RESOURCE-AWARE REASONING ORCHESTRATION FOR DECISION-MAKING IN EMBODIED AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Embodied agents built on multimodal large language models (MLLMs) increasingly rely on explicit reasoning, such as replanning and verification, to recover from execution failures and resolve ambiguity. However, invoking such reasoning uniformly is wasteful: many decisions in a household task are unambiguous and do not benefit from additional deliberation, while others critically require it. Existing systems address this trade-off with fixed strategies, always reasoning, never reasoning, or reasoning on a static schedule, none of which adapt to the evolving execution state. We introduce **RARRL** (Resource-Aware Reasoning via Reinforcement Learning), an orchestration framework that learns *when* to invoke reasoning rather than prescribing it a priori. A lightweight policy, trained once with reinforcement learning in an Abstract Decision Environment (ADE), observes a compact state summarizing task progress, execution reliability, and remaining budget, and decides at each timestep whether to act directly or invoke a Planner-Verifier reasoning procedure before acting. Deploying this policy on a new simulated platform requires an environment-specific grounding adapter that exposes this shared state; the policy transfers zero-shot without retraining. We evaluate RARRL on EB-ALFRED and EB-Habitat, two embodied benchmarks built on distinct simulators with non-overlapping action spaces, and find that it consistently outperforms never-reasoning, always-reasoning, and fixed-schedule reasoning baselines across six task categories, while invoking reasoning substantially less often than always-reasoning agents. We further evaluate RARRL on VirtualHome and ALFWorld, achieving the best task success rate. We further perform matched-state counterfactual interventions in the target environments, showing that the value of invoking reasoning varies across execution-time decision states.

1 INTRODUCTION

Recent advances in Multimodal Large Language Models (MLLMs) have accelerated the development of embodied agents (Brown et al., 2020; Wei et al., 2022; Ahn et al., 2022; Zitkovich et al., 2023; Driess et al., 2023). By combining visual perception, language understanding, and reasoning, MLLM-based agents can execute long-horizon tasks from natural language instructions (Liu et al., 2024; Shridhar et al., 2020; 2022).

However, reasoning is not equally valuable throughout task execution. During routine interactions (e.g., navigating along a clear path or executing a verified plan), additional deliberation may provide little benefit while incurring substantial latency and computational cost. In contrast, when execution becomes uncertain, such as after repeated failures, task stagnation, or ambiguous observations, additional reasoning can be critical for successful recovery. This raises a fundamental question: *When should an embodied agent invest additional reasoning?*

Existing embodied reasoning systems typically adopt one of several reasoning invocation strategies. Some rely solely on the standard planner without additional deliberation (Never-Think) (Yang et al., 2025), which can struggle to recover from execution failures or high-risk states. Others invoke reasoning before every decision (Always-Think) (Zawalski et al., 2024), providing stronger deliberation at the cost of excessive latency and token consumption. A third class employs fixed reasoning schedules, such as invoking reasoning every k steps (Fixed-Think) (Erdogan et al., 2025),

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

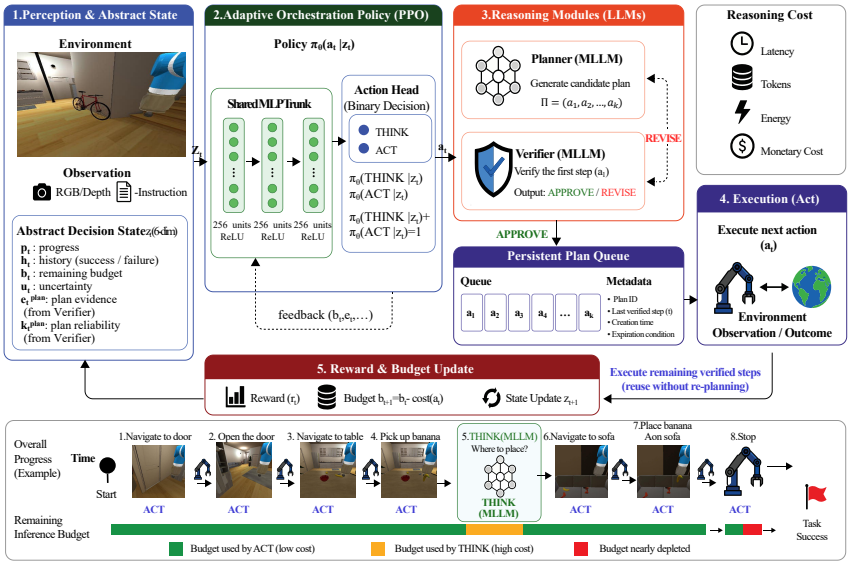

Figure 1: Overview of RARRL. The policy observes a 6-dimensional abstract state z_t and outputs a binary action (THINK or ACT). If THINK is selected, the Planner (MLLM) generates a candidate plan, and the Verifier (MLLM) checks its first action, producing an APPROVE/REJECT signal. Approved plans are added to a persistent plan queue and executed by the agent. The reward, reasoning budget, and decision state are then updated based on the execution outcome.

while another uses handcrafted uncertainty thresholds to trigger reasoning (Heuristic-Think) (Ren et al., 2023). However, these strategies do not adapt reasoning allocation to the evolving execution state and remaining computational budget.

To address this challenge, we propose RARRL. A lightweight orchestration policy, trained once in an abstract decision environment (ADE), dynamically decides whether to invoke an enhanced reasoning module based on the current execution state, remaining computational budget, and task progress. The learned policy is transferred without retraining to the two evaluated simulated embodied benchmarks, EB-ALFRED and EB-Habitat, enabling zero-shot policy transfer across the evaluated simulators. Extensive experiments on EmbodiedBench demonstrate that RARRL consistently improves embodied task performance while significantly reducing unnecessary reasoning. We evaluate RARRL under zero-shot transfer from ADE to both EB-ALFRED and EB-Habitat (Yang et al., 2025), showing that **resource-aware reasoning allocation** generalizes across multiple embodied environments. Further analyses reveal that RARRL successfully avoids both under-thinking and over-thinking by selectively invoking deliberation only when its expected benefit outweighs its computational cost. Our main contributions are summarized as follows:

(1) **Resource-Aware Reasoning Orchestration.** We formulate reasoning allocation as a sequential resource-constrained decision-making problem and learn a lightweight orchestration policy that dynamically determines *when* additional reasoning should be invoked and *when* previously verified plans should be reused under a limited reasoning budget, replacing handcrafted reasoning schedules with resource-aware decision making.

(2) **Resource-aware Persistent Planning and Execution Recovery.** We introduce a resource-aware persistent planning framework that combines verifier-based plan validation with persistent plan reuse, allowing verified plans to be executed efficiently without repeatedly invoking expensive reasoning modules.

(3) **Zero-shot Orchestration Transfer.** We train a reasoning-orchestration policy entirely in an Abstract Decision Environment (ADE) and deploy its frozen parameters to EB-ALFRED and EB-Habitat without policy adaptation, fine-tuning, or additional demonstrations. We further validate the transferred decision rule through matched target-state counterfactual analysis.

(4) **Extensive Evaluation.** Extensive experiments on EB-ALFRED, EB-Habitat, VirtualHome, and ALFWorld demonstrate that RARRL consistently outperforms the official baselines while significantly improving reasoning efficiency.

2 Related Work

Embodied Decision-Making in Robotics. Embodied robotic systems have long been studied as sequential decision-making problems under uncertainty, where agents must perceive, reason, and act through continuous interaction with the environment Thrun (2002).Classical robotics frameworks typically adopt hierarchical architectures that decouple high-level decision-making from low-level perception and control, enabling modularity and interpretability Siciliano et al. (2008). While effective, these approaches often rely on manually designed decision rules or fixed abstractions, limiting their adaptability to complex and dynamic tasks. Learning-based methods, particularly reinforcement learning (RL), have introduced data-driven alternatives for robotic decision-making and hierarchical control Sutton et al. (1998). However, most RL-based systems focus on learning action-selection policies or task-specific controllers and do not explicitly model the computational cost of high-level reasoning or the dynamic allocation of reasoning resources during execution.

Large Models for Embodied Reasoning. Recent advances in large language models (MLLMs) have enabled robots to perform high-level semantic reasoning, task planning, and instruction following (Wei et al., 2022; Brown et al., 2020; Li et al., 2024; Chen et al., 2023). Prior work has explored grounding language models in robotic action spaces, allowing robots to translate natural language instructions into executable behaviors (Ahn et al., 2022; Liang et al., 2023). More recent systems integrate multimodal language models to support embodied reasoning across perception, language, and action Driess et al. (2023); Zitkovich et al. (2023). Despite their success, most existing MLLM-enabled robotic systems invoke high-level reasoning in a static or heuristic manner. These approaches typically assume that reasoning is always beneficial, without accounting for the substantial computational cost and latency of MLLM inference (Chen et al., 2025). As a result, excessive reasoning may delay action execution, while insufficient reasoning can lead to task failure, motivating resource-aware allocation of reasoning over the execution process.

Resource-Aware Reasoning Orchestration. Resource-aware autonomy has been studied in robotics concerning computation, energy, and real-time constraints Mohamed et al. (2021). Existing approaches investigate scheduling, bounded rationality, or task prioritization under limited resources, but often treat reasoning modules as cost-free or rely on static allocation strategies Eiffert et al. (2022); Xu & Tzoumas (2022). SwiftSage (Lin et al., 2023) switches to a slower module upon detecting exceptions; AdaPlanner (Sun et al., 2023) revises plans based on in-/out-of-plan feedback via prompting; Reflexion (Shinn et al., 2023) and Re2 Agent (Chen et al., 2025) accumulate verbal feedback or rules across repeated attempts. RARRL instead formulates reasoning invocation as a sequential resource-allocation problem and learns, via reinforcement learning within a single episode, when to invoke additional reasoning, which reasoning role to use, and how to allocate a limited reasoning budget.

3 The Proposed Method

3.1 Problem Formulation and Overview

We formalize the reasoning-allocation problem introduced above as a sequential decision process. At each timestep t, an embodied agent receives an observation o_t and must produce an action $a_t \in \mathcal{A}$ toward completing instruction I. The agent additionally chooses a mode $m_t \in \{\text{ACT}, \text{THINK}\}$: under ACT, a_t is produced directly by a vision-language-model controller; under THINK, a_t is obtained through a Planner-Verifier reasoning procedure. We define the reasoning budget as the number of available THINK invocations: every invocation of THINK consumes one unit of a finite per-episode budget $b \in \{0, \ldots, b_{\max}\}$, regardless of whether the resulting candidate is subsequently approved, rejected, or corrected, reflecting the computational cost of invoking the Planner-Verifier reasoning pipeline (Sec. 3.3). The objective is to maximize task success within a fixed step limit while invoking THINK selectively rather than uniformly.

We refer to our system as **RARRL** (Resource-Aware Reasoning via Reinforcement Learning). As illustrated in Figure 1, RARRL learns *when* to reason via an orchestration policy trained once in a lightweight Abstract Decision Environment (Sec. 3.2,3.4), while *how* to reason is implemented by a fixed Planner-Verifier procedure (Sec. 3.5). The learned policy transfers to downstream embodied platforms without further adaptation (Sec. 3.6,3.7).

3.2 WHY AN ABSTRACT DECISION ENVIRONMENT?

Training orchestration inside a real simulator with LLM calls faces two fundamental obstacles: *credit assignment* (a THINK action's value manifests only after tens of primitive steps) and *sample cost* (on-policy RL requires orders of magnitude more LLM calls than is tractable). Hand-crafted rules fail because the beneficial-reasoning boundary depends on a high-dimensional combination of evidence signals, failure history, and remaining budget—a space too large to enumerate by rule, and whose optimal partition depends on a hidden decision context the policy must infer from noisy evidence alone. RARRL resolves both by training in an Abstract Decision Environment (ADE) that preserves only orchestration-relevant structure, requiring no real simulator and no real LLM calls during training.

3.3 WHAT STRUCTURE DOES THE ADE PRESERVE?

Design Rationale. The ADE is designed to capture the minimal decision process required for resource-aware reasoning orchestration rather than to reproduce any particular simulator. Its design follows four requirements from the generic orchestration problem: (i) a hidden decision context that the policy cannot observe directly, requiring it to infer whether reasoning is worthwhile from observable evidence; (ii) a context-dependent transition in which the value of THINK depends on the hidden context, preventing degeneration to a fixed invocation schedule; (iii) imperfect evidence with delayed outcomes, so the utility of a reasoning call is revealed only through subsequent execution, creating a sequential credit-assignment problem; and (iv) a hard budget ceiling that forces the policy to weigh the value of information against its cost. The ADE therefore does not aim to reproduce ALFRED or any particular simulator. Its components are derived from the generic structure of reasoning orchestration (Appendix B).

Abstract decision state. The policy observes a 6-dimensional state $z_t = (p_t, h_t, b_t, u_t, e_{\text{plan}}, \kappa_{\text{plan}}) \in \mathcal{Z}$, where p_t is estimated decision-level progress, h_t encodes recent execution outcomes, b_t is normalized remaining budget, u_t is observable uncertainty (failure frequency), and $(e_{\text{plan}}, \kappa_{\text{plan}})$ are the decision evidence and reliability acquired through prior THINK actions. The latent context ϑ_t and its associated decision facets are hidden from the policy. Thus, the policy must decide whether additional reasoning is valuable based only on the observable decision state and accumulated evidence. The latent context determines which decision facets are relevant, while the evidence variables record information acquired through previous THINK invocations (Appendix C).

ADE generative process. At each decision step t, the ADE samples a latent execution context ϑ_t and specifies its task-relevant decision facets. Given the current context and acquired evidence, the orchestration policy selects either ACT or THINK. ACT attempts to advance the current stage according to the transition model below, whereas THINK consumes one unit of reasoning budget and generates evidence about the relevant facets. The resulting evidence updates $(e_t^{\text{plan}}, \kappa_t^{\text{plan}})$, while execution outcomes update p_t, h_t, and u_t. All context types, facet requirements, evidence-generation rules, advance probabilities, and transition dynamics are fixed by the ADE generator and specified in Appendix B.

The correspondence to embodied execution is through decision variables rather than simulator dynamics: p_t, h_t, b_t, u_t, and e_t represent task progress, progress stagnation, remaining reasoning budget, recent execution reliability, and planner-verifier evidence, respectively. Their concrete grounding on EB-ALFRED and EB-Habitat is described in Section 3.6 and Appendix D.

Action space and transition. The orchestration action is $a_t \in \{\texttt{ACT}, \texttt{THINK}\}$. Under ACT, the agent attempts to advance the current stage directly. Under THINK, the agent acquires additional decision evidence through the ADE evidence-generation process, updating $(e_{\text{plan}}, \kappa_{\text{plan}})$. The latent context ϑ determines which decision requirements are relevant at the current decision point. Let F_ϑ denote the set of required decision facets, and let $M_t \subseteq F_\vartheta$ denote the facets for which sufficient evidence has been acquired. If ϑ requires no facet, p_{adv} takes a fixed high value; otherwise,

$$p_{\text{adv}} = \prod_{j \in F_\vartheta} \begin{cases} \rho_j, & j \in M_t, \\ p_{\text{low}}, & j \notin M_t, \end{cases} \tag{1}$$

where ρ_j is the reliability associated with an addressed facet and p_{low} is the fixed low-reliability value for an unresolved facet. A THINK call acquires evidence toward one facet through the fixed ADE evidence-generation process, with a small probability of targeting a different facet, thereby updating M_t and the resulting $(e_{\text{plan}}, \kappa_{\text{plan}})$. The latent context is resampled from the same fixed prior whenever the agent advances to a new stage. Recent execution outcomes enter the observable decision state through h_t and u_t, rather than directly through p_{adv}. The full parameterization, including the context prior, reliability function, and evidence-acquisition noise model, is given in Appendix B. The per-step reward is

$$r_t = \mathbb{1}[\text{success}] - \lambda_{\text{call}}\,\mathbb{1}[\text{THINK}] - \mu\,\mathbb{1}[\text{fail}], \tag{2}$$

where $\mathbb{1}[\cdot]$ is the indicator function, and $\lambda_{\text{call}}, \mu > 0$ are swept in sensitivity experiments (full derivation in Appendix C; parameters in Appendix Table 5).

3.4 What Does PPO Learn?

The orchestration policy $\pi_\theta(a \mid z)$ is implemented as a multi-layer perceptron with three hidden layers of 256 units and ReLU activations, with a binary THINK/ACT output head. A separate value network of identical architecture is used for advantage estimation. PPO (Schulman et al., 2017) trains π_θ on ADE rollouts. The policy learns context-appropriate evidence acquisition: acting directly when the current evidence is sufficient and invoking reasoning when additional evidence is valuable, all from reward alone, without ever observing the latent context ϑ. The resulting policy is frozen and conditioned only on the observable decision state, including progress, execution history, remaining budget, uncertainty, and accumulated decision evidence. In short, the policy learns *when* additional decision evidence is worth acquiring under a limited reasoning budget, rather than which internal reasoning procedure to invoke.

3.5 How Does Think Reason?

When the policy selects THINK, control is passed to a fixed reasoning pipeline consisting of a *Planner* followed by a *Verifier*. The Planner proposes a candidate multi-step plan, while the Verifier reviews its first action before any action is submitted to the environment. The resulting verification judgment provides decision evidence and a reliability signal to the orchestration policy.

Planner. The Planner receives the current egocentric observation, task instruction, recent interaction history, and a small number of in-context examples. It generates a multi-step candidate plan, whose first action is immediately verified while the remaining actions are retained for subsequent execution.

Verifier. The Verifier receives the first candidate action and the current observation. It performs two checks: a *relevance check*, which assesses whether the action is relevant to the task, and a *visual drift check*, which compares selected state variables derived from the current observation with those tracked in the interaction history. It returns a binary judgment, APPROVE or REJECT, together with a risk level. A candidate is rejected if the estimated risk is high or either check identifies an inconsistency. The resulting judgment is summarized as decision evidence and reliability $(e_{\text{plan}}, \kappa_{\text{plan}})$, which is incorporated into the orchestration state z_t (Sec. 3.3).

Persistent Plan Queue. If the first action is approved, the remaining steps of the proposed plan are inserted into a Persistent Plan Queue. The queued actions are then executed sequentially without re-invoking the Planner or Verifier for each action. If a queued action fails or progress stalls for several consecutive steps, the queue is discarded and control returns to the orchestration policy for a fresh decision. This design amortizes the cost of a THINK invocation across multiple actions while allowing fresh reasoning when the current plan becomes unreliable.

3.6 Why Does the Policy Transfer Zero-Shot?

The grounding map $f : \mathcal{S} \to \mathcal{Z}$ abstracts the embodied state to the decision state z and is environment-specific: each platform requires its own implementation of f (Sec. 3.5, Appendix D). Across EB-ALFRED and EB-Habitat, the abstract features are computed using the same definitions: p_t is the milestone completion ratio, h_t captures progress stagnation from recent p_t history, and u_t measures recent action success or failure. The evidence variables e_t^{plan} and κ_t^{plan} are derived from the shared Planner–Verifier pipeline and are simulator-independent. Only the underlying simulator

feedback fields used to compute these features differ across platforms. Thus, platform differences affect how raw execution feedback is mapped to the shared abstract state z, rather than the policy operating on z itself.

What transfers zero-shot, without retraining or target-task optimization, is the orchestration policy π_θ itself. The policy operates only on the shared abstract state z and is never re-optimized for a new platform. The Q^*-irrelevance condition (Li et al., 2006) provides a sufficient abstraction-level condition under which such transfer can be expected: if f satisfies this property, the optimal decisions over $\mathcal{Z}$ are preserved across the corresponding environments, and the behavioral-distance bound (Ferns et al., 2004) implies that residual sub-optimality is controlled by the behavioral discrepancy induced by the abstraction. The formal treatment is provided in Appendix E.

3.7 Deployment Transition

At deployment, π_θ is frozen. The grounding map f extracts z_t from runtime signals (execution outcomes, budget, LLM evidence); the policy selects ACT or THINK. A THINK decision may update the Current Subgoal Buffer, whose contents are subsequently used by the *Task Executor* for primitive-action execution. The Task Executor is a standard hierarchical-control component (Brooks, 1986), not part of the learned policy. The policy is never retrained; only f (Appendix D) and the executor are environment-specific.

4 Experiments

Environments and datasets. We evaluate our approach on four complementary embodied environments: EB-ALFRED (Yang et al., 2025), EB-Habitat (Yang et al., 2025), ALFWorld (Shridhar et al., 2020), and VirtualHome (Puig et al., 2018). EB-ALFRED and EB-Habitat are built on the AI2-THOR (Kolve et al., 2017) and Habitat (Szot et al., 2021) simulators, respectively. EB-ALFRED focuses on long-horizon household manipulation, whereas EB-Habitat focuses on navigation and object rearrangement, providing complementary embodied task domains. ALFWorld provides a text-based household task environment, while VirtualHome provides household activity scenarios with varying task and scene distributions. For EB-ALFRED and EB-Habitat, each benchmark contains 300 test instances, evenly distributed across six capability-oriented subsets: Base, Commonsense Reasoning, Complex Instruction, Visual Appearance, Spatial Relationship, and Long Horizon, with 50 instances per subset. This capability-oriented evaluation enables fine-grained analysis of resource-aware reasoning across different embodied capabilities. We additionally evaluate on ALFWorld and VirtualHome to assess their behavior under different task interfaces and task distributions in Appendix G. The EB-ALFRED and EB-Habitat experiments use Qwen2.5-VL-72B-Instruct (Wu et al., 2025) as the primary MLLM backend for both the Planner and Verifier, with Gemma-3-27B-it (Team et al., 2025) used for reasoning-backend robustness evaluation. ALFWorld uses Llama-3.1-8B-Instruct (Grattafiori et al., 2024) and VirtualHome uses the same Qwen2.5-VL-72B-Instruct as its evaluation backend. Implementation details are provided in Appendix A.

Baselines. We compare RARRL against four reasoning invocation baselines.

(1) **EmbodiedBench** (Yang et al., 2025): the agent never invokes the THINK module and directly predicts the next action (Never-THINK). (2) **ECoT** (Zawalski et al., 2024): the agent invokes the THINK module before every action (Always-THINK). (3) **Plan-and-Act** (Erdogan et al., 2025): the agent invokes THINK periodically every k environment steps ($k = 3$), representing a fixed-interval strategy (Fixed-Think). (4) **KnowNo** (Ren et al., 2023): the agent invokes THINK only when a predefined uncertainty measure exceeds a fixed threshold (Heuristic-Think). We instantiate uncertainty as the number of consecutive steps without task progress, using the same agent-observable milestone-based signal as RARRL. All compared reasoning-enabled methods use the same Planner, Verifier, Persistent Plan Queue, MLLM backend, and execution pipeline, differing only in when THINK is invoked.

Evaluation Metrics. We adopt two evaluation metrics: Task Success Rate (TSR) and Pending Steps (PS). TSR denotes the ratio of successfully completed tasks. PS stands for the average timestep taken to complete tasks.

ADE Training. We train the orchestration policy exclusively through online interaction with the procedurally generated ADE. Each episode independently samples a hidden decision context and

Table 1: Capability-wise comparison across EB-ALFRED and EB-Habitat. TSR↑ denotes task success rate, while PS↓ denotes the average number of timesteps taken to complete the task. Results are reported as mean ± 95% CI over 15 random seeds.

	Base		Common		Complex		Visual		Spatial		Long		Avg. ↑
Method	TSR↑	PS↓	TSR↑	PS↓	TSR↑	PS↓	TSR↑	PS↓	TSR↑	PS↓	TSR↑	PS↓	
EB-ALFRED													
EmbodiedBench	50.1±2.1	15.52	42.7±2.0	18.68	42.4±2.1	16.18	36.2±2.0	16.96	34.5±1.9	17.74	34.1±2.2	24.12	40.00
ECoT	60.4±2.2	15.72	52.8±2.1	15.94	48.3±2.2	16.46	44.1±2.1	16.08	48.4±2.2	17.18	42.2±2.0	26.34	49.37
Plan-and-Act	56.6±2.1	16.68	50.4±2.1	17.10	48.7±2.2	17.10	40.1±2.0	17.40	44.3±2.1	18.56	46.5±2.1	23.78	47.77
KnowNo	55.9±2.1	15.26	51.2±2.1	16.32	52.5±2.2	15.52	42.1±2.0	17.04	42.6±2.0	18.16	54.4±2.2	24.28	49.78
RARRL(Ours)	**62.8±2.2**	**14.58**	**54.5±2.2**	**14.02**	**56.2±2.2**	**15.32**	**48.6±2.1**	**15.68**	**52.4±2.2**	**16.44**	**61.9±2.2**	**23.07**	**55.97**
EB-Habitat													
EmbodiedBench	70.3±1.7	7.32	30.7±1.9	17.72	36.3±2.0	15.02	32.8±1.9	15.07	28.5±1.8	24.60	18.2±1.6	18.00	36.13
ECoT	78.4±1.9	8.58	41.2±2.0	17.63	48.8±2.1	13.46	44.5±2.0	16.86	33.6±1.9	23.44	42.3±2.0	21.00	48.13
Plan-and-Act	80.4±1.8	7.60	43.1±2.0	16.56	44.6±2.0	16.36	46.5±2.1	15.50	34.2±1.9	20.48	46.5±2.1	19.92	49.22
KnowNo	79.2±1.8	8.20	41.4±2.0	16.32	51.2±2.1	15.52	44.6±2.0	17.04	32.1±1.9	18.16	44.7±2.0	24.28	48.87
RARRL (Ours)	**86.6±1.7**	**6.96**	**48.2±2.1**	**15.58**	**64.5±2.1**	**12.12**	**60.3±2.1**	**15.02**	**36.7±1.9**	**17.76**	**52.4±2.1**	**17.40**	**58.12**

Table 2: Ablation study of RARRL across the six capability categories. Each ablation removes one component while keeping the remaining mechanisms unchanged. TSR↑ denotes task success rate. Results are reported as mean ± 95% CI over 15 random seeds. Δ reports the paired TSR difference relative to RARRL, with 95% paired bootstrap CIs computed over matched seeds.

	Base	Common	Complex	Visual	Spatial	Long	Avg.	Paired Δ vs. RARRL
EB-ALFRED								
w/o Persistent Planning	57.4 ± 2.1	50.8 ± 2.5	51.3 ± 2.3	43.2 ± 2.8	47.6 ± 2.4	57.9 ± 2.1	51.4 ± 1.9	-4.8 $[-6.8, -2.2]$
w/o Adaptive Orch	55.8 ± 2.2	49.1 ± 2.4	50.2 ± 2.5	41.7 ± 2.7	45.8 ± 2.6	55.6 ± 2.3	49.7 ± 2.0	-6.3 $[-8.7, -3.8]$
w/o Resource Awareness	59.1 ± 2.0	52.0 ± 2.3	53.4 ± 2.2	45.0 ± 2.5	49.8 ± 2.3	58.2 ± 2.0	52.9 ± 1.8	-3.2 $[-5.2, -0.8]$
RARRL	$\mathbf{62.8 \pm 2.2}$	$\mathbf{54.5 \pm 2.2}$	$\mathbf{56.2 \pm 2.2}$	$\mathbf{48.6 \pm 2.1}$	$\mathbf{52.4 \pm 2.2}$	$\mathbf{61.9 \pm 2.2}$	$\mathbf{55.9 \pm 1.7}$	–
EB-Habitat								
w/o Persistent Planning	82.1 ± 2.0	43.6 ± 2.8	57.8 ± 2.4	53.4 ± 2.6	34.9 ± 2.7	48.7 ± 2.5	53.4 ± 2.1	-4.7 $[-7.1, -2.3]$
w/o Adaptive Orch	80.5 ± 2.1	40.8 ± 2.7	54.1 ± 2.5	50.6 ± 2.7	32.7 ± 2.8	46.2 ± 2.6	50.8 ± 2.2	-7.3 $[-9.8, -4.9]$
w/o Resource Awareness	84.0 ± 1.9	45.7 ± 2.6	59.6 ± 2.3	55.1 ± 2.5	35.8 ± 2.6	50.1 ± 2.4	55.1 ± 2.0	-3.2 $[-5.1, -0.9]$
RARRL	$\mathbf{86.6 \pm 1.7}$	$\mathbf{48.2 \pm 2.1}$	$\mathbf{64.5 \pm 2.1}$	$\mathbf{60.3 \pm 2.1}$	$\mathbf{36.7 \pm 1.9}$	$\mathbf{52.4 \pm 2.1}$	$\mathbf{58.1 \pm 1.9}$	–

exposes the policy only to observable state and noisy evidence. No trajectories, demonstrations, simulator interactions, or MLLM calls from the target embodied environments are used during ADE training. PPO is optimized using the resulting ADE trajectories; implementation details and hyperparameters are provided in Appendix F.

4.1 Main Results

We evaluate all methods under a zero-shot cross-environment transfer setting. The orchestration policy is trained exclusively in the Abstract Decision Environment (ADE) and directly deployed to EB-ALFRED and EB-Habitat without policy adaptation, fine-tuning, or additional demonstrations. Each evaluation instance corresponds to a unique task-scene combination, requiring the policy to generalize across diverse embodied environments.

Table 1 shows that RARRL achieves the highest TSR across both EB-ALFRED and EB-Habitat and across all six capability categories. The largest gains occur on tasks involving greater ambiguity or perceptual difficulty, namely Complex and Visual on EB-Habitat and Long on EB-ALFRED, suggesting resource-aware reasoning is particularly beneficial in these settings. We compare RARRL with a non-RL learned trigger, SC-MLP, in Appendix M. Additional comparison in Appendix N. RARRL retains this advantage when the reasoning backend is replaced with Gemma-3-27B-it (Team et al., 2025) in Appendix P. We further evaluate RARRL on VirtualHome (Puig et al., 2018) and ALFWorld (Shridhar et al., 2020) in Appendix G.

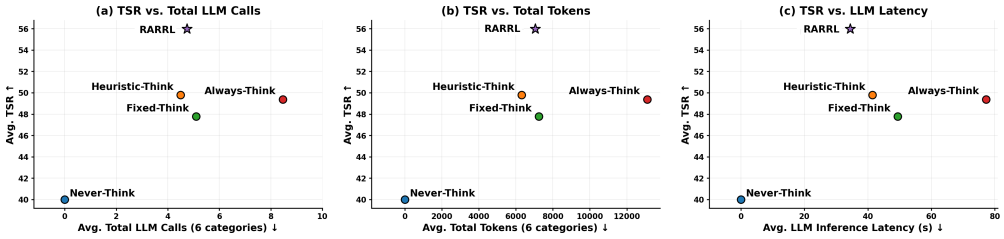

Figure 2: Accuracy-efficiency trade-off of RARRL and baseline reasoning strategies, averaged across all six EB-ALFRED capability categories. Panel (a) characterizes reasoning frequency through the total number of LLM calls, panel (b) quantifies the resulting token consumption, and panel (c) measures the actual wall-clock inference cost using Qwen2.5-VL-72B-Instruct. In all panels, higher TSR indicates better task performance, while lower values on the horizontal axes indicate lower reasoning cost.

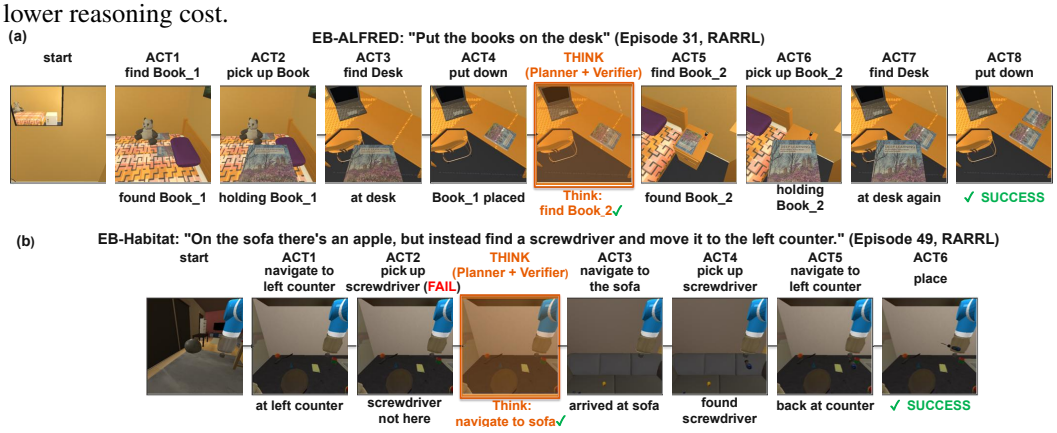

Figure 3: Qualitative trajectories from EB-ALFRED and EB-Habitat, illustrating how RARRL uses targeted reasoning at different execution-time decision points. (a) In EB-ALFRED (Episode 31, "Put the books on the desk"), the agent places one book successfully with no execution failure. RARRL then invokes THINK: the Planner proposes the candidate sequence shown in the figure, and the Verifier re-derives the agent's state directly from the current image, confirming it is consistent with the tracked history before approving the candidate. (b) In EB-Habitat (Complex Instruction Episode 49, "On the sofa there's an apple, but instead find a screwdriver and move it to the left counter"), the agent first fails to pick up the screwdriver at the left counter. RARRL then invokes THINK, and the Verifier similarly confirms the revised candidate against the current observation before approval.

4.2 Ablation Studies

Table 2 evaluates the contributions of RARRL's three core mechanisms. Removing Adaptive Orchestration causes the largest performance degradation on both platforms, followed by Persistent Planning and Resource Awareness. Each ablation removes only the corresponding mechanism while retaining the others. Removing Adaptive Orchestration replaces the learned THINK trigger with the fixed stall-based rule; removing Persistent Planning disables the plan queue; and removing Resource Awareness removes the budget-dependent state and constraint. We also evaluate the Verifier separately. Corrupting its judgments on EB-Habitat and removing it on EB-ALFRED both reduce TSR significantly (Appendix J), suggesting that Verifier outputs provide useful decision evidence.

5 Mechanism Analysis

Accuracy–efficiency trade-off. Figure 2 shows that RARRL achieves the highest average TSR while using substantially fewer LLM calls, tokens, and inference time than Always-Think. Compared with Fixed-Think and Heuristic-Think, RARRL also achieves higher TSR without a larger reasoning budget. These results suggest that RARRL's efficiency gains stem from selectively allocating reasoning to execution states where it is most useful, rather than simply increasing reasoning frequency or computation.

Table 3: Grounding robustness on EB-Habitat (Base category, $n = 50$ episodes each). Clean uses the unperturbed milestone-based grounding. Noise-10%/20% inject Gaussian noise $\mathcal{N}(0, \sigma^2)$ into the progress signal p_t before clipping to $[0, 1]$; Delay-1 exposes the policy to the progress history from one timestep earlier. TSR remains stable under moderate grounding perturbations, with no systematic increase in reasoning cost.

Setting	TSR (%)	THINK Invocations	PS	Latency (s)
Clean	86.5	2.06 ± 1.12	7.04 ± 1.83	4.12 ± 1.89
Noise-10%	84.0	2.28 ± 1.08	7.42 ± 1.74	4.73 ± 1.84
Noise-20%	82.0	2.56 ± 1.10	7.98 ± 1.85	4.77 ± 1.77
Delay-1	80.0	2.82 ± 1.14	8.08 ± 1.86	4.96 ± 1.64

Grounding robustness. To assess sensitivity to grounding errors, we perturb the progress signal p_t on EB-Habitat using Gaussian noise with standard deviation 10% and 20%, and a one-step delay (Table 3). TSR degrades under the evaluated grounding perturbations, while reasoning frequency remains broadly stable, indicating that the transferred policy is robust to moderate grounding perturbations.

Matched-state counterfactual intervention. To directly evaluate the incremental value of THINK at the decision point where it is invoked, we perform matched-state counterfactual interventions in the target environments. For each selected decision state, we create two rollout branches from the same underlying execution and planning state, forcing one branch to execute ACT and the other to invoke THINK. Both branches then continue under the same deployment pipeline. Thus, the comparison holds the decision state fixed and isolates the effect of the orchestration choice. Across 150 matched target decision states, THINK changes the eventual outcome in 117 states (78.0%): it converts ACT failure into success in 78 states (52.0%), while converting ACT success into failure in 39 states (26.0%). In the remaining 33 states (22.0%), the two interventions lead to the same outcome. Notably, the learned policy assigns substantially higher THINK probabilities to the positive states (0.82–0.96) than to the negative states (0.18–0.46), indicating that its preference for reasoning is aligned with the counterfactual value of the orchestration decision. To examine whether the observed zero-shot transfer depends on the specific inductive bias of the ADE, we further conduct an ADE mismatch stress test in Appendix Q.

Qualitative trajectories. Figure 3 provides concrete examples of how RARRL uses reasoning at execution-time decision points. The two trajectories illustrate complementary uses of THINK: RARRL invokes reasoning proactively in EB-ALFRED to identify a missing sub-goal despite successful execution so far, while it invokes reasoning reactively in EB-Habitat to recover from an execution failure. In both cases, the Planner's proposed continuation is reviewed by the Verifier against the current observation before being approved, after which the approved plan is executed through the Persistent Plan Queue without repeated reasoning, leading to successful task completion. These examples provide qualitative evidence that RARRL can invoke reasoning at different execution contexts rather than following a single fixed trigger. A step-by-step expansion of these trajectories, including the policy's action probabilities and the full state z_t at each decision point, is provided in Appendix L.

6 CONCLUSION

We introduced RARRL, a resource-aware orchestration framework that learns *when* to invoke reasoning in embodied agents. Trained in an Abstract Decision Environment, RARRL transfers to EB-ALFRED and EB-Habitat without retraining or target-task optimization. Additional evaluations on VirtualHome and ALFWorld further show that the frozen orchestration policy remains effective under different task distributions and task interfaces. Across these environments, RARRL achieves a favorable accuracy–efficiency trade-off, maintaining task performance while substantially reducing LLM calls, token usage, and inference latency. These results support treating embodied reasoning as a resource allocation problem, where a learned orchestration policy selectively invokes costly reasoning according to execution context.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing and improving the clarity and readability of the manuscript. We have not used generative AI tools for other tasks, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for language editing and proofreading. We have reviewed all AI-assisted work. We verified the accuracy and correctness of the final manuscript. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We have no ethical concerns related to the research presented in this work.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the framework and training procedure in the main paper and Appendix. The experimental setup, benchmark configurations, baseline implementations, evaluation metrics, and hyperparameters are documented in the main paper and Appendix. Additional implementation details, ablation settings, robustness analyses, and statistical evaluation procedures are provided in the Appendix to facilitate reproduction of the reported experiments.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Mohammad Ali Alomrani, Yingxue Zhang, Derek Li, Qianyi Sun, Soumyasundar Pal, Zhanguang Zhang, Yaochen Hu, Rohan Deepak Ajwani, Antonios Valkanas, Raika Karimi, et al. Reasoning on a budget: A survey of adaptive and controllable test-time compute in llms. *arXiv preprint arXiv:2507.02076*, 2025.

R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23, 1986. doi: 10.1109/JRA.1986.1087032.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Liang Chen, Yichi Zhang, Shuhuai Ren, Haozhe Zhao, Zefan Cai, Yuchi Wang, Peiyi Wang, Tianyu Liu, and Baobao Chang. Towards end-to-end embodied decision making via multi-modal large language model: Explorations with gpt4-vision and beyond. *arXiv preprint arXiv:2310.02071*, 2023.

Yang Chen, Hong-Jie You, Jie-Jing Shao, Xiao-Wen Yang, Ming Yang, Yu-Feng Li, and Lan-Zhe Guo. re^2 agent: Reflection and re-execution agent for embodied decision making. In *NeurIPS 2025 Challenge on Foundation Models for Embodied Agents*, 2025.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Stuart Eiffert, Nathan D Wallace, He Kong, Navid Pirmarzdashti, and Salah Sukkarieh. Resource and response aware path planning for long-term autonomy of ground robots in agriculture. *Field Robotics*, 2:1–33, 2022.

Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-act: Improving planning of agents for long-horizon tasks. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 15419–15462, 2025.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *Uai*, volume 4, pp. 162–169, 2004.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. *AI&M*, 1(2):3, 2006.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li E Li, Ruohan Zhang, et al. Embodied agent interface: Benchmarking llms for embodied decision making. *Advances in Neural Information Processing Systems*, 37:100428–100534, 2024.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International conference on robotics and automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. *Advances in Neural Information Processing Systems*, 36:23813–23825, 2023.

Junhong Lin, Xinyue Zeng, Jie Zhu, Song Wang, Julian Shun, Jun Wu, and Dawei Zhou. Plan and budget: Effective and efficient test-time scaling on reasoning large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ctspw4CqbS.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Sherif AS Mohamed, Mohammad-Hashem Haghbayan, Antonio Miele, Onur Mutlu, and Juha Plosila. Energy-efficient mobile robot control via run-time monitoring of environmental complexity and computing workload. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7587–7593. IEEE, 2021.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *2018 IEEE/cvf conference on computer vision and pattern recognition*, pp. 8494–8502. IEEE, 2018.

Allen Z. Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, Zhenjia Xu, Dorsa Sadigh, Andy Zeng, and Anirudha Majumdar. Robots that ask for help: Uncertainty alignment for large language model planners. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

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

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on robot learning*, pp. 894–906. pmlr, 2022.

Bruno Siciliano, Oussama Khatib, and Torsten Kröger. *Springer handbook of robotics*, volume 200. Springer, 2008.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. *Advances in neural information processing systems*, 36:58202–58245, 2023.

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

Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in neural information processing systems*, 34:251–266, 2021.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

Zirui Xu and Vasileios Tzoumas. Resource-aware distributed submodular maximization: A paradigm for multi-robot decision-making. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 5959–5966. IEEE, 2022.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, et al. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. *arXiv preprint arXiv:2502.09560*, 2025.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In *Proceedings of The 8th Conference on Robot Learning*, volume 270, pp. 3157–3181. PMLR, 2024.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

APPENDIX

A IMPLEMENTATION DETAILS

Evaluation protocol. All main experiments use Qwen2.5-VL-72B-Instruct with deterministic decoding (temperature $= 0$). RARRL's correction budget $b_{\max} = 3$ is selected using only the ADE training environment and fixed before target evaluation. Fixed-Think uses the task-independent period $k = 3$, specified before target-test evaluation. Heuristic-Think uses the threshold reported in KnowNo, without target-test tuning. No EB-ALFRED or EB-Habitat test episodes are used for hyperparameter selection, prompt tuning, or target-specific calibration. Detailed model, decoding, prompting, simulator, and computational settings are provided in Table 4.

Table 4: Implementation details for reproducibility.

Item	Setting
MLLM	Qwen2.5-VL-72B-Instruct
Decoding	temperature $= 0$ (greedy), max_tokens $= 1024$
Action parsing	JSON schema-constrained decoding (xgrammar backend)
Context window	16,384 tokens
Prompt	Fixed template with 10-shot in-context examples
Retry policy	None; guided decoding enforces valid JSON
Step cap	30 steps or 10 invalid actions per episode
$b_{\max}$	3 (selected via ADE-only sweep; fixed before target evaluation)
PPO	PPO (Schulman et al., 2017)
Random seeds	15 independent PPO training seeds
Hardware	4× NVIDIA RTX A6000, 4-way tensor parallelism

Each of the 15 random seeds was used to train an independent PPO policy, and each resulting policy was evaluated on the same fixed set of test episodes using deterministic decoding. We first computed the performance of each policy over the complete evaluation set and then calculated the mean and 95% confidence interval across these 15 results. Thus, the reported confidence interval captures variation due to PPO training randomness rather than uncertainty from resampling test episodes.

B ADE DESIGN AND PARAMETERS

The ADE instantiates the orchestration problem with the following parameters. All values are swept in sensitivity experiments (Sec. F.1); no single value is load-bearing.

Table 5: ADE instantiation parameters.

Parameter	Value	Description
P_{lo}	0.35 (swept [0.30, 0.45])	Base risk-survival probability
λ	1.0 (swept [0.1, 1.0])	THINK cost weight
μ	0.5 (swept [0.1, 0.7])	Failure penalty
N_{stages}	3	Decision stages per episode
κ_{hi}	0.9	Evidence reliability upper bound
q	0.0 (swept [0, 0.3])	Evidence noise level

Execution-time budget enforcement. While Sec. 3.3 models the reasoning budget as an abstract cost that the policy learns to conserve, at deployment this budget is enforced as a hard constraint on the executed trajectory. Each invocation of THINK decrements the remaining budget $\tilde{b}_t$ by one, regardless of whether the proposed candidate is approved or rejected. Once $\tilde{b}_t$ reaches zero, the policy is restricted to `ACT` for the remainder of the episode: π_θ is no longer queried for the reasoning decision, and all subsequent actions are produced directly by the Task Executor without further verification. This is the "no recourse once exhausted" property referenced in Sec. 3.3: the policy cannot borrow against future budget or request additional reasoning once depleted, which makes $b_{\max}$ a hard ceiling on the number of reasoning invocations rather than a soft latency penalty.

On the granularity of budget accounting. The budget $b_{\max}$ constrains the number of *decision points* at which the policy may request additional deliberation, rather than the raw number of underlying LLM calls. A single `THINK` invocation executes the fixed Planner and Verifier reasoning pipeline. The Planner performs one LLM invocation, and the Verifier performs one LLM invocation. Consequently, one `THINK` invocation corresponds to two underlying LLM invocations in the current implementation. This internal decomposition does not introduce an additional decision variable for the orchestration policy; it specifies how a requested deliberation is executed. Importantly, we do not use $b_{\max}$ as a proxy for computational cost when evaluating resource efficiency.

Isolating adaptive orchestration from execution architecture. Among the four baselines, Always-Think, Fixed-Think, and Heuristic-Think share the identical execution architecture with RARRL: the same MLLM backbone (Qwen2.5-VL-72B-Instruct), the same Planner and Verifier modules, and the same Persistent Plan Queue implementation, directly reused from the RARRL codebase. The only difference across these three methods and RARRL is the triggering rule that determines when `THINK` is invoked. Never-Think, by contrast, represents the zero-reasoning lower bound defined by the EmbodiedBench benchmark itself and does not invoke any reasoning module; it is included as a reference point rather than as a controlled comparison for isolating the effect of adaptive orchestration.

C Reasoning-Effect Model and Independent Risks

Reasoning-effect model (decision evidence). We model the effect of reasoning as the acquisition of *decision-relevant evidence* about the latent decision context, not as a direct edit of a success probability.

Definition 1 (Decision Evidence)**.** *Decision evidence is task information, acquired through reasoning, that reduces the policy's uncertainty about the latent decision context ϑ.*

Invoking THINK at budget c returns a noisy observation of one decision facet of ϑ: reasoning surfaces task-relevant evidence that reduces uncertainty about that facet, and budget c sets the precision of the observation. The policy accumulates this evidence in its observation and chooses an orchestration action accordingly. Because a single THINK observes only one noisy facet, no single call fully reveals ϑ; the policy must decide *when* to acquire more evidence and *when* to stop and act.

Decision risk. Each decision facet resolves an independent source of risk. We model this risk as contributing independently to the task-advance probability: when a facet is addressed by THINK, it contributes a high survival probability; when unaddressed, it contributes the low base rate P_{lo}. Accordingly, an unresolved required risk contributes a low base survival probability. This is what gives each facet a stable, non-substitutable value: in a context that carries multiple simultaneous risks (our "critical" context), addressing only one facet leaves the other risk unresolved, so a competent policy must acquire evidence for *all* active facets, a prediction we verify directly in Section F.3 (Table 10). The specific survival-probability constants are environment-instantiation parameters, not method parameters; Section F.2 reports a sensitivity sweep confirming the qualitative structure is stable across a range of their values, so no single value is load-bearing. Reasoning is therefore valued exactly as an information source: it changes the policy's decision, and the environment transition responds to that changed decision. This is what lets the abstract environment and the deployment setting share a single mechanism: in both, THINK $\rightarrow$ evidence $\rightarrow$ better decision $\rightarrow$ transition $\rightarrow$ return. Crucially, this matches what a real language model does for an embodied agent: an LLM that replans, verifies, or reflects does not lower the physical failure rate of a fixed motor command—it supplies the agent with information it did not have, changing which decision it makes. This information-based view yields a clean characterization of when reasoning helps. We state it as a *property of the environment construction* (it follows directly from how the ADE is defined, and is not a claim about the RARRL algorithm or about real reasoning systems).

Environment Property (Evidence-dependent reasoning). *Within the ADE, accumulated decision evidence determines the probability of task advance through the evidence state represented by M_t. The orchestration policy uses the resulting evidence state, together with the execution state, to select between* ACT *and* THINK. *Thus, M_t is part of the ADE transition model rather than a separate transition channel, while the policy determines how evidence acquisition is allocated over time.*

This is immediate from the construction: the transition and cost depend on the chosen orchestration o and on ϑ, but not on the acquired evidence except through the action the policy selects given that evidence. For a policy that already knows ϑ, additional evidence cannot change the selected orchestration and so cannot change return; within the ADE, the value of evidence is therefore exactly the reduction of uncertainty about ϑ that a context-uninformed policy would otherwise lack. We include an empirical check of this behavior (an oracle that acquires evidence beyond what it needs gains no success, only cost) in Section F.2.

This is a statement about the ADE, not a general claim about real reasoning systems (in which reasoning may also create new internal state beyond context disambiguation); we scope it deliberately so it is not read as an assertion about LLMs at large. The regularity it formalizes—that reasoning is valuable exactly when the agent is uncertain about how to decide, and wasteful when the decision is already clear—matches the test-time-compute literature Snell et al. (2024); Alomrani et al. (2025); Lin et al. (2026), where large reasoning models are observed to "overthink simple problems while underthinking hard ones" Alomrani et al. (2025) and uncertainty-aware budgeting improves the correctness–cost trade-off Lin et al. (2026). We emphasize that the ADE's evidence model is a deliberate *surrogate* for a real reasoning backend, used only during training. We adopt a surrogate rather than invoking a real LLM inside the training loop because on-policy reinforcement learning requires on the order of hundreds of thousands of interaction steps, for which in-the-loop LLM inference is prohibitively expensive; decoupling policy learning from real inference is precisely what makes large-scale, many-seed orchestration training feasible, and is a deliberate design choice rather than a limitation. Critically, our conclusions do not rest on how reliable this surrogate evidence is.

Section F.2 shows that sweeping the evidence-noise level over a wide range leaves the ranking of methods unchanged and the learned policy degrades gracefully, relying less on reasoning as it becomes less reliable.

D GROUNDING MAP

Grounding the abstraction in embodied environments. The analysis above establishes, in principle, when a policy trained in the ADE transfers: exactly when the mapping $f : \mathcal{S} \to \mathcal{Z}$ (Eq. equation 3) satisfies Q^*-irrelevance (Assumption 1), with graceful degradation controlled by behavioral distance (Eq. equation 5). That analysis is deliberately agnostic to any particular environment; here we make it concrete. The decisive practical question is *grounding*: whether a usable f can be constructed for a target environment without access to privileged simulator state. We evaluate on two complementary embodied benchmarks: **EB-ALFRED** (Yang et al., 2025) and **EB-Habitat** (Yang et al., 2025), built on the AI2-THOR (Kolve et al., 2017) and Habitat (Szot et al., 2021) simulators, respectively. EB-ALFRED focuses on long-horizon household manipulation, whereas EB-Habitat focuses on navigation and object rearrangement, representing two complementary embodied task domains. These two benchmarks span different physics engines, task distributions, and action spaces (70 vs. 232 actions), making them a strong test of whether the grounding map f can be constructed from general principles rather than benchmark-specific engineering. The concrete mapping uses only quantities a deployed agent could realistically access. *Task progress* p_t is estimated from agent-observable execution outcomes (e.g., the number of decision-level commitments executed successfully), without relying on ground-truth task-completion signals from the simulator. *Execution history* h_t is the recent sequence of action and outcome pairs, maintained from per-step environment feedback. *Remaining budget* b_t is the normalized remaining reasoning-call budget, which we control directly. *Observable uncertainty* u_t is estimated from a sliding window of recent failures, without any privileged uncertainty signal. Decision evidence $(e_{\text{plan}}, \kappa_{\text{plan}})$ is extracted from each LLM reasoning call via the grounding map f (Sec. C), completing the full 6-dimensional abstract observation. The quality of transfer is governed by the behavioral-distance bound of Eq. equation 5: the more nearly f preserves the decision-relevant statistics, the smaller the induced sub-optimality.

Limitation. A key limitation of this work is that our evaluation of transfer is conducted solely on two simulated embodied benchmarks, without real-robot experiments. Constructing the grounding map f for a physical robot would require additional perceptual and task-state estimation components beyond what is evaluated here. This reflects a common trade-off between reproducibility, cost, and safety on one hand, and real-world applicability on the other; simulated benchmarks offer a standardized, reproducible setting for evaluating orchestration policies while substantially reduc-

Table 6: Empirical analysis of abstraction sufficiency on EB-ALFRED and EB-Habitat. We report the unconditional variance of the binary episode-success outcome, the average within-bin variance after grouping states by discretized z_t, and the resulting variance reduction. Only bins containing at least three observations are included. Coverage denotes the fraction of state instances assigned to valid bins.

Environment	N	Precision	Overall Var.	Within-bin Var.	Reduction	Valid Bins	Coverage
EB-ALFRED	2,745	0 decimals	0.1974	0.1789	9.4%	26	99.7%
		1 decimal	0.1974	0.1563	20.8%	128	89.2%
		2 decimals	0.1974	0.1562	20.9%	111	86.6%
EB-Habitat	3,108	0 decimals	0.1972	0.1860	5.7%	36	99.7%
		1 decimal	0.1972	0.1663	15.7%	153	89.8%
		2 decimals	0.1972	0.1660	15.8%	152	89.7%

ing evaluation cost and risk. Extending grounding to real-robot deployment remains an important direction for future work.

D.1 Deployment Transition

At deployment, the frozen policy selects $o_t \in \{\textsc{Act}, \textsc{Think}\}$ from the grounded state $z_t = f(s_t, Q_t, e_t, \kappa_t, b_t)$. A THINK decision invokes the Planner–Verifier pipeline and updates the planning state, including the Current Subgoal Buffer and evidence:

$$(Q_t, e_t, \kappa_t, b_t) \to (Q_{t+1}, e_{t+1}, \kappa_{t+1}, b_{t+1}).$$

A subsequent ACT decision is executed by the Task Executor, which selects a concrete primitive action

$$a_t^{\text{exec}} \sim \pi_{\text{exec}}(\cdot \mid s_t, Q_t, e_t, \kappa_t), \qquad s_{t+1} \sim P_{\text{env}}(\cdot \mid s_t, a_t^{\text{exec}}).$$

Thus, at deployment, evidence affects physical outcomes through the updated planning state and the concrete action selected by the executor; it does not directly modify P_{env}. In contrast, Eq. (1) defines the evidence-dependent surrogate transition used within the ADE.

E Conditions for Transfer: A Design Rationale

E.1 Why Can the Policy Transfer?

Conditional transfer rationale. The abstraction $f : \mathcal{S} \to \mathcal{Z}$ provides a conditional rationale for transferring the orchestration policy across embodied environments. If states that map to the same abstract decision state have equivalent orchestration consequences, then a policy defined over $\mathcal{Z}$ need not depend on environment-specific state details. The Q^*-irrelevance condition of Li et al. (2006) formalizes this intuition: if

$$f(s) = f(s') \Rightarrow Q^*(s, a) = Q^*(s', a),$$

then the abstraction preserves optimal orchestration decisions. However, we do not claim that this condition is established for EB-ALFRED and EB-Habitat. Likewise, the behavioral-distance bound used in our discussion is a within-MDP result and does not constitute a proof of transfer between the two embodied environments. Thus, these results provide a conditional design rationale rather than a theoretical guarantee of cross-environment transfer.

Empirically, the same frozen orchestration policy transfers from the ADE to both EB-ALFRED and EB-Habitat without retraining or target-task optimization, providing empirical evidence that the shared abstraction captures useful orchestration structure across these platforms.

E.2 Empirical Analysis

Experimental setup. We pool the abstract decision state z_t observed by the policy at every decision point across all six capability categories on EB-ALFRED and EB-Habitat, together with the eventual binary success outcome (0/1) of the episode that decision point belongs to. States are

grouped by rounding each dimension of z_t to a fixed number of decimal places; states falling into the same discretized bin are treated as approximately equivalent under the abstraction. For each bin with at least three observations, we compute the variance of the success outcome within that bin and average the within-bin variance across valid bins, weighted by bin size. We compare this value with the unconditional variance of the success outcome across all observations and repeat the procedure at zero, one, and two decimal places (Table 6).

Empirical abstraction sufficiency. Across both environments, grouping states by z_t consistently reduces outcome variance. At one-decimal precision, the reduction is 20.8% on EB-ALFRED and 15.7% on EB-Habitat; two-decimal precision produces nearly identical reductions of 20.9% and 15.8%, respectively. The consistent pattern across two structurally different environments provides empirical support for the usefulness of the abstract decision state in capturing orchestration-relevant variation, while not constituting a test or proof of the Q^*-irrelevance condition. The consistent reduction across both environments provides empirical support for the usefulness of the abstract decision state for capturing orchestration-relevant variation, while not constituting a test or proof of the Q^*-irrelevance condition. Because EB-ALFRED and EB-Habitat differ in simulator, action space, and task distribution, this cross-environment consistency suggests the effect is not an artifact specific to either platform.

E.3 Decision Abstraction: Theoretical Grounding

We use established state-abstraction theory to motivate why an orchestration policy trained in the Abstract Decision Environment may transfer across embodiments, and—equally importantly—the conditions under which this transfer may degrade. This discussion is intended as a design motivation rather than a formal guarantee of cross-environment performance. Our treatment is grounded in the established theory of state abstraction for MDPs Li et al. (2006); Ferns et al. (2004); we do *not* claim a new abstraction theorem. Instead, we cast our design as a specific instance of a known abstraction class, and treat the degree to which it holds across real embodied environments as an *empirical* question.

Abstraction mapping. Let

$$f : \mathcal{S} \to \mathcal{Z} \tag{3}$$

map an embodiment-specific robot state $s \in \mathcal{S}$ to an abstract decision state $z = (p, h, b, u, e^{\text{plan}}, \kappa^{\text{plan}}) \in \mathcal{Z}$ as defined in the abstract decision state z_t. The orchestration policy $\pi_\theta(a \mid z)$ acts solely on $\mathcal{Z}$ and is therefore, by construction, agnostic to robot morphology, simulator implementation, and perceptual front-end. The question is not whether π_θ ignores embodiment, it does, by design, but whether ignoring embodiment is *sufficient* for near-optimal orchestration in a new environment.

The transfer assumption (Q^*-irrelevance). Transferability is grounded in the following sufficient abstraction condition, namely the Q^*-irrelevance abstraction (ϕ_{Q^*}) of Li et al. (2006), instantiated for the orchestration action space. We do not introduce a new abstraction notion.

Assumption 1 (Q^*-irrelevance of f)**.** *The mapping f satisfies Q^*-irrelevance for reasoning orchestration if, for all robot states s, s' with $f(s) = f(s')$,*

$$Q^*(s, a) = Q^*(s', a), \qquad \forall a \in \{\texttt{ACT}, \texttt{THINK}(c)\}. \tag{4}$$

Assumption 1 is a sufficient theoretical condition that we posit for the abstraction; we do not claim to prove or empirically verify that it holds exactly across embodied environments. Under this condition, the classical result of Li et al. (2006) implies that the optimal decisions can be preserved by the abstraction. Thus, if the condition approximately holds across two environments sharing the same abstract decision semantics, an orchestration policy defined over $\mathcal{Z}$ can transfer without being re-optimized for the target environment. This provides the theoretical motivation for RARRL's transfer design, rather than an empirical claim that the condition is universally satisfied.

Operational sufficiency of evidence. A second, orthogonal assumption concerns the evidence channel rather than the abstraction mapping. Because the decision context ϑ is latent, the policy conditions on the acquired evidence e rather than on ϑ itself; for this to suffice we posit:

Assumption 2 (Operational sufficiency of evidence)**.** *The decision evidence acquired through reasoning provides sufficient decision-relevant information for near-optimal orchestration: a policy conditioned on the accumulated evidence e can approach the return of a policy that observes the latent context ϑ directly.*

This is an *operational* adequacy assumption—sufficiency of the *information* for near-optimal orchestration—not a claim of statistical or information-theoretic sufficiency; we do not assert that e is a sufficient statistic for ϑ. Under this assumption, decision abstraction becomes *applicable* in the presence of a latent context: the policy need not recover ϑ exactly, only acquire enough evidence to orchestrate well. We do not claim the assumption *establishes* transfer; it is a condition under which the abstraction framework applies, and the extent to which it holds is an empirical matter that our evidence-noise study (Section F.2) quantifies.

Why exact sufficiency will not hold—and why this is acceptable. In practice f is unlikely to satisfy Q^*-irrelevance *exactly* across heterogeneous embodiments: the same abstract state z may correspond to subtly different optimal reasoning values in, say, a kitchen-manipulation task versus a text-navigation task. This is not a defect peculiar to our abstraction; Li et al. (2006) give explicit counterexamples in which learning under an approximately-Q^* abstraction can converge to suboptimal behavior. We therefore do not rely on exact sufficiency. Instead we appeal to the *quantitative* robustness of behaviorally grounded abstractions. For states within a common MDP, the bisimulation metric of Ferns et al. (2004) bounds the difference in optimal values by behavioral distance: there exists a constant K such that

$$\left|Q^*(s,a) - Q^*(s',a)\right| \;\le\; K\, d(s,s'), \tag{5}$$

where $d(\cdot,\cdot)$ is the bisimulation metric. Equation equation 5 reframes transfer as a controlled trade-off rather than an all-or-nothing guarantee: the smaller the behavioral discrepancy that f collapses, the smaller the induced sub-optimality. The design principle that follows is concrete—f should retain exactly those variables whose perturbation moves Q^* (fractional progress, recent execution outcomes, remaining-budget ratio, observable uncertainty) and discard those that do not (raw observations, kinematics, and absolute platform-dependent costs).

From within-MDP bounds to cross-environment transfer. We are careful about the scope of Eq. equation 5: as stated, it is a within-MDP result. Since ADE, AI2-THOR, and Habitat have different state spaces, transition kernels, action spaces, and embodiment interfaces, Eq. equation 5 is not a formal performance bound between these environments. Instead, we use it as a design motivation for constructing an abstract state that preserves decision-relevant information. The assumption that such decision-relevant structure is shared across environments is a modeling hypothesis, and the resulting transfer behavior is evaluated empirically in Section 4.

Platform-independent cost shaping. A subtlety in the reward of Eq. equation 2 must be resolved for Assumption 1 to be meaningful. The raw reasoning cost,

$$r_t^{\text{raw}} = \lambda_L\, L_t + \lambda_C\, C_t, \tag{6}$$

where L_t is API latency and C_t is token count, is platform-dependent; its absolute scale differs across LLM backends and runtimes, which would make the reward—and hence Q^*—depend on embodiment, contradicting Eq. equation 4. We therefore train on a *normalized* cost,

$$r_t^{cost} = w_L\, \tilde{L}_t + w_C\, \tilde{C}_t, \qquad \tilde{L}_t = \frac{L_t}{L_{\max}}, \quad \tilde{C}_t = \frac{C_t}{C_{\max}}, \tag{7}$$

where $L_{\max}, C_{\max}$ are the per-decision budget caps that define the ADE, so that $\tilde{L}_t, \tilde{C}_t \in [0,1]$ express cost as a fraction of budget rather than in absolute units. Under this shaping the reward is a function of the abstract decision state alone, keeping the orchestration objective embodiment-invariant. Absolute latencies and token counts re-enter only as *reporting* quantities in Section 4, never as training signal.

Table 7: PPO hyperparameters used in all experiments.

Hyperparameter	Value
Learning rate	3×10^{-4}
Clip ratio ϵ	0.2
GAE λ	0.95
Discount γ	0.99
Entropy coefficient	0.03
Seeds	15
Iterations	300

Table 8: Sensitivity to λ (cost weight) and μ (failure penalty), mean $\pm$ 95% CI over 15 seeds. TSR is stable across a wide range while RF/Tok vary monotonically, indicating controllable, non-brittle behavior. **RF** reasoning invocations per episode; **Tok** LLM token per episode

λ **sweep** (μ=0.5)			
λ	TSR (%)	RF	Tok
0.1	81.2 $\pm$0.8	2.83	726
0.3	80.3 $\pm$0.9	2.76	706
0.5	81.7 $\pm$1.0	2.60	667
0.7	81.0 $\pm$0.9	2.49	638
1.0	80.9 $\pm$0.7	2.34	599
μ **sweep** (λ=1.0)			
μ	TSR (%)	RF	
0.1	80.9 $\pm$0.9	2.09	
0.3	80.7 $\pm$0.8	2.19	
0.5	80.9 $\pm$0.7	2.34	
0.7	80.4 $\pm$0.9	2.61	

F PPO Training Details

F.1 Sensitivity to Reward Coefficients

Table 8 sweeps the cost weight λ and the failure penalty μ. The reward coefficients act as clean control knobs on *how much* the policy reasons while leaving success essentially unchanged. Across $\lambda \in [0.1, 1.0]$ (a $10\times$ range) task success stays within 80.3–81.7%, while reasoning frequency and tokens decrease monotonically as λ grows ($2.83 \to 2.34$ calls, $726 \to 599$ tokens): λ cleanly trades reasoning cost for nothing measurable in success, which is exactly the controllability one wants. The failure penalty μ monotonically increases reasoning frequency ($2.09 \to 2.61$ calls as μ grows): a heavier failure penalty makes the policy reason more to avoid mistakes, as expected. Performance is therefore *robust* to both coefficients rather than reliant on fine-tuning; we fix λ=1.0 and μ=0.5, chosen from this sweep rather than per-task tuning.

Conclusions do not depend on the surrogate's exact values. A natural concern is whether our conclusions hinge on the hand-set constants of the surrogate evidence model. Two analyses address this. First, the evidence-noise study in Table 9 Section F.2 sweeps the reliability of the surrogate reasoning over a wide range and shows the ranking of methods is invariant, so our conclusions do not depend on how reliable the surrogate reasoning is. Second, the transition's risk-survival constants are environment-instantiation parameters, not method parameters; Appendix H reports a sensitivity sweep over the base incident rate confirming that the qualitative structure, in particular the non-substitutable value of each decision facet, is stable across a continuous range of their values, so no single constant is load-bearing.

Table 9: Robustness to unreliable reasoning (mean $\pm$ 95% CI, 15 seeds). As evidence-noise q rises, RARRL's advantage over the strongest baseline is preserved at every level, and RARRL reduces its reliance on reasoning (RF falls) while degrading gracefully rather than collapsing. RARRL-adv: RARRL minus the best of single-step and heuristic.

q	RARRL TSR (%)	Best baseline	RARRL-adv	RARRL RF
0.0	80.9 $\pm$0.7	59.0	+21.9	2.34
0.1	75.3 $\pm$1.1	56.7	+18.6	2.25
0.2	70.3 $\pm$1.5	53.6	+16.7	2.13
0.3	65.1 $\pm$1.8	48.3	+16.7	1.96

Table 10: Evidence acquisition by latent context (mean calls per episode $\pm$ 95% CI, 15 seeds). Within THINK, the Planner and Verifier are invoked as needed based on the acquired evidence: little reasoning in routine contexts, Planner evidence alone in ambiguous ones, and both Planner and Verifier evidence in contexts carrying multiple simultaneous risks.

Latent context	Planner calls	Verifier calls
Routine	0.27 $\pm$0.02	0.00 $\pm$0.00
Ambiguous	0.47 $\pm$0.02	0.00 $\pm$0.00
Recovery	0.34 $\pm$0.02	0.30 $\pm$0.02
Critical	0.35 $\pm$0.02	0.31 $\pm$0.03

F.2 Robustness to Unreliable Reasoning

A reasoning module is not always helpful: its guidance may be unreliable. We model this with an evidence-noise level q: with probability governed by q, an acquired evidence bit is flipped, and the reliability of resolving a decision risk degrades accordingly (Sec. C). This single knob varies how trustworthy reasoning is, and lets us ask two questions: does the ranking of methods survive unreliable reasoning, and how does the learned policy adapt?

Two findings emerge. First, the ranking is *invariant* to reasoning reliability: RARRL beats the strongest baseline at every noise level, by a margin that stays above $+16$ points across the whole range. Our conclusions therefore do not depend on how reliable the surrogate reasoning is — a stronger statement than robustness to a particular gain constant, since it varies the very trustworthiness of reasoning. Second, RARRL *adapts*: as q rises, it reduces reasoning frequency ($2.34 \rightarrow 1.96$ calls) and its success degrades smoothly rather than collapsing. When reasoning becomes unreliable, the policy leans more on direct action, treating reasoning as an option with state-dependent value rather than an always-beneficial oracle. This directly addresses the concern that our framework assumes reasoning is always helpful: the learned controller discovers, from reward alone, that unreliable reasoning should be trusted less.

F.3 What Evidence Does the Policy Acquire?

The preceding results show *that* RARRL orchestrates reasoning well; we now examine *how*, by asking which decision facets the learned policy acquires evidence for in each latent decision context. Because the context is hidden from the policy, any context-appropriate structure in its behavior must be inferred from the decision evidence it acquires, not read off a label. Table 10 reports, for each latent context, the mean number of calls addressing each decision facet that the trained policy issues per episode (context is known to the environment and used only for this analysis, never provided to the policy).

The acquisition pattern mirrors the decision risks each context carries. Routine contexts, which need no reasoning, elicit almost none. Ambiguous contexts, which carry only subgoal-selection risk, draw Planner evidence and essentially no Verifier evidence. Most tellingly, *critical* contexts, which carry both subgoal and execution risk, draw *both* Planner and Verifier evidence (0.35 and 0.31 calls), while *recovery* contexts, carrying execution and consistency risk, recruit Verifier evidence, far above its near-zero use elsewhere. This is the behavioral signature the independent-risk design predicts:

because a single module resolves only one risk, the policy must learn to acquire evidence from both Planner and Verifier in exactly the contexts that carry multiple risks, and it does so from reward alone, without ever observing the latent context. This is also the result that the earlier, additive formulation of the environment could not produce (Appendix H).

An honest residual. The correspondence is not perfect: in recovery contexts the policy still acquires evidence for the facet addressing planning risk (0.34) at a rate comparable to the facets that context actually needs, even though that facet addresses no recovery risk. A marginal-utility analysis (Appendix H) confirms that acquiring this evidence in recovery contexts is suboptimal under the environment dynamics, adding it strictly lowers return, so we attribute this residual to imperfect policy convergence (this facet is useful in two of four contexts and appears to retain some default-action tendency), not to the environment design. The qualitative facet and context correspondence, which is the reportable result, holds clearly.

F.4 Discussion: Scope and Limitations of the Transfer Claim

Is f a universal feature extractor, or a hand-crafted adapter? The grounding map f requires environment-specific engineering: extracting task progress, failure history, and uncertainty from a new environment's runtime signals requires code specific to that environment's API. We do not claim that f is a universal feature extractor requiring zero engineering effort. What we claim is more precise: the *orchestration policy* π_θ transfers zero-shot, with its weights are never updated after ADE training, while f serves as a lightweight, retraining-free deployment adapter. This distinction is standard in hierarchical transfer learning. The decision variables $(p_t, h_t, b_t, u_t, e_r, \kappa_r)$ are not ALFWorld-specific constructs; they are general signals present in any embodied task with subgoal structure, a budget, and observable execution history. The most direct empirical refutation of the "purpose-built key" hypothesis is our EB-Habitat result: the *same* policy, with *identical* weights and *identical* f variable definitions, achieves positive contribution on all six EB-Habitat evaluation subsets ($+6.0\%$ to $+22.0\%$) despite Habitat-Sim and AI2-THOR being completely different physics engines, furniture rearrangement and kitchen manipulation being completely different task distributions, and Habitat's action space (70 actions) being entirely disjoint from ALFRED's (232 actions). A key purpose-built for ALFWorld cannot open a Habitat lock.

Were the ADE parameters tuned against ALFWorld data? The ADE parameters (in particular P_{lo}, the base risk-survival constant) were determined entirely without access to any real-environment data. The design process was: (1) specify the independent-risk multiplicative transition on theoretical grounds; (2) sweep P_{lo} over $[0.30, 0.45]$ and verify that Verifier's marginal value is robustly positive throughout (Appendix H); (3) select a representative value from within this range. No ALFWorld trajectories, EB-ALFRED results, or Habitat statistics were consulted at any stage of ADE design or parameter selection. The cross-environment evidence reinforces this: the same P_{lo} supports positive transfer to three structurally different environments (ALFWorld, EB-ALFRED, EB-Habitat) spanning two physics engines, three task distributions, and action spaces differing by a factor of three. Parameters reverse-engineered from one environment would not generalize this broadly.

Transfer, not mere deployment replacement. Two pieces of evidence indicate that RARRL's zero-shot performance reflects genuine transfer of decision-level structure, rather than the ADE happening to replicate any one target environment. First, the transfer gap is not zero: comparing the policy's performance within ADE (Table 8), where reasoning outcomes are modeled by a backbone-agnostic surrogate (Appendix C), to its zero-shot performance on EB-ALFRED and EB-Habitat with a real MLLM backend (Table 1) reveals a substantial, non-trivial drop, indicating real distribution shift between the abstract training environment and the deployed platforms rather than the two settings being behaviorally equivalent. Second, and most decisively, the EB-ALFRED and EB-Habitat results together resolve this question empirically: the ADE shares no task distribution, no physics engine, and no action vocabulary with either AI2-THOR or Habitat-Sim. The positive contribution across all subsets of both benchmarks cannot be explained by the ADE being a "shadow simulator" of any one target environment. What the ADE captures is the *decision-level structure* of orchestration, when to reason and when to act, and this structure transfers across environments not

Table 11: Zero-shot transfer from ADE to ALFWorld (3,553 episodes, Llama-3.1-8B-Instruct). The ADE-trained policy is evaluated with frozen weights; only the environment-specific grounding map is constructed for ALFWorld.

Environment	TSR (%)↑
ADE (trained)	80.9 ±0.7
ALFWorld (zero-shot)	79.6 ±0.8
Transfer gap	**1.3 pts**

Table 12: Performance on ALFWorld. (mean ± 95% CI over 15 seeds). **Tok** LLM token per episode

Method	TSR (%)↑	Tok	PS
Always-Think	29.0 ±1.3	1270±3.1	8.31±0.3
Never-Think	38.9 ±1.2	0	7.10±0.5
Fixed-Think (k=5)	50.9 ±0.6	330±2.6	7.65±0.4
Heuristic-Think	55.6 ±1.1	862±2.7	7.55±0.3
Fixed-Think (k=3)	59.9 ±1.3	580±2.2	7.34±0.2
RARRL (ours)	**79.6 ±0.8**	614±2.6	6.49±0.3

because the environments are similar, but because the orchestration problem has the same abstract form regardless of the specific task or simulator.

G Evaluation on ALFWorld and VirtualHome

G.1 Evaluation on ALFWorld

We further evaluate whether the ADE-trained orchestration policy transfers zero-shot to ALFWorld (Shridhar et al., 2020), a text-based household task benchmark with six task categories. We serve Llama-3.1-8B-Instruct (Grattafiori et al., 2024) locally via vLLM as the reasoning backend. Unlike the visual embodied environments, ALFWorld provides textual observations; accordingly, both planning and verification operate on the textual task state.

Transfer gap. Table 11 reports the zero-shot transfer result over all 3,553 ALFWorld episodes. The ADE-trained policy achieves 79.6% TSR on ALFWorld without any retraining or parameter update, compared with 80.9% in the ADE, corresponding to a 1.3 point absolute transfer gap. The transfer is evaluated under a different environment interface, task language, execution dynamics, and LLM backend: the ADE policy is trained without LLM calls, whereas ALFWorld uses Llama-3.1-8B-Instruct for reasoning. The small empirical gap provides evidence that the learned orchestration policy transfers across these changes in deployment conditions.

In Table 12, RARRL achieves the highest TSR among all compared methods.

G.2 Evaluation on VirtualHome

VirtualHome setup. We further evaluate the frozen RARRL policy on VirtualHome (Puig et al., 2018) using Qwen2.5-VL-72B-Instruct as the Planner and Verifier backend. In VirtualHome, 78 tasks (16 seen and 62 unseen) are paired with 20 distinct scenes (10 seen and 10 unseen), forming 445 demonstrations. We consider three task–scene generalization regimes: *Seen/Seen*, *Seen/Unseen*, and *Unseen/Unseen*, corresponding to the seen/unseen status of the task and scene, respectively. The frozen orchestration policy is evaluated without retraining across these regimes.

In Table 13, RARRL achieves the highest TSR among all compared methods across all three generalization regimes. Its advantage is preserved as the setting shifts from Seen/Seen to Seen/Unseen and Unseen/Unseen, while the reasoning cost remains lower than Always-Think and the success rate degrades more gracefully under task and scene distribution shifts.

Table 13: Performance on VirtualHome. Results are reported as mean ± 95% CI over 15 random seeds.

	Seen/Seen		Seen/Unseen		Unseen/Unseen	
Method	**TSR↑**	**PS↓**	**TSR↑**	**PS↓**	**TSR↑**	**PS↓**
Never-Think	60.2±1.2	20.1±1.7	51.0±1.5	21.3±2.1	37.2±1.1	24.2±1.4
Always-Think	55.1±1.7	32.4±1.9	46.0±1.3	33.7±1.4	31.0±1.6	36.1±1.4
Fixed-Think	65.4±2.0	24.8±1.5	56.7±1.6	25.6±1.8	43.1±1.4	28.2±1.3
Heuristic-Think	69.1±0.7	23.1±1.3	61.4±1.8	24.0±1.6	47.8±1.2	27.0±1.2
RARRL	**77.3±1.1**	**18.9±1.2**	**71.2±0.9**	**19.7±1.4**	**60.3±1.4**	**22.4±1.2**

H Environment Validation via Marginal Utility Analysis

A recurring risk when an interpretability result does not match a design intention is misattribution: treating an *environment* deficiency as an *optimization* artifact, or vice versa. We adopted a simple, reusable discipline to separate the two *before* modifying the environment: fix the latent context, enumerate candidate reasoning strategies, and measure each strategy's real return (Monte Carlo, using the actual dynamics, costs, and retry mechanics). We call this a marginal-utility analysis, and it drove two decisions in this work in opposite directions.

Case 1: a genuine environment deficiency (verifier). In an earlier, additive formulation of the transition—where the advance probability depended on the *count* of addressed facets rather than on independent risks—the marginal value of acquiring verification evidence in critical contexts was near zero and its sign depended on incidental modeling choices. Measuring the return of "planner only" versus "planner + verifier" across several backdrop policies gave inconsistent signs (e.g. -0.12, $+0.15$, -0.00), i.e. no robust incentive to acquire verification evidence. A policy that never learned to verify was therefore behaving near-optimally: the environment semantics, not the optimizer, were insufficient. This diagnosis motivated the independent-risk (multiplicative) transition used throughout the paper, under which planning and verification resolve distinct, non-substitutable risks.

Case 2: a genuine optimization artifact (planner in recovery). Under the redesigned dynamics, the trained policy still occasionally invokes the planner in recovery contexts. Here the same analysis gives the opposite verdict: fixing the context to recovery, the return of the correct strategy (verifier) is 2.36, whereas adding the planner *lowers* it to 1.82 (task success 72.9% vs. 63.4%). The environment already penalizes the extra planner call; the optimal strategy unambiguously excludes it. The residual planner use is therefore not explained by the local optimality of the Recovery context alone; it may reflect imperfect convergence or uncertainty over the latent context under partial observability.

Takeaway. We use marginal-utility analysis to distinguish environment deficiencies from optimization artifacts before modifying the environment. The principle—verify *where* the optimum lies before concluding the optimizer failed to reach it—prevented both an unnecessary environment change (Case 2) and the perpetuation of a real one (Case 1).

Sensitivity of the risk-survival constant. The multiplicative transition uses a base survival probability P_{lo} for an unresolved required risk (the survival probability for a resolved risk reuses the evidence-reliability curve of Sec. C and introduces no new constant). P_{lo} is an environment-instantiation parameter, not a method parameter. Table 14 sweeps it and reports the marginal value of acquiring additional evidence in critical contexts across the same backdrop policies used above. The marginal value is positive throughout $P_{\text{lo}} \in [0.30, 0.45]$, so the qualitative structure, each decision facet carrying stable, non-substitutable value, does not depend on a single tuned constant.

Table 14: Sensitivity of the verifier's marginal value in critical contexts to the base risk-survival constant P_{lo}. Entries are the change in return from adding verification (planner → planner+ verifier) under three backdrop policies. Positive throughout the range.

P_{lo}	planner-only bkgd	act-only bkgd	heuristic bkgd
0.30	$+0.424$	$+0.544$	$+0.540$
0.35	$+0.254$	$+0.420$	$+0.467$
0.40	$+0.150$	$+0.325$	$+0.407$
0.45	$+0.056$	$+0.242$	$+0.235$

Table 15: Execution-level analysis of RARRL's THINK events on EB-ALFRED across six task categories. "Recovered" indicates that the reasoning intervention successfully recovered the trajectory.

Category	Scenario	# THINK Events	Recovered	Recovery Rate
Base	Execution failure	24	9	37.5%
	State inconsistency	0	0	0.0%
	Missing sub-goal	5	1	20.0%
	Unclassified	0	0	0.0%
	Total	**29**	**10**	**34.5%**
Comm.	Execution failure	29	10	34.5%
	State inconsistency	1	1	100.0%
	Missing sub-goal	6	3	50.0%
	Unclassified	4	3	75.0%
	Total	**40**	**17**	**42.5%**
Complex	Execution failure	22	6	27.3%
	State inconsistency	0	0	0.0%
	Missing sub-goal	9	3	33.3%
	Unclassified	1	1	100.0%
	Total	**32**	**10**	**31.2%**
Visual	Execution failure	31	6	19.4%
	State inconsistency	1	0	0.0%
	Missing sub-goal	4	3	75.0%
	Unclassified	3	1	33.3%
	Total	**39**	**10**	**25.6%**
Spatial	Execution failure	34	11	32.4%
	State inconsistency	0	0	0.0%
	Missing sub-goal	2	2	100.0%
	Unclassified	2	2	100.0%
	Total	**38**	**15**	**39.5%**
Long	Execution failure	33	13	39.4%
	State inconsistency	0	0	0.0%
	Missing sub-goal	0	0	0.0%
	Other	1	0	0.0%
	Total	**34**	**13**	**38.2%**
Overall	**Total**	**212**	**75**	**35.4%**

I Error Analysis

Experimental setup. We analyze all THINK invocations produced by RARRL across the six capability categories on EB-ALFRED ($n = 212$ events) and EB-Habitat ($n = 237$ events). Each THINK event is classified by its primary triggering condition: *execution failure* if the immediately preceding action fails; *missing sub-goal* if THINK follows a successful action at intermediate task progress and indicates an incomplete sub-goal; and *state inconsistency* if the Verifier's visual recheck identifies a mismatch between the observed state and the tracked execution history. Events

Table 16: THINK-event analysis on EB-Habitat across six task categories. "Recovered" indicates that the reasoning intervention successfully recovered the trajectory.

Category	Scenario	# THINK Events	Recovered	Recovery Rate
Base	Execution failure	17	10	58.8%
	State inconsistency	3	3	100.0%
	Missing sub-goal	0	0	0.0%
	Other	0	0	0.0%
	Total	**20**	**13**	**65.0%**
Comm.	Execution failure	40	12	30.0%
	State inconsistency	5	2	40.0%
	Missing sub-goal	0	0	0.0%
	Other	0	0	0.0%
	Total	**45**	**14**	**31.1%**
Complex	Execution failure	36	22	61.1%
	State inconsistency	2	0	0.0%
	Missing sub-goal	0	0	0.0%
	Other	0	0	0.0%
	Total	**38**	**22**	**57.9%**
Visual	Execution failure	38	15	39.5%
	State inconsistency	5	1	20.0%
	Missing sub-goal	0	0	0.0%
	Other	1	1	100.0%
	Total	**44**	**17**	**38.6%**
Spatial	Execution failure	27	6	22.2%
	State inconsistency	17	1	5.9%
	Missing sub-goal	0	0	0.0%
	Other	0	0	0.0%
	Total	**44**	**7**	**15.9%**
Long	Execution failure	43	13	30.2%
	State inconsistency	3	1	33.3%
	Missing sub-goal	0	0	0.0%
	Other	0	0	0.0%
	Total	**46**	**14**	**30.4%**
Overall	**Total**	**237**	**87**	**36.7%**

that do not clearly satisfy any of these conditions are labeled *unclassified*; this reflects a limitation of the post-hoc classification rather than a distinct triggering mechanism used by the policy. *Recovery rate* is the fraction of THINK events in each category whose episode ultimately succeeds. This classification is post-hoc and diagnostic; it is not used by the policy at execution time.

THINK-event error analysis. Table 15 and Table 16 summarize the triggering conditions of THINK events on EB-ALFRED and EB-Habitat, respectively. On EB-ALFRED, 173/212 THINK events (81.6%) are triggered by execution failures, 26/212 (12.3%) occur after a successful action with intermediate task progress and correspond to missing sub-goals, 2/212 (0.9%) are associated with state inconsistencies, and the remaining 11/212 (5.2%) are unclassified. The missing-sub-goal cases achieve a recovery rate of 46.2%, compared with 31.8% for failure-triggered THINK events. On EB-Habitat, 201/237 THINK events (84.8%) are triggered by execution failures, 35/237 (14.8%) are associated with state inconsistencies, no events are attributed to missing sub-goals, and the remaining 1/237 (0.4%) is unclassified. Overall, these results show that RARRL does not rely on a single fixed trigger for reasoning. Instead, the learned orchestration policy invokes THINK under different execution conditions, including failure recovery, missing sub-goals, and state inconsistency, with the triggering patterns adapting to the execution context of each environment.

J Verifier Corruption and Ablation Analysis

Experimental setup. To directly test whether RARRL's gains depend on the correctness of individual verifier judgments, we rerun the same 50 EB-Habitat episodes (Base category) under identical seeds. The same policy, Planner, and environment dynamics are used in all three settings below; only the Verifier judgment supplied to the orchestration policy is overridden. To avoid introducing additional LLM stochasticity as a confound, none of the three settings re-invokes the real Verifier model; judgments are instead drawn from a pre-recorded pool obtained during the Original run.

Settings.

- **Original**: uses the verifier judgment recorded at each decision point during a normal RARRL rollout.
- **Shuffled**: replaces the judgment at each decision point with one drawn at random from the pool of judgments recorded across all 50 episodes.
- **Corrupted**: replaces the judgment at each decision point with its flipped counterpart (an originally approved candidate becomes rejected, and vice versa).

Because a substituted judgment can change the action taken at that step, the environment trajectory following a substitution may diverge from the recorded Original trajectory; we do not enforce an exact match in the number or timing of subsequent THINK invocations, and report the realized THINK counts for each setting directly.

Planner-only and non-informative Verifier ablations (EB-ALFRED). To further test the Verifier's contribution using an orthogonal form of intervention, we evaluate two additional variants on EB-ALFRED Base: *Planner-only*, where the Verifier step is skipped entirely and the Planner's proposed continuation is always accepted; and *Non-informative Verifier*, where the Verifier step is retained (and its computational cost still incurred) but its judgment is generated independently of the actual candidate or observation. Relative to full RARRL (TSR $= 62.8\%$), both variants show a significant reduction in TSR: Planner-only reaches 52.0% ($\Delta = -10.8$pp, McNemar's $p = 0.044$), and Non-informative Verifier reaches 50.0% ($\Delta = -12.8$pp, McNemar's $p = 0.045$).

Table 17: Verifier ablation and corruption results. Planner-only and Non-informative Verifier are evaluated on EB-ALFRED (Base, $n = 50$); Shuffled and Corrupted are evaluated on EB-Habitat (Base, $n = 50$). ΔTSR and p-values are computed via McNemar's test on paired episode outcomes relative to the corresponding Original/full RARRL condition on the same platform.

Platform	Setting	TSR (%)	ΔTSR	p-value
EB-ALFRED	RARRL (Original)	62.0	–	–
	Planner-only	52.0	-10.0	0.044
	Non-informative Verifier	50.0	-12.0	0.045
EB-Habitat	RARRL (Original)	86.0	–	–
	Shuffled	72.0	-14.0	0.04
	Corrupted	70.0	-16.0	0.05

Statistical test. In Table 17, for the McNemar comparisons, the pairing unit is the benchmark test episode rather than the execution trajectory. Each condition is evaluated on the same episode identifier, task specification, and initial environment state. The Original and Altered verifier conditions are then rolled out independently, and their trajectories may diverge after the verifier judgment is applied. Each rollout produces one binary episode-level outcome, indicating task success or failure. We apply McNemar's exact test to these matched episode-level outcomes, which is appropriate because the comparisons are paired at the episode level rather than based on independent samples. Only discordant pairs, namely episodes whose outcome differs between the two settings, contribute to the test statistic.

Interpretation. Unlike an earlier, smaller-scale version of this analysis, both the EB-Habitat corruption results and the EB-ALFRED ablation results show a significant reduction in TSR relative to

the corresponding Original/full RARRL condition (all $p < 0.05$). Together, these results indicate that the Verifier makes a measurable, statistically significant contribution to RARRL's performance. The two analyses probe different aspects of this contribution: corrupting or shuffling the recorded judgment on EB-Habitat tests whether the *content* of the Verifier's judgment matters, while removing or replacing the Verifier step on EB-ALFRED tests whether its *presence* and *informativeness* matter. Both indicate that the Verifier contributes beyond serving as a mere delay or a trigger for Planner re-invocation, rather than being a component whose correctness is largely inconsequential to task success.

K State-Dependent Orchestration Analysis: Full Statistics

Method. We pool policy decision points (mode $\in \{\texttt{ACT}, \texttt{THINK}\}$) across all six EB-ALFRED capability categories, together with the corresponding abstract state $z_t = (p_t, h_t, b_t, u_t, e_{\text{plan}}, \kappa_{\text{plan}})$ recorded at each decision point. Decision points where the correction budget is exhausted ($b_t \leq 0.05$) are excluded from this analysis, since THINK is mechanically unavailable once the budget is depleted and including these points would conflate a hard architectural constraint with the policy's learned behavior. This excludes 1,087 of 2,745 pooled decision points, leaving $n = 1{,}658$. For p_t, b_t, and u_t, we group decision points into quantile-based bins (approximately equal sample size per bin) rather than equal-width bins, since these variables are unevenly distributed across their range. For h_t (progress stagnation), which takes a small number of recurring values by construction (Sec. 3.3) rather than a continuous distribution, we report the empirical THINK rate at each observed value with at least 15 decision points, rather than imposing an artificial binning scheme. For each condition, we report the empirical THINK rate together with the Wilson score 95% confidence interval, which remains well-behaved at small sample sizes and extreme proportions (unlike the normal approximation).

Table 18: Empirical THINK rate conditioned on individual state variables, pooled across all six EB-ALFRED categories (decision points with the correction budget exhausted are excluded, since THINK is mechanically unavailable in that regime), including 95% Wilson confidence intervals.

State variable	Condition	THINK / Total	Rate	95% CI
Progress p_t	$[0, 0.3]$	894/1300	68.8%	[66.2%, 71.2%]
	$[0.3, 0.7]$	268/358	74.9%	[70.1%, 79.1%]
Stagnation h_t	0.00	0/345	0.0%	[0.0%, 1.1%]
	0.22	18/18	100.0%	[82.4%, 100.0%]
	0.48	12/16	75.0%	[50.5%, 89.8%]
	0.67	16/16	100.0%	[80.6%, 100.0%]
	0.81	18/19	94.7%	[75.4%, 99.1%]
	1.00	988/1110	89.0%	[87.0%, 90.7%]
Budget b_t	$[0.33, 0.67]$	362/397	91.2%	[88.0%, 93.6%]
	$[0.67, 1.0]$	800/1261	63.4%	[60.7%, 66.1%]
Uncertainty u_t	$[0, 0.25]$	182/507	35.9%	[31.8%, 40.2%]
	$[0.25, 0.5]$	680/730	93.2%	[91.1%, 94.8%]
	$[0.5, 1.0]$	300/421	71.3%	[66.8%, 75.4%]

Interpretation. The strength of evidence differs across the four state variables. Progress stagnation h_t shows the clearest association: the rate at $h_t = 0$ (0.0%, $n = 345$) and at $h_t = 1.0$ (89.0%, $n = 1{,}110$) are both estimated from large samples with non-overlapping confidence intervals, providing strong empirical evidence of state-dependent THINK allocation with respect to this signal. Uncertainty u_t and remaining budget b_t also show substantial variation across conditions, although the pattern is non-monotonic and, for b_t, runs counter to the intuition that more remaining budget should be associated with more frequent reasoning; we do not attempt to explain this direction here and report it as observed. Progress p_t shows a smaller difference between conditions (68.8% vs. 74.9%) and provides comparatively weaker evidence of state dependence specific to this variable. Taken together, these results indicate that the policy's orchestration decisions are not explained by task category alone, but they remain an observational, state-conditioned analysis rather than a causal counterfactual test of whether invoking THINK improves outcomes at any specific state.

L Detailed Decision Rollout

Figure 4 provides a step-by-step expansion of the EB-Habitat Episode 49 trajectory shown in Figure 3 (main text), including the policy's action probabilities $\pi(\texttt{ACT})$ and $\pi(\texttt{THINK})$, the observed state z_t, the Planner's proposed candidate, and the Verifier's judgment at each decision point.



Figure 4: Complete decision rollout for EB-Habitat Episode 49 ("On the sofa there's an apple, but instead find a screwdriver and move it to the left counter."), showing the abstract state z_t, policy action probabilities, Planner/Verifier outputs, and Persistent Plan Queue updates at each of the seven decision steps.

M Comparison with a Non-RL Learned Trigger

Motivation. To isolate whether RARRL's gains arise simply from learning a state-conditioned reasoning trigger, we introduce SC-MLP as a targeted control baseline. SC-MLP receives the same observable abstract state z_t used by RARRL and predicts whether to execute ACT or invoke THINK. Unlike RARRL, SC-MLP does not use sequential policy optimization or PPO. This comparison therefore tests whether a simple non-RL state-conditioned trigger can reproduce the performance of RARRL.

Training. SC-MLP is trained only on the Abstract Decision Environment (ADE). Rather than optimizing a sequential policy through PPO, we construct supervised training targets using the ADE-derived analytic teacher. For each observed abstract state z_t, the teacher evaluates the available

Table 19: Targeted comparison with SC-MLP, a non-RL learned state-conditioned trigger. SC-MLP receives the same abstract state z_t as RARRL but is trained by supervised imitation of an ADE-derived analytic teacher, without sequential policy optimization. Both methods are trained only on ADE and evaluated zero-shot across the six capability categories.

	Base	Common	Complex	Visual	Spatial	Long	Avg.
EB-ALFRED							
SC-MLP (no PPO)	52.0±2.3	46.0±1.2	48.0±1.5	42.0±2.3	41.0±2.5	50.0±2.6	46.5
RARRL (Ours)	**62.8±2.2**	**54.5±2.2**	**56.2±2.2**	**48.6±2.1**	**52.4±2.2**	**61.9±2.2**	**55.97**
EB-Habitat							
SC-MLP (no PPO)	74.0±2.2	41.0±2.6	52.0±1.9	52.0±1.2	30.0±2.2	38.0±1.7	47.8
RARRL (Ours)	**86.6±1.7**	**48.2±2.1**	**64.5±2.1**	**60.3±2.1**	**36.7±1.9**	**52.4±2.1**	**58.12**

decision choices under the ADE transition and resource-cost model and provides the corresponding preferred decision. SC-MLP is then trained to imitate these teacher decisions using a multilayer perceptron. The target platform is not used during SC-MLP training.

Importantly, SC-MLP and RARRL operate on the same abstract state interface. Thus, this comparison does not give SC-MLP access to a different class of execution signals; the primary distinction is that SC-MLP learns a direct state-conditioned trigger through supervised imitation, whereas RARRL learns a sequential orchestration policy through PPO.

Zero-shot evaluation. After training on ADE, the SC-MLP parameters are fixed and transferred to the target embodied environments without target-environment training. At each orchestration decision point, the current abstract state z_t is provided to SC-MLP, which selects the corresponding `ACT` or `THINK` decision. The downstream Planner, Verifier, execution mechanism, and other components remain unchanged from the corresponding RARRL evaluation setting.

We evaluate SC-MLP across all six EB-ALFRED and EB-Habitat capability categories. Each category contains 50 evaluation episodes. The results are compared with the corresponding RARRL results in Table 19.

Results. RARRL consistently matches or outperforms SC-MLP across all six capability categories on both embodied platforms. On EB-ALFRED, RARRL achieves an average TSR of 56.0%, compared with 46.5% for SC-MLP, corresponding to a 9.5 percentage-point improvement. On EB-Habitat, the corresponding averages are 58.1% and 47.8%, yielding a 10.3 percentage-point improvement.

At the individual category level, RARRL outperforms SC-MLP in all 12 platform–category comparisons. The gains are particularly pronounced for Base, Spatial, and Long on EB-ALFRED and for Base, Complex, Visual, and Long on EB-Habitat. Even in categories where the difference is small, RARRL does not underperform the non-RL learned trigger.

These results provide a targeted control for the role of sequential policy optimization. Because SC-MLP receives the same abstract state and is trained on the same ADE while removing PPO-based sequential optimization, its lower performance suggests that RARRL's advantage is not explained solely by learning a nonlinear state-conditioned ACT/THINK trigger. Instead, the results support the value of learning the orchestration policy through sequential resource-aware optimization.

N Additional Baseline and Control Analysis

This section provides additional control experiments to disentangle the contributions of persistent-plan execution, reasoning-call frequency, trigger design, and ADE-trained orchestration. These experiments are complementary to the capability-wise comparison in Table 1. In particular, the original baseline comparison evaluates different reasoning invocation paradigms, whereas the present analysis further controls the execution infrastructure and the training environment.

Table 20: Control experiments for isolating the effects of persistent-plan execution, THINK-call frequency, trigger design, and ADE-trained orchestration. All methods use the same Planner, Verifier, action executor, and budget accounting unless otherwise specified. Results are reported as mean ± 95% CI over 15 random seeds.

	EB-ALFRED		EB-Habitat		Overall	
Method	**TSR↑**	**PS↓**	**TSR↑**	**PS↓**	**TSR↑**	**THINK↓**
Queue and trigger controls						
Random trigger	44.1 ± 1.1	18.76	44.8 ± 1.5	20.42	44.5 ± 0.8	2.2
Fixed trigger	47.8 ± 0.9	18.31	49.2 ± 1.3	20.05	48.5 ± 1.3	2.6
Heuristic trigger	50.2 ± 1.4	18.14	49.7 ± 1.2	19.92	49.9 ± 0.9	2.9
Training and transfer controls						
Target-trained policy on z_t	44.8 ± 1.1	17.96	47.2 ± 1.2	19.64	46.0 ± 1.4	2.1
RARRL (Ours)	$\mathbf{55.97 \pm 1.7}$	**17.87**	$\mathbf{58.12 \pm 1.9}$	**16.14**	$\mathbf{57.04 \pm 0.8}$	**2.1**

N.1 Baseline Definitions

We compare RARRL with the following baseline strategies.

Random trigger. The random-trigger baseline samples THINK decisions randomly while matching the average number of THINK invocations made by RARRL. The matching is performed under the same episode budget and the same shared-queue execution mechanism. This baseline tests whether the performance of RARRL can be explained solely by the number of reasoning invocations, rather than by the state-dependent placement of those invocations.

Fixed trigger. The fixed-trigger baseline invokes THINK every K eligible orchestration decision points. We evaluate the candidate intervals

$$K \in \{1, 2, 3, 5, 10\}$$

on a held-out validation set under the same THINK-call budget as RARRL. Among these candidates, $K = 3$ achieves the best validation performance and is therefore selected for test evaluation. The selected value is kept fixed throughout testing. This baseline uses the same Planner, Verifier, persistent queue, action executor, and budget accounting as RARRL, but replaces the learned state-dependent trigger with a periodic schedule.

Heuristic trigger. The heuristic-trigger baseline invokes THINK using a manually specified threshold rule over the same agent-observable execution-progress signal available to RARRL. Let $g(s_t)$ denote this progress signal. THINK is triggered when the prescribed condition $g(s_t) \leq \tau$ is satisfied, where the threshold τ is selected on a held-out validation set under the same THINK-call or computational budget constraint as RARRL. The selected threshold is fixed before test evaluation. This baseline uses the same Planner, Verifier, persistent-queue mechanism, action executor, and budget accounting as RARRL, while replacing the learned orchestration policy with a manually specified, progress-based rule.

Target-trained policy. The target-trained policy has the same policy architecture, input state z_t, action space, and PPO optimization procedure as RARRL. However, it is trained directly in the target embodied environment rather than in the Abstract Decision Environment (ADE). This baseline tests whether the performance of RARRL is attributable to the orchestration architecture itself or to its zero-shot transfer from ADE.

RARRL. RARRL uses the learned orchestration policy trained exclusively in ADE. At deployment, the policy observes the six-dimensional abstract decision state and selects between ACT and THINK according to the current execution context, task progress, execution history, and remaining resource budget. The policy is frozen during evaluation and is not adapted to either EB-ALFRED or EB-Habitat.

N.2 Interpretation of the Control Experiments

Table 20 should be interpreted along four dimensions.

Effect of THINK-call frequency. The random-trigger baseline is matched to RARRL in its average number of THINK invocations. If RARRL achieves higher TSR than the random-trigger baseline under comparable reasoning frequency and budget usage, the difference cannot be attributed solely to invoking THINK more often. Instead, it supports the importance of selecting appropriate execution-time decision points.

Effect of trigger design. The fixed-trigger and heuristic-trigger baselines test whether a simple manually designed scheduling rule is sufficient. The fixed trigger uses a periodic schedule, while the heuristic trigger uses an observable progress-based condition. RARRL differs from both baselines by learning the trigger policy from the abstract decision state and by conditioning its decisions on execution history, uncertainty feedback, and remaining budget.

Effect of ADE training and zero-shot transfer. The target-trained policy uses the same six-dimensional state and policy architecture as RARRL, but receives training directly in the target environment. Comparing this policy with RARRL isolates the effect of the training environment and tests whether ADE-trained orchestration can transfer beyond the environment in which the policy was optimized.

N.3 Discussion

The results in Table 20 show that the control strategies differ in both task performance and reasoning behavior. effect of persistent-plan execution independently of THINK invocation. The random-trigger baseline further controls for the number of reasoning calls, allowing us to distinguish reasoning frequency from reasoning placement. The fixed and heuristic triggers provide increasingly stronger manually designed alternatives, but neither explicitly optimizes the long-term trade-off between task progress and reasoning cost.

The target-trained policy provides an additional reference for assessing the role of ADE. It uses the same abstract state and policy architecture as RARRL, but is trained directly in the target environment. RARRL achieves higher TSR on both EB-ALFRED and EB-Habitat under the same evaluation protocol (Table 20). This comparison shows that the observed performance does not require target-environment training of the orchestration policy.

O Additional Analysis of State-Dependent Orchestration

Motivation. The central purpose of the orchestration policy is to learn when additional reasoning is warranted, rather than to invoke the reasoning module according to a manually specified rule. To examine whether the learned THINK/ACT decisions depend on the observed decision state beyond task-category effects, we conduct an auxiliary multivariate analysis of the policy decisions recorded during evaluation.

In particular, we model the binary decision $d_t \in \{\text{ACT}, \text{THINK}\}$ as a function of the observed state variables, queue status, and task category. The analysis is intended to characterize the statistical structure of the learned orchestration behavior. It is not intended to estimate the causal utility of THINK relative to ACT.

O.1 Average Marginal Effects

We use the decision-level logs collected from the EB-ALFRED and EB-Habitat evaluation runs and exclude decision points where the correction budget is exhausted ($b_t \leq 0.05$), yielding 2038 orchestration decision points, with each observation corresponding to one policy decision. We compute average marginal effects for the standardized continuous state variables, reported in Table 21. The estimated average changes in the probability of selecting THINK associated with a one-unit increase in each standardized variable are

The average marginal effects are consistent with the odds-ratio analysis: the largest changes in the predicted probability of selecting THINK are associated with b_t and h_t, followed by p_t. The estimated average marginal effect of u_t is close to zero in this reduced model.

Table 21: Average marginal effects in the reduced logistic model.

Variable	Average marginal effect
p_t	$+0.0570$
h_t	$+0.2615$
b_t	$+0.3111$
u_t	$+0.0019$

Table 22: Task success rates (TSR, %) on EB-ALFRED for different THINK invocation strategies with Gemma-3-27B-it as the evaluation MLLM, assessing the robustness of the frozen RARRL policy to a different reasoning backbone. Results are reported as mean ± 95% CI over 15 random seeds.

Method	Base	Common	Complex	Visual	Spatial	Long Horizon	Avg.
Never-Think	42±1.8	40±2.1	48±2.7	30±2.5	36±1.3	26±2.4	37.00
Always-Think	43±1.9	42±2.5	50±1.9	31±2.0	40±2.3	35±2.6	40.17
Fixed-Think ($k = 3$)	44±1.5	43±1.7	53±2.7	32±2.4	42±2.1	34±2.6	41.33
Heuristic-Think	44 ±2.2	43±2.5	52±1.6	32±1.3	41±1.7	34±1.2	41.00
RARRL	**45±2.2**	**44±2.3**	**55±1.2**	**34±2.4**	**43±1.4**	**37±1.1**	**43.00**

These quantities should be interpreted as model-based changes in the predicted decision probability, not as experimentally measured changes in task success or execution quality.

O.2 Interpretation and Limitations

The analysis provides evidence that the learned orchestration decisions are associated with the observed decision state beyond task-category labels. In particular, h_t and b_t remain strongly associated with THINK selection in the reduced specification, even after controlling for queue status and task category.

However, three limitations are important.

First, the exclusion of `preceding_failure` is required to obtain a numerically stable ordinary logistic regression, but it also means that the reduced model does not condition on failure history. The coefficients of h_t and b_t may therefore partly absorb information related to the omitted failure-history variable.

Second, the analysis is observational and describes conditional associations in the logged policy decisions. It does not identify causal effects of the state variables on the decision, nor does it establish that THINK causally improves task success relative to ACT.

Third, the strong imbalance for `queue_active` limits the extent to which its coefficient can be generalized. A more complete analysis of the role of failure history would require a separation-aware estimator, such as a penalized or bias-reduced logistic regression, together with a careful audit of the temporal alignment between execution outcomes and subsequent policy decisions.

Accordingly, the present analysis should be viewed as an auxiliary audit of state-dependent orchestration. It supports the claim that the policy's decisions are not adequately characterized by task category alone, but it does not by itself establish the causal utility of invoking the reasoning module.

P Robustness to the Reasoning Backend

To examine whether RARRL generalizes beyond Qwen2.5-VL-72B-Instruct, we replace its Planner and Verifier with the smaller, different-family Gemma-3-27B-it (Team et al., 2025) while keeping the orchestration policy frozen. Since e_t^{plan} and κ_t^{plan} are generated by this backend, the experiment tests whether RARRL depends on a specific MLLM's reliability profile.

On EB-ALFRED (Table 22), RARRL maintains its advantage across all six capability categories, achieving 43.0% average TSR versus 37.0–41.3% for alternative strategies. On EB-Habitat (Ta-

Table 23: Task success rates (TSR, %) on EB-Habitat for different THINK invocation strategies with Gemma-3-27B-it as the evaluation MLLM, assessing the robustness of the frozen RARRL policy to a different reasoning backbone. Results are reported as mean ± 95% CI over 15 random seeds.

Method	Base	Common	Complex	Visual	Spatial	Long Horizon	Avg.
Never-Think	68±1.4	26±1.5	30±2.4	40±1.8	28±1.6	22±2.5	35.67
Always-Think	64±1.5	23±1.3	25±1.8	37±2.4	24±1.9	18±2.4	31.83
Fixed-Think ($k = 3$)	67±2.3	25±2.4	29±1.6	39±2.4	27±1.8	22±1.7	34.83
Heuristic-Think	67±2.5	26±1.1	31±2.5	39±2.2	29±2.6	24±1.9	36.00
RARRL	**73±2.2**	**31±1.8**	**34±2.4**	**43±1.9**	**35±2.4**	**29±1.5**	**40.83**

ble 23), Always-Think underperforms Never-Think (31.83% vs. 35.67%), whereas RARRL achieves the highest TSR of 40.83%. These results suggest that RARRL's benefit comes from selectively deciding WHEN to reason rather than simply reasoning more.

Q ADE MISMATCH STRESS TEST

To examine whether the learned orchestration policy relies on ADE-specific inductive biases, we construct a mismatched ADE in which the evidence-composition mechanism is deliberately changed while preserving the remaining decision structure. In the original ADE, the advancement probability for a context with required facets $\mathcal{F}$ is defined as

$$p_{\text{adv}} = \prod_{f \in \mathcal{F}} p_f, \tag{8}$$

where p_f denotes the advancement probability associated with facet f. We construct a mismatched ADE by replacing Eq. equation 8 with a weakest-link aggregation:

$$p_{\text{adv}} = \min_{f \in \mathcal{F}} p_f. \tag{9}$$

We use weakest-link aggregation as a deliberately contrasting alternative in which advancement is determined by the least-satisfied required facet, while preserving the same facet structure and evidence acquisition mechanism. This modification therefore changes only the evidence-composition functional form. The context prior, required-facet sets, reasoning cost, reasoning budget, observation space, action space, and PPO training procedure remain unchanged. Thus, the two ADEs share the same decision interface while inducing different relationships between evidence and task advancement.

We train two policies independently: π_{Prod} on the original ADE and π_{Min} on the mismatched ADE. After training, both policies are frozen and evaluated zero-shot on the identical EB-ALFRED test set using the same deployment pipeline, reasoning backend, and resource budget. We report the resulting transfer performance in Table 25.

Table 24 first illustrates the evidence-composition difference itself with a concrete, hypothetical example: for a Recovery context (which requires the Verifier and Reflection facets), it shows how the two aggregation rules compute p_{adv} differently before and after one facet is addressed by THINK. This is a worked numerical example of Eq. equation 8 and Eq. equation 9, not an empirical measurement, so no confidence interval applies.

Table 25 then reports the actual zero-shot transfer performance of the two frozen policies deployed on EB-ALFRED, which is the empirical evidence for whether this evidence-composition difference affects transfer.

This result suggests that the zero-shot transfer of RARRL is not tightly coupled to the specific multiplicative evidence-composition rule used by the original ADE. In particular, the policy trained under the mismatched weakest-link structure retains comparable target-task performance despite being exposed to a different relationship between multi-facet evidence and task advancement during training.

Table 24: Worked example of the evidence-composition difference for a Recovery context, which requires the Verifier and Reflection facets. Under the original ADE, facet-level advancement probabilities are multiplied; under the mismatched ADE, they are combined via a weakest-link (minimum) rule. With the training configuration $q_{\text{noise}} = 0$, an addressed facet has $p_f = 1.0$, while an unaddressed facet has $p_f = 0.35$. This is a hypothetical illustration of Eq. equation 8 and Eq. equation 9, not an empirical result.

ADE	**Before THINK**	**After THINK**	Δp_{adv}
Original (Product)	$0.35 \times 0.35 = 0.1225$	$0.35 \times 1.0 = 0.35$	$+0.2275$
Mismatch (Weakest-link)	$\min(0.35, 0.35) = 0.35$	$\min(0.35, 1.0) = 0.35$	0

Table 25: Transfer performance under ADE mismatch. Both π_{Prod} (trained on the original, product-aggregation ADE) and π_{Min} (trained on the mismatched, weakest-link-aggregation ADE) are frozen after training and evaluated zero-shot on the identical EB-ALFRED test set, using the same deployment pipeline, reasoning backend, and resource budget. Results are reported as mean ± 95% CI over 15 random seeds., a smaller-scale diagnostic protocol distinct from the 15-seed main results reported in Table 1.

ADE (training source)	**EB-ALFRED TSR (%)**
Original (Product)	$\mathbf{56.12} \pm 1.3$
Mismatch (Weakest-link)	$\mathbf{54.15} \pm 1.6$