Reason for Future, Act for Now: A Principled Architecture for Autonomous LLM Agents

Zhihan Liu [*1] **Hao Hu** [*2] **Shenao Zhang** [*1] **Hongyi Guo** [1] **Shuqi Ke** [3] **Boyi Liu** [1] **Zhaoran Wang** [1]

Abstract

Large language models (LLMs) demonstrate impressive reasoning abilities, but translating reasoning into actions in the real world remains challenging. In particular, it is unclear how to complete a given task provably within a minimum number of interactions with the external environment, e.g., through an internal mechanism of reasoning. To this end, we propose the first framework with provable regret guarantees to orchestrate reasoning and acting, which we call "reason for future, act for now" (`RAFA`). Specifically, we design a prompt template for reasoning that learns from the memory buffer and plans a future trajectory over a long horizon ("reason for future"). At each step, the LLM agent takes the initial action of the planned trajectory ("act for now"), stores the collected feedback in the memory buffer, and reinvokes the reasoning routine to replan the future trajectory from the new state. The key idea is to cast reasoning in LLMs as learning and planning in Bayesian adaptive Markov decision processes (MDPs). Correspondingly, we prompt LLMs with the memory buffer to estimate the unknown environment (learning) and generate an optimal trajectory for multiple future steps that maximize a value function (planning). The learning and planning subroutines are performed in an "in-context" manner to emulate the actor-critic update for MDPs. Our theoretical analysis establishes a $\sqrt{T}$ regret, while our experimental validation demonstrates superior empirical performance. Here, T denotes the number of online interactions.

[*]Equal contribution [1]Northwestern University, USA [2]Institute for Interdisciplinary Information Sciences, Tsinghua University, China [3]The Chinese University of Hong Kong, China. Correspondence to: Zhihan Liu <ZhihanLiu2027@u.northwestern.edu>, Zhaoran Wang <zhaoranwang@gmail.com>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

1. Introduction

Large language models (LLMs) exhibit remarkable reasoning abilities, which open a new avenue for agents to interact with the real world autonomously. However, turning reasoning into actions remains challenging. Specifically, although LLMs are equipped with the prior knowledge obtained through pretraining, it is stateless in nature and ungrounded in the real world, which makes the resulting action suboptimal. To bridge the reasoning-acting gap, we aim to design an internal mechanism of reasoning on top of LLMs, which optimizes actions iteratively by incorporating feedback from the external environment. In particular, we focus on the sample efficiency of autonomous LLM agents in interactive decision-making tasks, which plays a key role in their practical adoption, especially when interactions are costly and risky. Our primary goal is to enable agents to complete a given task in a guaranteed manner through reasoning within a minimum number of interactions with the external environment.

Reinforcement learning (RL) is a well-studied paradigm for improving actions by collecting feedback. However, to tailor existing RL techniques for autonomous LLM agents, we lack a rigorous mapping between RL and LLMs, which leads to various conceptual discrepancies. For example, RL operates in a numerical system, where rewards and transitions are defined by scalars and probabilities. In comparison, the inputs and outputs of LLMs are described by tokens in a linguistic system. As another example, LLMs are trained on a general-purpose corpus and remain fixed throughout the interactive process. In contrast, RL trains actors and critics via parameter updates on the collected feedback iteratively. Thus, it appears inappropriate to treat LLMs as actors or critics under the RL framework, although all of them are parameterized by deep neural networks. Moreover, it remains unclear what reasoning with LLMs means under the RL framework, e.g., what are the inputs and outputs of a reasoning routine and how reasoning should be coordinated with acting. Such conceptual discrepancies prevent us from establishing a principled framework beyond borrowing the "trial and error" concept from RL straightforwardly and make it difficult to establish the theoretical guarantee.

To address such conceptual discrepancies, we formalize reasoning and acting with LLMs under a Bayesian adaptive Markov decision process (MDP) framework, where the latent variable of interest is the unknown environment. The starting point is to cast the full history of states (of the external environment), actions, rewards, and their linguistic summaries in the memory buffer as the information state of Bayesian adaptive MDPs. Throughout the interactive process, the information state accumulates a growing collection of feedback from the external environment, which is mapped to an optimized action at each step by an internal mechanism of reasoning. As detailed below, we construct the reasoning routine through two key subroutines, namely learning and planning, which are instantiated by LLMs with specially designed prompts. **(a)** The learning subroutine forms an estimate of the external environment given the memory buffer, where LLMs are prompted to infer the transition and reward models (model) or/and the value function (critic). **(b)** The planning subroutine generates an optimal policy (actor) or trajectory for multiple future steps, which maximizes the value function (up to a certain error). Depending on the specific configuration of the state and action spaces (continuous versus discrete) and the transition and reward models (stochastic versus deterministic), the planning subroutine emulates the value iteration algorithm, the random shooting algorithm, or the Monte-Carlo tree-search algorithm.

Although LLMs remain fixed throughout the interactive process, we can reduce their estimation uncertainty by prompting the growing collection of feedback from the external environment as contexts, which is verified both theoretically and empirically in this paper. From the perspective of Bayesian adaptive MDPs, LLMs can be considered as some functional of the posterior of the environment (for example, Bayesian model averaging (Wasserman, 2000)), hence the estimation uncertainty is reduced with increasing information via interactions. For several tasks, we demonstrate that LLMs can make a more precise prediction when prompted with more data as contexts. Hence, LLMs can play a similar role of model estimators in the design of online RL algorithms for interactions. We improve the accuracy of LLMs by simply adding the new feedback to the memory buffer as contexts, instead of performing explicit parameter updates (such as gradient descent) on deep neural networks as in existing RL methods.

We conclude our contributions in this paper from three perspectives. **(a)** We establish the LLM-RL correspondence and design a principled framework `RAFA` for orchestrating the reasoning and acting of LLMs. **(b)** Our empirical validation shows that `RAFA` outperforms various existing frameworks in interactive decision-making tasks, including ALFWorld, BlocksWorld, Game of 24, and a new benchmark based on Tic-Tac-Toe. **(c)** Our theoretical analysis proves that `RAFA` achieves a $\sqrt{T}$ regret, explaining why `RAFA` demonstrates strong empirical performance. We also provide two provably efficient variants of `RAFA` to implement efficient exploration for more complex tasks.

1.1. Literature

Due to the page limit, we defer the detailed discussion on large language model (LLM), in-context learning (ICL), and reinforcement learning (RL) under a Bayesian framework to Appendix A.

Reasoning with LLM. We build on a recent line of work that develops various prompting schemes to improve the reasoning performance of LLMs. "Chain of thoughts" ("CoT") (Wei et al., 2022) decomposes a challenging problem into several reasoning stages and guides LLMs to solve them one by one. As generalizations, "tree of thoughts" (Yao et al., 2023a), "graph of thoughts" (Yao et al., 2023b), "algorithm of thoughts" (Sel et al., 2023), and "cumulative reasoning" (Zhang et al., 2023a) provide different graph-search schemes to guide LLMs. See also (Wang et al., 2022a; Creswell et al., 2022; Creswell & Shanahan, 2022; Guo et al., 2024; Zhang et al., 2024). Also, "reasoning via planning" ("RAP") (Hao et al., 2023) emulates the Monte-Carlo tree-search (MCTS) algorithm to reduce the search complexity. (Pouplin et al., 2024) improve LLM reasoning process with MCTS, and formulated the reasoning process as an MDP and (Sun et al., 2023a) use offline inverse RL to optimize the prompts for arithmetic problems. For embodied LLM agents, (Huang et al., 2022a) propose to decompose a complex task into multiple executable steps. Most of them focus on general reasoning tasks, e.g., solving a mathematical or logic puzzle, where LLMs generate a detailed trace (trajectory) of arguments through an internal mechanism to reach a final answer. Here, LLMs play the same role as the planning subroutine in `RAFA`. In contrast, we focus on interactive decision-making tasks, where autonomous LLM agents collect feedback from the external environment to optimize actions iteratively. In particular, we aim to complete a given task within a minimum number of interactions with the external environment. To this end, it is essential to operate three interleaved modules, namely learning, planning, and acting, in a closed loop. While it is feasible to incorporate existing graph-search or MCTS schemes as the planning subroutine for generating trajectories, our core contribution is a principled framework that executes a selected subset of the planned trajectory to collect feedback ("act for now") and replans an improved trajectory from the new state by learning from feedback ("reason for future"). From an RL perspective, existing graph-search or MCTS schemes are analogous to an open-loop method, e.g., motion planning or trajectory optimization (Betts, 1998), which does not involve interactions with the external environment. To integrate them into a closed-loop approach, e.g., model predictive control (Rawlings, 2000),

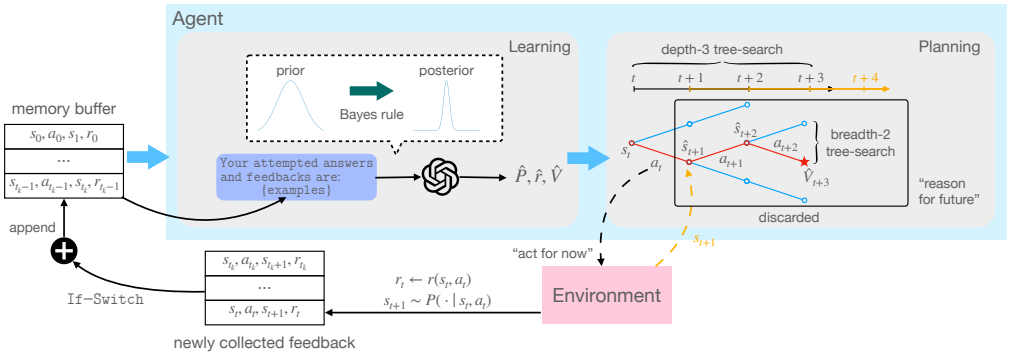


Figure 1. Illustration of the RAFA ("reason for future, act for now") framework.

one has to specify how to act given the planned trajectory and when to reinvoke the reasoning (learning and planning) routine, which is the key technique of RAFA. Another recent line of work tackles more complex tasks by allowing LLMs to access various additional modules, e.g., tools, programs, and other learning algorithms (Ahn et al., 2022; Shen et al., 2023; Lu et al., 2023; Liu et al., 2023a; Cai et al., 2023), or by finetuning LLMs on the feedback (Zelikman et al., 2022; Li et al., 2022; Paul et al., 2023; Sun, 2023).

Acting (and Reasoning) with LLM. We build on a recent line of work that develops various closed-loop frameworks for interacting with the external environment. "Inner monologue" (Huang et al., 2022b) and "ReAct" (Yao et al., 2022) combine reasoning and acting to refine each other for the first time. In comparison, RAFA provides a specific schedule for orchestrating reasoning and acting (as discussed above). As generalizations, "Reflexion" (Shinn et al., 2023) enables autonomous LLM agents to revise the current action of a pregenerated trajectory by learning from feedback, especially when they make mistakes. See also (Kim et al., 2023). However, making a local revision to the pre-generated trajectory is myopic because it fails to consider the long-term consequences of actions. Consequently, the obtained policy may get trapped by a local optimum. From an RL perspective, "Reflexion" (Shinn et al., 2023) is an oversimplified version of RAFA, where the planning subroutine revises the current action to maximize the reward function ("reason for now") instead of planning multiple future steps to maximize the value function ("reason for future"), which measures the expected cumulative future reward. To remedy this issue, "AdaPlanner" (Sun et al., 2023b) regenerates the whole trajectory at each step, which yields a global improvement. See also (Wang et al., 2023b). However, the reasoning routine of "AdaPlanner" requires a handcrafted set of programs to reject suboptimal candidate trajectories. Without the domain knowledge of the current task, the regenerated trajectory is not necessarily optimal, i.e., maximizing the value function (up to a certain error). In contrast, the reasoning routine of RAFA is designed following the principled approach in RL. In particular, the learning subroutine infers the transition and reward models (model) or/and the value function (critic), while the planning subroutine emulates the value iteration algorithm, the random shooting algorithm, or the MCTS algorithm, none of which use any domain knowledge. RAFA also achieves provable sample efficiency guarantees for the first time and outperforms those existing frameworks empirically.

Closed-Loop Mechanisms	No Parameter Update	Theoretical Guarantee
RAFA	✓	✓
Model-Based Deep RL	✗	✓
Model Predictive Control	✗	✓
Thompson Sampling	✗	✓
"React", "Reflexion", and "Adaplanner"	✓	✗

Table 1. Comparison between RAFA and other mechanisms.

1.2. Notations

We provide a table of notations in Appendix C.1.

2. Bridging LLM and RL

Interaction Protocol. We use Bayesian adaptive Markov decision processes (MDPs) (Ghavamzadeh et al., 2015) to model how autonomous LLM agents interact with the external environment. We consider an infinite-horizon MDP $M = (\mathcal{S}, \mathcal{A}, P, r, \rho, \gamma, \mathbb{P}_0)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$ is the transition kernel, $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, ρ is the initial distribution of states, $\gamma \in (0, 1)$ is the discount factor, and $\mathbb{P}_0$ is the prior distribution of the transition kernel and the reward function. Here, P gives the probability distribution of the next state given the current state and action, while r is assumed to be deterministic without loss of generality. For notational simplicity, we parameterize P and r by a shared parameter $\theta \in \Theta$ and denote them as P_θ and r_θ. At the beginning of the interaction, the data-generating parameter $\theta^\star$ is sampled from the prior $\mathbb{P}_0$. At the t-th step during the interaction, the LLM agent receives

a state $s_t \in \mathcal{S}$, takes an action $a_t \in \mathcal{A}$ following the current policy $\pi_t : \mathcal{S} \mapsto \mathcal{A}$, and receives a reward $r_t = r_{\theta^\star}(s_t, a_t)$. Subsequently, the external environment transits to the next state $s_{t+1} \sim P_{\theta^\star}(\cdot \mid s_t, a_t)$, while the LLM agent computes the updated policy π_{t+1} through an internal mechanism of reasoning (as discussed below). Note that $\mathcal{S}$ and $\mathcal{A}$ are represented by tokens in a linguistic system. Here, $\pi \in \Pi$ is assumed to be deterministic without loss of generality, where Π is the feasible set of policies.

Value Function. For a policy π and a parameter θ of the transition and reward models, we define the state-value and action-value functions

$$V_\theta^\pi(s) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r_\theta(s_t, a_t) \Big| \, s_0 = s\Big],$$
$$Q_\theta^\pi(s,a) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r_\theta(s_t, a_t) \Big| \, s_0 = s, a_0 = a\Big], \quad (2.1)$$

where $\mathbb{E}$ is taken with respect to $a_t = \pi(s_t)$ and $s_{t+1} \sim P_\theta(\cdot \mid s_t, a_t)$ for all $t \geq 0$. In other words, V_θ^π (and Q_θ^π) gives the expected cumulative future reward from the current state s (and action a).

We define the optimal policy $\pi_\theta^\star$ with respect to a given parameter θ as $\pi_\theta^\star = \operatorname{argmax}_{a \in \mathcal{A}} Q_\theta^\star$, where $Q_\theta^\star$ is the fixed point of the Bellman optimality equation (Sutton & Barto, 2018). See the detailed discussions in Appendix C.2.

Posterior, Entropy, and Information Gain. By Bayes' rule, the posterior of $\theta^\star$ given any in-context dataset $\mathcal{D}$ is

$$\mathbb{P}_{\text{post}}(\theta \mid \mathcal{D}) \propto \mathbb{P}_0(\theta) L(\mathcal{D} \mid \theta), \quad (2.2)$$

where we denote by $L(\mathcal{D} \mid \theta)$ the likelihood of $\mathcal{D}$ conditioned on θ. We define the random variable $\xi_{(s,a)}$ as the pair of the next state and the current reward (s', r) given the query state-action pair (s, a). Given any in-context dataset $\mathcal{D}$ and query state-action pair (s, a), the posterior of $\xi_{(s,a)}$ can be specified as

$$\mathbb{P}_{\text{post}}(\xi_{(s,a)} \mid \mathcal{D}, s, a) = \mathbb{E}_{\theta \sim \mathbb{P}_{\text{post}}(\cdot \mid \mathcal{D})}\big[P_\theta(s' \mid s, a) \cdot \mathbf{1}(r = r_\theta(s,a))\big], \quad (2.3)$$

where we use Bayes' rule and the fact that the query state-action pair (s, a) is conditionally independent of $\theta^\star$ given $\mathcal{D}$. To characterize the uncertainty of $\theta^\star$ conditioned on $\mathcal{D}$, we define the posterior entropy $H(\theta \mid \mathcal{D})$ as

$$H(\theta \mid \mathcal{D}) = \mathbb{E}_{\theta \sim \mathbb{P}_{\text{post}}(\cdot \mid \mathcal{D})}\big[-\log\big(p_{\text{post}}(\theta \mid \mathcal{D})\big)\big], \quad (2.4)$$

where p_{post} is the probability mass (or density) function of $\mathbb{P}_{\text{post}}$. High posterior entropy $H(\theta \mid \mathcal{D})$ means high uncertainty of $\theta^\star$, which suggests that it is hard for the agent to make a precise prediction given $\mathcal{D}$. We also define the information gain $I(\theta; \xi \mid \mathcal{D})$ as $H(\theta \mid \mathcal{D}) - H(\theta \mid \mathcal{D}, \xi)$, which characterizes how much information $\xi_{(s,a)}$ carries to reduce the uncertainty of $\theta^\star$ conditioned on $\mathcal{D}$.

Sample Efficiency. As the performance metric, we define the Bayesian regret after T steps of interactions,

$$\mathfrak{R}(T) = \mathbb{E}\Big[\sum_{t=0}^{T-1} V_{\theta^\star}^{\pi^\star}(s_t) - V_{\theta^\star}^{\pi_t}(s_t)\Big], \quad (2.5)$$

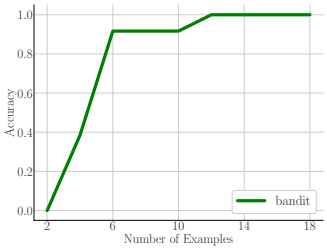
where $\pi^\star = \pi_{\theta^\star}^\star$, $\mathbb{E}$ is taken with respect to the prior distribution $\mathbb{P}_0$ of $\theta^\star$, the stochastic outcome of s_t, and the iterative update of π_t, which involves states, actions, and rewards until the t-th step, i.e., the full history $\mathcal{D}_t = \{(s_i, a_i, s_{i+1}, r_i)\}_{i=0}^{t-1}$. We aim to design a sample-efficient agent that satisfies $\mathfrak{R}(T) = o(T)$, i.e., the Bayesian regret is sublinear in the total number of interactions T.

What Reasoning Means and Role of LLM. To bridge LLM mechanisms with online RL algorithms, we claim that LLMs can play a similar role of model estimators in the design of online RL algorithms for interactions, which is one aspect of In-Context Learning (ICL) ability of LLMs.

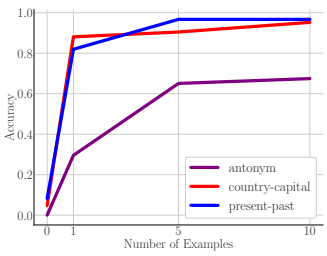
Claim 2.1. *LLMs provide a more accurate estimate for the environment with more feedback from online interactions.*

In Proposition C.4 in Section 5, we prove that LLMs with posterior alignment perform Bayesian model averaging (BMA). This theoretical result supports Claim 2.1, as the estimation uncertainty of BMA is reduced given more feedback from the interactions with the environment (Wasserman, 2000). We also provide empirical evidence on three tasks for Claim 2.1 as follows.

(a) Information Bandit. The goal of our 100-arm bandit experiment is to find the arm with the highest reward. There is an informative arm whose reward is the index of the best arm. We prompt the LLM (`gpt-4`) to pull the arm by providing it with the historical data of several bandit instances that share the same informative arm. It can be observed from the right figure that the LLM can learn the best arm with only 6 examples, and is thus an effective reward estimator.

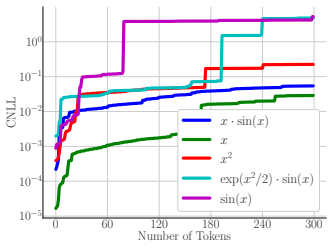


(b) Concept Learning. We evaluate LLMs (`Llama 2-7B`) in three tasks (Todd et al., 2023) with hidden concepts: (1) Antonym: Generate the word with



the opposite meaning given an input word; (2) Country-Capital: Generate the capital city of a given country; and (3) Present-Past: Generate the verb's past inflection given a verb in the present tense. We observe that with more in-context examples provided to the LLM, the accuracy of the test instance monotonically increases, indicating that the hidden concepts of the tasks are learned.

(c) Function Value Prediction. The goal of this experiment is to let the LLM (`gpt-3`) predict the values of a function on unseen data points given the values on the points with fixed intervals. Following Gruver et al. (2023), we report the t-interval cumulative negative log-likelihood CNLL $= -\sum_i^t \log P(v_i|\text{prompt}_{i-1})$, where v_i is the value of the function at data point i. It can be observed that the LLMs are good time series forecasters.



Under Claim 2.1, we establish the correspondence between LLMs and RL by using LLMs as the model estimators in RL algorithms, which opens the door to creating a practical algorithm that combines the strengths of both LLMs and RL. LLMs excel in accuracy with minimal feedback, which improves the sample efficiency. LLMs can also refine estimates using new feedback as prompts, which avoids explicit parameter updates. RL algorithms benefit from online interaction to improve estimates and policies and have theoretical guarantees with optimal planning algorithms like value iteration. This LLM-RL correspondence inspires us to introduce a new framework in the next section, aiming to orchestrate the reasoning (learning and planning) and acting of LLMs.

3. Algorithm

Architecture of `RAFA`. By leveraging the LLM-RL correspondence in Section 2, we provide a principled framework for orchestrating reasoning and acting, namely "reason for future, act for now" (`RAFA`), in Algorithms 1 and 2. At the t-th step of Algorithm 1, the LLM agent invokes the reasoning routine, which learns from the memory buffer and plans a future trajectory over a long horizon ("reason for future" in Line 6), takes the initial action of the planned trajectory ("act for now" in Line 7), and stores the collected feedback (state, action, and reward) in the memory buffer (Line 8). Upon the state transition of the external environment, the LLM agent reinvokes the reasoning routine to replan another future trajectory from the new state (Line 6 following Line 9). To ensure the learning and planning stability, we impose the switching condition (Line 10) to decide whether to incorporate the newest chunk of history, i.e., the set difference $\mathcal{D}_t - \mathcal{D}_{t_k}$, into the information state, which is used in the reasoning routine as contexts. In other words, the reasoning routine uses the same history $\mathcal{D}_{t_k}$ for all $t_k \le t < t_{k+1}$ until the $(k+1)$-th switch at the $(t_{k+1}-1)$-th step, which guarantees that the posterior distribution and the optimized action or the corresponding policy are updated in a conservative manner. We specify the switching condition in Sections 4 and 5.

Algorithm 1 Reason for future, act for now (`RAFA`): The LLM version.

1: **input**: An LLM learner-planner `LLM-LR-PL`, which aims at generating an optimal trajectory given an initial state and returns the initial action (e.g., Algorithm 2), and a switching condition `If-Switch`.
2: **initialization**: Sample the initial state $s_0 \sim \rho$, set $t = 0$, and initialize the memory buffer $\mathcal{D}_0 = \varnothing$.
3: **for** $k = 0, 1, \ldots,$ **do**
4: Set $t_k \leftarrow t$.
5: **repeat**
6: Learn and plan given memory $\mathcal{D}_{t_k}$ to get action $a_t \leftarrow$ `LLM-LR-PL`$(\mathcal{D}_{t_k}, s_t)$. ("reason for future")
7: Execute action a_t to receive reward r_t and state s_{t+1} from environment. ("act for now")
8: Update memory $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(s_t, a_t, s_{t+1}, r_t)\}$.
9: Set $t \leftarrow t + 1$.
10: **until** `If-Switch`$(\mathcal{D}_t)$ is `True`. (the switching condition is satisfied)
11: **end for**

"Reason for Future" (Line 6 in Algorithm 1 and Lines 3-11 in Algorithm 2). As detailed below, the reasoning routine composes the learning and planning subroutines to map the full history $\mathcal{D}_{t_k}$ (until the t_k-th step) to an optimized action a_t. Note that the reasoning routine does not interact with the external environment throughout the learning and planning subroutines.

• The learning subroutine (Lines 3-4 in Algorithm 2) maps the memory buffer $\mathcal{D}_{t_k}$ to a transition kernel (`Model`) and a value function (`Critic`), which are used in the planning subroutine. In practice, we prompt LLMs to form an estimate of the external environment based on the memory buffer. Here, the estimate is instantiated by two LLMs: `Model` and `Critic`, which estimate their ground-truth counterparts in association with the data-generating parameter. Under Claim 2.1, the learning subroutine yields more accurate versions of `Model` and `Critic` when we prompt them with a growing collection of feedback from the external environment. Consequently, the planning subroutine can use them to assess the long-term outcome of actions with a higher accuracy. Depending on whether we emulate the model-based or model-free approach of RL, we may choose to emulate `Model` or `Critic` individually. Compared with the learning subroutine in RL, we replace the parameterized function approximation (usually deep neural networks) with LLMs and use an "in-context" manner

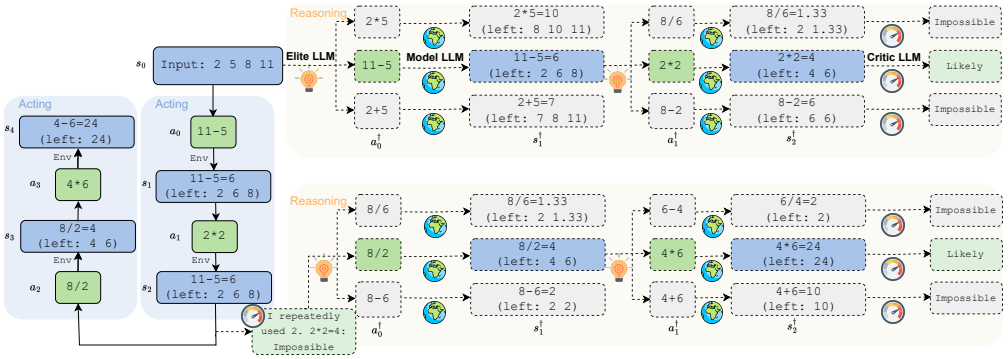


Figure 2. RAFA for Game of 24. Actions are proposed (dotted) and selected (green). Hallucinations that the same number can be reused are mitigated through interactions.

to update the LLMs, which eliminates the need for explicit parameter updates. Because LLMs are pretrained and undergo supervised fine-tuning, they provide much better estimates compared to learning from scratch, leading to an improvement in sample efficiency for online interactions.

Algorithm 2 The LLM learner-planner (`LLM-LR-PL`): A tree-search example. (the deterministic case)

1: **input**: The memory buffer $\mathcal{D}$, the initial state s, the search breadth B, and the search depth U.
2: **initialization**: Initialize the state array $\mathcal{S}_0 \leftarrow \{s\}$ and the action array $\mathcal{A}_0 \leftarrow \varnothing$.
——————— (the learning subroutine) ———————
3: Set `Model` as an LLM instance prompted to use $\mathcal{D}$ as contexts to *generate the next state*.
4: Set `Critic` as an LLM instance prompted to use $\mathcal{D}$ as contexts to *estimate the value function*.
——————— (the planning subroutine) ———————
5: Set `Elite` as an LLM instance prompted to use $\mathcal{D}$ as contexts to *generate multiple candidate actions*.
6: **for** $u = 0, \ldots, U$ **do**
7: For each current state in $\mathcal{S}_u$, invoke `Elite` to generate B candidate actions and store them in $\mathcal{A}_u$.
8: For each candidate action in $\mathcal{A}_u$, invoke `Model` to generate the next state and store it in $\mathcal{S}_{u+1}$.
9: **end for**
10: For all resulting rollouts in $\mathcal{S}_0 \times \mathcal{A}_0 \times \cdots \times \mathcal{S}_U \times \mathcal{A}_U$, invoke `Critic` to evaluate the expected cumulative future reward and select the best one $(s_0^\dagger, a_0^\dagger, \ldots, s_U^\dagger, a_U^\dagger)$, where $s_0^\dagger = s$.
11: **output**: The initial action $a_0^\dagger$ of the selected rollout.

• The planning subroutine (Lines 5-11 in Algorithm 2) maps `Model` and `Critic` to a future trajectory $(s_0^\dagger, a_0^\dagger, \ldots, s_U^\dagger, a_U^\dagger)$, where $s_0^\dagger$ is the current state s_t and $a_0^\dagger$ is executed in the external environment as the current action a_t during the acting phase. Intuitively, we prompt LLMs to generate an optimal policy (actor) for multiple future steps, which maximizes the value function (`Critic`). From an RL perspective (Sections 2 and 5), the planning subroutine approximately solves the Bellman equation (Sutton & Barto, 2018), where we solve the optimal policy (or the corresponding action) given the estimated transition kernel and reward (or critic) by LLMs. As two LLM instances from the learning subroutine, `Model` and `Critic` instantiate the estimated transition kernel and the estimated value function. Hence, we can simulate a given number of trajectories with `Model`, evaluate them with `Critic`, and obtain an improved policy, which is achieved by specially designed prompts instead of a numerical algorithm. By maximizing the expected cumulative future reward (instead of the immediate reward), the planning subroutine returns an optimized action that improves the long-term outcome. There are two error sources that affect the planning subroutine, namely the posterior uncertainty, which is inherited from `Model` and `Critic` due to the finite size of $\mathcal{D}_{t_k}$, and the planning suboptimality, which is induced by the limited capacity for computation, e.g., the bounded width and depth of tree-search (Lines 6-9 in Algorithm 2). Depending on the specific configuration of the state and action spaces (continuous versus discrete) and the transition and reward models (stochastic versus deterministic), we may choose to emulate the value iteration algorithm, the random shooting algorithm, or the Monte-Carlo tree-search algorithm. All of them allow `RAFA` to achieve provable sample efficiency guarantees as long as they satisfy a specific requirement of optimality (Definition C.1). For illustration, we emulate the tree-search algorithm and defer its stochastic variant to Appendix B.

"Act for Now" (Lines 7-10 in Algorithm 1). At the current state s_t, the LLM agent executes the optimized action a_t in the external environment, which is obtained from the reasoning routine. Specifically, we take the initial action $a_0^\dagger$ of the planned trajectory $(s_0^\dagger, a_0^\dagger, \ldots, s_U^\dagger, a_U^\dagger)$, where $s_0^\dagger = s_t$ and $a_0^\dagger = a_t$, and discard the remaining subset. At the next state s_{t+1}, the LLM agent replans another future trajectory $(s_0^\dagger, a_0^\dagger, \ldots, s_U^\dagger, a_U^\dagger)$ with $s_0^\dagger = s_{t+1}$ and $a_0^\dagger = a_{t+1}$. In other words, the acting phase follows a short-term subset of the long-term plan, which is regenerated at every new state. The LLM agent stores the collected feedback (s_t, a_t, r_t, s_{t+1}) in the memory buffer $\mathcal{D}_t$ and queries a switching condition `If-Switch` to decide when to update the memory buffer $\mathcal{D}_{t_k}$, which is used in the reasoning routine as contexts for learning and planning. Intuitively, we incorporate the newest chunk of his-

	gpt-4	gpt-3.5
RAFA ($B = 1$)	89%	29%
RAFA ($B = 2$)	**93%**	**46%**
ToT ($B = 1$)	73%	10%
ToT ($B = 2$)	81%	17%
Reflexion	21%	16%

Figure 3. Game of 24 results.

	Pick	Clean	Heat	Cool	Exam	Pick2	Total
BUTLER	46.00	39.00	74.00	**100.00**	22.00	24.00	37.00
ReAct	66.67	41.94	91.03	80.95	55.56	35.29	61.94
AdaPlanner	**100.00**	**96.77**	95.65	**100.00**	**100.00**	47.06	91.79
Reflexion	**100.00**	90.32	82.61	90.48	**100.00**	94.12	92.54
RAFA	**100.00**	**96.77**	**100.00**	**100.00**	**100.00**	**100.00**	**99.25**

Figure 4. ALFWorld results (success rates %).

tory $\mathcal{D}_t - \mathcal{D}_{t_k}$ to improve the current policy only when it carries significant novel information, e.g., when the LLM agent loses for the first time following a winning streak.

4. Experiment

We evaluate RAFA in several text-based benchmarks, e.g., Game of 24, ALFWorld, BlocksWorld, and Tic-Tac-Toe. The detailed setups, results, and ablations are provided in Appendix G, while the detailed prompts are in Appendix H. Our code is available at https://agentification.github.io/RAFA/.

4.1. Game of 24

Game of 24 (Yao et al., 2023a) is a mathematical puzzle where the player uses basic arithmetic operations (i.e., addition, subtraction, multiplication, division) with four given numbers to get 24. The state is the (possibly unfinished) current formula and the action is the next formula (or the modified part).

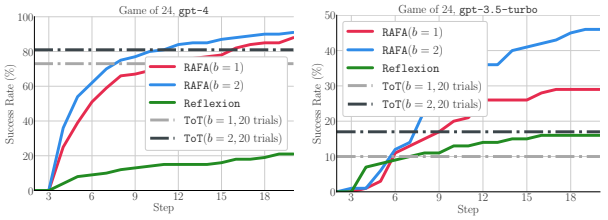


Figure 5. Sample efficiency on Game of 24.

Setup. We emulate the tree-search algorithm to plan ($B \in \{1, 2\}$). At the t-th step, RAFA learns from the memory buffer and switches to a new policy upon receiving an unexpected reward, which is the switching condition. After the t-th step, RAFA digests the collected feedback and generates a linguistic summary, which is saved into the memory buffer to avoid similar previous mistakes.

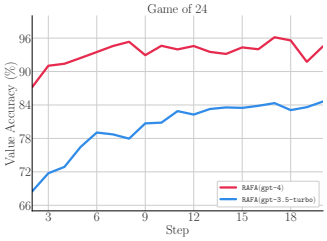


Figure 6. Value estimation accuracy on Game of 24.

Result. We report the performance of RAFA in Table 3, which attains the SOTA. As shown in Figures 2 and 5, superior sample efficiency is achieved by mitigating hallucinations via interactions and the enhanced planning ability to avoid careless trials (see Appendix G.1 for a detailed discussion). To verify that feedback from the environment can help mitigate hallucinations, we report the value estimation accuracy with RAFA agents for different timesteps, as shown in Figure 6. The value estimation is in general more accurate as the number of interaction increases, verifying our claim in Claim 2.1.

4.2. ALFWorld

ALFWorld (Shridhar et al., 2020) is an interactive environment for embodied agent simulations, which encompasses 134 household tasks in six overall categories (Table 4). We use gpt-3 (text-davinci-003).

Setup. We emulate the tree-search algorithm to plan ($B = 2$). RAFA invokes Critic to evaluate the completed portion of the desired goal and switches to a new policy after 20 consecutive failures.

Result. RAFA outperforms various existing frameworks (right figure). The better performance of AdaPlanner at the initial episode is attributed to a handcrafted set of programs for rejecting suboptimal candidate trajectories, which is challenging to construct without the domain knowledge of a specific task. One such example is the PickTwo category, where a substantial performance disparity emerges compared to other tasks.

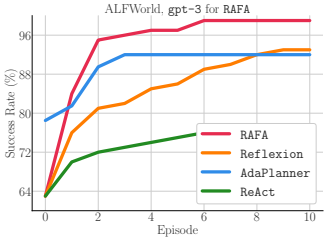


Figure 7. Sample efficiency on ALFWorld.

4.3. Blocksworld

BlocksWorld (Hao et al., 2023) is a rearrangement puzzle. For the RAFA algorithm, we use the Vicuna (Zheng et al., 2023) model and emulate the MCTS algorithm to plan (see Figure 15 in Appendix). RAFA achieves superior success rates across multiple Vicuna versions (Figure 8). Com-

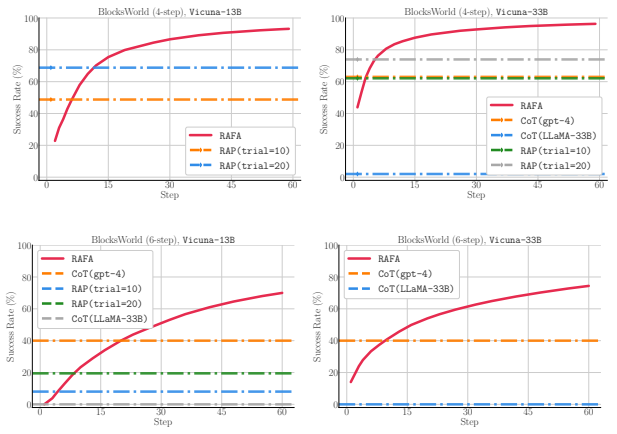


Figure 8. Sample efficiency on BlocksWorld (4 and 6 are the minimum numbers of steps for solving a specific task). `CoT` is prompted by four in-context examples.

O \ X	`gpt-4`		
`gpt-4`	90%,	0%,	**10%**
`RAFA`$(T=1)$	90%,	0%,	**10%**
`RAFA`$(T=5)$	50%,	0%,	**50%**
`RAFA`$(T=7)$	0%,	0%,	**100%**

Table 2. Tic-Tac-Toe results. We set $B = 4$ and report the winning rate of X, the tie rate, and the winning rate of O.

parisons with `CoT` and `RAP` demonstrate how the learning subroutine improves the planning optimality.

4.4. Tic-Tac-Toe

Tic-Tac-Toe (Beck, 2008) is a competitive game where the X and O sides take turns to place marks. `RAFA` invokes `Model` to simulate the transition and opponent dynamics (see Figure 16 in Appendix).

Setup. We use `gpt-4` and emulate the tree-search algorithm to plan ($B \in \{3, 4\}$). `RAFA` switches to a new policy when (a) the predicted state differs from the observed one, (2) the predicted action of opponents differs from the observed one, or (3) `Critic` gives the wrong prediction of the game status. Here, X has an asymmetric advantage (winning surely if played properly).

Result. As shown in Table 2, `RAFA` (playing O) matches and beats `gpt-4` for $T = 5$ and $T = 7$, although O is destined to lose. The ablation study ($B = 3$ versus $B = 4$) illustrates how the planning suboptimality affects the sample efficiency (see Figure 9).

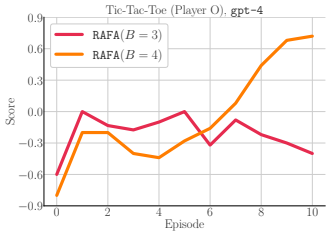


Figure 9. Sample efficiency on Tic-Tac-Toe (0 means tie).

5. Theory

In this section, we provide the theoretical explanation of `RAFA`. For LLMs used in `RAFA`, we denote $P^{\texttt{LLM}}(\xi_{(s,a)} \,|\, \mathcal{D}, s, a)$ as the probability measure of the predicted state-reward pair given the query state-action pair and the memory buffer $\mathcal{D}$ as the in-context dataset. Induced by $P^{\texttt{LLM}}$, we also denote $P_{\texttt{LLM}(\mathcal{D})}$ and $r_{\texttt{LLM}(\mathcal{D})}$ as the estimated transition kernel and reward by LLMs.

LLMs with Posterior Alignments Perform Bayesian Model Averaging (BMA). In the following, we analyze Claim 2.1 from the theoretical perspective. For the simplicity of analysis, we assume that all LLMs have posterior alignments in the tasks that we study, whose posterior distribution of the reward and the next state given the current state-action pair and any in-context dataset matches the posterior in these tasks.

Assumption 5.1 (Posterior Alignment)**.** *We assume that LLMs used in `RAFA` are aligned with the posterior of the state and reward in the underlying MDP, which is formulated as*

$$P^{LLM}\big(\xi_{(s,a)} \,\big|\, \mathcal{D}, s, a\big) = \mathbb{P}_{post}\big(\xi_{(s,a)} \,\big|\, \mathcal{D}, s, a\big),$$

for any in-context dataset $\mathcal{D} = \{(s_i, a_i, r_i, s_i')\}_{i=0}^{I}$ with size I, query state-action pair (s, a), reward r, and state s'. Here, the posterior $\mathbb{P}_{post}$ is defined in (2.3).

We remark that the posterior alignment in Assumption 5.1 comes from the in-context ability of LLMs, which is widely studied in Lee et al. (2023); Wies et al. (2024); Xie et al. (2021). We also remark that Assumption C.3 does not mean that LLMs can make the optimal decision at each step naively in the underlying MDP: (1) Though the posterior distributions of state and reward are aligned, LLMs still need to be instructed to maximize the long-term value (via explicit planning) instead of the myopic reward. (2) LLMs still require online interactions to enlarge the in-context dataset $\mathcal{D}$ such that the prediction uncertainty can be reduced from the prior uncertainty. In Appendix D.6, we also discuss how to relax Assumption 5.1 to accommodate a generalization error in the regret bound derived by our analysis, where we assume that LLMs are MLEs on the pretraining dataset with uniform coverage. Based on Assumption 5.1, we prove that LLMs with posterior alignments perform BMA in model estimation.

Proposition 5.2. *Under Assumption 5.1, it holds that*

$$r_{LLM(\mathcal{D})}(s,a) + (P_{LLM(\mathcal{D})}V)(s,a) = \mathbb{E}_{\theta\sim\mathbb{P}_{post}(\cdot\,|\,\mathcal{D})}[(B_\theta V)(s,a)]$$

for any in-context dataset $\mathcal{D} = \{(s_i, a_i, r_i, s_i')\}_{i=0}^{I}$, value function V, and query state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$.

The proof of Proposition 5.2 can be found in Appendix D.1. Some variants of Proposition 5.2 can be found in various literature (Lee et al., 2023; Zhang et al., 2022; 2023b). In particular, Zhang et al. (2022) establish the theoretical equivalence between BMA and the ideal attention architecture and analyze the generalization error rate of LLMs. By Proposition 5.2, LLMs can provide a more certain and accurate estimate for the data-generating model $\theta^\star$ with more collected feedback, as the uncertainty of $\theta^\star$ in the posterior is reduced with more data. Thus, Proposition 5.2 supports Claim 2.1 in theory, explaining why LLMs can predict more precisely with more feedback from online interactions.

Regret Bound of RAFA. We propose the theoretical version of RAFA in Algorithm 6 in Appendix C, where we instantiate the switching condition of RAFA by measuring the reduction of the posterior entropy and describe the planning subroutine in RAFA as an ϵ-planner $\texttt{PL}^\epsilon$ (defined in Definition C.1 in Appendix C). Next, we impose a regularity assumption on the structure of MDPs to measure the learning difficulty. Recall that we define the posterior entropy H_t in (2.4), the information gain $I(\theta; \xi \,|\, \mathcal{D})$, and $\xi_{(s,a)}$ as the pair of the next state and the current reward (s', r) given the query state-action pair (s, a) in Section 2. Define H_t as the posterior entropy $H(\theta \,|\, \mathcal{D}_t)$ given the dataset $\mathcal{D}_t = \{(s_i, a_i, r_i, s_{i+1})\}_{i=0}^{t-1}$.

Assumption 5.3 (MDPs Regularity)**.** *We assume that there exists a coefficient $\eta > 0$ such that, if $H_{t_1} - H_{t_2} \le \log 2$, then it holds that*

$$I(\theta; \xi_{(s,a)} \,|\, \mathcal{D}_{t_1}) \le 4\eta \cdot I(\theta; \xi_{(s,a)} \,|\, \mathcal{D}_{t_2})$$

for any given value function V, $t_1 < t_2$ and $(s, a) \in \mathcal{S} \times \mathcal{A}$.

Assumption 5.3 is a regularity assumption on MDPs and is intrinsic to the agent design. In Appendix F, we prove that d-dimensional Bayesian linear kernel MDPs (defined in Definition F.1), satisfy Assumption 5.3 with the coefficient $\eta = d/\log(1+d)$. Intuitively, Assumption 5.3 restricts the increase of the information gain given one bit ($\log 2$) reduction of the posterior entropy.

Similar to other theoretical work on deep RL (Lazaric et al., 2010; Fan et al., 2020; Zhang et al., 2020), we introduce the concentrability coefficient κ to bound the distribution shift between the current policy and the optimal policy. Intuitively, κ measures the hardness to generalize the low prediction error $(B_k - B_{\theta^\star})V_t$ on the current trajectory induced by $\{\pi^k\}_{k=0}^{K-1}$ to the optimal trajectory induced by $\pi^\star$ in the underlying MDP. for any $t < T$. See the full statement in Assumption C.6 in Appendix C.

In the following theorem, we give the bound of the Bayesian regret of RAFA (Algorithm 6), where the proof can be found in Appendix D.3.

Theorem 5.4. *Suppose Assumptions 5.3 and C.6 hold. For the underlying MDP satisfying Assumption 5.1,the Bayesian regret $\mathfrak{R}(T)$ of RAFA (Algorithm 6) is*

$$\mathcal{O}\Big((1-\gamma)^{-1} L\big(\kappa\eta\sqrt{\mathbb{E}[H_0 - H_T]} \cdot \sqrt{T} + \mathbb{E}[H_0 - H_T]\big)\Big),$$

if the planning suboptimality $\epsilon = \mathcal{O}(1/\sqrt{T})$. Here, L is the bound of $|r + V(s)|$ for any $r \in \mathcal{R}$, $s \in \mathcal{S}$, and value V.

Theorem 5.4 highlights an intriguing interplay between the prior knowledge obtained through pretraining and the uncertainty reduction achieved by reasoning and acting. Since H_0 quantifies the prior uncertainty of LLMs before incorporating any collected feedback, $H_0 - H_T$ highlights the uncertainty reduction achieved by reasoning and acting, as H_T quantifies the posterior uncertainty of LLMs after incorporating the collected feedback. We introduce the d-dimensional Bayesian linear kernel MDP in Appendix F, where we prove that $H_0 - H_T = \mathcal{O}(d \cdot \log T)$ and specialize Theorem 5.4 to obtain $\sqrt{T}$ regret. Hence, we show that RAFA (Algorithm 8) is sample-efficient, which explains the superior empirical performance of RAFA in Section 4.

RAFA with Efficient Exploration Strategies. In Appendix D.6, we show that Assumption 5.1 can be relaxed for Theorem 5.4 to accommodate a generalization error. We also remark that we can drop the dependency of the concentrability coefficient κ (Assumption C.6) if we modify RAFA to encourage efficient exploration in MDPs. In Appendix C.5, we provide two variants of RAFA: (1) RAFA with optimistic bonus (Algorithm 7) and (2) RAFA with posterior sampling (Algorithm 8). In Theorems C.8 and C.10, we bound the Bayesian regret of each variant as

$$\mathcal{O}\Big((1-\gamma)^{-1} L\big(\eta\sqrt{\mathbb{E}[H_0 - H_T]} \cdot \sqrt{T} + \mathbb{E}[H_0 - H_T]\big)\Big),$$

which removes the dependency of the concentrability coefficient κ in the Bayesian regret of Algorithm 6. This theoretical result can also shed light on the design of practical variants of RAFA for more complex environments. Though we propose the theoretical algorithms of these two variants of RAFA, it is still unclear how to design their practical implementations, which we remain for future work.

6. Conclusions

In this paper, we establish the LLM-RL correspondence and propose a principled framework RAFA for orchestrating reasoning and acting, which achieves provable sample efficiency guarantees in autonomous LLM agents for the first time. We prove the $\sqrt{T}$ regret bound of RAFA to highlight the synergy between prior knowledge from pretraining and the iterative process of reasoning and acting. RAFA's outstanding empirical performance underscores its potential for autonomous and adaptive decision-making in various complex tasks, which we remain for future work.

Impact Statement

This paper presents work whose goal is to advance the field of LLMs and online interactions. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

Acknowledgement

Zhaoran Wang acknowledges National Science Foundation (Awards 2048075, 2008827, 2015568, 1934931), Simons Institute (Theory of Reinforcement Learning), Amazon, J.P. Morgan, and Two Sigma for their supports.

References

Abbasi-Yadkori, Y. and Szepesvári, C. Bayesian optimal control of smoothly parameterized systems. In *Uncertainty in Artificial Intelligence*, 2015.

Abbasi-Yadkori, Y., Pál, D., and Szepesvári, C. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, 2011.

Abernethy, J., Agarwal, A., Marinov, T. V., and Warmuth, M. K. A mechanism for sample-efficient in-context learning for sparse retrieval tasks. *arXiv preprint arXiv:2305.17040*, 2023.

Agarwal, A., Kakade, S., Krishnamurthy, A., and Sun, W. Flambe: Structural complexity and representation learning of low rank mdps. *Advances in neural information processing systems*, 33:20095–20107, 2020.

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., et al. Do as I can, not as I say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. What learning algorithm is in-context learning? Investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

Beck, J. *Combinatorial games: Tic-Tac-Toe theory*. 2008.

Betts, J. T. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 1998.

Bickel, P. J. and Freedman, D. A. Some asymptotic theory for the bootstrap. *The annals of statistics*, 9(6):1196–1217, 1981.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Cai, Q., Yang, Z., Jin, C., and Wang, Z. Provably efficient exploration in policy optimization. In *International Conference on Machine Learning*, 2020.

Cai, T., Wang, X., Ma, T., Chen, X., and Zhou, D. Large language models as tool makers. *arXiv preprint arXiv:2305.17126*, 2023.

Chen, Y., He, J., and Gu, Q. On the sample complexity of learning infinite-horizon discounted linear kernel MDPs. In *International Conference on Machine Learning*, 2022.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. PaLM: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Creswell, A. and Shanahan, M. Faithful reasoning using large language models. *arXiv preprint arXiv:2208.14271*, 2022.

Creswell, A., Shanahan, M., and Higgins, I. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*, 2022.

Davison, A. C. and Hinkley, D. V. *Bootstrap methods and their application*. Number 1. Cambridge university press, 1997.

Dong, K., Wang, Y., Chen, X., and Wang, L. Q-learning with UCB exploration is sample efficient for infinite-horizon MDP. *arXiv preprint arXiv:1901.09311*, 2019.

Efron, B. *The jackknife, the bootstrap and other resampling plans*. SIAM, 1982.

Fan, J., Wang, Z., Xie, Y., and Yang, Z. A theoretical analysis of deep q-learning. In *Learning for dynamics and control*, pp. 486–489. PMLR, 2020.

Garg, S., Tsipras, D., Liang, P. S., and Valiant, G. What can transformers learn in-context? A case study of simple function classes. In *Advances in Neural Information Processing Systems*, 2022.

Ghavamzadeh, M., Mannor, S., Pineau, J., Tamar, A., et al. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015.

Ghosh, M. Exponential tail bounds for chisquared random variables. *Journal of Statistical Theory and Practice*, 15 (2):35, 2021.

Gruver, N., Finzi, M., Qiu, S., and Wilson, A. G. Large language models are zero-shot time series forecasters. *arXiv preprint arXiv:2310.07820*, 2023.

Guo, H., Liu, Z., Zhang, Y., and Wang, Z. Can large language models play games? a case study of a self-play approach. *arXiv preprint arXiv:2403.05632*, 2024.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019.

Hao, B., Abbasi Yadkori, Y., Wen, Z., and Cheng, G. Bootstrapping upper confidence bound. *Advances in neural information processing systems*, 32, 2019.

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

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

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*, 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.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

Jiang, H. A latent space theory for emergent abilities in large language models. *arXiv preprint arXiv:2304.09960*, 2023.

Jin, C., Yang, Z., Wang, Z., and Jordan, M. I. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pp. 2137–2143. PMLR, 2020.

Kim, G., Baldi, P., and McAleer, S. Language models can solve computer tasks. *arXiv preprint arXiv:2303.17491*, 2023.

Kirsch, L., Harrison, J., Sohl-Dickstein, J., and Metz, L. General-purpose in-context learning by meta-learning transformers. *arXiv preprint arXiv:2212.04458*, 2022.

Lazaric, A., Ghavamzadeh, M., and Munos, R. Analysis of a classification-based policy iteration algorithm. In *ICML-27th International Conference on Machine Learning*, pp. 607–614. Omnipress, 2010.

Lee, J. N., Xie, A., Pacchiano, A., Chandak, Y., Finn, C., Nachum, O., and Brunskill, E. Supervised pretraining can learn in-context reinforcement learning. *arXiv preprint arXiv:2306.14892*, 2023.

Li, B. Z., Nye, M., and Andreas, J. Language modeling with latent situations. *arXiv preprint arXiv:2212.10012*, 2022.

Li, Y., Ildiz, M. E., Papailiopoulos, D., and Oymak, S. Transformers as algorithms: Generalization and implicit model selection in in-context learning. *arXiv preprint arXiv:2301.07067*, 2023.

Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A., et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

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

Liu, Z., Lu, M., Wang, Z., Jordan, M., and Yang, Z. Welfare maximization in competitive equilibrium: Reinforcement learning for markov exchange economy. In *International Conference on Machine Learning*, pp. 13870–13911. PMLR, 2022a.

Liu, Z., Zhang, Y., Fu, Z., Yang, Z., and Wang, Z. Learning from demonstration: Provably efficient adversarial policy imitation with linear function approximation. In *International conference on machine learning*, pp. 14094–14138. PMLR, 2022b.

Liu, Z., Lu, M., Xiong, W., Zhong, H., Hu, H., Zhang, S., Zheng, S., Yang, Z., and Wang, Z. Maximize to explore: One objective function fusing estimation, planning, and exploration. In *Advances in Neural Information Processing Systems*, volume 36, pp. 22151–22165, 2023b.

Liu, Z., Lu, M., Xiong, W., Zhong, H., Hu, H., Zhang, S., Zheng, S., Yang, Z., and Wang, Z. Maximize to explore: One objective function fusing estimation, planning, and

exploration. *Advances in Neural Information Processing Systems*, 36, 2024.

Lu, P., Peng, B., Cheng, H., Galley, M., Chang, K.-W., Wu, Y. N., Zhu, S.-C., and Gao, J. Chameleon: Plug-and-play compositional reasoning with large language models. *arXiv preprint arXiv:2304.09842*, 2023.

Lu, X. and Van Roy, B. Information-theoretic confidence bounds for reinforcement learning. *Advances in Neural Information Processing Systems*, 2019.

Morari, M. and Lee, J. H. Model predictive control: past, present and future. *Computers & chemical engineering*, 23(4-5):667–682, 1999.

Newton, M. A. and Raftery, A. E. Approximate bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 56(1):3–26, 1994.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

OpenAI. GPT-4 technical report, 2023.

Osband, I., Russo, D., and Van Roy, B. (More) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, 2013.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016.

Paul, D., Ismayilzada, M., Peyrard, M., Borges, B., Bosselut, A., West, R., and Faltings, B. REFINER: Reasoning feedback on intermediate representations. *arXiv preprint arXiv:2304.01904*, 2023.

Pouplin, T., Sun, H., Holt, S., and Van der Schaar, M. Retrieval-augmented thought process as sequential decision making. *arXiv preprint arXiv:2402.07812*, 2024.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners, 2019.

Rawlings, J. B. Tutorial overview of model predictive control. *IEEE Control Systems Magazine*, 2000.

Razeghi, Y., Logan IV, R. L., Gardner, M., and Singh, S. Impact of pretraining term frequencies on few-shot reasoning. *arXiv preprint arXiv:2202.07206*, 2022.

Russo, D. and Van Roy, B. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 2014a.

Russo, D. and Van Roy, B. Learning to optimize via information-directed sampling. In *Advances in Neural Information Processing Systems*, 2014b.

Russo, D. and Van Roy, B. An information-theoretic analysis of Thompson sampling. *Journal of Machine Learning Research*, 2016.

Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., and Pathak, D. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pp. 8583–8592. PMLR, 2020.

Sel, B., Al-Tawaha, A., Khattar, V., Wang, L., Jia, R., and Jin, M. Algorithm of thoughts: Enhancing exploration of ideas in large language models. *arXiv preprint arXiv:2308.10379*, 2023.

Shen, Y., Song, K., Tan, X., Li, D., Lu, W., and Zhuang, Y. HuggingGPT: Solving AI tasks with ChatGPT and its friends in HuggingFace. *arXiv preprint arXiv:2303.17580*, 2023.

Shin, S., Lee, S.-W., Ahn, H., Kim, S., Kim, H., Kim, B., Cho, K., Lee, G., Park, W., Ha, J.-W., et al. On the effect of pretraining corpora on in-context learning by a large-scale language model. *arXiv preprint arXiv:2204.13509*, 2022.

Shinn, N., Cassano, F., Labash, B., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023.

Shridhar, M., Yuan, X., Côté, M.-A., Bisk, Y., Trischler, A., and Hausknecht, M. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Singh, K. On the asymptotic accuracy of efron's bootstrap. *The Annals of Statistics*, pp. 1187–1195, 1981.

Strens, M. A Bayesian framework for reinforcement learning. In *International Conference on Machine Learning*, 2000.

Sun, H. Reinforcement learning in the era of llms: What is essential? what is needed? an rl perspective on rlhf, prompting, and beyond, 2023.

Sun, H., Hüyük, A., and van der Schaar, M. Query-dependent prompt evaluation and optimization with offline inverse rl. In *The Twelfth International Conference on Learning Representations*, 2023a.

Sun, H., Zhuang, Y., Kong, L., Dai, B., and Zhang, C. AdaPlanner: Adaptive planning from feedback with language models. *arXiv preprint arXiv:2305.16653*, 2023b.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. 2018.

Todd, E., Li, M. L., Sharma, A. S., Mueller, A., Wallace, B. C., and Bau, D. Function vectors in large language models. *arXiv preprint arXiv:2310.15213*, 2023.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. LLaMa: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., and Vladymyrov, M. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, 2023.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022a.

Wang, X., Zhu, W., and Wang, W. Y. Large language models are implicitly topic models: Explaining and finding good demonstrations for in-context learning. *arXiv preprint arXiv:2301.11916*, 2023a.

Wang, Z., Wang, J., Zhou, Q., Li, B., and Li, H. Sample-efficient reinforcement learning via conservative model-based actor-critic. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8612–8620, 2022b.

Wang, Z., Cai, S., Liu, A., Ma, X., and Liang, Y. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*, 2023b.

Wang, Z., Pan, T., Zhou, Q., and Wang, J. Efficient exploration in resource-restricted reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(8):10279–10287, Jun. 2023c.

Wasserman, L. Bayesian model selection and model averaging. *Journal of mathematical psychology*, 44(1):92–107, 2000.

Wei, C.-Y., Jahromi, M. J., Luo, H., Sharma, H., and Jain, R. Model-free reinforcement learning in infinite-horizon average-reward Markov decision processes. In *International Conference on Machine Learning*, 2020.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Wies, N., Levine, Y., and Shashua, A. The learnability of in-context learning. *arXiv preprint arXiv:2303.07895*, 2023.

Wies, N., Levine, Y., and Shashua, A. The learnability of in-context learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Xie, S. M., Raghunathan, A., Liang, P., and Ma, T. An explanation of in-context learning as implicit Bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

Yang, L. and Wang, M. Sample-optimal parametric q-learning using linearly additive features. In *International Conference on Machine Learning*, 2019.

Yang, L. and Wang, M. Reinforcement learning in feature space: Matrix bandit, kernels, and regret bound. In *International Conference on Machine Learning*, 2020.

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

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023a.

Yao, Y., Li, Z., and Zhao, H. Beyond chain-of-thought, effective graph-of-thought reasoning in large language models. *arXiv preprint arXiv:2305.16582*, 2023b.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022.

Zhang, S., Zheng, S., Ke, S., Liu, Z., Jin, W., Yuan, J., Yang, Y., Yang, H., and Wang, Z. How can llm guide rl? a value-based approach. *arXiv preprint arXiv:2402.16181*, 2024.

Zhang, Y., Cai, Q., Yang, Z., and Wang, Z. Generative adversarial imitation learning with neural network parameterization: Global optimality and convergence rate. In *International Conference on Machine Learning*, pp. 11044–11054. PMLR, 2020.

Zhang, Y., Liu, B., Cai, Q., Wang, L., and Wang, Z. An analysis of attention via the lens of exchangeability and latent variable models. *arXiv preprint arXiv:2212.14852*, 2022.

Zhang, Y., Yang, J., Yuan, Y., and Yao, A. C.-C. Cumulative reasoning with large language models. *arXiv preprint arXiv:2308.04371*, 2023a.

Zhang, Y., Zhang, F., Yang, Z., and Wang, Z. What and how does in-context learning learn? Bayesian model averaging, parameterization, and generalization. *arXiv preprint arXiv:2305.19420*, 2023b.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging LLM-as-a-judge with MT-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023.

Zheng, S., Wang, L., Qiu, S., Fu, Z., Yang, Z., Szepesvari, C., and Wang, Z. Optimistic exploration with learned features provably solves markov decision processes with neural dynamics. In *The Eleventh International Conference on Learning Representations*, 2022.

Zhong, H., Xiong, W., Zheng, S., Wang, L., Wang, Z., Yang, Z., and Zhang, T. Gec: A unified framework for interactive decision making in mdp, pomdp, and beyond. *arXiv preprint arXiv:2211.01962*, 2022.

Zhou, D., Gu, Q., and Szepesvári, C. Nearly minimax optimal reinforcement learning for linear mixture Markov decision processes. In *Conference on Learning Theory*, 2021a.

Zhou, D., He, J., and Gu, Q. Provably efficient reinforcement learning for discounted MDPs with feature mapping. In *International Conference on Machine Learning*, 2021b.

Contents

A. More Literature

Large Language Model (LLM) and In-Context Learning (ICL). LLMs (Radford et al., 2019; Brown et al., 2020; Hoffmann et al., 2022; Chowdhery et al., 2022; OpenAI, 2023; Touvron et al., 2023) display notable reasoning abilities. A pivotal aspect of reasoning is the ICL ability (Liang et al., 2022; Razeghi et al., 2022; Shin et al., 2022; Olsson et al., 2022; Akyürek et al., 2022; Kirsch et al., 2022; Garg et al., 2022; Von Oswald et al., 2023; Li et al., 2023; Abernethy et al., 2023), which allows LLMs to solve a broad range of tasks with only a few in-context examples instead of finetuning parameters on a specific dataset. We focus on harnessing the ICL ability of LLMs to optimize actions in the real world, which is crucial to autonomous LLM agents. In particular, we build on a recent line of work (Xie et al., 2021; Zhang et al., 2022; 2023b; Wang et al., 2023a; Wies et al., 2023; Jiang, 2023; Lee et al., 2023) that attributes the ICL ability to implicit Bayesian inference, i.e., an implicit mechanism that enables LLMs to infer a latent concept from those in-context examples, which is verified both theoretically and empirically. In `RAFA`, the latent concept is the transition and reward models (model) of the unknown environment or/and the value function (critic), which is inferred from the memory buffer in the learning subroutine. Claim 2.1 can also be considered as a result of ICL ability.

Reinforcement Learning (RL) under a Bayesian Framework. We build on a recent line of work on the infinite-horizon (Abbasi-Yadkori & Szepesvári, 2015; Dong et al., 2019; Wei et al., 2020; Zhou et al., 2021a;b; Chen et al., 2022; Chua et al., 2018; Hafner et al., 2019; Sekar et al., 2020) and Bayesian (Strens, 2000; Osband et al., 2013; Russo & Van Roy, 2014b;a; 2016; Lu & Van Roy, 2019) settings of RL, which include model-based deep RL (Janner et al., 2019; Liu et al., 2023b; Wang et al., 2022b; Liu et al., 2024), model predictive control (Morari & Lee, 1999), and Thompson sampling (Russo & Van Roy, 2014a). The infinite-horizon setting allows `RAFA` to interact with the external environment continuously without resetting to an initial state, while the Bayesian setting allows us to connect `RAFA` with BMA and establish the theoretical guarantee. RL operates in a numerical system, where rewards and transitions are defined by scalars and probabilities, and trains actors and critics on the collected feedback iteratively. We focus on emulating the actor-model or actor-critic update in RL through an internal mechanism of reasoning on top of LLMs, which allows data and actions to be tokens in a linguistic system while bypassing the explicit update of parameters in model-based RL (Chua et al., 2018; Hafner et al., 2019; Sekar et al., 2020; Liu et al., 2022b; Zhong et al., 2022; Zheng et al., 2022; Liu et al., 2022a). In particular, the learning and planning subroutines of `RAFA` emulate the posterior update and various planning algorithms in RL. Moreover, `RAFA` orchestrates reasoning (learning and planning) and acting following the principled approach in RL, i.e., (re)planning a future trajectory over a long horizon ("reason for future") at the new state and taking the initial action of the planned trajectory ("act for now"). As a result, `RAFA` inherits provable sample efficiency guarantees from RL. We summarize the comparison between `RAFA` and other closed-loop mechanisms in Table 1.

B. More Algorithms

Depending on the specific configuration of the state and action spaces (continuous versus discrete) and the transition and reward models (stochastic versus deterministic), we may choose to emulate the tree-search algorithm, the value iteration algorithm, the random shooting algorithm, or the MCTS algorithm. All of them allow `RAFA` to achieve provable sample efficiency guarantees as long as they satisfy a specific requirement of optimality (Definition C.1). For illustration, we emulate the beam-search algorithm (an advanced version of the tree-search algorithm) in Algorithm 3 and the MCTS algorithm in Algorithm 4. For the theoretical discussion, we present the value iteration algorithm in Algorithm 5.

Algorithm 3 The LLM learner-planner (`LLM-LR-PL`): A beam-search example (for the deterministic case).

input: The memory buffer $\mathcal{D}$, the initial state s, the proposal width L, the search breadth B, and the search depth U.

initialization: Initialize the state array $\mathcal{S}_0 \leftarrow \{s\}$ and the action array $\mathcal{A}_0 \leftarrow \varnothing$.

———————————————————- (the learning subroutine) ———————————————————

Set `Model` as an LLM instance prompted to use $\mathcal{D}$ as contexts to generate the next state.

Set `Critic` as an LLM instance prompted to use $\mathcal{D}$ as contexts to estimate the value function.

———————————————————- (the planning subroutine) ———————————————————-

Set `Elite` as an LLM instance prompted to use $\mathcal{D}$ as contexts to generate multiple candidate actions.

for $u = 0, \ldots, U$ **do**

  For each current state s_u in $\mathcal{S}_u$, invoke `Elite` to generate L candidate actions.

  For each candidate action $a_u^{(\ell)}$, invoke `Model` to generate the next state $s_{u+1}^{(\ell)}$ and the received reward $r_u^{(\ell)}$.

  For each resulting tuple $(s_u, a_u^{(\ell)}, s_{u+1}^{(\ell)}, r_u^{(\ell)})$, invoke `Critic` to evaluate the expected cumulative future reward $\widehat{Q}(s_u, a_u^{(\ell)}) \leftarrow r_u^{(\ell)} + \gamma \widehat{V}(s_{u+1}^{(\ell)})$, where $\widehat{V}$ is given by `Critic`.

  Select B best tuples $(s_u, a_u^{(\ell)}, s_{u+1}^{(\ell)})$ with the highest value $\widehat{Q}(s_u, a_u^{(\ell)})$ and write them to $\mathcal{S}_u \times \mathcal{A}_u \times \mathcal{S}_{u+1}$.

end for

For B preserved rollouts in $\mathcal{S}_0 \times \mathcal{A}_0 \times \cdots \times \mathcal{S}_U \times \mathcal{A}_U \times \mathcal{S}_{U+1}$, invoke `Critic` to evaluate the expected cumulative future reward $\sum_{u=0}^{U} \gamma^u r_u^{(b)} + \gamma^{U+1} \widehat{V}(s_{U+1}^{(b)})$ and select the best one $(s_0^\dagger, a_0^\dagger, \ldots, s_U^\dagger, a_U^\dagger, s_{U+1}^\dagger)$, where $\widehat{V}$ is given by `Critic` and $s_0^\dagger = s$.

output: The initial action $a_0^\dagger$ of the selected rollout and its initial value.

Algorithm 4 LLM learner-planner (`LLM-PL`) for `RAFA`: A Monte-Carlo tree-search example (for the stochastic case).

input: The memory buffer $\mathcal{D}$, the initial state s, the proposal width L, L', and the expansion budget E.

initialization: Initialize the root node $n \leftarrow s$ and the child function $c(\cdot) \leftarrow \varnothing$.

——————————————————— (the learning subroutine) ———————————————————

Set `Model` as an LLM instance prompted to use $\mathcal{D}$ as contexts to generate the next state.

Set `Critic` as an LLM instance prompted to use $\mathcal{D}$ as contexts to estimate the value function.

——————————————————— (the planning subroutine) ———————————————————

Set `Elite` as an LLM instance prompted to use $\mathcal{D}$ as contexts to generate multiple candidate actions.

for $e = 0, \ldots, E$ **do**

  Set $s_e \leftarrow n$.

  while s_e is not a leaf node, i.e., $c(s_e) \neq \varnothing$, **do**

    Invoke `Critic` to evaluate the expected cumulative future reward and select the child node a_e in $c(s_e)$ with the highest value $\widehat{Q}(s_e, a_e)$.

    Set s_e as a child node in $c(a_e)$.

  end while

  For the current state s_e, invoke `Elite` to generate L candidate actions.

  Write each candidate action $a_e^{(\ell)}$ to $c(s_e)$, i.e., $c(s_e) \leftarrow \{a_e^{(\ell)}\}_{\ell=1}^{L}$.

  For each candidate action $a_e^{(\ell)}$, invoke `Model` to sample L' next states.

  Write each next state $s_e^{(\ell,\ell')}$ to $c(a_e^{(\ell)})$, i.e., $c(a_e^{(\ell)}) \leftarrow \{s_e^{(\ell,\ell')}\}_{\ell'=1}^{L'}$.

  For each generated state $s_e^{(\ell,\ell')}$, invoke `Critic` to evaluate the expected cumulative future reward and update the estimated value $\widehat{V}$ for all ancestor nodes. (Optional)

end for

Set $s_0^\dagger \leftarrow n$ and $i \leftarrow 0$.

while $s_i^\dagger$ is not a leaf node, i.e., $c(s_i^\dagger) \neq \varnothing$, **do**

  Invoke `Critic` to evaluate the expected cumulative future reward and select the child node $a_{i+1}^\dagger$ in $c(s_i^\dagger)$ with the highest value $\widehat{Q}(s_i^\dagger, a_i^\dagger)$.

  Set $s_{i+1}^\dagger$ as a child node in $c(a_i^\dagger)$ and $i \leftarrow i + 1$.

end while

output: The initial action $a_0^\dagger$ of the selected rollout $(s_0^\dagger, a_0^\dagger, \ldots, s_i^\dagger, a_i^\dagger)$ and its initial value.

C. Theoretical Analysis

In this section, we provide the full theoretical analysis for the statements in Section 5.

C.1. Notations

We provide a table of notations here.

Notations	Explanation
$\xi_{(s,a)}$	the pair of the next state and the current reward (s', r) given the query state-action pair (s, a)
$P^{\texttt{LLM}}(\xi_{(s,a)} \mid \mathcal{D}, s, a)$	the probability measure of the LLM predicted $\xi_{(s,a)}$ given the memory buffer $\mathcal{D}$ as contexts
$r_{\text{LLM}(\mathcal{D})}$	the LLM reward estimator with the memory buffer $\mathcal{D}$ prompted as contexts
$P_{\text{LLM}(\mathcal{D})}$	the LLM transition kernel estimator with the memory buffer $\mathcal{D}$ prompted as contexts
$\mathcal{D}_t$	the history at the t-th step, which includes $\{(s_i, a_i, r_i, s_{i+1})\}_{i=0}^{t-1}$
$\mathbb{P}_0(\theta)$	the prior of $\theta^\star$
$\mathbb{P}_{\text{post}}(\theta \mid \mathcal{D})$	the posterior of $\theta^\star$ conditioned on $\mathcal{D}$
$\mathbb{P}_t(\theta)$	the abbreviation of $\mathbb{P}_{\text{post}}(\theta \mid \mathcal{D}_t)$
$H(\theta \mid \mathcal{D})$	the posterior entropy of the posterior of θ conditioned on $\mathcal{D}$
$I(\theta; \xi \mid \mathcal{D})$	the information gain of ξ, defined by $H(\theta \mid \mathcal{D}) - H(\theta, \xi \mid \mathcal{D})$
H_t	the abbreviation of $H(\theta \mid \mathcal{D}_t)$
t_k	the timestep when `RAFA` switches the policy for the k-th time
π_k	the abbreviation of π_{t_k}
B_θ	the Bellman operator induced by θ
B_k	the Bellman operator induced by $\texttt{LLM}(\mathcal{D}_{t_k})$
$d_{\text{TV}}(p\|q)$	total variation (TV) between two probability measures p and q
$d_{\text{KL}}(p\|q)$	Kullback–Leibler divergence between two probability measures p and q
$\mathbb{V}_{x\sim p}[f(x)]$	the variance of $f(X)$, where X follows the distribution p
$(PV)(s,a)$	$\mathbb{E}_{s'\sim P(\cdot \mid s,a)}[V(s')]$
L	the bound of $\lvert r + V(s)\rvert$ for any $r \in \mathcal{R}$, $s \in \mathcal{S}$, and value V
$\nu^\star(\cdot \mid s)$	the optimal γ-discounted visitation measure starting from state s
$\mathbb{N}$	the set of natural numbers
$\mathbf{1}(x = y)$	the indicator with value 1 if x equals y and value 0 otherwise
$\mathbb{E}$	the expectation
$\mathbb{V}$	the variance

C.2. Planning Optimality

To define the optimal policy, we define the Bellman optimality equation as

$$\begin{aligned} Q_\theta^\star(s,a) &= \left(B_\theta V_\theta^\star\right)(s,a), \\ V_\theta^\star(s) &= \max_{a\in\mathcal{A}} Q_\theta^\star(s,a), \end{aligned} \tag{C.1}$$

where $Q_\theta^\star$ and $V_\theta^\star$ are the fixed-point solutions Here, we define $(B_\theta V_\theta^\star)(s,a) = r_\theta(s,a) + \mathbb{E}[V_\theta^\star(s')]$, where $\mathbb{E}$ is taken with respect to $s' \sim P_\theta(\cdot|s,a)$. Let $\pi_\theta^\star(s) = \arg\max_{a\in\mathcal{A}} Q_\theta^\star(s,a)$. See Sutton & Barto (2018) for the existence and uniqueness guarantees for $Q_\theta^\star$, $V_\theta^\star$, and $\pi_\theta^\star$.

To measure the performance of the planning subroutine in `RAFA` (Algorithm 1), we define the ϵ-optimal planner as follows.

Definition C.1 (ϵ-Optimal Planner)**.** A planning algorithm $\texttt{PL}^\epsilon : (\mathcal{P},\mathcal{R}) \mapsto \Pi$ is an ϵ-optimal planner if $\texttt{PL}^\epsilon(P,r) = (\pi, V)$, where $|Q(s,a) - r(s,a) - (PV)(s,a)| \le \epsilon$ and $V(s) = \max_a Q(s,a) = Q(s,\pi(s))$ for all $(s,a) \in \mathcal{S}\times\mathcal{A}$.

As a special case of Definition C.1, We present the value iteration algorithm (Algorithm 5) with a truncated horizon U, i.e., a finite length of the lookahead window as the ϵ-optimal planner in Algorithm 6. The following proposition ensures that Algorithm 5 satisfies Definition C.1.

Proposition C.2. Algorithm 5 is an ϵ-optimal planner as long as we set $U \ge 1 + \lceil \log_\gamma(\epsilon/L)\rceil$ and any value function is upper bounded by $L \ge 0$.

Proof. See Appendix E.2 for a detailed proof. □

Algorithm 5 ϵ-Optimal planner: The value iteration algorithm with a truncated horizon.

1: **input**: The model (P,r) and the truncated horizon U.
2: **initialization**: Set the value function $V_\theta^{(U)}(\cdot) \leftarrow 0$.
3: **for** $u = U-1,\dots,1$ **do**
4: Set the value function $V^{(u)}(\cdot) \leftarrow \max_{a\in\mathcal{A}} Q^{(u)}(\cdot,a)$, where $Q^{(u)}(\cdot,\cdot) \leftarrow r(\cdot,\cdot) + \gamma(PV^{(u+1)})(\cdot,\cdot)$.
5: **end for**
6: **output**: The greedy policy $\pi(\cdot) = \arg\max_{a\in\mathcal{A}} Q^{(1)}(\cdot,a)$ and the value $V^{(1)}$.

Alternatively, we may choose to emulate the tree-search algorithm, the random shooting algorithm, or the Monte-Carlo tree-search algorithm. In the tree-search example (Lines 5-11 in Algorithm 2), ϵ decreases as the search breadth B and depth U increase. Note that, as long as we emulate an ϵ-optimal planner, we are able to establish provable sample efficiency guarantees.

C.3. LLMs with Posterior Alignments Perform Bayesian Model Averaging (BMA)

In the following, we analyze Claim 2.1 from the theoretical perspective. For LLMs used in `RAFA`, we denote $P^{\texttt{LLM}}(\xi_{(s,a)} \,|\, \mathcal{D}, s, a)$ as the probability measure of the predicted state-reward pair given the query state-action pair and the memory buffer $\mathcal{D}$ as the in-context dataset. Induced by $P^{\texttt{LLM}}$, we also denote $P_{\texttt{LLM}(\mathcal{D})}$ and $r_{\texttt{LLM}(\mathcal{D})}$ as the estimated transition kernel and reward by LLMs.

For the simplicity of analysis, we assume that all LLMs have posterior alignments in the tasks that we study, whose posterior distribution of the reward and the next state given the current state-action pair and any in-context dataset matches the posterior in these tasks. We formulate this assumption as follows.

Assumption C.3 (Posterior Alignment)**.** *We assume that LLMs are aligned with the posterior of the state and reward in the underlying MDP, which is formulated as*

$$P^{LLM}\big(\xi_{(s,a)} \,\big|\, \mathcal{D}, s, a\big) = \mathbb{P}_{post}\big(\xi_{(s,a)} \,\big|\, \mathcal{D}, s, a\big),$$

for any in-context dataset $\mathcal{D} = \{(s_i, a_i, r_i, s_i')\}_{i=0}^{I}$ with size I, query state-action pair (s, a), reward r, and state s'. Here, the posterior $\mathbb{P}_{post}$ is defined in (2.3).

We remark that the posterior alignment in Assumption 5.1 comes from the in-context ability of LLMs, which is widely studied in Lee et al. (2023); Wies et al. (2024); Xie et al. (2021). We also remark that Assumption C.3 does not mean that LLMs can make the optimal decision at each step naively in the underlying MDP: (1) Though the posterior distributions of state and reward are aligned, LLMs still need to be instructed to maximize the long-term value (via explicit planning) instead of the myopic reward. (2) LLMs still require online interactions to enlarge the in-context dataset $\mathcal{D}$ such that their prediction uncertainty can be reduced from the prior uncertainty. In Appendix D.6, we also discuss how to relax Assumption 5.1 to accommodate a generalization error in the regret bound derived by our analysis, where we assume that LLMs are MLEs on the pretraining dataset with uniform coverage. Based on Assumption 5.1, we prove that LLMs with posterior alignments perform BMA in model estimation in the following proposition.

Proposition C.4 (LLMs with Posterior Alignments Perform BMA)**.** *Under Assumption C.3, the LLM predictions satisfy*

$$r_{LLM(\mathcal{D})}(s, a) + (P_{LLM(\mathcal{D})}V)(s, a) = \mathbb{E}_{\theta \sim \mathbb{P}_{post}(\cdot \,|\, \mathcal{D})}[(B_\theta V)(s, a)]$$

for any dataset $\mathcal{D} = \{(s_i, a_i, r_i, s_i')\}_{i=0}^{I}$ with size I, value function V, and query state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$. Here, $\mathbb{P}_{post}(\theta \,|\, \mathcal{D})$ is the posterior of $\theta^\star$ given $\mathcal{D}$ in the underlying MDP.

Proof of Proposition C.4. See the detailed proof in Appendix D.1. □

The proof of Proposition 5.2 can be found in Appendix D.1. Some variants of Proposition 5.2 can be found in various literature (Lee et al., 2023; Zhang et al., 2022; 2023b). In particular, Zhang et al. (2022) establish the theoretical equivalence between BMA and the ideal attention architecture and analyze the generalization error rate of LLMs. By Proposition 5.2, LLMs can provide a more certain and accurate estimate for the data-generating model with more collected feedback, as the uncertainty in the posterior is reduced with more data. Thus, Proposition 5.2 supports Claim 2.1 in theory.

C.4. Regret Bound of RAFA

To analyze RAFA in theory, we propose the theoretical version of RAFA in Algorithm 6, where we instantiate the switching condition of RAFA in Line 10 by measuring the reduction of the posterior entropy. At the t-th step and the k-th switching times, Algorithm 8 only makes the $(k+1)$-th switch when the reduction of posterior entropy $H_{t_k} - H_t$is greater than one bit. In Line 6 of Algorithm 6, we describe the planning subroutine in RAFA (Algorithm 1) by an ϵ-planner $\mathtt{PL}^\epsilon$ (defined in Definition C.1). We specify the terminating condition for Algorithm 6. Let $(K-1)$ be the total number of switches until t reaches $(T-1)$. Let $t_K = T$. At the $(T-1)$-th step, Algorithm 6 executes $a_{T-1} = \pi_{T-1}(s_{T-1})$, where we have $\pi_{T-1} = \mathtt{PL}^\epsilon(P_{\mathtt{LLM}(\mathcal{D}_{t_{K-1}})}, r_{\mathtt{LLM}(\mathcal{D}_{t_{K-1}})})$. Upon receiving r_{T-1} and s_T from the external environment, Algorithm 6 updates $\mathcal{D}_T = \{(s_t, a_t, s_{t+1}, r_t)\}_{t=0}^{T-1}$ and terminates. Since the agent in Algorithm 6 executes the same policy until making a switch, we have $\pi_t = \pi_{t_k}$ for any $t_k \le k < t_{k+1}$. We denote by $\pi^k = \pi_{t_k}$ for the notational simplicity. Next, we impose a regularity assumption on the structure of MDPs to measure the learning difficulty. Recall that we define the posterior entropy H_t in (2.4), the information gain $I(\theta; \xi \,|\, \mathcal{D})$, and $\xi_{(s,a)}$ as the pair of the next state and the current reward (s', r) given the query state-action pair (s, a) in Section 2. Define H_t as the posterior entropy $H(\theta \,|\, \mathcal{D}_t)$ given the dataset $\mathcal{D}_t = \{(s_i, a_i, r_i, s_{i+1})\}_{i=0}^{t-1}$.

Assumption C.5 (MDPs Regularity)**.** *We assume that there exists a coefficient $\eta > 0$ such that, if $H_{t_1} - H_{t_2} \le \log 2$, then it holds that*

$$I(\theta; \xi_{(s,a)} \,|\, \mathcal{D}_{t_1}) \le 4\eta \cdot I(\theta; \xi_{(s,a)} \,|\, \mathcal{D}_{t_2})$$

Algorithm 6 Reason for future, act for now (RAFA): The theoretical version.

input: An ϵ-optimal planner $\texttt{PL}^{\epsilon}$, which returns an ϵ-optimal policy that maximizes the value function up to an ϵ accuracy (Definition C.1), and the LLMs with posterior alignments.

initialization: Sample the initial state $s_0 \sim \rho$, set $t = 0$, and initialize the memory buffer $\mathcal{D}_0 = \varnothing$.

for $k = 0, 1, \ldots,$ **do**

  Set $t_k \leftarrow t$.

  repeat

    Planning ahead with the ϵ-optimal planner and LLMs $(\pi_t, V_t) \leftarrow \texttt{PL}^{\epsilon}(P_{\texttt{LLM}(\mathcal{D}_{t_k})}, r_{\texttt{LLM}(\mathcal{D}_{t_k})})$. ("reason for future")

    Execute action $a_t = \pi_t(s_t)$ to receive reward r_t and state s_{t+1} from environment. ("act for now")

    Update the memory buffer $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(s_t, a_t, s_{t+1}, r_t)\}$.

    Set $t \leftarrow t + 1$.

  until $H_{t_k} - H_t > \log 2$, where H_t denotes the posterior entropy of $\theta^\star$ conditioned on $\mathcal{D}_t$. (the switching condition is satisfied)

end for

for any given value function V, $t_1 < t_2$ and $(s, a) \in \mathcal{S} \times \mathcal{A}$.

Assumption C.5 is a regularity assumption on MDPs and is intrinsic to the agent design. In Appendix F, we prove that d-dimensional Bayesian linear kernel MDPs (defined in Definition F.1), satisfy Assumption C.5 with the coefficient $\eta = d/\log(1+d)$. Intuitively, Assumption C.5 restricts the increase of the information gain given one bit ($\log 2$) reduction of the posterior entropy.

Similar to other theoretical work on deep RL (Lazaric et al., 2010; Fan et al., 2020; Zhang et al., 2020), we introduce the concentrability coefficient κ to bound the distribution shift between the current policy and the optimal policy. For the simplicity of discussions, we define the optimal γ-discounted visitation measure $\nu^\star$ starting from state s as

$$\nu^\star(s' \,|\, s) = \frac{1}{1-\gamma} \cdot \sum_{\tau=0}^{\infty} \gamma^\tau \cdot \mathbb{P}\big(s_\tau = s' \big|\, s_0 = s, s_{i+1} \sim P_{\theta^\star}(\cdot \,|\, s_i, \pi^\star(s_i)) \text{ for any } 0 \le i < \tau\big), \tag{C.2}$$

for any state $s, s' \in \mathcal{S}$. Here, $\nu^\star(\cdot \,|\, s)$ describes the discounted average probability measure of the state that the optimal policy $\pi^\star$ visits starting from state s in the current infinite horizon MDP. Now, we are ready to provide the full statement of the concentrability coefficient as follows.

Assumption C.6 (Concentrability Coefficient)**.** *For RAFA (Algorithm 6), we assume that there exists a constant $\kappa < \infty$ such that*

$$\mathbb{E}\left[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\left[\sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}_{\theta^\star \sim \mathbb{P}_{t_k}}\left[\frac{\mathbb{E}_{s \sim \nu^\star(\cdot \,|\, s_t)}\big[\big((B_k - B_{\theta^\star})V_t\big)^2(s, \pi^\star(s))\big]}{\big((B_k - B_{\theta^\star})V_t\big)^2(s_t, \pi^k(s_t))} \,\middle|\, \mathcal{D}_{t_k}\right]\right]\right]$$

is bounded by $\kappa^2 \cdot T$, where we define $(B_k V)(s,a) = r_{\texttt{LLM}(\mathcal{D}_{t_k})}(s,a) + \gamma \cdot (P_{\texttt{LLM}(\mathcal{D}_{t_k})} V)(s,a)$.

Intuitively, κ measures the hardness to generalize the low prediction error $(B_k - B_{\theta^\star})V_t$ on the current trajectory induced by $\{\pi^k\}_{k=0}^{K-1}$ to the optimal trajectory induced by $\pi^\star$ in the underlying MDP. We remark that we can drop the dependency of the concentrability coefficient κ (Assumption C.6) if we modify RAFA to encourage efficient exploration in MDPs. We will discuss the variants of RAFA with efficient exploration strategies in Appendix C.5.

In the following theorem, we give the bound of the Bayesian regret of RAFA (Algorithm 6) as follows.

Theorem C.7. *Under Assumptions C.3, C.5 , and C.6,the Bayesian regret of* RAFA *(Algorithm 6) satisfies*

$$\mathfrak{R}(T) = \mathcal{O}\left(\frac{(\kappa+1)L\cdot\sqrt{\mathbb{E}[H_0 - H_T]}}{1-\gamma}\cdot\sqrt{T} + \frac{\epsilon}{1-\gamma}\cdot T + \frac{L\cdot\mathbb{E}[H_0 - H_T]}{1-\gamma}\right),$$

where κ is the concentrability coefficient defined in Assumption C.6, H_t is the posterior entropy of $\theta^\star$ given the history $\mathcal{D}_t = \{(s_i, a_i, r_i, s_{i+1})\}_{i=0}^{t-1}$, and L is the bound of $|r + V(s)|$ for any reward r, state s, and value V.

Proof of Theorem C.7. See the detailed proof in Appendix D.3. □

Theorem C.10 establishes the $\sqrt{T}$ regret of RAFA (Algorithm 6) for a proper choice of the planning suboptimality ϵ, e.g., $\epsilon = \mathcal{O}(1/\sqrt{T})$. Here, the first term in the upper bound in Theorem C.10 is the leading term and involves several multiplicative factors, namely the effective horizon $1/(1-\gamma)$, the value bound L, and the cumulative posterior entropy reduction $H_0 - H_T$ throughout the T steps, which are common in the RL literature (Abbasi-Yadkori & Szepesvári, 2015; Osband et al., 2013; Russo & Van Roy, 2014b;a; 2016; Lu & Van Roy, 2019). In particular, H_0 highlights the prior knowledge obtained through pretraining, as H_0 quantifies the prior uncertainty of LLMs before incorporating any collected feedback. Hence, $H_0 - H_T$ highlights the uncertainty reduction achieved by reasoning and acting, as H_T quantifies the posterior uncertainty of LLMs after incorporating the collected feedback. In Appendix F, we prove that $H_0 - H_T = \mathcal{O}(d\cdot\log T)$ and the $1-\delta$ probability bound on value functions $L = \mathcal{O}(\sqrt{d}\cdot\log(dT/\delta))$ for the d-dimensional Bayesian linear kernel MDPs, which implies $\mathfrak{R}(T) = \widetilde{\mathcal{O}}((1-\gamma)^{-1}(\kappa+1)\cdot\sqrt{d^3T})$ with probability at least $1-\delta$. Here $\widetilde{\mathcal{O}}$ hides the logarithmic factor.

C.5. RAFA with Efficient Exploration Strategies

In this section, we provide two variants of RAFA (Algorithm 6): (1) RAFA with optimistic bonus (Algorithm 7) and (2) RAFA with posterior sampling (Algorithm 8). We also prove the bound of the Bayesian regret of each variant, which demonstrates the effectiveness of the efficient exploration strategies.

C.5.1. Optimistic Bonus.

We incorporate the *Optimism in Face of Uncertainty* (OFU) principle to encourage efficient exploration. We design the optimistic bonus by the information gain and implement a variant of RAFA in Algorithm 7. In particular, the bonus $\Gamma_k(s,a)$ takes the following form

$$\Gamma_k(s,a) = \sqrt{2}L\cdot\sqrt{I(\theta;\xi_{(s,a)}\,|\,\mathcal{D}_{t_k})} \tag{C.3}$$

for any $(s,a)\in\mathcal{S}\times\mathcal{A}$ and $k<K$. In Line 7 of Algorithm 7, we generate the policy π^t by $\texttt{PL}^\epsilon(P_{\texttt{LLM}(\mathcal{D}_{t_k})}, r_{\texttt{LLM}(\mathcal{D}_{t_k})} + \Gamma_k)$ for any $t_k \le t < t_{k+1}$. Intuitively, the bonus is higher at the state-action pair with higher information gain, which incentivizes the agent to explore those less visited states (with higher information gain). The design of optimistic bonus is popular in RL literature (Cai et al., 2020; Zhou et al., 2021b; Jin et al., 2020; Liu et al., 2022b; Wang et al., 2023c) as it can drop the concetrablility coefficient κ in the regret bound. In the following theorem, we prove the regret bound of RAFA with optimistic bonus (Algorithm 7).

Theorem C.8. *Under Assumptions C.3 and C.5, the Bayesian regret of* RAFA *with optimistic bonus (Algorithm 7) satisfies*

$$\mathfrak{R}(T) = \mathcal{O}\left(\frac{L\cdot\sqrt{\mathbb{E}[H_0 - H_T]}}{1-\gamma}\cdot\sqrt{T} + \frac{\epsilon}{1-\gamma}\cdot T + \frac{L\cdot\mathbb{E}[H_0 - H_T]}{1-\gamma}\right),$$

where all the variables have the same definitions in Theorem C.7.

Proof of Theorem C.8. See the detailed proof in Appendix D.4. □

Algorithm 7 Reason for future, act for now (RAFA): The theoretical version with optimistic bonus.

input: An ϵ-optimal planner $\texttt{PL}^{\epsilon}$, which returns an ϵ-optimal policy that maximizes the value function up to an ϵ accuracy (Definition C.1), and the LLMs with posterior alignments.

initialization: Sample the initial state $s_0 \sim \rho$, set $t = 0$, and initialize the memory buffer $\mathcal{D}_0 = \varnothing$.

for $k = 0, 1, \ldots,$ **do**

  Set $t_k \leftarrow t$.

  repeat

    Design optimistic bonus $\Gamma_k(s,a) = \sqrt{2}L \cdot \sqrt{I(\theta; \xi_{(s,a)} \,|\, \mathcal{D}_{t_k})}$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$.

    Planning ahead with the ϵ-optimal planner and LLMs $(\pi_t, V_t) \leftarrow \texttt{PL}^{\epsilon}(P_{\texttt{LLM}(\mathcal{D}_{t_k})}, r_{\texttt{LLM}(\mathcal{D}_{t_k})} + \Gamma_k)$.

    ("reason for future")

    Execute action $a_t = \pi_t(s_t)$ to receive reward r_t and state s_{t+1} from environment. ("act for now")

    Update memory $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(s_t, a_t, s_{t+1}, r_t)\}$.

    Set $t \leftarrow t + 1$.

  until $H_{t_k} - H_t > \log 2$, where H_t denotes the posterior entropy of $\theta^\star$ conditioned on $\mathcal{D}_t$.

  (the switching condition is satisfied)

end for

Compared with Theorem C.7, the regret bound in Theorem C.8 is not dependent on the concentrability coefficient κ, which demonstrates of effectiveness of efficient exploration in Algorithm 7. In Appendix F, we prove that $H_0 - H_T = \mathcal{O}(d \cdot \log T)$ and the $1-\delta$ probability bound on value functions $L = \mathcal{O}(\sqrt{d} \cdot \log(dT/\delta))$ for the d-dimensional Bayesian linear kernel MDPs, which implies $\mathfrak{R}(T) = \widetilde{\mathcal{O}}((1-\gamma)^{-1} \cdot \sqrt{d^3 T})$ with probability at least $1-\delta$. Here $\widetilde{\mathcal{O}}$ hides the logarithmic factor.

C.5.2. POSTERIOR SAMPLING.

Algorithm 8 Reason for future, act for now (RAFA): The theoretical version with posterior sampling.

input: An ϵ-optimal planner $\texttt{PL}^{\epsilon}$, which returns an ϵ-optimal policy that maximizes the value function up to an ϵ accuracy (Definition C.1), and the LLMs satisfying Assumption C.9.

initialization: Sample the initial state $s_0 \sim \rho$, set $t = 0$, and initialize the memory buffer $\mathcal{D}_0 = \varnothing$.

for $k = 0, 1, \ldots,$ **do**

  Set $t_k \leftarrow t$.

  repeat

    Planning ahead with the ϵ-optimal planner and the posterior sampling mechanism of LLMs (defined in Assumption C.9) $(\pi_t, V_t) \leftarrow \texttt{PL}^{\epsilon}(P_{\texttt{LLM+PS}(\mathcal{D}_{t_k})}, r_{\texttt{LLM+PS}(\mathcal{D}_{t_k})})$.

    ("reason for future")

    Execute action $a_t = \pi_t(s_t)$ to receive reward r_t and state s_{t+1} from environment. ("act for now")

    Update memory $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(s_t, a_t, s_{t+1}, r_t)\}$.

    Set $t \leftarrow t + 1$.

  until $H_{t_k} - H_t > \log 2$, where H_t denotes the posterior entropy of $\theta^\star$ conditioned on $\mathcal{D}_t$.

  (the switching condition is satisfied)

end for

As another method for efficient exploration, we assume there exists a mechanism that deploys posterior sampling and we use this mechanism to encourage exploration for RAFA.

Assumption C.9 (LLMs with Posterior Sampling Mechanism)**.** *We assume that there exists a mechanism* LLM+PS *that maps the memory buffer to the transition kernel and the reward, such that* $(r_{\texttt{LLM+PS}(\mathcal{D})}(s,a) + \gamma \cdot (P_{\texttt{LLM+PS}(\mathcal{D})}V)(s,a)) \,|\, \mathcal{D}$ *and* $(B_{\theta^\star}V(s,a)) \,|\, \mathcal{D}$ *are identically independent distributed for any* $(s,a) \in \mathcal{S} \times \mathcal{A}$*, in-context dataset* $\mathcal{D}$*, and value function* V*. Here,* $\theta^\star$ *is the data-generating parameter.*

We remark that the bootstrap method (Efron, 1982) can approximate the mechanism in Assumption C.9. Widely used in applied statistics (Davison & Hinkley, 1997) and the design of RL algorithms (Osband et al., 2016; Hao et al., 2019), the bootstrap method takes an in-context dataset $\mathcal{D}$ and a functional estimator g as inputs. Depending on the configuration of bootstrap, we generate the bootstrapped dataset $\widetilde{\mathcal{D}}$ from $\mathcal{D}$ by uniform sampling with replacement (Efron, 1982) or weighted sampling with replacement (Newton & Raftery, 1994). Selecting LLMs as the functional estimator g and the memory buffer $\mathcal{D}$ as the dataset D, we can use this bootstrap method to approximate the mechanism LLM+PS that is introduced in Assumption C.9. From the statistics literature (Bickel & Freedman, 1981; Singh, 1981; Newton & Raftery, 1994), we also know that bootstrap distribution recovers the posterior distribution asymptotically.

Based on the mechanism introduced in Assumption C.9, we propose a variant of RAFA in Algorithm 8, where we use the mechanism LLM+PS as the model estimator in the learning subroutine of RAFA. In Line 7 of Algorithm 7, we generate the policy π^t by $\texttt{PL}^{\epsilon}(P_{\texttt{LLM+PS}(\mathcal{D}_{t_k})}, r_{\texttt{LLM+PS}(\mathcal{D}_{t_k})})$. In the following, we give a simple explanation of how this mechanism helps the agent to efficiently explore. By the Bayes' rule, we have $p(\theta \,|\, \mathcal{D}) \propto L(\mathcal{D} \,|\, \theta)\mathbb{P}_0(\theta)$, where $L(\mathcal{D} \,|\, \theta)$ is the likelihood of $\mathcal{D}$ given θ and $\mathbb{P}_0$ is the prior of θ. Taking the logarithm, we have $\log(p(\theta \,|\, \mathcal{D})) = c + \log(\mathbb{P}_0(\theta)) + \log(L(\mathcal{D} \,|\, \theta))$ for a constant c. Hence, the uncertainty of the posterior is higher ($p(\theta \,|\, \mathcal{D})$ is closer to 0) at the less visited states (the likelihood of these states is closer to 0). Hence, if we sample the model estimator from the posterior, the agent is more exploratory at the less visited states, which explains why the mechanism LLM+PS helps RAFA efficiently explore.

In the following theorem, we prove the regret bound of RAFA with posterior sampling (Algorithm 8).

Theorem C.10 (Bayesian Regret)**.** Under Assumptions C.5 and C.9, the Bayesian regret of RAFA with posterior sampling (Algorithm 8) satisfies

$$\mathfrak{R}(T) = \mathcal{O}\left(\frac{L \cdot \sqrt{\mathbb{E}[H_0 - H_T]}}{1-\gamma} \cdot \sqrt{T} + \frac{\epsilon}{1-\gamma} \cdot T + \frac{L \cdot \mathbb{E}[H_0 - H_T]}{1-\gamma}\right),$$

where all the variables have the same definitions in Theorem C.7.

Proof of Theorem C.10. See the detailed proof in Appendix D.5. □

Compared with Theorem C.7, the regret bound in Theorem C.10 is not dependent on the concentrability coefficient κ, which demonstrates of effectiveness of efficient exploration in Algorithm 8. In Appendix F, we prove that $H_0 - H_T = \mathcal{O}(d \cdot \log T)$ and the $1-\delta$ probability bound on value functions $L = \mathcal{O}(\sqrt{d} \cdot \log(dT/\delta))$ for the d-dimensional Bayesian linear kernel MDPs, which implies $\mathfrak{R}(T) = \widetilde{\mathcal{O}}((1-\gamma)^{-1} \cdot \sqrt{d^3 T})$ with probability at least $1-\delta$. Here $\widetilde{\mathcal{O}}$ hides the logarithmic factor.

D. Main Proofs

D.1. LLMs with Posterior Alignments Perform BMA

Proof of Proposition C.4. Recall that $P_{\texttt{LLM}(\mathcal{D})}$ and $r_{\texttt{LLM}(\mathcal{D})}$ are the estimated transition kernel and reward induced by $P^{\texttt{LLM}}$ that satisfies Assumption C.3. For any query state-action pair (s,a) and in-context dataset $\mathcal{D}$, it holds that

$$\begin{aligned}
(P_{\texttt{LLM}(\mathcal{D})}V)(s,a) &= \int_{\mathcal{S}} V(s') P_{\texttt{LLM}(\mathcal{D})}(\mathrm{d}s' \,|\, s,a) \\
&= \int_{\mathcal{S}} V(s') \left(\int_{\Theta} P_\theta(\mathrm{d}s' \,|\, s,a) P_{\text{post}}(\mathrm{d}\theta \,|\, \mathcal{D})\right) \\
&= \int_{\Theta} P_{\text{post}}(\mathrm{d}\theta \,|\, \mathcal{D}) \left(\int_{\mathcal{S}} P_\theta(\mathrm{d}s' \,|\, s,a) V(s')\right) \\
&= \mathbb{E}_{\theta \sim \mathbb{P}_{\text{post}}(\cdot \,|\, \mathcal{D})}[(P_\theta V)(s,a)], \qquad \text{(D.1)}
\end{aligned}$$

where the second equality uses Assumption C.3 (Posterior Alignment), the third equality uses Fubin's theorem, and the last equality uses (2.3). For any query state-action pair (s,a) and in-context dataset $\mathcal{D}$, it holds that

$$\begin{aligned}
r_{\texttt{LLM}(\mathcal{D})}(s,a) &= \mathbb{E}_{P^{\texttt{LLM}}}[r \,|\, \mathcal{D}, s, a] \\
&= \mathbb{E}_{P_{\text{post}}}[r \,|\, \mathcal{D}, s, a] \\
&= \mathbb{E}_{\theta \sim \mathbb{P}_{\text{post}}(\cdot \,|\, \mathcal{D})}[r_\theta(s,a)], \qquad \text{(D.2)}
\end{aligned}$$

where the second equality uses Assumption C.3 (Posterior Alignment) and the last equality uses (2.3). By the linearity of expectation, we combine (D.1) and (D.2) to obtain

$$\begin{aligned}
r_{\texttt{LLM}(\mathcal{D})}(s,a) + (P_{\texttt{LLM}(\mathcal{D})}V)(s,a) &= \mathbb{E}_{\theta \sim \mathbb{P}_{\text{post}}(\cdot \,|\, \mathcal{D})}[r_\theta(s,a) + (P_\theta V)(s,a)] \\
&= \mathbb{E}_{\theta \sim \mathbb{P}_{\text{post}}(\cdot \,|\, \mathcal{D})}[(B_\theta V)(s,a)],
\end{aligned}$$

where the last equality uses the definition of B_θ. Thus, we finish the proof of Proposition C.4. □

D.2. Contraction Property of the Posterior Variance

Proposition D.1 (Contraction Property of the Posterior Variance)**.** *Under Assumptions C.5, the posterior variance in Algorithms 6, 7, and 8 satisfies the following two properties:*

$$\textit{(i)} \qquad \mathbb{V}_{\theta\sim\mathbb{P}_{t_k}}\left[(B_\theta V_t)(s,a)\middle|\mathcal{D}_{t_k}\right] \le 2L^2 \cdot I(\theta;\xi_{(s,a)}\,|\,\mathcal{D}_{t_k})$$

$$\textit{(ii)} \qquad \mathbb{E}\left[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\left[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{V}_{\theta\sim\mathbb{P}_{t_k}}\left[(B_\theta V_t)(s_t,a_t)\middle|\mathcal{D}_{t_k}\right]\right]\right] \le 8\eta L^2\cdot\mathbb{E}[H_0 - H_T],$$

where we denote the upper bound of the sum of any value function and the reward by a positive constant L, that is, $|r + V(s)| \le L$ for any reward r, state s, and estimated value function V.

Proof of Proposition D.1. We begin with the proof of the first property in Proposition D.1. Recall the definition that $\xi_{(s,a)}$ denotes random variables (s', r) in the underlying MDP given the current state s and action a. Define that $g_t(\xi_{(s,a)}) = (r + V_t(s'))/(2L)$. Since the sum of any reward and value function is bounded by L, we know that $|g_t| \le 1/2$. for any $t_k \le t < t_{k+1}$, we have

$$\begin{aligned} 2L\cdot\mathbb{E}[g_t(\xi_{(s,a)})\,|\,\theta,\mathcal{D}_{t_k}] &= (B_\theta V_t)(s,a) \\ 2L\cdot\mathbb{E}[g_t(\xi_{(s,a)})\,|\,\mathcal{D}_{t_k}] &= \mathbb{E}_{\theta\sim\mathbb{P}_t}\left[(B_\theta V_t)(s,a)\right], \end{aligned} \tag{D.3}$$

for any query state s and action a. By the variational form of total variation (TV) distance d_{TV}, we have

$$\begin{aligned} d_{\mathrm{TV}}^2(\mathbb{P}(\xi_{(s,a)}\,|\,\theta,\mathcal{D}_t)\|\mathbb{P}(\xi_{(s,a)}\,|\,\mathcal{D}_t)) &= \Big(\sup_{g:|g|\le 1/2}\mathbb{E}[g(\xi_{(s,a)})\,|\,\theta,\mathcal{D}_{t_k}] - \mathbb{E}[g(\xi_{(s,a)})\,|\,\mathcal{D}_{t_k}]\Big)^2 \\ &\ge \left(\mathbb{E}[g_t(\xi_{(s,a)})\,|\,\theta,\mathcal{D}_{t_k}] - \mathbb{E}[g_t(\xi_{(s,a)})\,|\,\mathcal{D}_{t_k}]\right)^2 \\ &= \frac{1}{4L^2}\cdot\left((B_\theta V_t)(s,a) - \mathbb{E}_{\theta\sim\mathbb{P}_t}\left[(B_\theta V_t)(s,a)\right]\right)^2, \end{aligned} \tag{D.4}$$

where the last equality is the result of (D.3). By taking the expectation with respect to $\theta\sim\mathbb{P}_t$ on (D.4), we have

$$\begin{aligned} \mathbb{V}_{\theta\sim\mathbb{P}_{t_k}}\left[(B_\theta V_t)(s,a)\middle|\mathcal{D}_{t_k}\right] &= \mathbb{E}_{\theta\sim\mathbb{P}_t}\left[\left((B_\theta V_t)(s,a) - \mathbb{E}_{\theta\sim\mathbb{P}_t}\left[(B_\theta V_t)(s,a)\right]\right)^2\right] \\ &\le 4L^2\cdot\mathbb{E}_{\theta\sim\mathbb{P}_t}\left[d_{\mathrm{TV}}^2(\mathbb{P}(\xi_{(s,a)}\,|\,\theta,\mathcal{D}_{t_k})\|\mathbb{P}(\xi_{(s,a)}\,|\,\mathcal{D}_{t_k}))\right] \\ &\le 2L^2\cdot\mathbb{E}_{\theta\sim\mathbb{P}_t}\left[d_{\mathrm{KL}}(\mathbb{P}(\xi_{(s,a)}\,|\,\theta,\mathcal{D}_{t_k})\|\mathbb{P}(\xi_{(s,a)}\,|\,\mathcal{D}_{t_k}))\right] \\ &= 2L^2\cdot\left(H(\xi_{(s,a)}\,|\,\mathcal{D}_{t_k}) - H(\xi_{(s,a)}\,|\,\theta,\mathcal{D}_{t_k})\right) \\ &= 2L^2\cdot I(\xi_{(s,a)};\theta\,|\,\mathcal{D}_{t_k}) \\ &= 2L^2\cdot I(\theta;\xi_{(s,a)}\,|\,\mathcal{D}_{t_k}), \end{aligned} \tag{D.5}$$

where the first equality uses the definition of variance, the first inequality uses (D.4) by taking the expectation with respect to $\theta\sim\mathbb{P}_t$, and the second inequality uses Pinsker's inequality. Here, the second equality uses the definition of entropy and the second last equality uses the definition of the information gain. Here, the last equality uses the fact that $I(X;Y) = I(Y;X)$ for any two random variables X and Y. Thus, we finish the proof of the first property in Proposition D.1.

Next, we prove the second property in Proposition D.1. By the fact that $a_t = \pi^k(s_t)$ for any $t_k \le t < t_{k+1}$, we have

$$\begin{aligned}
&\mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\big[\sum_{t=t_k}^{t_{k+1}-1} \mathbb{V}_{\theta\sim\mathbb{P}_{t_k}}\big[(B_\theta V_t)(s_t,a_t)\big|\mathcal{D}_{t_k}\big]\big]\Big] \\
&\quad= \mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\big[\sum_{t=t_k}^{t_{k+1}-1} \mathbb{V}_{\theta\sim\mathbb{P}_{t_k}}\big[(B_\theta V_t)(s_t,\pi^k(s_t))\big|\mathcal{D}_{t_k}\big]\big]\Big] \\
&\quad\le 2L^2\cdot\mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\big[\sum_{t=t_k}^{t_{k+1}-1} I(\theta;\xi_{(s_t,\pi^k(s_t))}\,|\,\mathcal{D}_{t_k})\big]\Big],
\end{aligned}$$

where the inequality invokes (D.5). Under Assumption C.5 and the same switching condition in Algorithms 6, 7, and 8, we have

$$\begin{aligned}
&\mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\big[\sum_{t=t_k}^{t_{k+1}-1} \mathbb{V}_{\theta\sim\mathbb{P}_{t_k}}\big[(B_\theta V_t)(s_t,a_t)\big|\mathcal{D}_{t_k}\big]\big]\Big] \\
&\quad\le 8\eta L^2\cdot\mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} I(\theta;\xi_{(s_t,\pi^k(s_t))}\,|\,\mathcal{D}_t)\Big]\Big] \\
&\quad\le 8\eta L^2(H_0 - H_T),
\end{aligned}$$

where the last inequality uses the chain rule of information gain. Thus, we finish the proof of the second property in Proposition D.1. □

D.3. Proof of Theorem C.7

Proof of Theorem C.7. Recall that we denote by $\pi^k = \pi_{t_k}$ and V_t is the estimated value function returned by the ϵ-optimal planner $\texttt{PL}^\epsilon$ in Algorithm 6. By the definition of the Bayesian regret $\mathfrak{R}(T)$ and the tower property of the conditional expectation, we have

$$\begin{aligned}
\mathfrak{R}(T) &= \mathbb{E}\Big[\sum_{k=0}^{K-1}\sum_{t=t_k}^{t_{k+1}-1} V_{\theta^\star}^{\pi^\star}(s_t) - V_{\theta^\star}^{\pi_t}(s_t)\Big] \\
&= \mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} V_{\theta^\star}^{\pi^\star}(s_t) - V_{\theta^\star}^{\pi^k}(s_t)\Big]\Big] \\
&= \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} V_t(s_t) - V_{\theta^\star}^{\pi^k}(s_t)\Big]\Big]}_{\text{term (a)}} + \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} V_{\theta^\star}^{\pi^\star}(s_t) - V_t(s_t)\Big]\Big]}_{\text{term (b)}},
\end{aligned} \tag{D.6}$$

where the first equality uses the fact that $\pi_t = \pi_{t_k}$ for any $t_k \le t < t_{k+1}$ in Algorithm 6. By Definition C.1, we know that $V_t(s) = Q_t(s,\pi^k(s))$ for any $t_k \le t < t_{k+1}$. Then, we introduce the following performance difference lemma to bound terms (a) and (b) in (D.6), respectively.

Lemma D.2 (Performance Difference)**.** For an algorithm $\texttt{ALG}$ with switching times K, estimated value functions $\{(Q_t,V_t)\}_{t=0}^{T-1}$, and the corresponding output policy $\{\pi^k\}_{k=0}^{K-1}$ for T-steps interaction. We assume that $\texttt{ALG}$ switches to the policy π^k at the t_k-th timestep for the k-th switch and $V_t(s) = Q_t(s,\pi^k(s))$ for any $s\in\mathcal{S}$ and $k<K$. Then, we have two parts of performance difference results for $\texttt{ALG}$ as follows,

- (Part I)

$$
\begin{aligned}
&(1-\gamma)\cdot\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} V_t(s_t)-V_{\theta^\star}^{\pi^k}(s_t)\Big]\Big]\\
&\quad=\underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} Q_t(s_t,a_t)-\big(B_{\theta^\star}V_t\big)(s_t,a_t)\Big]\Big]}_{\text{term (A): model prediction error}}\\
&\qquad+\underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\big(V_t(s_{t_{k+1}})-V_{\theta^\star}^{\pi^k}(s_{t_{k+1}})\big)-\big(V_t(s_{t_k})-V_{\theta^\star}^{\pi^k}(s_{t_k})\big)\Big]\Big]}_{\text{term (B): value inconsistency}},
\end{aligned}
\tag{D.7}
$$

- (Part II)

$$
\begin{aligned}
&(1-\gamma)\cdot\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} V_{\theta^\star}^{\pi^\star}(s_t)-V_t(s_t)\Big]\Big]\\
&\quad=\underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{s\sim\nu^\star(\cdot\,|\,s_t)}\big[(B_{\theta^\star}V_t)(s,\pi^\star(s))-Q_t(s,\pi^\star(s))\big]\Big]\Big]}_{\text{term (A): model prediction error}}\\
&\qquad+\underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{s'\sim\nu^\star(\cdot\,|\,s_t)}\big[(Q_t(s,\pi^\star(s))-Q_t(s,\pi^k(s))\big]\Big]\Big]}_{\text{term (B): optimization error}}.
\end{aligned}
\tag{D.8}
$$

where $\mathbb{E}_{\pi^k}$ is taken with respect to the state-action sequence following $s_{t+1}\sim P_{\theta^\star}(\cdot\,|\,s_t,a_t)$ and $a_t=\pi^k(s_t)$ for any $t_k\le t<t_{k+1}$, while $\mathbb{E}$ is taken with respect to the prior distribution $\mathbb{P}_0$ of $\theta^\star$, the iterative update of π^k, and the randomness of $\{(Q_t,V_t)\}_{t=0}^{T-1}$. Here, the optimal γ-discounted visitation measure $\nu^\star$ is defined in (C.2).

Proof of Lemma D.2. See the detailed proof in Appendix E.1. □

By the first part of Lemma D.2, we analyze term (a) as follows,

$$
\begin{aligned}
(1-\gamma)\cdot\text{term (a)}=&\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} Q_t(s_t,a_t)-\big(B_{\theta^\star}V_t\big)(s_t,a_t)\Big]\Big]\\
&+\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\big(V_t(s_{t_{k+1}})-V_{\theta^\star}^{\pi^k}(s_{t_{k+1}})\big)-\big(V_t(s_{t_k})-V_{\theta^\star}^{\pi^k}(s_{t_k})\big)\Big]\Big].
\end{aligned}
\tag{D.9}
$$

Recall that we define $(B_kV)(s,a)=r_{\texttt{LLM}(\mathcal{D}_{t_k})}(s,a)+(P_{\texttt{LLM}(\mathcal{D}_{t_k})}V)(s,a)$ for any (s,a) and value V. By the definition of ϵ-optimal planner (Definition C.1) and the planning procedure $(\pi_t,V_t)\leftarrow\texttt{PL}^\epsilon(P_{\texttt{LLM}(\mathcal{D}_{t_k})},r_{\texttt{LLM}(\mathcal{D}_{t_k})})$ in Line 6 of Algorithm 6, we have

$$
\begin{aligned}
|Q_t(s_t,a_t)-\big(B_{\theta^\star}V_t\big)(s_t,a_t)|&\le\epsilon+\big((B_k-B_{\theta^\star})V_t\big)(s_t,a_t)\\
&=\epsilon+\big|\big((B_k-B_{\theta^\star})V_t\big)(s_t,a_t)\big|
\end{aligned}
\tag{D.10}
$$

for any $t_k \le t < t_{k+1}$. Then, we plug (D.10) into (D.9) to obtain

$$(1-\gamma)\cdot \text{term (a)} \le \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\big|\big((B_k - B_{\theta^\star})V_t\big)(s_t,a_t)\big|\Big]\Big]}_{\text{term (a1)}} + \epsilon\cdot T$$
$$+ \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\big(V_t(s_{t_{k+1}}) - V_{\theta^\star}^{\pi^k}(s_{t_{k+1}})\big) - \big(V_t(s_{t_k}) - V_{\theta^\star}^{\pi^k}(s_{t_k})\big)\Big]\Big]}_{\text{term (a2)}}. \tag{D.11}$$

Under Assumption C.3 and C.5, we have

$$\begin{aligned}
\text{term (a1)} &= \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{\theta\sim\mathbb{P}_{t_k}}\Big[\big|\big((B_k-B_\theta)V_t\big)(s_t,a_t)\big|\Big|\mathcal{D}_{t_k}\Big]\Big]\Big] \\
&\le \sqrt{T}\cdot\Big(\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{\theta\sim\mathbb{P}_{t_k}}\Big[\big|\big((B_k-B_\theta)V_t\big)(s_t,a_t)\big|^2\Big|\mathcal{D}_{t_k}\Big]\Big]\Big]\Big)^{1/2} \\
&= \sqrt{T}\cdot\Big(\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{V}_{\theta\sim\mathbb{P}_{t_k}}\big[(B_kV_t)(s_t,a_t)\big|\mathcal{D}_{t_k}\big]\Big]\Big]\Big)^{1/2},
\end{aligned} \tag{D.12}$$

where the first equality uses the tower property of the conditional expectation and the definition that the posterior distribution of $\theta^\star$ given $\mathcal{D}_{t_k}$ is $\mathbb{P}_{t_k}$. Here, the first inequality uses Cauchy-Schwarz inequality and the last equality invokes Proposition C.4 and the definition of variance. Under Assumption C.5, we apply the second property in Proposition D.1 on the right-hand side of (D.12) to have

$$\text{term (a1)} \le 2\sqrt{2\eta}L\cdot\sqrt{\mathbb{E}[H_0 - H_T]}\cdot\sqrt{T}. \tag{D.13}$$

Since any value function is bounded by L, we bound term (a2) in (D.11) as follows,

$$\text{term (a2)} \le 4L\cdot\ \mathbb{E}[K]. \tag{D.14}$$

To characterize the upper bound of the switching times K, we introduce the following lemma.

Lemma D.3 (Upper bound of Switching Times)**.** *If $H_{t_k} - H_{t_{k+1}} \ge \log 2$ for any $k < K$, then it holds that*

$$K - 1 \le (H_0 - H_{t_{K-1}})/\log 2 \le (H_0 - H_T)/\log 2.$$

Proof of Lemma D.3. Since $H_{t_k} - H_{t_{k+1}} \ge \log 2$, we have

$$H_0 - H_{t_{K-1}} = \sum_{k=0}^{K-2} H_{t_k} - H_{t_{k+1}} \ge (K-1)\cdot\log 2,$$

which implies

$$K - 1 \le (H_0 - H_{t_{K-1}})/\log 2 \le (H_0 - H_T)/\log 2.$$

Thus, we finish the proof of Lemma D.3. □

As the switching condition in Algorithm 8 implies $H_{t_k} - H_{t_{k+1}} \geq \log 2$, we apply Lemma D.3 to have

$$\mathbb{E}[K] \leq 1 + \mathbb{E}[H_0 - H_{t_{K-1}}]/\log 2 \leq (H_0 - H_T)/\log 2. \tag{D.15}$$

Combining (D.14) and (D.15), we upper bound term (a2) in (D.11) as

$$\text{term (a2)} \leq 4L \cdot \mathbb{E}[K] \leq 4L + \frac{4L \cdot \mathbb{E}[H_0 - H_T]}{\log 2}. \tag{D.16}$$

Plugging (D.13) and (D.16) into (D.11), we have

$$(1-\gamma) \cdot \text{term (a)} \leq 2\sqrt{2\eta}L \cdot \sqrt{\mathbb{E}[H_0 - H_T]} \cdot \sqrt{T} + \epsilon \cdot T + 4L + \frac{4L \cdot \mathbb{E}[H_0 - H_T]}{\log 2}. \tag{D.17}$$

Then, we invoke the second part of Lemma D.2 to decompose term (b) in (D.6) as follows,

$$\begin{aligned}
(1-\gamma) \cdot \text{term (b)} &= \mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}_{s\sim\nu^\star(\cdot\,|\,s_t)}\big[\big(B_{\theta^\star}V_t\big)(s,\pi^\star(s)) - Q_t(s,\pi^\star(s))\big]\Big]\Big] \\
&\quad + \mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}_{s\sim\nu^\star(\cdot\,|\,s_t)}\big[Q_t(s,\pi^\star(s)) - Q_t(s,\pi^k(s))\big]\Big]\Big] \\
&\leq \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}_{s\sim\nu^\star(\cdot\,|\,s_t)}\big[\big|\big((B_{\theta^\star} - B_k)V_t\big)(s,\pi^\star(s))\big|\big]\Big]\Big]}_{\text{term (b1)}} + \epsilon \cdot T \\
&\quad + \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}_{s\sim\nu^\star(\cdot\,|\,s_t)}\big[Q_t(s,\pi^\star(s)) - Q_t(s,\pi^k(s))\big]\Big]\Big]}_{\text{term (b2)}},
\end{aligned} \tag{D.18}$$

where the inequality uses (D.10). For term (b1) in (D.18), we use the tower property of the conditional expectation to obtain

$$\begin{aligned}
\text{term (b1)} &= \mathbb{E}\Bigg[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\Bigg[\sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}_{\theta^\star\sim\mathbb{P}_{t_k}}\Bigg[\frac{\mathbb{E}_{s\sim\nu^\star(\cdot\,|\,s_t)}\big[\big|\big((B_k - B_{\theta^\star})V_t\big)(s,\pi^\star(s))\big|\big]}{\big((B_k - B_{\theta^\star})V_t\big)(s_t,\pi^k(s_t))} \cdot \big((B_k - B_{\theta^\star})V_t\big)(s_t,\pi^k(s_t))\Big|\mathcal{D}_{t_k}\Bigg]\Bigg]\Bigg] \\
&= \mathbb{E}\Bigg[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\Bigg[\sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}_{\theta^\star\sim\mathbb{P}_{t_k}}\Big[G_{t,k}(\theta^\star) \cdot \big((B_k - B_{\theta^\star})V_t\big)(s_t,\pi^k(s_t))\Big|\,\mathcal{D}_{t_k}\Big]\Bigg]\Bigg],
\end{aligned} \tag{D.19}$$

where we define

$$G_{t,k}(\theta^\star) = \frac{\mathbb{E}_{s\sim\nu^\star(\cdot\,|\,s_t)}\big[\big|\big((B_k - B_{\theta^\star})V_t\big)(s,\pi^\star(s))\big|\big]}{\big((B_k - B_{\theta^\star})V_t\big)(s_t,\pi^k(s_t))}$$

and $\mathbb{E}_{\theta^\star\,|\,\mathcal{D}_{t_k}}[\cdot] = \mathbb{E}_{\theta^\star\sim\mathbb{P}_{t_k}}[\cdot\,|\,\mathcal{D}_{t_k}]$ for notational simplicity.

Applying Cauchy Schwartz inequality on the left-hand side of (D.19) several times, we have

$$\begin{aligned}
\text{term (b1)} &\le \mathbb{E}\Bigg[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Bigg[\sum_{t=t_k}^{t_{k+1}-1}\Big(\mathbb{E}_{\theta^\star\,|\,\mathcal{D}_{t_k}}\big[G_{t,k}^2(\theta^\star)\big]\Big)^{1/2}\cdot\Big(\mathbb{E}_{\theta^\star\,|\,\mathcal{D}_{t_k}}\Big[\big|\big((B_k-B_{\theta^\star})V_t\big)(s_t,\pi^k(s_t))\big|^2\Big]\Big)^{1/2}\Bigg]\Bigg]\\
&\le \mathbb{E}\Bigg[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Bigg[\Big(\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{\theta^\star\,|\,\mathcal{D}_{t_k}}\big[G_{t,k}^2(\theta^\star)\big]\Big)^{1/2}\cdot\Big(\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{\theta^\star\,|\,\mathcal{D}_{t_k}}\Big[\big|\big((B_k-B_{\theta^\star})V_t\big)(s_t,\pi^k(s_t))\big|^2\Big]\Big)^{1/2}\Bigg]\Bigg]\\
&\le \Bigg(\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{\theta^\star\,|\,\mathcal{D}_{t_k}}\big[G_{t,k}^2(\theta^\star)\big]\Big]\Big]\Bigg)^{1/2}\\
&\quad\cdot\Bigg(\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{\theta^\star\,|\,\mathcal{D}_{t_k}}\Big[\big|\big((B_k-B_{\theta^\star})V_t\big)(s_t,\pi^k(s_t))\big|^2\Big]\Big]\Big]\Bigg)^{1/2}\\
&\le \sqrt{\kappa^2\cdot T}\cdot\Bigg(\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{\theta^\star\,|\,\mathcal{D}_{t_k}}\Big[\big|\big((B_k-B_{\theta^\star})V_t\big)(s_t,\pi^k(s_t))\big|^2\Big]\Big]\Big]\Bigg)^{1/2},
\end{aligned}$$

where the first three inequalities are all based on Cauchy Schwartz inequality and the last inequality uses the definition of κ in Assumption C.6. Under Assumptions C.3 and C.5, we have

$$\begin{aligned}
\text{term (b1)} &\le \sqrt{\kappa^2\cdot T}\cdot\Bigg(\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{V}_{\theta\sim\mathbb{P}_{t_k}}\big[(B_\theta V_t)(s_t,a_t)\big|\,\mathcal{D}_{t_k}\big]\Big]\Big]\Bigg)^{1/2}\\
&\le 2\sqrt{2\eta}L\kappa\cdot\sqrt{\mathbb{E}[H_0-H_T]}\cdot\sqrt{T},
\end{aligned}\tag{D.20}$$

where the first inequality invokes Proposition C.4 and the definition of variance. Here, the second inequality invokes Proposition D.1. By the definition of ϵ-optimal planner (Definition C.1), we know term (b2) in (D.18) is non-positive. Then, plugging (D.20) into (D.18), we have

$$(1-\gamma)\cdot\text{term (b)} \le 2\sqrt{2\eta}L\kappa\cdot\sqrt{\mathbb{E}[H_0-H_T]}\cdot\sqrt{T}+\epsilon\cdot T.\tag{D.21}$$

Combining (D.6), (D.17), and (D.21), we have

$$\begin{aligned}
\mathfrak{R}(T) &= \frac{1}{1-\gamma}\cdot\big(\text{term (a)}+\text{term (b)}\big)\\
&\le \frac{2\sqrt{2}(\kappa+1)L\cdot\sqrt{\mathbb{E}[H_0-H_T]}}{1-\gamma}\cdot\sqrt{T}+\frac{2\epsilon}{1-\gamma}\cdot T+\frac{4L}{1-\gamma}+\frac{4L\cdot\mathbb{E}[H_0-H_T]}{(1-\gamma)\log 2}\\
&= \mathcal{O}\Bigg(\frac{(\kappa+1)L\cdot\sqrt{\mathbb{E}[H_0-H_T]}}{1-\gamma}\cdot\sqrt{T}+\frac{\epsilon}{1-\gamma}\cdot T+\frac{L\cdot\mathbb{E}[H_0-H_T]}{1-\gamma}\Bigg).
\end{aligned}$$

Thus, we finish the proof of Theorem C.7. □

D.4. Proof of Theorem C.8

Proof of Theorem C.8. Recall that we denote by $\pi^k=\pi_{t_k}$, and V_t is the estimated value function returned by the ϵ-optimal planner $\texttt{PL}^\epsilon$ in Algorithm 6. By the definition of the Bayesian regret $\mathfrak{R}(T)$ and the tower property of the conditional

expectation, we have

$$\begin{aligned}\mathfrak{R}(T) &= \mathbb{E}\Big[\sum_{k=0}^{K-1}\sum_{t=t_k}^{t_{k+1}-1} V_{\theta^\star}^{\pi^\star}(s_t) - V_{\theta^\star}^{\pi_t}(s_t)\Big] \\ &= \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} V_{\theta^\star}^{\pi^\star}(s_t) - V_{\theta^\star}^{\pi^k}(s_t)\Big]\Big] \\ &= \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} V_t(s_t) - V_{\theta^\star}^{\pi^k}(s_t)\Big]\Big]}_{\text{term (a)}} + \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} V_{\theta^\star}^{\pi^\star}(s_t) - V_t(s_t)\Big]\Big]}_{\text{term (b)}},\end{aligned} \tag{D.22}$$

where the first equality uses the fact that $\pi_t = \pi_{t_k}$ for any $t_k \le t < t_{k+1}$ in Algorithm 6. By Definition C.1, we know that $V_t(s) = Q_t(s, \pi^k(s))$ for any $t_k \le t < t_{k+1}$. Then, we apply the first part of Lemma D.2 to analyze term (a) as follows,

$$\begin{aligned}(1-\gamma)\cdot\text{term (a)} =& \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} Q_t(s_t,a_t) - \big(B_{\theta^\star}V_t\big)(s_t,a_t)\Big]\Big] \\ &+ \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\big(V_t(s_{t_{k+1}}) - V_{\theta^\star}^{\pi^k}(s_{t_{k+1}})\big) - \big(V_t(s_{t_k}) - V_{\theta^\star}^{\pi^k}(s_{t_k})\big)\Big]\Big].\end{aligned}$$

Recall that we define $(B_k V)(s,a) = r_{\texttt{LLM}(\mathcal{D}_{t_k})}(s,a) + (P_{\texttt{LLM}(\mathcal{D}_{t_k})}V)(s,a)$ for any (s,a) and value function V. By the definition of ϵ-optimal planner (Definition C.1) and the planning procedure $(\pi_t, V_t) \leftarrow \texttt{PL}^\epsilon(P_{\texttt{LLM}(\mathcal{D}_{t_k})}, r_{\texttt{LLM}(\mathcal{D}_{t_k})} + \Gamma_k)$ in Algorithm 7, we have

$$|Q_t(s,a) - \big(B_{\theta^\star}V_t\big)(s_t,a_t)| \le \epsilon + \big((B_k - B_{\theta^\star})V_t\big)(s,a) + \Gamma_k(s,a) \tag{D.23}$$

for any $t_k \le t < t_{k+1}$ and any $(s,a) \in \mathcal{S}\times\mathcal{A}$. Then, we plug (D.23) into (D.11) to obtain

$$\begin{aligned}(1-\gamma)\cdot\text{term (a)} \le& \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\big((B_k - B_{\theta^\star})V_t\big)(s_t,a_t)\Big]\Big]}_{\text{term (a1)}} + \epsilon\cdot T + \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\Gamma_k(s_t,a_t)\Big]\Big]}_{\text{term (a2)}} \\ &+ \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\big(V_t(s_{t_{k+1}}) - V_{\theta^\star}^{\pi^k}(s_{t_{k+1}})\big) - \big(V_t(s_{t_k}) - V_{\theta^\star}^{\pi^k}(s_{t_k})\big)\Big]\Big]}_{\text{term (a3)}}\end{aligned} \tag{D.24}$$

Under Assumption C.3, we have

$$\begin{aligned}\text{term (a1)} &= \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{\theta\sim\mathbb{P}_{t_k}}\Big[\big((B_k - B_\theta)V_t\big)(s_t,a_t)\Big|\,\mathcal{D}_{t_k}\Big]\Big]\Big] \\ &\le \sqrt{T}\cdot\Big(\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{\theta\sim\mathbb{P}_{t_k}}\Big[\big|\big((B_k - B_\theta)V_t\big)(s_t,a_t)\big|^2\Big|\,\mathcal{D}_{t_k}\Big]\Big]\Big]\Big)^{1/2} \\ &= \sqrt{T}\cdot\Big(\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{V}_{\theta\sim\mathbb{P}_{t_k}}\big[(B_\theta V_t)(s_t,a_t)\big|\,\mathcal{D}_{t_k}\big]\Big]\Big]\Big)^{1/2},\end{aligned} \tag{D.25}$$

where the first equality uses the tower property of the conditional expectation and the definition that the posterior distribution of $\theta^\star$ given $\mathcal{D}_{t_k}$ is $\mathbb{P}_{t_k}$. Here, the first inequality uses Cauchy-Schwarz inequality and the last equality invokes

Proposition C.4 and the definition of variance. Under Assumption C.5, we apply the second property in Proposition D.1 on the right-hand side of (D.25) to have

$$\text{term (a1)} \le 2\sqrt{2\eta}L \cdot \sqrt{\mathbb{E}[H_0 - H_T]} \cdot \sqrt{T}. \tag{D.26}$$

Recall that the bonus Γ_k used in Algorithm 7 is defined by $\Gamma_k(s,a) = \sqrt{2}L \cdot \sqrt{I(\theta; \xi_{(s,a)} \,|\, \mathcal{D}_{t_k})}$. For term (a2) in (D.24), we have

$$\begin{aligned}
\text{term (a2)} &\le \sqrt{T} \cdot \left(\mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\big[\sum_{t=t_k}^{t_{k+1}-1} \Gamma_k^2(s_t, a_t) \big] \Big] \right)^{1/2} \\
&= \sqrt{2}L \cdot \sqrt{T} \cdot \left(\mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\big[\sum_{t=t_k}^{t_{k+1}-1} I(\theta; \xi_{(s_t,a_t)} \,|\, \mathcal{D}_{t_k}) \big] \Big] \right)^{1/2} \\
&\le 2\sqrt{2\eta}L \cdot \sqrt{T} \cdot \left(\mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\big[\sum_{t=t_k}^{t_{k+1}-1} I(\theta; \xi_{(s_t,a_t)} \,|\, \mathcal{D}_{t}) \big] \Big] \right)^{1/2},
\end{aligned} \tag{D.27}$$

where the equality uses the definition of Γ_k in Algorithm 7. Here, the last inequality invokes Assumption C.5 and the switching condition in Algorithm 7.

As $a_t = \pi^k(s_t)$, we further bound the right-hand side of (D.27) as follows,

$$\begin{aligned}
\text{term (a2)} &\le 2\sqrt{2\eta}L \cdot \sqrt{T} \cdot \left(\mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\big[\sum_{t=t_k}^{t_{k+1}-1} I(\theta; \xi_{(s_t,\pi^k(s_t))} \,|\, \mathcal{D}_{t}) \big] \Big] \right)^{1/2} \\
&= 2\sqrt{2\eta}L \cdot \sqrt{\mathbb{E}[H_0 - H_T]} \cdot \sqrt{T},
\end{aligned} \tag{D.28}$$

where the last inequality uses the chain rule of the information gain. Using the fact that any value function is bounded by L, we bound term (a3) in (D.11) as

$$\text{term (a3)} \le 4L \cdot \mathbb{E}[K].$$

As the switching condition in Algorithm 7 implies $H_{t_k} - H_{t_{k+1}} \ge \log 2$, we apply Lemma D.3 to have

$$\text{term (a3)} \le 4L + \frac{4L \cdot \mathbb{E}[H_0 - H_T]}{\log 2}. \tag{D.29}$$

Plugging (D.26), (D.28) and (D.29) into (D.11), we have

$$(1-\gamma) \cdot \text{term (a)} \le 4\sqrt{2}L \cdot \sqrt{\mathbb{E}[H_0 - H_T]} \cdot \sqrt{T} + \epsilon \cdot T + 4L + \frac{4L \cdot \mathbb{E}[H_0 - H_T]}{\log 2}. \tag{D.30}$$

Then, we invoke the second part of Lemma D.2 to decompose term (b) in (D.22) as follows,

$$
\begin{aligned}
(1-\gamma)\cdot \text{term (b)} &= \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{s\sim\nu^\star(\cdot\,|\,s_t)}\big[\big(B_{\theta^\star}V_t\big)(s,\pi^\star(s))-Q_t(s,\pi^\star(s))\big]\Big]\Big] \\
&\quad+\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{s\sim\nu^\star(\cdot\,|\,s_t)}\big[Q_t(s,\pi^\star(s))-Q_t(s,\pi^k(s))\big]\Big]\Big] \\
&\le \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{s\sim\nu^\star(\cdot\,|\,s_t)}\big[\big((B_{\theta^\star}-B_k)V_t\big)(s,\pi^\star(s))-\Gamma_k(s,\pi^\star(s))\big]\Big]\Big]}_{\text{term (b1)}}+\epsilon\cdot T \\
&\quad+\underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{s\sim\nu^\star(\cdot\,|\,s_t)}\big[Q_t(s,\pi^\star(s))-Q_t(s,\pi^k(s))\big]\Big]\Big]}_{\text{term (b2)}},
\end{aligned}
\tag{D.31}
$$

where the inequality uses (D.23). Under Assumption C.3, we bound term (b1) in (D.31) as follows,

$$
\begin{aligned}
\text{term (b1)} &= \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{s\sim\nu^\star(\cdot\,|\,s_t)}\Big[\mathbb{E}_{\theta\sim\mathbb{P}_{t_k}}\Big[\big((B_k-B_\theta)V_t\big)(s,\pi^\star(s))\Big|\,\mathcal{D}_{t_k}\Big]-\Gamma_k(s,\pi^\star(s))\Big]\Big]\Big] \\
&\le \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{s\sim\nu^\star(\cdot\,|\,s_t)}\Big[\sqrt{\mathbb{E}_{\theta\sim\mathbb{P}_{t_k}}\Big[\big|\big((B_k-B_\theta)V_t\big)(s,\pi^\star(s))\big|^2\Big|\,\mathcal{D}_{t_k}\Big]}-\Gamma_k(s,\pi^\star(s))\Big]\Big]\Big] \\
&= \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{s\sim\nu^\star(\cdot\,|\,s_t)}\Big[\sqrt{\mathbb{V}_{\theta\sim\mathbb{P}_{t_k}}\big[(B_\theta V_t)(s,\pi^\star(s))\big|\,\mathcal{D}_{t_k}\big]}-\Gamma_k(s,\pi^\star(s))\Big]\Big]\Big],
\end{aligned}
\tag{D.32}
$$

where the first equality uses the tower property of the conditional expectation and the definition that the posterior distribution of $\theta^\star$ given $\mathcal{D}_{t_k}$ is $\mathbb{P}_{t_k}$. Here, the first inequality uses Cauchy-Schwarz inequality and the last equality invokes Proposition C.4 and the definition of variance. Under Assumption C.5, we apply the first property in Proposition D.1 to have

$$
\begin{aligned}
\sqrt{\mathbb{V}_{\theta\sim\mathbb{P}_{t_k}}\big[(B_\theta V_t)(s,\pi^\star(s))\big|\,\mathcal{D}_{t_k}\big]}-\Gamma_k(s,\pi^\star(s)) &\le \sqrt{2}L\cdot\sqrt{I(\theta;\xi_{(s,\pi^\star(s))})}-\Gamma_k(s,\pi^\star(s)) \\
&= \sqrt{2}L\cdot\sqrt{I(\theta;\xi_{(s,\pi^\star(s))})}-\sqrt{2}L\cdot\sqrt{I(\theta;\xi_{(s,\pi^\star(s))})} \\
&\le 0,
\end{aligned}
\tag{D.33}
$$

for any $t_k \le t < t_{k+1}$, $k < K$, and state $s \in \mathcal{S}$. Here, the equality uses the definition of Γ_k in Algorithm 7. Plugging (D.33) into (D.32), we have

$$
\text{term (b1)} \le 0. \tag{D.34}
$$

By the definition of ϵ-optimal planner (Definition C.1), we know term (b2) in (D.18) is non-positive. Then, plugging (D.31) into (D.31), we have

$$
(1-\gamma)\cdot\text{term (b)} \le \epsilon\cdot T. \tag{D.35}
$$

Combining (D.22), (D.30), and (D.35), we have

$$\begin{aligned}\mathfrak{R}(T) &= \frac{1}{1-\gamma}\cdot\big(\text{term (a)} + \text{term (b)}\big)\\ &\le \frac{4\sqrt{2}L\cdot\sqrt{\mathbb{E}[H_0 - H_T]}}{1-\gamma}\cdot\sqrt{T} + \frac{2\epsilon}{1-\gamma}\cdot T + \frac{4L}{1-\gamma} + \frac{4L\cdot\mathbb{E}[H_0 - H_T]}{(1-\gamma)\log 2}\\ &= \mathcal{O}\left(\frac{L\cdot\sqrt{\mathbb{E}[H_0 - H_T]}}{1-\gamma}\cdot\sqrt{T} + \frac{\epsilon}{1-\gamma}\cdot T + \frac{L\cdot\mathbb{E}[H_0 - H_T]}{1-\gamma}\right).\end{aligned}$$

Thus, we finish the proof of Theorem C.8. □

D.5. Proof of Theorem C.10

Proof of Theorem C.10. For notational simplicity, we denote by θ^k the corresponding parameter for the mechanism `LLM+PS` in Algorithm 8, which satisfies

$$(B_{\theta^k}V)(s,a) = r_{\texttt{LLM+PS}(\mathcal{D}_{t_k})}(s,a) + \gamma\cdot(P_{\texttt{LLM+PS}(\mathcal{D}_{t_k})}V)(s,a), \tag{D.36}$$

for any $k < K$, $(s,a)\in\mathcal{S}\times\mathcal{A}$, and value function V. Recall the definition of optimal value $V^\star_\theta$ given the parameter θ in (C.1). Under Assumption C.9, we know that $(B_{\theta^k}V)(s,a)\,|\,\mathcal{D}_{t_k}$ and $(B_{\theta^\star}V)(s,a)\,|\,\mathcal{D}_{t_k}$ follows the same distribution for any $k<K$, $(s,a)\in\mathcal{S}\times\mathcal{A}$, and value function V. By Bellman optimality equation in (C.1), we have that $V^\star_{\theta^k}(s,a)\,|\,\mathcal{D}_{t_k}$ and $V^\star_{\theta^\star}(s,a)\,|\,\mathcal{D}_{t_k}$ follows the same distribution for any $k<K$, $(s,a)\in\mathcal{S}\times\mathcal{A}$, and value function V. Recall that we denote by $\pi^k = \pi_{t_k}$ and V_t is the estimated value function returned by the ϵ-optimal planner $\texttt{PL}^\epsilon$ in Algorithm 8. By the definition of the Bayesian regret $\mathfrak{R}(T)$ and $\pi_t = \pi^k$ for any $t_k \le t < t_{k+1}$ and $k<K$, we have

$$\begin{aligned}\mathfrak{R}(T) &= \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} V^{\pi^\star}_{\theta^\star}(s_t) - V^{\pi^k}_{\theta^\star}(s_t)\Big]\Big]\\ &= \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} V^{\star}_{\theta^\star}(s_t) - V^{\pi^k}_{\theta^\star}(s_t)\Big]\Big]\\ &= \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}_{\theta^\star\sim\mathbb{P}_{t_k}}\big[V^{\star}_{\theta^\star}(s_t) - V^{\pi^k}_{\theta^\star}(s_t)\,|\,\mathcal{D}_{t_k}\big]\Big]\Big],\end{aligned} \tag{D.37}$$

where the second equality uses the definition of optimal policy and (C.1). Here, the last equality uses the tower property of the conditional expectation. Recall that $\theta^\star\,|\,\mathcal{D}_{t_k}$ and $\theta^k\,|\,\mathcal{D}_{t_k}$ follows the same distribution $\mathbb{P}_{t_k}$ for any $t_k\le t<t_{k+1}$, which implies

$$\begin{aligned}\mathbb{E}_{\theta^\star\sim\mathbb{P}_{t_k}}\big[V^\star_{\theta^\star}(s_t) - V^{\pi^k}_{\theta^\star}(s_t)\,|\,\mathcal{D}_{t_k}\big] &= \mathbb{E}_{\theta^\star\sim\mathbb{P}_{t_k}}\big[V^\star_{\theta^\star}(s_t)\,|\,\mathcal{D}_{t_k}\big] - \mathbb{E}_{\theta^\star\sim\mathbb{P}_{t_k}}\big[V^{\pi^k}_{\theta^\star}(s_t)\,|\,\mathcal{D}_{t_k}\big]\\ &= \mathbb{E}_{\theta^k\sim\mathbb{P}_{t_k}}\big[V^\star_{\theta^k}(s_t)\,|\,\mathcal{D}_{t_k}\big] - \mathbb{E}_{\theta^\star\sim\mathbb{P}_{t_k}}\big[V^{\pi^k}_{\theta^\star}(s_t)\,|\,\mathcal{D}_{t_k}\big]\\ &= \mathbb{E}_{\theta^\star,\theta^k\sim\mathbb{P}_{t_k}}\big[V^\star_{\theta^k}(s_t) - V^{\pi^k}_{\theta^\star}(s_t)\,|\,\mathcal{D}_{t_k}\big],\end{aligned} \tag{D.38}$$

where the first and the second inequalities use the linear property of the conditional expectation. Plugging (D.38) into (D.37), we have

$$\begin{aligned}\mathfrak{R}(T) &= \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{\theta^\star,\theta^k\sim\mathbb{P}_{t_k}}\big[V^\star_{\theta^k}(s_t)-V^{\pi^k}_{\theta^\star}(s_t)\,|\,\mathcal{D}_{t_k}\big]\Big]\Big]\\ &= \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}V^\star_{\theta^k}(s_t)-V^{\pi^k}_{\theta^\star}(s_t)\Big]\Big],\end{aligned} \tag{D.39}$$

where the last equality uses the tower property of the conditional expectation.

Meanwhile, by Definition C.1, we have

$$\begin{aligned}\max_{s\in\mathcal{S}}|V^\star_{\theta^k}(s_t)-V_t(s_t)| &= \max_{s\in\mathcal{S}}|\max_{a\in\mathcal{A}}Q^\star_{\theta^k}(s,a)-\max_a Q_t(s,a)|\\ &\le \max_{(s,a)\in\mathcal{S}\times\mathcal{A}}|Q^\star_{\theta^k}(s,a)-Q_t(s,a)|\\ &= \max_{(s,a)\in\mathcal{S}\times\mathcal{A}}\big|\big((B_{\theta^k}V^\star_{\theta^k})(s,a)-(B_{\theta^k}V_t)(s,a)\big)+\big((B_{\theta^k}V_t)(s,a)-Q_t(s,a)\big)\big|\\ &\le \gamma\cdot\max_{s\in\mathcal{S}}|V^\star_{\theta^k}(s_t)-V_t(s_t)|+\max_{(s,a)\in\mathcal{S}\times\mathcal{A}}|(B_{\theta^k}V_t)(s,a)-Q_t(s,a)|,\end{aligned} \tag{D.40}$$

where the equality and the second equality uses the definitions of $(Q^\star_\theta, V^\star_\theta)$ in (C.1). Here, the first inequality uses the fact that the maximum operator is a contraction map, and the last inequality uses triangle inequality and (D.36). Rearranging (D.40), we have

$$\begin{aligned}\max_{s\in\mathcal{S}}|V^\star_{\theta^k}(s_t)-V_t(s_t)| &\le \frac{1}{1-\gamma}\cdot\max_{(s,a)\in\mathcal{S}\times\mathcal{A}}|(B_{\theta^k}V_t)(s,a)-Q_t(s,a)|\\ &\le \frac{\epsilon}{1-\gamma},\end{aligned} \tag{D.41}$$

where the last inequality uses the definition of ϵ-optimal planner (Definition C.1), the planning procedure $(\pi_t, V_t) \leftarrow \texttt{PL}^\epsilon(P_{\texttt{LLM+PS}(\mathcal{D}_{t_k})}, r_{\texttt{LLM+PS}(\mathcal{D}_{t_k})})$ in Algorithm 8 and the definition of θ^k in (D.36). Then, we upper bound the right-hand side of (D.39) as

$$\mathfrak{R}(T) \le \frac{\epsilon}{1-\gamma}\cdot T + \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}V_t(s_t)-V^{\pi^k}_{\theta^\star}(s_t)\Big]\Big]}_{\text{term (a)}}. \tag{D.42}$$

By the first part of Lemma D.2, we analyze term (a) in (D.42) as follows,

$$\begin{aligned}(1-\gamma)\cdot\text{term (a)} =& \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}Q_t(s_t,a_t)-\big(B_{\theta^\star}V_t\big)(s_t,a_t)\Big]\Big]\\ &+\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\big(V_t(s_{t_{k+1}})-V^{\pi^k}_{\theta^\star}(s_{t_{k+1}})\big)-\big(V_t(s_{t_k})-V^{\pi^k}_{\theta^\star}(s_{t_k})\big)\Big]\Big].\end{aligned} \tag{D.43}$$

By the last inequality in (D.41), we have

$$\begin{aligned}|Q_t(s_t,a_t)-\big(B_{\theta^\star}V_t\big)(s_t,a_t)| &\le \epsilon+\big((B_k-B_{\theta^\star})V_t\big)(s_t,a_t)\\ &= \epsilon+\big|\big((B_k-B_{\theta^\star})V_t\big)(s_t,a_t)\big|\end{aligned} \tag{D.44}$$

for any $t_k \le t < t_{k+1}$. Then, we plug (D.44) into (D.43) to obtain

$$(1-\gamma)\cdot \text{term (a)} \le \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\big|\big((B_k - B_{\theta^\star})V_t\big)(s_t,a_t)\big|\Big]\Big]}_{\text{term (a1)}} + \epsilon\cdot T$$
$$+ \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\big(V_t(s_{t_{k+1}}) - V_{\theta^\star}^{\pi^k}(s_{t_{k+1}})\big) - \big(V_t(s_{t_k}) - V_{\theta^\star}^{\pi^k}(s_{t_k})\big)\Big]\Big]}_{\text{term (a2)}}. \tag{D.45}$$

Recall that $\mathbb{P}_{t_k}$ denotes the posterior distribution of $\theta^\star$ given $\mathcal{D}_{t_k}$. Under Assumption C.9, we know that the distribution of $\theta^k \,|\, \mathcal{D}_{t_k}$ is also $\mathbb{P}_{t_k}$, where θ^k is defined in (D.36). Then, we have

$$\begin{aligned}\text{term (a1)} &= \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{\theta^\star,\theta^k\sim\mathbb{P}_{t_k}}\Big[\big|\big((B_{\theta^k}-B_{\theta^\star})V_t\big)(s_t,a_t)\big|\,\Big|\,\mathcal{D}_{t_k}\Big]\Big]\Big]\\ &\le \sqrt{T}\cdot\Big(\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{\theta^\star,\theta^k\sim\mathbb{P}_{t_k}}\Big[\big|\big((B_{\theta^k}-B_{\theta^\star})V_t\big)(s_t,a_t)\big|^2\,\Big|\,\mathcal{D}_{t_k}\Big]\Big]\Big]\Big)^{1/2},\end{aligned} \tag{D.46}$$

where the first equality uses the tower property of the conditional expectation and the first inequality uses Cauchy-Schwarz inequality. Note that $\mathbb{E}[|X-X'|^2] = 2\mathbb{V}[X]$, if X and X' are two identically independently distributed variables. Recall that Assumption C.9 tells that θ^k (defined in (D.36)) and the data-generating parameter $\theta^\star$ are identically independently distributed given $\mathcal{D}_{t_k}$, which implies

$$\text{term (a1)} \le \sqrt{2T}\cdot\Big(\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{V}_{\theta\sim\mathbb{P}_{t_k}}\big[(B_{\theta^k}V_t)(s_t,a_t)\big|\,\mathcal{D}_{t_k}\big]\Big]\Big]\Big)^{1/2}. \tag{D.47}$$

Under Assumption C.5, we apply the second property in Proposition D.1 on the right-hand side of (D.47) to have

$$\text{term (a1)} \le 4L\cdot\sqrt{\mathbb{E}[H_0 - H_T]}\cdot\sqrt{T}. \tag{D.48}$$

Using the fact that any value function is bounded by L, we bound term (a2) in (D.11) as

$$\text{term (a3)} \le 4L\cdot\ \mathbb{E}[K].$$

As the switching condition in Algorithm 8 implies $H_{t_k} - H_{t_{k+1}} \ge \log 2$, we apply Lemma D.3 to have

$$\text{term (a2)} \le 4L + \frac{4L\cdot\mathbb{E}[H_0 - H_T]}{\log 2}. \tag{D.49}$$

Plugging (D.48) and (D.49) into (D.45), we have

$$(1-\gamma)\cdot\text{term (a)} \le 4L\cdot\sqrt{\mathbb{E}[H_0-H_T]}\cdot\sqrt{T} + \epsilon\cdot T + 4L + \frac{4L\cdot\mathbb{E}[H_0-H_T]}{}. \tag{D.50}$$

Combining (D.42) and (D.50), we obtain

$$\begin{aligned}\mathfrak{R}(T) &\le \frac{4\sqrt{2}L\cdot\sqrt{\mathbb{E}[H_0-H_T]}}{1-\gamma}\cdot\sqrt{T} + \frac{2\epsilon}{1-\gamma}\cdot T + \frac{4L}{1-\gamma} + \frac{4L\cdot\mathbb{E}[H_0-H_T]}{(1-\gamma)\log 2}\\ &= \mathcal{O}\Big(\frac{L\cdot\sqrt{\mathbb{E}[H_0-H_T]}}{1-\gamma}\cdot\sqrt{T} + \frac{\epsilon}{1-\gamma}\cdot T + \frac{L\cdot\mathbb{E}[H_0-H_T]}{1-\gamma}\Big).\end{aligned}$$

Thus, we finish the proof of Theorem C.10. □

D.6. Relaxing Assumption C.3 for Theorem C.7

In this section, we show that Assumption C.3 (posterior alignment) can be relaxed for Theorem C.7 to accommodate a generalization error. We remove the dependency on Assumption C.3 (Posterior Alignment) by introducing the following assumptions.

Algorithm 9 The data collection process for the pretraining dataset.

1: **input**: Some (mixed) data collection policy π_{collect}.
2: **initialization**: Initialize the pretraining dataset $\mathcal{D}_{\text{pre}} = \varnothing$.
3: **for** $n = 1, \dots, N$ **do**
4: Reset the environment such that $\theta^\star \sim \mathbb{P}_0$.
5: Receives s_0 from the environment.
6: Initialize the memory buffer $\mathcal{D}_0 = \varnothing$.
7: **for** $t = 0, \dots, T$ **do**
8: Execute action $a_t \sim \pi_{\text{collect}}(s_t)$ to receive reward $r_t = r_{\theta^\star}(s_t, a_t)$ and state $s_{t+1} \sim P_{\theta^\star}(\cdot \,|\, s_t, a_t)$ from the environment.
9: Update memory buffer $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(s_t, a_t, s_{t+1}, r_t)\}$.
10: **end for**
11: Uniformly sample $t_0 \in \{0, \dots, T\}$ and update the pretraining dataset $\mathcal{D}_{\text{pre}} = \mathcal{D}_{\text{pre}} \cup \{(s_{t_0+1}, r_{t_0}, \mathcal{D}_{t_0}, s_{t_0}, a_{t_0})\}$
12: **end for**
13: **output**: The pretraining dataset $\mathcal{D}_{\text{pre}}$.

First, we characterize the data generation process for the pretraining dataset in the following assumption.

Assumption D.4 (Pretraining Dataset Generation)**.** *We assume that the pretraining dataset $\mathcal{D}_{pre}$ consists N i.i.d. tuples of $(s', r, \mathcal{D}, s, a)$ generated by Algorithm 9.*

For the pretraining dataset $\mathcal{D}_{\text{pre}}$, we denote by $\mathbb{P}_{\text{pre}}$ the conditional distribution of (s', r) given $(\mathcal{D}, s, a)$. We show that $\mathbb{P}_{\text{pre}}$ is equivalent to $\mathbb{P}_{\text{post}}$ as follows,

$$\begin{aligned}
\mathbb{P}_{\text{pre}}(s', r \,|\, \mathcal{D}, s, a) &= \int_\theta \mathbb{P}_{\text{pre}}(s', r \,|\, s, a, \theta) \mathbb{P}_{\text{pre}}(\mathrm{d}\theta \,|\, \mathcal{D}, s, a) \\
&= \int_\Theta P_\theta(s' \,|\, s, a) \mathbf{1}(r = r_\theta(s, a)) \mathbb{P}_{\text{pre}}(\mathrm{d}\theta \,|\, \mathcal{D}, s, a) \\
&= \int_\Theta P_\theta(s' \,|\, s, a) \mathbf{1}(r = r_\theta(s, a)) \mathbb{P}_{\text{post}}(\mathrm{d}\theta \,|\, \mathcal{D}) \\
&= \mathbb{P}_{\text{post}}(\xi_{(s,a)} \,|\, \mathcal{D}, s, a), \qquad \text{(D.51)}
\end{aligned}$$

where the first equality uses Line 8 in Algorithm 9, the second equality uses the definition of the posterior of $\theta^\star$ in (2.2) and the fact that θ and (s, a) are conditionally independent given $\mathcal{D}$, and the last equality uses (2.3). Denote by $\mathcal{F}_{\text{LLM}}$ the function class of LLMs. In the next assumption, we assume that the function class $\mathcal{F}_{\text{LLM}}$ contains the posterior of $\xi_{(s,a)}$ in the underlying MDP, which is the conditional distribution of (s', r) given $(\mathcal{D}, s, a)$ from $\mathcal{D}_{\text{pre}}$.

Assumption D.5 (Realizability)**.** *We assume that there exists a LLM* `LLMPA` *with a posterior alignment, that is, there exists $P^{\texttt{LLMPA}} \in \mathcal{F}_{LLM}$, such that $P^{\texttt{LLMPA}}\big(\xi_{(s,a)} \big| \mathcal{D}, s, a\big) = \mathbb{P}_{post}\big(\xi_{(s,a)} \big| \mathcal{D}, s, a\big)$, for any query state-action pair (s, a) and in-context dataset $\mathcal{D}$.*

We introduce the following assumption to require that LLMs are MLEs in the pretraining dataset with uniform coverage.

Assumption D.6. *We assume that LLMs used in* RAFA *are Maximum Likelihood Estimators (MLEs) in the pretraining dataset $\mathcal{D}_{pre}$ satisfying Assumption D.4, that is,*

$$P^{LLM} = \operatorname*{argmax}_{\widehat{P}\in\mathcal{F}_{LLM}} \sum_{(s',r,\mathcal{D},s,a)\in\mathcal{D}_{pre}} \log \widehat{P}(s', r \mid \mathcal{D}, s, a).$$

Denote by ρ_{pre} the marginal population distribution of $(\mathcal{D}, s, a)$ from $\mathcal{D}_{pre}$. We also assume that the pretraining dataset satisfies the following coverage condition:

$$\zeta = \sup_{t<T} \left\{ \left\| \frac{\mu_t}{\rho_{pre}} \right\|_\infty + \left\| \frac{\mu_t^\star}{\rho_{pre}} \right\|_\infty \right\} < \infty. \tag{D.52}$$

Here, μ_t is the marginal distribution of $(\mathcal{D}_{t_k}, s_t, \pi^k(s_t))$ and $\mu_t^\star$ is the marginal distribution of $(\mathcal{D}_{t_k}, s_t, \pi^\star(s))$ with $s \sim \nu^\star(\cdot \mid s_t)$ and $(s_t, \mathcal{D}_{t_k})$ following the trajectory distribution of RAFA *(Algorithm 6), where $\nu^\star$ is defined in (C.2).*

We provide the generalization of Theorem C.7 in the following corollary, which removes the dependency on Assumption C.3 (Posterior Alignment).

Corollary D.7 (Generalization of Theorem C.7)**.** *Under Assumptions D.5, D.6, C.5 , and C.6,the Bayesian regret of* RAFA *(Algorithm 6) satisfies*

$$\mathfrak{R}(T) = \mathcal{O}\Bigg(\frac{(\kappa+1)L \cdot \sqrt{\mathbb{E}[H_0 - H_T]}}{1-\gamma} \cdot \sqrt{T} + \frac{\epsilon}{1-\gamma} \cdot T + \frac{L \cdot \mathbb{E}[H_0 - H_T]}{1-\gamma} + \underbrace{\zeta \cdot \sqrt{\frac{\log(|\mathcal{F}_{\mathrm{LLM}}|/\delta)}{N} \cdot T}}_{\text{Additional Regret Compared with Theorem C.7}} \Bigg),$$

with probability at least $1-\delta$. Here, ζ is defined in (D.52), $|\mathcal{F}_{LLM}|$ is the cardinality of the function class for LLMs, and N is the size of the pretraining dataset.

Comparing Corollary D.7 with Theorem C.7, we remark that the additional regret decays to zero if N_{pre} tends to infinity. Hence, we can recover the regret bound based on Assumption C.3 (posterior alignment) approximately if the pretraining dataset has uniform coverage and is large enough.

Proof of Corollary D.7. We start with a standard concentration result for the maximum-likelihood estimate (MLE).

Lemma D.8 (MLE Concentration)**.** *Let $\mathcal{F}$ be a finite function class used to model a conditional distribution $\mathbb{P}_{Y\mid X}(y\mid x)$ for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Assume there is $f^\star \in \mathcal{F}$ such that $\mathbb{P}(y\mid x) = f^\star(y\mid x)$ (realizablility condition). Let $\{(x_i, y_i)\}_{i=1}^N$ denote a dataset of i.i.d. samples where $x_i \sim \mathbb{P}_X(x)$ and $y_i \sim \mathbb{P}_{Y\mid X}(\cdot\mid x_i)$. Let $\widehat{f}$ be the MLE, which satisfies*

$$\widehat{f} = \operatorname*{argmax}_{f\in\mathcal{F}} \sum_{i=1}^N \log f\left(y_i \mid x_i\right).$$

Then, it holds that

$$\mathbb{E}_{x\sim\mathbb{P}_X}\left[d_{\mathrm{TV}}\left(\widehat{f}(\cdot\mid x), p_{Y\mid X}(\cdot\mid x) \right) \right] \le \frac{8\log(|\mathcal{F}|/\delta)}{N_{pre}},$$

with probability at least $1-\delta$.

Proof of Lemma D.8. See the proof of Theorem 21 of Agarwal et al. (2020). □

Under Assumptions D.5 and D.6, we apply Lemma D.8 to show that

$$\mathbb{E}_{(\mathcal{D},s,a)\sim\rho_{\text{pre}}}\left[d_{\text{TV}}(P^{\texttt{LLMPA}}(\cdot\,|\,\mathcal{D},s,a)\|P^{\texttt{LLM}}(\cdot\,|\,\mathcal{D},s,a))\right]\leq\sqrt{\frac{8\log(|\mathcal{F}_{\texttt{LLM}}|/\delta)}{N}}$$

holds with probability at least $1-\delta$.

For any fixed distribution μ of $(\mathcal{D},s,a)$ satisfying $\|\mu/\rho_{\text{pre}}\|_\infty<\infty$, we use Hölder's inequality to know that

$$\begin{aligned}\mathbb{E}_{(\mathcal{D},s,a)\sim\mu}\left[d_{\text{TV}}(P^{\texttt{LLMPA}}(\cdot\,|\,\mathcal{D},s,a)\|P^{\texttt{LLM}}(\cdot\,|\,\mathcal{D},s,a))\right]&\leq\left\|\frac{\mu}{\rho_{\text{pre}}}\right\|_\infty\cdot\mathbb{E}_{(\mathcal{D},s,a)\sim\rho_{\text{pre}}}\left[d_{\text{TV}}(P^{\texttt{LLMPA}}(\cdot\,|\,\mathcal{D},s,a)\|P^{\texttt{LLM}}(\cdot\,|\,\mathcal{D},s,a))\right]\\&\leq\left\|\frac{\mu}{\rho_{\text{pre}}}\right\|_\infty\cdot\sqrt{\frac{8\log(|\mathcal{F}_{\texttt{LLM}}|/\delta)}{N}}\end{aligned}\tag{D.53}$$

holds with probability at least $1-\delta$. Here, $\|\cdot\|_\infty$ denotes the infinity norm. We denote the Bellman operator induced by `LLMPA` (the LLM with a posterior alignment) and $\mathcal{D}_{t_k}$ as $\widetilde{B}_k$, which is defined as $(\widetilde{B}_kV)(s,a)=r_{\texttt{LLMPA}(\mathcal{D}_{t_k})}(s,a)+(P_{\texttt{LLMPA}(\mathcal{D}_{t_k})}V)(s,a)$ for any s,a, and value function V. Then, by the definition of B_k, we have

$$\begin{aligned}\left|\left((\widetilde{B}_k-B_k)V_t\right)(s,a)\right|&=\left|\mathbb{E}_{P^{\texttt{LLMPA}}}[r+\gamma\cdot V(s')]-\mathbb{E}_{P^{\texttt{LLM}}}[r+\gamma\cdot V(s')]\right|\\&\leq 2L\cdot d_{\text{TV}}(P^{\texttt{LLMPA}}(\cdot\,|\,\mathcal{D}_{t_k},s,a)\|P^{\texttt{LLM}}(\cdot\,|\,\mathcal{D}_{t_k},s,a)),\end{aligned}\tag{D.54}$$

where the first inequality uses the definition of L (recall that L is the bound of $|\,r+V(s)|$ for any reward r, state s, and value V) and Hölder's inequality. In the proof of Theorem C.7 (the analysis of the regret of `RAFA`), we need to modify (D.11) and (D.18) with the following inequality

$$\begin{aligned}\left|\left((B_k-B_{\theta^\star})V_t\right)(s,a)\right|&=\left|\left((\widetilde{B}_k-B_{\theta^\star})V_t\right)(s,a)\right.\\&\qquad\left.+\left((\widetilde{B}_k-B_k)V_t\right)(s,a)\right|\\&\leq\left|\left((\widetilde{B}_k-B_{\theta^\star})V_t\right)(s,a)\right|\\&\qquad+2L\cdot d_{\text{TV}}(P^{\texttt{LLMPA}}(\cdot\,|\,\mathcal{D}_{t_k},s,a)\|P^{\texttt{LLM}}(\cdot\,|\,\mathcal{D}_{t_k},s,a)),\end{aligned}$$

which holds for any state $s\in\mathcal{S}$ and action $a\in\mathcal{A}$. By Proposition C.4 (LLMs with posterior alignments perform BMA) and the fact that $\widetilde{B}_k$ is the Bellman operator induced by `LLMPA` (the LLM with a posterior alignment) and $\mathcal{D}_{t_k}$, we can analyze $[(\widetilde{B}_k-B_{\theta^\star})V_t)(s,a)]$ in the same way as in the previous proof of Theorem C.7 (the analysis of the regret of `RAFA`). It is clear that the additional regret by relaxing Assumption C.3 (Posterior Alignment) is less than

$$\begin{aligned}&\frac{1}{1-\gamma}\cdot\mathbb{E}\left[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\left[\sum_{t=t_k}^{t_{k+1}-1}2L\cdot d_{\text{TV}}(P^{\texttt{LLMPA}}(\cdot\,|\,\mathcal{D}_{t_k},s,a)\|P^{\texttt{LLM}}(\cdot\,|\,\mathcal{D}_{t_k},s,a))\right]\right]\\&+\frac{1}{1-\gamma}\cdot\mathbb{E}\left[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\left[\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}_{s\sim\nu^\star(\cdot\,|\,s_t)}\left[2L\cdot d_{\text{TV}}(P^{\texttt{LLMPA}}(\cdot\,|\,\mathcal{D}_{t_k},s,a)\|P^{\texttt{LLM}}(\cdot\,|\,\mathcal{D}_{t_k},s,a))\right]\right]\right]\\&\leq 8L\cdot\sqrt{\frac{2\log(|\mathcal{F}_{\texttt{LLM}}|/\delta)}{N}}\cdot\zeta\cdot T,\end{aligned}\tag{D.55}$$

with probability at least $1-\delta$, where the inequality uses (D.53) and the definition of ζ in (D.52). Combining (D.55) and Theorem C.7, we conclude the proof of Corollary D.7. □

E. Missing Proofs in Appendix D

E.1. Proof of Lemma D.2

Proof of Lemma D.2. We prove the first part as follows. The Bellman equation (Sutton & Barto, 2018) connects $Q_\theta^\pi(s,a)$ and $V_\theta^\pi(s)$ by

$$Q_\theta^\pi(s,a) = r_\theta(s,a) + \gamma\left(P_\theta V_\theta^\pi\right)(s,a), \quad V_\theta^\pi = Q_\theta^\pi\big(s,a\big). \tag{E.1}$$

By the definition of B_θ, we rewrite (E.1) as $Q_\theta^\pi(s,a) = (B_\theta V_\theta^\pi)(s,a)$. For the left-hand side of (D.7) in the first part of Lemma D.2, we have

$$\begin{aligned}
&\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} V_t(s_t) - V_{\theta^\star}^{\pi^k}(s_t)\Big]\Big] \\
&= \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} Q_t(s_t,a_t) - (B_{\theta^\star}V_{\theta^\star}^{\pi^k})(s_t,a_t)\Big]\Big] \\
&= \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} Q_t(s_t,a_t) - (B_{\theta^\star}V_t)(s_t,a_t)\Big]\Big]}_{\text{term (A)}} \\
&\quad + \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} (B_{\theta^\star}V_t)(s_t,a_t) - r_{\theta^\star}(s_t,a_t) - \gamma\cdot V_t(s_{t+1})\Big]\Big]}_{\text{term (C1)}} \\
&\quad + \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} r_{\theta^\star}(s_t,a_t) + \gamma\cdot V_{\theta^\star}^{\pi^k}(s_{t+1}) - (B_{\theta^\star}V_{\theta^\star}^{\pi^k})(s_t,a_t)\Big]\Big]}_{\text{term (C2)}} \\
&\quad + \underbrace{\gamma\cdot\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} V_t(s_{t+1}) - V_{\theta^\star}^{\pi^k}(s_{t+1})\Big]\Big]}_{\text{term (D)}},
\end{aligned} \tag{E.2}$$

where the first equality uses $a_t = \pi^k(s_t)$, the condition $Q(s,\pi^k(s)) = V_t(s)$ for any $t_k \le t < t_{k+1}$ and $k < K$ in Lemma D.2, and (E.1) . Since we have

$$(B_{\theta^\star}V)(s_t,a_t) = r_{\theta^\star}(s,a) + \gamma\cdot\mathbb{E}_{s_{t+1}\sim P_{\theta^\star}(\cdot\,|\,s_t,a_t)}\big[V(s_{t+1})\big],$$

terms (C1) and (C2) in (E.2) are zero. Meanwhile, term (D) in (E.2) satisfies

$$\begin{aligned}
\text{term (D)} &= \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1}\big(V_t(s_t) - V_{\theta^\star}^{\pi^k}(s_t)\big)\Big]\Big] \\
&\quad + \underbrace{\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\big(V_t(s_{t_{k+1}}) - V_{\theta^\star}^{\pi^k}(s_{t_{k+1}})\big) - \big(V_t(s_{t_k}) - V_{\theta^\star}^{\pi^k}(s_{t_k})\big)\Big]\Big]}_{\text{term (B)}},
\end{aligned} \tag{E.3}$$

where term (B) is defined in the first part of Lemma D.2. Rearranging (E.2) and (E.3), we prove the first part of Lemma D.2.

Next, we show the proof of the second part of Lemma D.2, we. For the left-hand side of (D.8), we have

$$
\begin{aligned}
&\mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} V_{\theta^\star}^{\pi^\star}(s_t) - V_t(s_t)\Big]\Big] \\
&= \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} (B_{\theta^\star}V_{\theta^\star}^{\pi^\star})(s_t,\pi^\star(s_t)) - Q_t(s_t,\pi^k(s_t))\Big]\Big] \\
&= \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} (B_{\theta^\star}V_{\theta^\star}^{\pi^\star})(s_t,\pi^\star(s_t)) - (B_{\theta^\star}V_t)(s_t,\pi^\star(s_t))\Big]\Big] \\
&\quad + \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} (B_{\theta^\star}V_t)(s_t,\pi^\star(s_t)) - Q_t(s_t,\pi^k(s_t))\Big]\Big] \qquad \text{(E.4)} \\
&= \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} \gamma\cdot\mathbb{E}_{s'\sim P_{\theta^\star}(\cdot\,|\,s_t,\pi^\star(s_t))}[V_{\theta^\star}^{\pi^\star}(s') - V_t(s')]\Big]\Big] \\
&\quad + \mathbb{E}\Big[\sum_{k=0}^{K-1}\mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} (B_{\theta^\star}V_t)(s_t,\pi^\star(s_t)) - Q_t(s_t,\pi^k(s_t))\Big]\Big], \qquad \text{(E.5)}
\end{aligned}
$$

where the first equality uses the Bellman optimality equation in (C.1), the condition $Q_t(s,\pi^k(s)) = V_t(s)$, and $Q_{\theta^\star}^\star(s,\pi^\star(s)) = V_{\theta^\star}^\star(s)$ for any state s, $t_k \le t < t_{k+1}$, and $k < K$. Here, the last equality uses the definition of $B_{\theta^\star}$. For the simplicity of discussions, we define functions $F_t, M_t \in \{\mathcal{S}\mapsto\mathbb{R}\}$ and the linear operator $\mathcal{T}\in\big\{\{\mathcal{S}\mapsto\mathbb{R}\}\mapsto\{\mathcal{S}\mapsto\mathbb{R}\}\big\}$ as

$$
\begin{aligned}
F_t(s) &= V_{\theta^\star}^{\pi^\star}(s) - V_t(s), \\
M_t(s) &= (B_{\theta^\star}V_t)(s,\pi^\star(s)) - Q_t(s_t,\pi^k(s)), \\
(\mathcal{T}f)(s) &= \mathbb{E}_{s'\sim P_{\theta^\star}(\cdot\,|\,s,\pi^\star(s))}[f(s')], \qquad \text{(E.6)}
\end{aligned}
$$

for any state s and function $f\in\{\mathcal{S}\mapsto\mathbb{R}\}$. Here, we denote by $\{\mathcal{S}\mapsto\mathbb{R}\}$ the class of all the functions defined on $\mathcal{S}$. By the definitions of F_t, M_t, and $\mathcal{T}$ in (E.6), it is clear that

$$
F_t(s) = M_t(s) + \gamma\cdot\big(\mathcal{T}F_t\big)(s), \qquad \text{(E.7)}
$$

for any state $s\in\mathcal{S}$. Then, we introduce the following lemma to bound F_t by (E.7).

Lemma E.1. *For the operator $\mathcal{T}$ defined in (E.6), two arbitrary bounded functions f, m defined on the state space $\mathcal{S}$, and any $\gamma\in[0,1)$, if*

$$
f(s) = m(s) + \gamma\cdot(\mathcal{T}f)(s) \qquad \text{(E.8)}
$$

holds for any state $s\in\mathcal{S}$, then it holds that for any state $s\in\mathcal{S}$,

$$
f(s) = \sum_{\tau=0}^{\infty}\gamma^\tau\cdot\big(\underbrace{(\mathcal{T}\circ\ldots\circ\mathcal{T})}_{\tau\text{ times}}m\big)(s). \qquad \text{(E.9)}
$$

Proof of Lemma E.1. By the condition in (E.8), we have

$$\begin{aligned} f(s) &= m(s) + \gamma \cdot \Big(\mathcal{T}\big(m + \gamma \cdot (\mathcal{T} f)\big)\Big)(s) \\ &= m(s) + \gamma \cdot (\mathcal{T} m)(s) + \gamma^2 \cdot \big((\mathcal{T} \circ \mathcal{T}) f\big)(s), \end{aligned} \tag{E.10}$$

where the last equality relies on the linearity of the operator $\mathcal{T}$. Repeating the process in (E.10) for $N \in \mathbb{N}$ times, we have that

$$f(s) = \gamma^{N+1} \cdot \big(\underbrace{(\mathcal{T} \circ \ldots \circ \mathcal{T})}_{(N+1) \text{ times}} f\big)(s) + \sum_{\tau=0}^{N} \gamma^{\tau} \cdot \big(\underbrace{(\mathcal{T} \circ \ldots \circ \mathcal{T})}_{\tau \text{ times}} m\big)(s). \tag{E.11}$$

Since both f and m are bounded functions, we use (E.6) to know that $\underbrace{(\mathcal{T} \circ \ldots \circ \mathcal{T})}_{\tau \text{ times}} f$ and $\underbrace{(\mathcal{T} \circ \ldots \circ \mathcal{T})}_{\tau \text{ times}} m$ are also bounded for any $\tau \in \mathbb{N}$. As $\gamma \in [0, 1)$, we let N tend to the infinity to transform (E.11) to

$$f(s) = \sum_{\tau=0}^{\infty} \gamma^{\tau} \cdot \big(\underbrace{(\mathcal{T} \circ \ldots \circ \mathcal{T})}_{\tau \text{ times}} m\big)(s),$$

for any state $s \in \mathcal{S}$. Then, we conclude the proof for Lemma E.1. □

By (E.7) and Lemma E.1, we have

$$F_t(s) = \sum_{\tau}^{\infty} \gamma^{\tau} \cdot \big(\underbrace{(\mathcal{T} \circ \ldots \circ \mathcal{T})}_{\tau \text{ times}} M_t\big)(s)$$

Recalling the definition of the optimal γ-discounted visitation measure in (C.2), we further have

$$F_t(s) = \frac{1}{1-\gamma} \cdot \mathbb{E}_{s' \sim \nu^{\star}(\cdot \mid s)}[M_t(s')]. \tag{E.12}$$

Plugging (E.12) and (E.6) into (E.5), we have

$$\begin{aligned} &\mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} V_{\theta^{\star}}^{\pi^{\star}}(s_t) - V_t(s_t)\Big]\Big] \\ &= \mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} \frac{1}{1-\gamma} \cdot \mathbb{E}_{s' \sim \nu^{\star}(\cdot \mid s)}\big[(B_{\theta^{\star}} V_t)(s_t, \pi^{\star}(s_t)) - Q_t(s_t, \pi^k(s_t))\big]\Big]\Big] \\ &= \frac{1}{1-\gamma} \cdot \mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}_{s \sim \nu^{\star}(\cdot \mid s)}\big[(B_{\theta^{\star}} V_t)(s, \pi^{\star}(s)) - Q_t(s, \pi^{\star}(s))\big]\Big]\Big] \\ &\quad + \frac{1}{1-\gamma} \cdot \mathbb{E}\Big[\sum_{k=0}^{K-1} \mathbb{E}_{\pi^k}\Big[\sum_{t=t_k}^{t_{k+1}-1} \mathbb{E}_{s \sim \nu^{\star}(\cdot \mid s)}\big[(Q_t(s, \pi^{\star}(s)) - Q_t(s, \pi^k(s))\big]\Big]\Big]. \end{aligned} \tag{E.13}$$

Multiplying $(1-\gamma)$ on the two sides of (E.13), we prove the second part of Lemma D.2. □

E.2. Proof of Proposition C.2

Proof of Proposition C.2. We now prove that the value iteration algorithm with a truncated horizon U (Algorithm 5) satisfies the definition of ϵ-optimal planner (Definition C.1), where U is dependent on ϵ. For notational simplicity, we denote $\max_{s \in \mathcal{S}}$ and $\max_{a \in \mathcal{A}}$ as $\max_s$ and $\max_a$.

Let

$$\epsilon^\dagger = \max_{s,a} \left| Q^{(1)}(s,a) - r(s,a) - \gamma (P V^{(1)})(s,a) \right|. \tag{E.14}$$

Note that the convergence analysis of the value iteration algorithm in Sutton & Barto (2018) gives

$$\max_{s,a} \left| Q^{(1)}(s,a) - Q^{(2)}(s,a) \right| \leq \gamma^{U-2} \max_{s,a} \left| Q^{(U-1)}(s,a) - Q^{(U)}(s,a) \right|,$$

which implies

$$\max_{s,a} \left| Q^{(1)}(s,a) - Q^{(2)}(s,a) \right| \leq \gamma^{U-2} L. \tag{E.15}$$

We have

$$\begin{aligned}
\epsilon^\dagger &= \max_{s,a} \Big| Q_\theta^{(1)}(s,a) - r(s,a) - \gamma (P V^{(2)})(s,a) \\
&\qquad + \gamma \mathbb{E}_{s' \sim P(\cdot \,|\, s,a)} \left[V^{(1)}(s') - V^{(2)}(s') \right] \Big| \\
&= \gamma \cdot \max_{s,a} \left| \mathbb{E}_{s' \sim P(\cdot \,|\, s,a)} \left[V^{(1)}(s') - V^{(2)}(s') \right] \right| \\
&= \gamma \cdot \max_{s,a} \left| \mathbb{E}_{s' \sim P(\cdot \,|\, s,a)} \left[\max_a Q^{(1)}(s',a) - \max_a Q^{(2)}(s',a) \right] \right| \\
&\leq \gamma \cdot \max_{s,a} \left| \mathbb{E}_{s' \sim P(\cdot \,|\, s,a)} \left[\max_a \left| Q^{(1)}(s',a) - Q^{(2)}(s',a) \right| \right] \right| \\
&\leq \gamma^{U-1} L,
\end{aligned} \tag{E.16}$$

where the first and third equalities are based on Algorithm 5, the second last inequality uses the contraction property of the maximum operator, and the last inequality uses (E.15). To let $\epsilon^\dagger < \epsilon$, it suffices to set $U \geq 1 + \lceil \log_\gamma(\epsilon / L) \rceil$. Note that the policy π returned by Algorithm 5 satisfies $\pi(s) = \operatorname{argmax}_a Q^{(1)}(s,a)$. Thus, we prove Proposition C.2. □

F. Linear Special Case

We specialize RAFA to a linear setting and characterize the Bayesian regret. In particular, we define a Bayesian variant of linear kernel MDPs (Yang & Wang, 2020; 2019; Cai et al., 2020; Zhou et al., 2021b). Here, $\mathbb{E}_{s' \sim P_\theta(\cdot \,|\, s,a)} V(s')$ is linear in a feature $\psi_V(s,a) \in \mathbb{R}^d$ for an arbitrary parameter $\theta \in \mathbb{R}^d$, while the prior and posterior distributions of the data-generating parameter $\theta^\star \in \mathbb{R}^d$ are Gaussian. Specifically, $\psi_V(s,a)$ maps the value function V and the state-action pair (s,a) to a d-dimensional vector. Recall that ρ is the initial distribution of states, t is the step index, and T is the total number of steps. Also, $\mathbb{P}_t$ is the posterior distribution at the t-th step.

Definition F.1 (Bayesian Linear Kernel MDP (Ghavamzadeh et al., 2015; Yang & Wang, 2020; 2019; Cai et al., 2020; Zhou et al., 2021b))**.** A Bayesian linear kernel MDP M satisfies

$$V(s_{t+1}) \,|\, s_t, a_t \sim \mathcal{N}(\psi_V(s_t,a_t)^\top \theta, 1)$$

for all $t \geq 0$, $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$, $s_{t+1} \sim P_\theta(\cdot \,|\, s_t, a_t)$, $\theta \in \mathbb{R}^d$, as well as all value function V. Here, $\psi_V(s,a)$ maps V and (s,a) to a d-dimensional vector, which satisfies $\|\psi_V(s,a)\|_2 \leq R$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ and all V. Also, M also satisfies $|\mathbb{E}_{s_0 \sim \rho} V(s_0)| \leq R$ for all V. Here, R is a positive constant that is independent of t and T. The prior distribution of the data-generating parameter $\theta^\star \in \mathbb{R}^d$ is $\mathcal{N}(0, \lambda I_d)$, where λ is a positive constant. Here, ψ_V is known

and $\theta^\star$ is unknown. Without loss of generality, we assume that the reward function is deterministic and known, i.e., $(B_\theta V)(\cdot,\cdot) = r(\cdot,\cdot) + \gamma \cdot (P_\theta V)(\cdot,\cdot)$ for a known reward function r and any θ.

By Definition F.1, we obtain the closed form of the posterior $\mathbb{P}_t$ as follows,

$$\theta \,|\, \mathcal{D}_t \sim \mathcal{N}(\widehat{\theta}_t; \Sigma_t^{-1}),$$

where

$$\widehat{\theta}_t = \Big(\lambda I_d + \sum_{i=0}^{t-1} \psi_{V_i}(s_i,a_i)\psi_{V_i}(s_i,a_i)^\top\Big)^{-1}\Big(\sum_{i=0}^{t-1} \psi_{V_i}(s_i,a_i)V_i(s_{i+1})\Big) \tag{F.1}$$

and

$$\Sigma_t = \lambda I_d + \sum_{i=0}^{t-1} \psi_{V_i}\left(s_i,a_i\right)\psi_{V_i}\left(s_i,a_i\right)^\top. \tag{F.2}$$

Hence, the posterior entropy is

$$H_t = H(\mathbb{P}_t) = 1/2 \cdot \log(\det(\Sigma_t)) + d/2 \cdot (1 + \log(2\pi)). \tag{F.3}$$

We specialize the switching condition in Algorithm 8 as follows,

$$H_{t_k} - H_t = 1/2 \cdot \log(\det(\Sigma_{t_k})) - 1/2 \cdot \log(\det(\Sigma_t)) > \log 2, \tag{F.4}$$

which is equivalent to $\det(\Sigma_{t_k}) > 4 \cdot \det(\Sigma_t)$. This switching condition is also similarly adopted in work for RL (Zhou et al., 2021b; Abbasi-Yadkori & Szepesvári, 2015). As a result, we have

$$\det(\Sigma_{t_k}) \le 4 \cdot \det(\Sigma_t) \tag{F.5}$$

for all $t_k \le t < t_{k+1}$ and $k < K$.

Verification of Assumption C.5 We verify the regularity assumption (Assumption C.5.) on Bayesian linear kernel MDPs as follows. By (F.4), the condition

$$H_{t_1} - H_{t_2} = 1/2 \cdot \log(\det(\Sigma_{t_1})) - 1/2 \cdot \log(\det(\Sigma_{t_2})) \le \log 2$$

is equivalent to $\det(\Sigma_{t_1}) \le 4\det(\Sigma_{t_2})$. Since $t_1 < t_2$ and the posterior variance matrix is positive definite, we have $\Sigma_{t_1}^{-1} \succeq \Sigma_{t_1}^{-1}$ and $\det(\Sigma_{t_2}^{-1}) \le 4\det(\Sigma_{t_1}^{-1})$. By the definition of the information gain and (F.4), we have

$$\begin{aligned} I(\theta; \xi_{(s,a)} \,|\, \mathcal{D}_t) &= H(\theta \,|\, \mathcal{D}_t) - H(\theta \,|\, \xi_{(s,a)}, \mathcal{D}_t) \\ &= 1/2 \cdot \log\left(\frac{\det(\psi_{V_{t_2}}(s,a)\psi_{V_{t_2}}^\top(s,a) + \Sigma_t)}{\det(\Sigma_t)}\right) \\ &= 1/2 \cdot \log\big(1 + \psi_{V_{t_2}}(s,a)^\top \Sigma_t^{-1} \psi_{V_{t_2}}(s,a)\big), \end{aligned} \tag{F.6}$$

for $t = t_1$ and t_2. Here, the last equality uses the matrix determinant lemma.

Plugging (F.9) into (F.6), we have

$$\begin{aligned} I(\theta; \xi_{(s,a)} \,|\, \mathcal{D}_{t_2}) &= 1/2 \cdot \log(1 + \psi_{V_{t_2}}(s,a)^\top \Sigma_t^{-1} \psi_{V_{t_2}}(s,a)) \\ &= 1/2 \cdot \log(1 + \psi_{V_{t_2}}(s,a)^\top \Sigma_t^{-1} \psi_{V_{t_2}}(s,a)) \\ &\ge \log(1+d)/(2d) \cdot \|\psi_{V_{t_2}}(s,a)\|^2_{\Sigma_{t_2}^{-1}}, \end{aligned} \tag{F.7}$$

where the second equality uses the matrix determinant lemma and the first inequality uses the fact that $\log(1+x)/x$ is an increasing function for $x \geq 0$ and

$$\begin{aligned}
0 &\leq \psi_{V_{t_2}}(s,a)^\top \Sigma_t^{-1} \psi_{V_{t_2}}(s,a) \\
&\leq \psi_{V_{t_2}}(s,a)^\top \big(\psi_{V_{t_2}}(s,a)\psi_{V_{t_2}}(s,a)^\top\big)^{-1} \psi_{V_{t_2}}(s,a) \\
&= \mathrm{tr}\Big(\psi_{V_{t_2}}(s,a)\psi_{V_{t_2}}(s,a)^\top \big(\psi_{V_{t_2}}(s,a)\psi_{V_{t_2}}(s,a)^\top\big)^{-1}\Big) \\
&= d. \qquad \text{(F.8)}
\end{aligned}$$

Here, the first inequality uses the nonnegativity of a quadratic form, the first equality uses $\mathrm{tr}(a^\top b) = \mathrm{tr}(ba^\top)$ for two arbitrary vectors a and b, and the second inequality uses (F.2). By (F.2), we know that

$$\|\psi_{V_{t_2}}(s,a)\|^2_{\Sigma_{t_1}^{-1}} \leq 4 \cdot \|\psi_{V_{t_2}}(s,a)\|^2_{\Sigma_{t_2}^{-1}}, \qquad \text{(F.9)}$$

where the inequality invokes the following lemma (Lemma F.2).

Lemma F.2 (Lemma 12 in Abbasi-Yadkori et al. (2011))**.** *Suppose $A, D \in \mathbb{R}^{d\times d}$ are two positive definite matrices satisfying that $A \succeq D$, then for any $\mathbf{x} \in \mathbb{R}^d$, $\|\mathbf{x}\|_A \leq \|\mathbf{x}\|_D \cdot \sqrt{\det(A)/\det(D)}$.*

Rearranging (F.7) , we have

$$\begin{aligned}
\frac{8d}{\log(1+d)} \cdot I(\theta; \xi_{(s,a)} \,|\, \mathcal{D}_{t_2}) &\geq 4 \cdot \|\psi_{V_{t_2}}(s,a)\|^2_{\Sigma_{t_2}^{-1}} \\
&\geq \|\psi_{V_{t_2}}(s,a)\|^2_{\Sigma_{t_1}^{-1}} \\
&\geq \log(1 + \psi_{V_{t_2}}(s,a)^\top \Sigma_{t_1}^{-1} \psi_{V_{t_2}}(s,a)) \\
&= 2 \cdot I(\theta; \xi_{(s,a)} \,|\, \mathcal{D}_{t_1}), \qquad \text{(F.10)}
\end{aligned}$$

where the second inequality uses (F.9), the last inequality uses the fact that $x \geq \log(1+x)$ for any $x \geq 0$, and the last equality use (F.6). By (F.10), we know that Bayesian linear kernel MDPs (Definition F.1) satisfy Assumption C.5 with the coefficient $\eta = d/\log(1+d)$.

Analysis of the Cumulative Posterior Entropy $H_0 - H_T$. Next, we study the upper bound of the cumulative Information gain $H_0 - H_T$ in Bayesian linear kernel MDPs. By the definition of Σ_t in (F.2), we have $\log\det(\Sigma_0) = d \cdot \log\lambda$ and

$$\begin{aligned}
\log\det(\Sigma_T) &= \log\det\Big(\lambda I_d + \sum_{t=0}^{T-1} \psi_{V_t}(s_t,a_t)\psi_{V_t}^\top(s_t,a_t)\Big) \\
&\leq d \cdot \log\Big(1/d \cdot \mathrm{tr}\big(\lambda I_d + \sum_{t=0}^{T-1} \psi_{V_t}(s_t,a_t)\psi_{V_t}^\top(s_t,a_t)\big)\Big) \\
&= d \cdot \log\Big(1/d \cdot \big(\lambda d + \sum_{t=0}^{T-1} \|\psi_{V_t}(s_t,a_t)\|_2^2\big)\Big) \\
&\leq d \cdot \log(\lambda + TR^2/d) \qquad \text{(F.11)}
\end{aligned}$$

almost surely. Here, the first inequality uses the relationship between the trace and the determinant of a square matrix, the second equality uses $\mathrm{tr}(a^\top b) = \mathrm{tr}(ba^\top)$ for two arbitrary vectors a and b, and the last inequality uses the fact that $\|\psi_V(s,a)\|_2$ is upper bounded by R for all $(s,a) \in \mathcal{S}\times\mathcal{A}$ and V. Hence, we have

$$H_0 - H_T = \mathcal{O}\big(d \cdot \log(1 + TR^2/(d\lambda))\big) \qquad \text{(F.12)}$$

almost surely.

Regret Bounds. With the verification of Assumption C.5 and the analysis of the upper bound of the cumulative Information gain $H_0 - H_T$ in Bayesian linear kernel MDPs, we are ready to specialize the theorems in Appendix C in Bayesian linear kernel MDPs if we determine an appropriate L (the bound of the value). We analyze the bound of $V(s')$ for $s' \sim P_\theta(s,a)$ and $\theta \sim \mathcal{N}(\mu, \lambda I_d)$. Define that $\widetilde{\epsilon} \sim \mathcal{N}(0,1)$. By Definition F.1, we have

$$\begin{aligned}
|V(s')| &= |\widetilde{\epsilon} + \psi_V(s,a)^\top \theta| \\
&\leq |\widetilde{\epsilon}| + |\psi_V(s,a)^\top (\theta - \mu)| + |\psi_V(s,a)^\top \mu| \\
&\leq |\widetilde{\epsilon}| + \|\psi_V(s,a)\|_2 \cdot \|\theta - \mu\|_2 + \|\psi_V(s,a)\|_2 \cdot \|\mu\|_2 \\
&\leq |\widetilde{\epsilon}| + R \cdot \|\theta\|_2 + R \cdot \|\mu\|_2,
\end{aligned} \tag{F.13}$$

where the first inequality uses the triangle inequality, the second inequality uses the Cauchy-Schwartz inequality, and the last inequality uses the definition of R in Definition F.1. By Definition F.1, (F.13), and the tail behavior of the Gaussian distribution (Ghosh, 2021), we have

$$|V(s)| \leq \sqrt{2 \cdot \log(2/\delta)} + R \cdot \|\mu\|_2 + R \cdot \sqrt{2\lambda d \cdot \log(2d\delta)}$$

for any $s \in \mathcal{S}$ and value function V with probability at least $1 - \delta$. Since the prior distribution of θ is $\mathcal{N}(0, \lambda I_d)$, it is natural to restrict μ such that $\|\mu\|_2 \leq cd \cdot \log(2d))$ for some absolute constant c. Then, we apply the union bound of all T value functions in RAFA and the variants (Algorithms 6, 7, and 8) to have

$$\begin{aligned}
|V_t(s)| &\leq \sqrt{2 \cdot \log(2T/\delta)} + R \cdot \|\mu\|_2 + R \cdot \sqrt{2\lambda d \cdot \log(2dT\delta)} \\
&\leq (c+1)R \cdot \sqrt{2\lambda d \log(2dT/\delta))}
\end{aligned} \tag{F.14}$$

for any $t < T$, $s \in \mathcal{S}$, and value function V with probability at least $1 - \delta$. Hence, we can select $L = (c+1)R \cdot \sqrt{2\lambda d \log(2dT/\delta)}$ in Theorems C.7, C.8, and C.10. By specializing Theorems C.7, C.8, and C.10, we summarize the corresponding regret bounds in Table 3 for Algorithms 6, 7, and 8, respectively. Here, we choose the planning suboptimality of $\texttt{PL}^\epsilon$ to be $\epsilon = \mathcal{O}(1/\sqrt{T})$ and all the bounds hold with probability at least $1 - \delta$.

Algorithm	Bayesian Regret
RAFA (Algorithm 6)	$\mathcal{O}((1-\gamma)^{-1}(\kappa+1)\sqrt{d^3T} \cdot \log(dT/\delta))$
RAFA with Optimistic Bonus (Algorithm 7)	$((1-\gamma)^{-1}\sqrt{d^3T} \cdot \log(dT/\delta))$
RAFA with Posterior Sampling (Algorithm 8)	$\mathcal{O}((1-\gamma)^{-1}\sqrt{d^3T} \cdot \log(dT/\delta))$

Table 3. Bayesian regret of variants of RAFA in Bayesian linear kernel MDPs (see Definition F.1). Here, we choose the planning suboptimality of $\texttt{PL}^\epsilon$ to be $\epsilon = \mathcal{O}(1/\sqrt{T})$ and all the bounds hold with probability at least $1 - \delta$.

G. More Experiments

In what follows, we provide the detailed setups and additional results of our experiments.

G.1. Game of 24

Task Setup. Figure 10 gives an illustrative example for Game of 24.

[Illustrative example for Game of 24]

- Numbers: [2, 5, 8, 11]
- Arithmetic Operations: $[+, -, \times, /, (,)]$
- **Solution**:

$$(11 - 5) \times 8/2 = 24$$

Figure 10. An illustrative example of the Game of 24. The player uses combinations of basic arithmetic operations with four given numbers to get 24.

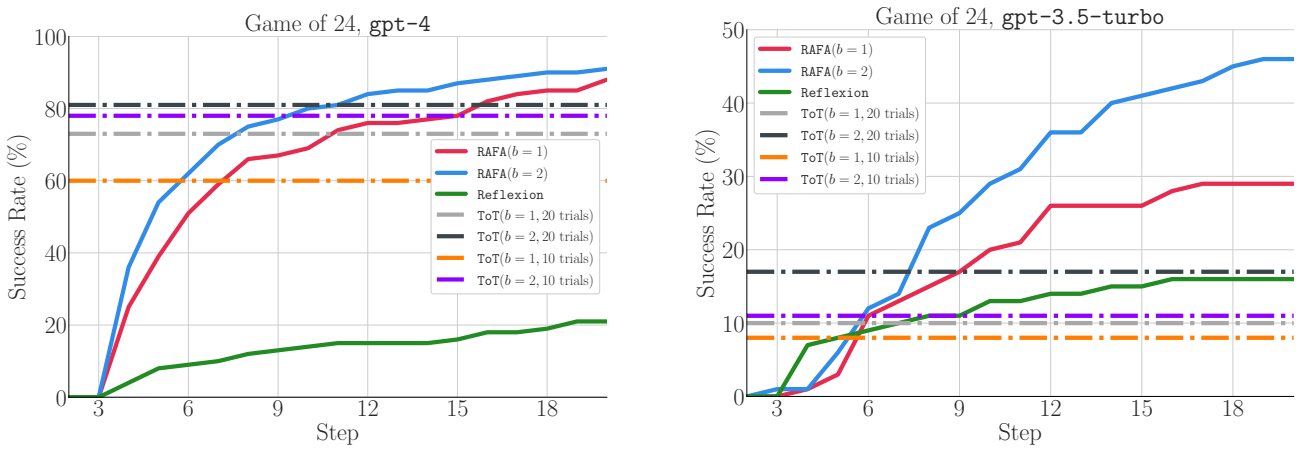


Figure 11. Sample efficiency on Game of 24. `RAFA` agent achieves strong performance due to an orchestration of reasoning and acting. The success rate at a given step is the number of tasks that is solved within the given step.

Following Yao et al. (2023a), we use the same subset indexed 901-1,000 from a total of 1,362 tasks collected from `4nums.com`. The index is arranged from easy to hard by human solving time so the subset is relatively challenging. The agent receives a reward of 1 if the proposed formula is correct and the proposed formula is accepted and concatenated into the state; if the final result is exactly 24, the agent receives a reward of 10, and the episode terminates. Otherwise, the agent receives a reward of 0, and the proposed formula is not accepted. We limit the maximum trials for each task to 20 to avoid meaningless retries. The task is successful if the agent receives a return larger than 10 [1] (i.e., find a valid solution within 20 steps). We report the final success rate and sample efficiency for each method on the subset of 100 tasks. Notably, a task is considered successful if the `RAFA` agent returns one and only one correct formula, which is more strictly evaluated than Tree of Thoughts (`ToT`, Yao et al. (2023a)): we allow open-loop agents like `ToT` to retry 20 times and consider them successful if they generate a valid solution in any of the 20 trials. For `CoT` (Wei et al., 2022) and `Reflexion` (Shinn et al., 2023) agents, we allow them to reflect on the environment's feedback but require them to generate a plan immediately without sophisticated reasoning.

RAFA Setup. In the Game of 24, the `RAFA` agent uses ToT as the planner, regenerates a plan when the agent receives a zero reward and continues acting according to the previous plan when the agent receives a positive reward. We set the base ToT planner with beam search width $b = 1, 2$ and use both `gpt-3.5-turbo` and `gpt-4` to test the RAFA's boost-up over LLM agents with different reasoning abilities. We set the temperature $t = 0.2$ by default to favor rigorous reasoning

[1] For `gpt-3.5-turbo`, we report the success rate when the agent receives a return no less than 3 (i.e., find all sub-steps to get 24 but not necessarily generate a whole correct formula). This is because `ToT` with `gpt-3.5-turbo` is known to suffer from correctly get a whole formula due to limited reasoning ability and non-perfect prompts. See https://github.com/princeton-nlp/tree-of-thought-llm/issues/24 for more details.

and $t = 0.7$ for majority voting.

Reduced Hallucination Through Interaction. A comprehensive review of various method proposals revealed significant hallucination, especially with `gpt-3.5-turbo`. A common hallucination is that the agent believes she can reuse the same number (e.g. using the number 2 twice as illustrated in Figure 2). `RAFA` efficiently mitigates such hallucination by actively interacting with the environment, displaying exceptional hallucination resistance and improved performance.

Enhanced Efficiency Through Planning. Evidenced in Figure 5, the `RAFA` agent substantially surpasses the `Reflexion` baseline, reflecting heightened efficiency and minimized regret by negating careless trials. For example, without carefully planning, agent may give negative answers, e.g,"Impossible to obtain 24 with the given numbers, or unchecked answers, e.g.,"Answer: 6 * 9 / (3 - 2) = 24". This reduction of careless trails is especially achieved when a strong backbone LLMs (e.g., `gpt-4`) is used, even with a basic planning method, such as BFS with $B = 1$.

Ablation Study. The `RAFA` agent's performance is dissected by individually examining its components: (1) Planning modules or model/elite LLM, (2) Reflection modules or critic LLM, and (3) Different LLMs. Results, displayed in Table 3 and Figure 5, affirm the substantial contribution of each segment to the aggregate performance. Compared to absent or rudimentary zero-shot planning, a basic planner markedly enhances overall performance. However, augmenting planner strength only offers marginal performance enhancements. Both critic LLM and robust LLM usage emerge as pivotal for optimal performance.

G.2. ALFWorld

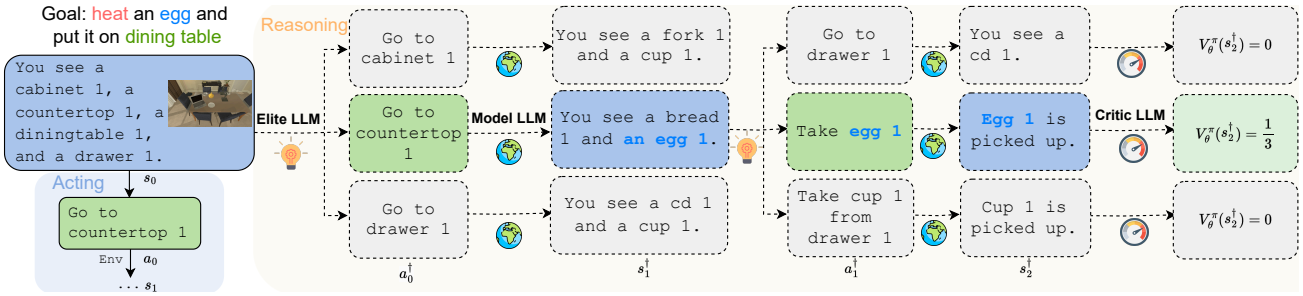


Figure 12. An illustration of `RAFA` in the ALFWorld environment.

Task Setup. The action space of ALFWorld consists of high-level actions such as "heat a potato with a microwave", which is executed in the underlying embodied simulator through low-level action primitives. The egocentric visual observations of the simulator are translated into natural language before being provided to the agent. The state is the history of the observations. If a task goal can be precisely achieved by the agent, it will be counted as a success.

RAFA Setup. In the ALFWorld environment, the `RAFA` planner is instantiated as Breadth First Search (BFS). Specifically, B and U are both set to 2, and we use `gpt-3` (`text-davinci-003`) for the `Critic`, `Model`, and `Elite` modules. Besides, since it is challenging to prompt the LLM with the stored full trajectories in the memory buffer due to the token limit, we make the following modifications: the `Model` LLM instance uses only the partial trajectory executed so far in the current episode, and the `Elite` LLM instance uses the same partial executed trajectory with additional model-generated state-action pairs during the planning subroutine. When switching is triggered after 20 failed timesteps (i.e., an episode), a summary from the failure trajectory is generated by `gpt-4` and added to the `Critic` prompt.

Reduced Hallucination Through Interaction. The baselines are more likely to hallucinate when the target object is not found after exploring many locations. On the other hand, the critic LLM used in `RAFA` is able to probe the hallucination by generating the summary "In this environment, my critic assigned a 1/3 value after taking a knife. However, the task is to take and cool a tomato." and avoid it in the next episode. Therefore, `RAFA` is more sample-efficient due to an orchestration of reasoning and acting and the ability to mitigate hallucination through interaction.

Ablation Study. To better understand the role that the planning subroutine plays in the RAFA algorithm, we conduct ablation studies on the search depth U and search breadth B. The results are shown in Figure 13 and 14, respectively. We observe that when setting the search depth to $B = U = 2$, the success rate is higher than when setting the search depth to $U = 1$ or setting the search breadth $B = 1$, especially at the initial episode. This indicates that the reasoning ability of RAFA is enhanced through the planning subroutine. Besides, the algorithm is also more sample-efficient when setting $B = U = 2$, indicating a better capacity for learning and planning through interaction and reasoning.

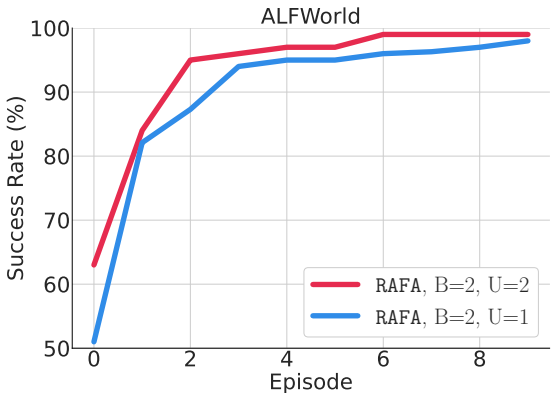


Figure 13. Ablation on the search depth U in the ALFWorld environment.

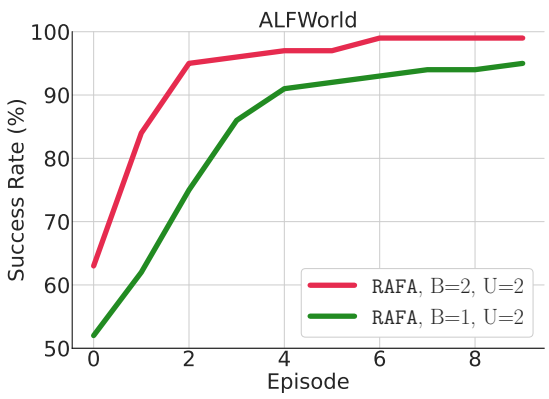


Figure 14. Ablation on the search breadth B in the ALFWorld environment.

G.3. BlocksWorld

Task Setup. The reported success rates are averaged in tasks that require different minimum steps. Specifically, the evaluation is conducted in 57 4-step tasks and 114 6-step tasks. We set the state as the current arrangement of the blocks and the actions contain Stack, Unstack, Put, and Pickup, coupled with a block being operated.

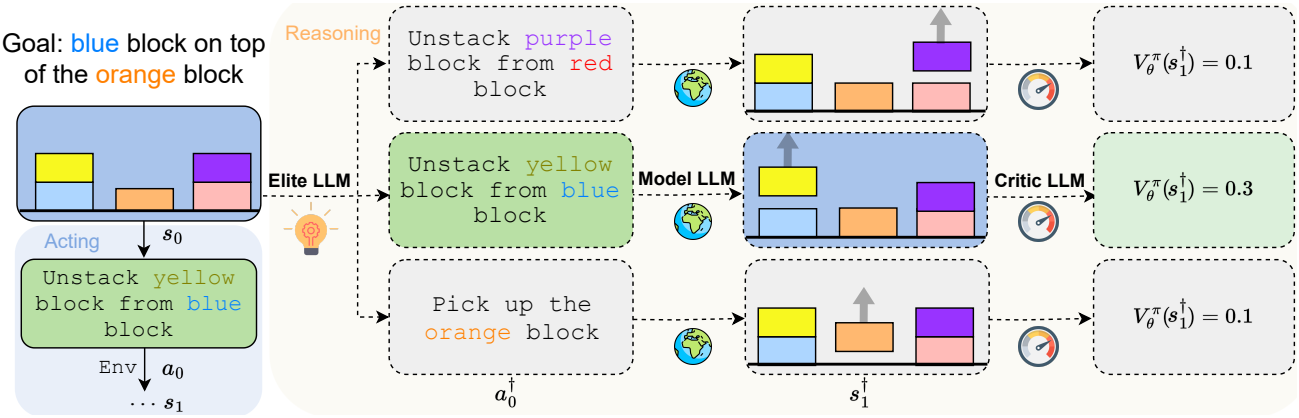


Figure 15. Illustration of RAFA in the BlocksWorld environment.

RAFA Setup. The search space is up to 5^4 for a 4-step task and is up to 5^6 for a 6-step task. For 4-step tasks, RAFA can achieve over 50% success rate within 8 learning steps with `Vicuna-13B(v1.3)` and achieve over 80% success rate within 8 learning steps with `Vicuna-33B(v1.3)`. For 6-step tasks, RAFA can achieve over 40% success rate within 20 learning steps with `Vicuna-13B(v1.3)` and achieve over 50% success rate within 20 learning steps with `Vicuna-33B(v1.3)`. Empirical results show that `Vicuna` could produce wrong state transition in the planning phase. RAFA can mitigate hallucination with feedback from failure trajectories and active exploration. One can draw such a conclusion by comparing RAFA with RAP as RAP does not receive feedback from the real environment.

G.4. Tic-Tac-Toe

Task Setup. Tic-Tac-Toe (Beck, 2008) is a competitive game in which two players take turns to mark a three-by-three grid with X or O, and a player succeeds when their marks occupy a diagonal, horizontal, or vertical line.

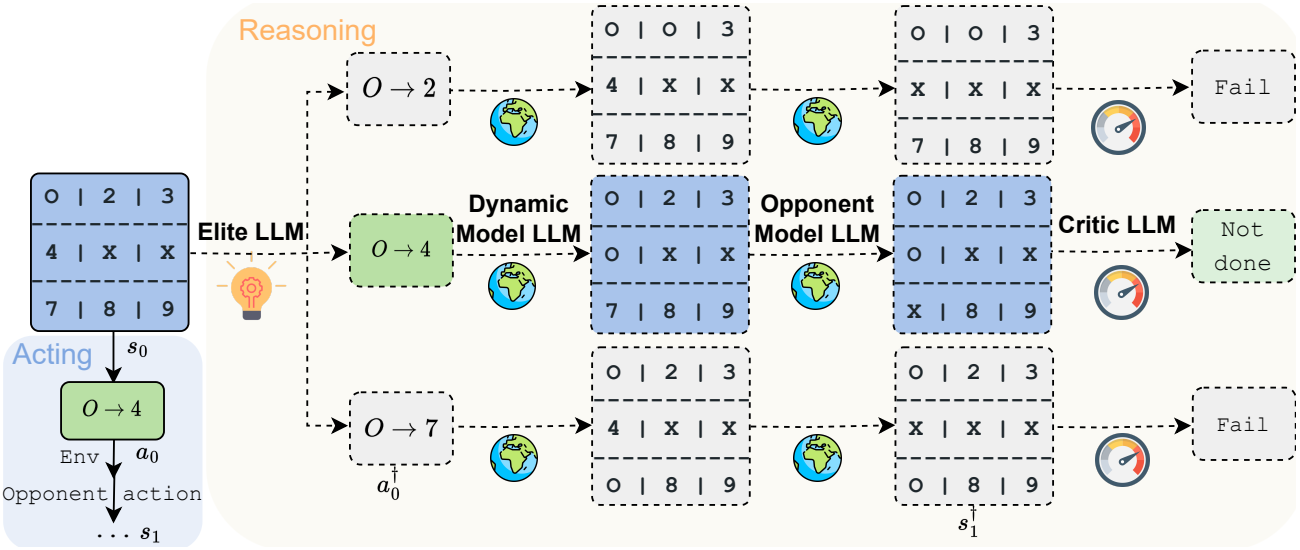


Figure 16. Illustration of RAFA (playing O) in the Tic-Tac-Toe game. States are represented by a numbered 3×3 grid and actions are represented by a number between 1-9. The opponent is considered part of the environment.

We adopt the convention that X plays first. As illustrated below in Figure 16, we use a numbered 3×3 grid to represent a state and a number between 1 and 9 to represent an action, which also illustrates the transition and reward function. Although Tic-Tac-Toe is a solved game with a forced draw assuming the best play from both players, it remains a challenge for LLMs to accomplish this task even when prompted to play only the optimal moves. We collected the battle outcomes between different LLM models in Table 4, where we notice that `gpt-4` performs worse when playing as "O". Thus, in our experiments, we let RAFA play as "O" and let baseline LLM models play as "X".

X wins : Tie : O wins		O	
		`gpt-3.5`	`gpt-4`
X	`gpt-3.5`	55% : 35% : 10%	90% : 0% : 10%
	`gpt-4`	65% : 15% : 20%	90% : 0% : 10%

Table 4. Probability of "X wins," "Tie," and "O wins" in Tic-Tac-Toe. The results are obtained by averaging over 20 simulated games.

RAFA Setup. For implementation, we set $B = 3$ and adopt MCTS to evaluate the proposed actions. We set $U = 4$ which is the maximum game depth. We set a prediction-based switching condition triggered when the prediction does not agree with the observation. Specifically, policy switches when one of the following events occurs:

- The RAFA agent takes an action and predicts the next state, which is different from the observed next state.
- Before the opponent takes an action, the RAFA agent tries to predict such an action, which is different from the actual action that the opponent takes.
- After the opponent takes an action, RAFA agent predicts the next state, which is different from the observed next state.

- The RAFA agent predicts the current game status (X wins, O wins, Tie, Not finished), which is different from the environment's feedback.

Besides, we use the ground truth of those predictions to update the agent's belief of the world, which also implicitly affects the agent's policy.

We define a discrete reward function with $r = -1, 0, 1$ corresponding to lose, tie, and win. The agent only gets rewards when the current episode is completed. We define the score of an agent as its expected reward which can be approximated by simulation. The empirical results are shown in figure 17. We conduct experiments using both gpt-4 as the backend. The score of RAFA ($B = 4$) increases as it interacts more with the environment. By analyzing the generated trajectories, we also notice that although RAFA agent is not perfect, it exploits the weakness of the baseline model well, which is why it almost never loses after 7 episodes.

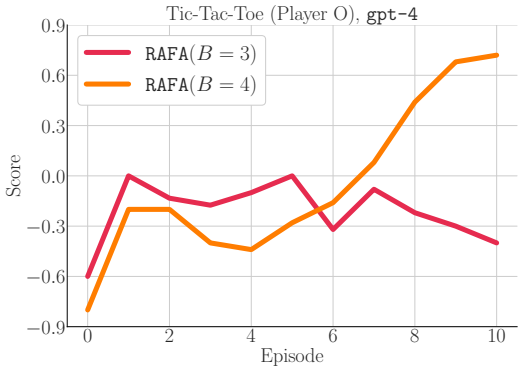


Figure 17. Score curves in the Tic-Tac-Toe game. We use gpt-4 as backend. Results are averaged across 10 simulations and smoothed with a window size of 5.

H. Prompts

In this section, we give details of the prompts used for each task.

H.1. Game of 24

Critic LLM. For the LLM instance of the Critic, we prompt it with the current action (formula) with reward and feedback from the environment. The critic is required to determine whether each formula is valid or not and give a "sure" or "impossible" label for the formula. We use critic prompts to generate reflection for formula proposal and evaluation, respectively.

Critic prompt (for formula proposal)

```
Now we would like to play a game of 24. That is, given 4 numbers, try to use them with
arithmetic operations (+ - * /) to get 24. Now we consider the following puzzle:
{input}.
Here is an attempt answer:
{answer}
And we have the following feedback:
{feedback}
```

```
Now using the above feedback, give 'sure' or 'impossible' labels for each formula with
left numbers from each step. Give 'sure' if the formula is correct and can lead to 24
and give 'impossible' if the formula is incorrect or illegal. First repeat the formula
with left numbers from each step above and then give the label, with the following
form: {{formula}} (left: {{left numbers}}): {{label}}.
```

Critic prompt (for formula evaluation)

```
Now we would like to play a game of 24. That is, given 4 numbers, try to use them with
arithmetic operations (+ - * /) to get 24. Now we consider the following puzzle:
{input}.
Here is an attempt answer:
{answer}
And we have the following feedback:
{feedback}
Now using the above feedback, give 'sure' or 'impossible' labels for left numbers from
each step. Give 'sure' if the formula is correct and left numbers can lead to 24 and
give 'impossible' if the formula is incorrect or illegal. First repeat the left numbers
from each step above and then give the label, with the following form: {{left
numbers}}: {{label}}.
```

Elite LLM. We adopt the same prompts used in Tree-of-Thoughts (Yao et al., 2023a) to propose and evaluate formulas, except that we concatenate the reflections from each step to avoid making repeated mistakes.

Elite prompt (for formula proposal)

```
Now we would like to play a game of 24. That is, given 4 numbers, try to use them with
arithmetic operations (+ - * /) to get 24.
Evaluate if given numbers can reach 24 and choose labels from 'sure', 'likely' and
'impossible'.
What you have learned about the puzzle are summarized below.
{reflections}
Now use numbers and basic arithmetic operations (+ - * /) to generate possible next
steps. Make sure use steps that is sure to leads to 24 and avoid steps that are
impossible to generate 24. Note that it is possible that we are considering
intermediate steps so the numbers of the input may be less than 4.
Example:
Input: 2 8 8 14
Possible next steps:
2 + 8 = 10 (left: 8 10 14)
8 / 2 = 4 (left: 4 8 14)
14 + 2 = 16 (left: 8 8 16)
2 * 8 = 16 (left: 8 14 16)
8 - 2 = 6 (left: 6 8 14)
14 - 8 = 6 (left: 2 6 8)
```

```
14 /  2 = 7 (left: 7 8 8)
14 - 2 = 12 (left: 8 8 12)
Example:
Input: 2 5 8
5 - 2 = 3 (left: 3 8)
5 * 2 = 10 (left: 10 8)
8 / 2 = 4 (left: 4 5)
Now try with the following input:
Input: {input}
Possible next steps:
 {input}
```

Elite prompt (for formula evaluation)

```
Now we would like to play a game of 24. That is, given 4 numbers, try to use them with
arithmetic operations (+ - * /) to get 24.
Evaluate if given numbers can reach 24 and choose labels from 'sure', 'likely' and
'impossible'.
What you have learned about the puzzle are summarized below.
{reflections}
If the given numbers are already in the feedback above, just give the answer. Otherwise
enumerate possible steps and try to give an approximate answer. Give the final answer
in a separated line.
 {input}
```

Elite prompt (for last step formula evaluation)

```
Now we would like to play a game of 24. That is, given 4 numbers, try to use them with
arithmetic operations (+ - * /) to get 24.
Evaluate if given numbers can reach 24 and choose labels from 'sure', 'likely' and
'impossible'.
What you have learned about the puzzle are summarized below.
{reflections}
Use numbers and basic arithmetic operations (+ - * /) to obtain 24. Given an input and
an answer, give a judgement (sure/impossible) if the answer is correct, i.e., it uses
each input exactly once and no other numbers, and reach 24.
Input: 4 4 6 8
Answer: (4 + 8) * (6 - 4) = 24
Judge:
sure
Input: 2 9 10 12
Answer: 2 * 12 * (10 - 9) = 24
Judge:
sure
```

```
Input: 4 9 10 13
Answer: (13 - 9) * (10 - 4) = 24
Judge:
sure
Input: 4 4 6 8
Answer: (4 + 8) * (6 - 4) + 1 = 25
Judge:
impossible
Input: 2 9 10 12
Answer: 2 * (12 - 10) = 24
Judge:
impossible
Input: 4 9 10 13
Answer: (13 - 4) * (10 - 9) = 24
Judge:
impossible
Input: {input}
Answer: {answer}
Judge:
```

For Chain-of-Thought baselines, we adopt the same methodology, and keep the original prompts except for adding reflections as below.

Elite prompt (for chain-of-thought proposals)

```
Now we would like to play a game of 24. That is, given 4 numbers, try to use them with
arithmetic operations (+ - * /) to get 24.
Evaluate if given numbers can reach 24 and choose labels from 'sure', 'likely' and
'impossible'.
What you have learned about the puzzle are summarized below.
{reflections}
Now just remember the tips from before (if any) and focus on the new task. Use numbers
and basic arithmetic operations (+ - * /) to obtain 24. Each step, you are only allowed
to choose two of the remaining numbers to obtain a new number.
Input: 4 4 6 8
Steps:
4 + 8 = 12 (left: 4 6 12)
6 - 4 = 2 (left: 2 12)
2 * 12 = 24 (left: 24)
Answer: (6 - 4) * (4 + 8) = 24
Input: 2 9 10 12
Steps:
12 * 2 = 24 (left: 9 10 24)
10 - 9 = 1 (left: 1 24)
24 * 1 = 24 (left: 24)
```

```
Answer: (12 * 2) * (10 - 9) = 24
Input: 4 9 10 13
Steps:
13 - 10 = 3 (left: 3 4 9)
9 - 3 = 6 (left: 4 6)
4 * 6 = 24 (left: 24)
Answer: 4 * (9 - (13 - 10)) = 24
Input: 1 4 8 8
Steps:
8 / 4 = 2 (left: 1 2 8)
1 + 2 = 3 (left: 3 8)
3 * 8 = 24 (left: 24)
Answer: (1 + 8 / 4) * 8 = 24
Input: 5 5 5 9
Steps:
5 + 5 = 10 (left: 5 9 10)
10 + 5 = 15 (left: 9 15)
15 + 9 = 24 (left: 24)
Answer: ((5 + 5) + 5) + 9 = 24
Input: {input}
```

H.2. ALFWorld

Critic LLM. For the LLM instance of the `Critic`, we prompt it with successful examples for each of the six task types to generate the value within each episode. Specifically, the critic LLM first determines if the value of the current state changes. If it changes, the critic LLM then generates the scalar value with the corresponding reason. We provide the following two examples in the put and cool tasks, respectively.

Critic prompt (for value generation), "put" task

```
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.
```

Critic prompt (for value generation), "cool" task

```
You are in the middle of a room. Looking quickly around you, you see a cabinet 16, a
cabinet 15, a cabinet 14, a cabinet 13, a cabinet 12, a cabinet 11, a cabinet 10, a
cabinet 9, a cabinet 8, a cabinet 7, a cabinet 6, a cabinet 5, a cabinet 4, a cabinet
3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 2, a countertop 1, a
diningtable 1, a drawer 5, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a fridge 1,
a garbagecan 1, a microwave 1, a safe 1, a sinkbasin 1, a stoveburner 4, a stoveburner
3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: cool some pan and put it in stoveburner.
> critic: My task requires three sub-goals in order: take a pan, cool the pan with
fridge, and put the pan in the stoveburner. My current state satisfies zero of the
three sub-goals. The value is 0/3=0.
> OK.
On the stoveburner 1, you see nothing.
> OK.
On the stoveburner 2, you see a pot 1.
> OK.
On the stoveburner 3, you see a pan 1.
> OK.
You pick up the pan 1 from the stoveburner 3.
> critic: Now I take a pan. My current state satisfies the first of the three
sub-goals: take a pan. The value is 1/3=0.333.
> OK.
The fridge 1 is closed.
> OK.
You cool the pan 1 using the fridge 1.
> critic: Now I cool the pan. My current state satisfies the first two of the three
sub-goals: take a pan and cool the pan with fridge. The value is 2/3=0.667.
> OK.
On the stoveburner 1, you see nothing.
> OK.
You put the pan 1 in/on the stoveburner 1.
```

```
> critic: Now I put the pan in the stoveburner. My current state satisfies all the
three sub-goals. The value is 3/3=1.
```

In addition to the prompt for value generation within each episode, we also prompt the `Critic` LLM to generate summary feedback according to the past failure trajectory. We use the same prompt for all six task types. Specifically, it contains the full failure trajectory (observation only), as well as the failure feedback at the end of each trajectory.

Critic prompt (for feedback generation)

```
You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a
cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a
countertop 3, a countertop 2, a countertop 1, a drawer 3, a drawer 2, a drawer 1, a
fridge 1, a garbagecan 1, a microwave 1, a shelf 3, a shelf 2, a shelf 1, a sinkbasin
1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: put a clean plate in countertop.
> critic: My task requires three sub-goals in order: take a plate, clean the plate with
sinkbasin, and put the plate on the countertop. My current state satisfies zero of the
three sub-goals. The value is 0/3=0.
> OK.
The drawer 1 is closed.
> OK.
You open the drawer 1. The drawer 1 is open. In it, you see a creditcard 2, a knife 1,
and a spatula 2.
> OK.
On the stoveburner 2, you see a pan 1.
> OK.
You pick up the pan 1 from the stoveburner 2.
> critic: Now I take a pan. My current state satisfies the first of the three
sub-goals: take a plate. The value is 1/3=0.167.
> OK.
On the sinkbasin 1, you see nothing.
> OK.
You put the pan 1 in/on the sinkbasin 1.
> OK.
The microwave 1 is closed.
> OK.
You open the microwave 1. The microwave 1 is open. In it, you see a potato 1.
> OK.
On the garbagecan 1, you see a soapbottle 2.
> OK.
The microwave 1 is open. In it, you see a potato 1.
> OK.
On the coffeemachine 1, you see nothing.
> OK.
On the countertop 2, you see a bread 1, a cellphone 2, a cellphone 1, a papertowelroll
1, a plate 2, and a soapbottle 1.
```

```
> OK.
The drawer 2 is closed.
> OK.
You open the drawer 2. The drawer 2 is open. In it, you see a spatula 1.
> OK.
On the sinkbasin 1, you see a pan 1.
> OK.
On the cabinet 3, you see a cup 1.
> OK.
On the countertop 1, you see a apple 2, a dishsponge 2, a potato 3, and a potato 2.
STATUS: FAIL
Failure feedback: In this environment, my critic assigned a 1/3 value after taking a
pan. However, the task is to take and clean a plate. I noticed that the plate was found
on countertop 2. In the next trial, I will go to countertop 2 to take the plate, then
go to a sinkbasin to clean the plate.

You are in the middle of a room. Looking quickly around you, you see a cabinet 20, a
cabinet 19, a cabinet 18, a cabinet 17, a cabinet 16, a cabinet 15, a cabinet 14, a
cabinet 13, a cabinet 12, a cabinet 11, a cabinet 10, a cabinet 9, a cabinet 8, a
cabinet 7, 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 two saltshaker in drawer.
> critic: My task requires four sub-goals in order: take the first saltshaker, put the
first saltshaker in the drawer, take the second saltshaker, and put the second
saltshaker in the drawer. My current state satisfies zero of the four sub-goals. The
value is 0/4=0.
> OK.
On the coffeemachine 1, you see a cup 1, a fork 4, and a potato 2.
> OK.
On the toaster 1, you see a pan 1 and a knife 1.
> OK.
On the countertop 3, you see a bowl 2, a houseplant 1, a knife 2, a peppershaker 1, a
saltshaker 4, a soapbottle 1, a spatula 2, and a tomato 3.
> OK.
You pick up the saltshaker 4 from the countertop 3.
> critic: Now I take the first saltshaker. My current state satisfies the first of the
four sub-goals: take the first saltshaker. The value is 1/4=0.25.
> OK.
The drawer 1 is open. In it, you see nothing.
> OK.
```

```
You put the saltshaker 4 in/on the drawer 1.
> critic: Now I put the first creditcard in the dresser. My current state satisfies the
first two of the four sub-goals: take the first saltshaker and put the first saltshaker
in the drawer. The value is 2/4=0.5.
> OK.
On the stoveburner 1, you see a bread 1.
> OK.
On the stoveburner 2, you see a potato 1, a pot 2, and a knife 3.
> OK.
On the stoveburner 3, you see a fork 1, a fork 2, and a cup 2.
> OK.
On the stoveburner 4, you see a peppershaker 1 and a pan 2.
> OK.
You pick up the peppershaker 1 from the stoveburner 4.
> critic: Now I take the second saltshaker. My current state satisfies the first three
of the four sub-goals: take the first saltshaker, put the first saltshaker in the
drawer, and take the second saltshaker. The value is 3/4=0.75.
> OK.
The drawer 1 is open. In it, you see a saltshaker 4.
> OK.
You put the peppershaker 1 in/on the drawer 1.
> critic: Now I put the second creditcard in the dresser. My current state satisfies
all the four sub-goals. The value is 4/4=1.
STATUS: FAIL
Failure feedback: In this environment, my critic assigned a 1/4 value after taking a
saltshaker and a 3/4 value after taking a peppershaker. However, the task is to put two
saltshaker in the drawer. I noticed that one saltshaker was put on the countertop 3. I
didn't find the second saltshaker in this trial. In the next trial, I will first go to
countertop 3 to take the first saltshaker and then explore other locations to find a
second saltshaker, including drawer 2-3, shelf 1-3, and cabinet 1-20.

You are in the middle of a room. Looking quickly around you, you see 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 cool tomato in microwave.
> critic: My task requires three sub-goals in order: take a tomato, cool the tomato
with fridge, and put the tomato in the microwave. My current state satisfies zero of
the three sub-goals. The value is 0/3=0.
> OK.
On the countertop 1, you see a mug 1, and a saltshaker 1.
> OK.
```

```
On the sinkbasin 1, you see a fork 1, a glassbottle 2, a spatula 1, and a spoon 3.
> OK.
You pick up the fork 1 from the sinkbasin 1.
> critic: Now I take a fork. My current state satisfies the first of the six sub-goals:
take a fork. The value is 1/3=0.333.
> OK.
On the countertop 2, you see a butterknife 2, a butterknife 1, a cellphone 1, a
glassbottle 1, a knife 1, a lettuce 3, a peppershaker 1, a statue 2, a statue 1, and a
vase 1.
> OK.
On the countertop 3, you see a bread 1, a butterknife 3, a creditcard 2, a houseplant
1, a knife 3, a knife 2, a mug 2, a peppershaker 2, and a spatula 2.
> OK.
On the stoveburner 4, you see a pan 1.
> OK.
The drawer 3 is closed.
> OK.
You open the drawer 3. The drawer 3 is open. In it, you see a saltshaker 3.
> OK.
The fridge 1 is closed.
> OK.
On the countertop 3, you see a bread 1, a butterknife 3, a creditcard 2, a houseplant
1, a knife 3, a knife 2, a mug 2, a peppershaker 2, and a spatula 2.
> OK.
On the cabinet 1, you see a dishsponge 1.
> OK.
The cabinet 2 is closed.
> OK.
You open the cabinet 2. The cabinet 2 is open. In it, you see nothing.
> OK.
On the cabinet 3, you see a cd 1 and a plate 4.
STATUS: FAIL
Failure feedback: In this environment, my critic assigned a 1/3 value after taking a
fork. However, the task is to take and cool a tomato. I didn't find the tomato in this
trial. In the next trial, I will explore other locations to find a tomato, including
cabinet 4, cabinet 5, coffeemachine 1, microwave 1, shelf 1-3, stoveburner 1-4 and
toaster 1, etc.
```

Model LLM and Elite LLM. We use the same prompt for both the `Model` LLM and the `Elite` LLM. Specifically, we simply prompt these two instances with the successful past trajectories (observations and actions) for each task type and expect the `Model` LLM and `Elite` LLM to generate the possible next observation and the potential actions, respectively. Below, we provide two prompt examples in the put and cool tasks, respectively.

Model & Elite prompt, "put" task

```
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.
```

Model & Elite prompt, "cool" task

```
You are in the middle of a room. Looking quickly around you, you see a cabinet 16, a
cabinet 15, a cabinet 14, a cabinet 13, a cabinet 12, a cabinet 11, a cabinet 10, a
cabinet 9, a cabinet 8, a cabinet 7, a cabinet 6, a cabinet 5, a cabinet 4, a cabinet
3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 2, a countertop 1, a
diningtable 1, a drawer 5, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a fridge 1,
a garbagecan 1, a microwave 1, a safe 1, a sinkbasin 1, a stoveburner 4, a stoveburner
3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: cool some pan and put it in stoveburner.
> go to stoveburner 1
On the stoveburner 1, you see nothing.
> go to stoveburner 2
On the stoveburner 2, you see a pot 1.
> go to stoveburner 3
On the stoveburner 3, you see a pan 1.
> take pan 1 from stoveburner 3
You pick up the pan 1 from the stoveburner 3.
> go to fridge 1
The fridge 1 is closed.
> cool pan 1 with fridge 1
You cool the pan 1 using the fridge 1.
> go to stoveburner 1
On the stoveburner 1, you see nothing.
```

```
> put pan 1 in/on stoveburner 1
You put the pan 1 in/on the stoveburner 1.
```

H.3. Blocksworld

Critic LLM. We evaluate RAFA and RAP with the reward scheme proposed by (Hao et al., 2023). We prompt the language model with the previous state-action trajectory and calculate the log probabilities of taking each feasible action. Given the action taken in the current state, the `Model` LLM predicts the next state and we calculate the percentage of subgoals completed in the next state. We adopt the prompt examples from (Hao et al., 2023) to ensure the fairness in comparison.

Critic prompt example (for log probability), "step-4" task

```
I am playing with a set of blocks where I need to arrange the blocks into stacks. Here
are the actions I can do

Pick up a block
Unstack a block from on top of another block
Put down a block
Stack a block on top of another block

I have the following restrictions on my actions:
I can only pick up or unstack one block at a time.
I can only pick up or unstack a block if my hand is empty.
I can only pick up a block if the block is on the table and the block is clear. A block
is clear if the block has no other blocks on top of it and if the block is not picked
up.
I can only unstack a block from on top of another block if the block I am unstacking
was really on top of the other block.
I can only unstack a block from on top of another block if the block I am unstacking is
clear.
Once I pick up or unstack a block, I am holding the block.
I can only put down a block that I am holding.
I can only stack a block on top of another block if I am holding the block being
stacked.
I can only stack a block on top of another block if the block onto which I am stacking
the block is clear.
Once I put down or stack a block, my hand becomes empty.

[STATEMENT]
As initial conditions I have that, the red block is clear, the yellow block is clear,
the hand is empty, the red block is on top of the blue block, the yellow block is on
top of the orange block, the blue block is on the table and the orange block is on the
table.
My goal is to have that the orange block is on top of the red block.
```

```
My plan is as follows:

[PLAN]
unstack the yellow block from on top of the orange block
put down the yellow block
pick up the orange block
stack the orange block on top of the red block
[PLAN END]

[STATEMENT]
As initial conditions I have that, the orange block is clear, the yellow block is
clear, the hand is empty, the blue block is on top of the red block, the orange block
is on top of the blue block, the red block is on the table and the yellow block is on
the table.
My goal is to have that the blue block is on top of the red block and the yellow block
is on top of the orange block.

My plan is as follows:

[PLAN]
pick up the yellow block
stack the yellow block on top of the orange block
[PLAN END]

[STATEMENT]
As initial conditions I have that, the red block is clear, the blue block is clear, the
orange block is clear, the hand is empty, the blue block is on top of the yellow block,
the red block is on the table, the orange block is on the table and the yellow block is
on the table.
My goal is to have that the blue block is on top of the orange block and the yellow
block is on top of the red block.

My plan is as follows:

[PLAN]
unstack the blue block from on top of the yellow block
stack the blue block on top of the orange block
pick up the yellow block
stack the yellow block on top of the red block
[PLAN END]

[STATEMENT]
```

```
As initial conditions I have that, the red block is clear, the blue block is clear, the
yellow block is clear, the hand is empty, the yellow block is on top of the orange
block, the red block is on the table, the blue block is on the table and the orange
block is on the table.
My goal is to have that the orange block is on top of the blue block and the yellow
block is on top of the red block.

My plan is as follows:

[PLAN]
unstack the yellow block from on top of the orange block
stack the yellow block on top of the red block
pick up the orange block
stack the orange block on top of the blue block
[PLAN END]
```

Model LLM. we prompt the `Model` LLM with few-shot examples and the current state and action. The `Model` LLM generates the predicted next state description. We adopt the prompt examples from (Hao et al., 2023) to ensure the fairness in comparison.

`Model prompt template, "Pick up" action`

```
I am playing with a set of blocks where I need to arrange the blocks into stacks. Here
are the actions I can do

Pick up a block
Unstack a block from on top of another block
Put down a block
Stack a block on top of another block

I have the following restrictions on my actions:
I can only pick up or unstack one block at a time.
I can only pick up or unstack a block if my hand is empty.
I can only pick up a block if the block is on the table and the block is clear. A block
is clear if the block has no other blocks on top of it and if the block is not picked
up.
I can only unstack a block from on top of another block if the block I am unstacking
was really on top of the other block.
I can only unstack a block from on top of another block if the block I am unstacking is
clear. Once I pick up or unstack a block, I am holding the block.
I can only put down a block that I am holding.
I can only stack a block on top of another block if I am holding the block being
stacked.
I can only stack a block on top of another block if the block onto which I am stacking
the block is clear. Once I put down or stack a block, my hand becomes empty.
```

```
After being given an initial state and an action, give the new state after performing
the action.

[SCENARIO 1]
[STATE 0] I have that, the white block is clear, the cyan block is clear, the brown
block is clear, the hand is empty, the white block is on top of the purple block, the
purple block is on the table, the cyan block is on the table and the brown block is on
the table.
[ACTION] Pick up the brown block.
[CHANGE] The hand was empty and is now holding the brown block, the brown block was on
the table and is now in the hand, and the brown block is no longer clear.
[STATE 1] I have that, the white block is clear, the cyan block is clear, the brown
block is in the hand, the hand is holding the brown block, the white block is on top of
the purple block, the purple block is on the table and the cyan block is on the table.

[SCENARIO 2]
[STATE 0] I have that, the purple block is clear, the cyan block is clear, the white
block is clear, the hand is empty, the white block is on top of the brown block, the
purple block is on the table, the cyan block is on the table and the brown block is on
the table.
[ACTION] Pick up the cyan block.
[CHANGE] The hand was empty and is now holding the cyan block, the cyan block was on
the table and is now in the hand, and the cyan block is no longer clear.
[STATE 1] I have that, the cyan block is in the hand, the white block is clear, the
purple block is clear, the hand is holding the cyan block, the white block is on top of
the brown block, the purple block is on the table and the brown block is on the table.
```

Model prompt template, "Unstack" action

```
I am playing with a set of blocks where I need to arrange the blocks into stacks. Here
are the actions I can do

Pick up a block
Unstack a block from on top of another block
Put down a block
Stack a block on top of another block

I have the following restrictions on my actions:
I can only pick up or unstack one block at a time.
I can only pick up or unstack a block if my hand is empty.
I can only pick up a block if the block is on the table and the block is clear. A block
is clear if the block has no other blocks on top of it and if the block is not picked
up.
```

```
I can only unstack a block from on top of another block if the block I am unstacking
was really on top of the other block.
I can only unstack a block from on top of another block if the block I am unstacking is
clear. Once I pick up or unstack a block, I am holding the block.
I can only put down a block that I am holding.
I can only stack a block on top of another block if I am holding the block being
stacked.
I can only stack a block on top of another block if the block onto which I am stacking
the block is clear. Once I put down or stack a block, my hand becomes empty.

After being given an initial state and an action, give the new state after performing
the action.

[SCENARIO 1]
[STATE 0] I have that, the white block is clear, the cyan block is clear, the brown
block is clear, the hand is empty, the white block is on top of the purple block, the
purple block is on the table, the cyan block is on the table and the brown block is on
the table.
[ACTION] Unstack the white block from on top of the purple block.
[CHANGE] The hand was empty and is now holding the white block, the white block was on
top of the purple block and is now in the hand, the white block is no longer clear, and
the purple block is now clear.
[STATE 1] I have that, the purple block is clear, the cyan block is clear, the brown
block is clear, the hand is holding the white block, the white block is in the hand,
the purple block is on the table, the cyan block is on the table and the brown block is
on the table.

[SCENARIO 2]
[STATE 0] I have that, the purple block is clear, the cyan block is clear, the white
block is clear, the hand is empty, the cyan block is on top of the brown block, the
purple block is on the table, the white block is on the table and the brown block is on
the table.
[ACTION] Unstack the cyan block from on top of the brown block.
[CHANGE] The hand was empty and is now holding the cyan block, the cyan block was on
top of the brown block and is now in the hand, the cyan block is no longer clear, and
the brown block is now clear.
[STATE 1] I have that, the purple block is clear, the brown block is clear, the cyan
block is in the hand, the white block is clear, the hand is holding the cyan block, the
purple block is on the table, the white block is on the table and the brown block is on
the table.
```

Model prompt template, "Put down" action

```
I am playing with a set of blocks where I need to arrange the blocks into stacks. Here
are the actions I can do

Pick up a block
Unstack a block from on top of another block
Put down a block
Stack a block on top of another block

I have the following restrictions on my actions:
I can only pick up or unstack one block at a time.
I can only pick up or unstack a block if my hand is empty.
I can only pick up a block if the block is on the table and the block is clear. A block
is clear if the block has no other blocks on top of it and if the block is not picked
up.
I can only unstack a block from on top of another block if the block I am unstacking
was really on top of the other block.
I can only unstack a block from on top of another block if the block I am unstacking is
clear. Once I pick up or unstack a block, I am holding the block.
I can only put down a block that I am holding.
I can only stack a block on top of another block if I am holding the block being
stacked.
I can only stack a block on top of another block if the block onto which I am stacking
the block is clear. Once I put down or stack a block, my hand becomes empty.

After being given an initial state and an action, give the new state after performing
the action.

[SCENARIO 1]
[STATE 0] I have that, the white block is clear, the purple block is clear, the cyan
block is in the hand, the brown block is clear, the hand is holding the cyan block, the
white block is on the table, the purple block is on the table, and the brown block is
on the table.
[ACTION] Put down the cyan block.
[CHANGE] The hand was holding the cyan block and is now empty, the cyan block was in
the hand and is now on the table, and the cyan block is now clear.
[STATE 1] I have that, the cyan block is clear, the purple block is clear, the white
block is clear, the brown block is clear, the hand is empty, the white block is on the
table, the purple block is on the table, the cyan block is on the table, and the brown
block is on the table.

[SCENARIO 2]
```

```
[STATE 0] I have that, the purple block is clear, the black block is in the hand, the
white block is clear, the hand is holding the black block, the white block is on top of
the brown block, the purple block is on the table, and the brown block is on the table.
[ACTION] Put down the black block.
[CHANGE] The hand was holding the black block and is now empty, the black block was in
the hand and is now on the table, and the black block is now clear.
[STATE 1] I have that, the black block is clear, the purple block is clear, the white
block is clear, the hand is empty, the white block is on top of the brown block, the
purple block is on the table, the brown block is on the table, and the black block is
on the table.
```

Model prompt template, "Stack" action

```
I am playing with a set of blocks where I need to arrange the blocks into stacks. Here
are the actions I can do

Pick up a block
Unstack a block from on top of another block
Put down a block
Stack a block on top of another block

I have the following restrictions on my actions:
I can only pick up or unstack one block at a time.
I can only pick up or unstack a block if my hand is empty.
I can only pick up a block if the block is on the table and the block is clear. A block
is clear if the block has no other blocks on top of it and if the block is not picked
up.
I can only unstack a block from on top of another block if the block I am unstacking
was really on top of the other block.
I can only unstack a block from on top of another block if the block I am unstacking is
clear. Once I pick up or unstack a block, I am holding the block.
I can only put down a block that I am holding.
I can only stack a block on top of another block if I am holding the block being
stacked.
I can only stack a block on top of another block if the block onto which I am stacking
the block is clear. Once I put down or stack a block, my hand becomes empty.

After being given an initial state and an action, give the new state after performing
the action.

[SCENARIO 1]
```

[STATE 0] I have that, the white block is clear, the purple block is clear, the cyan block is in the hand, the brown block is clear, the hand is holding the cyan block, the white block is on the table, the purple block is on the table, and the brown block is on the table.
[ACTION] Stack the cyan block on top of the brown block.
[CHANGE] The hand was holding the cyan block and is now empty, the cyan block was in the hand and is now on top of the brown block, the brown block is no longer clear, and the cyan block is now clear.
[STATE 1] I have that, the cyan block is clear, the purple block is clear, the white block is clear, the hand is empty, the cyan block is on top of the brown block, the brown block is on the table, the purple block is on the table, and the white block is on the table.

[SCENARIO 2]
[STATE 0] I have that, the purple block is clear, the black block is in the hand, the white block is clear, the hand is holding the black block, the white block is on top of the brown block, the purple block is on the table, and the brown block is on the table.
[ACTION] Stack the black block on top of the purple block.
[CHANGE] The hand was holding the black block and is now empty, the black block was in the hand and is now on top of the purple block, the purple block is no longer clear, and the black block is now clear.
[STATE 1] I have that, the black block is clear, the white block is clear, the hand is empty, the black block is on top of the purple block, the white block is on top of the brown block, the brown block is on the table, and the purple block is on the table.

H.4. Tic-Tac-Toe

Elite LLM

Elite prompt, propose n actions

```
In the game of Tic-Tac-Toe, two players, "X" and "O," alternate placing their symbols
on a 3x3 grid. The objective is to be the first to get three of their symbols in a row,
either horizontally, vertically, or diagonally. We use numbers to indicate empty
positions, and then replace them with "X" or "O" as moves are made. For example, an
empty board is denoted by

1 | 2 | 3
---------
4 | 5 | 6
---------
7 | 8 | 9

Your task is to identify the optimal position for the next move based on the current
board state. Assume that it's your turn and you're playing as "{role}". Please make
sure the optimal position is EMPTY. For example, in the following Tic-Tac-Toe Board:

1 | 2 | 3
---------
4 | X | 6
---------
7 | 8 | 9

Position 5 is occupied by "X". Thus, position 5 is not an optimal position. Provide
only the optimal position in the first line. In the second line, give a brief
explanation for this choice.

Current Tic-Tac-Toe Board:

{state}

Role: {role}

Optimal Position:
```

Model LLM

Model prompt, predict next state

```
Predict the Next State of the Tic-Tac-Toe Board
```

```
In a game of Tic-Tac-Toe, two players, "X" and "O," take turns to place their symbols
on a 3x3 grid. Your task is to predict what the board will look like after a specified
move has been made.

Examples
{examples}
Now, Predict the Next State of the Following Tic-Tac-Toe Board:
Initial Tic-Tac-Toe Board:

{state}

Move: Player puts "{role}" in position {action}.

Updated Board:
```

Model prompt, predict opponent's action

```
In Tic-Tac-Toe, each player takes turns placing their respective symbols ("X" or "O")
on a 3x3 board. Your task is to predict where the opponent will place their symbol
based on their past moves and the current board state.

Example
Tic-Tac-Toe Board:

O | X | O
---------
X | O | X
---------
7 | 8 | X

Opponent's Move: "O" in position 7
{examples}
Here's how the Tic-Tac-Toe board currently looks:
Tic-Tac-Toe Board:

{state}

Given the history and current board state, where do you think the opponent will place
their "{role}" next? Please make sure the output is an empty position without "X" or
"O".

Opponent's Move: "{role}" in position
```

Critic LLM

Critic prompt, evaluate winner

```
Determine the Winner in a Tic-Tac-Toe Game

In Tic-Tac-Toe, two players, "X" and "O" take turns to place their respective symbols
on a 3x3 board. The first player to get three of their symbols in a row, either
horizontally, vertically, or diagonally, wins the game. Your task is to evaluate the
board state and determine if there is a winner.

Examples

Example
Tic-Tac-Toe Board:

O | X | O
---------
X | X | X
---------
O | O | X

Question: Is there a winner?

Answer: Let's think step by step.

First row: O X O, no winner
Second row: X X X, X wins

Therefore, "X" wins

Example
Tic-Tac-Toe Board:

X | 2 | O
---------
4 | O | X
---------
O | X | 9

Question: Is there a winner?

Answer: Let's think step by step.

First row: X 2 O, no winner
```

```
Second row: 4 O X, no winner
Third row: O X 9, no winner
First column: X 4 O, no winner
Second column: 2 O X, no winner
Thrid column: O X 9, no winner
Main diagonal: X O 9, no winner
Anti-diagonal: O O O, O wins

Therefore, "O" wins.
{examples}
Now, for the Current Tic-Tac-Toe Board:
Tic-Tac-Toe Board:

{state}

Question: Is there a winner?

Answer: Let's think step by step.
```

Critic prompt, evaluate tie (when there is no winner)

```
In the game of Tic-Tac-Toe, two players alternate turns to fill a 3x3 grid with their
respective symbols: "X" and "O". A board is considered "completely filled" when all
nine cells of the grid contain either an 'X' or an 'O', with no empty spaces or other
characters.

Examples:
{examples}
Now for the Current Tic-Tac-Toe Board:
Tic-Tac-Toe Board:

{state}

Is the board completely filled?

Answer:
```