Evaluating LLM Planning in Partially Observable Environments via Observation Representations and Action Sequences

Hayeong Lee[1*] **Jun Ho Seo**[1*] **Sunguk Shin**[1*] **Jinho Lee**[1]
Myunsoo Kim[1] **Minsuk Chang**[2] **Byung-Jun Lee**[1,3]
[1]Korea University [2] Google DeepMind [3]Gauss Labs Inc.
{hayeong_lee,junhoseo,ssw1419,jinho0997,m970326,byungjunlee}@korea.ac.kr
minsukchang@google.com
*Equal contribution

Abstract

Recent evaluation of large language models (LLMs) has increasingly shifted from static, single-turn benchmarks to interactive environments that demand sequential decision-making, long-term planning, and adaptation. LLM-as-agents show strong potential in these settings, leveraging broad pretraining for generalizable planning and offering more interpretability than traditional reinforcement learning methods. However, their core reasoning abilities remain contested, with evidence of limitations in logical consistency and a tendency toward pattern matching over causal inference. To probe these challenges, we study LLM planning in partially observable environments that require reasoning under uncertainty. We propose two strategies to assess and enhance their capabilities: (i) evaluating three types of observation representations: natural language, structured symbolic, and a hybrid format that combines both; and (ii) prompting LLMs to generate extended action sequences per decision step to exploit their long-horizon planning capacity. These approaches aim to clarify the extent to which LLMs can reason, plan, and act effectively in the face of partial observability. Our code is available at: https://github.com/ku-dmlab/llm-planning-po.

1 Introduction

While early evaluations of large language models (LLMs) primarily focused on one-off tasks such as question answering [28], mathematics [15], and code generation [5], recent attention has shifted toward assessing their performance in complex, iterative interactions within real-world environments [32, 26, 21]. These environments provide a dynamic and interactive testbed, crucial for evaluating an agent's ability to perform sequential decision-making, long-term planning, and adaptation in response to environmental feedback. Unlike static natural language processing benchmarks, sequential decision-making tasks allow LLM agents to demonstrate their capacity for exploration, information acquisition, and strategic refinement within a simulated world, leveraging their inherent language understanding to interpret observations and formulate actions. LLM-as-agents, such as those proposed by [21, 40], have demonstrated strong performance in such complex environments, attributed to enhanced reasoning and decision-making capabilities.

These examples underscore a key advantage of LLM-as-agents: their generalizability, enabling them to tackle novel tasks and operate in complex environments without extensive task-specific training, unlike traditional reinforcement learning (RL) methods. While RL agents have achieved success in domains like StarCraft II through specialized training [39], LLMs leverage broad pretraining on

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Bridging Language, Agent, and World Models for Reasoning and Planning.

diverse text corpora to interpret instructions, plan strategically, and anticipate long-term outcomes [21]. Their capacity for commonsense reasoning, abstract understanding, and symbolic inference supports robust and adaptive planning. Additionally, LLM-as-agents offer greater interpretability, as their text-based reasoning makes internal decision-making more transparent than the opaque policies of deep RL models—facilitating debugging, refinement, and trust in deployment.

Despite the promises, the true reasoning and planning capabilities of LLMs remain controversial. Many researchers contend that LLMs primarily excel at sophisticated pattern matching and statistical correlations, rather than genuine logical deduction or causal inference [18, 35, 37]. Studies indicate LLMs struggle with multi-step logic, exhibit inconsistent reasoning with minor prompt variations, and may fail to identify contradictions, highlighting the "fragility" of their reasoning [18, 23]. This limitation suggests LLMs may rely on memorization and interpolation rather than true open-ended computation or synthesizing new solutions. Motivated by ongoing debates regarding the reasoning capabilities of LLMs and their known limitations in genuine reasoning, we examine the planning abilities of LLMs in partially observable environments—settings where agents must operate under incomplete information and strategically structure their exploration to acquire missing but essential knowledge. By evaluating LLMs on tasks that demand decision-making under partial observability, we aim to assess their fundamental reasoning abilities when instantiated as agents.

To tackle the challenges posed by sequential decision-making under partial observability, we propose two complementary strategies to enhance LLM performance. First, we provide three distinct types of observation representations to support and evaluate the agent's decision-making capabilities. Many prior environments for LLM-based agents rely on literal, natural language observations [9, 42, 3], which LLMs are generally well-equipped to handle due to their pretraining on similar textual data. However, such representations often lack explicit spatial structure, which is essential for tasks involving navigation under partial observability. In contrast, some environments have adopted symbolically structured or formatted observation schemes [29], which better capture spatial relationships but introduce a higher level of abstraction by encoding highly compressed symbolic information, posing a greater challenge for LLMs. In this work, we compare these representation methods and further introduce a hybrid representation that combines the strengths of both natural language and symbolic formats.

Second, we prompt the LLM to generate action sequences that leverage its inherent planning capabilities. Prior work has typically evaluated LLMs in environments such as ALFWorld [32] or TextStarCraft II [21], where models are prompted to produce high-level plans that are subsequently decomposed into atomic actions (e.g., `Put the pan on the dining table`). However, executing these human-interpretable actions often requires a dedicated decoder to translate them into low-level executable commands, introducing significant engineering overhead and complexity. Moreover, relying solely on low-level action planning can underutilize the model's capacity for long-horizon reasoning. To mitigate this limitation, we enable LLM-as-agents to output multiple actions at each decision step—drawing inspiration from skill-based reinforcement learning—which facilitates more expressive and temporally extended plans.

Our results demonstrate that these approaches improve agent performance; however, we also observe specific cases where performance declines. We further analyze these failure modes to highlight the limitations of current LLMs. Specifically, our contributions are as follows:

- TextMiniGrid, a partially observable environment that provides three types of observation representations, built upon a modified version of BabyAI-Text [3].
- An analysis of LLM planning behaviors in partially observable RL settings using various prompting techniques.
- A detailed investigation of both success and failure cases to uncover the limitations and potential of LLM-based planning.

2 Related Works

LLMs for planning and decision making Recent research has explored the use of large language models (LLMs) for planning tasks that demand procedural reasoning, however, findings consistently reveal their limitations in structured, symbolic environments. Autoregressive LLMs, such as GPT-4 [1] and Claude, often fall short when asked to produce valid, multi-step plans, especially in domains

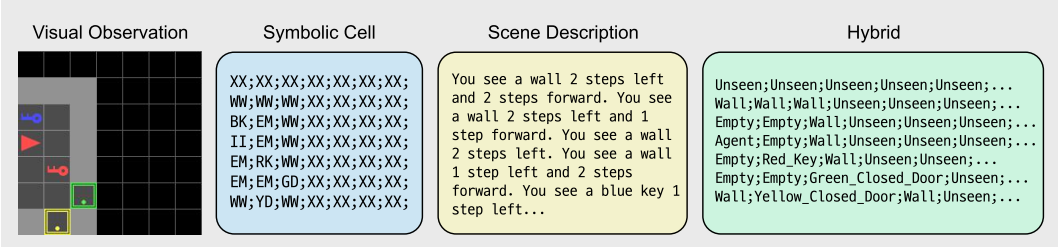


Figure 1: Examples of observation representation: Symbolic Cell, Scene Description, and Hybrid.

where surface cues are obfuscated or where precise action semantics are critical [33, 36]. In response, some approaches have proposed hybrid frameworks that embed LLMs into generate-verify loops, pairing them with external verifiers to enforce correctness [19]. More recently, large reasoning models (LRMs), including OpenAI's o1 [25] and DeepSeek R1 [13], have been proposed to improve inference-time reasoning through architectural and training changes. While these models represent progress, they still lack reliability guarantees and incur high inference costs—factors that limit their broader applicability [37]. Our work builds on this trajectory by empirically analyzing how LLMs perform across diverse sequential decision making scenarios and configurations, focusing on input abstractions, prompt structure, and environmental complexity that influence their planning effectiveness.

Hierarchical structure and representation in RL Reinforcement learning (RL) research has long emphasized the role of internal structure, such as temporal abstraction and representation, in improving agent performance, particularly in partially observable or procedurally generated environments. One line of work focuses on temporal abstraction, where hierarchical reinforcement learning methods employ macro-actions or options to simplify long-horizon decision making [34, 38, 24, 12]. More recent work on option discovery, such as Successor Options, leverages successor representations to identify well-connected subgoal states that facilitate exploration and transfer [30]. These abstractions help agents plan at higher levels and generalize across tasks. Complementarily, memory- and history-based architectures have shown promise in dealing with partial observability by modeling temporal context via recurrence [14], attention [27], or sequence modeling [4]. While these methods rely on learned structure, they suggest that external guidance, such as demonstrations or natural language descriptions, may similarly support planning in LLM-as-agents.

Prompting and input-level guidance for LLMs Beyond architectural innovations and internal structure, recent work has explored how structured inputs, such as few-shot demonstrations, task descriptions, and curated exemplars, can shape the behavior of language models and reinforcement learning agents. Few-shot prompting, originally developed in NLP, has been adapted to decision-making settings to provide implicit behavior templates or task decompositions [11, 2]. These prompts often act as demonstrations that guide LLMs toward desirable behaviors, especially under limited supervision. In language-conditioned agents, natural language goals, object references, and scene descriptions have also been used to scaffold decision-making and planning [20, 17, 10]. Despite promising results, it remains unclear which forms of input guidance are most effective in environments with different levels of complexity and abstraction. Our work extends this line of investigation by systematically analyzing how input-side factors, including demonstration format, temporal structure, and environment encoding, influence planning performance in a LLM-driven sequential decision making environment.

3 Domain and Problem Statement

We use BabyAI [6], a simulated, partially observable 2D gridworld environment. Built upon the MiniGrid platform, BabyAI supports efficient simulation and a suite of instruction-following tasks using a subset of a synthetic language called Baby Language. Each BabyAI environment features a randomly generated room layout, object configuration, and a natural language mission, all sampled from a distribution over n rooms. Objects are described by color and type, with doors requiring keys of the corresponding color to open. At each time step, the agent receives a partial observation representing its field of view, along with a textual instruction in Baby Language.

Among other alternatives, such as Blocks World [22] or Sokoban, we chose BabyAI as it presents unique challenges, involving object manipulation (e.g. moving the object to unblock the door), pathfinding and exploration-exploitation due to its partially observable nature. Another key advantage of BabyAI is the availability of an Oracle Solver (referred to as BOT in [6]), which generates step-by-step solutions for any given environment. This is accomplished using hand-coded rules and an internal stack machine that produces plans for solving tasks. The Oracle Solver enables us to systematically analyze the difficulty of different layout configurations, independent of the performance of individual agents. Moreover, classical domains such as Blocks World and Mystery Blocks World [22] are known to be insufficient for evaluating the planning capabilities of the advanced LLM-as-agents [37].

3.1 TextMiniGrid

We implement the TextMiniGrid environment based on the work of Carta et al. [3], which adapts the BabyAI environment into a text-based interface. The environment provides three types of text-based partial observations, as illustrated in Figure 1: Symbolic Cell, introduced in this work, Scene Description, a modified version of the representation proposed by Carta et al. [3]; and Hybrid, which combines elements from both previous representations. Each episode terminates either when the mission is successfully completed or when the predefined step limit is reached (200 steps in our experiments). The action space comprises six distinct actions, encompassing both movement and object interactions. Further details on the action and observation formats are provided in Appendix A.

Across all cases, the agent's perception is limited by occlusions such as walls and closed doors. The Symbolic Cell encodes each grid cell using predefined two-character symbols: the first character indicates the color of the object, and the second character denotes its type. The partially observed grid is rotated according to the agent's facing direction. We modify the Scene Description from [3] by adding the locations of all walls within the agent's field of view. In the single-room setting, only the distance to the wall is relevant. However, in the multi-room setting, since the agent must traverse doors located between walls, it is necessary to include descriptions for all observed wall grids. The Hybrid representation combines the two approaches by translating symbolic elements into general terms expressed in natural language.

Each mission is randomly selected from the four default single-action instruction missions, with the focus restricted to the `PickUp` task. In this task, the agent is required to retrieve a target object that is uniquely specified by its color and type across all rooms. The `GoTo` task is considered a simpler variant of `PickUp`, while the `OpenDoor` task is inherently required for completing missions that involve exploration across multiple rooms. The `PutNextTo` task introduces additional complexity, as it requires the agent to maintain memory of the target object from the moment it is first observed. Due to these considerations, we focus our study on the `PickUp` action to isolate core challenges related to planning and navigation. Given the partially observable setting, the agent is strongly encouraged to explore unseen areas in order to successfully complete the mission. Searching for the target object under limited perceptual range constitutes the primary source of overall difficulty.

4 Analyzing Behavioral Difficulty in Partially Observable Environments

In this section, we discuss the challenges posed by our TextMiniGrid domain for LLM-based agents. In a partially observable setting, several factors contribute to the overall difficulty of finding effective solutions. These include the density of objects and the distance to the target.

We procedurally generate spatial layouts by varying two primary factors: object density and room count. Object density is stratified into three levels—low, medium, and high—based on the proportion of occupied cells within a 4×4 room (excluding doors), corresponding to configurations with 1 (6.25%), 3 (18.75%), and 5 (32.25%) objects, respectively. Each room is enclosed by walls and connected to adjacent rooms via doors, which may be either locked or unlocked. The rooms are arranged in a square grid layout, allowing for various spatial connectivity patterns. Doors serve a dual role in the environment, functioning either as static barriers, similar to walls, or as dynamic passageways that facilitate navigation. Additional implementation details are provided in Appendix A.1.

Behavioral Difficulty Due to the random assignment of the agent's initial location, orientation, and target object, even highly complex spatial configurations can result in trivially solvable tasks. For

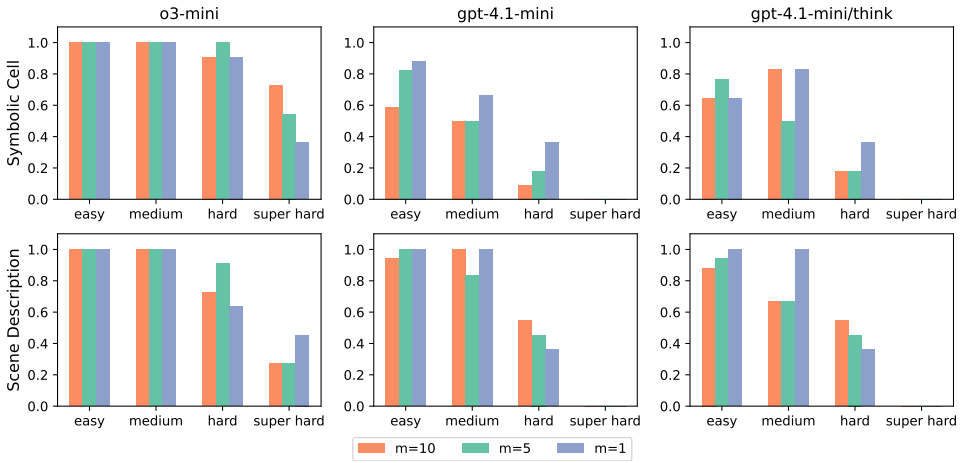


Figure 2: Success rates of actions taken at each decision step using the Symbolic Cell representation and Scene Description across different LLMs. Success rates are averaged across difficulty levels.

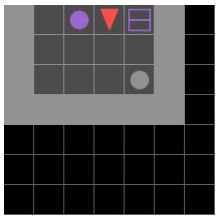
example, in a nine-room layout with high object density, the agent may begin the episode directly facing the target object, significantly reducing the task's difficulty. For our experiments, we randomly sample a total of 45 tasks across 9 layout configurations, defined by the Cartesian product of 3 object density levels and 3 room-number levels, using 5 different random seeds. Tasks are categorized into four difficulty levels based on the number of steps required for completion by the Oracle Solver. This classification follows the notion of Behavioral Difficulty, which is determined by the length of the action sequence needed to complete a task [16]. The distribution of tasks across these difficulty levels is 37.8% easy, 13.3% medium, 24.4% hard, and 24.4% super-hard. Additional representative examples for each difficulty level are provided in Appendix F.

5 Experiments

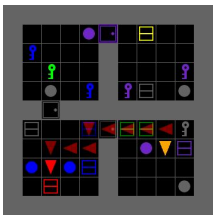
For evaluation, we use two OpenAI LLMs: o3-mini-2025-01-31 (o3-mini) and gpt-4.1-mini-2025-04-14 (gpt-4.1-mini), chosen to compare models with and without advanced reasoning capabilities—an increasingly prominent focus in contemporary research. o3-mini has demonstrated relatively strong performance compared to non-reasoning models like gpt-4.1-mini on Chollet [7], a benchmark designed to assess an agent's ability to infer abstract rules from visual patterns, making it suitable for our spatial reasoning tasks. Additionally, we include gpt-4.1-mini with a zero-shot reasoning prompt (referred to as gpt-4.1-mini/think) to assess the impact of prompting strategies on the performance of non-reasoning agents, following the approach of [43]. We evaluate these LLM-as-agents on 45 tasks in total, spanning 9 layout configurations and using 5 random seeds per configuration.

At each decision step, t', the agent is provided with a user prompt containing the mission description, the current observation, the agent's facing direction, and the inventory status. Additionally, we supply the agent with a stack of the previous $q = 5$ observation-action pairs with intentions, serving as short-term memory necessary for completing the missions. The agents are instructed to output actions along with their underlying intentions, based on the provided information, to allow for interpretation of their planning. The input prompt is provided in Appendix E.

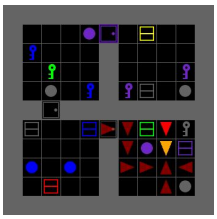
System Prompt The system prompt, which takes precedence over the input prompt, helps LLMs understand essential background knowledge or establish the tone of the conversation. Therefore, providing an appropriate system prompt is crucial for effectively guiding the model's behavior. [21] demonstrated that the performance of LLM-as-agent can vary significantly depending on the type of system prompt used in TextStarCraft II. To enable the LLM to effectively interpret the BabyAI environment, we augment the system prompt with three components: the Action Description, which provides detailed explanations of each available action; the Mission Description, which outlines how to successfully complete various missions; and the Strategic Guide, which informs the LLM that it is operating in a partially observable environment and offers strategies for effective exploration. We provide detailed descriptions of these components and the ablation study in Appendix D.



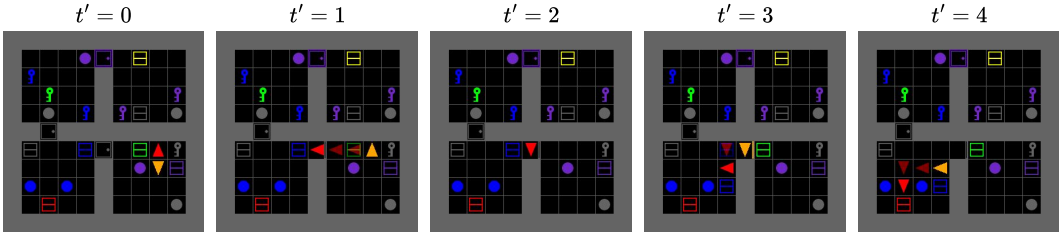
(a) The initial observation



(b) Successful trajectory ($m = 10$)



(c) Failure trajectory ($m = 1$)



(d) Skill transitions of the successful trajectory ($m = 10$)

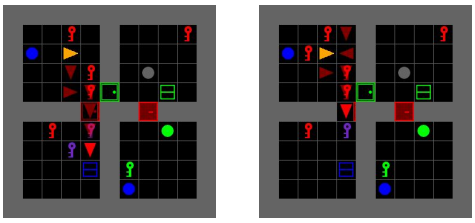
Figure 3: Visualization of agent trajectories under the Symbolic Cell representation with action sequence lengths $m = 10$ and $m = 1$, from the start position (▸) to the final position (▸). Transparent agents indicate the trajectory of movement, and transparent objects represent items that have been moved by the agent. The mission is to pick up the red box.

5.1 Planning Abilities with Varying Number of Taken Actions

In prior works on solving sequential decision-making environments with LLM-as-agents, such as [31] and [21], agents typically generate high-level actions, demonstrating the planning capabilities of LLMs. For example, in AlfWorld [32], the agent receives observations such as `You put the pan 1 in countertop 1` and subsequently selects an action like `take pan 1 from stoveburner 2`. However, these environments require an action decoder that outputs low-level actions to interact with the environment, such as the dynamics of robot manipulation to pick up a pan. In contrast, our TextMiniGrid requires agents to generate atomic, low-level actions, such as `Go Forward`, and `Turn Right`. To better leverage the planning capabilities of LLMs as agents and to achieve temporal abstraction, we introduce a maximum number of actions, denoted by m, that an agent can take in a single decision step. Similar to human decision-making, which often involves planning and selecting among temporally extended options across various time scales [34], our approach encourages agents to reason over sequences of actions (i.e., skills) rather than individual steps.

We allow agents to take $m = 1, 5, 10$ actions in a single decision step to investigate how their flexible planning abilities manifest across different levels of temporal abstraction and task difficulty. Since agents are capable of executing multiple actions within a single decision step, providing only the final observation after the sequence may result in the loss of critical information. For instance, an agent may momentarily observe a target object while turning, but this information would be lost if only the initial and final observations are retained. To mitigate this issue, we provide the full sequence of transitions, each consisting of an observation and its corresponding action, for each individual step within the action sequence.

In Figure 2, the reasoning model (o3-mini) and the non-reasoning model (gpt-4.1-mini) exhibit different performance patterns. While o3-mini achieves perfect success rates on easy and medium-level tasks, both gpt-4.1-mini and its prompted variant, gpt-4.1-mini/think, consistently underperform across all difficulty levels. Varying the value of m yields different benefits depending on the observation type. While o3-mini's overall performance declines on harder tasks, longer action sequences en-



(a) Success ($m = 10$) (b) Fail ($m = 1$)

Figure 4: Visualization of trajectories under the Scene Description. The mission is to pick up the blue box.

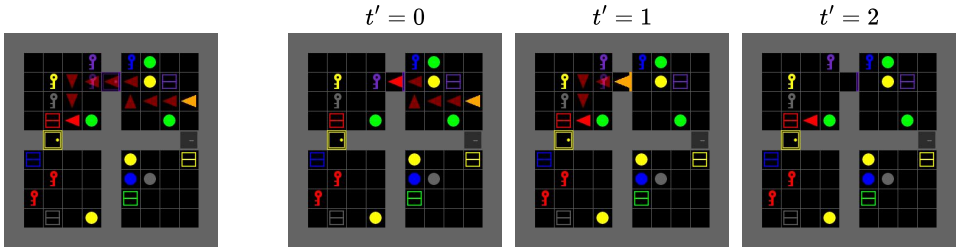


Figure 5: Illustration of a trajectory generated by o3-mini under the Symbolic Cell representation for each decision step $t' = \{0, 1, 2\}$. This figure depicts the trajectory beginning at the initial agent position (▸) and ending at the final position (▸). The mission is to pick up the red box.

Table 1: Number of tasks solved under Symbolic Cell (SC) but failed under Scene Description (SD), and vice versa, reported across different action sequence lengths per decision step (m).

LLM-as-agent	m	SC(✓) & SD(✗)	SC(✗) & SD(✓)
o3-mini	10	**14**	0
	5	**8**	0
	1	**8**	4
gpt-4.1-mini	10	0	**28**
	5	2	**18**
	1	2	**10**
gpt-4.1-mini/think	10	6	**20**
	5	4	**18**
	1	2	**16**

hance success under Symbolic Cell observations, whereas shorter sequences are generally more effective with Scene Description. In super-hard tasks, under Symbolic Cell, 4 out of 11 tasks are solved with $m = 10$ but not with $m = 1$, and none are solved with $m = 1$ but not $m = 10$. Conversely, under Scene Description, only 1 out of 11 tasks is solved with $m = 10$ but not $m = 1$, whereas 3 are solved with $m = 1$ but not $m = 10$. These super-hard tasks often involve many rooms and high object density, requiring exploration and complex planning.

Figure 3 illustrates a representative super-hard task where the o3-mini agent succeeds with $m = 10$ but fails with $m = 1$. In this task, the agent must open the grey door and move the blue box to access and pick up the red box. With a higher action budget per decision step, the agent exhibits coherent and strategic behavior, as illustrated in Figure 3b. The agent completed the task in 27 steps, outputting an average of 5.4 actions per decision step. Specifically, as shown in Figure 3d, the agent initially turns to explore beyond the current room, removes the blocking green box, opens the grey door, drops the carried object to clear the blue box, and finally approaches the target. In contrast, with $m = 1$, the agent struggles to remove the blue box and access the adjacent room. Similarly, under the Scene Description representation, we observe that the agent demonstrates coherent behavior to accomplish the mission when allowed longer action sequences. However, with $m = 1$, the agent becomes stuck between two objects and fails to reach the target, as illustrated in Figure 4. We hypothesize that this limitation stems from the agent's tendency to forget previous intentions when constrained by a limited stack length q, as evidenced by its repeated behavior of dropping and picking up the blue box.

When comparing gpt-4.1-mini and gpt-4.1-mini/think under the Scene Description representation, gpt-4.1-mini/think shows lower success rates than gpt-4.1-mini. In contrast, under the Symbolic Cell representation, their performances are similar or slightly improved due to the benefit of zero-shot reasoning. As discussed in Section 3.1, agents can often complete the mission by directly following the target object description. In these scenarios, the additional zero-shot reasoning prompt may introduce unnecessary inference steps, leading to suboptimal decision, which is particularly critical when the agent is allowed to take longer action sequences. For example, under the medium difficulty setting, the 100% success rate of gpt-4.1-mini with Scene Description drops to below 70% for gpt-4.1-mini/think.

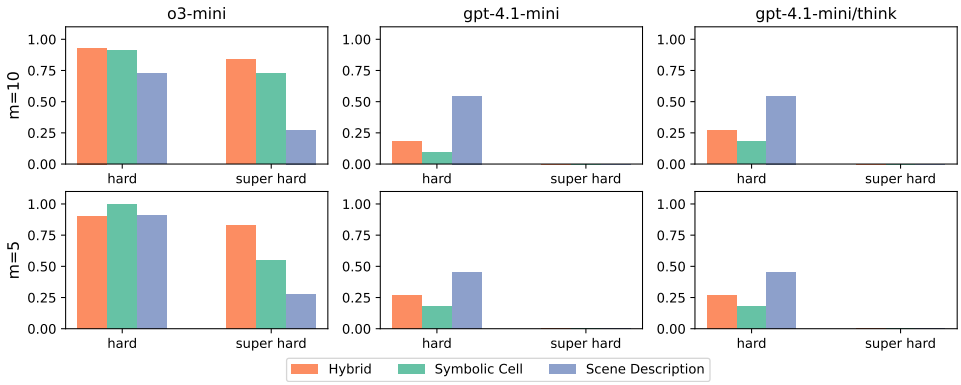


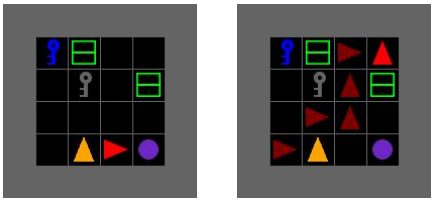
Figure 7: Success rates of three observation representations with $m = 10$ and $m = 5$ across various LLMs. Results are averaged over all difficulty levels.

5.2 Comparison of Observation Representations

The Symbolic Cell encodes not only objects and walls but also empty cells, whereas the Scene Description does not indicate distance to empty cells. Furthermore, the Symbolic Cell allows the agent to easily recognize spatial relationships among adjacent objects, while in the Scene Description, such relationships should be inferred through reasoning. On the other hand, the Scene Description provides natural language observations, which may be easier for LLM-as-agents to interpret than Symbolic Cell representations. Additionally, it implicitly guides how to reach an object by providing relative positions. For instance, in Figure 1, the agent can follow the instruction "2 steps right and 2 steps forward" to reach the closed green door. However, note that such descriptions do not account for obstacles, so if the agent blindly follows them without considering potential blockers, it may fail to reach the goal.

Table 1 reports the number of tasks (out of 45) that were successfully completed using the Symbolic Cell representation but failed under the Scene Description, and vice versa. The results show that the Symbolic Cell representation is more effective than the Scene Description when paired with a reasoning-capable model (o3-mini). Conversely, the Scene Description yields better performance with a non-reasoning model (gpt-4.1-mini), likely due to its richer natural language context. Interestingly, when comparing the non-reasoning model to its counterpart enhanced with zero-shot reasoning capabilities (gpt-4.1-mini/think), the latter demonstrates a better understanding of Symbolic Cell observations. This suggests that reasoning ability plays a critical role in leveraging structured representations for improved task performance.

The trajectory illustrated in Figure 5 depicts a scenario ($m = 10$) where the LLM-as-agent (o3-mini) successfully completes the task under the Symbolic Cell representation (referred to as the SC agent), but fails under the Scene Description (SD agent). The SC agent effectively explores the environment, sequentially reaching intermediate sub-targets while accurately reasoning about object locations.This highlights the effectiveness of the Symbolic Cell representation in enabling structured reasoning and robust long-horizon planning. By contrast, Figure 6 illustrates a case ($m = 10$) where a non-reasoning LLM-as-agent (gpt-4.1-mini) succeeds with the Scene Description but fails with Symbolic Cell. In this scenario, the agent completes the task in the first decision step by directly interpreting the cue: `"You see a purple ball 2 steps right"`. However, under the Symbolic Cell representation, the agent struggles to infer spatial relations, leading to a mismatch between its plan and the resulting actions.



(a) Scene Description (b) Symbolic Cell

Figure 6: Visualization of trajectories under two representations. The mission is to pick up the purple ball.

To leverage the strengths of both observation representation methods, we evaluate LLMs using a Hybrid representation, which augments the Symbolic Cell format by replacing abstract symbols (e.g., `RB`) with their natural language equivalents (e.g., `Red_Box`). This approach combines the spatial

structure of symbolic observations with the linguistic familiarity of natural language, enabling LLMs to reason more effectively in structured environments. Figure 7 shows that the Hybrid observation method improves reasoning performance compared to the Symbolic Cell representation. Notably, o3-mini exhibits substantial gains, especially at the very hard difficulty level. While Scene Description remains more effective for gpt-4.1-mini and gpt-4.1-mini/think, their performance also improves with the Hybrid method than the Symbolic Cell. These results support our hypothesis that the Hybrid representation enhances LLMs' spatial understanding by combining structural clarity with linguistic familiarity. In Appendix B, we also evaluate LLM-as-agents by presenting common failure cases as few-shot demonstrations, thereby leveraging their in-context learning abilities [2, 41].

5.3 Adaptation to Dynamic Reasoning Model

We investigate the influence of action sequence length and observation representation on the performance of LLM-based agents, both with and without advanced reasoning capabilities. Our results show that as reasoning capacity increases, agents benefit from longer action sequences and more expressive observation encodings that integrate compact, spatially structured information. These enhancements enable the LLM-as-agent to tackle increasingly complex tasks, as illustrated in Figure 7. To further examine these effects, we evaluate the dynamic reasoning model Gemini 2.5 Flash [8], which adaptively invokes advanced reasoning processes when it determines that a problem requires deeper deliberation. We hypothesize that the model exhibits performance patterns similar to o3-mini when addressing hard and super-hard tasks.

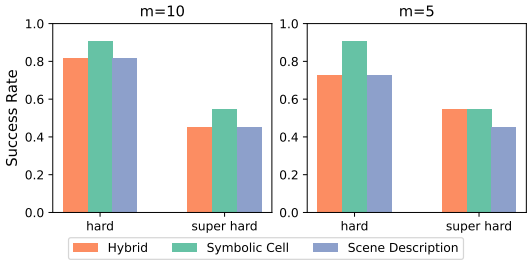


Figure 8: Success rates of three observation representation with $m = 10$ (left) and $m = 5$ (right) using Gemini 2.5 Flash model. Results are averaged over all difficulty levels.

Figure 8 reports success rates for three observation representations using extended action sequence budgets ($m = 5, 10$) with the Gemini 2.5 Flash model. When allocated a longer sequence budget, the LLM-as-agent maintains comparable performance under the Symbolic Cell representation. In super-hard tasks, the agent successfully removes a blocker and reaches the target in an average of 5.83 steps under Symbolic Cell. In contrast, the Scene Description representation disrupts the agent's planning, as discussed in Section 5.2. These results suggest that dynamic reasoning enables LLM-as-agents to interpret and leverage complex information more effectively, thereby improving their ability to solve difficult tasks. Moreover, increasing m improves performance on hard-level tasks across observation representations. Under Scene Description, for instance, the agent requires an average of 3.85 steps ($m = 5$) and 5.10 steps ($m = 10$), indicating that a longer action budget supports more flexible planning strategies.

6 Conclusion and Limitation

In this paper, we investigate the use of large language models (LLMs) as agents, emphasizing their strengths in generalization and language-based interpretability. We evaluate their planning capabilities in partially observable environments with varying behavioral challenges using two key strategies: (i) introducing diverse observation representations tailored to LLMs, and (ii) prompting LLM-as-agents to generate action sequences that leverage their inherent planning abilities. In addition, we assess the adaptability of our approach by evaluating a dynamic reasoning model. Our analysis demonstrates that LLM-as-agents, regardless of explicit reasoning capacity, exhibit enhanced problem-solving performance.

However, LLM performance varies with inherent properties such as model capacity and reasoning ability, which affect their responsiveness to observation types, action budgets, and example formats. Reasoning-capable models benefit from structured observations, longer action horizons, and examples that aid planning and generalization. In contrast, non-reasoning models are more sensitive to observation formats and show limited improvements from extended planning or few-shot prompts. Additionally, in partially observable environments, LLMs often lack mechanisms to retain and use

past information, leading to redundant behaviors like revisiting areas. Future work should explore memory integration or external tools to improve information retention and task efficiency.

7 Acknowledgement

This work was partly supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2022-II220311, Development of Goal-Oriented Reinforcement Learning Techniques for Contact-Rich Robotic Manipulation of Everyday Objects, No. RS-2024-00457882, AI Research Hub Project, No. RS-2019-II190079, Artificial Intelligence Graduate School Program (Korea University), and No. RS-2025-25410841, Beyond the Turing Test: Human-Level Game-Playing Agents with Generalization and Adaptation), the IITP (Institute of Information & Communications Technology Planning & Evaluation)-ITRC (Information Technology Research Center) grant funded by the Korea government (Ministry of Science and ICT) (IITP-2025-RS-2024-00436857), the NRF (RS-2024-00451162) funded by the Ministry of Science and ICT, Korea, BK21 Four project of the National Research Foundation of Korea, and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-00560367), and the IITP under the Artificial Intelligence Star Fellowship support program to nurture the best talents (IITP-2025-RS-2025-02304828) grant funded by the Korea government (MSIT).

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

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

[3] Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, pages 3676–3713. PMLR, 2023.

[4] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[5] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[6] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: First steps towards grounded language learning with a human in the loop. In *International Conference on Learning Representations*, volume 105. New Orleans, LA, 2019.

[7] François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

[8] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[9] Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*, pages 41–75. Springer, 2018.

[10] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. In *International Conference on Machine Learning*, pages 8469–8488. PMLR, 2023.

[11] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *Advances in neural information processing systems*, 30, 2017.

[12] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

[13] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[14] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*, 2015.

[15] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[16] Shengran Hu and Jeff Clune. Thought cloning: Learning to think while acting by imitating human thinking. *Advances in Neural Information Processing Systems*, 36:44451–44469, 2023.

[17] 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.

[18] Subbarao Kambhampati. Can large language models reason and plan? 1534(1):15–18. ISSN 1749-6632. doi: 10.1111/nyas.15125. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/nyas.15125`.

[19] Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. Llms can't plan, but can help planning in llm-modulo frameworks, 2024. *URL https://arxiv. org/abs/2402.01817*, 2024.

[20] Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. *arXiv preprint arXiv:2005.07648*, 2020.

[21] Weiyu Ma, Qirui Mi, Yongcheng Zeng, Xue Yan, Runji Lin, Yuqiao Wu, Jun Wang, and Haifeng Zhang. Large language models play starcraft ii: Benchmarks and a chain of summarization approach. *Advances in Neural Information Processing Systems*, 37:133386–133442, 2024.

[22] Drew M. McDermott. The 1998 ai planning systems competition. *AI Magazine*, 21(2):35, Jun. 2000. doi: 10.1609/aimag.v21i2.1506. URL `https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1506`.

[23] Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models. URL `http://arxiv.org/abs/2410.05229`.

[24] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

[25] OpenAI. Learning to reason with llms. *OpenAI*, September 2024. URL `https://openai.com/ko-KR/index/learning-to-reason-with-llms/`.

[26] Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wolczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, et al. Balrog: Benchmarking agentic llm and vlm reasoning on games. *arXiv preprint arXiv:2411.13543*, 2024.

[27] Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pages 7487–7498. PMLR, 2020.

[28] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[29] Santhosh Kumar Ramakrishnan, Erik Wijmans, Philipp Kraehenbuehl, and Vladlen Koltun. Does spatial cognition emerge in frontier models? *arXiv preprint arXiv:2410.06468*, 2024.

[30] Rahul Ramesh, Manan Tomar, and Balaraman Ravindran. Successor options: An option discovery framework for reinforcement learning. *arXiv preprint arXiv:1905.05731*, 2019.

[31] 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.

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

[33] Tom Silver, Varun Hariprasad, Reece S Shuttleworth, Nishanth Kumar, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Pddl planning with pretrained large language models. In *NeurIPS 2022 foundation models for decision making workshop*, 2022.

[34] R. S. Sutton, D. Precup, and S. Singh. Between mops and semi-mop: Learning, planning & representing knowledge at multiple temporal scales. Technical report, University of Massachusetts, USA, 1998.

[35] Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. LLMs Still Can't Plan; Can LRMs? A Preliminary Evaluation of OpenAI's o1 on PlanBench. URL `http://arxiv.org/abs/2409.13373`.

[36] Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36: 38975–38987, 2023.

[37] Karthik Valmeekam, Kaya Stechly, Atharva Gundawar, and Subbarao Kambhampati. A systematic evaluation of the planning and scheduling abilities of the reasoning model o1. *Transactions on Machine Learning Research*, 2025.

[38] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International conference on machine learning*, pages 3540–3549. PMLR, 2017.

[39] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. 575(7782):350–354. ISSN 1476-4687. doi: 10.1038/s41586-019-1724-z. URL `https://www.nature.com/articles/s41586-019-1724-z`.

[40] 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. URL `http://arxiv.org/abs/2305.16291`.

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

[42] Yue Wu, Xuan Tang, Tom M Mitchell, and Yuanzhi Li. Smartplay: A benchmark for llms as intelligent agents. *arXiv preprint arXiv:2310.01557*, 2023.

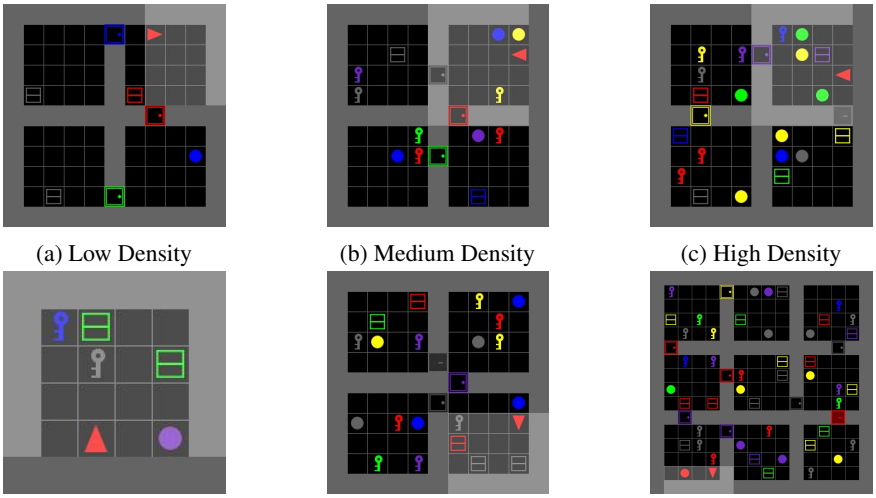
[43] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

A TextMiniGrid

Given a language vocabulary $\mathcal{V}$, TextMiniGrid generates a text-based observation $o \in \mathcal{O} \subset \mathcal{V}^N$, a mission description $g \in \mathcal{G} \subset \mathcal{V}^N$, and accepts a text-based action $a \in \mathcal{A} \subset \mathcal{V}^N$. Such an environment can be framed as a goal-conditioned Partially Observable Markov Decision Process (POMDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{V}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{G}, \mathcal{O} \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A} \subset \mathcal{V}^N$ is the action space, $\mathcal{G} \subset \mathcal{V}^N$ is the mission space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ is the transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \mapsto \mathbb{R}$ is the goal-conditioned reward function, and $\mathcal{O} : \mathcal{S} \mapsto \mathcal{V}^N$ is the observation function mapping a state to its textual description.

We employ BabyAI [6], a simulated, partially observable 2D gridworld environment. The BabyAI platform built upon the MiniGrid environment, which supports efficient simulation and a suite of instruction-following tasks formulated using a subset of a synthetic language (Baby Language). Each BabyAI environment consists of a randomly generated room layout, object configuration, and a natural language mission, all sampled from a distribution over n rooms. Objects are described by their color (red, green, blue, yellow, purple, grey) and type (balls, keys, boxes, doors). Doors can be opened using keys that match their corresponding color. At every time step, the agent receives a partial observation representing its field of view, the grid cells directly in front of it, along with a textual instruction expressed in Baby Language.

The action space supports six actions: `Go Forward`, `Turn Left`, `Turn Right`, `Pickup`, `Drop`, and `Toggle`. The `Toggle` action allows the agent to interact with doors, such as opening, closing, or unlocking them. The environment offers two types of text-based partial observations: Symbolic Cell and Scene Description. The agent's field of view is limited to a 7×7 and is obstructed by walls and closed doors. The Symbolic Cell encodes each grid cell using predefined two-character symbols, where the first character indicates the color of the object and the second character denotes its type. A cell can either be empty (EM), unseen (XX), contain a wall (WW), or hold an object. Doors are categorized into three types: a closed door is represented by D with its corresponding color (e.g., BD for a blue door), a locked door is denoted by L with its color prefix (e.g., PL for a purple locked door), and an open door is represented by __. When the agent toggles an open door, it becomes closed and is represented by D with its original color. The agent itself is represented by II, and its facing direction is provided separately as textual information. The partially observed grid is rotated according to the agent's facing direction. For example, in Figure 1, the agent (II) appears on the left side because it is facing East. We modify the Scene Description from [3] by adding the locations of all walls within the agent's field of view, as illustrated in Figure 1. In the single room setting, only the distance to the wall is relevant. However, in the multi room setting, since the agent must traverse doors located between walls, it is necessary to include descriptions for all observed wall grids. An episode terminates either when the mission is successfully completed or when the pre-set maximum number of steps (200 in our experiments) is reached.



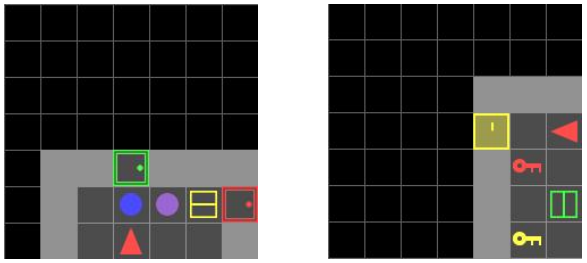
(a) Low Density (b) Medium Density (c) High Density

(d) Single Room (e) Four Rooms (f) Nine Rooms

Figure 9: Visualization of various spatial layout configurations. The first row illustrates object distributions across varying density levels. The second row presents layouts with different numbers of rooms enclosed by walls and connected by doors. The highlighted region represent the agent's field of view.

A.1 Object Density

Figure 9 visualizes examples of randomly generated spatial layouts, categorized by object density levels and the number of rooms. The first row displays object distributions corresponding to varying density levels, which we classify into three categories (low, medium, and high). These levels are determined based on the proportion of occupied cells within a single room (excluding doors), corresponding 1 object (6.25%), 3 objects (18.75%), and 5 objects (32.25%), respectively. The second row presents layouts with varying numbers of rooms, each of size 4×4, enclosed by walls and connected by doors. The rooms are arranged in a square grid formation, with each room potentially connected to adjacent rooms via either locked or unlocked doors. Doors serve a dual role in the environment as they can function as blockers similar to walls or as passageways like corridors.



(a) Blockers (b) A Locked Door

Figure 10: Examples of agent observations in the presence of blockers and a locked door, illustrating the challenges of partial observability and environmental clutter. The shaded door indicates a locked state.

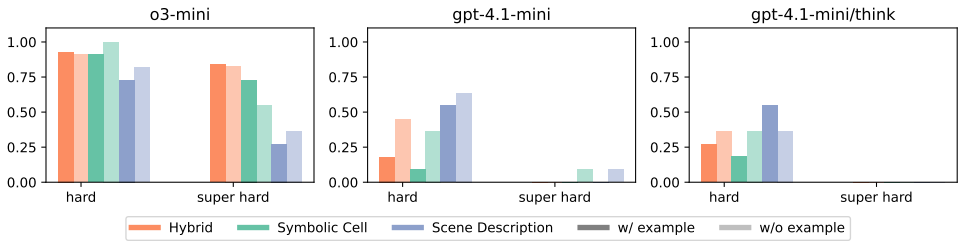


Figure 11: Success rates for three observation representations, with and without few-shot examples ($m = 10$), evaluated across various LLMs. Results are averaged over all difficulty levels.

B Analyzing Absorptive Capacity with Few-shot Examples

It is well established that providing examples can improve the performance of LLMs across a variety of tasks by leveraging their in-context learning capabilities [2, 41]. While there is no standardized approach for incorporating examples into sequential decision-making tasks, we observed recurring failure modes across different models and difficulty levels, and designed targeted examples to address these cases. These failures often arise from complex spatial structures in the environment, such as blockers and locked doors, which substantially increase task difficulty. Distractors are objects that are irrelevant to the optimal solution path and do not impede the agent's movement. Blockers are obstacles that obstruct the agent's direct access to the goal; for instance, the blue ball in Figure 10a prevents the agent from reaching and opening the green door to explore the area behind it. Locked doors create scenarios in which the agent must first retrieve a key of the corresponding color to proceed. As illustrated in Figure 10b, if the agent initially plans to pass through a yellow door, it must first locate and pick up the yellow key even in the absence of distracting clutter, resulting in a necessary deviation from its planned path.

The examples we provide are designed to address three key scenarios: (i) unblocking an object, where the agent must move a blocker before proceeding; (ii) unblocking a door, where the agent needs to open a door to continue exploration; and (iii) opening a locked door, where the agent must pick up the correct key to unlock it. Additionally, we include an example focused on exploration, in which the agent begins in a room that should be revealed. While exploration is not explicitly part of Behavioral Difficulty, it often plays a critical role in mitigating it. To avoid overlap with test layouts, we generated a random layout featuring four rooms and medium object density for these examples.

We evaluate our approach with $m = 10$ using few-shot examples. As shown in Figure 11, few-shot prompting generally improves performance across configurations and, notably, in some cases enables agents to solve previously unsolvable very hard tasks. However, these examples often require models to generalize to situations that are similar but not identical, which remains challenging for LLMs. In hard-level tasks, o3-mini shows modest gains with the Hybrid observation type, while other models continue to struggle. These results suggest that few-shot examples do not always help LLM-as-agents improve performance in sequential decision-making environments under partial observability. While they can provide meaningful gains in some settings, especially when models can effectively generalize, they may offer limited benefit when the examples do not sufficiently align with the target scenario or when the model lacks the capacity for robust abstraction.

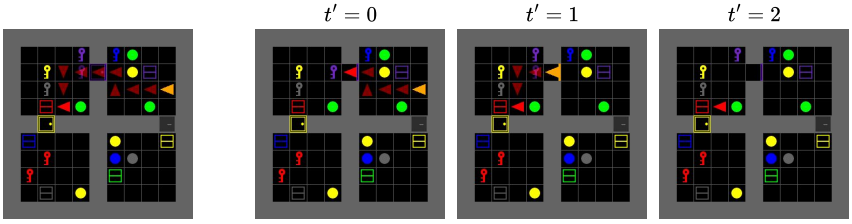
C Analysis of Agent Behavior under Two Observation Representations

Table 2: Number of tasks solved under Symbolic Cell (SC) but failed under Scene Description (SD), and vice versa, reported across different action sequence lengths per decision step (m) and the presence or absence of few-shot examples.

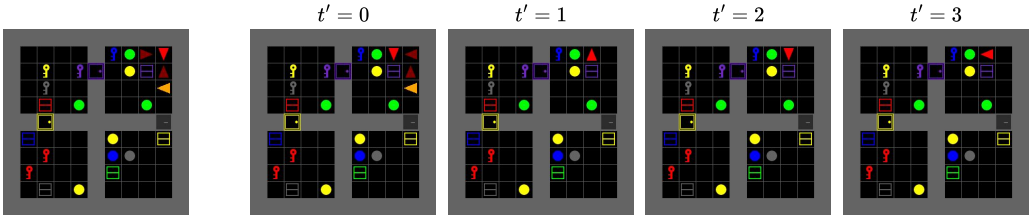
	LLM-as-agent	m	SC(✓) & SD(✗)	SC(✗) & SD(✓)
no example	o3-mini	10	**14**	0
		5	**8**	0
		1	**8**	4
	gpt-4.1-mini	10	0	**28**
		5	2	**18**
		1	2	**10**
	gpt-4.1-mini/think	10	6	**20**
		5	4	**18**
		1	2	**16**
show example	o3-mini	10	**12**	4
		5	**8**	0
		1	**12**	0
	gpt-4.1-mini	10	2	**14**
		5	0	**8**
		1	0	**18**
	gpt-4.1-mini/think	10	2	**16**
		5	4	**12**
		1	**10**	4

Table 2 reports the number of tasks (out of 45) that were successfully completed under the Symbolic Cell representation but failed under the Scene Description, and vice versa—tasks that failed under Symbolic Cell but succeeded under Scene Description. The Symbolic Cell representation proves more effective than the Scene Description when used with a reasoning model (o3-mini). In contrast, the Scene Description benefits a non-reasoning model (gpt-4.1-mini), leading to better performance. Interestingly, when comparing a non-reasoning model to one equipped with zero-shot reasoning capabilities (gpt-4.1-mini/think), the latter demonstrates a better understanding of Symbolic Cell observations, indicating the importance of reasoning ability in leveraging structured representations.

Case 1 This case illustrates a scenario ($m = 10$) where the LLM-as-agent (o3-mini) successfully completes the task under the Symbolic Cell representation (referred to as the SC agent) but fails under the Scene Description (referred to as the SD agent), as shown in Figure 12. The SC agent successfully plans to explore the environment and reach intermediate sub-targets while accurately reasoning about object positions (Figure 13a), demonstrating the effectiveness of the Symbolic Cell representation, which provides structured spatial information, as discussed in Section 3.1. In contrast, the SD agent struggled to extract positional information from the linearly represented observation and consequently miscomputed object locations, leading to invalid actions, as shown in Figure 13b.



(a) Successful trajectory (Symbolic Cell).



(b) Failure trajectory (Scene Description). Only the first four steps of the trajectory are shown.

Figure 12: Visualization of agent trajectories under the Symbolic Cell representation and Scene Description, from the start position (▸) to the final position (▸). Transparent agents indicate the trajectory of movement, and transparent objects represent items that have been moved by the agent. The mission is to pick up the red box.

	Intention
$t' = 0$	Advance westward from our starting position to explore new territory. After moving from (3,6) to (3,3), turn north to approach the purple door ahead; then face west to toggle (i.e. open) the door, and finally step through the opened door to reveal additional areas where the red box might be located.
$t' = 1$	Pick up the purple key blocking our only exit so that its cell becomes free, then move into that cell and drop the key off the main path. Next, reorient and proceed west then south to reach (5,4) adjacent to the red box (RB), facing West so that we can pick it up in a subsequent turn.
$t' = 2$	Since the red box (RB) is immediately in front of us (to our west) and adjacent, we simply pick it up to complete the final mission.

(a) Symbolic Cell representation ($t' = \{0, 1, 2\}$).

	Intention
$t' = 0$	Navigate to the area adjacent to the closed purple door from the north so that I can open it. I suspect the red box might be behind this door, so I'm preparing to explore that area.
$t' = 1$	I want to change direction from facing South to exploring the corridor toward the west, where objects like the blue key and closed purple door are visible. My aim is to search further for clues or the red box, since it is not currently in view.
$t' = 2$	I'm turning around to backtrack and explore an alternative route that might lead me closer to the red box, since it's not visible in the current corridor.
$t' = 3$	My plan is to approach the closed purple door. From my current position (at (0,0) facing South), I'll move forward to get closer (to (0,1)), then turn right to face West and move forward two cells to reach a position just east of the door (at (-2,1)). Once there, I will toggle the door to open it and step through, expecting that the red box may be found in the adjacent area.

(b) Scene Description representation ($t' = \{0, 1, 2, 3\}$). Only the first four steps of intentions are shown.

Figure 13: Intended outputs at each decision step under two observation representation methods.

Case 2 This case illustrates a scenario ($m = 10$) in which the LLM-as-agent (gpt-4.1-mini) successfully completes the task under the Scene Description representation but fails under the Symbolic Cell representation. This task, which requires picking up the purple ball located to the right of the agent's initial position, is categorized as having easy difficulty and features a single-room layout with medium object density. The agent successfully completes the mission in the first decision step ($t' = 0$) by directly following the observation description: `"You see a purple ball 2 steps right."`. Its generated intention reflects a straightforward interpretation and execution of the instruction, indicating that the Scene Description representation effectively guides the agent's behavior in simple scenarios. However, the agent struggled with computing the object's location under Symbolic Cell representation. As shown in Figure 15, despite observing the target from the initial state, the agent turned left while generating the intention, indicating a misalignment between its planned intention and executed actions, as well as a misinterpretation of the spatial relationship.

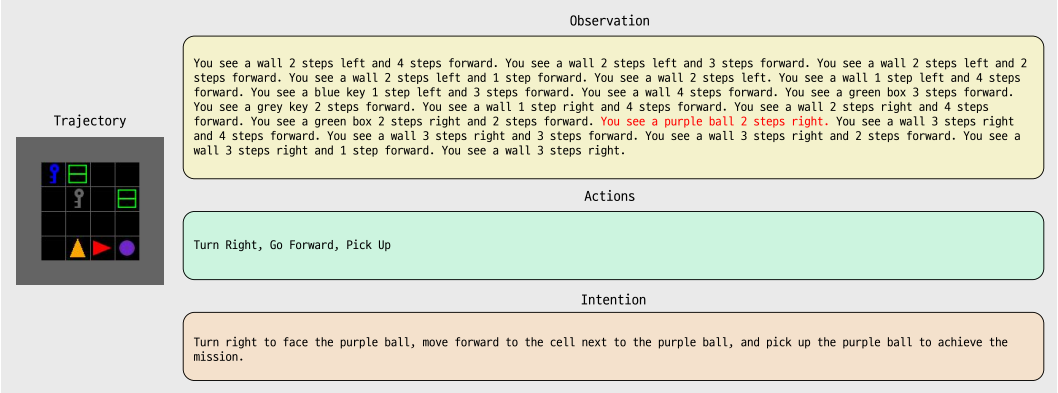


Figure 14: Visualization of the observation, output actions, generated intention, and corresponding trajectory under the Scene Description representation. The mission is to pick up the purple ball.

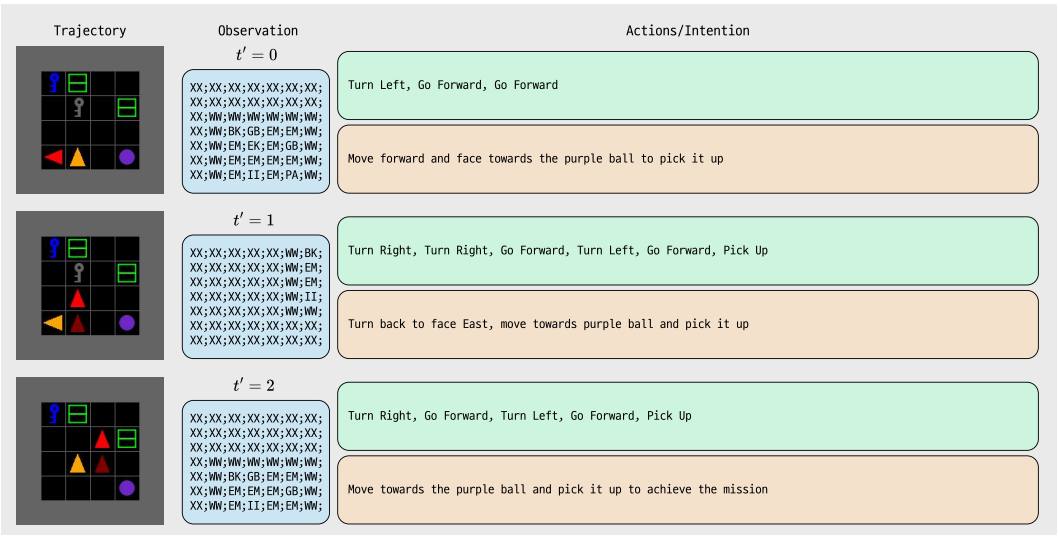
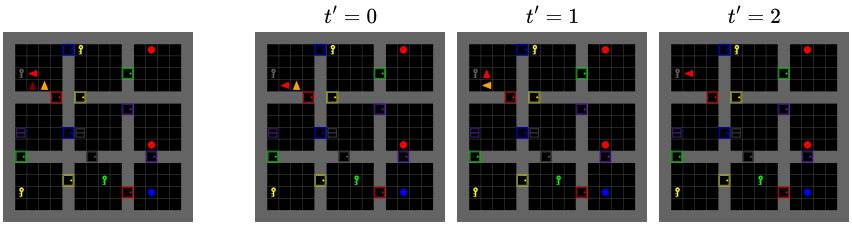
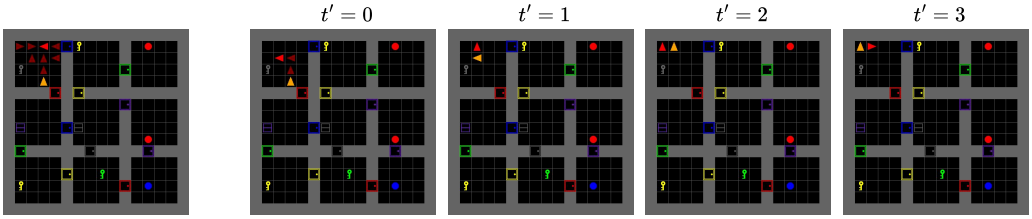


Figure 15: Visualization of the observation, output actions, generated intention, and corresponding trajectory under the Symbolic Cell representation. The mission is to pick up the purple ball. Only the first three steps of intentions are shown.

Case 3 This case illustrates a scenario with $m = 10$, in which the LLM-as-agent (gpt-4.1-mini/think) successfully completes the task under the Scene Description representation but fails under the Symbolic Cell representation. Figure 16 depicts the corresponding success and failure trajectories for Scene Description and Symbolic Cell, respectively. As shown in Figure 17, the LLM-as-agent continues to struggle in computing the correct action sequence needed to reach its intended position.

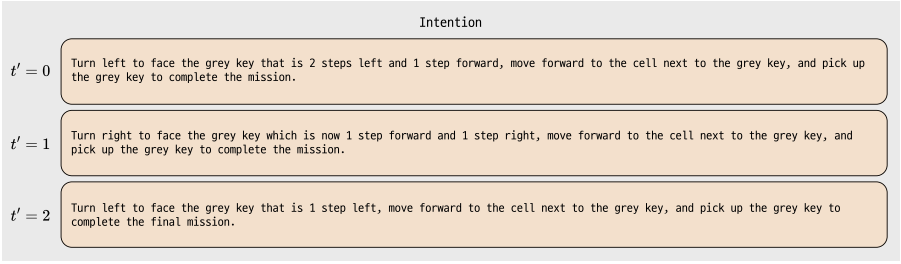


(a) Successful trajectory (Scene Description).

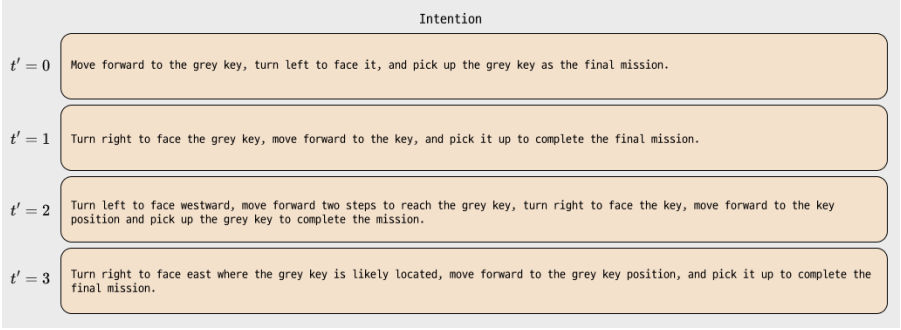


(b) Failure trajectory (Symbolic Cell). Only the first four steps of the trajectory are shown.

Figure 16: Visualization of agent trajectories under the Symbolic Cell representation and Scene Description. The mission is to pick up the grey key.



(a) Scene Description representation ($t' = \{0, 1, 2\}$).



(b) Symbolic Cell representation ($t' = \{0, 1, 2, 3\}$). Only the first four steps of intentions are shown.

Figure 17: Intended outputs at each decision step under two observation representation methods.

D System Prompt

```
System prompt:
You are an agent exploring a grid world. Your goal is to achieve specified missions.
Observation Description:
You will be given partially observed 7 by 7 grid cells, where each cell is represented by 2 characters. The first
character represents the color of cell and the second character represents the type of cell. Each cell can be
either empty (EM), contain a wall (WW), or contain an object. The color can be either red (R), green (G), blue
(B), purple (P), yellow (Y), or grey (E). The object can be either a key (K), a closed, unlocked door (D), a
closed, locked door (L), an open door (__), a ball (A), or a box (B). You are represented by 'II', and your facing
orientation is given as a separate text. You can face North, East, South, or West. You can only reveal the cells
that are within your field of view, which is a 7 by 7 square in front of you. Your view can be blocked by walls
and closed doors. Other objects do not block your sight. Unreachable cells are represented by 'XX'.
Action Description:
You can perform the following actions: Turn Left, Turn Right, Go Forward, Pick Up, Drop, Toggle. Turning left or
right will rotate the agent 90 degrees in the corresponding direction. Going forward will move the agent one cell
in the direction it is facing. You can only move to an empty cell or a cell with an open door. Picking up will
pick up the object in the cell in front of you. Dropping will drop the object you are currently carrying in the
cell in front of you. You can only drop an object in an empty cell. Toggling will toggle the state of the door in
the cell in front of you. To open a door, you just toggle it regardless of whether you have a key with a matching
color in your inventory. However, To open a locked door, you must have a key with a matching color in your
inventory and then toggle the door. You can only carry one object at a time. To pick up a new object, you must
first drop the object you are currently carrying.
Mission Description:
You will be given missions to achieve in the grid world. The missions can be to go to a specific object, pick up a
specific object, open a specific door, or put one specific object next to the other specific object. The
followings are the examples of the missions, and how to achieve them: 1. Go to the red key: You need to move to
the cell next to where the red key is located, and turn to face the red key. 2. Pick up the green ball: You need
to move to the cell next to where the green ball is located, and pick up the green ball. 3. Open the blue door:
You need to move to the cell next to where the blue door is located, and open the blue door. If the blue door is
locked, you need to find the blue key, pick up the blue key, move to the cell next to where the blue door is
located, and open the blue door. 4. Put the purple box next to the yellow box: You need to move to the cell next
to where the purple box is located, pick up the purple box, move to the cell near where the yellow box is located,
and drop the purple box next to the yellow box. The objects are specified by their type and color.
Strategic Guide:
 You can use the following strategies to explore the grid world efficiently: 1. Since the grid is partially
observed, you need to explore the grid to find the objects to achieve the missions. Remember that you can always
look for the closed doors to access new areas and to find new objects. 2. When the doors are locked, you need to
find the keys with the matching color to unlock the doors. Since the door are important to access new areas, you
should prioritize finding the keys for the locked doors. 3. If the doors are blocked by the objects, you can pick
up the objects to open the door. When the blocked door is also locked, you should first move the object to the
cell not in your path, pick up the key, unlock the door, and move the object to the cell in front of the door.
This is because you can carry only one object at a time, thus cannot move blocking objects while holding the key.
4. You don't have to keep the objects you picked up in your inventory to achieve the 'Pick Up' mission. Once you
pick up the object, the mission is considered achieved, and you can drop the object to empty your inventory. 5.
The grid is static, and the objects do not move unless you interact with them. 6. After picking up the object to
unblock the cells, you can drop the object to empty your inventory. However, you should be careful not to block
your path with the dropped object. 7. When you have to pick up an object to open the path, but you are already
carrying an object, you should drop the object in the cell not in your path. 8. When you encounter a locked door,
you can try to find the key with the matching color to unlock the door.
```

Figure 18: System Prompt. Text colors represent different system prompt categories.

Figure 18 presents the system prompt, color-coded by categories: observation description, action description, mission description, and strategic guide. The observation description explains the symbols used when the agent receives observations represented in the Symbolic Cell format. The Action Description provides detailed explanations of each action, including the conditions under which actions like drop and toggle can be executed. The Mission Description outlines how to successfully complete tasks (`GoTo`, `PickUp`, `OpenDoor`, and `PutNextTo`). For instance, in a `GoTo` mission, the agent fails if it moves next to the target object without facing it; without this information, the LLM may not be able to infer the cause of failure. The Strategic Guide informs the LLM that it operates in a partially observable setting and offers strategies for effective exploration. For example, in contrast to single room settings, exploration in multiple room environments requires navigating through doors, making strategic guidance supportive of solving tasks.

Table 3: Total number of actions taken by o3-mini under different ablations of the system prompt components. The maximum number of steps the agent can take per episode is 200.

Action Description	Mission Description	Strategic Guide	Seed 1	Seed 2	Seed 3	Seed 4	Seed 5
✓	✓	✓	**18**	15	**21**	**11**	**5**
✓	✓	✗	20	40	Fail	101	**5**
✓	✗	✓	47	**11**	171	23	7
✓	✗	✗	93	34	126	12	49
✗	✓	✓	51	13	108	46	**5**
✗	✓	✗	27	74	23	**11**	8
✗	✗	✓	33	13	Fail	39	**5**
✗	✗	✗	99	98	112	17	43

Table 4: Total number of actions taken by gpt-4.1-mini under different ablations of the system prompt components. The maximum number of steps the agent can take per episode is 200.

Action Description	Mission Description	Strategic Guide	Seed 1	Seed 2	Seed 3	Seed 4	Seed 5
✓	✓	✓	Fail	Fail	111	28	22
✓	✓	✗	Fail	Fail	152	40	Fail
✓	✗	✓	**22**	Fail	Fail	12	**10**
✓	✗	✗	71	Fail	Fail	11	29
✗	✓	✓	187	Fail	Fail	10	Fail
✗	✓	✗	Fail	Fail	**33**	43	60
✗	✗	✓	140	Fail	Fail	37	68
✗	✗	✗	Fail	Fail	Fail	**7**	Fail

D.1 Ablation Study on System Prompt Design

To evaluate the effectiveness of our system prompt, we conduct an ablation study using the various models. We test eight different configurations of the system prompt by selectively including (✓) or excluding (✗) the Action Description, Mission Description, and Strategic Guide. Each configuration is evaluated across five random seeds to ensure robustness. In this experiment, the agent receives Symbolic Cell observations and is prompted to generate a single action from the defined action space at each decision step, t'. When all components are removed, the agent requires the most steps to complete the task. In contrast, including all components enables the agent to solve tasks more efficiently. While there is one failure case when the Strategic Guide is either excluded or used in isolation, its inclusion generally improves performance. The two failure cases shown in Table 3 occurred when the mission required picking up the grey box, but the LLM instead executed rollouts near the green box. Table 4 presents the results of an ablation study on the system prompt components using gpt-4.1-mini. When all components were removed, the agent succeeded only once, whereas including all components resulted in three successful completions.

Case 1 Figure 19 shows the trajectory of the agent in seed 3, illustrating a successful example of o3-mini when all three system prompt components are included. Given the instruction `"Pick up the grey box"`, the agent, initialized at the orange starting position, successfully navigates to the grey box and completes the pickup task.

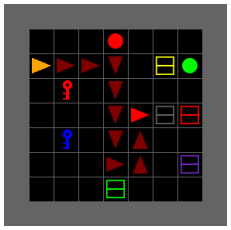


Figure 19: Trajectory of o3-mini on seed 3, illustrating a successful completion when all system prompt components are included.

Figure 20 presents a detailed rollout corresponding to Figure 19. Initially, o3-mini correctly identifies the location of the grey box and generates an action to navigate toward it. However, at time step $t' = 4$, o3-mini mistakenly identifies the green box as the grey box and generates an action to move toward the incorrect target. From $t' = 4$ to $t' = 15$, the agent moves toward the green box and attempts to pick it up. After picking up and dropping the green box, o3-mini responds with an intention to explore to find the grey box and turns to the left. From $t' = 16$, the LLM correctly recognizes the grey box, navigates to its location, and successfully completes the pick up task.

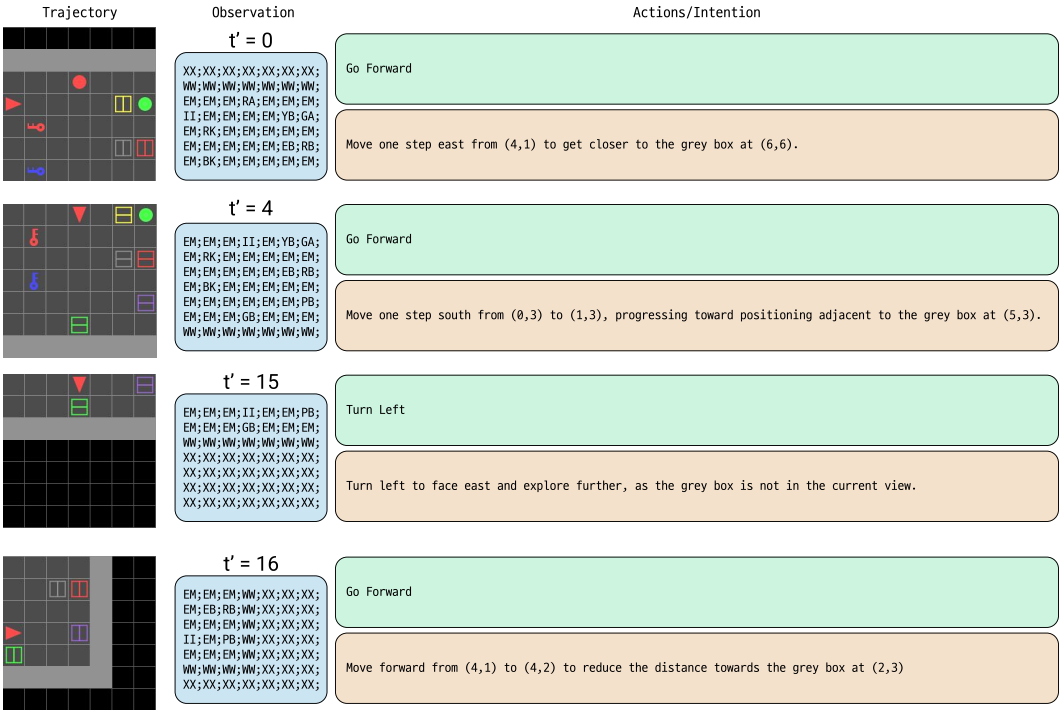


Figure 20: Time-step observations, actions, and intentions of o3-mini on seed 3, successfully completed with all system prompt components included.

Case 2 Figure 21 shows the trajectory of the agent in seed 4, illustrating a successful example of gpt-4.1-mini when all three system prompt components are included. Given the instruction `"Pick up the green key"`, the agent, initialized at the orange starting position, successfully navigates to the green key and completes the pickup task.

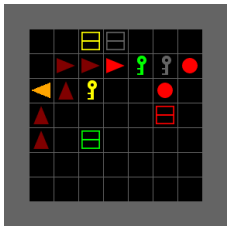


Figure 21: Trajectory of gpt-4.1-mini on seed 4, illustrating a successful completion when all system prompt components are included.

Figure 22 presents a detailed rollout corresponding to Figure 21. In the early stages of the trajectory, the agent has not yet observed the green key, but the LLM generates the action based on non-existent objects, indicating hallucination. At time steps $t' = 2$, $t' = 8$, and $t' = 15$, the LLM repeatedly responds with actions intended to approach the green key or green ball, although these objects are not present in the observation. These cases demonstrate that during the exploration phase, the LLM produces incorrect responses grounded in hallucinated object references. After the green key is actually observed at $t' = 18$, the LLM begins generating the appropriate action, completing the pick-up task successfully.

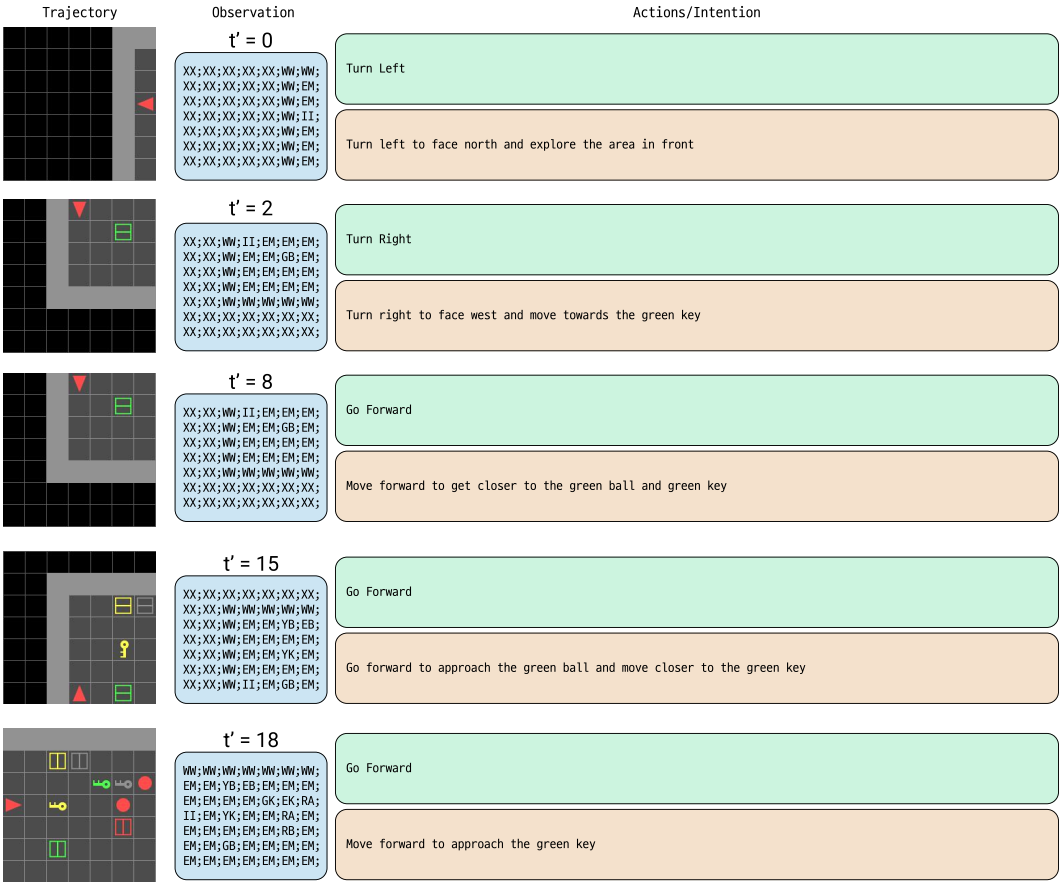


Figure 22: Time-step observations, actions, and intentions of o3-mini on seed 4, successfully completed with all system prompt components included.

Case 3 Figure 23 shows the trajectory of the agent in seed 5, illustrating a failure example of gpt-4.1-mini when all three system prompt components are excluded. The agent, initialized at the orange starting position with the mission `"Pick up the blue key"`, overlooks the blue key and passes by it, resulting in inefficient exploration and task failure.

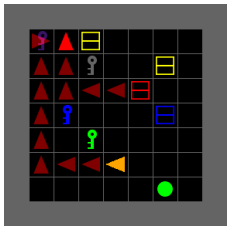


Figure 23: Trajectory of gpt-4.1-mini on seed 5, illustrating a failure when all system prompt components are excluded.

Figure 24 presents a detailed rollout corresponding to Figure 23. In seed 5, the blue key—the goal of the mission—is already visible in the observation at the initial step ($t' = 0$). However, due to the absence of the Strategic Guide component in the system prompt, the LLM exhibits behavior attempting to move toward the blocked area at $t' = 2$, despite the presence of a blocker. Additionally, as observed at $t' = 9$, $t' = 23$, and $t' = 199$, gpt-4.1-mini generates actions and intentions that indicate challenges in understanding object locations and directions within the Symbolic Cell structure.

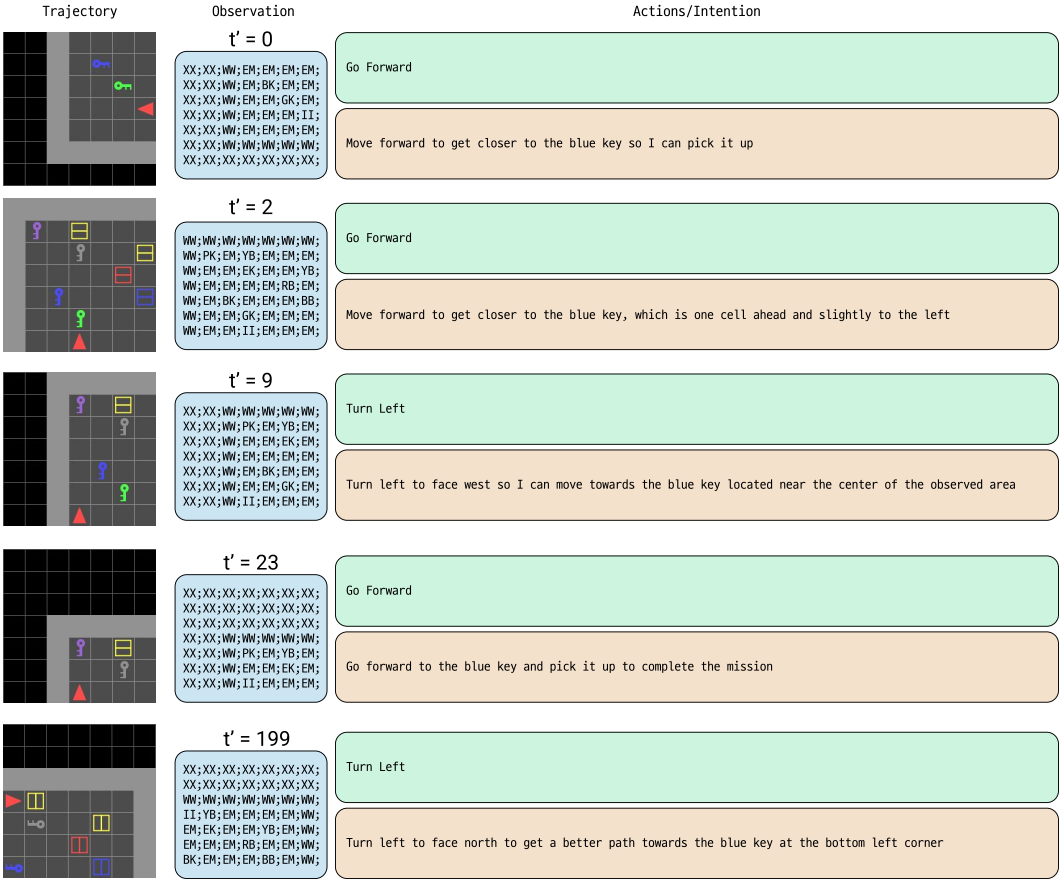


Figure 24: Time-step observations, actions, and intentions of gpt-4.1 on seed 5, which failed without all system prompt components.

E Prompts

We provide the prompts used in our experiments in Figure 25.

```
Your final mission is to pick up the purple ball.
You have no subgoals.
The following is the result of exectuing your immediate past actions:
Action0: Turn Left
Observation0:
XX;XX;XX;XX;WW;BK;GB;
XX;XX;XX;XX;WW;EM;EK;
XX;XX;XX;XX;WW;EM;EM;
XX;XX;XX;XX;WW;EM;II;
XX;XX;XX;XX;WW;WW;WW;
XX;XX;XX;XX;XX;XX;XX;
XX;XX;XX;XX;XX;XX;XX;
Direction0:West
Inventory0:nothing
Action1: Go Forward
Observation1:
XX;XX;XX;XX;XX;WW;BK;
XX;XX;XX;XX;XX;WW;EM;
XX;XX;XX;XX;XX;WW;EM;
XX;XX;XX;XX;XX;WW;II;
XX;XX;XX;XX;XX;WW;WW;
XX;XX;XX;XX;XX;XX;XX;
XX;XX;XX;XX;XX;XX;XX;
Direction1:West
Inventory1:nothing
Action2: Go Forward
Observation2:
XX;XX;XX;XX;XX;WW;BK;
XX;XX;XX;XX;XX;WW;EM;
XX;XX;XX;XX;XX;WW;EM;
XX;XX;XX;XX;XX;WW;II;
XX;XX;XX;XX;XX;WW;WW;
XX;XX;XX;XX;XX;XX;XX;
XX;XX;XX;XX;XX;XX;XX;
Direction2:West
Inventory2:nothing
Your current observation:
XX;XX;XX;XX;XX;WW;BK;
XX;XX;XX;XX;XX;WW;EM;
XX;XX;XX;XX;XX;WW;EM;
XX;XX;XX;XX;XX;WW;II;
XX;XX;XX;XX;XX;WW;WW;
XX;XX;XX;XX;XX;XX;XX;
XX;XX;XX;XX;XX;XX;XX;
You are facing West.
You are carrying nothing.
Based on the observation and information, output the actions you want to take. You can take up to
10 actions in a single turn. You should format the chosen actions between <actions> and </actions>,
separated by ',' without the whitespace between each action. For example, if you want to turn left,
go forward, and pick up the object in front of you, you should output '<actions>Turn Left,Go
Forward,Pick Up</actions>'. You should also output your intentions of the actions between
<intentions> and </intentions>. For example, if you want to turn left to explore the left side of
the grid, you should output '<intentions>Explore the left side of the grid</intentions>'.
```

Figure 25: An example of the input prompt at $t' \geq 1$.

F Examples Across Difficulty levels

As discussed in Section 4, we randomly sample 45 tasks across nine layout configurations, defined as the Cartesian product of three object-density levels and three room-number levels, using five random seeds. Tasks are categorized into four difficulty levels according to the number of steps required for completion by the Oracle solver. We provide illustrative examples of each difficulty level in Figure 26. The hard and super-hard levels require the agent to explore previously unseen rooms in order to locate the target.

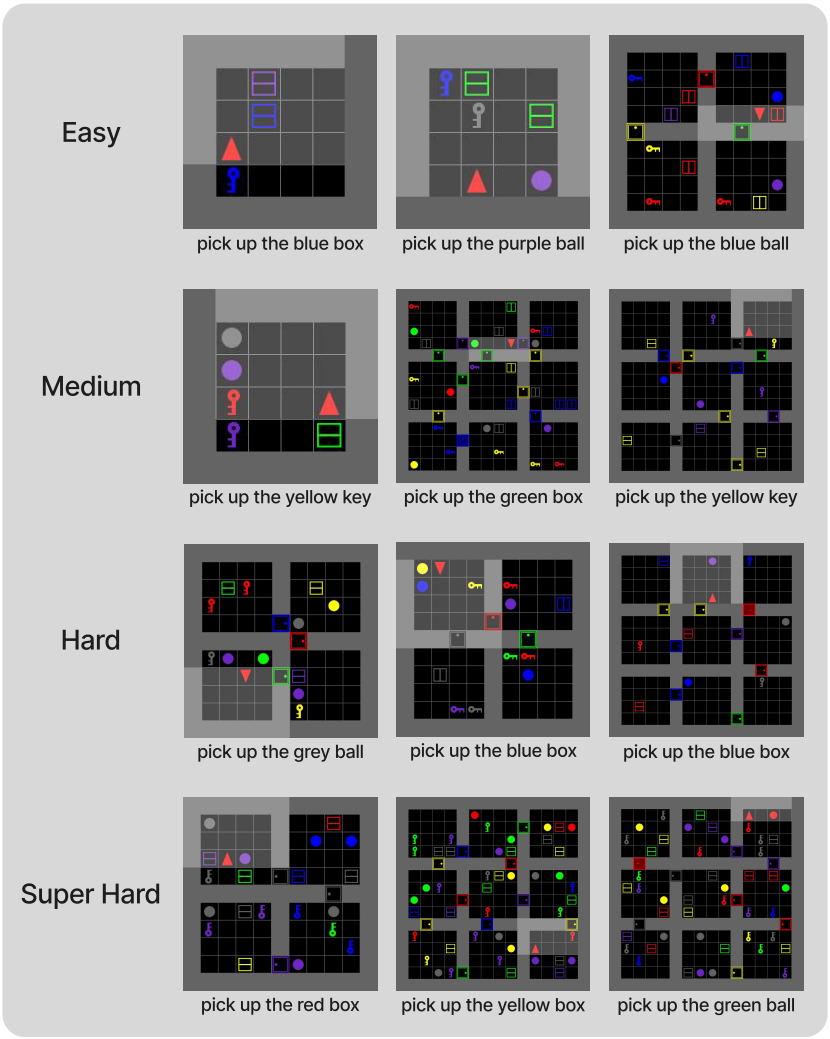


Figure 26: Illustrative examples of behavioral difficulty levels: easy, medium, hard, and super-hard. Shaded regions denote the agent's initial partial observations, and the corresponding missions are described beneath each layout.