Re^2: Reflective Rule Induction and Rule-Guided Refinement for Embodied Planning

Yang Chen[1,2] Hong-Jie You[1,3] Jie-Jing Shao[1] Xiao-Wen Yang[1,3]
Ming Yang[1,3] Yu-Feng Li[1,3] Lan-Zhe Guo[1,2] *
[1]State Key Laboratory of Novel Software Technology, Nanjing University, China
[2]School of Intelligence Science and Technology, Nanjing University, China
[3]School of Artificial Intelligence, Nanjing University, China
{chenyang, guolz}@lamda.nju.edu.cn

Abstract

*Embodied planning requires agents to translate high-level language instructions into executable behavior under physical, logical, and temporal constraints. Although recent LLM- and VLM-based agents have shown promising reasoning abilities, their performance in embodied environments remains limited by partial observability, evolving world states, and strict action preconditions, making semantically plausible plans often fail during execution. Moreover, interaction feedback is typically used only for local correction rather than accumulated as reusable procedural knowledge for future decisions. To address this limitation, we propose Re^2, a closed-loop framework for test-time improvement in embodied planning through interaction-driven procedural memory. Re^2 converts execution feedback into reusable rules, selectively reactivates task-relevant knowledge for subsequent planning, and continuously refines rule utility based on downstream outcomes. Each rule is represented in dual form, combining natural-language guidance with symbolic structure for retrieval and refinement. We evaluate Re^2 in both text-based and multimodal embodied planning settings. Re^2 achieves a state-of-the-art overall score of **81.36** on EAI and improves the average performance on EB-Habitat from **33.0** to **51.3** (+55.5%) over the reflective baseline. These results show that closed-loop procedural memory provides an effective mechanism for improving executability and robustness in embodied planning.*

1. Introduction

Recent advances in large language models (LLMs) and vision-language models (VLMs) have improved the reasoning capabilities of embodied agents [3, 6]. However, embodied planning is different from open-ended language reasoning [5, 30, 31]. To act successfully in the physical world, an agent must generate actions that are not only semantically plausible, but also consistent with environment dynamics, action preconditions, and temporal dependencies [21, 25]. As a result, plans that appear reasonable in latent reasoning may still fail during execution.

This difficulty is amplified by partial observability and constantly changing world states. An embodied agent must ground language instructions in its current observations, track object states and spatial relations, and coordinate multi-step actions in the correct order [17, 32]. For example, it may need to open a cabinet before retrieving an item, navigate to a target location before placing an object, or recover from a failed grasp before continuing a task. These challenges make robust embodied planning substantially harder than pure language reasoning.

Existing methods still struggle to acquire and reuse such constraints reliably across episodes. Prompt-based planning often overlooks environment-specific executability [1, 13]. Feedback-driven replanning improves local recovery, but typically treats failure as a sample-specific repair signal rather than an opportunity to distill reusable procedural knowledge [22, 26, 33, 37]. Neuro-symbolic methods introduce explicit constraints or structured world models [9, 38], and some test-time approaches further inject reflected rules during inference [11, 20, 23]. Yet our results show that direct rule injection alone is not sufficient for multimodal embodied planning: while stronger models can benefit from additional guidance, smaller VLMs may instead be destabilized by long and unfiltered rule memory. This exposes a more fundamental challenge: embodied agents need a closed-loop mechanism that can transform trajectory feedback into compact, task-relevant, and utility-grounded procedural memory, so that accumulated experience improves

*Corresponding author: guolz@lamda.nju.edu.cn

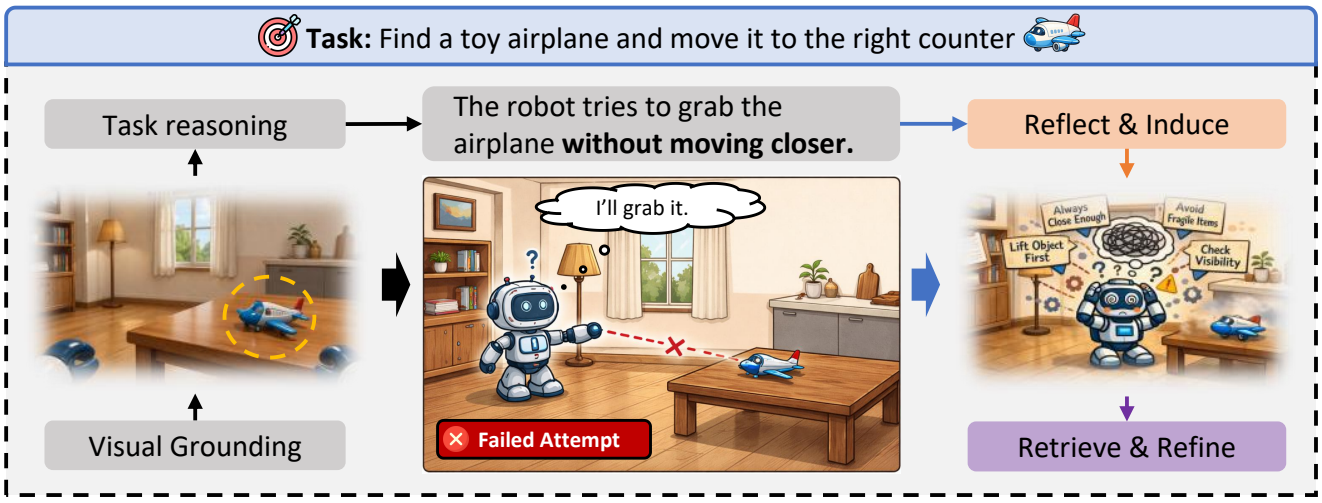


Figure 1. Interaction-driven procedural memory for embodied planning. To complete embodied tasks, agents must ground task instructions in observations while satisfying implicit embodied constraints. However, excessive rule injection can confuse the planner and destabilize decision making. Re^2 addresses this challenge through a closed loop of reflection, rule induction, and retrieve-and-refine planning, converting transient interaction feedback into compact and reusable procedural memory.

planning rather than overwhelming it.

To address this challenge, we propose Re^2, a closed-loop framework for test-time improvement in embodied planning through interaction-driven procedural memory, as illustrated in Fig. 1. Instead of treating reflected rules as static auxiliary prompts, Re^2 turns execution feedback into an evolving external memory that distills reusable procedural knowledge from experience, selectively activates rules according to the current task context, and continuously revises their utility based on downstream outcomes. Concretely, Re_1 abstracts reusable rules from successful and failed trajectories, while Re_2 grounds planning in a small set of task-relevant rules and updates them according to their actual contribution to execution. Each rule is represented in dual form, combining a natural-language description for prompt augmentation with a symbolic constraint for retrieval, attribution, and structural refinement. Together, these two stages transform transient interaction feedback into an external procedural memory that can be expanded, consolidated, and reused across episodes.

To verify that this closed-loop mechanism is not tied to a specific observation modality, we instantiate Re^2 in two representative embodied planning settings that differ in how the environment is perceived and grounded. In EAI [17], the framework operates primarily over textual states and symbolic execution feedback. In EB-Habitat [32], the same principle extends to multimodal grounded planning, where rules are induced and applied with respect to visual observations, spatial context, and interaction history. Despite these differences, Re^2 maintains the same core objective: converting execution feedback into reusable procedural memory for test-time improvement. Empirically, Re^2 achieves a state-of-the-art overall score of **81.36** on EAI. On EB-Habitat, it improves the average performance across from **33.0** to **51.3**, yielding a **+18.3** absolute gain and a **55.5%** relative improvement over the reflective baseline. These results show that closed-loop procedural memory provides an effective and modality-robust mechanism for improving executability and robustness in embodied planning.

Our contributions are summarized as follows:

- We propose Re^2, a closed-loop embodied planning framework that converts execution feedback into reusable procedural memory for test-time improvement.
- We develop a dual-form rule memory that combines natural-language and symbolic representations, together with utility-aware mechanisms for selective retrieval, refinement, and continual consolidation.
- We demonstrate the effectiveness of Re^2 across two embodied planning settings with different grounding modalities, achieving a state-of-the-art overall score of **81.36** on EAI and improving the EB-Habitat average from **33.0** to **51.3** (**+55.5%**) over the reflective baseline.

2. Background and Problem Formulation

We study task-oriented embodied planning, where an agent interprets a natural-language instruction, reasons over environment constraints, and executes a sequence of actions to reach a goal-satisfying state. Despite differences in observation modality and environment interface, embodied planning generally requires the agent to jointly reason about task semantics, object states, spatial relations, action preconditions, and temporal dependencies in order to produce exe-

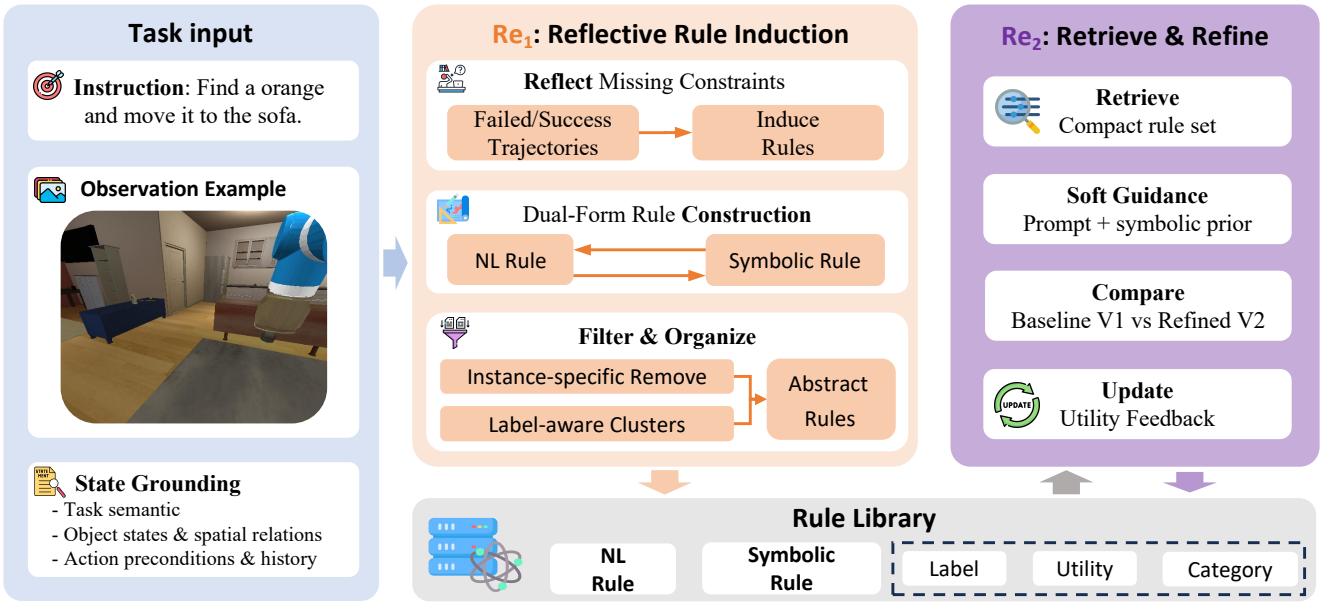


Figure 2. Overview of the Re^2 framework. Re^2 operates in a closed loop: Re_1 induces reusable procedural rules from embodied interaction trajectories and writes them into an external rule memory, while Re_2 retrieves task-relevant rules for plan refinement and updates their utility according to execution outcomes.

cutable behavior.

We formulate embodied planning as a grounded sequential decision process. Given a task instruction x, an observation sequence $o_{\leq t}$, an interaction history $h_{<t}$, and an action space $\mathcal{A}$, the agent first constructs a grounded state

$$s_t = \mathcal{G}(x, o_{\leq t}, h_{<t}), \tag{1}$$

where $\mathcal{G}(\cdot)$ denotes a modality-dependent grounding function that maps language, observations, and prior interactions into a planning-relevant state representation. Based on the grounded state s_t, the agent selects an action $a_t \in \mathcal{A}$ and interacts with the environment, yielding a trajectory

$$\tau = \{(o_t, a_t, f_t)\}_{t=1}^{T}, \tag{2}$$

where f_t denotes environment feedback at step t. The objective is to reach a terminal state that satisfies the task goal.

This formulation abstracts over both symbolic and perceptually grounded embodied environments. In some settings, the grounded state is constructed primarily from textual observations and structured execution feedback; in others, it must additionally incorporate visual observations, spatial context, and interaction history under partial observability. Regardless of modality, the central challenge remains the same: the agent must convert high-level task intent into causally valid and executable action sequences under environment-specific constraints. Under this unified formulation, our goal is to induce reusable procedural rules from interaction trajectories and maintain them as external memory for subsequent planning. Rather than treating execution feedback as an episode-specific repair signal, we seek to transform it into persistent procedural knowledge that can guide future decisions, improve executability, and support continual test-time refinement.

3. Method

We propose the Re^2 framework, a closed-loop approach for test-time improvement in embodied planning. Rather than treating execution feedback as a one-off repair signal, Re^2 converts interaction experience into an external procedural memory that can be expanded, updated, and reused across episodes. The framework consists of two coupled stages: Re_1, which induces reusable procedural rules from interaction trajectories, and Re_2, which retrieves and updates these rules as adaptive guidance for test-time plan refinement. The overall framework of Re^2 is illustrated in Fig. 2. The same mechanism applies to both text-only and multimodal embodied environments, while the source of grounded evidence differs across settings.

Formally, Re^2 maintains an external rule library

$$\mathcal{R} = \{r_i\}_{i=1}^{N}, \tag{3}$$

where each rule r_i stores reusable procedural knowledge induced from past interactions. The framework operates in a recurrent loop of *state grounding* $\rightarrow$ *rule induction* $\rightarrow$ *rule retrieval* $\rightarrow$ *plan refinement* $\rightarrow$ *memory updating*. Through this iterative process, transient interaction experi-

ence is gradually consolidated into reusable long-term procedural memory, thereby enabling interaction-driven memory expansion at test time.

3.1. Re_1: Reflective Rule Induction

The goal of Re_1 is to induce stable and transferable procedural knowledge from embodied interactions and to organize it into structured rules for subsequent planning. During this stage, the agent executes tasks without external rule injection and obtains a trajectory τ as in Eq. 2, where each step records observations, executed actions, and environment feedback. Re_1 then performs reflective analysis over these trajectories, identifying missing constraints from failure cases and extracting recurrent high-level operation patterns from successful cases. The former captures invalid state transitions, missing preconditions, and incorrect temporal dependencies, while the latter summarizes reusable procedural templates that recur across tasks and environments.

Unlike methods that induce rules solely from executor-side textual feedback, Re_1 reasons over the grounded state representation s_t, allowing task semantics, environment states, and execution outcomes to be analyzed in a shared space. In EAI, this process relies primarily on textual observations, symbolic feedback, and execution traces. In EB-Habitat, it is additionally grounded in visual observations that expose object locations, receptacle states, spatial relations, and interaction affordances. This enables the system to localize failure causes directly from perception rather than inferring them only from textual error messages. As a result, Re_1 treats interaction not merely as a source of local correction, but as a source of reusable procedural knowledge that can be externalized and accumulated.

We represent each rule as a structured entry

$$r_i = \big(r_i^{nl},\, r_i^{sym},\, \ell_i,\, u_i,\, c_i\big), \tag{4}$$

where r_i^{nl} is a natural-language rule description that provides high-level guidance to the planner, r_i^{sym} is a formal constraint encoding verifiable preconditions, state constraints, or temporal dependencies, ℓ_i is a rule label indicating its functional category, trigger pattern, or object type, u_i stores utility statistics that are initialized at induction time and updated during Re_2, and c_i denotes a coarse task category. This dual representation allows rules to operate in two complementary spaces: the natural-language component preserves interpretable procedural guidance suitable for prompt augmentation, while the symbolic component provides precise templates for retrieval, attribution, and consistency checking. In this way, Re_1 converts transient interaction outcomes into a memory substrate that is both semantically expressive and operationally verifiable.

To improve transferability, Re_1 applies explicit quality filtering before writing rules into memory. This step removes rules that are tightly bound to specific instances, overly short rules that lack meaningful constraints, and verbose descriptions that are unlikely to generalize. As a result, the memory is biased toward abstract procedural constraints rather than example-specific action scripts. In addition, Re_1 organizes rules through a label-aware structure that groups rules with similar functionality into shared semantic clusters. This improves alignment between retrieved rules and task contexts and also facilitates later rule merging and compression. Consequently, the output of Re_1 is not a loose collection of textual heuristics, but an expandable procedural memory composed of natural-language guidance, formal constraints, and semantic labels.

3.2. Re_2: Rule-Guided Refinement

The goal of Re_2 is to use the current rule library as adaptive test-time guidance for plan refinement, while continuously updating the memory according to the retrieved rules. Given the current instruction and grounded state s_t, the system first retrieves a task-relevant subset of rules

$$\tilde{\mathcal{R}}(x, s_t) \subseteq \mathcal{R}, \tag{5}$$

where retrieval considers each rule's current utility, the consistency between its labels and the current task semantics, and the compatibility between its formal constraints and the grounded state. In this way, the system selects a compact subset of highly relevant rules while avoiding unnecessary context clutter. Accordingly, Re_2 treats memory not as a static repository, but as an adaptive resource that is selectively activated for test-time refinement.

During planning, Re_2 injects the retrieved rules into the planner. The natural-language component r_i^{nl} is organized as a compact prompt augmentation block that encourages the planner to consider object states, preconditions, and temporal dependencies during action generation. The symbolic component r_i^{sym} serves as an implicit constraint template for consistency checking, rule attribution, and subsequent utility updating. This process yields a baseline plan V_1 and a rule-refined plan V_2, whose execution outcomes are then compared in the environment. The planner remains responsible for scene understanding, history reasoning, and action generation, while the rule library functions as an external procedural prior at test time. This unified mechanism supports both text-based planners in EAI and vision-language planners in EB-Habitat.

After rule-conditioned execution, Re_2 must determine which rules actually contributed to the observed outcome. When explicit rule references are available in the trajectory, they are used directly for attribution. Otherwise, the system performs heuristic matching based on the grounded state, environment feedback, and the natural-language descriptions, symbolic constraints, and labels of the injected rules to identify the set of hit rules. Compared with at-

tribution based only on surface textual overlap, this state-aware strategy yields more reliable rule attribution. It distinguishes rules that were injected but not causally involved in the decision process from rules that are directly relevant to the observed failure or successful correction. Attribution therefore serves as the bridge between transient rule usage and long-term memory updating.

Based on this attribution, Re_2 evaluates rule utility by comparing execution outcomes before and after rule-guided refinement. For each sample, rule utility is categorized into four outcome types:

$$S_{\text{outcome}} = \begin{cases} (V_1\checkmark, V_2\checkmark) \rightarrow \text{Neutral}, \\ (V_1\checkmark, V_2\times) \rightarrow \text{Harmful}, \\ (V_1\times, V_2\checkmark) \rightarrow \text{Effective}, \\ (V_1\times, V_2\times) \rightarrow \text{Ineffective}. \end{cases} \tag{6}$$

Here, **Effective** rules convert a failed plan into a successful one and should therefore be reinforced. **Harmful** rules corrupt originally successful plans and should be down-weighted or removed. **Neutral** rules are retained for future observation, while **Ineffective** rules indicate that the current correction is insufficient and may require later revision or replacement. This rule-level feedback closes the loop from execution outcomes to memory updating.

To distinguish beneficial from harmful interventions more stably, we adopt an asymmetric score update scheme:

$$\Delta(r_i) = \begin{cases} 3.0, & \text{if } F \rightarrow S, \\ 1.0, & \text{if } S \rightarrow S, \\ -1.0, & \text{if } F \rightarrow F, \\ -4.0, & \text{if } S \rightarrow F, \end{cases} \tag{7}$$

where F and S denote failure and success, respectively. For each attributed rule, the utility field is updated as $u_i \leftarrow u_i + \Delta(r_i)$. Rules that convert failures into successes receive strong positive reinforcement, whereas rules that corrupt previously successful episodes receive the strongest penalty. Through this asymmetric update, the rule library gradually retains rules with stable positive utility and suppresses rules that interfere with planning.

Beyond per-rule utility updates, Re_2 also performs structural refinement over the rule library. When multiple rules correspond to the same constraint type in label space and their formal templates are logically equivalent, one subsumes another, or they partially overlap, the system merges them into a more abstract shared rule by consolidating redundant natural-language descriptions and compressing repeated or overlapping symbolic conditions. As a result, Re_2 not only updates rule priorities from interaction outcomes, but also transforms local experience into a more compact and transferable long-term procedural memory through utility updating, label aggregation, and rule merging.

4. Experiments

We first evaluate Re^2 on the text-only EAI benchmark [17], and then extend our study to the multimodal EB-Habitat benchmark [32] to assess whether the same rule-memory framework generalizes to vision-language embodied planning. Finally, we present ablation studies and case studies to analyze the contribution of each component.

4.1. Experimental Setup

Text-based embodied benchmark: EAI. We evaluate Re^2 on EAI [17], a text-based embodied planning benchmark built on BEHAVIOR and VirtualHome. EAI tests whether agents can solve tasks from textual or symbolic states without visual grounding. It comprises four modules: *Goal Interpretation*, *Action Sequencing*, *Subgoal Decomposition*, and *Transition Modeling*, covering instruction understanding, long-horizon planning, and operator-level reasoning. Following the official protocol, we report standard metrics, including module-level success rates and F1 score. The benchmark provides a development split with step-wise feedback from a symbolic verifier, and an evaluation split with only sparse module-level metrics. This setting is well suited to assess whether rules induced from interaction feedback generalize to unseen tasks.

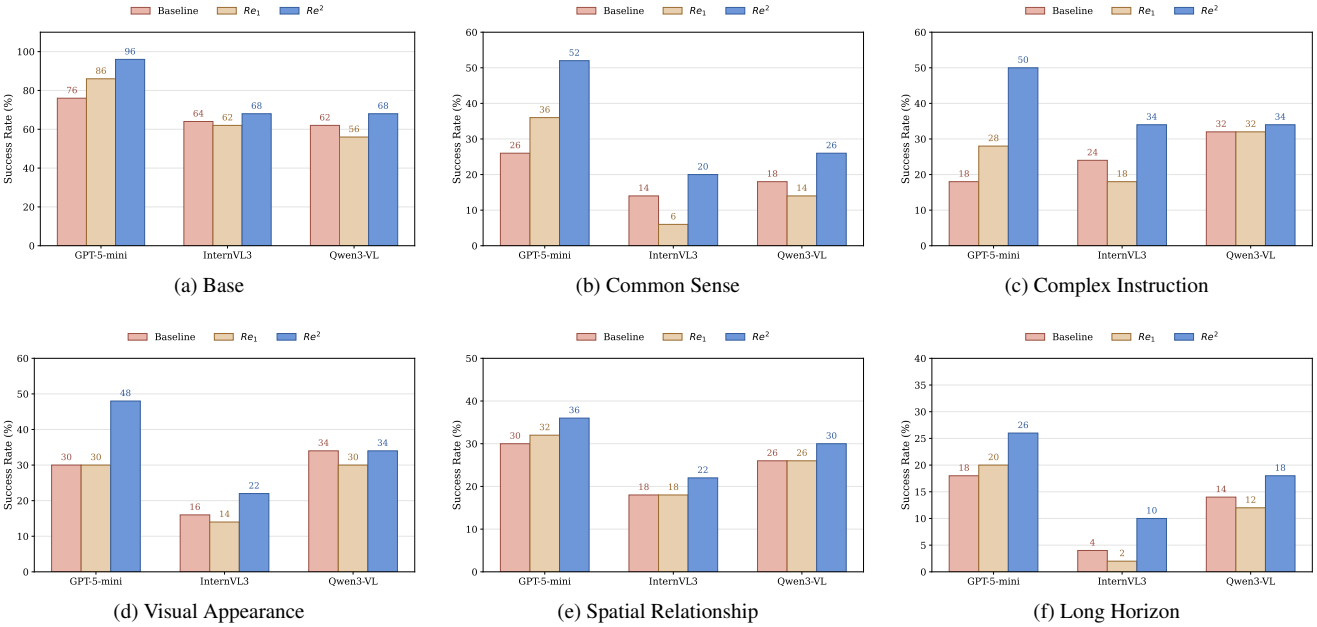
Multimodal embodied benchmark: EB-Habitat. We further evaluate Re^2 on EB-Habitat [32], a multimodal embodied planning benchmark built on Habitat 2.0 and the Language Rearrangement setting [27]. Unlike EAI, EB-Habitat requires grounding language in egocentric visual observations and executing high-level skills such as navigation, pick, place, open, and close. The agent must infer object identities, receptacle states, spatial relations, and action feasibility from visual inputs and interaction history, rather than relying on full symbolic states. This imposes stronger demands on perceptual grounding and execution during planning. We evaluate three VLM backbones (GPT-5-mini, InternVL3, Qwen3-VL) under three settings: *Baseline*, Re_1, and Re^2. Results are reported across six task dimensions: *Baseline*, *Common Sense*, *Complex Instruction*, *Visual Appearance*, *Spatial Relationship*, and *Long Horizon*, to identify where rule induction and refinement yield the largest gains.

4.2. Results on Text-Only EAI Benchmark

Main results on the evaluation split. We report the results on EAI in Table 1. The baseline planners exhibit substantial variance across modules, indicating that text-only embodied planning remains challenging in symbolically structured environments. Re_1 improves performance over the *Baseline (GPT-5)*, showing that execution trajectories from the development split can be converted into useful procedural rules. Building on induced rule memory,

Table 1. Overall results (%) on the EAI benchmark. *V*: VirtualHome, *B*: BEHAVIOR.

Model	Overall Perf.	Average Perf. *Module SR* V	Average Perf. *Module SR* B	Goal Interpretation F_1 V	Goal Interpretation F_1 B	Action Sequencing *Task SR* V	Action Sequencing *Task SR* B	Action Sequencing *Execution SR* V	Action Sequencing *Execution SR* B	Subgoal Decomposition *Task SR* V	Subgoal Decomposition *Task SR* B	Subgoal Decomposition *Execution SR* V	Subgoal Decomposition *Execution SR* B	Transition Modeling F_1 V	Transition Modeling F_1 B	Transition Modeling *Planner SR* V	Transition Modeling *Planner SR* B
GPT-4o	-	-	-	24.9	77.4	68.3	40.0	83.6	50.0	68.1	45.0	85.1	51.0	43.1	62.9	29.6	53.0
GPT-o3	-	-	-	35.3	79.1	68.9	64.6	83.2	70.8	74.3	52.0	87.1	57.3	44.6	52.4	92.4	93.0
EAI-Baseline	67.28	61.19	73.36	37.8	**85.4**	67.6	72.0	81.9	80.0	71.3	59.0	**88.3**	65.0	45.1	58.1	91.0	96.0
GPT-5	67.58	64.21	70.95	36.3	79.2	72.3	71.0	84.9	74.0	73.2	54.0	87.2	60.0	36.4	43.0	86.3	97.0
Re_1 Agent	75.29 (↑ 7.71)	67.57 (↑ 3.36)	83.01 (↑ 12.06)	49.0 (↑ 12.7)	84.9 (↑ 5.7)	71.2 (↓ 1.1)	73.0 (↑ 2.0)	84.9 -	78.0 (↑ 4.0)	74.7 (↑ 1.5)	80.0 (↑ 26.0)	88.0 (↑ 0.8)	86.6 (↑ 26.6)	54.8 (↑ 18.4)	43.0 -	91.4 (↑ 5.1)	97.0 -
Re^2 Agent	**81.36** (↑ 6.07)	**76.36** (↑ 8.79)	**86.35** (↑ 3.34)	**57.9** (↑ 8.9)	85.0 (↑ 0.1)	**73.5** (↑ 2.3)	**81.0** (↑ 8.0)	**84.9** -	**91.0** (↑ 1.4)	**74.7** -	**80.0** -	86.6 (↓ 1.4)	**88.0** (↑ 1.4)	**98.8** (↑ 44.0)	**99.8** (↑ 56.8)	**99.9** (↑ 8.5)	**99.0** (↑ 2.0)



(a) Base (b) Common Sense (c) Complex Instruction

(d) Visual Appearance (e) Spatial Relationship (f) Long Horizon

Figure 3. Performance of *Baseline*, Re_1, and Re^2 on EB-Habitat across six task categories

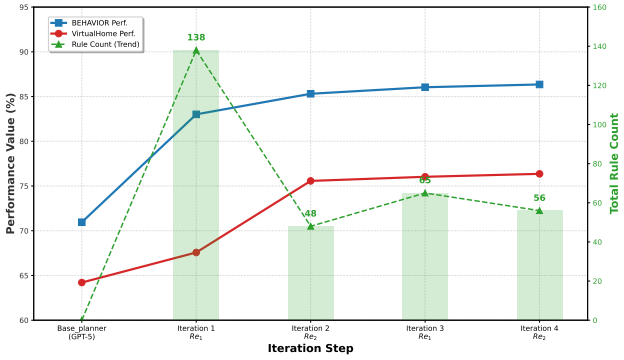


Figure 4. Rule-library evolution and evaluation performance during EAI development. As rule-set changes stabilize, Re^2 also becomes more stable.

Re^2 yields further gains by selectively retrieving and updating rules according to their actual utility during planning. Overall, these results show that Re^2 improves text-only embodied planning not through static rule injection alone, but through iterative rule refinement and memory updating.

Development-to-evaluation generalization. In EAI, symbolic verifier feedback is available only during development. Accordingly, rule induction, filtering, and refinement must be completed before evaluation. We track the evolution of the rule library during development and visualize the relation between rule-set changes and evaluation performance in Fig. 4. Performance improvement closely follows the stabilization of the rule library, suggesting that the learned memory gradually converges to a compact and transferable set of procedural constraints. This explains why Re_2 generalizes better than Re_1: beyond inducing candidate rules from failures, it further reweights and consolidates them according to their utility.

4.3. Results on Multimodal EB-Habitat Benchmark

Main results. We next evaluate the proposed Re^2 framework on EB-Habitat to examine its generalization to mul-

timodal embodied planning. Importantly, the *Baseline* already includes a reflection module; Re_1 further introduces induced rule memory on top of this reflective planner, and Re_2 denotes the full two-stage instantiation of the Re^2 framework. Fig. 3 compares *Baseline*, Re_1, and Re^2 across GPT-5-mini, InternVL3, and Qwen3-VL. Overall, Re^2 achieves the strongest performance, showing that rule retrieval and utility-aware refinement remain effective even when planning depends on both visual grounding and language reasoning. Compared with the text-only EAI setting, EB-Habitat is more challenging because the model must reason over visually grounded object states, affordances, and spatial relations.

A clear model-dependent pattern also emerges. For GPT-5-mini, adding Re_1 on top of the reflective *Baseline* already improves the average score from 33.0 to 38.7, and Re^2 further raises it to 51.3. In contrast, smaller multimodal models do not consistently benefit from direct injection of induced rules: InternVL3 drops from 23.3 to 20.0 under Re_1, and Qwen3-VL drops from 31.0 to 28.3. Both models recover under Re^2, reaching 29.3 and 35.0, respectively. This suggests that rules induced from reflection are useful, but their effectiveness in multimodal planning depends critically on whether they are selectively filtered and refined before being injected back into the planner.

Per-dimension analysis. The gains of Re^2 are most evident on *Complex Instruction*, *Common Sense*, and *Long Horizon*, where planning requires stronger temporal coordination and multi-step reasoning. For GPT-5-mini, Re^2 improves *Complex Instruction* from 18 to 50, *Common Sense* from 26 to 52, and *Long Horizon* from 18 to 26. InternVL3 improves from 24 to 34, 14 to 20, and 4 to 10 on the same three dimensions, while Qwen3-VL improves from 32 to 34, 18 to 26, and 14 to 18. We also observe gains on visually grounded dimensions such as *Visual Appearance* and *Spatial Relationship*, especially for GPT-5-mini and InternVL3, indicating that refined procedural memory can complement perception when properly controlled.

The comparison between Re_1 and Re^2 highlights an important multimodal effect. While GPT-5-mini benefits from direct rule augmentation, InternVL3 and Qwen3-VL often degrade under Re_1, suggesting that smaller multimodal models cannot reliably balance the dual burden of perception and additional language-side reasoning. Injecting an unfiltered rule library may therefore interfere with visual grounding and lead to suboptimal plans. This observation directly motivates the need for the second stage, Re^2: Re_1 is useful for harvesting candidate procedural knowledge, whereas Re^2 is necessary for compressing, filtering, and selectively retrieving rules at test time. The consistent advantage of Re^2 therefore shows that multimodal embodied planning requires not only reflection-based rule induction, but also utility-aware rule refinement.

4.4. Ablation Study

Ablation on Re_1 and Re^2. We first compare the *Baseline* planner with its Re_1 and Re^2 variants in Fig. 3. The results show that direct rule injection in Re_1 helps the stronger closed-source model, but may degrade smaller VLMs. By contrast, Re^2 consistently improves performance, demonstrating the necessity of selective retrieval and rule refinement.

Rule retrieval without utility updating. We next remove utility updating while retaining test-time retrieval, and report the results on GPT-5-mini in Table 2. Although the model can still retrieve relevant rules, it can no longer adapt the rule memory based on rollout outcomes. The performance gap to full Re^2 shows that utility-aware updating is essential for reliable rule memory.

Natural-language-only rules. Finally, we remove the symbolic component and keep only natural-language rules. This variant also underperforms the full model, confirming that symbolic rules are important for retrieval, consistency checking, and rule attribution.

Overall, the ablation results show that Re^2 benefits from not only reflective rule induction, but also utility-aware refinement and dual rule representation.

4.5. Case Study

Finally, we present qualitative case studies from EB-Habitat. Fig. 5 shows three representative multimodal failure modes: missing proximity before pick, incorrect pick-before-place ordering, and inaccurate grounding of attribute-based object references. In all three cases, Re^2 converts failed trajectories into reusable natural-language (NL) and symbolic (SYM) rules, which then support successful correction in subsequent planning. These examples illustrate how local interaction feedback can be consolidated into transferable procedural memory for more executable and robust embodied planning. We will display more rule libraries in the appendix.

5. Related Work

Embodied Benchmarks. Recent embodied AI research has been supported by interactive simulation platforms such as BEHAVIOR, VirtualHome, AI2-THOR, and iGibson [15, 16, 21, 25], which enable grounded evaluation of planning and interaction. Building on these platforms, prior benchmarks have studied embodied planning, navigation, manipulation, and long-horizon decision making with LLMs and VLMs [7, 8, 18, 34, 35]. More recent unified benchmarks, including EAI and EmbodiedBench, further standardize task categories and feedback signals across

Table 2. Ablation results of GPT-5-mini on EB-Habitat. The *Baseline* already includes the reflection module. Re_1 further introduces induced rule memory, while Re^2 denotes the full two-stage instantiation with retrieval and utility-aware refinement.

Variant	Induction	Retrieval	Utility Update	Symbolic Rule	Avg.	Baseline	Common	Complex	Visual	Spatial	Long
Reflective Baseline (GPT-5-mini)	✗	✗	✗	✗	33.0	76	26	18	30	30	18
+ Induced Rules (Re_1)	✓	✗	✗	✓	38.7	86	36	28	30	32	20
+ Retrieval w/o Update	✓	✓	✗	✓	46.2	92	45	42	41	34	23
+ Retrieval w/ NL-only Rules	✓	✓	✓	✗	44.3	90	43	40	39	32	22
Re^2 **Agent (full)**	✓	✓	✓	✓	**51.3**	**96**	**52**	**50**	**48**	**36**	**26**

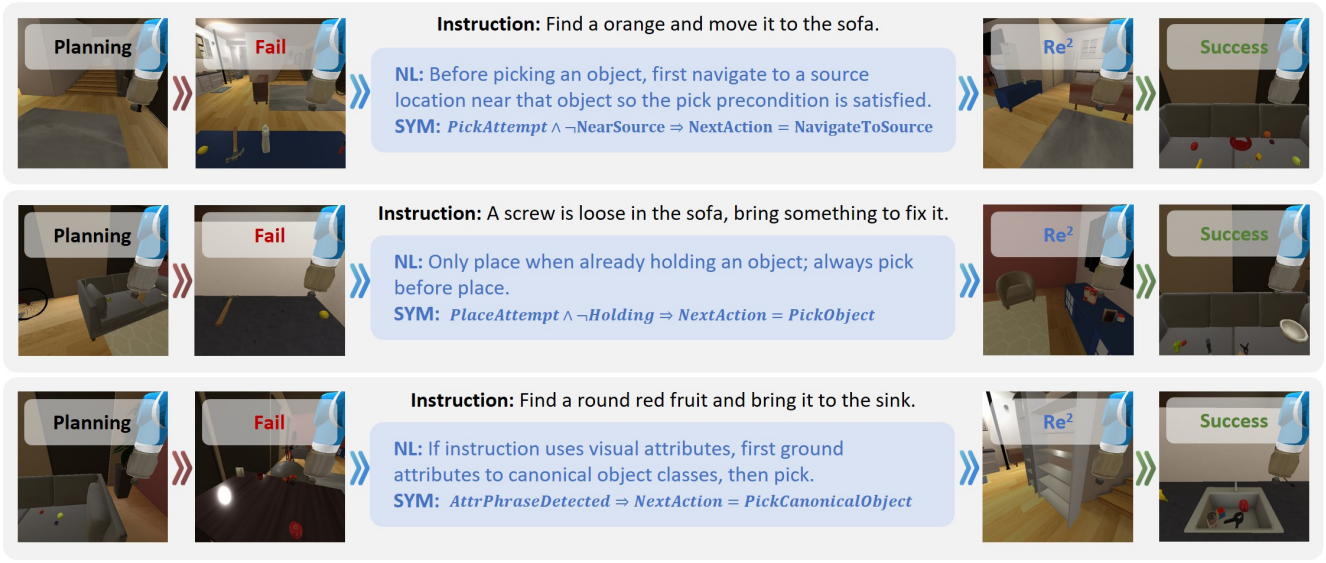


Figure 5. Qualitative Case Studies of Rule-Guided Failure Correction on EB-Habitat.

diverse embodied settings [17, 32], making them suitable testbeds for studying whether procedural knowledge can be accumulated, reused, and transferred across tasks and environment modalities.

Embodied Decision Making. Existing embodied decision-making methods have improved the planning ability of LLM/VLM-based agents, but they still struggle to accumulate and reuse procedural constraints reliably across episodes. Static prompting methods generate plans directly from task descriptions [12, 24], yet often overlook environment-specific executability. Feedback-driven methods introduce self-correction through interaction [4, 19, 28, 36], but usually treat failures as transient repair signals rather than reusable knowledge. Neuro-symbolic approaches improve executability by introducing explicit constraints or structured world models [2, 9, 10, 14, 29], while recent test-time methods further explore rule injection during inference [11, 20, 23]. However, directly accumulating or injecting rules is not always effective, especially in multimodal embodied planning, where long and unfiltered rule memory can instead destabilize decision making and reduce planning reliability. This limitation motivates our framework, which treats interaction feedback as an evolving external procedural memory and improves it through a closed loop of rule induction, selective retrieval, utility-aware updating, and rule-guided refinement.

6. Conclusion

We presented Re^2, a closed-loop framework that converts interaction feedback into reusable procedural memory for embodied planning. Re_1 abstracts transferable procedural rules from successful and failed interaction trajectories, while Re_2 selectively retrieves and refines these rules as adaptive guidance for test-time planning. Together, they support selective rule reuse and continual test-time improvement across both text-based and multimodal settings. Re^2 achieves a state-of-the-art overall score of **81.36** on EAI and improves the average performance on EB-Habitat from **33.0** to **51.3** (**+55.5%**) over the reflective baseline. These results highlight the effectiveness of interaction-driven procedural memory for improving executability, robustness, and cross-setting generalization in embodied planning. Future work will explore more dynamic open-world environments and real-world robotic deployment under practical constraints.

References

[1] 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. 1

[2] Sanghyun Ahn, Wonje Choi, Junyong Lee, Jinwoo Park, and Honguk Woo. Towards reliable code-as-policies: A neuro-symbolic framework for embodied task planning. *arXiv preprint arXiv:2510.21302*, 2025. 8

[3] Alisson Azzolini, Junjie Bai, Hannah Brandon, Jiaxin Cao, Prithvijit Chattopadhyay, Huayu Chen, Jinju Chu, Yin Cui, Jenna Diamond, Yifan Ding, et al. Cosmos-reason1: From physical common sense to embodied reasoning. *arXiv preprint arXiv:2503.15558*, 2025. 1

[4] Hanyang Chen, Mark Zhao, Rui Yang, Qinwei Ma, Ke Yang, Jiarui Yao, Kangrui Wang, Hao Bai, Zhenhailong Wang, Rui Pan, et al. Era: Transforming vlms into embodied agents via embodied prior learning and online reinforcement learning. *arXiv preprint arXiv:2510.12693*, 2025. 8

[5] Wenhu Chen, Pat Verga, Michiel De Jong, John Wieting, and William Cohen. Augmenting pre-trained language models with qa-memory for open-domain question answering. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1597–1610, 2023. 1

[6] Sijie Cheng, Zhicheng Guo, Jingwen Wu, Kechen Fang, Peng Li, Huaping Liu, and Yang Liu. Egothink: Evaluating first-person perspective thinking capability of vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14291–14302, 2024. 1

[7] Zhili Cheng, Yuge Tu, Ran Li, Shiqi Dai, Jinyi Hu, Shengding Hu, Jiahao Li, Yang Shi, Tianyu Yu, Weize Chen, et al. Embodiedeval: Evaluate multimodal llms as embodied agents. *arXiv preprint arXiv:2501.11858*, 2025. 7

[8] Jae-Woo Choi, Youngwoo Yoon, Hyobin Ong, Jaehong Kim, and Minsu Jang. Lota-bench: Benchmarking language-oriented task planners for embodied agents. *arXiv preprint arXiv:2402.08178*, 2024. 7

[9] Wonje Choi, Jinwoo Park, Sanghyun Ahn, Daehee Lee, and Honguk Woo. Nesyc: A neuro-symbolic continual learner for complex embodied tasks in open domains. *arXiv preprint arXiv:2503.00870*, 2025. 1, 8

[10] Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. *Advances in Neural Information Processing Systems*, 36:79081–79094, 2023. 8

[11] Yining Hong, Huang Huang, Manling Li, Li Fei-Fei, Jiajun Wu, and Yejin Choi. Learning from trials and errors: Reflective test-time planning for embodied llms. *arXiv preprint arXiv:2602.21198*, 2026. 1, 8

[12] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147, 2022. 8

[13] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022. 1

[14] Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. Position: Llms can't plan, but can help planning in llm-modulo frameworks. In *Forty-first International Conference on Machine Learning*, 2024. 8

[15] 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. 7

[16] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, et al. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *arXiv preprint arXiv:2108.03272*, 2021. 7

[17] Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Erran Li Li, Ruohan Zhang, et al. Embodied agent interface: Benchmarking llms for embodied decision making. *Advances in Neural Information Processing Systems*, 37: 100428–100534, 2024. 1, 2, 5, 8

[18] Yulin Luo, Chun-Kai Fan, Menghang Dong, Jiayu Shi, Mengdi Zhao, Bo-Wen Zhang, Cheng Chi, Jiaming Liu, Gaole Dai, Rongyu Zhang, et al. Robobench: A comprehensive evaluation benchmark for multimodal large language models as embodied brain. *arXiv preprint arXiv:2510.17801*, 2025. 7

[19] Sean Memery, Kevin Denamganaï, Jiaxin Zhang, Zehai Tu, Yiwen Guo, and Kartic Subr. Cuetip: An interactive and explainable physics-aware pool assistant. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–11, 2025. 8

[20] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20286–20332, 2025. 1, 8

[21] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8494–8502, 2018. 1, 7

[22] 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. 1

[23] 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. 1, 8

[24] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2998–3009, 2023. 8

[25] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Elliott Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, Karen Liu, et al. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on robot learning*, pages 477–490, 2022. 1, 7

[26] 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. 1

[27] Andrew Szot, Max Schwarzer, Harsh Agrawal, Bogdan Mazoure, Rin Metcalf, Walter Talbott, Natalie Mackraz, R Devon Hjelm, and Alexander T Toshev. Large language models as generalizable policies for embodied tasks. In *The Twelfth International Conference on Learning Representations*, 2023. 5

[28] Wanxin Tian, Shijie Zhang, Kevin Zhang, Xiaowei Chi, Chunkai Fan, Junyu Lu, Yulin Luo, Qiang Zhou, Yiming Zhao, Ning Liu, et al. Seea-r1: Tree-structured reinforcement fine-tuning for self-evolving embodied agents. *arXiv preprint arXiv:2506.21669*, 2025. 8

[29] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. 8

[30] Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, et al. Webwalker: Benchmarking llms in web traversal. *arXiv preprint arXiv:2501.07572*, 2025. 1

[31] Jihan Yang, Shusheng Yang, Anjali W Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10632–10643, 2025. 1

[32] 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. 1, 2, 5, 8

[33] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022. 1

[34] Sheng Yin, Xianghe Pang, Yuanzhuo Ding, Menglan Chen, Yutong Bi, Yichen Xiong, Wenhao Huang, Zhen Xiang, Jing Shao, and Siheng Chen. Safeagentbench: A benchmark for safe task planning of embodied llm agents. *arXiv preprint arXiv:2412.13178*, 2024. 7

[35] Lingfeng Zhang, Yuening Wang, Hongjian Gu, Atia Hamidizadeh, Zhanguang Zhang, Yuecheng Liu, Yutong Wang, David Gamaliel Arcos Bravo, Junyi Dong, Shunbo Zhou, et al. Et-plan-bench: Embodied task-level planning benchmark towards spatial-temporal cognition with foundation models. *arXiv preprint arXiv:2410.14682*, 2024. 7

[36] Ruoxuan Zhang, Bin Wen, Hongxia Xie, Yi Yao, Songhan Zuo, Jian-Yu Jiang-Lin, Hong-Han Shuai, and Wen-Huang Cheng. Cookanything: A framework for flexible and consistent multi-step recipe image generation. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 7854–7863, 2025. 8

[37] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 19632–19642, 2024. 1

[38] Siyu Zhou, Tianyi Zhou, Yijun Yang, Guodong Long, Deheng Ye, Jing Jiang, and Chengqi Zhang. Wall-e 2.0: World alignment by neurosymbolic learning improves world model-based llm agents. *arXiv preprint arXiv:2504.15785*, 2025. 1

Re^2: Reflective Rule Induction and Rule-Guided Refinement for Embodied Planning

Supplementary Material

7. Supplementary experiments

We further provide a fine-grained analysis on the EAI development split in Tables 3, 4, and 5. The detailed breakdown across modules shows that task difficulty varies substantially even within the same benchmark, making a single aggregate score insufficient to characterize model behavior. In this setting, Re_1 captures task-specific discrepancies revealed by execution reports, while Re_2 further improves the alignment between rules and task requirements through iterative selection and refinement. Overall, these results show that symbolic feedback and structured rule memory together provide a data-efficient mechanism for test-time self-improvement.

8. Rule Library

8.1. EAI

We first present a case study of the rule library induced by Re^2 in EAI. In this text-based setting, the agent must interpret natural language instructions and execute actions that satisfy symbolic task constraints. Since the environment provides verifier feedback over execution outcomes, the agent can analyze discrepancies between intended goals and actual transitions to induce reusable procedural rules across four dimensions. Below, we highlight representative rules from the learned rule library to illustrate the procedural knowledge acquired in EAI.

8.1.1. Goal Interpretation

These rules ensure that the interpreted goal state accurately reflects the user's intent. Table 6 presents the error patterns for selected tasks and the analysis of error causes provided by Re_1. Some refined rules are shown below:

[Rule-1] Exclusive Content Constraint. When distributing distinct categories into containers, the agent must model the exclusion of non-target categories. *Formal Logic:* $\forall c \in \mathcal{C}, \forall o_1 \in \text{Cat}_A : \text{Inside}(o_1, c) \rightarrow \neg\exists o_2 \in \text{Cat}_B : \text{Inside}(o_2, c)$. **Rationale:** Ensures the predicted state captures necessary negative constraints for multi-category sorting.

[Rule-2] State Transition Persistence. For movement instructions, the interpreter prioritizes the terminal state and suppresses redundant negations of initial coordinates. *Formal Logic:* $\forall o, l_{\text{final}} : \text{On}(o, l_{\text{final}}) \implies \text{Goal}(\text{On}(o, l_{\text{final}})) \wedge \text{Ignore}(\neg\text{On}(o, l_{\text{initial}}))$. **Rationale:** Reduces hallucination of irrelevant negative conditions.

[Rule-3] Functional Surface Affordance. Verbs like "serve" or "organize" map to `ontop` for elevated surfaces, strictly avoiding `under` unless specified. *Formal Logic:* $\text{Verb} \in \{\text{'serve', 'shelve'}\} \rightarrow \text{Goal}(\text{Ontop}(o, S)) \wedge \neg\text{Under}(o, S)$. **Rationale:** Corrects logical inversion errors in spatial tasks.

8.1.2. Action Sequencing & Subgoal Decomposition

Governance rules for optimizing planning logic and handling physical constraints. Table 7 and Table 8 presents the error patterns for selected tasks and the analysis of error causes provided by Re_1. Some refined rules are shown below:

[Rule-4] Container Access Constraint. An agent cannot interact with objects inside a container unless the container is in the `open` state. *Formal Logic:* $\forall o, c, s : (\text{Inside}(o, c) \wedge \text{Grasp}(o, s)) \rightarrow \text{Open}(c, s)$. **Rationale:** Eliminates `MISSING_STEP` errors in storage tasks.

[Rule-5] Action-Affordance Guard. Actions can only be sequenced if the target object supports the resulting state within the domain definition. *Formal Logic:* $\forall a, o : \text{Action}(a, o) \rightarrow \text{HasAffordance}(o, \text{State}(a))$. **Rationale:** Prevents `AFFORDANCE_ERROR` (e.g., soaking non-absorbent items).

[Rule-6] Cleaning Tool Pipeline. Preparation for cleaning follows a strict sequence: Grasp $\prec$ PlaceIn(Sink) $\prec$ Toggle(Sink) $\prec$ Soak. *Formal Logic:* $(\text{Grasp}(T) \prec \text{PlaceIn}(T, \text{Sink}) \prec \text{Toggle}(\text{Sink})) \rightarrow \text{Soaked}(T)$. **Rationale:** Ensures tool readiness before sanitation tasks.

8.1.3. Transition Modeling

Rules synthesized to correct the agent's internal transition model and ensure physical consistency. Table 9 presents the error patterns for selected tasks and the analysis of error causes provided by Re_1. Some refined rules are shown below:

[Rule-7] Recursive Location Update. Navigation must update the coordinates of the agent and all objects in the agent's inventory. *Formal Logic:* $\text{Walk}(A, L_{new}) \rightarrow (At(A, L_{new}) \wedge \forall o \in \text{Inv}, At(o, L_{new}))$. **Rationale:** Resolves the Frame Problem where held objects are "left behind" in latent space.

[Rule-8] Spatial Containment Reciprocity. Placing an object into a container must create a containment relationship and terminate agent possession. *Formal Logic:* $\text{PutInside}(A, o, c) \rightarrow \text{Effect}^{+}(\text{Inside}(o, c)) \wedge \text{Effect}^{-}(\text{Holds}(A, o))$. **Rationale:** Prevents hallucinated possession and ensures environment graph consistency.

Table 3. Logic form accuracy (%) for goal interpretation on the EAI benchmark.

Model	Overall Performance						State Goal						Relation Goal						Action Goal					
	Precision		*Recall*		F_1		*Precision*		*Recall*		F_1		*Precision*		*Recall*		F_1		*Precision*		*Recall*		F_1	
	V	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*
Claude-3.5 Sonnet	21.7	81.1	69.6	84.4	33.0	82.7	25.3	74.0	60.9	94.8	35.8	83.1	31.1	84.4	63.8	81.3	41.8	82.9	14.0	-	**98.8**	-	24.5	-
Gemini 1.5 Pro	33.6	78.8	39.9	80.4	36.2	79.6	47.2	**94.0**	47.5	92.8	47.3	**93.4**	42.0	74.4	7.0	76.7	12.4	75.6	24.1	-	81.4	-	37.2	-
Deepseek-chat	21.1	81.5	64.2	82.5	31.8	82.0	24.2	83.8	59.7	91.5	34.4	87.5	28.2	80.9	53.0	79.8	36.8	80.3	14.8	-	94.4	-	25.5	-
Deepseek-reasoner	34.8	83.4	67.4	87.1	45.9	85.2	29.7	79.8	65.9	**98.0**	40.9	88.0	45.8	84.7	63.8	**83.8**	53.3	84.3	33.1	-	77.2	-	46.3	-
Qwen3-next-80b-a3b-instruct	15.5	61.1	51.5	74.6	23.9	67.2	22.7	47.7	54.4	93.5	32.0	63.1	13.9	69.3	24.6	69.0	17.7	69.2	11.7	-	95.6	-	20.9	-
GPT-4o	26.4	76.5	59.1	82.2	36.5	79.2	29.0	67.1	60.0	94.8	39.1	78.6	31.5	81.1	43.6	78.5	36.6	79.8	20.5	-	85.8	-	33.1	-
o1-preview	31.8	78.1	65.4	85.4	42.7	81.6	28.2	66.8	60.3	94.8	38.5	78.4	44.9	82.9	62.4	82.7	52.2	82.8	26.0	-	81.5	-	39.5	-
GPT-5	28.4	74.6	56.8	84.5	35.7	79.2	30.2	75.9	58.4	96.7	41.2	85.1	33.5	74.1	45.8	81.0	38.9	77.4	**83.5**	-	37.5	-	36.4	-
Re_1 Agent	35.5	83.6	**75.0**	86.5	48.2	85.0	36.5	78.3	**78.3**	96.7	49.8	86.5	45.6	85.6	**64.9**	83.5	53.6	84.5	26.3	-	87.0	-	40.3	-
Re^2 **Agent**	**52.6**	**83.6**	63.2	**86.5**	**57.5**	**85.0**	**51.0**	78.3	65.9	96.7	**57.5**	86.5	**54.2**	**85.6**	64.4	83.5	**58.9**	**84.5**	53.6	-	55.6	-	**54.5**	-

Table 4. Goal satisfaction rates (%) for action sequencing and subgoal decomposition on the EAI benchmark.

Model	Action Sequencing								Subgoal Decomposition							
	Total		State Goal		Relation Goal		Action Goal		Total		State Goal		Relation Goal		Action Goal	
	V	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*
Claude-3.5 Sonnet	79.9	62.2	87.8	63.0	83.3	62.4	60.8	-	90.1	39.9	92.9	41.0	88.6	39.5	87.0	-
Gemini 1.5 Pro	77.2	42.6	85.6	41.0	76.7	43.2	62.2	-	83.9	35.4	91.2	31.0	72.5	37.1	89.5	-
Deepseek-chat	62.5	68.0	49.6	71.5	78.3	66.7	67.6	-	**90.8**	38.8	93.8	34.0	85.9	40.6	**93.2**	-
Deepseek-reasoner	83.0	**94.5**	90.3	**93.0**	83.9	**95.1**	68.2	-	89.6	73.5	90.9	62.5	87.2	77.6	91.4	-
Qwen3-next-80b-a3b-instruct	79.4	50.0	84.9	48.5	83.3	50.6	64.2	-	84.1	39.9	92.1	45.0	70.8	38.0	92.0	-
GPT-4o	76.2	46.5	87.1	49.0	76.1	45.5	56.1	-	89.4	52.3	92.1	50.0	84.2	53.2	93.2	-
o1-preview	67.8	85.8	80.9	89.5	65.0	84.4	46.6	-	90.6	65.9	91.8	56.5	88.3	69.4	92.6	-
GPT-5	83.8	71.0	93.5	68.0	82.8	68.2	68.2	-	90.1	60.9	**93.8**	67.0	**89.1**	58.7	84.3	-
Re_1 Agent	81.8	83.3	**93.5**	78.5	82.8	89.1	58.8	-	90.3	86.1	91.0	80.0	88.7	88.4	90.3	-
Re^2 **Agent**	**85.3**	85.6	92.4	89.5	**87.8**	84.0	**68.9**	-	88.5	**86.1**	90.6	**80.0**	84.6	**88.4**	91.4	-

Table 5. Logic form accuracy (F1) and planner success rate (SR) (%) for transition modeling on the EAI benchmark.

Model	Object States				Object Orientation				Object Affordance				Spatial Relations				Non-Spatial Relations			
	F_1		*SR*		F_1		*SR*		F_1		*SR*		F_1		*SR*		F_1		*SR*	
	V	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*
Claude-3.5 Sonnet	60.5	78.8	67.4	86.7	95.3	-	96.4	-	76.6	-	67.7	-	42.4	58.6	96.6	80.9	5.9	73.6	91.9	80.3
Gemini 1.5 Pro	18.8	55.9	94.4	35.6	90.9	-	89.3	-	77.1	-	95.8	-	38.7	35.9	89.0	40.4	7.8	52.8	83.8	39.3
Deepseek-chat	50.3	68.7	43.8	60.0	5.6	-	21.4	-	54.9	-	55.1	-	46.0	49.3	73.1	67.0	10.1	76.3	54.1	62.3
Deepseek-reasoner	57.5	70.5	78.1	82.2	75.3	-	75.0	-	67.5	-	80.2	-	37.4	53.4	91.7	83.0	10.3	78.4	**95.9**	83.6
Qwen3-next-80b-a3b-instruct	61.9	57.5	9.0	46.7	65.2	-	0.0	-	73.5	-	6.0	-	34.2	44.9	15.2	52.1	1.8	66.2	21.6	45.9
GPT-4o	54.6	71.3	71.9	68.9	52.8	-	78.6	-	74.9	-	63.5	-	40.8	45.9	66.9	64.9	7.5	73.0	68.9	68.9
o1-preview	58.5	78.3	69.1	86.7	78.4	-	**100.0**	-	77.5	-	76.1	-	38.8	56.3	76.6	89.4	11.8	83.5	78.4	90.2
GPT-5	49.3	54.5	74.2	93.4	16.7	-	82.1	-	37.7	-	74.2	-	31.7	51.3	93.8	96.5	8.8	73.1	90.5	95.6
Re_1 Agent	55.2	54.5	89.9	94.5	26.1	-	96.4	-	60.7	-	90.4	-	40.6	51.3	86.2	97.0	6.7	73.1	79.7	97.1
Re^2 **Agent**	**99.8**	**99.9**	**98.9**	**98.9**	**100.0**	-	96.4	-	**99.8**	-	**98.2**	-	**99.8**	**99.8**	**95.9**	**99.2**	**96.3**	**100.0**	94.6	**99.4**

8.2. EB-Habitat

We further present a case study of the rule library induced by Re^2 in EB-Habitat. Unlike EAI, EB-Habitat requires the agent to ground language instructions in egocentric visual observations and perform embodied skills such as navigation, picking, placing, opening, and closing. Without access to a complete symbolic state, the agent must infer object identities, receptacle states, spatial relationships, and interaction feasibility from visual observations and interaction history. Below, we highlight representative rules from the learned rule library to illustrate the types of procedural knowledge acquired in this multimodal setting.

8.2.1. Base

These rules are summarized in the base task module. Figure 6 shows the original error trajectory and the corrected plan driven by rules.

[Rule-1] Proximity Before Pick. When attempting to pick any object, the agent must first be near that object. *Formal Logic:* $\forall o,\ \text{Pick}(o) \rightarrow \text{Near}(o)$. **Rationale:** Prevents invalid pick actions caused by distance mismatch.

[Rule-2] Hold Before Place. Any placement action requires that the target object is currently held. *Formal Logic:* $\forall o, d,\ \text{Place}(o, d) \rightarrow \text{Hold}(o)$. **Rationale:** Enforces state consistency between manipulation precondition and execution.

[Rule-3] Relocation Execution Order. Object relocation must follow source navigation, pickup, destination navigation, and placement in order. *Formal Logic:* $\forall o, s, d,\ \text{Move}(o, s, d) \rightarrow \text{Nav}(s) \prec \text{Pick}(o) \prec \text{Nav}(d) \prec \text{Place}(o, d)$. **Rationale:** Converts high-level move intent into a valid executable manipulation chain.

Table 6. Error Analysis of Goal Interpretation Module.

Task Name	Error Mode	Evidence	Root Cause
bottling_fruit_0	Recall Failure (Low Recall)	not inside(peach.n.03_1, jar.n.01_1), not inside(strawberry.n.01_1, jar.n.01_2)	Logical Gap: Failed to capture "exclusive containment" constraints (ensuring only one fruit type per jar).
cleaning_up_the_kitchen_only_0	Precision Failure (Hallucination)	not dusty(plate.n.04_1), not stained(cabinet.n.01_1), not touching(blender.n.01_1, countertop.n.01_1)	Over-specification: Predicted generic "cleanliness" states for every object in the scene regardless of the specific "kitchen-only" instruction.
bringing_in_wood_0	Precision Failure (Hallucination)	not onfloor(plywood.n.01_1, floor.n.01_1) (x3 for all plywood)	Redundancy: Incorrectly inferred that moving an object to a new floor requires an explicit negation of the old floor state.
filling_an_Easter_basket_0	Type Mismatch	ontop(basket.n.01_1, countertop.n.01_1), nextto(book.n.02_1, basket.n.01_1)	Spatial Misalignment: Failed to distinguish functional "inside" relations from necessary spatial "ontop" positioning.
serving_a_meal_0	Logical Failure (Total Mismatch)	under(table.n.02_1, chicken.n.01_1) (all objects placed under table)	Semantic Inversion: Interpreted "serving on a table" as placing items "under" the table, resulting in 0% precision and recall.

Table 7. Error Analysis of Action Sequencing Module.

Task Name	Error Type	Evidence	Root Cause
bottling_fruit_0	MISSING_STEP	RIGHT_GRASP strawberry_0 — fridge_97 is closed	Precondition Violation: Failed to ensure the container (fridge) was open before attempting to interact with the internal object.
cleaning_floors_0	AFFORDANCE_ERROR	SOAK toothpaste_0 — toothpaste_0 does not have target state	Semantic Misalignment: The planner attempted to apply a SOAK action to an object (toothpaste) that lacks the physical affordance for that state.
chopping_vegetables_0	WRONG_TEMPORAL_ORDER	LEFT_GRASP tomato_61 (to place in casserole)	Subgoal Sequencing Error: Attempted to move sliced ingredients to a container before the container itself was retrieved/prepared.
cleaning_bathtub_0	MISSING_STEP	CLEAN bathtub_35 — No soaked cleaner in inventory	Resource Dependency: Failed to sequence the prerequisite "Soak Cleaner" step before executing a "Clean Stain" action.
packing_adult_s_bags_0	MISSING_STEP	RIGHT_PLACE_INSIDE backpack_112 — backpack_112 is closed	State Dependency: Similar to the fridge error; the planner fails to verify the open state of a destination receptacle.
cleaning_the_hot_tub_0	AFFORDANCE_ERROR	RIGHT_GRASP pool_50 — Object pool_50 too big to grasp	Physical Constraint: Failure to account for the agent's morphology/capacity relative to the object's scale.

Table 8. Error Analysis of Subgoal Decomposition Module.

Task Name	Error Tpye	Evidence	Root Cause
cleaning_out_drawers_0	MISSING_STEP	step 0: RIGHT_GRASP bowl_0 — bowl_0 is inside bottom_cabinet_no_top_16	The planner attempted to interact with a nested object without first satisfying the open precondition of the container.
cleaning_up_after_a_meal_0	AFFORDANCE_ERROR	step 27: SOAK detergent_154	The agent attempted an action (SOAK) on an object (detergent) that lacks the physical affordance for that specific target state.
storing_the_groceries_0	AFFORDANCE_ERROR	step 0: TOGGLE_OFF fridge_57	Incorrect mapping of functional properties; the environment does not support toggle actions for refrigerators.
organizing_school_stuff_0	MISSING_STEP	step 3: RIGHT_PLACE_INSIDE backpack_108	Failed to open the destination receptacle (backpack) prior to the placement action.
cleaning_kitchen_cupboard_0	MISSING_STEP	step 9: RIGHT_PLACE_INSIDE top_cabinet_25	Similar to container access; placement into cupboards requires an explicit OPEN action first.

Table 9. Error Analysis of Transition Modeling Module.

Task Name	Error Mode	Evidence	Root Cause
sorting_groceries_0	Missed Effect	inside(carrot_2, fridge_97)	Containment Physics: Failed to update the location of objects placed into a container; the model only updated the agent's inventory.
chopping_vegetables_0	Hallucinated Effect	sliced(tomato_61)	Action-Condition Dependency: The model predicted a state change without verifying the required tool (knife) was correctly applied.
cleaning_bathrooms_0	Missed Effect	not stained(sink_7)	State Persistence Error: Model failed to remove the old state (stained) after the CLEAN action was successfully performed.
packing_adult_s_bags_0	Inertia Violation	onfloor(backpack_112, floor_0)	Frame Problem: Model maintained initial floor coordinates despite the object being successfully moved into the agent's inventory.

8.2.2. Common Sense

These rules are summarized in the common-sense task module. Figure 7 shows typical failure cases and corrected planning with semantic constraints.

[Rule-4] Semantic Tool/Container Compatibility. When a function-specific goal is required, the selected object must be semantically compatible and delivered to the specified destination. *Formal Logic:* $\forall f, d,\ \text{NeedFor}(f, d) \rightarrow \exists o\,(\text{Compatible}(o, f) \wedge \text{Goal}(\text{At}(o, d)))$. **Rationale:** Ensures object choice is functionally correct instead of only spatially reachable.

[Rule-5] Hold Before Place. Placing an object requires that the object is in hand. *Formal Logic:* $\forall o, d,\ \text{Place}(o, d) \rightarrow \text{Hold}(o)$. **Rationale:** Avoids invalid place attempts after failed pick or unintended state transitions.

[Rule-6] Relocation Execution Order. Common-sense relocation tasks must still respect the full manipulation sequence. *Formal Logic:* $\forall o, s, d,\ \text{Move}(o, s, d) \rightarrow \text{Nav}(s) \prec \text{Pick}(o) \prec \text{Nav}(d) \prec \text{Place}(o, d)$. **Rationale:** Maintains action validity while integrating semantic reasoning.

8.2.3. Visual Appearance

These rules are summarized in the visual-appearance task module. Figure 8 presents errors from attribute misbinding and the corrected attribute-grounded plans.

[Rule-7] Attribute-to-Object Binding. Attribute constraints must be grounded to a matched object before manipulation. *Formal Logic:* $\forall a,\ \text{Need}(\text{Attr} = a) \rightarrow$

(a) ep_1: Initial state (b) ep_1: Baseline failure (c) ep_1: Re^2 success

(d) ep_11: Initial state (e) ep_11: Baseline failure (f) ep_11: Re^2 success

Figure 6. Case visualizations for the `base` dimension.

$\exists o$ Match(o, a). **Rationale:** Prevents picking a wrong object class when multiple candidates exist.

[Rule-8] Proximity Before Pick. The agent must be near the selected visual target before picking. *Formal Logic:* $\forall o$, Pick$(o) \rightarrow$ Near(o). **Rationale:** Reduces repeated "not near" failures in visual search loops.

[Rule-9] Visual Relocation Macro. After attribute binding, transfer follows a fixed source-to-destination sequence. *Formal Logic:* $\forall o, d$, VisualMove$(o, d) \rightarrow$ Nav(src$(o)) \prec$ Pick$(o) \prec$ Nav$(d) \prec$ Place(o, d). **Rationale:** Couples perception-grounded object selection with executable manipulation.

8.2.4. Spatial Relationship

These rules are summarized in the spatial-relationship task module. Figure 9 shows failures from coarse location grounding and corrected target-specific navigation plans.

[Rule-10] Exact Receptacle Targeting. Placement must be executed near the exact target receptacle, not only its parent region. *Formal Logic:* $\forall o, r_t$, Place$(o, r_t) \rightarrow$ Near(r_t). **Rationale:** Enforces precise spatial grounding for relation-sensitive goals.

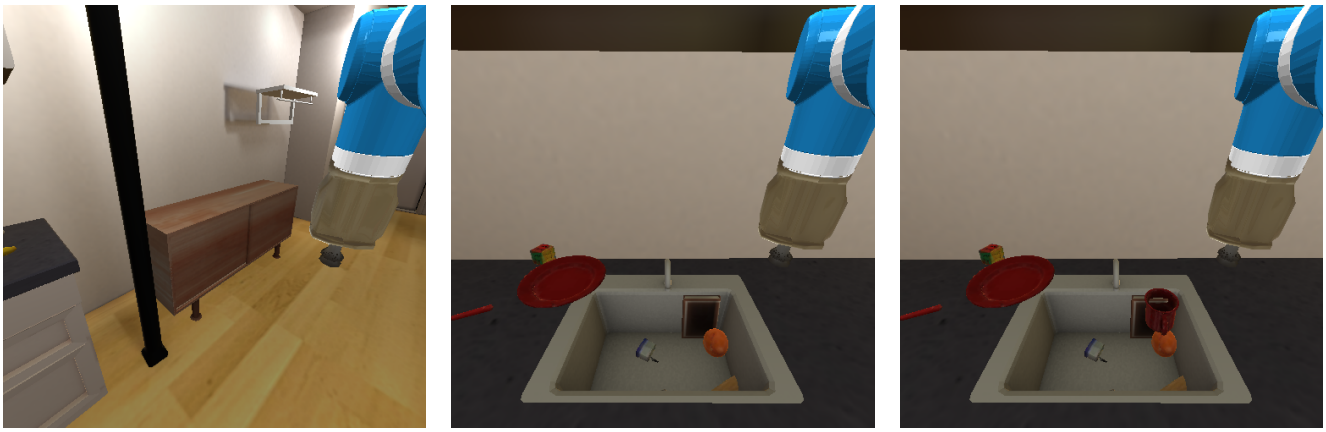
[Rule-11] Proximity Before Pick. Object pickup requires local proximity to the object. *Formal Logic:* $\forall o$, Pick$(o) \rightarrow$ Near(o). **Rationale:** Stabilizes spatial search by rejecting distance-invalid picks early.

[Rule-12] Spatial Relocation Order. Spatial relation tasks must still satisfy ordered relocate actions. *Formal Logic:* For MoveToSpatialTarget(o, d_s), the procedural constraint is Nav$(s) \prec$ Pick$(o) \prec$ Nav$(d_s) \prec$ Place(o, d_s). **Rationale:** Preserves both spatial constraints and manipulation feasibility.

8.2.5. Long Horizon

These rules are summarized in the long-horizon task module. Figure 10 highlights multi-step compounding errors and iterative recovery under rule-guided planning.

[Rule-13] Iterative Subgoal Completion. For multi-object tasks, each object must independently complete the full navigation-manipulation cycle. *Formal Logic:* $\forall o_i \in \mathcal{O}$, Nav$(s_i) \prec$ Pick$(o_i) \prec$ Nav$(d_i) \prec$ Place(o_i, d_i). **Rationale:** Prevents unfinished subgoals and supports stable



(a) ep_2: Initial state (b) ep_2: Baseline failure (c) ep_2: Re^2 success

(d) ep_20: Initial state (e) ep_20: Baseline failure (f) ep_20: Re^2 success

Figure 7. Case visualizations for the `common_sense` dimension.

long-sequence completion.

[Rule-14] Re-localize After Invalid Pick. After an invalid pick, the agent should re-localize instead of blindly repeating the same pick. *Formal Logic:* $\forall o$, InvalidPick$(o) \rightarrow$ Nav$(r_{new}) \wedge \neg$RepeatPickAtSameState(o). **Rationale:** Breaks error loops and improves recovery efficiency in long trajectories.

[Rule-15] Hold Before Place Across Subgoals. In every subgoal, placement requires holding the corresponding object. *Formal Logic:* $\forall o, d$, Place$(o, d) \rightarrow$ Hold(o). **Rationale:** Maintains action-state correctness throughout extended task horizons.

8.2.6. Complex Instruction

These rules are summarized in the complex-instruction task module. Figure 11 shows original misinterpretation traces and corrected plans after instruction-logic constraints are applied.

[Rule-16] Distractor Suppression in Contrastive Instructions. For "... but instead ..." instructions, the final objective must follow the target clause and exclude distractor clauses. *Formal Logic:* $\forall c_{ctx}, g_t$, Instr$(c_{ctx}$ but $g_t) \rightarrow$ Goal$(g_t) \wedge \neg$Goal(c_{ctx}). **Rationale:** Avoids acting on contextual distractors rather than the intended target.

[Rule-17] Conditional Branch Grounding. Instruction branches must be selected according to observed world state. *Formal Logic:* $(\text{Open}(fridge) \rightarrow g_1) \wedge (\neg\text{Open}(fridge) \rightarrow g_2)$. **Rationale:** Aligns branch execution with environment conditions for conditional instructions.

[Rule-18] Proximity Before Pick in Ambiguous Contexts. Even under ambiguity, pick actions require local proximity to the chosen object. *Formal Logic:* $\forall o$, Pick$(o) \rightarrow$ Near(o). **Rationale:** Combines instruction-level disambiguation with low-level action validity.

9. Prompt Templates

This section presents the exact prompt templates used in Re^2, formatted as reproducible prompt cards. The prompts correspond to the main stages of the framework, including reflective rule induction, rule consolidation, and rule-guided

(a) ep_11: Initial state (b) ep_11: Baseline failure (c) ep_11: Re^2 success

(d) ep_32: Initial state (e) ep_32: Baseline failure (f) ep_32: Re^2 success

Figure 8. Case visualizations for the `visual_appearance` dimension.

refinement in Figure 12 and 13. For each prompt, we provide its source, intended role, structural constraints, and expected output format, so as to facilitate faithful reproduction and practical deployment of the framework.

(a) ep_8: Initial state (b) ep_8: Baseline failure (c) ep_8: Re^2 success

(d) ep_44: Initial state (e) ep_44: Baseline failure (f) ep_44: Re^2 success

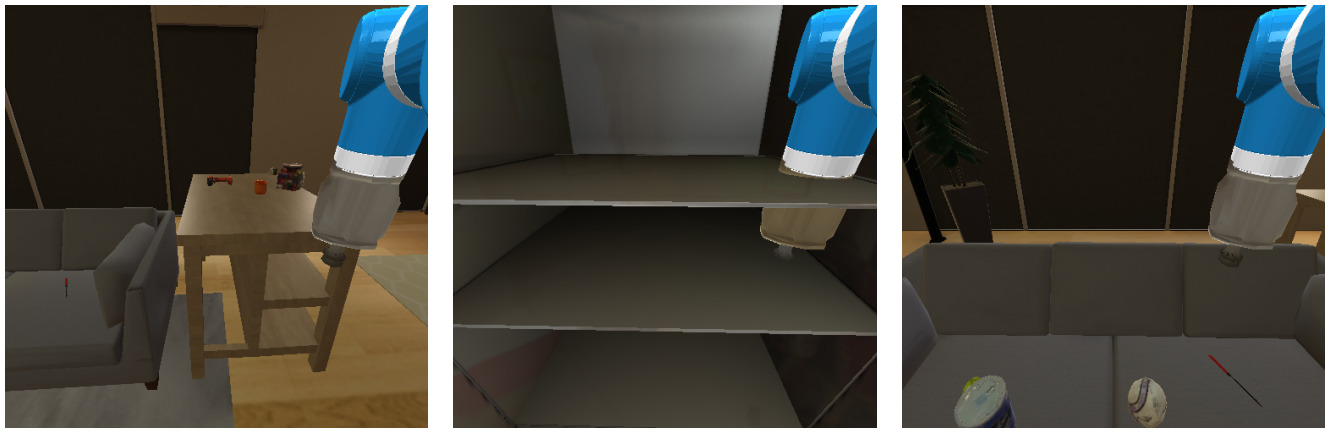
Figure 9. Case visualizations for the `spatial_relationship` dimension.

(a) ep_16: Initial state (b) ep_16: Baseline failure (c) ep_16: Re^2 success

(d) ep_48: Initial state (e) ep_48: Baseline failure (f) ep_48: Re^2 success

Figure 10. Case visualizations for the `long_horizon` dimension.

(a) ep_1: Initial state (b) ep_1: Baseline failure (c) ep_1: Re^2 success



(d) ep_11: Initial state (e) ep_11: Baseline failure (f) ep_11: Re^2 success

Figure 11. Case visualizations for the `complex_instruction` dimension.

System Prompt: Rule Extraction

Purpose: Extract 1–2 high-quality *general strategic principles* from a robot episode trace. The extracted rules should be abstract, reusable, and expressed in both natural-language and symbolic forms.

Instruction to the LLM:

Extract 1 to 2 high-quality GENERAL STRATEGIC PRINCIPLES from this robot episode trace.

Requirements:

- Return abstract, reusable rules.
- Do not mention specific object names, rooms, receptacles, or surfaces.
- Do not mention low-level action commands or action IDs.
- Prefer concepts such as proximity, possession, preconditions, sequencing, source location, destination, and retry conditions.
- Each rule should describe a reusable strategy or precondition, rather than an instance-specific command.
- For each rule, provide both natural-language strategy and symbolic metadata.
- Output only the final JSON object; do not output analysis, reasoning, or markdown.

Required JSON format:

```
{"rules":[{"rule_text":"...", "trigger_pattern":"short_snake_case_category",
"task_categories":["relocation","retrieval","state_change","tidying","general"],
"init_score":0.0,
"symbolic_conditions":{"task_type":"relocation"},
"symbolic_effect":"source_then_destination",
"applicable_actions":["navigate","pick","place"]}}]
```

Constraints:

- Generate 1 to {`MAX_RULES`} rules.
- `trigger_pattern` should be a short category label, e.g., `precondition_check` or `proximity_first`.
- `task_categories` indicates the task types to which the rule applies.
- `init_score` must lie in $[0.3, 1.0]$.
- `symbolic_effect` should be a short snake_case effect label.

Runtime Variables:

- {`INSTRUCTION`}: task instruction
- {`TASK_SUCCESS`}: episode success flag
- {`EPISODE_SUMMARY`}: summarized episode trace

Figure 12. Prompt template used for rule extraction in the Re_1 stage.

System Prompt: Rule Consolidation

Purpose: Consolidate the current rule library into a small set of high-quality, abstract strategic rules by removing instance-specific rules and merging semantically redundant ones. Each retained rule should preserve both natural-language and symbolic forms.

Instruction to the LLM:

> **You are maintaining a robot planning rule library. Your goal is to produce a SMALL set of HIGH-QUALITY, ABSTRACT strategic rules.**

Critical Requirements:

- Remove any rule that mentions specific object names (e.g., sofa, mug, pear, counter).
- Remove any rule that is merely a specific action command (e.g., navigate to X, pick up Y).
- Merge semantically similar rules into one concise abstract principle.
- Each rule must describe a general strategy applicable across many tasks.
- Preserve `task_categories` for each rule.
- Preserve and normalize symbolic metadata so that each rule has both natural-language and symbolic forms.

Required JSON format:

```
{"rules":[{"rule_text":"...", "trigger_pattern":"snake_case",
"score_hint":0.0, "task_categories":["general"],
"symbolic_conditions":{"task_type":"general"},
"symbolic_effect":"generic_strategy",
"applicable_actions":["navigate"]}}]
```

Constraints:

- `rule_text`: a short imperative sentence describing a general strategy.
- `trigger_pattern`: a stable category label.
- `score_hint` must lie in $[-3.0, 5.0]$.
- `symbolic_effect`: a short snake_case effect label.
- The output must not contain duplicated rules.

Runtime Variables:

- {N}: total number of input rules
- {RULE_PAYLOAD_JSON}: serialized input rule set

Instantiated Input Fields:

Input rules ({N} total):
{RULE_PAYLOAD_JSON}

Figure 13. Prompt template used for rule consolidation and structural merging in the rule library.