 2023) due to pre-training on large datasets. Previous work primarily categorizes LLMs into two roles: policies, which generate actions from trajectories, and critics, which evaluate actions based on trajectories and actions. Research falls into two categories: the first uses LLM-generated actions directly from previous trajectories (ichter et al., 2023; Huang et al., 2022b; Yao et al., 2023; Huang et al., 2022a; Shinn et al., 2024). However, the auto-regressive nature of LLMs limits their long-term planning capabilities, making them struggle with complex tasks (Huang & Chang, 2022; Mialon et al., 2023). The second category employs another LLM to evaluate actions by simulating outcomes, choosing the best one (Hao et al., 2023; Liu et al., 2023; Fu et al., 2024; Koh et al., 2024). While these methods enable long-term planning, they heavily depend on evaluation accuracy, which can lead to sub-optimal solutions.

To address these issues, we propose optimizing LLM's prior policy using action evaluations under constraints, enhancing long-term planning while mitigating evaluation inaccuracies. Previous attempts to combine LLM's prior policy and action evaluation (Zhang et al., 2023) have been simplistic and lacked theoretical guarantees, resulting in unsatisfactory performance.

Large Language Models with Reinforcement Learning Classical sequential decision-making methods, such as Reinforcement Learning (RL), have been widely adopted in embodied environments (Schulman et al., 2017; Fujimoto et al., 2018; Huang et al., 2020; Dong et al., 2022). However, these RL-based methods are typically sample-inefficient and require lots of samples for training. On the other hand, LLMs that contain rich prior knowledge about the world may alleviate this burden. To combine RL and LLM, one straightforward way is to use LLMs as base models and add policy/value heads on top of LLMs (Carta et al., 2023a; Tan et al., 2024). Then classical RL methods like PPO (Schulman et al., 2017) can be used for training (Szot et al., 2023; Zhou et al., 2024b). However, these methods still require lots of training samples of the same tasks, which reduces the benefits of using LLM to some extent and contradicts our settings. There are also other paradigms for combining. RLEM (Zhang et al., 2024) adopts Q-learning (Watkins & Dayan, 1992) and an experience memory to update policies, but it may get stuck in the tasks with extremely sparse rewards like ALFWorld and BabyAI-Text. Retroformer (Yao et al.) trains a smaller LLM with PPO to generate suitable prompts for a larger LLM for a specific task, while our method only needs a small model. ICPI (Brooks et al., 2024) uses LLMs to implement policy iteration by predicting future trajectories and accumulating future rewards, which may also struggle with sparse reward settings. We have compared it empirically in Section 5.

3. Preliminary & Previous Work

In this section, we describe the task setting and previous LLM decision-making methods for better understanding.

Task setup. Consider a general setup of an agent interacting with an environment for achieving a given goal g, *e.g.*, $g =$"put a clean egg in microwave" (from ALFWorld) or $g =$"pick up the green ball" (from BabyAI-Text). At time step t, the agent receives a natural language described observation $o_t \in \mathcal{O}$ from the environment. The agent then takes an action $a_t \in \mathcal{A}$ sampled from policy $\pi(a|g, h_t)$, where $h_t := (o_1, a_1, o_2, a_2 \cdots, o_t)$ is the history to the agent. During execution, there is no immediate reward and only at the end of each episode, the environment will give a signal to evaluate the completion of the task. For each testing task, the agent can only try once and cannot conduct improvements through repeated trials.

Methods that directly use LLM's prior as a policy. To solve the above tasks with LLMs, one simple method is to directly use LLM's prior as a policy: $a_t \leftarrow \arg\max_a \pi_{\text{LLM}}(a|g, h_t)$, which can be implemented by simply injecting instructions or few-show examples to the prompt as suggested in Yao et al. (2023). Despite its simplicity, the policy π_{LLM} generates actions solely relying on its auto-regression ability and it does not conduct long-term planning explicitly, which is typically necessary for sequential decision-making tasks. Additionally, this issue will be exacerbated when using lightweight models like CodeLlama-7B (Roziere et al., 2023) and Mistral-7B (Jiang et al., 2023). This problem is verified in Section 5.

Methods that incorporate planning and action evaluations. To handle the issue of lack of long-term planning, another line of research incorporates planning and action evaluations into decision-making (Hao et al., 2023; Liu et al., 2023; Fu et al., 2024). The basic idea is to first sample several candidate actions from policy $\{a_t^1, a_t^2, \cdots, a_t^n\} \sim \pi_{\text{LLM}}(\cdot|g, h_t)$, then evaluate each candidate action by other LLMs and finally select the action with the highest evaluation value. The evaluation procedure is the key to these methods, and many approaches can be adopted. For example, directly ask an LLM to evaluate the action candidate (Fu et al., 2024), or predict the future trajectory u_t of each action candidate by regarding an LLM as a forward world model

Algorithm 1 LAC: LLM-based Actor-Critic algorithm.

Input: current task goal g, history h_t, actor π_{LLM}, forward model f_{LLM}, value-based critic $\mathcal{Q}_{\text{LLM}}$, hyperparameter α, candidate action size n.
Output: selected action a_t^*
$\{a_t^i\}_{i=1}^n \sim \pi_{\text{LLM}}(\cdot|g, h_t)$; ▷ candidate actions
for $i = 1$ **to** n **do**
  $u_t^i \leftarrow f_{\text{LLM}}(g, h_t, a_t^i)$; ▷ predict future trajectory
  $\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i, u_t^i) \leftarrow \log \frac{P(y_w|g,h_t,a_t^i,u_t^i)}{P(y_l|g,h_t,a_t^i,u_t^i)}$; ▷ action evaluation (Sec. 4.1)
end for
$\pi_{\text{new}}(a_t^i|g, h_t) \leftarrow \pi_{\text{LLM}}(a_t^i|g, h_t) \exp(\alpha \mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i, u_t^i))$; ▷ policy optimization (Sec. 4.2)
$a_t^* \leftarrow \arg\max_{a_t^i} \pi_{\text{new}}(a_t^i|g, h_t)$

f_{LLM} and use the future outcome as evaluations, or use tree-search methods like Monte Carlo Tree Search (MCTS) (Kocsis & Szepesvári, 2006; Coulom, 2006) to expand each action candidate (Hao et al., 2023). Despite this progress, these approaches heavily depend on the model's accuracy in performing rollouts and evaluating outcomes. When rollouts diverge from reality and evaluations are inaccurate, which could be common when using lightweight LLMs, the action selection could be sub-optimal.

4. Method

In this section, we introduce our novel LLM-based Actor-Critic (LAC) algorithm, which effectively integrates the generative capabilities of LLMs (as the *actor*) with their evaluative reasoning capabilities (as the *critic*) in a principled and scalable manner. The actor, represented by the LLM-based prior policy (π_{LLM}), generates potential actions, while the critic ($\mathcal{Q}_{\text{LLM}}$) evaluates these actions by incorporating long-term reasoning. This synergy enables effective policy improvement through an innovative gradient-free optimization method.

We first explain how to leverage LLMs to extract meaningful action evaluation information (Section 4.1). Building on this, we propose a gradient-free policy optimization method that efficiently refines the policy based on the critic's feedback (Section 4.2). The overall workflow of the algorithm is summarized in Algorithm 1.

4.1. Critic Development for Long-term Action Evaluation

We propose a novel method for constructing the critic $\mathcal{Q}_{\text{LLM}}$, designed to provide numerical evaluations of candidate actions sampled from the policy $\pi_{\text{LLM}}(\cdot|g, h_t)$. This approach targets goal-based decision-making problems characterized by sparse rewards and binary outcomes, where the agent receives a reward only upon achieving the goal. By linking $\mathcal{Q}_{\text{LLM}}$ to the agent's success probability of task completion, we enable more effective guidance toward maximizing expected returns. Furthermore, we demonstrate how this evaluation can be derived directly from token logits associated with positive and negative outcomes.

4.1.1. Connect Critic to Success Probability

Let $\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i)$ be the value-based evaluation of each candidate action a_t^i given the task goal g and history h_t. We consider scenarios with sparse rewards, which are only provided at the end of each episode. Considering the binary outcomes, we hope $\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i)$ could reflect the possibility of achieving success effectively. We employ a logistics function (Jordan et al., 1995) to relate $\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i)$ to the success probability.

Let $P(y_w|g, h_t, a_t^i) \in [0, 1]$ denote the probability of successfully completing the task goal g after executing action a_t^i, where y_w represents a success signal at the end of the episode. Similarly, let $P(y_l|g, h_t, a_t^i)$ represent the failure probability. We use the following formulation to connect $P(y_w|g, h_t, a_t^i)$ with $\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i)$:

$$P(y_w|g, h_t, a_t^i) = \frac{1}{1 + \exp\left(-\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i)\right)}. \quad (1)$$

With this formulation, $\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i)$ is positively correlated with the success probability $P(y_w|g, h_t, a_t^i)$. Higher $\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i)$ values map to a greater likelihood of success, allowing the critic to guide the policy toward actions that maximize long-term returns. In this way, maximizing the Q-function corresponds to maximizing the success probability for the current task.

While other formulations could be used, we found that Equation (1) is both simple and effective for a wide range of tasks. For a comparison of alternative formulations, refer to Appendix A.2 and Appendix A.3.

4.1.2. Estimate Critic with LLMs

To estimate $\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i)$ using LLMs, we perform an equivalent transformation on Equation (1):

$$\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i) = \log \frac{P(y_w|g, h_t, a_t^i)}{P(y_l|g, h_t, a_t^i)}, \quad (2)$$

which uses the equation $P(y_w) + P(y_l) = 1$. With Equation (2), we can use the LLM to obtain $\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i)$ via first estimating $P(y_{\{w,l\}}|g, h_t, a_t^i)$. The basic idea is to prompt the LLM to predict the outcomes given the current trajectory (g, h_t) and action a_t^i. Specifically, we use special paired tokens with positive/negative meanings, *i.e.*, "GOOD"/"BAD" or "SUCCESS"/"FAILURE", to indicate success/failure outcomes. The corresponding generated

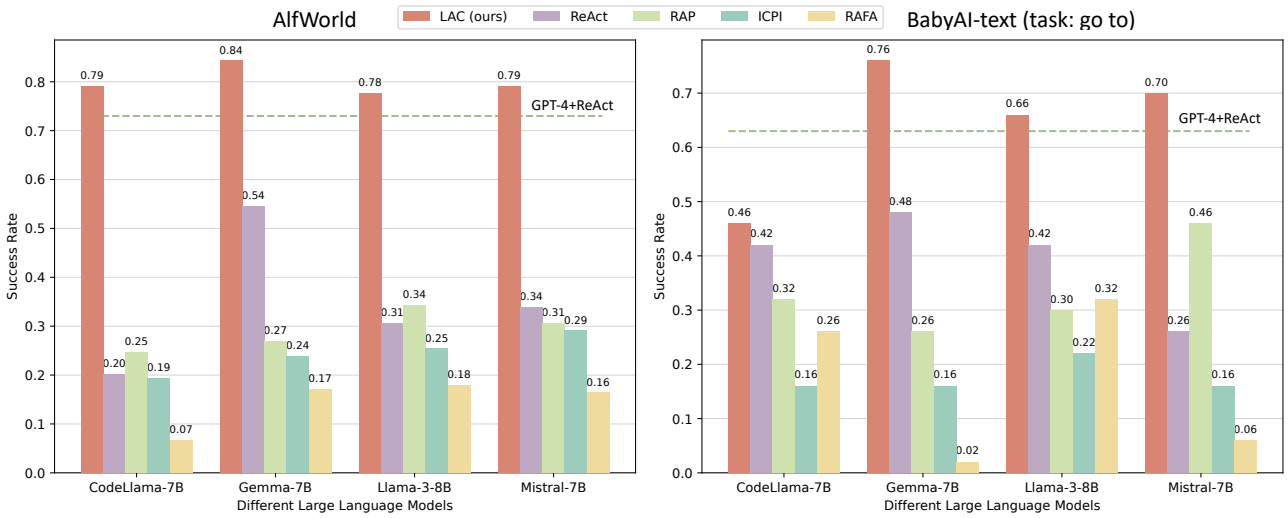

Figure 2. Performance of our LAC compared with various baselines in benchmarks ALFWorld and BabyAI-Text.

probabilities of LLMs for those special tokens reflect LLMs' internal belief in success/failure after taking action a_t^i. We let the generated probabilities of "GOOD" and "BAD" represent $P(y_w|g, h_t, a_t^i)$ and $P(y_l|g, h_t, a_t^i)$ respectively. Finally, using Equation (2), we can calculate $\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i)$ for action a_t^i. Though our action evaluation is designed for binary outcomes, we empirically show that this formulation is also effective in continuous-reward settings in Section 5.

To improve the accuracy of $\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i)$, we introduce future trajectory rollouts using a forward world model f_{LLM}, which can be implemented by prompting LLMs, *e.g.*, adding few-shot examples, or by fine-tuning on these examples. For each candidate action a_t^i, we rollout several future steps to predict the resulting trajectory $u_t^i \sim f_{\text{LLM}}(g, h_t, a_t^i)$. By considering the future trajectory u_t^i, we obtain more informed estimates of the success and failure probabilities $P(y_{\{w,l\}}|g, h_t, a_t^i, u_t^i)$. This approach accounts for the delayed consequences of actions and ensures that $\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i, u_t^i)$ reflects the long-term value of each action:

$$\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i, u_t^i) = \log \frac{P(y_w|g, h_t, a_t^i, u_t^i)}{P(y_l|g, h_t, a_t^i, u_t^i)}. \quad (3)$$

Trajectory rollouts are especially important in tasks where the outcomes of actions may unfold over several steps. By simulating the future impact of actions, the critic provides a more accurate assessment, guiding the policy towards actions that maximize the probability of long-term success.

Empirically, we also found that contextual reflections on previous actions can be helpful to improve performance. Specifically, given the task goal g and history h_t, which may contain the predicted future trajectory, we prompt LLMs to generate a short reflection such as "I have found object-X. This step is GOOD" or "I should take object-X instead of object-Y first. This step is BAD." These reflections provide judgments about whether and why the previous actions were appropriate. For more examples of these reflections, please refer to Table 27 and Table 28 of Appendix B.

These reflections serve as simple reasons, akin to a Chain-of-Thought (CoT) (Wei et al., 2022; Kojima et al., 2022; Prystawski et al., 2024), allowing the policy to sample better candidate actions by avoiding past mistakes. These reflections also enable the critic to evaluate candidate actions with greater accuracy, ultimately enhancing decision-making performance. For the difference comparison between our reflection and CoT, please refer to Appendix A.4.

4.2. Gradient-free Policy Optimization

In this subsection, we derive a gradient-free policy optimization method using the above estimated value-based evaluation. To effectively improve the LLM's prior policy, we propose to use the following KL-constrained policy optimization problem to maximize the expected value function:

$$\max_{\pi} \mathbb{E}_{a_t^i \sim \pi(\cdot|g,h_t), u_t^i \sim f_{\text{LLM}}(g,h_t,a_t^i)} \left[\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i, u_t^i)\right] - \frac{1}{\alpha} \mathbb{D}_{KL}\left[\pi(a_t^i|g, h_t) \| \pi_{\text{LLM}}(a_t^i|g, h_t)\right], \quad (4)$$

where α is a hyperparameter controlling the deviation from the original policy π_{LLM}. The KL-divergence term prevents the new policy π_{new} from deviating too far from the original policy, balancing the policy's prior knowledge and the critic's guidance.

Following prior work (Rafailov et al., 2024; Go et al., 2023;

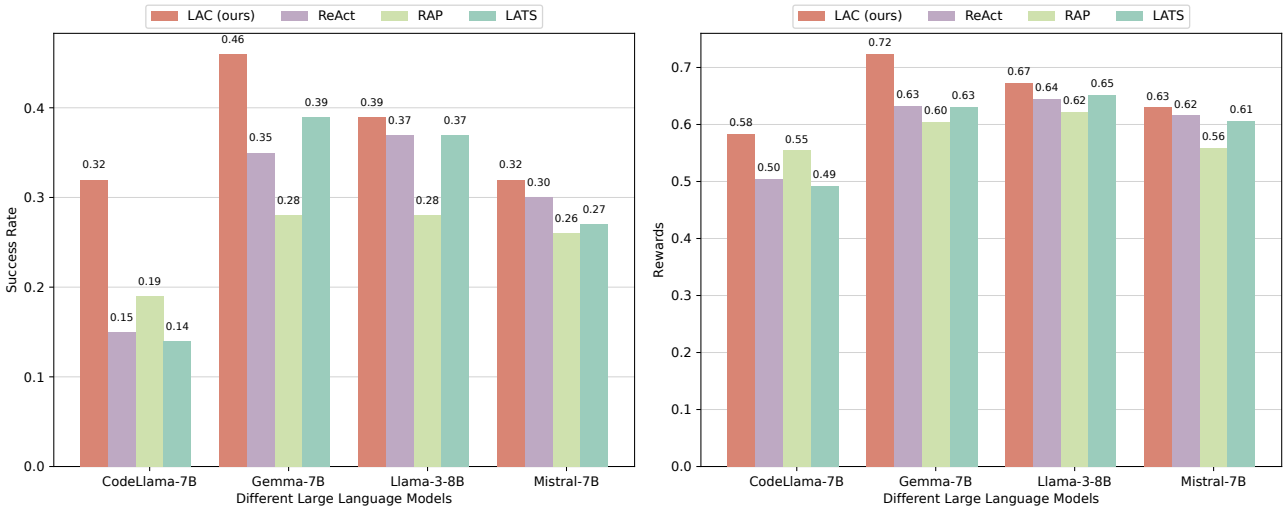

Figure 3. Performance of LAC in benchmark WebShop

Peng et al., 2019; Jain et al., 2013; Peters & Schaal, 2007), we can show that the optimal solution to the KL-constrained maximization objective in Equation (4) takes the following form:

$$\pi_{\text{new}}(a_t^i|g,h_t) = \frac{1}{Z(g,h_t)}\pi_{\text{LLM}}(a_t^i|g,h_t) \cdot \exp\left(\alpha \mathcal{Q}_{\text{LLM}}(g,h_t,a_t^i,u_t^i)\right), \quad (5)$$

where $Z(g,h_t) = \sum_{a_t^i} \pi_{\text{LLM}}(a_t^i|g,h_t)\exp(\alpha \mathcal{Q}_{\text{LLM}}(g,h_t,a_t^i,u_t^i))$ is the partition function. Please refer to Appendix B.1 for a complete derivation. As the partition function does not depend on action a_t^i, we can ignore it in practice:

$$\pi_{\text{new}}(a_t^i|g,h_t) \propto \pi_{\text{LLM}}(a_t^i|g,h_t)\exp\left(\alpha \mathcal{Q}_{\text{LLM}}(g,h_t,a_t^i,u_t^i)\right). \quad (6)$$

We simply take the action with maximum proportion $a_t \leftarrow \arg\max_{a_t^i} \pi_{\text{new}}(a_t^i|g,h_t)$. It is worth mentioning that if we let $\alpha = 0$ in Equation (6), we recover the methods that directly use LLM's prior as a policy, and if we let $\alpha \to +\infty$, we recover the methods that incorporate planning and action evaluations.

There are two key advantages of using Equation (6). Firstly, it updates the action distribution of policy π_{LLM} in the direction suggested by critic $\mathcal{Q}_{\text{LLM}}$ in a gradient-free way, which achieves policy improvement with much lower computation burden compared to gradient-based methods, especially when the actor is realized by LLMs. Secondly, the action distribution of the new policy π_{new} is a balanced integration of the policy's prior based on past information and the critic's posterior based on predicted future information.

5. Experiments

In this section, we benchmark our method LAC on three benchmarks that cover high-level action space (ALFWorld (Shridhar et al., 2021)), low-level action space (BabyAI-Text (Chevalier-Boisvert et al., 2018)) and potentially infinite action space (WebShop (Yao et al., 2022)). We evaluate the effectiveness of LAC by answering the following questions: (1) Can LAC outperform other decision-making with LLM-based methods? (Section 5.2) (2) How does each component of LAC contribute to its performance? (Section 5.3) (3) How do different large language models influence performance? (Section 5.2 and Section 5.3) (4) Is our method computationally consuming? (Section 5.4) (5) Why is LAC effective? (Section 5.5). The code of LAC is publicly available on GitHub[1] and website[2].

5.1. Experiment Setup

We compare our method with various decision-making with LLMs baselines. Here we briefly introduce these methods, and for more details, please refer to Appendix C.2.

Various baselines: (1) ReAct (Yao et al., 2023) combines reasoning and acting in the interaction with the environment and leverages the reasoning capabilities of LLMs to increase the probability of the LLM acting correctly as a policy. (2) RAP (Hao et al., 2023) utilizes LLMs as policy and world models and adopts tree-search planning methods to evaluate each possible action candidate. (3) ICPI (Brooks et al., 2024) implements policy iteration using LLMs by predicting future trajectories and selecting the action with

[1] https://github.com/drdh/LAC
[2] https://sites.google.com/view/lang-ac

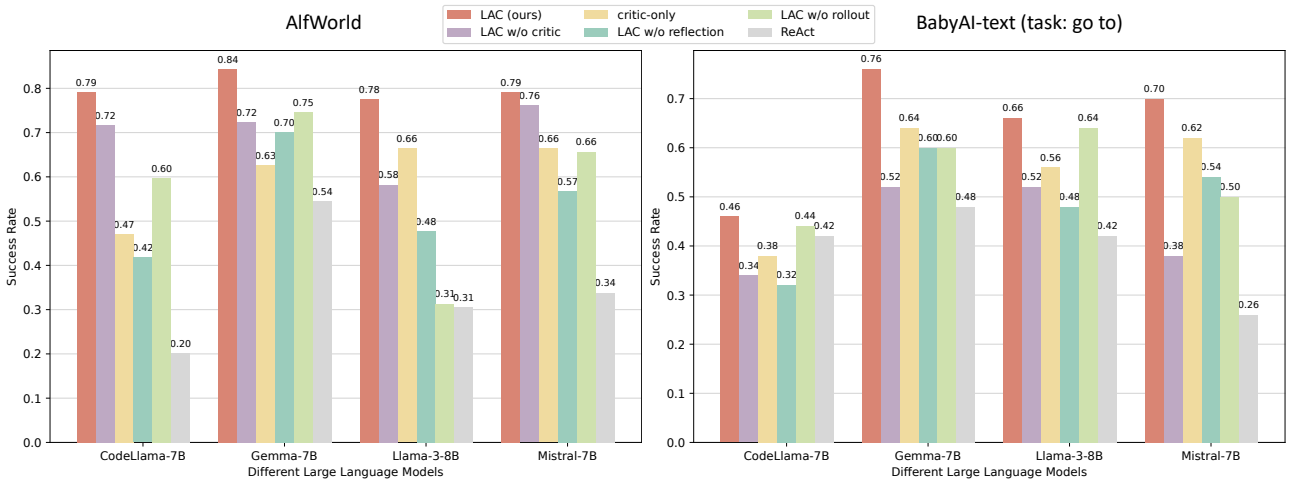

Figure 4. Ablation studies in benchmarks ALFWorld and BabyAI-Text.

the highest predicted cumulative rewards. (4) RAFA (Liu et al., 2023) evaluates each action candidate by tree-search and selects the action that may complete the most sub-goals. (5) LATS (Zhou et al., 2024a) combines the reasoning, acting, and planning capabilities of LLMs with MCTS (Kocsis & Szepesvári, 2006) and external feedback mechanisms to enhance decision-making across various domains, achieving competitive results in web navigation.

We evaluate LAC on three decision-making benchmarks with sparse rewards and distinct types of action space.

High-level actions: ALFWorld (Shridhar et al., 2021) is a widely used text-based household environment with 134 different evaluation tasks, which require the agent to achieve a goal through a sequence of high-level actions, *e.g.* "go to place-X", "take object-Y from place-X", *etc*. The agent gets a reward of 1 if it achieves the goal, and 0 otherwise. The main challenge of this benchmark is to locate the target object and fulfill household work with commonsense knowledge of LLMs. Following ReAct, we evaluate all 134 unseen evaluation games in a task-specific setup.

Low-level actions: BabyAI-Text (Carta et al., 2023b) is a Grid World environment that extended from the BabyAI platform (Chevalier-Boisvert et al., 2018), in which the agent and objects are placed in a room of 8×8 tiles. The agent has 6 primitive actions: turn left, turn right, go forward, pick up, drop, toggle, to solve a task described in natural language (*e.g.* "Pick up the red box"). Similar to ALFWorld, this benchmark also has binary rewards of 1 or 0. These tasks could be difficult because agents have to make a long-term plan, avoid obstacles, and find a short path to target objects based on partial observations that are described in natural language.

Potentially infinite actions: WebShop (Yao et al., 2022) requires an agent to purchase a product based on instructions (e.g. "I need a long clip-in hair extension which is natural looking, and price lower than 20.00 dollars") through web interactions (e.g. search "long clip-in hair extension", click buttons such as "[item ID]" or "back to search"). Within this context, the "search" and "click" actions can indeed lead to an unbounded set of potential actions, as the agent can continuously refine its queries and selections based on dynamic web results. Different from ALFWorld and BabyAI-Text, the final reward in this benchmark is a continuous value between 0 and 1, depending on the degree to which the final purchased product meets the requirements.

To show the stability of LAC, we adopt four open-source large language models from different organizations: CodeLlama-7B (Roziere et al., 2023), Mistral-7B (Jiang et al., 2023), Gemma-7B (Team et al., 2024), and Llama-3-8B (Meta, 2024a).

5.2. Main Performance

We report the results of our method LAC compared with other baselines in Figure 2 (ALFWorld and BabyAI-Text), Figure 3 (WebShop) and Figure 6 (more BabyAI-Text tasks). For all experiments, we set the temperature of LLMs to 0, hence the generation is deterministic. For this reason, there is no error bar in the figure.

LAC outperforms all other baselines in both ALFWorld and BabyAI-Text, and is even better than GPT-4+ReAct in most settings, which validates our method's effectiveness and stability. In WebShop, LAC consistently outperforms other baselines, in terms of both accumulated reward and success rate across various base models. This further demonstrates

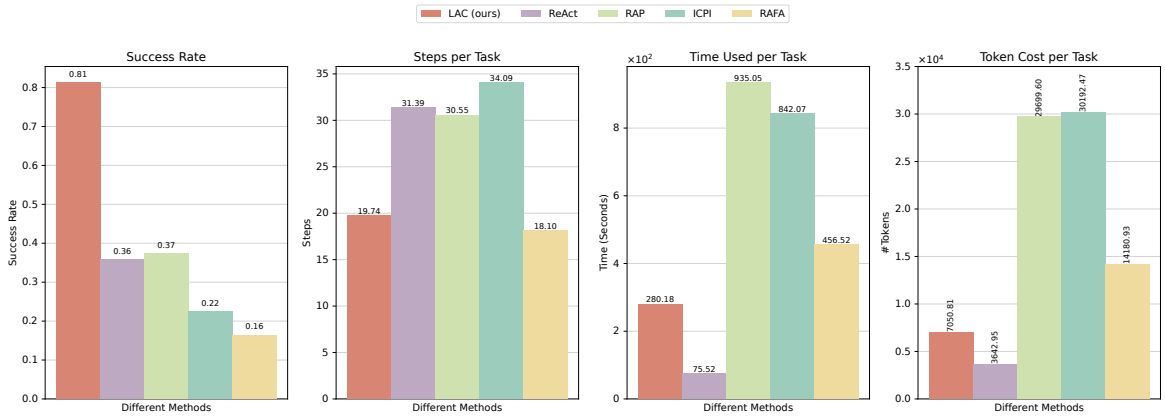

Figure 5. Computational cost analysis of LAC and baselines.

the robustness of our method in handling more complex and open-ended action spaces.

LAC's superior performance stems from its effective integration of the decision-making information contained in the LLM prior with the decision-making insights derived from action evaluations that incorporate long-term planning. For better understanding, we have also provided illustrative examples for ALFWorld and BabyAI-Text in Figure 10 and Figure 11 respectively. In summary, the policy and critic alone may make mistakes at different time steps, our LAC can select the correct action through their integration.

Regarding the performance of LAC with different base models, we highlight two key findings: (1) Our method is general and can be adapted to various base models, and (2) stronger base models, such as Gemma-7B, demonstrate higher performance when integrated with our approach. However, due to the incomplete public availability of training details for these base models, further in-depth analysis will require additional investigation.

Here we only compare LAC with LLM-based methods, for more results regarding other baselines, *e.g.*, finetuning the policy, LLM-based RL variants, and decision-transformer (Chen et al., 2021), please refer to Appendix A.5.

5.3. Ablation Studies

To investigate the contributions of each component of LAC, we conduct elaborate ablation studies. There are several main components that characterize our method: (1) policy optimization step using action evaluations provided by critic; (2) action evaluations by extracting internal information of LLMs. Therefore, to show the contribution of each component, we design the following ablation studies: (1) LAC *w/o* critic removes the policy optimization step from LAC; (2) LAC *w/o* rollout does not predict future trajectories during action evaluations; (3) LAC *w/o* reflection removes the reflection procedure when sampling and evaluating candidate actions; (4) *critic-only* only uses critic's action evaluation information for action selection.

We report the results in Figure 4. LAC is better than all other variants in both ALFWorld and BabyAI-Text. Specifically, the performance decrease in LAC *w/o* critic and *critic-only* compared with LAC verify the effectiveness LAC's policy optimization with critic's action evaluations. And the result that LAC *w/o* reflection and LAC *w/o* rollout perform worse than LAC also suggests the necessity for incorporating reflections and future trajectory predictions.

5.4. Computational Cost Analysis

Our method conducts action evaluations by predicting future trajectories, which may bring extra computational cost per step. In Figure 5, we compare computational costs concerning the number of tokens spent and running time between LAC and other baselines. Specifically, though LAC has a higher computational cost per step due to the extra inference procedure of critics and the forward model, the total cost of LAC is still lower than most LLM-based baselines because LAC requires fewer steps to finish each task. This is due to LAC's higher success rate, enabling it to complete tasks within the maximum step limit, while other baselines often reach this limit without completing the tasks. If we only consider successful tasks, the step cost is similar across methods: LAC: 15.32 steps, ReAct: 17.75 steps, and RAP: 16.36 steps. For more computational cost analysis of other baselines, please refer to Appendix A.6.

5.5. Statistical Analyses

Directly interpreting LLM-based decision-making is inherently challenging due to two factors: (1) LLMs are black-box models, making it difficult to directly explain their outputs; (2) In sequential decision-making tasks, ground-

Table 1. Pearson correlations between Q-values and timesteps.

	Successful Trajectory (1843 steps in total)	Failed Trajectory (1092 steps in total)
$\log P$("GOOD")	0.35 ± 0.18	-0.37 ± 0.19
$\log P$("BAD")	-0.32 ± 0.19	0.38 ± 0.19
Q-value of chosen action	0.34 ± 0.18	-0.41 ± 0.18

Table 2. Confidence analysis of policy improvement.

Case (Proportion)	Prior Policy Conf.	Q-function Conf.	the Improved Policy Conf.
Both agree (49.81%)	0.49 (✓)	0.21 (✓)	0.74
Prior agrees, Q disagrees (29.12% cases)	0.34 (✓)	0.06 (✗)	0.28
Prior disagrees, Q agrees (19.28% cases)	0.21 (✗)	0.34 (✓)	0.22
Prior disagrees, Q disagrees (1.79% cases)	0.06 (✗)	0.03 (✗)	0.07

truth actions are typically unavailable for analyzing each action the policy selects—only task-level success or failure signals are observed.

Therefore, we provide statistical analyses to shed light on why the two key components of LAC—Q-function estimation and gradient-free policy improvement—work effectively in practice.

Q-function estimation reflects task progression We estimate Q-values using LLM-internal beliefs (log-probabilities of "GOOD"/"BAD" tokens). To validate that these Q-values meaningfully track task success, we compute the **Pearson correlation** between Q-values and timesteps within each trajectory. The intuition is: if an agent is on a successful path, Q-values should increase as it progresses (and decrease otherwise). The results in Table 1, averaged across 134 ALFWorld tasks, confirm this trend: successful trajectories show increasing Q-values, while failed ones show the opposite. This demonstrates that our Q-function captures the evolving success likelihood of the policy throughout a trajectory.

Policy improvement reflects model confidence Our gradient-free policy improvement balances the prior policy and the Q-function based on **relative confidence**. Specifically, we define a model's confidence as the difference between its top two scores: (1) For the prior policy: $\log P(a_1|s) - \log P(a_2|s)$; (2) For the Q-function: $Q(a_1|s) - Q(a_2|s)$, where a_1 and a_2 are the actions with the highest and second-highest probability (or Q-value), respectively, under the corresponding model.

We analyze which action (from the prior policy or Q-function) is selected by the improved policy and what confidence each model had in its choice. Table 2 summarizes the outcomes. This analysis shows that: (1)When both models agree, the confidence of the improved policy is highest; (2) When they disagree, the improved policy tends to trust the more confident model; (3) In low-confidence cases, the policy remains conservative. These results suggest our method implicitly aggregates decision knowledge from both sources by weighting based on model confidence, enabling effective policy improvement without explicit gradients.

6. Discussion

In this work, we introduce a novel LLM-based Actor-Critic algorithm LAC that leverages action evaluations to optimize the prior policy of LLM, enabling an effective integration of the decision-making insights derived from LLM prior and action evaluations that incorporate long-term planning. Compared with previous methods, LAC achieves high performance on three benchmarks that cover various action spaces even using lightweight open-source LLMs.

Despite the advancement, our method also has limitations and inspires possible future directions. Firstly, the reflection of LAC is only used before action generation, which can also be applied after action generation. For example, it can provide reflections for predicted future trajectories to resample candidate actions if the previous candidate actions all fail to complete the target task. Secondly, though we only expand one node for each candidate action for simplicity and efficiency and find it works effectively, LAC can also adopt tree-search to provide a more accurate assessment of candidate actions. Thirdly, though LAC could deal with scenarios with continuous final reward empirically by only treating getting the highest reward as success, it is an exciting future direction to develop a more principal method to deal with such situations. Last but not least, the effectiveness of LAC when applied to larger and more powerful LLM models needs further investigation.

Impact Statement

Our method is built upon open-source large language models (LLMs). Like other methods that use LLMs, our method also inherits some benefits and challenges from LLMs. For the benefits, our method directly exploits the prior knowledge from LLMs, which may reduce potential carbon costs compared with training policies from scratch. For the challenges, our method might be susceptible to producing unintended output when confronted with harmful input, such as unethical text or input intended for adversarial attacks. To solve this problem, we suggest a thoughtful deployment of our method, such as adding a filtering component.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Barto, A. G., Sutton, R. S., and Watkins, C. *Learning and sequential decision making*, volume 89. University of Massachusetts Amherst, MA, 1989.

Brooks, E., Walls, L., Lewis, R. L., and Singh, S. Large language models can implement policy iteration. *Advances in Neural Information Processing Systems*, 36, 2024.

Bylander, T. The computational complexity of propositional strips planning. *Artificial Intelligence*, 69(1-2):165–204, 1994.

Carta, T., Romac, C., Wolf, T., Lamprier, S., Sigaud, O., and Oudeyer, P.-Y. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, pp. 3676–3713. PMLR, 2023a.

Carta, T., Romac, C., Wolf, T., Lamprier, S., Sigaud, O., and Oudeyer, P.-Y. Grounding large language models in interactive environments with online reinforcement learning, 2023b.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., and Bengio, Y. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*, 2018.

Coulom, R. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pp. 72–83. Springer, 2006.

Cuadron, A., Li, D., Ma, W., Wang, X., Wang, Y., Zhuang, S., Liu, S., Schroeder, L. G., Xia, T., Mao, H., et al. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*, 2025.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Ding, Y., Zhang, X., Paxton, C., and Zhang, S. Task and motion planning with large language models for object rearrangement. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2086–2092. IEEE, 2023.

Dong, H., Wang, T., Liu, J., and Zhang, C. Low-rank modular reinforcement learning via muscle synergy. *Advances in Neural Information Processing Systems*, 35: 19861–19873, 2022.

Fu, D., Huang, J., Lu, S., Dong, G., Wang, Y., He, K., and Xu, W. Preact: Predicting future in react enhances agent's planning ability. *arXiv preprint arXiv:2402.11534*, 2024.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Go, D., Korbak, T., Kruszewski, G., Rozen, J., Ryu, N., and Dymetman, M. Aligning language models with preferences through f-divergence minimization. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11546–11583. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/go23a.html.

Hafner, D. Benchmarking the spectrum of agent capabilities. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=1W0z96MFEoH.

Hao, S., Gu, Y., Ma, H., Hong, J., Wang, Z., Wang, D., and Hu, Z. Reasoning with language model is planning with world model. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.507. URL https://aclanthology.org/2023.emnlp-main.507/.

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

Huang, J. and Chang, K. C.-C. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*, 2022.

Huang, W., Mordatch, I., and Pathak, D. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pp. 4455–4464. PMLR, 2020.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pp. 9118–9147. PMLR, 2022a.

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022b.

ichter, b., Brohan, A., Chebotar, Y., Finn, C., Hausman, K., Herzog, A., Ho, D., Ibarz, J., Irpan, A., Jang, E., Julian, R., Kalashnikov, D., Levine, S., Lu, Y., Parada, C., Rao, K., Sermanet, P., Toshev, A. T., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Yan, M., Brown, N., Ahn, M., Cortes, O., Sievers, N., Tan, C., Xu, S., Reyes, D., Rettinghouse, J., Quiambao, J., Pastor, P., Luu, L., Lee, K.-H., Kuang, Y., Jesmonth, S., Joshi, N. J., Jeffrey, K., Ruano, R. J., Hsu, J., Gopalakrishnan, K., David, B., Zeng, A., and Fu, C. K. Do as i can, not as i say: Grounding language in robotic affordances. In Liu, K., Kulic, D., and Ichnowski, J. (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/ichter23a.html.

Jain, A., Wojcik, B., Joachims, T., and Saxena, A. Learning trajectory preferences for manipulators via iterative improvement. *Advances in neural information processing systems*, 26, 2013.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Jin, H., Zhang, Y., Meng, D., Wang, J., and Tan, J. A comprehensive survey on process-oriented automatic text summarization with exploration of llm-based methods. *arXiv preprint arXiv:2403.02901*, 2024.

Jordan, M. I. et al. Why the logistic function? a tutorial discussion on probabilities and neural networks, 1995.

Kocsis, L. and Szepesvári, C. Bandit based monte-carlo planning. In *European conference on machine learning*, pp. 282–293. Springer, 2006.

Koh, J. Y., McAleer, S., Fried, D., and Salakhutdinov, R. Tree search for language model agents. *arXiv preprint arXiv:2407.01476*, 2024.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Li, J., Tang, T., Zhao, W. X., Nie, J.-Y., and Wen, J.-R. Pretrained language models for text generation: A survey. *ACM Comput. Surv.*, 56(9), April 2024a. ISSN 0360-0300. doi: 10.1145/3649449. URL https://doi.org/10.1145/3649449.

Li, X., Zhou, Y., and Dou, Z. Unigen: A unified generative framework for retrieval and question answering with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 8688–8696, 2024b.

Littman, M. L. *Algorithms for sequential decision-making*. Brown University, 1996.

Liu, Z., Hu, H., Zhang, S., Guo, H., Ke, S., Liu, B., and Wang, Z. Reason for future, act for now: A principled framework for autonomous llm agents with provable sample efficiency. *arXiv preprint arXiv:2309.17382*, 2023.

McCarthy, J. et al. *Situations, actions, and causal laws*. Comtex Scientific, 1963.

Meta. Meta llama 3. https://llama.meta.com/llama3/, 2024a.

Meta. Meta llama 3.1. https://ai.meta.com/blog/meta-llama-3-1/, 2024b.

Mialon, G., Dessi, R., Lomeli, M., Nalmpantis, C., Pasunuru, R., Raileanu, R., Roziere, B., Schick, T., Dwivedi-Yu, J., Celikyilmaz, A., Grave, E., LeCun, Y., and Scialom, T. Augmented language models: a survey. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=jh7wH2AzKK. Survey Certification.

Paglieri, D., Cupiał, B., Coward, S., Piterbarg, U., Wołczyk, M., Khan, A., Pignatelli, E., Kuciński, Ł., Pinto, L., Fergus, R., Foerster, J. N., Parker-Holder, J., and Rocktäschel, T. Benchmarking agentic llm and vlm reasoning on games. *arXiv preprint arXiv:2411.13543*, 2024.

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable

off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Peters, J. and Schaal, S. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007.

Prystawski, B., Li, M., and Goodman, N. Why think step by step? reasoning emerges from the locality of experience. *Advances in Neural Information Processing Systems*, 36, 2024.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Roziere, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Remez, T., Rapin, J., et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10740–10749, 2020.

Shridhar, M., Yuan, X., Côté, M.-A., Bisk, Y., Trischler, A., and Hausknecht, M. Alfworld: Aligning text and embodied environments for interactive learning, 2021.

Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., and Garg, A. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11523–11530. IEEE, 2023.

Szot, A., Schwarzer, M., Agrawal, H., Mazoure, B., Metcalf, R., Talbott, W., Mackraz, N., Hjelm, R. D., and Toshev, A. T. Large language models as generalizable policies for embodied tasks. In *The Twelfth International Conference on Learning Representations*, 2023.

Tan, W., Zhang, W., Liu, S., Zheng, L., Wang, X., and An, B. True knowledge comes from practice: Aligning large language models with embodied environments via reinforcement learning. In *ICLR*, 2024.

Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8:279–292, 1992.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

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

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Yao, W., Heinecke, S., Niebles, J. C., Liu, Z., Feng, Y., Xue, L., Rithesh, R., Chen, Z., Zhang, J., Arpit, D., et al. Retroformer: Retrospective large language agents with policy gradient optimization. In *The Twelfth International Conference on Learning Representations*.

Zhang, B., Mao, H., Ruan, J., Wen, Y., Li, Y., Zhang, S., Xu, Z., Li, D., Li, Z., Zhao, R., et al. Controlling large language model-based agents for large-scale decision-making: An actor-critic approach. *arXiv preprint arXiv:2311.13884*, 2023.

Zhang, D., Chen, L., Zhang, S., Xu, H., Zhao, Z., and Yu, K. Large language models are semi-parametric reinforcement learning agents. *Advances in Neural Information Processing Systems*, 36, 2024.

Zhou, A., Yan, K., Shlapentokh-Rothman, M., Wang, H., and Wang, Y.-X. Language agent tree search unifies reasoning, acting, and planning in language models. In *Forty-first International Conference on Machine Learning*, 2024a.

Zhou, Y., Zanette, A., Pan, J., Levine, S., and Kumar, A. Archer: Training language model agents via hierarchical multi-turn rl. In *Forty-first International Conference on Machine Learning*, 2024b.

A. Extra results

A.1. Results of other tasks in BabyAI-Text

For a complete comparison, we show the performance of LAC and baselines in other tasks from BabyAI-Text in Figure 6. Our LAC outperforms all other baselines, which further validates the effectiveness of LAC.

A.2. Results of using other definition of $\mathcal{Q}_{\text{LLM}}$

In LAC we define critic $\mathcal{Q}_{\text{LLM}}$ as $\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i) = \log \frac{P(y_w|g,h_t,a_t^i)}{P(y_l|g,h_t,a_t^i)}$. There are also other definitions, for example, the simplest variant is $\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i) = \log P(y_w|g, h_t, a_t^i)$. In this subsection, we provide a performance comparison between them in Figure 7. LAC outperforms the variant in most situations across tasks and models. We speculate that this is because LAC uses more information, *i.e.*, both $P(y_w|g, h_t, a_t^i)$ and $P(y_l|g, h_t, a_t^i)$, than the variant, and the evaluation might be more accurate and more stable. There might be other definitions of $\mathcal{Q}_{\text{LLM}}$ and among them, our $\mathcal{Q}_{\text{LLM}}$ is simple and effective.

A.3. Results of directly prompting the LLMs to output action evaluation

To further demonstrate the strengths of our critic $\mathcal{Q}_{\text{LLM}}$. We conducted additional experiments comparing the performance of LAC and LAC w/ direct evaluation on the WebShop benchmark. LAC generates value-based evaluations by extracting LLMs' internal beliefs of success and failure as described in Equation (2). For LAC w/ direct evaluation, we prompt the LLMs to directly output the probability of success $P(y_w)$, while keeping all other components unchanged. The Q-value is then calculated as $\log \frac{P(y_w)}{1-P(y_w)}$.

The results, presented in Table 3 and Table 4, show that our LAC method outperforms LAC w/ direct evaluation in terms of success rate and reward across most base models. Analysis of the resulting trajectories revealed that LAC w/ direct evaluation often produces a non-informative success probability (*e.g.*, $P(y_w) = 0.5$), leading to ineffective improvements in policy.

Table 3. Success rate of LAC and LAC w/ direct evaluation in WebShop

	CodeLlama-7B	Gemma-7B	Llama-3-8B	Mistral-7B
LAC	**32%**	**46%**	**39%**	**32%**
LAC w/ direct evaluation	27%	29%	33%	24%

Table 4. Rewards of LAC and LAC w/ direct evaluation in WebShop

	CodeLlama-7B	Gemma-7B	Llama-3-8B	Mistral-7B
LAC	**0.5840**	**0.7237**	**0.6733**	0.6299
LAC w/ direct evaluation	0.5636	0.6975	0.6453	**0.6333**

A.4. Comparison of our reflection component and chain-of-thought

In this subsection, we empirically compare the reflection method used in LAC and the Chain-of-Thought (CoT) methods (Wei et al., 2022; Kojima et al., 2022).

Though the reflection draws inspiration from CoT, it is not trivial to determine what the reflection content should be. Our method generates judgments based on previous actions and their outcomes, whereas CoT uses arbitrary thought generation without a structured focus on past mistakes. The reflection is specially designed for decision-making tasks and owns two advantages over naive CoT:

Strength 1. Our reflection can improve the policy directly by avoiding previous mistakes when the policy is conditioned on its generation.

To substantiate this claim, we conducted experiments comparing the performance of prior policy w/ reflection and piror policy w/ CoT on the WebShop benchmark. Here are the details of these two variants:

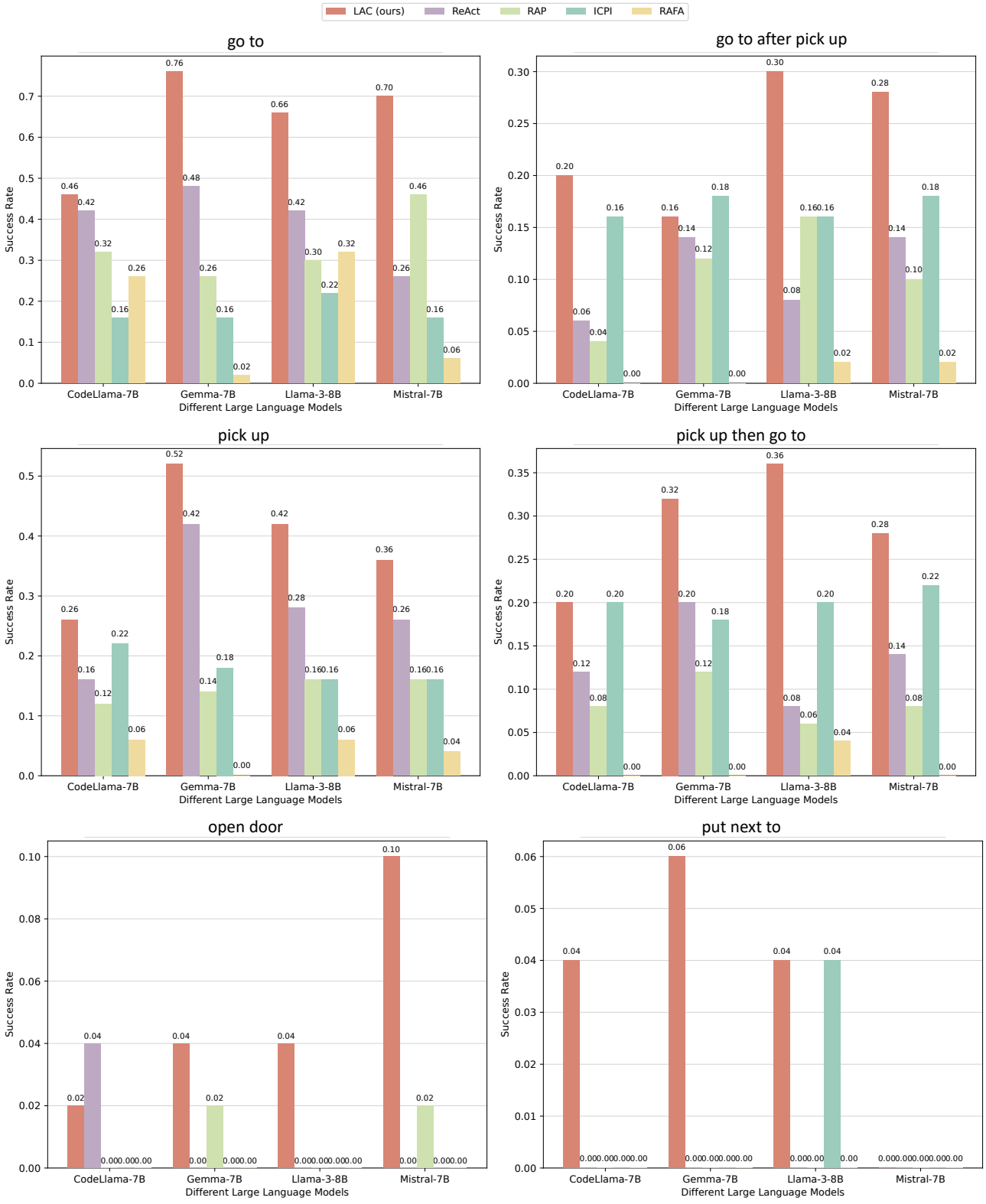

Figure 6. Performance of our LAC compared with various baselines in all tasks from BabyAI-Text.

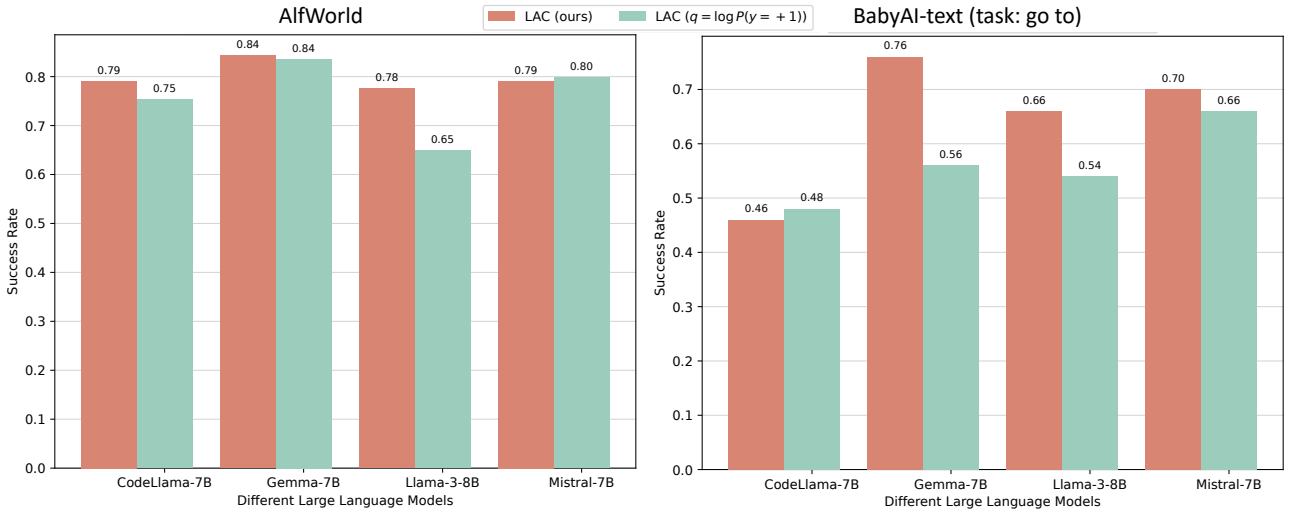

Figure 7. Performance of LAC when using different definition of critic $\mathcal{Q}_{\text{LLM}}$

(1) prior policy w/ reflection: We remove all components of LAC, leaving the policy and the reflection unchanged. Specifically, at each step, after observing the action results, the reflection first generates some judgments on previous actions, and then the policy selects the next action based on the judgments. (2) prior policy w/ CoT: We remove all components of LAC, except the policy. Additionally, we equip the policy with CoT by adding "Let's think step by step" to the prompt. Specifically, at each step, before choosing the next action, the CoT prompting component first outputs arbitrary thoughts that may help solve the task.

As shown in Table 5 and Table 6, prior policy w/ reflection consistently surpasses prior policy w/ CoT across most base models in terms of both success rate and reward. By analyzing the results, we found that prior policy w/ CoT may make the same mistake multiple times and get stuck at this mistake, while prior policy w/ reflection can largely avoid seen mistakes.

Table 5. Success rate of prior policy w/ reflection and prior policy w/ CoT in WebShop

	CodeLlama-7B	Gemma-7B	Llama-3-8B	Mistral-7B
prior policy w/ reflection	**21%**	**46%**	**39%**	**31%**
prior policy w/ CoT	**21%**	20%	31%	20%

Table 6. Rewards of prior policy w/ reflection and prior policy w/ CoT in WebShop

	CodeLlama-7B	Gemma-7B	Llama-3-8B	Mistral-7B
prior policy w/ reflection	**0.5739**	**0.6564**	**0.6556**	**0.6288**
prior policy w/ CoT	0.5520	0.5347	0.6379	0.4671

Strength 2. Our reflection can be seamlessly integrated with critic $\mathcal{Q}_{\text{LLM}}$, which helps the critic to generate more accurate value-based evaluations.

To further evaluate this integration, we compare the performance of LAC and LAC w/ CoT on the WebShop benchmark. The details of the two methods are as follows:

(1) LAC: Our original method. (2) LAC w/ CoT: We replace the reflection component of LAC with CoT and keep other components unchanged.

We show the results in Table 7 and Table 8. Our method LAC consistently outperforms LAC w/ CoT regarding success rate

and reward across all evaluated base models. This is because, without reflection's judgment on previous steps, the critic may output inaccurate value-based estimations, hindering the policy improvement phase.

Table 7. Success rate of LAC and LAC w/ CoT in WebShop

	CodeLlama-7B	Gemma-7B	Llama-3-8B	Mistral-7B
LAC	**32%**	**46%**	**39%**	**32%**
LAC w/ CoT	27%	29%	33%	24%

Table 8. Rewards of LAC and LAC w/ CoT in WebShop

	CodeLlama-7B	Gemma-7B	Llama-3-8B	Mistral-7B
LAC	**0.5840**	**0.7237**	**0.6733**	**0.6299**
LAC w/ CoT	0.5734	0.5270	0.6515	0.5101

In summary, the reflection's focused evaluations of past actions provide critical advantages over the simple reflections of CoT, leading to improved performance in decision-making tasks.

A.5. Comparison of LAC with more baselines on ALFWorld

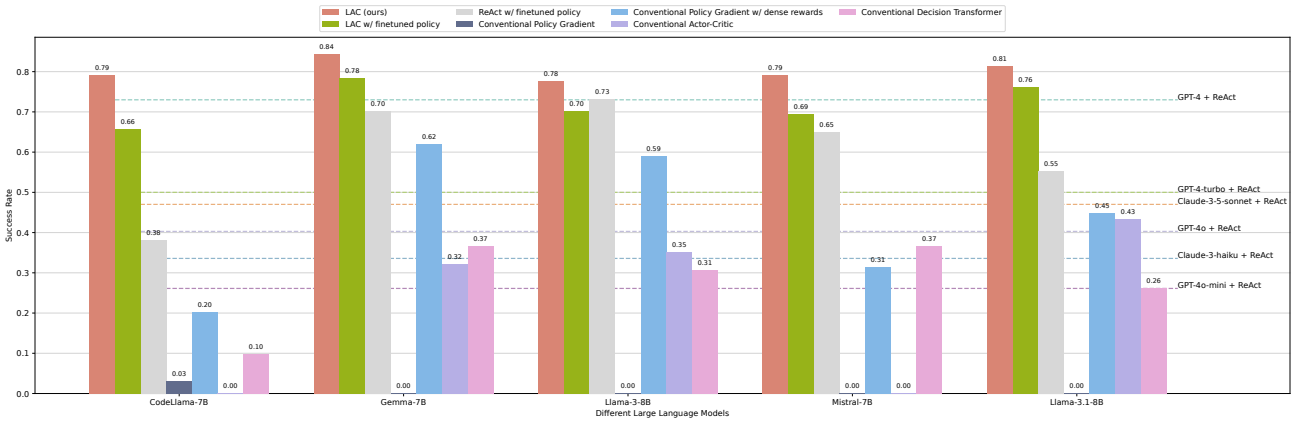

Figure 8. Performance of our LAC and LAC's variants compared with various baselines in benchmark ALFWorld.

In this subsection, we compare LAC with more baselines including some traditional RL methods implemented using LLMs on ALFWorld (Shridhar et al., 2021). The comparison is shown in Figure 8.

A.5.1. FINETUNE POLICY

While in LAC we fine-tune the model for generating reflections using a few trajectories, it is also possible to fine-tune the policy to generate actions using those trajectories. Therefore, a potential baseline could be fine-tuning the policy in ReAct. To demonstrate the improvement brought by fine-tuning the policy, we fine-tune the policy in ReAct (Yao et al., 2023) and show the results in Figure 8. We also show the results of LAC w/ fine-tuned policy in Figure 8. In brief, ReAct w/ fine-tuned policy is a strong baseline compared with other baselines, but still inferior to our method LAC and LAC w/ fine-tuned policy. Compared to LAC, the underperformance of LAC w/ fine-tuned policy arises from its tendency to overfit the training trajectories. This overfitting causes the policy to favor actions that are more frequent in the dataset, potentially leading to suboptimal action selection.

For example, in the ALFWorld training dataset, the action "take an apple from X" occurs frequently. After fine-tuning, the policy may disproportionately generate this action, even when it is irrelevant to the current goal. One case is that the current

goal is to "heat some egg and put it in the garbage can". When the agent sees an "apple 2" in "fridge 1", it generates and selects an irrelevant action "take apple 2 from fridge 1", which does not align with the task.

This tendency towards overfitting arises because the complexity of the policy function, which maps states s to actions a, often exceeds that of the critic. The policy often has to capture a wide variety of potential actions for each state, particularly in complex environments. However, the quite limited training dataset in our setting restricts its ability to generalize effectively, resulting in memorization of specific actions rather than flexible decision-making. In contrast, the model for reflection generation and rollouts focuses on capturing more predictable dynamics of the environment and simpler evaluation criteria. This typically requires simpler mappings than those needed for the policy, thus avoiding overfitting.

A.5.2. LLM-BASED RL VARIANTS

We also include some LLM-based RL variants as baselines to show the superiority of LAC over conventional RL algorithms. We design three LLM-based RL variants that are built upon pre-trained LLMs and directly extract actions/values information from LLMs without adding action/value heads, namely Conventional Policy Gradient, Conventional Policy Gradient w/ dense rewards and Conventional Actor-Critic in Figure 8.

For the implementation of the Conventional Policy Gradient, we need the probability of actions and the returns. To obtain the probability of actions, we directly use LLM to compute the conditional probability of each token in action $a_i = [w_1, w_2, \cdots, w_{|a_t|}]$ given the goal g, history h_t and then calculate their product:

$$\pi(a_t|g, h_t) = \Pi_{j=1}^{|a_t|} P_{LLM}(w_j|g, h_t, w_{<j})$$

in which $P_{LLM}(w_j|g, h_t, w_{<j})$ is the probability of token w_j given goal g, history h_t and previous tokens $w_{<j}$ computed by LLM. Then we regard the cumulative future rewards as the return G_t, which is $+1$ for successful trajectories and -1 for failed trajectories in the tasks we considered. Finally, the gradient of policy is $\mathbb{E}[\sum_t \nabla \log \pi(a_t|g, h_t) G_t]$. Conventional Policy Gradient w/ dense rewards is similar to Conventional Policy Gradient except that we manually add intermediate rewards for each step, and then use the cumulative future rewards as the return G_t.

For the implementation of the Conventional Actor-Critic, we additionally need a critic to estimate action values. As it is possible to train a new value head using only 18 trajectories, we instead approximate the action value similar to $\mathcal{Q}_{\text{LLM}}$ in our method LAC, *i.e.*

$$\mathcal{Q}_{\text{LLM}}(g, h_t, a_t, u_t) = \log \frac{P_{LLM}(y_w|g, h_t, a_t, u_t)}{P_{LLM}(y_l|g, h_t, a_t, u_t)}$$

in which $P_{LLM}(y_{\{w,l\}}|g, h_t, a_t, u_t)$ is the output probability of special positive/negative tokens like GOOD or BAD that indicate positive/negative results as LLM's belief on success/failure. Finally, the gradient of policy is $\mathbb{E}[\sum_t \nabla \log \pi(a_t|g, h_t) \mathcal{Q}_{\text{LLM}}(g, h_t, a_t, u_t)]$.

In summary, Conventional Policy Gradient exhibits almost all zero performance, which is due to the extremely sparse reward problems, compared with Conventional Policy Gradient w/ dense rewards. Conventional Actor-Critic demonstrates non-zero performance only on some stronger LLMs like Gemma-7B (Team et al., 2024), Llama-3-8B (Meta, 2024a) and Llama-3.1-8B (Meta, 2024b), which may be because the optimization method of conventional actor-critic is not suitable in insufficient data settings.

A.5.3. DECISION TRANSFORMER

In addition to the aforementioned LLM-based RL variant, Decision Transformer (Chen et al., 2021) is also a potential solution in combining RL and transformer-based LLMs. We fine-tune pretrained LLMs in a similar way as conventional decision transformers. We construct a dataset using decision-transformers' trajectory representation: $\tau = [R_1, s_1, a_1, R_2, s_2, a_2, \cdots]$, in which R_t is return-to-go, *i.e.*, +1 for successful trajectories and -1 for failed trajectories in our extremely sparse reward settings. Then we fine-tune LLMs with next-token prediction loss on these trajectories. During execution, we insert +1 before state s_t to specify the desired outcome. The results are shown in Figure 8 as Conventional Decision Transformer. In short, Conventional Decision Transformer exhibits a similar performance to ReAct, which may be because the 18 trajectories are insufficient for fine-tuning decision transformers.

Our method LAC is better than all considered baselines because of its ability to handle extremely sparse reward problems using LLM's prior knowledge and to fully utilize insufficient data.

A.6. Computational cost analysis of LAC with more baselines in ALFWorld

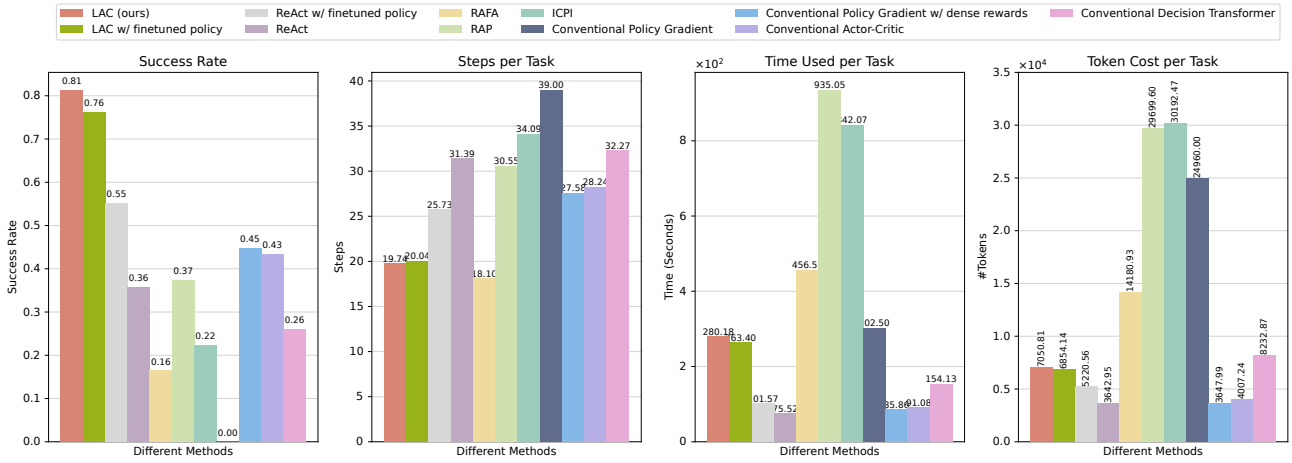

Figure 9. Computational cost analysis of our LAC compared with various baselines in benchmarks ALFWorld. Though LAC may have a higher computational cost per step due to the extra inference procedure of critics and the forward model, the total cost of LAC is still lower than most LLM-based baselines because LAC requires fewer steps to finish each task.

In this subsection, we demonstrate the computation cost of LAC and other baselines in Figure 9. We show the success rate, steps per task, time used per task, and token cost per task respectively. Specifically, though LAC has a higher computational cost per step due to the extra inference procedure of critics and the forward model, the total cost is still lower than most LLM-based baselines because LAC has a higher success rate and requires fewer steps to finish each task.

A.7. Illustration of ALFWorld and BabyAI-Text

We show the illustrative example of ALFWorld and BabyAI-Text in Figure 10 and Figure 11, respectively.

Figure 10 presents a concrete scenario where the agent's goal is to "put a saltshaker in the drawer". At a critical decision step (Step 4), we observe the following intuitive distinction between components: (1) LLM-based prior policy alone mistakenly suggests "go to drawer 1" because the base LLM model overlooks that the agent has already found the correct object ("saltshaker 1") in cabinet 2. This error exemplifies the common hallucination problem in LLMs, which occurs when the model disregards previous states and incorrectly recommends irrelevant actions. (2) In contrast, the critic suggests "take saltshaker 1 from cabinet 2" because it evaluates potential actions by predicting future trajectories and determines that this action will successfully pick up the correct object. (3) Our method leverages these distinct insights by optimizing the prior policy's action distribution based on the critic's evaluation. It effectively corrects the errors introduced by the prior policy, balancing the strengths of prior policy (flexible but sometimes inaccurate) and critic evaluations (accurate but computationally intensive).

This analysis shows that: (1)When both models agree, the confidence of the improved policy is highest; (2) When they disagree, the improved policy tends to trust the more confident model; (3) In low-confidence cases, the policy remains conservative.

These results suggest our method implicitly aggregates decision knowledge from both sources by weighting based on model confidence, enabling effective policy improvement without explicit gradients.

A.8. Results of different critic improvement methods

Empirically, we found that the critic can be improved via fine-tuning on a few trajectories. Please refer to Appendix B.2 for more fine-tuning details. To show the effectiveness of fine-tuning, we present the performance of LAC and other variants when we just add these examples into the prompt, *i.e.*, in-context learning, on task "go to" and "pick up" from BabyAI-Text in Table 9 and Table 10 respectively. We also show the the performance improvement if we do fine-tuning in the parentheses. This result indicates that fine-tuning can incorporate extra knowledge into LLMs better than in-context learning in our

Prompt

[Instruction]: Interact with a household to solve a task. Here are two examples. (...)
[Task]: 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 saltshaker in drawer.

Inputs & Outputs of LAC (Ours)

(...)

[Act 1]: go to cabinet 1

[Obs 2]: On the cabinet 1, you see a glassbottle 1.
[LAC Inference 2]
[Act 2]: go to cabinet 2 ✓

[Obs 3]: The cabinet 2 is closed.
[LAC Inference 3]
[Act 3]: open cabinet 2 ✓

[Obs 4]: You open the cabinet 2. The cabinet 2 is open. In it, you see a plate 1, a saltshaker 1, and a soapbottle 1.
[LAC Inference 4]
[Act 4]: take saltshaker 1 from cabinet 2 ✓
[Obs 5]: You pick up the saltshaker 1 from the cabinet 2.

(...)

[Obs 7]: You open the drawer 1. The drawer 1 is open. In it, you see a fork 1, and a peppershaker 1.
[LAC Inference 7]
[Act 7]: put saltshaker 1 in/on drawer 1 ✓
[Obs 8]: You put the saltshaker 1 in/on the drawer 1.

Action Probs. (Partial)

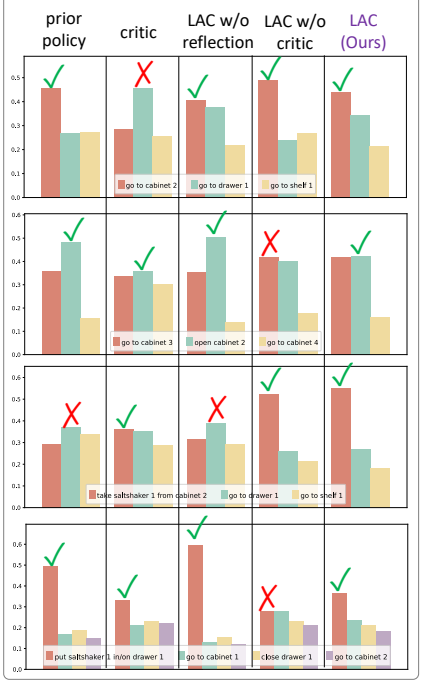

Figure 10. An illustrative explanation of our method LAC in ALFWorld. The histogram on the right shows the action probabilities of different methods. While LLM's prior policy (π_{LLM}) and critic, as well as LAC w/o reflection, make mistakes at different time steps, LAC (ours) can select the correct action by optimizing the policy given action evaluation. The LAC inference step is detailed in Figure 1.

Prompt

[Instruction]: Assume that you are an agent in a Grid World. Given a goal, your task is to execute a sequence of actions to achieve the goal.
[Possible actions]: turn left, turn right, go forward, pick up, drop, toggle.
[Goal]: go to the green ball

Inputs & Outputs of LAC (Ours)

[Obs 1]: You see a wall 5 steps forward, You see a wall 2 steps right, **You see a green ball 3 steps left and 4 steps forward**, You see a grey key 2 steps left and 2 steps forward, You see a blue key 1 step left and 3 steps forward, You see a green key 1 step left and 2 steps forward, You see a blue ball 1 step right and 1 step forward
[LAC Inference 1]
[Act 1]: go forward ✓

[Obs 2]: (...) You see a green ball 3 steps left and 3 steps forward, (...) You see a blue ball 1 step right
[LAC Inference 2]
[Act 2]: go forward ✓

(...)

[Obs 5]: You see a wall 1 step forward, (...) You see a green ball 3 steps left
[LAC Inference 5]
[Act 5]: turn left ✓

[Obs 6]: (...) You see a red ball 4 steps forward, You see a green ball 3 steps forward
[LAC Inference 6]
[Act 6]: go forward ✓

Action Probs. (Partial)

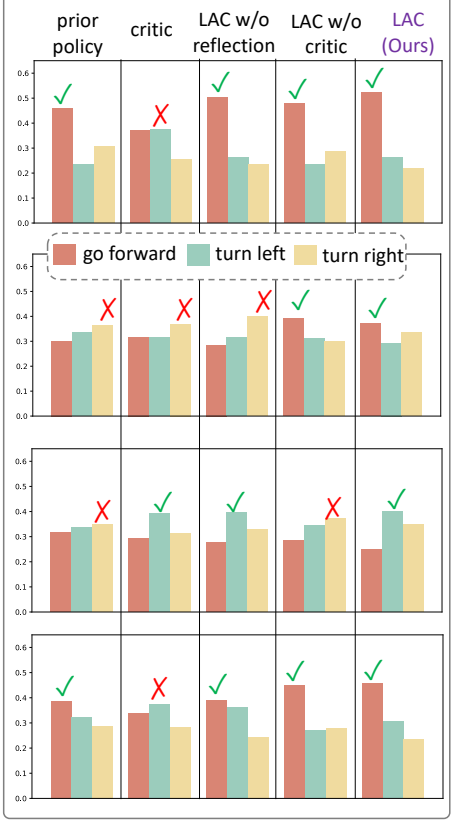

Figure 11. An illustrative explanation of our method LAC in BabyAI-Text. The histogram on the right shows the action probabilities of different methods. While LLM's prior policy (π_{LLM}) and critic, as well as LAC w/o reflection, make mistakes at different time steps, LAC (ours) can select the correct action by optimizing the policy given action evaluation. Please refer to Table 30 for the full trajectory.

case and improve the quality of action evaluation, yielding better performance. It is worth mentioning that our LAC still outperforms baselines without fine-tuning.

Table 9. Performance of two critic improvement methods: in-context learning or fine-tuning.

	CodeLlama-7B	Gemma-7B	Llama-3-8B	Mistral-7B
LAC	0.30 (↑ 0.16)	0.62 (↑ 0.14)	0.32 (↑ 0.34)	0.24 (↑ 0.46)
LAC *w/o* reflection	0.30 (↑ 0.02)	0.58 (↑ 0.02)	0.38 (↑ 0.10)	0.26 (↑ 0.28)
LAC *w/o* critic	0.28 (↑ 0.06)	0.48 (↑ 0.04)	0.42 (↑ 0.10)	0.10 (↑ 0.28)
critic-only	0.42 (↑ 0.04)	0.40 (↑ 0.24)	0.34 (↑ 0.22)	0.38 (↑ 0.24)

Table 10. Performance of two critic improvement methods: in-context learning or fine-tuning.

	CodeLlama-7B	Gemma-7B	Llama-3-8B	Mistral-7B
LAC	0.20 (↑ 0.06)	0.22 (↑ 0.20)	0.34 (↑ 0.08)	0.20 (↑ 0.16)
LAC *w/o* reflection	0.16 (↑ 0.08)	0.32 (↑ 0.04)	0.32 (↑ 0.04)	0.22 (↑ 0.06)
LAC *w/o* critic	0.12 (↑ 0.14)	0.36 (↑ 0.20)	0.28 (↑ 0.06)	0.26 (↑ 0.04)
critic-only	0.22 (↑ 0.04)	0.24 (↑ 0.26)	0.16 (↑ 0.16)	0.16 (↑ 0.26)

A.9. Analysis of the fine-tuning process in LAC

In order to improve the quality of the reflections generated by LLM, we finetune the LLM that generates the reflections. In this section, we analyze the finetuning process, showing the impact of finetuning, as well as the impact of different data amounts and positive and negative sample ratios on task success rates. The comparison can be seen in Figure 12.

In Figure 12 (a), we show the influence of fine-tuning data size. We use 9, 18, 27 and 36 trajectories to fine-tune LLMs, and show the final success rate on 134 evaluation tasks. In summary, larger data sizes (27 or 36 trajectories) generally bring higher success rate, while small data sizes (18 and even 9 trajectories in some cases) are already enough for LAC to achieve outperformance.

Figure 12 (b) shows the influence of different positive/negative sample ratio (positive:negative = 0:1, 1:3, 1:1, 3:1 and 1:0) on final performance. We keep the total number of samples the same and just change positive/negative ratio. In short, our LAC is robust to reasonable positive/negative ratios (*e.g.* 1:3, 1:1, 3:1), while LAC based on CodeLlama-7B (Roziere et al., 2023) and Gemma-7B (Team et al., 2024) even perform better when given all positive samples (1:0).

Figure 12 (c) shows the learning curves of the fine-tuning process. We plot the next prediction loss and positive/negative tokens prediction accuracy for CodeLlama-7B (Roziere et al., 2023). In short, as the next token prediction loss decreases during fine-tuning, the accuracy of predicting the special tokens (GOOD or BAD) increases, which exhibits the effect of the fine-tuning process.

A.10. Ablation studies on the impact of finetuning

The finetuning process is important in our method, but it can also be replaced by in-context learning, where several examples are inserted in each input. Here we compare our method with ReAct regarding the effect of finetuning in ALFWorld. The results, presented in Table 11, show that our method outperforms the baseline both with and without finetuning, and the performance can be further improved with finetuning. With finetuing, our method provides more consistent performance with different LLMs.

A.11. Results of different hyper-parameter α

The hyper-parameter α in Equation (4) controls the deviation from the original policy π_{LLM}. In this subsection, we grid-search this hyper-parameter over $\{1/2, 1, 2, 5, 10\}$ in task "go to" of BabyAI-Text, then we fix α for other tasks: $\alpha = 1$ for model CodeLlama-7B, $\alpha = 2$ for model Gemma-7B, $\alpha = 2$ for model Llama-3-8B and $\alpha = 10$ for model Mistral-7B.

As for benchmark ALFWorld, we fixed $\alpha = 1$ in all experiments.

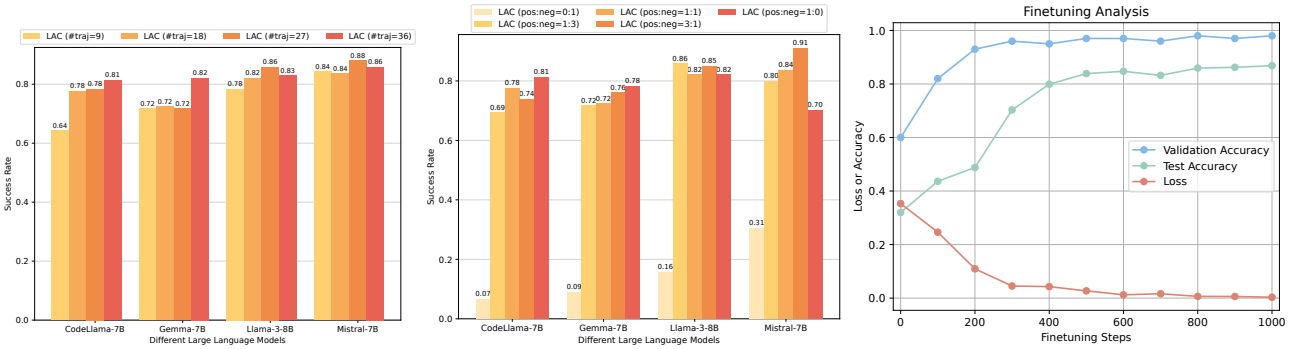

Figure 12. Analysis regarding the fine-tuning process of our LAC. (a) Influence of the fine-tuning data size. Larger data sizes (27, 36 trajectories) generally bring higher performance, but small data sizes (18 and even 9 trajectories) are already enough for our method to achieve outperformance. (b) Influence of the positive/negative data ratio. LAC is robust to reasonable positive/negative ratios (1:3, 1:1, 3:1) while CodeLlama-7B and Gemma-7B-based LAC even perform better given all positive data (1:0). (c) Learning curves of next-token prediction loss and positive/negative tokens prediction accuracy for CodeLlama-7B and ALFWorld.

Table 11. Ablation studies on the impact of finetuning.

	CodeLlama-7B	Gemma-7B	Llama-3-8B	Mistral-7B
LAC (*w/o* finetuning)	**0.39**	**0.59**	**0.71**	**0.57**
ReAct (*w/o* finetuning)	0.20	0.54	0.31	0.34
LAC (*w/* finetuning)	**0.79**	**0.84**	**0.78**	**0.79**
ReAct (*w/* finetuning)	0.38	0.70	0.73	0.65

A.12. Performance comparison in other benchmarks

To further show the strength of our method, we conduct preliminary experiments on Crafter (Hafner, 2022), using the implementation from BALROG (Paglieri et al., 2024). Crafter is a 2D survival game specifically designed to test long-horizon reasoning and sequential decision-making, with tasks involving resource gathering, crafting, and combat. It represents a significantly more complex setting than ALFWorld and WebShop.

We evaluated our method on this benchmark using Llama-3.1-8B-It, following BALROG's official evaluation protocol. We compare LAC with several representative baselines from BALROG's GitHub repository. The preliminary results are summarized in Table 13. Our method achieves higher performance than other available baselines under identical evaluation settings. These preliminary results provide further evidence of the robustness, effectiveness, and adaptability of our proposed actor-critic approach, particularly in significantly more challenging and complex decision-making environments. It is worth noting that CoT performs worse than the baseline Naive on the Crafter benchmark. We hypothesize that it is due to the model's inconsistency within their chain-of-thoughts. Two contiguous chains of thoughts might lead model to take actions which push towards different goals, which is not ideal. The authors also note that this is a problem especially with smaller, weaker models.

A.13. Preliminary experiments with reasoning LLMs

We have conducted preliminary experiments with reasoning LLMs like DeepSeek-R1-Distill-Qwen-7B (DeepSeek-AI, 2025). However, we observed that they often tend to overthink rather than output direct environmental actions in both our method and the baseline ReAct. For instance, even when we explicitly prompt the reasoning LLMs to output actions (*e.g.*, "Please make sure your response is in the following format:\n> {The action you choose}"), the models still generated detailed explanations but avoided selecting the next action. A typical response might be: "I need to find a key to open the safe or locate the pencil in the drawers. Since I can't (...), I'm unable (...), I must (...)". This issue has also been noted in prior work (Cuadron et al., 2025). We believe that using reasoning LLMs for decision-making tasks requires deeper

Table 12. Results of different hyper-parameter α

	CodeLlama-7B	Gemma-7B	Llama-3-8B	Mistral-7B
LAC ($\alpha = 1/2$)	**0.46**	0.54	0.62	0.68
LAC ($\alpha = 1$)	**0.46**	0.62	0.64	0.58
LAC ($\alpha = 2$)	0.44	**0.76**	**0.66**	0.64
LAC ($\alpha = 5$)	**0.46**	0.72	0.62	0.64
LAC ($\alpha = 10$)	0.40	0.58	0.60	**0.70**

Table 13. Performance comparison of LAC (Ours) and other baselines in Crafter (from BALROG)

	Llama-3.1-8B-It
LAC (Ours)	**25.91% ± 1.93%**
Naive (direct action generation)	20.45% ± 4.26%
Robust Naive (formatted actions)	4.55% ± 1.57%
CoT (ReAct, reason then act)	18.64% ± 3.24%
Robust CoT (reason + formmated actions)	15.46% ± 3.59%
Few-Shot (in-context examples)	12.73% ± 1.25%

exploration to balance internal reasoning with effective environmental interaction.

B. Method details

B.1. Deriving the solution of the KL-constrained maximization objective

In this subsection, we will derive Equation (5). We optimize the following objective:

$$\max_{\pi} \mathbb{E}_{a_t^i \sim \pi(a_t^i|g,h_t)} \left[\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i, u_t^i)\right] - \frac{1}{\alpha}\mathbb{D}_{KL}\left[\pi(a_t^i|g,h_t) \| \pi_{\text{LLM}}(a_t^i|g,h_t)\right].$$

We now have:

$$\begin{aligned}
&\max_{\pi} \mathbb{E}_{a_t^i \sim \pi(a_t^i|g,h_t)} \left[\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i, u_t^i)\right] - \frac{1}{\alpha}\mathbb{D}_{KL}\left[\pi(a_t^i|g,h_t) \| \pi_{\text{LLM}}(a_t^i|g,h_t)\right] \\
=&\max_{\pi} \mathbb{E}_{a_t^i \sim \pi(a_t^i|g,h_t)} \left[\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i, u_t^i) - \frac{1}{\alpha}\log\frac{\pi(a_t^i|g,h_t)}{\pi_{\text{LLM}}(a_t^i|g,h_t)}\right] \\
=&\min_{\pi} \mathbb{E}_{a_t^i \sim \pi(a_t^i|g,h_t)} \left[\log\frac{\pi(a_t^i|g,h_t)}{\pi_{\text{LLM}}(a_t^i|g,h_t)} - \alpha\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i, u_t^i)\right] \\
=&\min_{\pi} \mathbb{E}_{a_t^i \sim \pi(a_t^i|g,h_t)} \left[\log\frac{\pi(a_t^i|g,h_t)}{\frac{1}{Z(g,h_t)}\pi_{\text{LLM}}(a_t^i|g,h_t)\exp\left(\alpha\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i, u_t^i)\right)} - \log Z(g,h_t)\right]
\end{aligned}$$

where we have the partition function:

$$Z(g,h_t) = \sum_{a_t^i} \pi_{\text{LLM}}(a_t^i|g,h_t)\exp\left(\alpha\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i, u_t^i)\right).$$

Since the partition function is a function of only g, h_t and the original policy π_{LLM}, but does not depend on the optimized policy π, we define

$$\pi^*(a_t^i|g,h_t) = \frac{1}{Z(g,h_t)}\pi_{\text{LLM}}(a_t^i|g,h_t)\exp\left(\alpha\mathcal{Q}_{\text{LLM}}(g, h_t, a_t^i, u_t^i)\right).$$

This definition of policy if a valid probability distribution as $\pi^*(a_t^i|g,h_t)$ for all a_t^i and $\sum_{a_t^i}\pi^*(a_t^i|g,h_t)=1$. As $Z(g,h_t)$ is not a function of a_t^i, we can then re-organize the objective as:

$$\begin{aligned}
&\min_\pi \mathbb{E}_{a_t^i\sim\pi(a_t^i|g,h_t)}\left[\log\frac{\pi(a_t^i|g,h_t)}{\frac{1}{Z(g,h_t)}\pi_{\text{LLM}}(a_t^i|g,h_t)\exp\left(\alpha\mathcal{Q}_{\text{LLM}}(g,h_t,a_t^i,u_t^i)\right)}-\log Z(g,h_t)\right]\\
=&\min_\pi \mathbb{E}_{a_t^i\sim\pi(a_t^i|g,h_t)}\left[\log\frac{\pi(a_t^i|g,h_t)}{\pi^*(a_t^i|g,h_t)}-\log Z(g,h_t)\right]\\
=&\min_\pi \mathbb{D}_{KL}\left[\pi(a_t^i|g,h_t)\|\pi^*(a_t^i|g,h_t)\right]-\log Z(g,h_t).
\end{aligned}$$

Then since $Z(g,h_t)$ does not depend on π, we can only care about the KL-divergence, which is minimized at 0 if and only if the two distributions are identical. Therefore, the optimal solution is

$$\pi(a_t^i|g,h_t)=\pi^*(a_t^i|g,h_t)=\frac{1}{Z(g,h_t)}\pi_{\text{LLM}}(a_t^i|g,h_t)\exp\left(\alpha\mathcal{Q}_{\text{LLM}}(g,h_t,a_t^i,u_t^i)\right), \tag{7}$$

which completes the derivation.

B.2. Critic improvement of LAC

The reflections critic $\mathcal{Q}_{\text{LLM}}$, and forward model f_{LLM} we used can be easily implemented by prompting LLMs via providing instructions or few-shot examples from similar tasks like prior work (Yao et al., 2023; Liu et al., 2023). However, empirically, we found that they can be further improved via fine-tuning LLMs with simple next-token prediction loss on several samples collected from training tasks. In this work, we consider fine-tuning for ALFWorld and BabyAI-Text to show the impact of fine-tuning. We use 18 trajectories for each benchmark for fine-tuning, including two successful trajectories and one failed trajectory for each task-type. Though 18 trajectories are significantly fewer than what is required for conventional reinforcement learning algorithms, they are generally enough for our method. Each trajectory has the following format: $(g,o_0,a_1,o_1,c_1,\cdots,a_H,o_H,c_H)$, where H is the episode length and c_i is a reflection of action a_t. Each c_t includes an explanation about the action a_t (*e.g.*, "I have found object-X. This step is " or "I should take object-X instead of object-Y first. This step is ") and a special token that indicates positive/negative judgment (*e.g.*, "GOOD" or "BAD").

Practically, we just fine-tune the LLM once and use it to generate all the reflections, construct critic $\mathcal{Q}_{\text{LLM}}$, and forward model f_{LLM}, thanks to the fine-tuning with the above data format. Specifically, when minimizing the loss of predicting future trajectories, the forward model f_{LLM} is improved. When minimizing the loss of generating reflections, the reflections, critic $\mathcal{Q}_{\text{LLM}}$ are both improved. The latter is because reflections also contain special tokens that indicate positive/negative judgments, whose generated probabilities are used to calculate $\mathcal{Q}_{\text{LLM}}$ in Equation (3). We analyze this fine-tuning process in Appendix A.9. Some examples of the labeled trajectories in ALFWorld and BabyAI-Text are shown in Table 27 and Table 28 respectively.

C. Experiment details

C.1. Benchmark details

C.1.1. ALFWorld: benchmark with high-level actions

We choose ALFWorld (Shridhar et al., 2021), a text-based household environment, to demonstrate the effectiveness of LAC on high-level planning. ALFWorld is a synthetic text-based game aligned with ALFRED (Shridhar et al., 2020) benchmark. There are 6 types of tasks in this environment, which require the agent to achieve a high-level goal through a sequence of high-level actions, *e.g.* "go to place-X", "take object-Y from place-X", *etc*. The details about the 6 task types in ALFWorld are shown in Table 14.

A challenge built into ALFWorld is that the agent needs to explore the environment to find a target object. The commonsense knowledge in LLMs about the likely locations for common household items makes this environment suitable for LLMs to solve. The reward is 1 only when the agent reaches the goal. Following ReAct, we evaluate 134 unseen evaluation games in a task-specific setup.

Table 14. All the task types and the corresponding goals for ALFWorld

Type	Description
Pick & Place	The agent needs to put a target object to a target place, *e.g.* put some spraybottle on toilets, find some apple and put it in sidetable, *etc.*
Clean & Place	The agent needs to find a target object, clean it and put it to a target place, *e.g.* clean some apple and put it in sidetable, put a clean lettuce in diningtable, *etc.*
Heat & Place	The agent needs to find a target object, heat it and put it to a target place, *e.g.* heat some egg and put it in diningtable, put a hot apple in fridge, *etc.*
Cool & Place	The agent needs to find a target object, cool it and put it to a target place, *e.g.* cool some pan and put it in stoveburner, put a cool mug in shelf, *etc.*
Examine & Place	The agent needs to find a target object, and examine it with desklamp, *e.g.* look at bowl under the desklamp, examine the pen with the desklamp, *etc.*
Pick Two & Place	The agent needs to put two target objects to a target place, *e.g.* put two saltshaker in drawer, find two pen and put them in dresser, *etc.*

C.1.2. BABYAI-TEXT: BENCHMARK WITH LOW-LEVEL ACTIONS

For decision-making tasks with low-level planning, we adopt BabyAI-Text (Carta et al., 2023b) as our test-bed. BabyAI-Text is a text-only version environment extended from the BabyAI platform (Chevalier-Boisvert et al., 2018). BabyAI-Text is a Grid World environment, in which the agent and objects are placed in a room of 8×8 tiles. The agent has 6 primitive actions: turn left, turn right, go forward, pick up, drop, toggle, to solve a task described in natural language (*e.g.* Pick up the red box). The agent has access to a 7×7 partial view, which means it can only observe the objects belonging to the 7×7 grid in front of it. In addition to objects relevant to completing a given task, there are also other distractors in the room. All the task types in BabyAI-Text are shown in Table 15.

Table 15. All the task types and the corresponding goals for BabyAI-Text

Type	Description
go to	The agent needs to find target object and go to it, *e.g.* go to the green key, go to the red ball, *etc.*
pick up	The agent needs to find target object, go to it and pick up it, *e.g.* pick up the blue key, pick up the purple ball, *etc.*
go to after pick up	The agent needs to find and pick up one object, then go to another object, *e.g.* go to the blue key after you pick up the green key *etc.*
pick up then go to	The agent needs to find and pick up one object, then go to another object, *e.g.* pick up the green box, then go to the purple box *etc.*
put next to	The agent needs to find and pick up one object, then go to another object and put the first object next to it, *e.g.* put the grey key next to the yellow ball *etc.*
open door	The agent needs to know which key to pick up, then find and pick up it to open the door, *e.g.* open the door, open the blue door, *etc.*

Unlike ALFWorld, an agent interacting with BabyAI-Text needs to find out the suitable low-level action to execute at each step. We evaluate on the test environment in BabyAI-Text. The objects in a room are randomly chosen, and their position, as well as the agent's position, are also random. Considering the time and computational resource constraints, we evaluate on 50 tasks for each task type, yielding 300 tasks total.

C.1.3. WEBSHOP: BENCHMARK WITH POTENTIALLY INFINITE ACTION SPACE

In some scenarios, the space of possible actions can be infinite. We adopt a simplified environment, WebShop (Yao et al., 2022), with potentially infinite action space to demonstrate the effectiveness of our method. WebShop requires an agent to purchase a desired product based on textual instructions through the interaction with a simulated web environment. The available actions include search something using textual descriptions and choose button to click like "[item ID]", "back to search", "buy now", *etc.* Due to the nature of natural language, one can actually search anything, leading to an unbounded set of potential actions. We evaluate 100 tasks in WebShop for our method and all the baselines.

One thing that should be mentioned about this benchmark is that the final reward is a value between 0 and 1, depending on the degree to which the final purchased product meets the requirements. Thus in this work, we report the success rate and final reward simultaneously, where success rate indicates the proportion of products that the agent finally buys that fully meet the requirements, and final reward indicates the average final reward on all evaluated tasks.

C.2. Implementation details of baselines

C.2.1. REACT

ReAct (Yao et al., 2023) uses LLM as policy to generate action directly. They combine reasoning and acting during the interaction with the environment to improve action selection.

Table 16 shows an example of a trajectory interleaving between actions and observations in ALFWorld. To leverage the

reasoning ability of LLMs, ReAct adds an extra thinking action to the original action set in ALFWorld, allowing LLM agent to engage in reasoning about task decomposition and goal achievement during task execution. An annotated trajectory for ALFWorld given by ReAct is shown in Table 17. Text labeled in light green is the annotated thought to help LLM complete a task better.

Table 16. An example of successful trajectory for ALFWorld

```
You are in the middle of a room.  Looking quickly around you, you see a cabinet 4, a cabinet 3, a
cabinet 2, a cabinet 1, a countertop 1, a garbagecan 1, a handtowelholder 2, a handtowelholder 1, a
sinkbasin 2, a sinkbasin 1, a toilet 1, a toiletpaperhanger 1, and a towelholder 1.
Your task is to:  put some spraybottle on toilet.
> go to cabinet 1
On the cabinet 1, you see a cloth 1, a soapbar 1, a soapbottle 1.
> go to cabinet 2
The cabinet 2 is closed.
> open cabinet 2
You open the cabinet 2.  The cabinet 2 is open.  In it, you see a candle 1, and a spraybottle 2.
> take spraybottle 2 from cabinet 2
You pick up the spraybottle 2 from the cabinet 2.
> go to toilet 1
On the toilet 1, you see a soapbottle 2.
> put spraybottle 2 in/on toilet 1
You put the spraybottle 2 in/on the toilet 1.
```

Table 17. An example of annotated successful trajectory for ALFWorld in ReAct

```
You are in the middle of a room.  Looking quickly around you, you see a cabinet 4, a cabinet 3, a
cabinet 2, a cabinet 1, a countertop 1, a garbagecan 1, a handtowelholder 2, a handtowelholder 1, a
sinkbasin 2, a sinkbasin 1, a toilet 1, a toiletpaperhanger 1, and a towelholder 1.
Your task is to:  put some spraybottle on toilet.
> think:  To solve the task, I need to find and take a sparybottle, then put it on toilet.
OK.
> think:  First I need to find a spraybottle.  A spraybottle is more likely to appear in cabinet (1-4),
countertop (1), toilet (1), sinkbasin (1-2), garbagecan (1).  I can check one by one, starting with
cabinet 1.
OK.
> go to cabinet 1
On the cabinet 1, you see a cloth 1, a soapbar 1, a soapbottle 1.
> go to cabinet 2
The cabinet 2 is closed.
> open cabinet 2
You open the cabinet 2.  The cabinet 2 is open.  In it, you see a candle 1, and a spraybottle 2.
> think:  Now I find a spraybottle (2).  Next, I need to take it.
> take spraybottle 2 from cabinet 2
You pick up the spraybottle 2 from the cabinet 2.
> think:  Now I take a spraybottle (2).  Next, I need to put it in/on toilet 1.
OK.
> go to toilet 1
On the toilet 1, you see a soapbottle 2.
> put spraybottle 2 in/on toilet 1
You put the spraybottle 2 in/on the toilet 1.
```

During the interaction with the environment, two successful trajectories like in Table 17 are provided to LLMs to leverage the in-context learning ability of LLMs. The LLMs will generate an action to be executed in the environment or conduct some reasoning about how to achieve the final goal. The chosen action and new observation are appended to the prompt for LLMs to form a sequential decision-making process.

Similar to ALFWorld, we annotate some successful trajectories in BabyAI-Text to teach a ReAct agent how to complete a task in BabyAI-Text. Examples of successful trajectories without thoughts and with thoughts are shown in Table 18 and Table 19 respectively.

As to WebShop, we adopt the implementation of ReAct and use one example for in-context learning. The successful example without thoughts and with thoughts are shown in Table 20 and Table 21 respectively.

C.2.2. RAP

RAP(Hao et al., 2023) utilizes LLMs as policy and world model, and adopts tree-search planning methods to evaluate each possible action candidate. Trajectories interleaving between actions and observations like in Table 16, Table 18 and Table 20 are provided to LLMs to learn how to choose action as an policy and the dynamics of the environment as a world model.

Table 18. An example of successful trajectory for BabyAI-Text

```
Goal of the agent:  go to the green key
Observation:You see a wall 2 steps right, You see a wall 3 steps forward, You see a grey box 3 steps
left, You see a grey key 3 steps left and 1 step forward, You see a blue box 3 steps left and 2 steps
forward
Action:turn left
Observation:You see a wall 3 steps right, You see a blue key 3 steps left and 2 steps forward, You see
a green key 2 steps left and 1 step forward, You see a green ball 1 step left and 3 steps forward, You
see a grey box 3 steps forward, You see a grey key 1 step right and 3 steps forward, You see a blue box
2 steps right and 3 steps forward
Action:go forward
Observation:You see a wall 3 steps right, You see a blue key 3 steps left and 1 step forward, You see a
green key 2 steps left, You see a green ball 1 step left and 2 steps forward, You see a grey box 2 steps
forward, You see a grey key 1 step right and 2 steps forward, You see a blue box 2 steps right and 2
steps forward
Action:turn left
Observation:You see a wall 3 steps left, You see a green box 2 steps left and 3 steps forward, You see
a green box 2 steps left and 1 step forward, You see a green key 2 steps forward, You see a blue key 1
step right and 3 steps forward, You see a green ball 2 steps right and 1 step forward, You see a grey
box 2 steps right
Action:go forward
Observation:You see a green box 2 steps left and 2 steps forward, You see a green box 2 steps left, You
see a green key 1 step forward, You see a blue key 1 step right and 2 steps forward, You see a green
ball 2 steps right
```

Table 19. An example of annotated successful trajectory for BabyAI-Text in ReAct

```
Goal of the agent:  go to the green key
Observation:You see a wall 2 steps right, You see a wall 3 steps forward, You see a grey box 3 steps
left, You see a grey key 3 steps left and 1 step forward, You see a blue box 3 steps left and 2 steps
forward
Action:think:  To complete the task, I need to find a green key and go to it.
Observation:OK
Action:think:  I have not seen a green key.  Firstly, I need to find a green key.
Observation:OK
Action:turn left
Observation:You see a wall 3 steps right, You see a blue key 3 steps left and 2 steps forward, You see
a green key 2 steps left and 1 step forward, You see a green ball 1 step left and 3 steps forward, You
see a grey box 3 steps forward, You see a grey key 1 step right and 3 steps forward, You see a blue box
2 steps right and 3 steps forward
Action:think:  I have seen a green key 2 steps left and 1 step forward.  Next, I need to go to the green
key.
Observation:OK
Action:go forward
Observation:You see a wall 3 steps right, You see a blue key 3 steps left and 1 step forward, You see a
green key 2 steps left, You see a green ball 1 step left and 2 steps forward, You see a grey box 2 steps
forward, You see a grey key 1 step right and 2 steps forward, You see a blue box 2 steps right and 2
steps forward
Action:turn left
Observation:You see a wall 3 steps left, You see a green box 2 steps left and 3 steps forward, You see
a green box 2 steps left and 1 step forward, You see a green key 2 steps forward, You see a blue key 1
step right and 3 steps forward, You see a green ball 2 steps right and 1 step forward, You see a grey
box 2 steps right
Action:go forward
Observation:You see a green box 2 steps left and 2 steps forward, You see a green box 2 steps left, You
see a green key 1 step forward, You see a blue key 1 step right and 2 steps forward, You see a green
ball 2 steps right
```

Table 20. An example of successful trajectory for WebShop

```
Webshop
Instruction:
i would like a 3 ounce bottle of bright citrus deodorant for sensitive skin, and price lower than 50.00
dollars
[Search]

Action:  search[3 ounce bright citrus deodorant sensitive skin]
Observation:
[Back to Search]
Page 1 (Total results:  50)
[Next >]
[B078GWRC1J]
Bright Citrus Deodorant by Earth Mama | Natural and Safe for Sensitive Skin, Pregnancy and Breastfeeding,
Contains Organic Calendula 3-Ounce
$10.99
[B078GTKVXY]
Ginger Fresh Deodorant by Earth Mama | Natural and Safe for Sensitive Skin, Pregnancy and Breastfeeding,
Contains Organic Calendula 3-Ounce
$10.99
[B08KBVJ4XN]
Barrel and Oak - Aluminum-Free Deodorant, Deodorant for Men, Essential Oil-Based Scent, 24-Hour Odor
Protection, Cedar & Patchouli Blend, Gentle on Sensitive Skin (Mountain Sage, 2.7 oz, 2-Pack)
$15.95

Action:  click[B078GWRC1J]
Observation:
[Back to Search]
[< Prev]
scent [assorted scents][bright citrus][calming lavender][ginger fresh][simply non-scents]
size [travel set (4-pack)][3 ounce (pack of 1)][3-ounce (2-pack)]
Bright Citrus Deodorant by Earth Mama | Natural and Safe for Sensitive Skin, Pregnancy and Breastfeeding,
Contains Organic Calendula 3-Ounce
Price:  $10.99
Rating:  N.A.
[Description]
[Features]
[Reviews]
[Buy Now]

Action:  click[bright citrus]
Observation:  You have clicked bright citrus.

Action:  click[3 ounce (pack of 1)]
Observation:  You have clicked 3 ounce (pack of 1).

Action:  click[Buy Now]
Observation:  Your score (min 0.0, max 1.0):  1.0
```

Table 21. An example of successful trajectory for WebShop in ReAct

```
Webshop
Instruction:
i would like a 3 ounce bottle of bright citrus deodorant for sensitive skin, and price lower than 50.00
dollars
[Search]

Action:  search[3 ounce bright citrus deodorant sensitive skin]
Observation:
[Back to Search]
Page 1 (Total results:  50)
[Next >]
[B078GWRC1J]
Bright Citrus Deodorant by Earth Mama | Natural and Safe for Sensitive Skin, Pregnancy and Breastfeeding,
Contains Organic Calendula 3-Ounce
$10.99
[B078GTKVXY]
Ginger Fresh Deodorant by Earth Mama | Natural and Safe for Sensitive Skin, Pregnancy and Breastfeeding,
Contains Organic Calendula 3-Ounce
$10.99
[B08KBVJ4XN]
Barrel and Oak - Aluminum-Free Deodorant, Deodorant for Men, Essential Oil-Based Scent, 24-Hour Odor
Protection, Cedar & Patchouli Blend, Gentle on Sensitive Skin (Mountain Sage, 2.7 oz, 2-Pack)
$15.95

Action:  think[[B078GWRC1J] and [B078GTKVXY] are bright citrus deodorant less then 50 dollars.  I can
check [B078GWRC1J] first.]
Observation:  OK.

Action:  click[B078GWRC1J]
Observation:
[Back to Search]
[< Prev]
scent [assorted scents][bright citrus][calming lavender][ginger fresh][simply non-scents]
size [travel set (4-pack)][3 ounce (pack of 1)][3-ounce (2-pack)]
Bright Citrus Deodorant by Earth Mama | Natural and Safe for Sensitive Skin, Pregnancy and Breastfeeding,
Contains Organic Calendula 3-Ounce
Price:  $10.99
Rating:  N.A.
[Description]
[Features]
[Reviews]
[Buy Now]

Action:  think[For 3 ounce bottle of bright citrus deodorant for sensitive skin, the item has options
[bright citrus] and [3 ounce (pack of 1)] and seems good to buy.]
Observation:  OK.

Action:  click[bright citrus]
Observation:  You have clicked bright citrus.

Action:  click[3 ounce (pack of 1)]
Observation:  You have clicked 3 ounce (pack of 1).

Action:  click[Buy Now]
Observation:  Your score (min 0.0, max 1.0):  1.0
```

The assessment of each step is performed by a reward function, which can be the log probability of the action or self-evaluation given by LLMs, or based on some task-specific heuristics. In our implementation, we adopt the log probability of actions given by LLMs as the reward. For simplicity of implementation, we adopted a greedy approach to expand the tree, generating only one action at a time. More specifically, at each step, LLMs will sample some action candidates. For each action candidate, LLMs will generate a rollout trajectory until a maximum step or terminal state. The summation of log probabilities of all the actions on the rollout accessed by LLMs are used as Q value for each action candidate. The candidate with the highest Q value is chosen to be executed in the environment.

C.2.3. ICPI

Table 22. An example provided to critic in ICPI for ALFWorld

```
You are in the middle of a room.  Looking quickly around you, you see a cabinet 4, a cabinet 3, a
cabinet 2, a cabinet 1, a countertop 1, a garbagecan 1, a handtowelholder 2, a handtowelholder 1, a
sinkbasin 2, a sinkbasin 1, a toilet 1, a toiletpaperhanger 1, and a towelholder 1.
Your task is to:  put some spraybottle on toilet.
> go to cabinet 1
On the cabinet 1, you see a cloth 1, a soapbar 1, a soapbottle 1.
Reward:0
> go to cabinet 2
The cabinet 2 is closed.
Reward:0
> open cabinet 2
You open the cabinet 2.  The cabinet 2 is open.  In it, you see a candle 1, and a spraybottle 2.
Reward:0
> take spraybottle 2 from cabinet 2
You pick up the spraybottle 2 from the cabinet 2.
Reward:0
> go to toilet 1
On the toilet 1, you see a soapbottle 2.
Reward:0
> put spraybottle 2 in/on toilet 1
You put the spraybottle 2 in/on the toilet 1.
Reward:1
```

Table 23. An example provided to critic in ICPI for BabyAI-Text

```
Goal of the agent:  go to the green key
Observation:You see a wall 2 steps right, You see a wall 3 steps forward, You see a grey box 3 steps
left, You see a grey key 3 steps left and 1 step forward, You see a blue box 3 steps left and 2 steps
forward
Action:turn left
Observation:You see a wall 3 steps right, You see a blue key 3 steps left and 2 steps forward, You see
a green key 2 steps left and 1 step forward, You see a green ball 1 step left and 3 steps forward, You
see a grey box 3 steps forward, You see a grey key 1 step right and 3 steps forward, You see a blue box
2 steps right and 3 steps forward
Reward:0
Action:go forward
Observation:You see a wall 3 steps right, You see a blue key 3 steps left and 1 step forward, You see a
green key 2 steps left, You see a green ball 1 step left and 2 steps forward, You see a grey box 2 steps
forward, You see a grey key 1 step right and 2 steps forward, You see a blue box 2 steps right and 2
steps forward
Reward:0
Action:turn left
Observation:You see a wall 3 steps left, You see a green box 2 steps left and 3 steps forward, You see
a green box 2 steps left and 1 step forward, You see a green key 2 steps forward, You see a blue key 1
step right and 3 steps forward, You see a green ball 2 steps right and 1 step forward, You see a grey
box 2 steps right
Reward:0
Action:go forward
Observation:You see a green box 2 steps left and 2 steps forward, You see a green box 2 steps left, You
see a green key 1 step forward, You see a blue key 1 step right and 2 steps forward, You see a green
ball 2 steps right
Reward:1
```

ICPI (Brooks et al., 2024) proposes to implement policy iteration using LLMs through in-context learning. At each step, the policy in ICPI will sample some action candidates and the critic will compute the Q values for each action candidates. The action candidates with the highest Q values is chosen to be executed.

The policy is implemented using LLMs, and successful trajectories like in Table 16 and Table 18 are provided to it.

As to the critic, ICPI prompts LLMs to give the numerical reward for each step directly. Given the current history and an action candidate, the critic in ICPI will rollout a trajectory starting from the action candidate. Apart from predicting the observations, the critic will give the numerical reward for each step on the rollout trajectory. The (discounted) return on the rollout is treated as the Q value for the action candidate. For both ALFWorld and BabyAI-Text, we define the reward as 1 when the agent reaches the goal. All other steps will have a reward 0. The examples provided to the critic are like in Table 22 for ALFWorld and Table 23 for BabyAI-Text.

C.2.4. RAFA

Table 24. An example provided to critic in RAFA for ALFWorld

```
You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a
cabinet 2, a cabinet 1, a countertop 1, a garbagecan 1, a handtowelholder 2, a handtowelholder 1, a
sinkbasin 2, a sinkbasin 1, a toilet 1, a toiletpaperhanger 1, and a towelholder 1.
Your task is to: put some spraybottle on toilet.
> critic: My task requires two sub-goals in order: take a spraybottle and put the spraybottle on the
toilet. My current state satisfies zero of the two sub-goals. The value is 0/2=0.
> OK.
On the cabinet 1, you see a cloth 1, a soapbar 1, a soapbottle 1.
> OK.
The cabinet 2 is closed.
> OK.
You open the cabinet 2. The cabinet 2 is open. In it, you see a candle 1, and a spraybottle 2.
> OK.
You pick up the spraybottle 2 from the cabinet 2.
> critic: Now I take a spraybottle. My current state satisfies the first of the two sub-goals: take a
spraybottle. The value is 1/2=0.5.
> OK.
On the toilet 1, you see a soapbottle 2.
> OK.
You put the spraybottle 2 in/on the toilet 1.
> critic: Now I put the spraybottle on the toilet. My current state satisfies all the two sub-goals.
The value is 2/2=1.
```

Table 25. An example provided to critic in RAFA for BabyAI-Text

```
Goal of the agent: go to the green key
You see a wall 2 steps right, You see a wall 3 steps forward, You see a grey box 3 steps left, You see a
grey key 3 steps left and 1 step forward, You see a blue box 3 steps left and 2 steps forward
>critic: My task requires two sub-goals in order: find the green key, and go to the green key. My
current state satisfies zero of the two sub-goals. The value is 0/2=0.
>OK.
You see a wall 3 steps right, You see a blue key 3 steps left and 2 steps forward, You see a green key 2
steps left and 1 step forward, You see a green ball 1 step left and 3 steps forward, You see a grey box
3 steps forward, You see a grey key 1 step right and 3 steps forward, You see a blue box 2 steps right
and 3 steps forward
>critic: Now I find the green key. My current state satisfies the first of the two sub-goals: find
the green key. The value is 1/2=0.5.
>OK.
You see a wall 3 steps right, You see a blue key 3 steps left and 1 step forward, You see a green key 2
steps left, You see a green ball 1 step left and 2 steps forward, You see a grey box 2 steps forward,
You see a grey key 1 step right and 2 steps forward, You see a blue box 2 steps right and 2 steps
forward
>OK.
You see a wall 3 steps left, You see a green box 2 steps left and 3 steps forward, You see a green box 2
steps left and 1 step forward, You see a green key 2 steps forward, You see a blue key 1 step right and
3 steps forward, You see a green ball 2 steps right and 1 step forward, You see a grey box 2 steps right
>OK.
You see a green box 2 steps left and 2 steps forward, You see a green box 2 steps left, You see a green
key 1 step forward, You see a blue key 1 step right and 2 steps forward, You see a green ball 2 steps
right
>critic: Now I go to the green key. My current state satisfies all the two sub-goals. The value is
2/2=1.
```

The framework of RAFA (Liu et al., 2023) is also like RAP or ICPI. The main difference is how the action evaluation is conducted.

RAFA implements tree-search using LLM to evaluate each action candidate. Different from ICPI, RAFA uses the task completion progress as the value for each step. They have the LLMs decompose a goal into sub-goals, and use the completion status of the sub-goals after each step as the value for the step. RAFA evaluates the completion status of sub-goals based on the predicted observations. Examples provided in RAFA are like in Table 24 for ALFWorld and Table 25 for BabyAI-Text.

C.2.5. LATS

LATS (Zhou et al., 2024a) combines the reasoning, acting, and planning capabilities of LLMs with MCTS (Kocsis & Szepesvári, 2006) and external feedback mechanisms to enhance decision-making, achieving competitive results in web navigation. We adopt the official implementation to evaluate its performance on WebShop.

LATS cannot be directly applied to the other benchmarks we used for two main reasons. (1) LATS requires the ability to revert the agent to earlier states in the environment, which ALFWorld and BabyAI-Text do not support. LATS relies on model-free MCTS, using environment simulator as a world model, reverting simulators to earlier states during tree search. This limitation is also noted in their original paper (Page 9). (2) While it might be possible to modify these environments to make them reversible, it would create an unfair comparison. Our method and other baselines do not rely on simulators during reasoning in ALFWorld and BabyAI-Text, whereas, LATS would gain an advantage from this modification.

Nevertheless, we still attempted to adapt LATS for ALFWorld by using LLMs as world models, similar to our method, for a fair comparison. The results, presented in the Table 26, show that LATS fails in almost all tasks. This is because its tree search severely depends on the environment simulator for precise state transitions. With only LLM-based world models, the state transitions often deviate from the actual environments, due to LLMs' inherent hallucinations and partial observability of ALFWorld.

Table 26. Performance comparison of LAC (Ours) and LATS in ALFWorld.

	CodeLlama-7B	Gemma-7B	Llama-3-8B	Mistral-7B
LAC (Ours)	**0.79**	**0.84**	**0.78**	**0.79**
LATS	0.00	0.00	0.03	0.00

C.3. Implementation details of LAC, ablation studies, and variants

Table 27. An example of annotated successful trajectory for ALFWorld in LAC

```
You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a
cabinet 2, a cabinet 1, a countertop 1, a garbagecan 1, a handtowelholder 2, a handtowelholder 1, a
sinkbasin 2, a sinkbasin 1, a toilet 1, a toiletpaperhanger 1, and a towelholder 1.
Your task is to: put some spraybottle on toilet.
> think: To solve the task, I need to find and take a sparybottle, then put it on toilet.
OK.
==> critic: The thought is reasonable. This step is GOOD.
> think: First I need to find a spraybottle. A spraybottle is more likely to appear in cabinet (1-4),
countertop (1), toilet (1), sinkbasin (1-2), garbagecan (1). I can check one by one, starting with
cabinet 1.
OK.
==> critic: I need to find spraybottle firstly. The thought is reasonable. This step is GOOD.
> go to cabinet 1
On the cabinet 1, you see a cloth 1, a soapbar 1, a soapbottle 1.
==> critic: I have not seen spraybottle in this place. This step is UNKNOWN.
> go to cabinet 2
The cabinet 2 is closed.
==> critic: The cabinet 2 is closed. This step is UNKNOWN.
> open cabinet 2
You open the cabinet 2. The cabinet 2 is open. In it, you see a candle 1, and a spraybottle 2.
==> critic: I have seen spraybottle 2 in this place. This step is GOOD.
> think: Now I find a spraybottle (2). Next, I need to take it.
OK.
==> critic: I can take the spraybottle after finding it. The thought is reasonable. This step is
GOOD.
> take spraybottle 2 from cabinet 2
You pick up the spraybottle 2 from the cabinet 2.
==> critic: I have taken spraybottle 2 in this place. This step is GOOD.
> think: Now I take a spraybottle (2). Next, I need to put it in/on toilet 1.
OK.
==> critic: I can put the spraybottle in/on toilet after taking it. The thought is reasonable. This
step is GOOD.
> go to toilet 1
On the toilet 1, you see a soapbottle 2.
==> critic: I should go to toilet if I want to put the spraybottle on it. This step is GOOD.
> put spraybottle 2 in/on toilet 1
You put the spraybottle 2 in/on the toilet 1.
==> critic: I have put the spraybottle 2 in/on the toilet 1. This step is GOOD.
```

Table 28. An example of annotated successful trajectory for BabyAI-Text in LAC

```
Goal of the agent:  go to the green key
Observation:You see a wall 2 steps right, You see a wall 3 steps forward, You see a critic box 3 steps
left, You see a critic key 3 steps left and 1 step forward, You see a blue box 3 steps left and 2 steps
forward
Action:think:  To complete the task, I need to find a green key and go to it.
Observation:OK
Critic:The thought is reasonable.  This step is GOOD.
Action:think:  I have not seen a green key.  Firstly, I need to find a green key.
Observation:OK
Critic:I need to find a green key firstly.  The thought is reasonable.  This step is GOOD.
Action:turn left
Observation:You see a wall 3 steps right, You see a blue key 3 steps left and 2 steps forward, You see a
green key 2 steps left and 1 step forward, You see a green ball 1 step left and 3 steps forward, You see
a critic box 3 steps forward, You see a critic key 1 step right and 3 steps forward, You see a blue box
2 steps right and 3 steps forward
Critic:I have turned left and seen a green key 2 steps left and 1 step forward.  This step is GOOD.
Action:think:  I have seen a green key 2 steps left and 1 step forward.  Next, I need to go to the green
key.
Observation:OK
Critic:I can go to the green key after seeing it.  The thought is reasonable.  This step is GOOD.
Action:go forward
Observation:You see a wall 3 steps right, You see a blue key 3 steps left and 1 step forward, You see
a green key 2 steps left, You see a green ball 1 step left and 2 steps forward, You see a critic box 2
steps forward, You see a critic key 1 step right and 2 steps forward, You see a blue box 2 steps right
and 2 steps forward
Critic:I have gone forward.  The green key is 2 steps left.  This step is UNKNOWN.
Action:turn left
Observation:You see a wall 3 steps left, You see a green box 2 steps left and 3 steps forward, You see
a green box 2 steps left and 1 step forward, You see a green key 2 steps forward, You see a blue key 1
step right and 3 steps forward, You see a green ball 2 steps right and 1 step forward, You see a critic
box 2 steps right
Critic:I have turned left.  The green key is 2 steps forward.  This step is UNKNOWN.
Action:go forward
Observation:You see a green box 2 steps left and 2 steps forward, You see a green box 2 steps left, You
see a green key 1 step forward, You see a blue key 1 step right and 2 steps forward, You see a green
ball 2 steps right
Critic:I have gone forward.  The green key is 1 step forward.  This step is GOOD.
```

We build our method based on ReAct (Yao et al., 2023). Specifically, we adopt the in-context examples from ReAct and annotate reflections for those trajectories. For each task type in ALFWorld, we annotate two trajectories from ReAct as successful examples to teach LLMs how to complete this type of tasks. For BabyAI-Text, we annotate two trajectories with thoughts like in ReAct, as well as reflections, for each task type. As to WebShop, we annotate the example from ReAct with reflections, and we remove the thoughts annotated in ReAct because we found better results can be achieved without thoughts. The annotated trajectories for ALFWorld, BabyAI-Text and WebShop are shown in Table 27, Table 28, and Table 29 respectively. The reflections we annotated are labeled in light orange. We also use ReAct algorithm to collect some failed trajectories from the training set for each task type of ALFWorld and BabyAI-Text. The collected failed trajectories are also annotated with reflections, aiming to teach LLMs to distinguish between steps that are beneficial, harmful, or indeterminate for achieving a goal.

We annotate the reflections using the following criteria:

(1) If an action on a trajectory is necessary for achieving the final goal, the following reflections will end with "This step is GOOD." to indicate the completion of an essential step, such as action "take spraybottle 2 from cabinet 2" for the goal "put some spraybottle on toilet".

(2) If an action is crucial for obtaining essential information to complete a goal, the subsequent reflections will end with "This step is GOOD.", such as action "open cabinet 2" in Table 27 which finds the target object.

(3) If an action is definitely useless or incorrect for achieving a goal, the following reflections will end with "This step is BAD." to indicate this action is undesirable, such as action "take cloth 1 from cabinet 1" for the task in Table 27.

(4) If an action can not be evaluated as good or bad from the history, the following reflections will end with "This step is UNKNOWN.", such as action "go to cabinet 1" or "go to cabinet 2" in Table 27.

All the annotated successful and failed trajectories are used to construct fine-tuning examples to fine-tune LLMs to generate better reflections.

Table 29. An example of successful trajectory for WebShop in LAC

```
Webshop
Instruction:
i would like a 3 ounce bottle of bright citrus deodorant for sensitive skin, and price lower than 50.00
dollars
[Search]

Action: search[3 ounce bright citrus deodorant sensitive skin]
Observation:
[Back to Search]
Page 1 (Total results: 50)
[Next >]
[B078GWRC1J]
Bright Citrus Deodorant by Earth Mama | Natural and Safe for Sensitive Skin, Pregnancy and Breastfeeding,
Contains Organic Calendula 3-Ounce
$10.99
[B078GTKVXY]
Ginger Fresh Deodorant by Earth Mama | Natural and Safe for Sensitive Skin, Pregnancy and Breastfeeding,
Contains Organic Calendula 3-Ounce
$10.99
[B08KBVJ4XN]
Barrel and Oak - Aluminum-Free Deodorant, Deodorant for Men, Essential Oil-Based Scent, 24-Hour Odor
Protection, Cedar & Patchouli Blend, Gentle on Sensitive Skin (Mountain Sage, 2.7 oz, 2-Pack)
$15.95
Critic: The item required in the instruction is 'bright citrus deodorant for sensitive skin'. I have
searched '3 ounce bright citrus deodorant sensitive skin', which is similar to the required item. I
also see that [B078GWRC1J] is bright citrus deodorant. This step is GOOD.

Action: click[B078GWRC1J]
Observation:
[Back to Search]
[< Prev]
scent [assorted scents][bright citrus][calming lavender][ginger fresh][simply non-scents]
size [travel set (4-pack)][3 ounce (pack of 1)][3-ounce (2-pack)]
Bright Citrus Deodorant by Earth Mama | Natural and Safe for Sensitive Skin, Pregnancy and Breastfeeding,
Contains Organic Calendula 3-Ounce
Price: $10.99
Rating: N.A.
[Description]
[Features]
[Reviews]
[Buy Now]
Critic: I have clicked [B078GWRC1J] and found that this item is 'bright citrus deodorant' and it can
satisfy the attributes in instruction, which is '3 ounce' and 'price lower than 50.00 dollars'. This
step is GOOD.

Action: click[bright citrus]
Observation: You have clicked bright citrus.
Critic: I have clicked [bright cirtrus] to choose the option that satisfies the instruction. This step
is GOOD.

Action: click[3 ounce (pack of 1)]
Observation: You have clicked 3 ounce (pack of 1).
Critic: I have clicked [3 ounce (pack of 1)] to choose the option that satisfies the instruction. This
step is GOOD.

Action: click[Buy Now]
Observation: Your score (min 0.0, max 1.0): 1.0
Critic: I have bought the item that satisfies the instruction. This step is GOOD.
```

Considering the computational cost of fully finetuning of LLMs, we use LoRA (Hu et al., 2021) to finetune our models. In ALFWorld, with two successful trajectories and one failed trajectory for each task type, we have 485 (input,output) pairs in total of six task types to finetune models. In BabyAI-Text, the number of finetuning (input,output) pairs is 418. We finetune models for 1,000 steps with learning rate 2.5e-5 and batch size 2. We use A100 GPU with 80GB memory to fine-tune our model. With just about 400-500 (input,output) pairs and 1,000 fine-tuning steps, we can complete the fine-tuning process within one and a half hours.

During testing, the fine-tuned models are used to generate reflections after executing an action in the environment, as well as to forecast the potential outcomes of each action candidate.

The number of candidate actions n is a hyperparameter, set to 5 in our experiments, that is, we evaluate the top 5 candidate actions sampled from π_{LLM} per state. It is worth noting that sampling top candidate actions introduces a nonzero probability of missing the true argmax action, especially when distinctions among candidate actions are subtle. However, this choice is primarily driven by computational practicality: explicitly computing or evaluating the full action distribution for large or open-ended action spaces common in LLM-based decision-making is typically intractable. Empirically, we find that generating a small subset of candidate actions from a strong LLM prior is often sufficient to include promising actions, thus making the trade-off between computational efficiency and accuracy acceptable.

After sampling action candidates, we use the fine-tuned model to predict future outcomes for each action candidate. The model needs to predict the possible observation and generate reflections for each predicted step. We set the maximum prediction step as 4, the model will continue the prediction until it generates a reflection ending with "This step is GOOD." or "This step is BAD", or when it reaches the maximum prediction step.

For the optimization of π_{LLM}, we solve an optimization problem in Equation (4) with a hyper-parameter α, which balances the generating probabilities of π_{LLM} and the values given by $\mathcal{Q}_{\text{LLM}}$. For ALFWorld, we set α as 1, which yields superb performance over baselines. For BabyAI-Text, we conduct a grid-search over {1/2,1,2,5,10} for α, finding that different LLMs will have best performance with different α. The results can be seen in Table 12. For WebShop, we also conduct a grid search over {1/10,1,10} to find the best α. Equation (4) is a weighted combination of the original policy and the action evaluation values. It updates the distribution by adjusting the probabilities of the top candidate actions, while leaving the probabilities of other actions unchanged.

We set the maximum horizon length to 40 for ALFWorld, 30 for BabyAI-Text, and 15 for WebShop. If the agent has not reached the final goal after the maximum steps, this episode will be marked as failure.

We use A100 GPU with 80GB memory to evaluate our method. For `LAC`, the execution time for ALFWorld is about 10 hours for 134 tasks using single A100 GPU. And for BabyAI-Text, the execution time can be varied for different task types, ranging from 4 to 10 hours for 50 tasks using one A100 GPU. For WebShop, 100 tasks will be completed within 3 or 4 hours. The GPU memory usage may range from 15GB to over 70GB during the interaction according to the length of inputs to LLMs.

We compare our method with all the aforementioned baselines, demonstrating the effectiveness of our method on decision-making tasks with both high-level actions and low-level actions, even with potentially infinite actions. To demonstrate the effectiveness of each component in our method, we conduct ablation studies on each component. We removes the reflections from `LAC` as well as the integration during pre-action-generation phase. This variant is called `LAC` *w/o* reflection. We also evaluate the role of $\mathcal{Q}_{\text{LLM}}$ by removing it from `LAC` as well as the integration during post-action-generation phase. This variant is called `LAC` *w/o* critic. We also demonstrate the role of the action prior given by LLM policy by using only critic $\mathcal{Q}_{\text{LLM}}$ for decision-making. We call this variant as *critic-only*. The execution time of those variants during evaluation can be varied according to its performance because a method having poor performance typically will cost more time to execute. On ALFWorld, it may be 10-20 hours. The comparisons between those variants are shown in Figure 4.

We found that each component in `LAC` is crucial for the superb performance. Removing some components may lead to wrong choice of action candidates. Such an example is shown in Table 30. `LAC` can complete this task successfully, while eliminating some components in `LAC` will lead to failure. The comparison is shown in Figure 10.

Table 30. An example of successful trajectory in test environment in BabyAI-Text given by LAC

Goal of the agent:go to the green ball
Observation:You see a wall 5 steps forward, You see a wall 2 steps left, You see a yellow ball 1 step left and 4 steps forward, You see a blue ball 1 step left and 3 steps forward, You see a red ball 1 step left and 1 step forward, You see a grey key 2 steps right and 2 steps forward, You see a green key 2 steps right and 1 step forward, You see a blue key 3 steps right and 1 step forward
Action:think:To complete the task, I need to find a green ball and go to it.
Observation:OK
Critic:This thought is reasonable. This step is GOOD.
Action:think:I have seen a green ball 1 step left and 4 steps forward. Next, I need to go to the green ball.
Observation:OK
Critic:This thought is reasonable. I can go to the green ball after seeing it. This step is GOOD.
Action:turn right
Observation:You see a wall 5 steps forward, You see a wall 2 steps right, You see a green ball 3 steps left and 4 steps forward, You see a grey key 2 steps left and 2 steps forward, You see a blue key 1 step left and 3 steps forward, You see a green key 1 step left and 2 steps forward, You see a blue ball 1 step right and 1 step forward
Critic:I have turned right. The green ball is 3 steps left and 4 steps forward. This step is UNKNOWN.
Action:go forward
Observation:You see a wall 4 steps forward, You see a green ball 3 steps left and 3 steps forward, You see a grey key 2 steps left and 1 step forward, You see a blue key 1 step left and 2 steps forward, You see a green key 1 step left and 1 step forward, You see a blue ball 1 step right
Critic:I have gone forward. The green ball is 3 steps left and 3 steps forward. This step is UNKNOWN.
Action:go forward
Observation:You see a wall 3 steps forward, You see a wall 2 steps right, You see a green ball 3 steps left and 2 steps forward, You see a grey key 2 steps left, You see a blue key 1 step left and 1 step forward, You see a green key 1 step left
Critic:I have gone forward. The green ball is 3 steps left and 2 steps forward. This step is UNKNOWN.
Action:go forward
Observation:You see a wall 2 steps forward, You see a wall 2 steps right, You see a green ball 3 steps left and 1 step forward, You see a blue key 1 step left
Critic:But I have gone forward. The green ball is 3 steps left and 1 step forward. This step is UNKNOWN.
Action:go forward
Observation:You see a wall 1 step forward, You see a wall 2 steps right, You see a green ball 3 steps left
Critic:But I have gone forward. The green ball is 3 steps left. This step is UNKNOWN.
Action:turn left
Observation:You see a wall 1 step right, You see a grey key 2 steps left and 2 steps forward, You see a green key 2 steps left and 1 step forward, You see a blue key 1 step left and 1 step forward, You see a red ball 4 steps forward, You see a green ball 3 steps forward
Critic:But I have turned left. The green ball is 3 steps forward. This step is UNKNOWN.
Action:go forward
Observation:You see a wall 1 step right, You see a grey key 2 steps left and 1 step forward, You see a green key 2 steps left, You see a blue key 1 step left, You see a red ball 3 steps forward, You see a green ball 2 steps forward
Critic:But I have gone forward. The green ball is 2 steps forward. This step is UNKNOWN.
Action:go forward
Observation:You see a wall 1 step right, You see a grey key 2 steps left, You see a red ball 2 steps forward, You see a green ball 1 step forward
Critic:But I have gone forward. The green ball is 1 step forward. This step is GOOD.