Re^2 Agent: Reflection and Re-execution Agent for Embodied Decision Making

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

Accurate and efficient decision-making is essential for robots operating in open-world embodied environments. While large language models (LLMs) have shown promise in generating task plans from instructions, existing single-shot prompting methods often fail to adapt to dynamic constraints and environmental feedback during execution. In this work, we propose a **Reflective** and **Re-execution** (**Re^2**) agent that integrates prior knowledge-driven static prompting with task rule extraction for iterative refinement. Starting from an initial knowledge-grounded prompt, Re^2 agent executes an action sequence, reflects on environmental feedback, abstracts failure patterns into rules, and re-prompts itself with refined constraints. This closed-loop reasoning process enhances robustness and leads to progressive improvement in task performance. We evaluate our method on the Embodied Agent Interface (EAI) benchmark, achieving an average score of **81.36** (**86.35** on BEHAVIOR and **76.36** on VirtualHome), demonstrating the effectiveness of our approach in bridging the gap between language reasoning and grounded action. The code is available at https://github.com/chenyang126/Re-2-Agent.

1 Introduction

Accurate and efficient decision-making is critical for robots to accomplish diverse tasks in open-world environments. In recent years, a growing body of work has explored how agents can follow task descriptions and human instructions to generate action sequences and achieve specific decision-making goals in various embodied scenarios [1, 3, 6, 10] .

Embodied decision-making differs from open-ended QA tasks: while the outputs of QA systems can be diverse [5, 21, 24], embodied outputs must be executable and comply with either simulator configurations or real-world physical constraints [13, 25]. To adapt to task requirements in embodied environments, an agent must account for functional affordances, spatial layouts, and temporal dependencies, ensuring that each action satisfies its preconditions and effects.

Early research primarily focused on mapping natural language instructions to grounded actions in simulated environments, using platforms such as BEHAVIOR [18] or VirtualHome [28] to study navigation and household activities. These systems typically relied on structured programs or symbolic planners to bridge high-level goals with low-level control [14, 8, 20]. Large language models (LLMs) have recently emerged as an effective approach for interpreting task instructions and generating task plans. Initial attempts to apply LLMs in embodied settings often used single-shot prompting to produce complete plans [17, 10]. While suitable for simple tasks, such prompt-based methods struggle to adapt to dynamically changing environmental observations and complex

*Corresponding author: guolz@lamda.nju.edu.cn constraints during task execution. Frameworks like ReAct [26] and AutoGen [22] introduced iterative reasoning loops that interleave thought and action, leveraging environmental feedback to dynamically refine plans.

Therefore, we explore an approach that integrates prior knowledge-driven static prompting with task rule-driven interactive agent techniques. Despite recent progress, most existing embodied AI benchmarks still rely primarily on final task success rates as evaluation metrics, lacking detailed records of the execution process [25, 15, 7]. It is insufficient for analyzing multi-step reasoning, error recovery, and dynamic adaptation capabilities, posing significant challenges for deploying agents in embodied environments. The introduction of the Embodied Agent Interface (EAI) benchmark [13] addresses this gap: the EAI benchmark incorporates richer evaluation metrics, systematically records execution feedback, and supports fine-grained analysis of reasoning steps and failure modes. This benchmark provides a practical foundation for deploying agents capable of learning from feedback and continuously optimizing their policies.

Building on this foundation, we propose a Reflective and Re-execution (Re^2) capable agent tailored for embodied decision-making. Our method begins by constructing an initial prompt based on prior knowledge, ensuring the output aligns with basic task constraints. After executing the initial plan, Re^2 agent collects environmental feedback, abstracts task-specific rules, and incorporates them into the subsequent prompt design. Through reflection on observed errors, Re^2 agent enhances its re-execution ability, dynamically balancing environmental feedback with task constraints to progressively improve decision robustness and accuracy. This iterative mechanism not only bridges the gap between language understanding and grounded action but also, in combination with EAI's fine-grained evaluation, helps uncover the root causes of reasoning-execution misalignment. Through continuous iterative refinement, Re^2 agent effectively achieves the final task goals. We highlight two key contributions of this work: (i) the design of an agent framework that integrates **prior knowledge-driven reasoning with task rule extraction** for decision-making in embodied scenarios; (ii) successful deployment of **a reflective and re-execution capable agent** on the EAI benchmark, achieving average performance scores of **86.35** on BEHAVIOR and **76.36** on VirtualHome, yielding an overall performance of **81.36**, which demonstrates the effectiveness of our approach.

2 Related Work

Embodied Benchmark. Recent advances in simulation environments and benchmarks have significantly accelerated research in embodied AI. Embodied platforms [18, 28, 11, 12] provide interactive indoor environments for task execution and agent interaction. Building on these platforms, numerous benchmarks have been introduced to evaluate the decision making capabilities of LLMs in embodied settings. Existing works [15, 7, 27, 29, 9] offer systematic evaluations across navigation, manipulation, and long-horizon planning tasks. However, these benchmarks are often designed for specific objectives, limiting their ability to comprehensively assess model performance. To address this, unified evaluation platforms [13, 25, 16] integrate multiple benchmarks and standardize capability categorization, providing a robust foundation for decision-making assessment in embodied scenarios.

Embodied Decision Making. Most existing approaches evaluate LLMs in embodied settings using static, single-step prompts, where the model generates a complete plan without iterative refinement or feedback integration [10, 17]. Recent agent frameworks [19, 4, 2, 23] address this limitation by introducing interactive reasoning loops that leverage environmental feedback to dynamically adjust plans. Building on these foundations, our work design Re^2 agent agent, which integrates feedback and error summarization into an iterative optimization framework for embodied decision making.

3 Method

The EAI [13] benchmark comprises two simulation platforms with distinct characteristics—BEHAVIOR and VirtualHome—and covers four interrelated yet independent capability modules: goal interpretation, subgoal decomposition, action sequencing, and transition modeling. Based on existing prompt

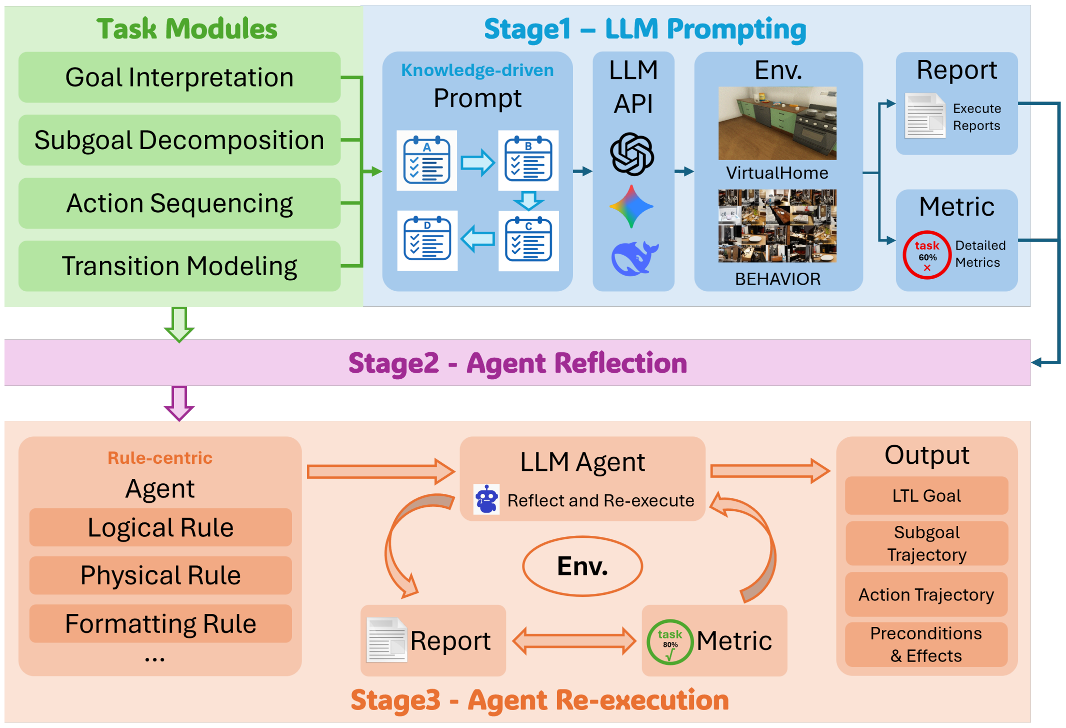

Figure 1: Overview of the Re^2 Agent framework. The system operates in three stages: (1) knowledge-driven prompt design; (2) reflective of execution reports and metrics to derive task rules; and (3) re-execution of the task through environmental interaction to produce the final output.

templates, we design and optimize a knowledge-driven prompting scheme that captures environment and task characteristics. Additionally, EAI provides a set of fine-grained metrics that categorize errors into specific types, such as hallucination errors, affordance errors, and various planning errors. These detailed metrics, along with execution reports, bridge the interaction between the agent and the environment. By analyzing task metrics and execution reports through retrospection and reflection, the agent extracts task-relevant rules. It then injects these distilled rules to re-execute tasks and refine its outputs, enabling closed-loop evaluation in an embodied environment. The overall framework is illustrated in Figure 1. The following sections analyze and discuss the knowledge-driven prompting strategies and the rule-centric agent design for the four tasks.

3.1 Goal Interpretation

Knowledge-Driven Prompt Design. We formalize goal interpretation as minimal end-state synthesis under hard constraints. The prompt instructs the agent to read the natural language goal and scene context, then output a deterministic JSON with three lists: `node goals`, `edge goals`, and `action goals`. Our design adopts relation-first modeling for containment and placement tasks, and a clear state-action separation that prefers state goals for device control and uses action goals only when states cannot capture intent. A minimization principle removes intermediate states and contradictions, which improves executability and reduces error propagation. This yields compact specifications that align with simulator constraints and produce stable outputs for downstream evaluation.

Rule-Centric Agent Design. We couple the generator with a corrector that uses reflective feedback and temporal logic. The agent consumes the original goal, the previous prediction, and either a ground-truth example or a similar reference, then produces a corrected JSON and an LTL-like formula aligned with the specification. Execution reports and success metrics from the benchmark drive rule induction and correction, which improves consistency with environment constraints and task semantics. The agent re-executes the task with injected rules, analyzes outcomes, and refines its outputs. This closes the loop and leads to higher success rates and cleaner specifications.

3.2 Subgoal Decomposition

Knowledge-Driven Prompt Design. Our subgoal decomposition method converts a high-level goal state g into an ordered sequence of intermediate subgoals $\bar{\phi} = \{\phi_i\}_{i=1}^{m}$, where each ϕ_i is a boolean expression over first-order predicates. The design introduces the **Critical Preconditions Checklist**, which enforces constraint satisfaction during decomposition to prevent common failures in LLM-based planning. Three constraint categories are prioritized: (1) **Container Accessibility** ensures that any `inside(obj, container)` transition is preceded by `open(container)`; (2) **Tool Activation** requires explicit activation sequences for tool-mediated operations, such as cleaning, to avoid invalid assumptions of operability; (3) **Instrument Requirements** mandate that property-modifying actions (e.g., slicing) occur only when the agent holds the appropriate instrument. These constraints are encoded in structured prompts, guiding the model to validate preconditions before generating each subgoal.

Rule-Centric Agent Design. We enforce physical and temporal constraints as hard rules: the agent has two manipulators, each holding at most one object, and operations like `open`, `clean`, and `slice` require a free hand, creating implicit dependencies. The output sequence satisfies prerequisite ordering, resource availability, and goal consistency, ensuring that the final subgoal meets all conditions in g. The prompt encodes prioritized rules, with container accessibility as the highest priority, and outputs subgoals as JSON: `{"output": [`ϕ_1`, ..., `ϕ_m`]}`, with quantifiers resolved to concrete object instances. Subgoal granularity balances atomic transitions with compound boolean expressions using `and`/`or`, avoiding both omission of critical states and excessive fragmentation. This rule-centric design enables systematic validation and correction during generation, improving reliability and reducing planning errors.

3.3 Action Sequencing

Knowledge-Driven Prompt Design. Given an initial state s_0 and target state g, the action sequencing module generates an executable sequence $\bar{a} = \{a_i\}_{i=1}^{n}$ that transforms the environment from s_0 to g. We adopt a one-shot autoregressive generation strategy using LLMs, where the complete sequence is produced in a single forward pass conditioned on s_0, g, and the set of interactable objects $\mathcal{O}$. The prompt enforces constraint satisfaction through a multi-level specification mechanism that encodes three categories: (1) **Physical constraints** such as hand occupancy limits and requirements for free manipulators; (2) **Temporal constraints** ensuring correct ordering, e.g., containers must be opened before placing objects inside; (3) **Logical constraints** requiring action-specific preconditions, such as `SLICE` needing a knife or `CLEAN` requiring a cleaning tool. These constraints are hierarchically structured in the input representation to guide reasoning and validate preconditions before generation.

Rule-Centric Agent Design. To ensure syntactic and semantic validity, the output format is a Python-evaluable list of dictionaries with strict prohibitions on explanatory text or markdown. Canonical examples are included in the prompt to bias the model toward correct syntax. Generated sequences undergo a three-stage validation pipeline: (1) **Format validation** for structural correctness; (2) **Hallucination detection** to confirm that actions and objects belong to valid sets $\mathcal{A}$ and $\mathcal{O}$; (3) **Argument validation** for parameter consistency. Executability is verified through trajectory evaluation against the transition model $\mathcal{M}$, where actions are executed sequentially. Failures are categorized as missing steps, extra steps, incorrect order, or affordance violations. When infeasible actions are detected, execution terminates early, enabling potential replanning from the current state. This rule-centric design ensures robust sequencing under domain constraints and minimizes planning errors.

3.4 Transition Modeling

Knowledge-Driven Prompt Design. Transaction Modeling targets *action-level synthesis* under typed domain predicates and PDDL semantics. Unlike other modules, which outputs minimal end-state JSON, this module generates executable PDDL actions whose `:precondition` and `:effect` guarantee that the initial state reaches the goal. The prompt constrains predicate usage to the domain's typing and arity, requires preconditions in Disjunctive Normal Form, and encodes effects with `and`, `when`, and `forall` when needed. It emphasizes spatial and containment dynamics through predicates such as `obj_inside`, `obj_ontop`, `inside_room`, and hand-state transitions via `holds_rh` and `holds_lh`. The design avoids intermediate-state leakage and enforces action-local consistency: for placement tasks use containment or support relations in effects; for device control pair `on` with `plugged_in` when the domain does not guarantee power; for movement use `next_to` updates rather than implicit teleportation. On the *Wash_clothes* task, actions such as `grab`, `open`, `put_on` (placing items on the washing machine), `close`, and `switch_on` are defined so that soap and clothes end up `obj_ontop` the washing machine, which is `closed`, `on`, and `plugged_in`. This yields compact, executable operators that reflect container usage, hand transitions, and device states.

Rule-Centric Agent Design. We pair the generator with a verifier-corrector that aligns actions with a reference logic standard and repairs model outputs. The verifier inspects candidate actions against the domain's patterns, fixes illegal quantification in effects by replacing `exists` with `forall` when appropriate, removes hallucinated objects, and adds missing guards that improve recall, such as checking closed containers before `grab` and validating proximity with `next_to`. It also prunes effects for observation-only actions to boost precision. Given task info and a reference logic, the agent emits corrected PDDL in the requested action schema, preserving names and parameters while enforcing typing and predicate consistency. Execution reports and goal satisfaction serve as feedback for iterative repair: when failures occur, the verifier tightens preconditions or relaxes over-constrained effects, then re-runs the actions. This closed loop increases action validity, reduces planner crashes, and improves end-to-end goal attainment on Transaction Modeling tasks.

3.5 Environment-Specific Adaptations

Our approach is designed to operate across heterogeneous embodied environments by leveraging a unified Reflection and Re-execution agent. The core idea—learning from interaction and execution feedback to refine planning—remains consistent in both **BEHAVIOR** and **VirtualHome**. However, due to differences in task structure, action spaces, and object ontologies, we introduce environment-specific adaptations to ensure robust performance.

Action Space and Object Ontology. BEHAVIOR provides a rich set of atomic actions (e.g., grasp, place, open) and realistic object types with fine-grained affordances, whereas VirtualHome represents tasks as high-level programs composed of symbolic actions (e.g., `Grab(obj)`, `SwitchOn(obj)`). To bridge this gap, we implement an *action vocabulary mapping* layer that normalizes environment-specific actions into a unified representation during planning and converts them back to environment-native formats during execution. Similarly, object names and categories are aligned through an ontology mapping table to handle synonyms and hierarchical differences (e.g., "mug" in VirtualHome vs. "cup" in BEHAVIOR).

Scene Complexity and Feedback Integration. BEHAVIOR features photorealistic, cluttered scenes with complex spatial constraints, while VirtualHome offers simplified layouts and deterministic object states. To adapt, our agent employs environment-specific preprocessing: in BEHAVIOR, we extract spatial graphs and collision-aware constraints to inform prompt generation, whereas in VirtualHome, we leverage program templates to enforce syntactic correctness. Post-processing also differs: BEHAVIOR requires validating reachability and grasp feasibility, while VirtualHome checks program executability within its simulator.

In summary, while the Re^2 agent paradigm is shared, our method incorporates lightweight environment-specific modules for action normalization, object alignment, and constraint validation.

Table 1: Results (%) overview. *V*: VirtualHome, *B*: BEHAVIOR..

Model	Overall Perf.	**Average Perf.**		**Goal Interpretation**		**Action Sequencing**				**Subgoal Decomposition**				**Transition Modeling**			
		Module SR		F_1		*Task SR*		*Execution SR*		*Task SR*		*Execution SR*		F_1		*Planner SR*	
		V	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*	*V*	*B*
GPT-4o	52.25	49.41	55.09	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	64.43	61.75	67.10	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
GPT-5	68.32	68.09	68.55	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] Prompt	74.54	71.58	77.50	49.0	84.9	71.2	73.0	84.9	78.0	74.7	80.0	**88.0**	86.6	54.8	43.0	91.4	97.0
Re[2] Agent	**81.36**	**76.36**	**86.35**	**57.9**	**85.0**	**73.5**	**81.0**	**84.9**	**91.0**	**74.7**	**80.0**	86.6	**88.0**	**98.8**	**99.8**	**99.9**	**99.0**

These adaptations enable consistent performance across environments with markedly different task representations and scene complexities.

4 Implementation Details

We adopt closed-source API calls and primarily evaluate GPT-series models, with GPT-5 as the main planner. The main computational cost lies in API token usage. Given the long-horizon nature of tasks, we set the maximum context length to **8192** tokens. Each task is allowed up to **five attempts** to prevent output freezing, and reflective summaries are appended after each failure to enable re-execution with improved constraints.

Preprocessing involves prompt restructuring to ensure initial outputs meet basic task requirements, followed by context augmentation with execution feedback and learned rules. Post-processing validates format and task feasibility, applying rule-based corrections before retry. All validation and adaptation steps—such as action mapping, object ontology alignment, and output format conversion—are implemented through explicit rules provided to the agent. We do not perform local training; inference runs entirely via API. Runtime depends on context length and retries, while all configuration files (prompt template, model endpoints, retry policy, and validation rules) are maintained for reproducibility.

5 Evaluation Results

We evaluated models from the GPT series, including GPT-4o, GPT-o3, and GPT-5. As shown in Table 1, while the performance metrics on the EAI tasks improved with enhanced model capabilities, the results still fell short of task requirements, indicating an overall unsatisfactory performance. By incorporating optimized prompts, the model performance saw significant improvement. However, the Re[1] Prompt model continued to perform poorly on certain tasks, suggesting that prompt optimization alone is insufficient to address the specific rules required across all task scenarios. To overcome this limitation, we introduced the Re[2] Agent, which led to a substantial performance gain in both goal interpretation and transition modeling tasks.

6 Analysis and Discussion

We conducted separate analyses and discussions for the four tasks in both the BEHAVIOR and VirtualHome environments to examine the strengths of our approach and identify potential areas for improvement. We analyze four core tasks: *Action Sequencing*, *Goal Interpretation*, *Subgoal Decomposition*, and *Transition Modeling*. Our study focuses on success rates, error patterns, and systematic limitations. Note that the analysis is based on outputs from a single stage—the development phase (i.e., for VirtualHome, only results from Scene 1 were considered).

6.1 BEHAVIOR Analysis

Action Sequencing. This task achieves a task success rate of 81% and execution success of 91%, with goal satisfaction rates of 89.5% (state) and 84.02% (relation). Success is common in simple object-placement tasks, while failures arise from missing steps (6%), additional steps (3%), and affordance errors (2%). Complexity strongly correlates with error rates: multi-object, multi-container tasks drop below 70%, and sequences longer than 20 steps show increased error accumulation.

Goal Interpretation. The model reaches an overall F1 score of 85.03% (precision: 83.62%, recall: 86.48%) with zero grammatical or hallucination errors. Errors stem from unsatisfied conditions (13.1%) and false positives (16.4%), especially in negation and compound logic. Relation goals exhibit higher error rates (16.5%) than state goals (3.3%). Performance declines with increasing object count and logical complexity.

Subgoal Decomposition. Task success is 80%, execution success 88%, with relation goals outperforming state goals (88.35% vs. 80%). Temporal ordering errors (6%) and additional steps (16%) dominate, indicating difficulty in dependency reasoning. Performance drops below 70% for tasks with four or more subgoals, especially when dependencies exist.

Transition Modeling. This task achieves an F1 score of 91.90% (precision: 94.61%, recall: 89.35%), with non-spatial relations performing best (F1: 94.95%). Spatial relations show lower recall (85.31%), indicating missed implicit changes.

Common failure patterns include insufficient state perception, weak temporal reasoning, limited affordance knowledge, and poor handling of negation and compound logic. Performance correlates with task complexity, object diversity, and environmental constraints (e.g., closed containers). Addressing these issues requires enhanced state tracking, improved dependency modeling, richer affordance representation, and better logical reasoning.

We fed part of the execution results back into the Re^2 Agent for reflection, enabling the model to learn task-specific rules and re-execute the plan. This process significantly improved overall performance.

6.2 VirtualHome Analysis

Action Sequencing. Task success reaches 78.36% and execution success 79%, with goal satisfaction rates of 93.53% (state), 82.78% (relation), and 64.86% (action). Missing step errors dominate (18.69%), mainly due to omitted navigation steps before object manipulation, revealing weak spatial reasoning. Hallucination errors (2.30%) stem from non-existent actions, while additional steps (2.95%) are rare. Performance declines sharply in navigation-heavy tasks and complex multi-step operations.

Goal Interpretation. Overall F1 score is 57.58% (precision: 52.76%, recall: 63.38%), far below BEHAVIOR dataset performance. Node prediction (F1: 57.51%) and action prediction (F1: 54.55%) lag behind edge prediction (F1: 59.20%). High recall but low precision indicates over-prediction, especially for complex semantic structures. Results suggest limited understanding of VirtualHome's abstract relationships and environment-specific rules.

Subgoal Decomposition. This task achieves the best performance: 89.94% task success and 93.49% execution success. However, additional step errors (16.57%) dominate, caused by redundant operations such as repeated FIND or unnecessary PLUGIN actions. Affordance errors (0.3%) and hallucinations (0.027%) are rare. Performance drops with increasing subgoal complexity, indicating incomplete state tracking and imprecise subgoal boundaries.

Transition Modeling. F1 score is 44.61% (precision: 54.10%, recall: 37.95%), yet planner success reaches 83.45%. Object states (F1: 55.22%) and affordances (F1: 60.75%) outperform spatial relations (F1: 40.62%) and object orientation (F1: 26.09%). Non-spatial relations perform worst (F1: 6.68%). Severe under-prediction limits transition accuracy, but coherent partial predictions enable feasible plans.

VirtualHome tasks exhibit distinct failure modes: missing navigation steps, redundant actions, hallucinated operations, and weak semantic reasoning. Performance strongly correlates with task complexity and environment-specific constraints. Addressing these issues requires improved navigation modeling, enhanced state tracking, richer action space knowledge, and stronger semantic understanding for virtual environments.

In practice, because the final evaluation stage allows corrections using complete data from the first stage, we performed secondary summarization and analysis of outputs and log files from the initial phase. This approach successfully improved model performance in the final evaluation stage, further validating the effectiveness of the Re^2 agent architecture.

6.3 Comparison of BEHAVIOR and VirtualHome

We compare performance and error patterns across BEHAVIOR and VirtualHome to identify commonalities and differences in model capabilities. Both environments share similar error types but differ significantly in semantic complexity and navigation requirements.

Common Patterns. Both environments exhibit strong grammatical correctness ($< 0.03\%$ errors), high state goal achievement (BEHAVIOR: 89.5%, VirtualHome: 93.53%), and comparable relation goal performance (84.02% vs. 82.78%). Common errors include missing steps, additional steps, and affordance violations, with error rates increasing under task complexity. State perception limitations and dependency reasoning challenges persist across both environments.

Performance Differences. Goal interpretation shows a large gap: BEHAVIOR achieves F1: 85.03%, while VirtualHome drops to 57.58%, reflecting VirtualHome's complex semantics. Transition modeling contrasts sharply: BEHAVIOR reaches F1: 91.90%, whereas VirtualHome achieves F1: 44.61% yet planner success of 83.45%, indicating different trade-offs between accuracy and coherence. Subgoal decomposition reverses the trend, with VirtualHome outperforming BEHAVIOR (89.94% vs. 80%).

Error Pattern Differences. Missing step errors are three times higher in VirtualHome (18.69% vs. 6%), driven by explicit navigation requirements. Hallucination errors appear in VirtualHome (2.30%) but are nearly absent in BEHAVIOR, suggesting weaker action space understanding. Temporal ordering errors affect BEHAVIOR (6%) but not VirtualHome, indicating better dependency handling in VirtualHome despite other limitations.

Environmental Characteristics. VirtualHome demands explicit navigation and includes a more complex action space, leading to hallucinations and navigation-related failures. Its semantic structures involve abstract non-spatial relations with critically low F1 (6.68%), contrasting with BEHAVIOR's simpler spatial relations (F1: 89.52%). Planning capability differs markedly: BEHAVIOR produces accurate but incoherent predictions, while VirtualHome yields incomplete yet coherent plans.

Summary. Both environments share systematic biases, but VirtualHome's navigation and semantic complexity introduce unique challenges. VirtualHome excels in planning coherence and subgoal decomposition but struggles with semantic interpretation and transition modeling. These findings highlight the need for environment-adaptive strategies to handle varying navigation requirements, action space complexity, and semantic abstraction.

7 Conclusion

Our work establishes that iterative reflection and rule abstraction are critical for robust embodied decision making. Evaluated on the Embodied Agent Interface benchmark, our Re^2 agent achieves a combined score of 81.36 (86.35 on BEHAVIOR, 76.36 on VirtualHome), significantly outperforming non-iterative baselines and demonstrating effective closing of the gap between language understanding and grounded action execution. Looking ahead, we aim to build upon the Re^2 agent framework by collecting more environmental interaction data, thereby truly integrating the learning capabilities of LLMs with dynamic feedback from embodied scenarios to achieve closed-loop responsiveness in open-world settings.

Reproducibility Statement

Our project code is available at `https://github.com/chenyang126/Re-2-Agent`, including the optimized prompt templates from the first stage and the task rules abstracted through the agent's reflection in the second stage. Corresponding deployment details are provided in the GitHub repository.

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.

[2] Mohamed Salim Aissi, Clémence Grislain, Mohamed Chetouani, Olivier Sigaud, Laure Soulier, and Nicolas Thome. Viper: Visual perception and explainable reasoning for sequential decision-making. *arXiv preprint arXiv:2503.15108*, 2025.

[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.

[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.

[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.

[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.

[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.

[8] Rohan Chitnis, Tom Silver, Joshua B Tenenbaum, Tomas Lozano-Perez, and Leslie Pack Kaelbling. Learning neuro-symbolic relational transition models for bilevel planning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4166–4173. IEEE, 2022.

[9] 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.

[10] 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. PMLR, 2022.

[11] 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.

[12] 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.

[13] 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.

[14] Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

[15] 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.

[16] Fei Ni, Min Zhang, Pengyi Li, Yifu Yuan, Lingfeng Zhang, Yuecheng Liu, Peilong Han, Longxin Kou, Shaojin Ma, Jinbin Qiao, et al. Embodied arena: A comprehensive, unified, and evolving evaluation platform for embodied ai. *arXiv preprint arXiv:2509.15273*, 2025.

[17] 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.

[18] 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. PMLR, 2022.

[19] 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.

[20] Lionel Wong, Jiayuan Mao, Pratyusha Sharma, Zachary S Siegel, Jiahai Feng, Noa Korneev, Joshua B Tenenbaum, and Jacob Andreas. Learning adaptive planning representations with natural language guidance. *arXiv preprint arXiv:2312.08566*, 2023.

[21] 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.

[22] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.

[23] Weimin Xiong, Yifan Song, Qingxiu Dong, Bingchan Zhao, Feifan Song, Xun Wang, and Sujian Li. Mpo: Boosting llm agents with meta plan optimization. *arXiv preprint arXiv:2503.02682*, 2025.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] Xiaoxue Zeng et al. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[29] 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.

A Biography of all team members

Team: nju-lamda12

Our team consists of researchers from the State Key Laboratory of Novel Software Technology at Nanjing University. The members are:

- **Yang Chen**, **Jie-Jing Shao**, **Xiao-Wen Yang**, and **Ming Yang** are Ph.D. students.
- **Hong-Jie You** is a Master's student.
- **Yu-Feng Li** is a Professor in the School of Artificial Intelligence. His main research interests lie in machine learning and data mining.
- **Lan-Zhe Guo** is an Assistant Professor in the School of Intelligence Science and Technology. His research focuses on artificial intelligence and machine learning, with a long-term goal of enhancing the reasoning and planning capabilities of AI models in both digital and physical worlds.