

Grounding Large Language Models with Natural Language World Models

Honors Thesis for the Department of Computer Science

William Xia

Tufts University, 2025

Abstract

Embodied planning tasks often require agents to use their environment to solve unfamiliar problems. A common obstacle when using large language models (LLMs) for embodied tasks involves grounding their reasoning to the environment itself. Prior work has used various means to ground LLMs in embodied environments, most of which involve complex overhead architectures. Such architectures can constrain models to specific domains, limiting their usefulness in a broader range of tasks. Additionally, they often come with high computational requirements and introduce points of failure that could harm agent reasoning abilities rather than enhance them. In this work, I examine how prompting alone can be used to ground LLMs. Specifically, I determine if LLMs can use natural language world models (NLWMs) to form accurate internal representations of their environment. I present ESCGRIDENV, a novel 2D OpenAI Gym environment for testing embodied reasoning and planning. Additionally, I analyze the performance of 3 off-the-shelf LLMs at reasoning tasks within ESCGRIDENV and compare their performance to that of a simple reinforcement learning agent. My results show that NLWMs increases the likelihood that LLMs utilize elements of their environment when performing embodied puzzle solving, but that natural language alone is insufficient for LLMs to form accurate internal representations of their environment, suggesting a disconnect between semantic knowledge and an implicit understanding of embodied environments, laying a foundation for future work in embodied insight problem solving.

The implementations of ESCGRIDENV and the reasoning agents described in this paper can be found at https://github.com/bill-from-ri/EscGridEnv_Public/.

Contents

Abstract	i
List of Figures	v
List of Tables	vi
1 Introduction	1
2 Background	4
2.1 Insight Problem Solving	4
2.2 Insight Problem Solving with LLMs	5
2.3 Grounding Techniques	6
3 The EscapeGrid Environment	9
3.1 Design Philosophy	9
3.2 Previous Work with 2D Simulation Environments	10
3.3 The ESCGRIDENV Environment	11
3.3.1 Observations and Action Space	11
3.3.2 Objects and Transition Dynamics	12
3.3.3 Level Configuration	13
4 Reinforcement Learning Experiments	15

4.1	Experiment Setup	15
4.1.1	Levels	16
4.2	Results	17
4.2.1	Implications	19
5	Large Language Model Experiments	20
5.1	Experiment Setup	20
5.2	Independent Variables	21
5.2.1	Natural Language World Models	22
5.2.2	Puzzle Difficulty	22
5.2.3	Observation Encodings	23
5.2.4	Prompting Methods	24
5.3	Metrics - Operationalizing the Hypotheses	24
6	Large Language Model Results	25
6.1	Effect of Natural Language World Models	25
6.2	Effect of Puzzle Difficulty	25
6.3	Vision Encoding Method	27
6.4	Prompting Technique	28
6.5	Model	28
6.6	Qualitative Analysis	29
6.6.1	Example 1	31
6.6.2	Example 2	31
6.6.3	Example 3	31
7	Reinforcement Learning vs. LLMs	32
7.1	Experiment Setup	32

7.2	Results	32
7.2.1	Implications	34
8	General Discussion	36
8.1	Implications Within This Study	36
8.2	Implications Beyond This Study	37
8.3	Reframing Grounding as a Multi-Stage Process	38
8.4	Summary	38
9	Future Work	39
10	Conclusion	40
11	Limitations	42
	Acknowledgments	43
	References	44
A	Prompting Strategies	48
A.1	Without NLWM, Zero-Shot Prompting	48
A.2	Without NLWM, CoT Prompting	48
A.3	With NLWM, Zero-Shot Prompting	49
A.4	With NLWM, CoT Prompting	50

List of Figures

2.1	The insight problem solving framework described in Sarathy, 2018	5
3.1	Example configuration of the ESCGRIDENV environment.	10
4.1	Comparison between PPO model and human participants.	18
5.1	Illustration of all the components of the experiment.	21
5.2	ESCGRIDENV configurations used in my experiments. Starting from the top left going clockwise, they are Levels 1, 3, 5, 12, 11, 9.	22
5.3	Comparison between visual representation and the two encoding methods.	23
6.1	Effect of NLWM prompting by metric.	26
6.2	Effect of puzzle difficulty by metric.	26
6.3	Effect of encoding method by metric.	27
6.4	Effect of prompting technique by metric.	28
6.5	Effect of model choice by metric.	29
7.1	Level 5 Variants.	33
7.2	Level 5 Variant Results.	34

List of Tables

3.1	Object – String Encoding Mapping	14
6.1	Pairwise p -values for performance difference between Llama3, Qwen2.5, and DeepSeek-R1. The first value is for Interaction and the second is for Completeness.	29
6.2	Example reasoning traces for qualitative analysis.	30

1 Introduction

Robots operating in high-risk or inaccessible environments, including search-and-rescue missions or extraplanetary exploration, must often solve novel problems independently (Chitikena et al., 2023, Verma et al., 2023). In such settings, human oversight is not guaranteed, meaning that agents must be able to autonomously generate plans and adapt to unpredictable environmental conditions. Effective autonomous planning depends on the agent’s ability to understand and leverage its environment, which requires the ability to form accurate internal models of the agent’s surroundings.

In recent years, large language models (LLMs) have gained attention for their strong performance in a wide range of natural language processing (NLP) tasks (Grattafiori et al., 2024; Qwen et al., 2025; DeepSeek-AI et al., 2025). Two domains that have seen intense research interest are reasoning and planning. Although much of the research in this domain relates to reasoning in language domains such as puzzle solving, commonsense reasoning, and mathematics (Giadikiaroglou et al., 2024; Wei et al., 2023, Plaat et al., 2024), there is also a growing body of research examining the embodied reasoning capabilities of LLMs (Ahn et al., 2022; Dasgupta et al., 2023a).

However, within embodied domains, LLMs face a significant obstacle: they are not inherently grounded in the physical world Ahn et al., 2022. Their knowledge, derived from pretraining on Internet-scale text corpora, is disconnected from the specific environments in which they must act (Kambhampati, 2024). Without grounding, LLMs risk hallucinating environment states, misunderstanding available actions, or producing plans that cannot be executed.

Therefore, grounding methods are needed for LLM agents operating in embodied domains. Past research has explored many solutions to this problem, most of which use external architectures to ground LLMs to their environments. Such approaches include reinforcement learning (RL) methods (Ahn et al., 2022), continuous feedback (Huang et al., 2022; Dasgupta et al., 2023a), knowledge representations (Shukla et al., 2023), and symbolic world models (Dainese et al., 2024; Smirnov et al., 2024). While effective in controlled settings, these methods suffer from several limitations.

First, external grounding (especially RL methods) typically comes with an increased computational overhead, both increasing the costs needed to train such systems, as well as increasing the time needed for these systems to run in real-time reasoning situations (Ahn et al., 2022), factors that may limit the use of autonomous agents in rapidly-evolving environments (Goel et al., 2022). Second, they often rely on fixed assumptions about the environment’s structure and affordances, which can harm agent flexibility when encountering novel scenarios (Dasgupta et al., 2023b). Additionally, these architectures introduce additional points of failure: each added component becomes a potential bottleneck or source of error (Dainese et al., 2024).

In this work, I investigate whether prompting alone, using minimal architectural additions, can sufficiently ground LLMs to their environment. Specifically, I explore the use of natural language world models (NLWMs) and whether they are sufficient to induce an implicit understanding of the environment in LLMs. Rather than encoding environment information in learned policies or symbolic structures, NLWMs offer concise natural language descriptions of the environment’s structure and dynamics, leveraging LLMs’ familiarity with Internet-scale natural language data to synthesize internal models of the described environment. My approach probes a potential disconnect between semantic and implicit understanding in LLMs. That is, the difference between what LLMs say they understand about their surroundings and the degree to which they can perform accurate reasoning with the environment.

To test my approach, I introduce ESCGRIDENV, a novel 2D OpenAI Gym environment (Brockman et al., 2016) extended from MiniGrid (Chevalier-Boisvert et al., 2023). This environment is specifically designed to test insight problem solving skills, as it presents a wide array of unfamiliar objects and transition dynamics for agents to learn and exploit. Additionally, ESCGRIDENV is constructed with a minimalist action space and simple graphics to abstract away potential distracts, allowing LLMs to focus entirely on reasoning and planning. ESCGRIDENV was also designed to allow for easy reconfiguration, allowing for the creation of levels that have a wide range of difficulties. I use this feature to test the reasoning and planning capabilities of LLMs in increasingly difficult embodied scenarios.

To demonstrate that ESCGRIDENV tasks cannot be solved without forming an internal representation of the environment, I test the performance of a basic RL agent on various tasks within the environment. I compare those results with human performance to show the importance of advanced reasoning on ESCGRIDENV tasks.

Additionally, I compare the performance of 3 off-the-shelf, open-source LLMs at reasoning tasks within the ESCGRIDENV environment. I demonstrate the effect that NLWMs have on planning abilities, and I also compare LLM performance with the RL

results to show how an internal representation improves performance in embodied planning tasks.

In summary, the contributions of this paper are as follows:

- I introduce ESCGRIDENV, a novel 2D OpenAI gym environment designed for embodied insight problem solving. I demonstrate its complexity by comparing the performance of an RL model and human participants on tasks in the environment.
- I demonstrate the performance of 3 off-the-shelf, open-source LLMs at reasoning tasks within the ESCGRIDENV environment, and compare their results with those of reinforcement learning agents and human participants.
- I provide a statistical analysis of prompting methods and other factors contributing to the implicit grounding of LLMs in ESCGRIDENV tasks.

2 Background

2.1 Insight Problem Solving

Insight problem solving (IPS; sometimes called creative problem solving or CPS) refers to a multi-step problem-solving framework that has seen considerable research in both cognitive computer science over the last few decades (Ohlsson, 1992; Boden, 2010; Sarathy, 2018; Gizzi et al., 2022; Nair et al., 2024). IPS problems are those where an agent’s initial concept space does not allow the agent to form a solution. Only by experimenting with the environment and expanding the concept space can agents overcoming these impasses and perform IPS.

How this expansion occurs may vary. Boden, 2010 defines three means of expanding the concept space (what she defines as “forms of creativity”): *exploration*, *combination*, and *transformation*. The first of these methods, *exploration*, is the most straightforward: agents explore their environment and discover entirely new concepts, such as a new object. It usually occurs when an agent is first introduced to an environment and begins discovering the most obvious concepts available.

Combination describes the novel fusion of two or more familiar concepts to create a new concept. An example of combinational creativity in an embodied domain would be the attachment of a tool and a stick to increase the tool’s effective range. Combinational creativity is one step more complicated than exploratory creativity because it requires agents to compose together potentially disparate ideas to distill new concepts.

The final form of creativity described by Boden is *transformation*, wherein familiar concepts are re-conceptualized in new ways. This is the most abstract form of creativity, with a good example appearing in Duncker, 1945. In that study, participants needed to re-conceptualize a box of thumb-tacks as a contain for a candle, which they were tasked with affixing to a wall. Such tasks, defined by Sarathy and Scheutz, 2018 as MacGyver Problems, are critical in IPS scenarios where limited resources are needed to solve complex problems, such as when astronauts and NASA ground control had to

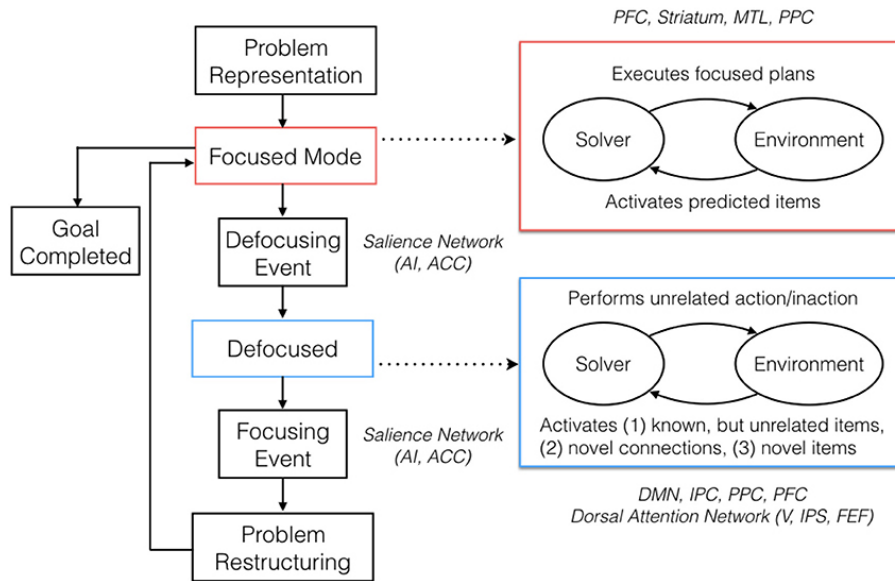


Figure 2.1: The insight problem solving framework described in [Sarathy, 2018](#).

create an improvised carbon dioxide scrubber during the Apollo 13 space mission.

A shared requirement across all three forms of creativity described by Boden is the capability of agents to form an internal representation of their surroundings. [Boden, 1998](#) describes this requirement as one of the largest barriers for computational systems attempting to emulate human IPS: “The reasons for this, in brief, are the difficulty of approaching the richness of human associative memory, and the difficulty of identifying our values and of expressing them in computational form.” Boden’s observation was accurate for the time, but in the years since 1998, systems approaching the capabilities of humans associative memory have emerged, most of which are derived from transformer architecture [Vaswani et al., 2017](#). In the next section, I describe the utility of LLMs, especially in relation to IPS and related reasoning/planning problems.

2.2 Insight Problem Solving with LLMs

Large Language Models (LLMs) are advanced neural networks trained on vast corpora of text data to predict and generate human-like language. Their architecture is based on the Transformer model, introduced by [Vaswani et al., 2017](#), which utilizes self-attention mechanisms to capture contextual relationships between words in a sequence. This design enables LLMs to process and generate coherent and contextually relevant text, making them highly effective in various natural language processing tasks.

The capabilities of LLMs resonate with [Boden, 1998](#)’s insights on human creative processes. Boden emphasized the richness of human associative memory as a corner-

stone of creativity and problem-solving, noting the challenges in replicating this aspect in computational systems. LLMs, with their ability to model complex linguistic patterns and associations, represent a significant step toward emulating the associative processes inherent in human cognition.

In tasks requiring reasoning and planning, particularly within disembodied domains, LLMs have demonstrated remarkable proficiency. Techniques such as chain of thought (CoT) prompting, as explored by [Wei et al., 2023](#), guide LLMs through intermediate reasoning steps, enhancing their performance in complex tasks such as arithmetic, commonsense reasoning, and symbolic logic ([Giadikiaroglou et al., 2024](#); [Plaata et al., 2024](#)). Because these domains are primarily language-based, they allow LLMs to operate effectively without the need for additional grounding in physical or sensory contexts.

In this work, I focus on planning in embodied domains. Within the IPS framework (see Figure 2.1), planning is a necessary step both in the initial “Focused Mode” setting as well as in the later “Unfocused Mode” setting. In embodied domains, agents requires a rich, implicit understanding of the environment to plan effectively. However, past methods have not needed such levels of understanding. Historically, embodied problem solving been dominated by RL agents. These models use extensive training to learn tasks, consequently encountering trouble adapting to novelties, such as new objects and even new configurations of the environment ([Shukla et al., 2023](#), [Goel et al., 2022](#)). In the ESCGRIDENV task, where learning precise object interactions is critical to solving tasks, and rewards are often sparse, RL approaches becomes infeasible without monstrously long training times and/or significant overhead in the form of custom reward design and/or hierarchical methods.

More recently, LLMs have gained attention for their performance in embodied reasoning tasks. They have achieved success primarily through approaches that seek to ground language models to the action space of embodied agents and the transition dynamics of the environment. In the next section, I discuss various past approaches for grounding LLMs in embodied domains.

2.3 Grounding Techniques

Past work has explored various methods to integrate LLMs into embodied reasoning tasks. For instance, [Ahn et al., 2022](#) introduces SayCan, which grounds LLMs using a “learned value function”, a reinforcement learning model used to determine the feasibility of natural language instructions given the capabilities of the LLM’s embodiment

as well as the environment the embodiment is acting within. SayCan also utilizes a vision model to ground its planner to images of the environment.

Another common form of grounding is continuous feedback. [Huang et al., 2022](#) leverages LLMs' ability to generate self-dialogue, allowing agents to plan and adjust actions through internal reasoning processes with continuous feedback. This approach uses multi-modal feedback to provide LLMs with a diverse set of information with which to condition its actions.

[Dasgupta et al., 2023b](#) also incorporates continuous feedback into their framework. They do so by introducing a three-module system consisting of a Planner, an Actor, and a Reporter. The LLM-powered Planner performs all the reasoning while the Actor executes actions within the environment and a Reporter provides language feedback to the Planner based on the results of the agent's actions.

Symbolic grounding methods have also been explored to improve the performance of LLM reasoning and planning in embodied domains. For example, [Dainese et al., 2024](#) tasks LLMs with building world models out of code. First proposed in RL research ([Ha and Schmidhuber, 2018](#)), a world model predicts the underlying dynamics of how actions taken by agents within an environment transition the environment to new states. In [Dainese et al., 2024](#), the researchers develop a method to represent the environment's transition dynamics using code, which can be executed by an LLM to predict the result of actions taken within the actual environment. They present an intricate code review system used by their framework to ensure that generated code is syntactically correct and accurate to the environment it is modeling.

A relatively lightweight approach to grounding can be found in [Cheng et al., 2024](#), where instructions given to LLMs include object affordances, which the LLM selects from for its actions based on a probabilistic feasibility function. Object affordances are a concept introduced first in [Gibson, 1979](#) and later refined by [Chemero, 2003](#). In essence, affordances are relationships between objects in the environment and agents acting within the environment, where new action possibilities are created based on the capabilities of the agent and the transition dynamics of the environment. Importantly, they exist only as surface features of the agent-environment relationship. They merely describe action possibilities, not the consequences of those actions. As such, the amount of information they provide is much less than what is provided by a full world model, which seeks to represent both the ways in which the agent and environment can interact, as well as the result of those interactions.

While all of these methods have shown promise, most involve extensive architectural overheads, which can increase computational overhead and introduce potential

points of failure. In contrast, my approach seeks to simplify the grounding of LLMs into embodied reasoning tasks by minimizing architectural complexity. It remains unclear how much of grounding is needed for an LLM to form an implicit (not just declarative) understanding of its environment. This distinction is important, as an LLM may be able to explain the transition dynamics of an environment without being able to model them effectively.

I propose using natural language world models (NLWMs) to provide LLMs with textual representations of their environments. This method aims to ground LLMs using language alone, determining if they can form internal representations of their surroundings without relying on additional sensory inputs or complex external architectures. I elaborate on my definition of NLWMs in 5.2.1.

3 The EscapeGrid Environment

In this section, I discuss on the design of the ESCGRIDENV environment, including the environment’s design philosophy, the work that inspired it, novel objects added to the environment, and how I extended the the original MiniGrid ([Chevalier-Boisvert et al., 2023](#)) API to accommodate the functionality introduced by the EscapeGrid library.

3.1 Design Philosophy

The primary aim of ESCGRIDENV was to construct an embodied environment that meaningfully tests agents’ ability to perform IPS. While designing EscGridEnv, I prioritized three core principles:

1. **Optimization for Insight Problem Solving (IPS).** The environment is structured to require the discovery and exploitation of novel interactions between objects (i.e. hidden transition dynamics). It therefore rewards agents that are best at understanding and executing on the complexities of the environment and penalizes those that rely on rote memorization of successful action sequences.
2. **Simplified Action Space and Visuals.** The environment minimizes action space complexity by constraining agent capabilities to basic locomotion primitives. This ensures that problem difficulty stems primarily from environmental understanding rather than from mechanical skill acquisition. For similar reasons, ESCGRIDENV needed to be visually simple so as to avoid the need for advanced vision encoding methods.
3. **Configurable Difficulty.** ESCGRIDENV allows for flexible tuning of problem complexity through the modular combination of objects and discoveries, enabling structured tests of agent capabilities across varying levels of challenge. The purpose of this design principle is to determine the limits of AI agents in embodied reasoning tasks.

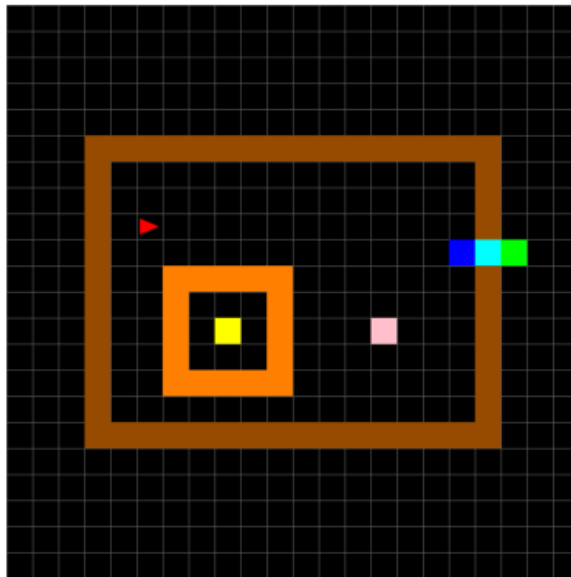


Figure 3.1: Example configuration of the ESCGRIDENV environment.

This design philosophy differentiates ESCGRIDENV from many previous 2D simulation environments, as discussed below.

3.2 Previous Work with 2D Simulation Environments

Several environments have previously been developed for studying embodied problem solving, but each has limitations that EscGridEnv seeks to address:

OpenAI Gym ([Brockman et al., 2016](#)) provides a broad benchmark suite primarily aimed at reinforcement learning. Gym environments generally emphasize action and visual perception challenges rather than logical reasoning tasks. Even still, this suite remains versatile enough to be extended in ways more conducive to my study, as discussed below.

PDDL Gym ([Silver and Chitnis, 2020](#)), extended from OpenAI Gym, introduces environments framed through symbolic PDDL specifications, focusing on logical planning. However, its focus is strongly tied to PDDL planning, a symbolic approach that is at odds with my intent of using natural language only. Additionally, PDDL Gym does not offer systematic control over problem difficulty through layered discoveries, instead presenting a wide variety of tasks that have no clear difficulty hierarchy in relation to one another.

MiniGrid ([Chevalier-Boisvert et al., 2023](#)), also extended from OpenAI Gym, offers a lightweight, flexible 2D grid environment tailored for RL agents. MiniGrid’s strengths

lie in its ease of customization and simple visual outputs. However, its transition dynamics are fairly simple, with only a single interaction that can be discovered via combinational creativity, limiting the depth and richness of discovery processes.

ESCAPE (Sarathy et al., 2024) is a framework for testing human insight problem solving across increasingly difficult levels. Although ESCAPE emphasizes concept discovery and problem restructuring, it is not engineered for direct machine learning interfaces or for training embodied agents.

EscGridEnv builds on these methods by combining the customizability of MiniGrid and the layered discovery elements of ESCAPE to create a unique platform for investigating embodied insight problem solving in artificial agents.

3.3 The ESCGRIDENV Environment

3.3.1 Observations and Action Space

In many ways, ESCGRIDENV remains very similar to MiniGrid. It is a modular 2D OpenAI Gym environment made up of $n \times m$ tiles, where each tile is either empty or occupied by an object (e.g., a wall, crate, or goal). I discuss objects in more detail in the next subsection.

Every time an action is executed in the environment (via the step function), an observation is returned, which is formatted in the same way as observations returned by native MiniGrid: a top-down view of the entire environment, with each tile encoded as a 3-tuple. The elements of the 3-tuple represent: the index of the object, the index of the object’s color, and 0.

The first major change from native MiniGrid is in the action space. MiniGrid has an action space with seven options, which I have constricted to three fundamental action primitives: turn-left, turn-right, and move-forward. This constriction was implemented to mimic the action space in Sarathy et al., 2024. In that study, the researchers demonstrate a sizable concept space that can be accessed solely through movement through the environment. By combining these action primitives, human participants could execute a wide variety of high-level plans. By heavily restricting the action repertoire, EscGridEnv ensures that challenge stems primarily from cognitive understanding of object relationships rather than complex motor control.

3.3.2 Objects and Transition Dynamics

Within the environment, I also introduce a collection of new `WorldObj` classes, which I define as a subclass of `WorldObj` called `EscObj`. `EscObjs` differ from the objects in native `MiniGrid` in two major ways. The first is that they are rendered with much more abstracted graphics. While `MiniGrid` uses distinctive visual representations of keys, doors, and lava, `ESCGRIDENV` uses colored squares for all `EscObjs`. The purpose of this change is to prevent models from relying on internal biases to perform problem solving, as the goal of `ESCGRIDENV` is to test the ability of agents to learn entirely foreign concept spaces via insight problem solving.

Herein lies second major difference between `EscObjs` and `MiniGrid`: the complexity of the environments' transition dynamics. In `MiniGrid`, most objects can only interact with the agent (the key and door being the only exception). `ESCGRIDENV` is a complex environment, where most `EscObjs` can interact with each other with the agent's intervention. The discovery of these interactions requires the initial discovery of properties associated with the individual objects in each interaction. The following is a full list of multi-object transition dynamics added in `ESCGRIDENV`:

- **Crates, Buttons, & Doors.** Pushing a crate (using the `move-forward` action while the agent is adjacent to and facing an orange object) toward an adjacent button (pink object) will open all `ButtonDoors` (blue objects) in the environment. Discovering this relationship requires the discovery that crates can be pushed.
- **Keys & Doors.** While holding a key (yellow object), the agent can walk into a `KeyDoor` (cyan object) to open it permanently. Discovering this relationship requires the agent to first discovery that moving into a key will cause the agent to pick up the key, turning it purple.
- **Crates & Keys.** Pushing a crate into a key will cause the key to disappear and the crate to turn into a `KeyCrate` (purple object). Additionally, pushing a crate while holding a key also causes the crate to turn into a `KeyCrate`. Upon doing so, the key disappears from the agent's possession. `KeyCrates` can be pushed into a `KeyDoor` to open the door permanently. Discovering this relationship requires the discovery that crates can be pushed.
- **KeyCrates & Crates.** Pushing a `KeyCrate` into a `Crate` will turn the `Crate` into a `KeyCrate`, thus allowing the agent to duplicate `KeyCrates`. This discovery can help solve problems that require the agent to open more `KeyDoors` than there are `Keys` in the environment.

In addition to the objects listed above, there is one other `EscObj` introduced in `ESCGRIDENV`: the `FalseWall` (brown object). This object appears to be a normal wall, but can be moved through. Crates and `KeyCrates` can be pushed through `FalseWalls` as well. The `FalseWall` introduces another discovery that the agent may have to learn to reach a Goal in the `ESCGRIDENV` environment. For many configurations of `ESCGRIDENV`, this property was the only one needed to solve the task. For example, Figure 3.1 can be solved by ignoring all other objects and moving forward 14 times (through a `FalseWall`), turning right, then moving forward 1 more time.

Put together, the new `EscObj` classes introduce a host of new objects and discoveries that must be made by the agent to accomplish its tasks. By adding multiple discoveries to the same environment, we can sequentially increase the difficulty of problems that agents must solve. Such a sequential difficulty system would allow us to more accurately measure the problem-solving capabilities of our models.

The process of discovery for puzzle solving has been explored in the past using GridWorlds similar to MiniGrid (Goel et al., 2022; Shukla et al., 2023). As in our work, the combination of multiple novelties is used to generate more complexity and more discoveries the agent must make.

3.3.3 Level Configuration

The `ESCGRIDENV` constructor contains a field called `grid_layout`, which can be used to initialize the starting configuration of objects in the environment. `grid_layout` expects a 2D matrix of strings, each of which must correspond to an object in the `ESCGRIDENV` environment. Table 3.1 shows the mapping of acceptable strings in `grid_layout` and their corresponding objects. Note that unlike `EscObjs`, the agent is placed in the environment with the `ESCGRIDENV` constructor parameters `agent_start_pos` and `agent_start_dir`, just as in MiniGrid.

Object	String Encoding
Wall	'w'
Goal	'g'
Crate	'c'
Button	'p'
ButtonDoor	'd'
PuzzleKey	'k'
PuzzleKeyDoor	't'
FalseWall	'f'
None	' '

Table 3.1: Object – String Encoding Mapping

4 Reinforcement Learning Experiments

To demonstrate that ESCGRIDENV requires agents capable of forming internal representations of their environment, I test the performance of a simple PPO agent [Schulman et al., 2017](#) on various configurations of the ESCGRIDENV environment. The hypothesis I seek to prove is that PPO agents, which have no means of forming environment representations on their own, will fail to effectively learn how to perform tasks in ESCGRIDENV.

I compare the results of PPO agents to that of human participants, whose performance is presented in [Sarathy et al., 2024](#). To ensure that results are comparable, I design a suite of ESCGRIDENV configurations to match the configurations used in that study.

4.1 Experiment Setup

I begin by formalizing the components of this experiment. The ESCGRIDENV action space is represented as $A = \{\text{turn-left}, \text{turn-right}, \text{move-forward}\}$. Every timestep t , the environment outputs observation from the observation space, $\omega_t \in \Omega$. Each observation ω_t is outputted by the environment’s transition function $T : (\Omega, A) \rightarrow \Omega$,

The PPO agent itself operates according to a trained policy $\pi : \Omega \rightarrow A$. Every timestep t , the agent receives ω_t and outputs a_t back to the environment, which uses T to determine ω_{t+1} .

Training and Testing

The PPO agent was trained on the Level 1 configuration (explained in Section 4.1.1) to emulate conditions faced by human participants in [Sarathy et al., 2024](#). Just as human participants were expected to perform the rest of the levels after only receiving Level 1 as guidance, so too was the LLM given only Level 1 to train with.

Training was performed across $5e5$ episodes with a discount factor of 0.99 on an NVIDIA RTX A6000 GPU.

The PPO agent was then evaluated on levels 1, 3, 5, 9, 11, and 12. These levels were chosen because they each introduce a new discovery (except for level 12, which tests agents on their understanding of all discoveries). The agent was evaluated on each level 100 times according to two binary metrics:

- **Interaction.** Each level has an associated discovery (or discoveries) that are considered necessary for the agent to find before it can complete the puzzle. For each run, I determined whether or not the agent performed an action that would reveal the discovery for that level (e.g. walking through a FalseWall, opening a KeyDoor). I explain what each level’s associated discoveries are in Section 4.1.1.
- **Completion.** As a harder metric than Interaction, I use level completion as a measure of agent performance. Completion is achieved when the agent reaches the Goal object.

4.1.1 Levels

Below, I provide a brief description of each ESCGRIDENV configuration used in this experiment. Each was inspired by a corresponding configuration used in [Sarathy et al., 2024](#). The initial state of each configuration can be found in Figure 5.2. Each configuration presents the agent with the same task: reach the green object.

- Level 1 is the simplest of the environment configurations, with just a ring of FalseWall objects and an easily accessible Goal object. It can be solved by moving forward in a straight line. The discovery associated with this level is the agent’s ability to move through the FalseWall.
- Level 3 presents three new objects: the Crate, the Button, and the ButtonDoor. It can be solved in two ways: opening the ButtonDoor and moving through the hole in the FalseWall ring, or moving through the FalseWall to reach the Goal. The discovery associated with this level is that pushing a Crate onto a Button opens the ButtonDoor.
- Level 5 introduces the Key and KeyDoor. Like the previous level, it can be solved either by leveraging the new objects, or by moving through the FalseWall ring to reach the Goal. The discovery associated with this level is the agent’s ability to pick up a Key and use it to open a KeyDoor.

- Level 9 combines the objects featured in the previous two levels, requiring agents to either push Crates and pick up the Key, or exploit the FalseWall property to solve. It has two “intended” discoveries, either of which could be found for the Interaction metric to be satisfied. Those discoveries were the Crate-Button discovery as well as the Key-KeyDoor discovery.
- Level 11 is the first configuration that cannot be solved by exploiting the FalseWall discovery. Instead, the agent must open the ButtonDoor to reach the Goal. As with the previous level, it has two associated discoveries: the FalseWall discovery and the Crate-Button discovery.
- Level 12, like the previous level, prevents the agent from exploiting the FalseWall discovery to succeed. In this level, the agent must make use of KeyCrates, which are the most complex dynamics in the environment. Its associated discoveries were the KeyCrate creation discovery as well as the Key-duplication discovery that is made using KeyCrates.

Note that for some levels, the intended discovery is not needed for the level to be solved. This happens when the agent exploits the FalseWall property on levels where doing so allows the agent to bypass the intended discovery. In instances like these, we can determine that the agent does not know how to use the other objects in the environment, and instead relies entirely on what it learned during training.

4.2 Results

The results of the reinforcement learning experiments show that while the RL agent excelled on several levels, there is a massive drop-off in completion rates on levels that where the Goal was not placed in the exact same spot as in Level 1. This result may indicate that the RL agent learned the movements needed for task completion rather than the actual properties of the environment itself. Considering the training regimen used for the RL agent, this behavior makes sense—having never learned the importance of objects other than FalseWalls and Goals, the agent could get away with simply learning to move towards the coordinate (19, 8).

More interesting findings appear when we examine the performance of the RL agent along the Interaction metric. For the most part, the agent does not make the discovery “needed” for each level. There’s an exception in the case of Level 5 however. In this level, the agent can simply walk forward 14 times and pass the level, just as in Level 1. As such, the agent could complete the whole level by executing the same movement

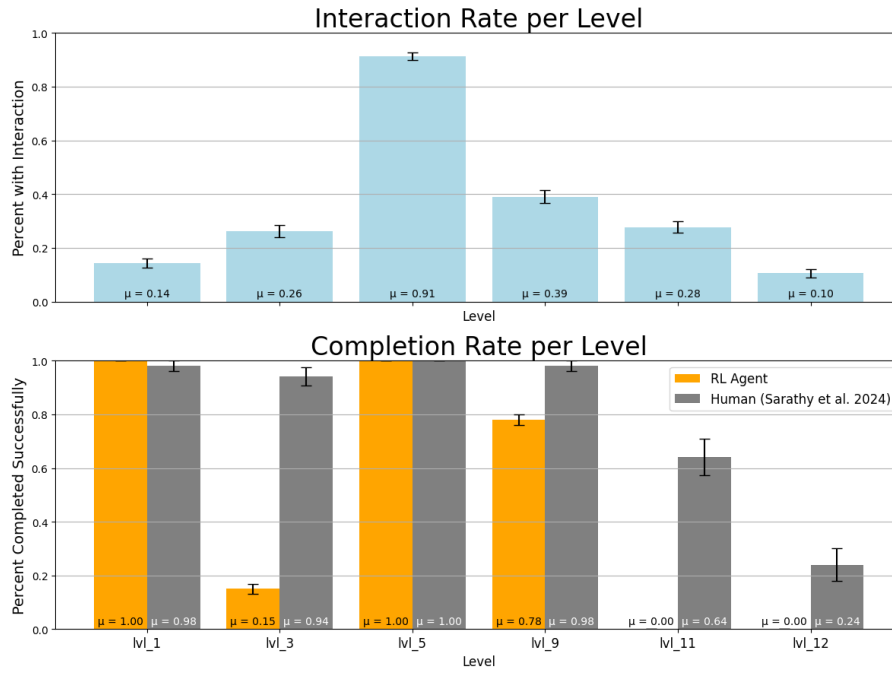


Figure 4.1: Comparison between PPO model and human participants.

sequence it learned in Level 1. This result indicates that agents are capable of exploiting discoveries even if they don't have a way of understanding those discoveries.

In addition to the definitive metrics used to evaluate the RL agent in this experiment, I assembled a GIF of all several runs of the experiment by layering all the runs of a single level on top of one another. The GIFs can be found in the public GitHub repository associated with this project. The GIFs reveal why the agents may have done so well in Level 5 compared to some of the other levels. Essentially, when the agent picks up the Key in Level 5, the configuration of the environment becomes very similar to that of Level 1. As soon as the KeyDoor is unlocked, the configuration becomes identical. Therefore, there aren't many chances for the new objects to cause confusion in the RL agent's action policy.

Level 3, on the other hand, has myriad ways of causing confusion. To start, there is one more new object in Level 3 compared to Level 5. Additionally, when the agent is pushing the Crate, the Crate does not disappear, meaning that the environment changes in unexpected ways every time the agent takes a step while pushing the Crate. Put together, these factors appear to confuse the agent greatly, as the overlays show that its actions become far less predictable as soon as the agent touches the Crate.

4.2.1 Implications

The results of the RL experiment, especially when compared to human performance from identical levels in [Sarathy et al., 2024](#), reveal several key takeaways. The first is that matching human performance in ESCGRIDENV requires a human-like ability to understand the environment. The levels that are sufficiently different from Level 1 were all basically impossible for the RL agent to complete, indicating the need for an internal representation of the environment’s transition dynamics that RL agents don’t possess on their own.

Another important takeaway is the fact that discoveries can be made without understanding their utility. This is shown in the Level 5 results, where the RL agent scores high on interaction despite not knowing about the Key-KeyDoor relationship.

Put together, these results show that with the right configuration, ESCGRIDENV can easily be used to test agents for qualities needed for IPS, including the ability to form an internal representation of the environment.

5 Large Language Model Experiments

To determine the effect of NLWM prompting on LLM environment utilization, I designed an experiment to test the following hypotheses:

1. **H1:** LLMs are more likely to make use of relevant objects within the environment when provided with the transition dynamics of the environment.
2. **H2:** LLMs are more likely to complete configurations of ESCGRIDENV when provided with the transition dynamics of the environment.

5.1 Experiment Setup

In Section 4.1, I formalized the environment in the context of the RL experiment. I will now expand upon that formalization to describe the LLM experiment.

The LLM agent possesses a Perception module $P : \Omega \rightarrow S$, where S represents the encoded state space of the environment. This is how the agent represents the environment for reasoning purposes. I explain later the precise forms that states $s \in S$ can take.

For every run of the LLM experiment, the agent generated a prompt $p = \tau(s)$, where $\tau : S \rightarrow p$ is a template that takes in state information and (optionally) information about transition dynamics in the environment. The prompt is then fed to LLM Plan-Actor $\Pi : p \rightarrow \eta$, where η is a string output containing both a high-level plan and a low-level action sequence based on the generated plan.

After the action sequence is generated, each of its actions are executed in sequence by transition function T , transitioning the environment from its initial state to its final state. This architecture was inspired by the Planner-Actor-Reporter system presented in [Dasgupta et al., 2023b](#), though it differs in a few ways. First, I condense the Planner and Actor modules into a single Plan-Actor. Because of how simple the action space

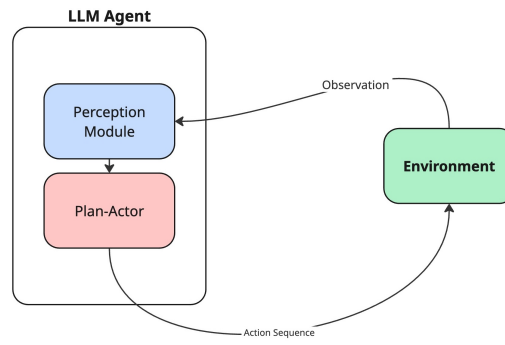


Figure 5.1: Illustration of all the components of the experiment.

is, there is no need for a separate module to handle motion control. Additionally, my system does not incorporate continuous environment feedback as a form of grounding. Instead, each run tasks the LLM with completed the puzzle in a single prompt-action cycle.

Each combination of independent variables (described below) was evaluated according to 5 separate runs. All runs were allowed as many steps as the LLM provided, and were performed on an NVIDIA RTX A6000 GPU.

5.2 Independent Variables

In addition to the use of NLWMs, I evaluated the effect of four other independent variables on LLM performance in this experiment. Together, the five independent variables of my experiment were:

- **NLWMs:** With and without a natural language world model.
- **Puzzle Difficulty:** Six different configurations of the environment that each require different discoveries to solve.
- **Vision Encoding Method:** 2D Matrix, Text
- **Prompting Techniques:** Base, CoT (Wei et al., 2023)
- **Model:** Llama3:70b (Grattafiori et al., 2024), Qwen2.5:14b (Qwen et al., 2025), Deepseek-R1:8b (DeepSeek-AI et al., 2025)

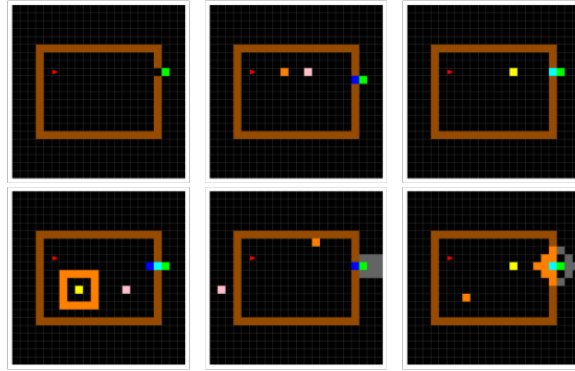


Figure 5.2: ESCGRIDENV configurations used in my experiments. Starting from the top left going clockwise, they are Levels 1, 3, 5, 12, 11, 9.

5.2.1 Natural Language World Models

In the context of this study, I define a natural language world model (NLWM) to be a list of statements about the environment describing the transition dynamics associated with each object found in the current configuration. The information contained in an NLWM should always be sufficient to solve the current configuration of the environment. Below is an example NLWM for Level 3:

- The green object is the Goal that the agent must reach.
- Brown objects are FalseWalls. The agent can move through FalseWalls.
- The orange object is a Crate. The agent can push the Crate by moving towards it while adjacent to the Crate.
- The pink objects is a Button. When the Crate is pushed onto the Button, the ButtonDoor opens.
- The blue object is a ButtonDoor. When the Crate is pushed onto the Button, the ButtonDoor opens. The agent cannot move through the ButtonDoor while it is closed.

5.2.2 Puzzle Difficulty

The difficulty of puzzles was modulated using six different configurations of the ESCGRIDENV environment, each described in Section 4.1.1. I chose the same configurations so I could draw comparisons between human, RL, and LLM results.

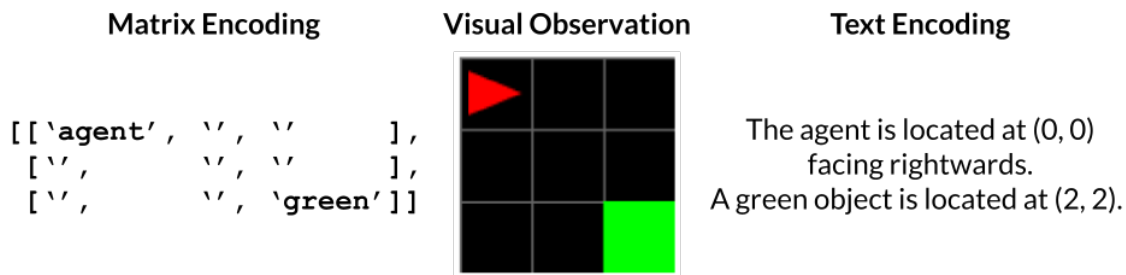


Figure 5.3: Comparison between visual representation and the two encoding methods.

5.2.3 Observation Encodings

One of the primary challenges of embodied reasoning for LLMs is that the observation space must be converted from the native representation to a text representation for LLMs to understand. I compare two different methods of observation encoding and how well LLMs are able to interpret embodied environments.

The first method encodes ESCGRIDENV observations as 2D Python matrices, with each cell in the matrix corresponding to a cell in the ESCGRIDENV environment. Objects in the environment are represented by their color only, thereby obscuring any information that could reveal how objects can be used without directly interacting with the objects themselves. This method preserves the spacial structure of the environment while translating the environment into a form that the LLM may be more familiar with. Because these LLMs are pretrained on large quantities of Python code, I predicted that they would have an easier time understanding these observations compared to visual observations of the environment.

The second method encodes ESCGRIDENV observations into high-level natural language descriptions. By sacrificing detail, this representation is able to provide the LLMs with observations that are more understandable than data structure representations. This method lists the coordinate positions of key objects in the environment while abstracting away less pertinent details. For example, the position of each FalseWall object is condensed into a single sentence describing a barrier of brown objects around the rest of the objects in the environment. As with the matrix encoding method, objects were referred to by their color only, requiring their utilities to be discovered through interaction alone.

Figure 5.3 provides a visual comparison between the two encoding methods and how they translate environment observations into text.

5.2.4 Prompting Methods

I compare the performance of two different prompting strategies: a straightforward zero-shot prompting method and a single-shot chain of thought prompting method. The former provides the LLM with just an observation of the environment and a simple instruction prompt regarding the task and action space of the environment.

With the chain of thought (CoT) prompting method, I follow the methodology first described in [Wei et al., 2023](#), wherein LLMs were provided with a step-by-step algorithm for solving the task at hand via single-shot prompting in addition to the initial instructions. The example provided in my CoT prompts uses Level 2 from ESCAPE ([Sarathy et al., 2024](#)). I chose this example because it was similar enough to Level 1 to provide just as little information about the rest of ESCGRIDENV’s transition dynamics, while still being different enough that it didn’t provide the exact solution to Level 1 itself.

Further details regarding prompting can be found in Appendix A.

5.3 Metrics - Operationalizing the Hypotheses

As with the RL experiments, each run of the LLM was evaluated according to two binary metrics:

H1: Interaction. Each level has an associated discovery (or discoveries) that are considered necessary for the agent to find before it can complete the puzzle. For each run, I determined whether or not the agent performed an action that would reveal the discovery for that level (e.g. walking through a FalseWall, opening a KeyDoor, etc.).

The purpose of this metric is to show whether or not LLMs are capable of making use of the knowledge they’ve been given via natural language. It is meant to be an indicator that semantic knowledge translates to implicit understanding.

H2: Completion. As a more concrete metric than interaction, I use level completion as a measure of agent performance. Completion is achieved when the agent reaches the Goal object.

Completion shows more definitive proof of grounding. Because the transition dynamics given in every run are sufficient to solve the problem with, an LLM capable of translating semantic knowledge into implicit understanding should be able to score very high on Completion.

6 Large Language Model Results

In this section, I analyze the effect of each independent variable on task performance across the two metrics defined in the previous section. Additionally, I perform a qualitative analysis on three example runs to show how LLMs reasoning about tasks in the ESCGRIDENV environment.

6.1 Effect of Natural Language World Models

Figure 6.1 shows the effect that NLWMs have on agent performance in ESCGRIDENV. The results indicate that NLWMs increase Interaction ($p = 0.0015$), but have no effect on Completion rate. This indicates that while NLWMs grant LLMs some degree of implicit understanding, the grounding they provide is not sufficient to significantly improve performance, revealing a disconnect between semantic knowledge and implicit grounding in ESCGRIDENV.

6.2 Effect of Puzzle Difficulty

Performance across both metrics varied between each puzzle configuration, suggesting Interaction and Completion rate are dependent on factors more specific than mere difficulty. That being said, there are a few key observations that can be made in regards to the affect of level configuration on agent performance.

Interaction appears to be most correlated with how many steps it takes for the agent to reach the nearest object. That is, in configurations where objects were placed relatively close to the agent, it was far more likely for them to make the intended discovery for that level. Meanwhile, Completion only rises above 2% in the levels where the most straightforward solution was to simply move forward until the agent reached the goal, suggesting that LLMs may have trouble generating more complex plans.

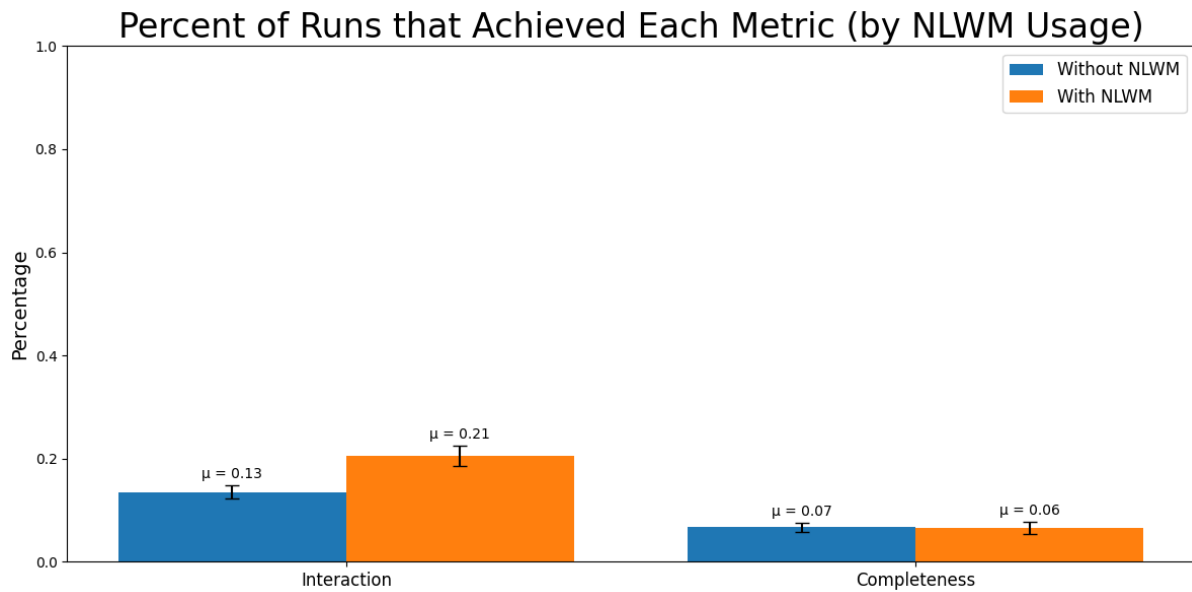


Figure 6.1: Effect of NLWM prompting by metric.

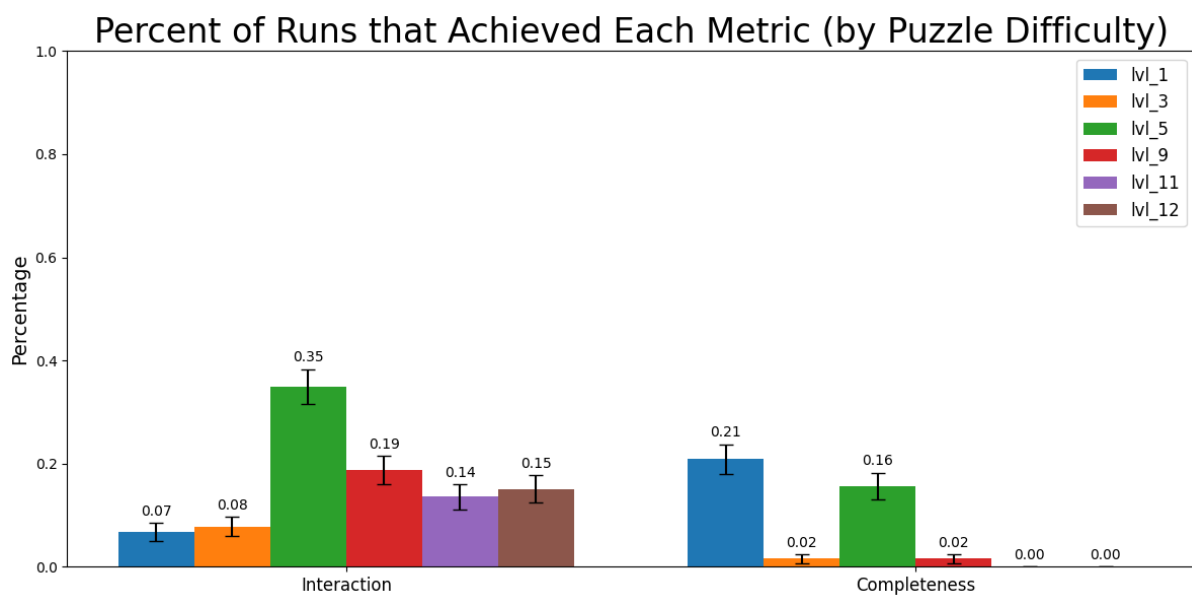


Figure 6.2: Effect of puzzle difficulty by metric.

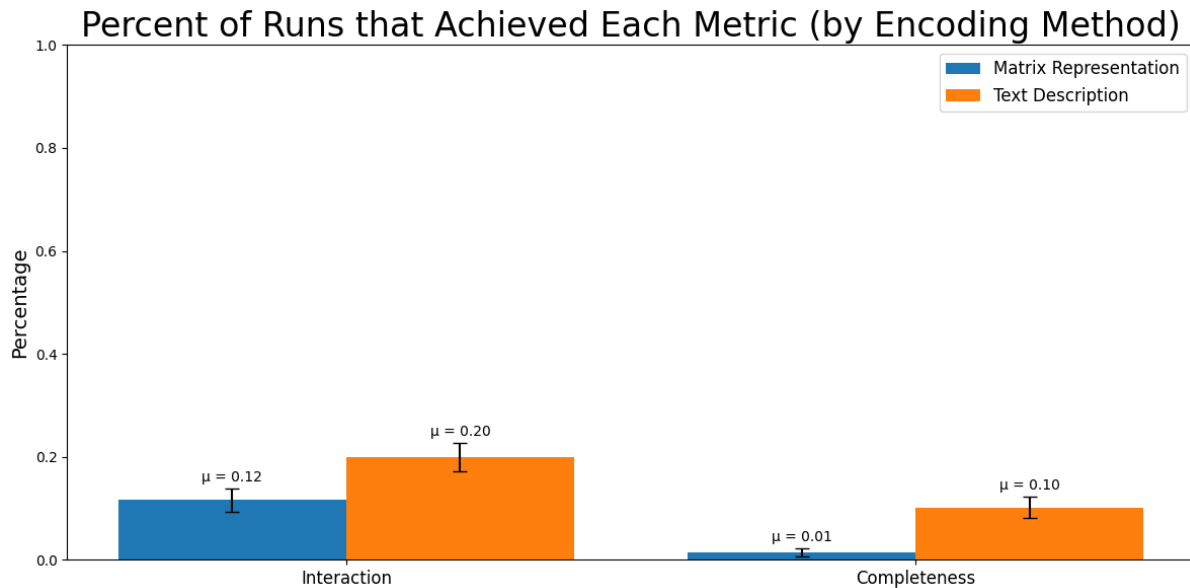


Figure 6.3: Effect of encoding method by metric.

Comparing the LLM results to the RL results for the same levels (Figure 4.1), I observe significantly higher performance from the RL agent on the levels that only require the FalseWall discovery to solve. This raises the question of how useful an internal representation of the environment is. In Section 7, I answer that question by comparing the performance of an RL agent against that of an LLM agent on levels where moving in a straight line is not sufficient to solve the task.

6.3 Vision Encoding Method

Natural language descriptions outperform matrix encodings of environment observations, with statistically significant increases in Interaction ($p = 0.0174$) and Completeness ($p < 0.0001$). This improvement is likely due to how the LLMs used for this experiment (all off-the-shelf) were fine-tuned. Because they were all primarily fine-tuned on natural language data, these models were able to understand the natural language descriptions more fluently than the matrix encodings, which would likely require specialized fine-tuning to understand fluently.

Fluency in natural language may not be the only reason why the descriptions improved overall model competence. Because the natural language descriptions included coordinates for objects in the environment, it's possible that the models were able to use coordinate arithmetic to solve the puzzles more easily. However, proving such a statement would require a more in-depth analysis of the LLM reasoning traces, as shown in Section 6.6.

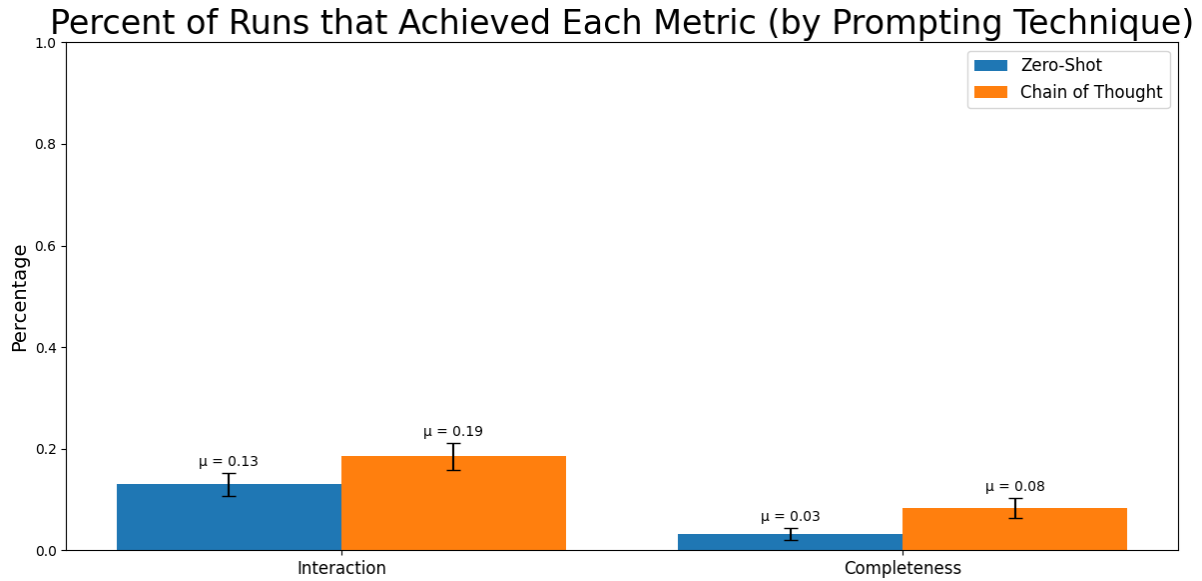


Figure 6.4: Effect of prompting technique by metric.

6.4 Prompting Technique

By guiding models to follow a step-by-step problem-solving routine, the Chain of Thought (CoT) prompting technique successfully increased the likelihood that models completed configurations of ESCGRIDENV ($p = 0.0234$). However, CoT did not significantly outperform zero-shot prompting in the Interaction metric ($p = 0.1129$). These results indicate that CoT encouraged agents to behave similarly to the RL agent, in the sense that the LLMs learned successful movements in-context rather than more implicit reasoning and problem-solving abilities.

These results may indicate that further exploration of prompting techniques could greatly improve the overall competency of embodied LLM agents. For example, my single-shot CoT prompting could easily be improved by providing more examples. Additionally, there exist more rigorous prompting strategies that could outperform CoT in the metrics defined in this study.

6.5 Model

Between the 3 models evaluated for this study, Llama3 and Qwen2.5 significantly outperform DeepSeek-R1 in both metrics (see p -values in Table 6.1). This difference may be due to the differences in size between the evaluated models, though there may be other factors at play, since neither the 70-billion parameter Llama model nor 14-billion parameter Qwen model significantly outperform each other in either metric.

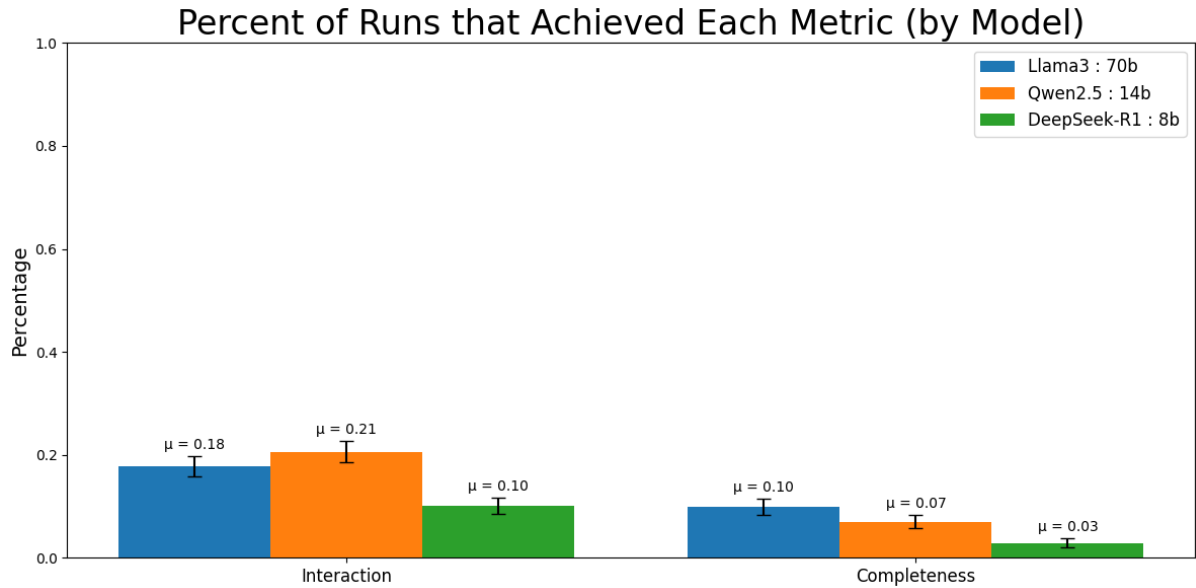


Figure 6.5: Effect of model choice by metric.

	Llama	Qwen	DeepSeek
Llama	–	0.3130 0.1538	0.0025 < 0.0001
Qwen	0.3130 0.1538	–	< 0.0001 0.0078
DeepSeek	0.0025 < 0.0001	< 0.0001 0.0078	–

Table 6.1: Pairwise p -values for performance difference between Llama3, Qwen2.5, and DeepSeek-R1. The first value is for Interaction and the second is for Completeness.

Factors besides size that may affect model performance may include the specific data used to pretrain and fine-tune each model or DeepSeek’s built-in reasoning capabilities causing distractions rather than aiding with planning.

6.6 Qualitative Analysis

To supplement the automated metrics used for this study, I examine three example runs to compare LLM reasoning traces to their coresponding action sequences. The purpose of this analysis is to show the disconnect between semantic understanding of the environment (represented as the high-level plan) and the LLMs’ implicit understanding of ESCGRIDENV (represented as the low-level action sequence).

Table 6.2 shows the independent variable settings and reasoning trace for each example. Visualization of the corresponding action sequences can be found in the GitHub repository associated with this project.

Ex.	Settings	Reasoning Trace
1	No NLWM, Zero-Shot, Llama, Text, Lvl 12	<p>Move to the goal object by following these high-level steps:</p> <ol style="list-style-type: none"> 1. Move RIGHT until reaching the edge of the brown border at (17, 8). 2. Move DOWN until reaching the opening in the enclosure at (18, 7). 3. Move RIGHT until reaching the cell containing the green object at (19, 9).
2	With NLWM, Zero-Shot, Qwen, Text, Lvl 5	<p>The high-level plan for the agent to achieve its goal involves navigating through an environment with various objects that affect the state of the agent in different ways. Here's a step-by-step high-level plan:</p> <ol style="list-style-type: none"> 1. The first action would be to move from (5,8) to (6,8), then continue moving right until reaching the opening at (18, 8). To avoid purple status before reaching cyan square, it is essential not to interact with yellow squares until necessary. Therefore, stay on path while avoiding yellow object and head straight for goal. 2. Once the agent reaches the opening at position (18,8), there is a cyan object in that cell. It should be noted that stepping on this cyan square will only disappear when the player is purple. To properly use the cyan square, the agent must first become purple by moving onto the yellow object located at (13, 8). 3. After turning purple, continue moving right to get to the cyan object and cause it to disappear once stepped on. 4. Then move one space further in that direction to reach the goal object which is green and located at position (19, 8).
3	No NLWM, Zero-Shot, DeepSeek, Matrix, Lvl 1	Go RIGHT, DOWN, LEFT, UP, RIGHT, DOWN, LEFT, RIGHT, DOWN, LEFT, RIGHT, DOWN, LEFT, RIGHT, DOWN, LEFT

Table 6.2: Example reasoning traces for qualitative analysis.

6.6.1 Example 1

The reasoning trace for this example shows a clear misunderstanding of how to navigate a coordinate grid. For example, the LLM plans to move down in step 2, which should increment the y-coordinate (like MiniGrid, ESCGRIDENV uses the top-left corner as (0, 0)). However, instead of incrementing the y-coordinate, the LLM increments the x-coordinate and decrements the y-coordinate. This confusion shows that LLMs have difficulty making sense of objective coordinate positions.

The agent's movements within the environment are fairly faithful to the generated plan, showing that the LLM's words are reflected (if only to a limited degree) in its implicit understanding of the environment.

6.6.2 Example 2

This example shows how an LLM makes use of its enhanced semantic knowledge of the environment. The resulting reasoning trace shows that even with the extra knowledge, the model's internal representation of the environment remains weak. Qwen makes the unfounded assumption that it must wait to pick up the Key until it reaches the KeyDoor. In doing so, its plan requires the agent to double back and retrieve the Key after reaching the door. The hallucinated assumption is likely an artifact of the model's pretraining or fine-tuning, as there was no indication in the original prompt that such an assumption needed to be made.

The action sequence associated with this plan is not entirely faithful. By moving in a straight line towards (18, 8), the agent picks up the Key earlier than expected. When it tries to double back, the agent does not retreat far enough (not that doing so was necessary anymore). It then fails to return to its previous position, so its final movement to the right does not take the agent to the Goal.

6.6.3 Example 3

Example three demonstrates a failure of reasoning. Despite the simplicity of the configuration, the agent fails to identify a goal or provide any justification for its actions. It instead provides a nonsensical action sequence that carries the agent in a circle. This action sequence is accurately executed by the agent, indicating a possible tradeoff between the complexity of plans and the accuracy with which the agent is able to execute those plans.

7 Reinforcement Learning vs. LLMs

To better understand the role of internal world models in agent performance on embodied tasks, I conducted a targeted comparison between a reinforcement learning (RL) agent and a large language model (LLM) in the ESCGRIDENV environment. This experiment was performed on variants of Level 5 from the previous experiments. While the RL agent demonstrated high success in the base configuration of Level 5, its behavior on the later variants suggest a tendency to memorize action sequences rather than develop an understanding of the environment’s dynamics. This experiment tests whether LLMs, despite their lack of embodied experience, can leverage their parametric memory to form implicit representations of task structure that support better generalization.

7.1 Experiment Setup

To determine if an internal representation of the environment improves generalization, I created four variants of Level 5, each modifying the Goal object’s position slightly while leaving the rest of the map unchanged. This manipulation was designed to penalize strategies based on fixed action sequences (e.g., RL memorization) while rewarding strategies that relied on a more nuanced understanding of the environment.

For the RL side of the experiment, I reused the policy trained for the original RL experiment and ran it on each variant 100 times. For the LLM, I used the highest-performing configuration from the previous chapter—Llama with NLWM, text-based observation encoding, and CoT prompting—and evaluated it 20 times per variant. Figure 7.1 shows the level layouts used in this experiment.

7.2 Results

The RL agent performed well on the initial Level 5 configuration, reliably unlocking the KeyDoor and progressing toward the Goal. However, its success rate quickly dropped

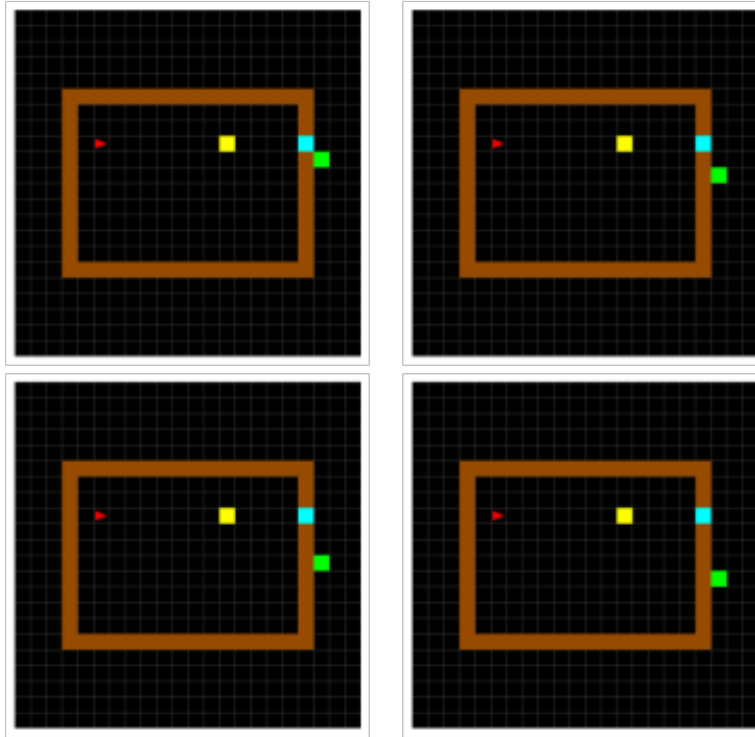


Figure 7.1: Level 5 Variants.

as the Goal object was moved even a single tile from its original position. This suggests that the agent relied on a memorized sequence of actions rather than adapting its behavior based on the Goal’s position. This claim is supported by the RL agent’s high Interaction score; the RL agent did not demonstrate meaningful spatial generalization and instead moved straight towards the tile where it expected success, rather than the object that would reward success.

In contrast, the LLM exhibited modest performance on the original Level 5 variant but maintained a relatively stable success rate across all goal-shifted variants. This stability indicates that the LLM was not rigidly following a fixed path but instead guided its behavior by interpreting the utility of objects in the environment, especially the Goal object.

Interestingly, the LLM performed worse than the RL agent on the Interaction metric (i.e., unlocking the KeyDoor), suggesting that it either did not fully grasp the purpose of the Key-KeyDoor interaction or chose to ignore it. Further inspection revealed that the LLM frequently exploited the FalseWall mechanic instead, suggesting a possible preference for simpler, less cognitively demanding solutions, even when a more emphasized “correct” solution (via Key-KeyDoor) was explicitly provided by the NLWM. This preference highlights both the flexibility and the limitations of LLM planning: while the LLM

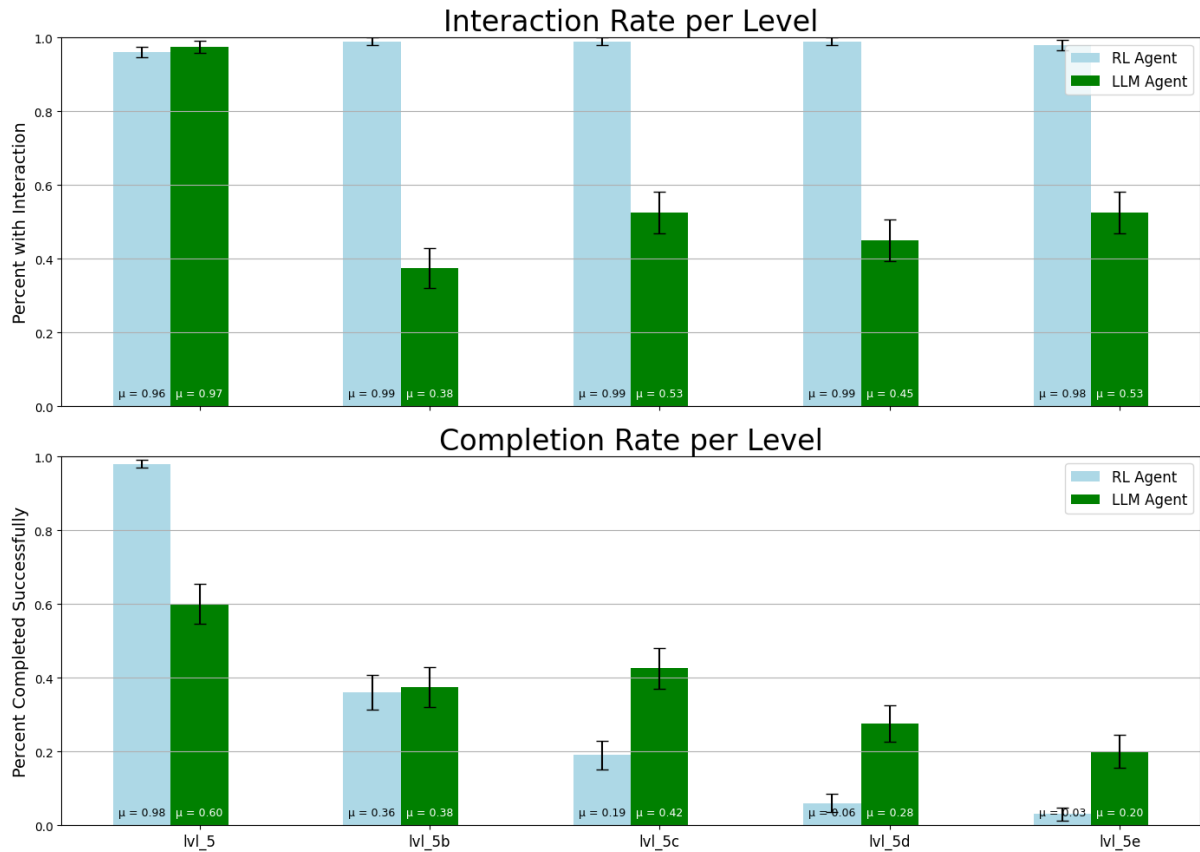


Figure 7.2: Level 5 Variant Results.

was able to adapt its strategy, it may also overlook multi-step dependencies in favor of more direct paths.

7.2.1 Implications

These results suggest that LLMs are capable of forming limited but meaningful internal representations of their environment, sufficient to guide adaptive planning in novel configurations. Unlike the RL agent, which collapses when its memorized action sequence is unsuccessful, the LLM dynamically integrates world knowledge and spatial information to reorient itself toward the Goal. This behavior mirrors the kind of flexible insight-driven problem solving found in humans, albeit in a primitive form.

Importantly, the LLM’s success despite minimal training data (only Level 1 used for RL training and Level 2 given as an in-context example) reinforces the idea that LLMs are well-suited for low-data, high-abstraction tasks. Their ability to form an internal representation of the environment allows them to generalize over structured domains even in the absence of embodied trial-and-error.

However, the LLM’s failure to reliably exploit the Key-KeyDoor interaction, despite

being told about it by the NLWM, reveals a limitation in their ability to perform complex reasoning. LLMs may struggle to internalize multi-step relational transition effects when those dependencies must be translated into concrete action sequences. This opens an important avenue for future work: augmenting LLMs with mechanisms to better integrate and reason over causal object relations, perhaps through structured memory modules or hybrid symbolic interfaces.

8 General Discussion

This study set out to explore the efficacy of natural language world model (NLWM) prompting as a means of grounding large language models (LLMs) in an embodied environment. The immediate takeaway is that while NLWMs do induce some degree of implicit environmental understanding in LLMs, this understanding is shallow and often insufficient for improving overall task competence. That is, LLMs can sometimes make use of knowledge provided through NLWMs, but this does not reliably translate into successful task execution. These results reinforce a crucial insight: possession of a semantic understanding of an environment is not equivalent to being grounded in it.

8.1 Implications Within This Study

The results offer strong evidence that semantic descriptions of an environment, such as those provided by NLWMs, can improve interaction-level behavior. LLMs exposed to NLWMs were more likely to utilize relevant objects in the environment (e.g., pushing crates onto buttons or walking through FalseWalls), suggesting a partial alignment between their linguistic understanding and the environment’s transition dynamics. However, this partial alignment did not result in a statistically significant increase in task completion, highlighting a disconnect between semantic knowledge and actionable understanding.

This limitation becomes particularly evident in tasks requiring multi-step planning or indirect object manipulation. LLMs often failed to chain together discoveries into coherent, goal-directed behaviors, even when they could articulate those discoveries in natural language. For instance, in Level 5, LLMs were explicitly told how to use the Key and KeyDoor to solve the task, but failed to execute the necessary action sequences to do so. Conversely, RL agents, despite being unable to explain the environment in any interpretable sense, could complete some tasks simply by repeating memorized action sequences. This contrast suggests that the capacity to describe the environment and the ability to act effectively within it are fundamentally distinct forms of knowledge.

In short, the use of NLWMs does enable LLMs to expand their conceptual representation of an environment, but it does not guarantee that this expansion will be leveraged for improved task competence. The bottleneck lies not in knowledge acquisition but in knowledge application.

8.2 Implications Beyond This Study

These findings resonate with broader concerns in the fields of AI alignment and grounded reasoning. Specifically, they challenge the assumption that semantic grounding can substitute for embodied understanding. An agent that “knows” what a button does, or can articulate how a key interacts with a door, may still fail to execute a plan involving those objects. This distinction mirrors a long-standing insight in cognitive science: possessing declarative knowledge about a process is not the same as possessing procedural fluency in enacting it [Anderson, 1982](#).

This discrepancy points toward a fundamental limitation of current LLMs. They are powerful pattern matchers trained on language, not agents with integrated sensory or motor experience. Thus, while they can learn from static descriptions, they struggle to apply what they learn to embodied environments. One promising direction for future work may involve giving LLMs the opportunity to self-discover environment dynamics through exploration, rather than merely being told about them. This would more closely mirror human learning, where knowledge is constructed through interaction, not just instruction.

Moreover, the finding that LLMs frequently avoided using the very transition dynamics described in NLWMs (opting instead for simpler, previously successful strategies) suggests that even when LLMs do possess the requisite information, they may not judge its utility correctly. This behavior implies a need for mechanisms that help models prioritize and sequence discovered knowledge appropriately. Integrating memory, causal inference, and structured reasoning modules may help close the gap between semantic representation and actionable understanding. These external tools go beyond the scope of mere prompting, reinforcing the idea that semantic understanding is insufficient for grounding LLMs.

Finally, the limitations of NLWM prompting suggest caution in applications where LLMs must operate in novel environments with limited real-time feedback. In robotics, autonomous navigation, or simulation-based education, reliance on language-based grounding alone may not be sufficient to ensure effective behavior. Grounding in these domains likely requires richer feedback loops, either through multi-modal sensors or

through mechanisms that allow agents to revise their internal models in response to failed actions.

8.3 Reframing Grounding as a Multi-Stage Process

Perhaps the most significant implication of this work is the reframing of grounding not as a binary property—grounded vs. ungrounded, but as a spectrum composed of multiple stages: semantic acquisition, implicit internalization, and procedural competence. NLWMs appear to support the first stage and partially enable the second, but fail to bridge the final gap to competence. Bridging that gap will require new mechanisms that connect high-level semantic understanding to embodied reasoning and planning.

This reframing also encourages more nuanced metrics for evaluating LLMs in embodied tasks. Success should not merely be judged by whether an agent can “name the parts” of an environment or describe its dynamics, but by its ability to use that understanding to reason, plan, and adapt behavior over time. The partial success of LLMs in the Interaction metric suggests that current models are on the verge of more meaningful grounding, but still lack the architectural components needed to cross that threshold.

8.4 Summary

This study contributes to the growing literature on LLMs in embodied domains by demonstrating both the promise and limitations of lightweight grounding via natural language. NLWMs can prime LLMs to recognize surface affordances and form rudimentary internal models of an environment, but these internal models often lack the robustness and adaptability needed to drive reliable planning or problem-solving. The findings call for a broader perspective on grounding, one that embraces embodiment, exploration, and reasoning as integral components of competent agency.

9 Future Work

There remains ample space for future work following this study. Our results show that while NLWMs grant LLMs’ some form of internal representation, they does not lead to an improvement in overall competency. A logical next step would be to take further lightweight grounding measures to determine how much is necessary to significantly improve model performance at this task.

Such methods could include providing LLMs with environment feedback, thereby allowing them to learn the consequences of their actions in-context. This form of grounding is common in previous work ([Dasgupta et al., 2023b](#); [Huang et al., 2022](#)), and my results show that is may still be necessary for developing LLM agents capable of embodied insight problem solving.

Another direction to explore focuses on self-discovery. My study shows that gold-standard NLWM prompting alone does not improve the overall competence of LLM agents. It’s possible that agents can learn the utility of discovers more effectively if they find them on their own. To that end, developing a system by which LLM agents can explore the environment on their own and utilize their findings may enhance their ability to solve problems like the ones presented in this study. Such a study would require methods to not only guide LLMs towards exploration, but also to verify the correctness of new theorized discoveries and later exploit verified discoveries. Potential approaches to these problems may include self-verification (though [Stechly et al., 2024](#) questions the efficacy of such methods) or the use of symbolic knowledge representations to supplement parametric memory and reasoning.

10 Conclusion

In this paper, I set out to answer a key question at the interaction of LLM reasoning and embodied problem solving: Can large language models be grounded in physical environments using natural language alone? Specifically, I explored whether natural language world models (NLWMS), concise textual descriptions of environment transition dynamics, could induce an internal representation of the environment in LLMs. This work is critical for the development of insight problem solving agents for embodied tasks.

To answer the central question of this paper, I presented ESCGRIDENV, a novel 2D OpenAI Gym environment designed for embodied insight problem solving. Specifically, the environment tests the abilities of AI agents to form and exploit internal representations of the environment and its transition dynamics. With this environment, I evaluated the performance of reinforcement learning and LLM agents across a variety of embodied puzzles.

The results of my experiments show that NLWM prompting reliably increased LLMs’ likelihood of interacting with relevant objects, showing that language-based knowledge can help structure agent behavior to embodied environments. However, the benefits of NLWM prompting did not translate into higher task completion rates. LLMs often failed to combine discoveries into coherent action plans, and tended to favor simpler solutions over new solutions that required more complex reasoning patterns. This finding reveals a disconnect between semantic understanding and embodied competence; that is, the difference between knowing how the environment works and knowing how to act within it.

The limitations observed extend beyond this particular environment. They speak to the broader challenge of grounding AI agents not just semantically, but functionally. The ability to describe the world is not the same as being situated within it. True grounding likely requires mechanisms for exploration, feedback, memory, and abstraction—components that go beyond prompting alone. Even so, the potential of NLWMS as lightweight grounding tools remains compelling. They offer a scalable, in-

interpretable way to seed knowledge into LLMs, laying a foundation upon which more robust grounded reasoning systems can be built.

In bridging natural language and embodied experience, this study contributes to our understanding of what it means for LLMs to understand their surroundings. It calls for a reconception of grounding as a layered, interactive process, and points toward a future where agents can not only read the rules of the world, but play by them with insight and intention.

11 Limitations

While this study has revealed several key insights, there remain several limitations to consider. First, all evaluation metrics were automated and binary in nature, which may not fully capture the nuances of LLM grounding. Although I analyzed a few of the LLM reasoning traces, more extensive human judgments are needed to gain a comprehensive picture of plan coherence, creativity, alignment with task goals, among other potential manual evaluation metrics. Incorporating manual evaluations would provide a richer and potentially more accurate understanding of how well LLMs understand their surrounds.

Secondly, this study did not include recent state-of-the-art language models (e.g., GPT-4o, Llama-4) or vision-language models that could perceive raw observations directly. As model capabilities continue to advance, it remains unclear whether the observed effects of NLWM prompting would scale to much larger or multi-modal architectures.

Third, while the abstractness of ESCGRIDENV allowed me to isolate agents' ability to reason about entirely unfamiliar environments and lessen the impact of functional fixedness, the real-world is rich with multi-modal stimuli that can often be effectively exploited using pretrained biases. As a result, my findings may not generalize to tasks involving more helpful visual or natural language feedback.

Acknowledgments

I am beyond thankful for the mentorship, guidance, and support of my thesis advisors, Vasanth Sarathy and Matthias Scheutz. This project wouldn't have been possible without them. They have pushed me to not only achieve my highest potential as a researcher, but also to think like an scientist. I've grown both as a person and an academic through working with them, and I look forward to continuing our collaborations next year during my MS-AI at Tufts.

I'd also like to thank the other talented researchers at Tufts who I consulted for this project, including Jivko Sinapov, Yash Shukla, and Panagiotis Lymperopoulos. Their input was invaluable in helping me identify my thesis topic.

Beyond this project, I would like to thank all my previous (and current) research mentors from the National Library of Medicine and Tufts, including Brian Ondov, Dina Demner-Fushman, Matt Russell, and Rob Jacob. They guided me during my first years of research, and are the ones who inspired me to pursue my senior thesis.

Last but not least, I want to thank the innumerable friends and family who have supported me during this project. Whether it was listening to me rant about my research, taking the time to proofread my manuscript, or just being there to take my mind off this project every now and then, I can't thank them enough. To all of you, *you* are the ones who keep me grounded.

References

- Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., . . . Zeng, A. (2022). Do as i can, not as i say: Grounding language in robotic affordances. <https://arxiv.org/abs/2204.01691>
- Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review*, 89(4), 369–406. <https://doi.org/10.1037/0033-295x.89.4.369>
- Boden, M. A. (1998). Creativity and artificial intelligence [Artificial Intelligence 40 years later]. *Artificial Intelligence*, 103(1), 347–356. [https://doi.org/https://doi.org/10.1016/S0004-3702\(98\)00055-1](https://doi.org/https://doi.org/10.1016/S0004-3702(98)00055-1)
- Boden, M. A. (2010). *Creativity and art: Three roads to surprise*. Oxford University Press.
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. <https://arxiv.org/abs/1606.01540>
- Chemero, A. (2003). An outline of a theory of affordances. *Ecological Psychology*, 15(2), 181–195. https://doi.org/10.1207/S15326969ECO1502_5
- Cheng, G., Zhang, C., Cai, W., Zhao, L., Sun, C., & Bian, J. (2024). Empowering large language models on robotic manipulation with affordance prompting. <https://arxiv.org/abs/2404.11027>
- Chevalier-Boisvert, M., Dai, B., Towers, M., Perez-Vicente, R., Willems, L., Lahlou, S., Pal, S., Castro, P. S., & Terry, J. (2023). Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *Advances in Neural Information Processing Systems 36, New Orleans, LA, USA*.
- Chitikena, H., Sanfilippo, F., & Ma, S. (2023). Robotics in search and rescue (sar) operations: An ethical and design perspective framework for response phase. *Applied Sciences*, 13(3). <https://doi.org/10.3390/app13031800>
- Dainese, N., Merler, M., Alakuijala, M., & Marttinen, P. (2024). Generating code world models with large language models guided by monte carlo tree search. <https://arxiv.org/abs/2405.15383>

- Dasgupta, I., Kaeser-Chen, C., Marino, K., Ahuja, A., Babayan, S., Hill, F., & Fergus, R. (2023a). Collaborating with language models for embodied reasoning. <https://arxiv.org/abs/2302.00763>
- Dasgupta, I., Kaeser-Chen, C., Marino, K., Ahuja, A., Babayan, S., Hill, F., & Fergus, R. (2023b). Collaborating with language models for embodied reasoning. <https://arxiv.org/abs/2302.00763>
- DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., ... Zhang, Z. (2025). Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. <https://arxiv.org/abs/2501.12948>
- Duncker, K. (1945). On problem-solving. (L. S. Lees, Ed.). *Psychological Monographs*, 58(5), i–113. <https://doi.org/10.1037/h0093599>
- Giadikiaroglou, P., Lymperaious, M., Filandrianos, G., & Stamou, G. (2024, November). Puzzle solving using reasoning of large language models: A survey. In Y. Al-Onaizan, M. Bansal, & Y.-N. Chen (Eds.), *Proceedings of the 2024 conference on empirical methods in natural language processing* (pp. 11574–11591). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2024.emnlp-main.646>
- Gibson, J. J. (1979). *The ecological approach to visual perception: Classic edition*. Houghton Mifflin.
- Gizzi, E., Nair, L., Chernova, S., & Sinapov, J. (2022). Creative problem solving in artificially intelligent agents: A survey and framework. *Journal of Artificial Intelligence Research*, 75. <https://doi.org/10.1613/jair.1.13864>
- Goel, S., Shukla, Y., Sarathy, V., Scheutz, M., & Sinapov, J. (2022). Rapid-learn: A framework for learning to recover for handling novelties in open-world environments. <https://arxiv.org/abs/2206.12493>
- Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., ... Ma, Z. (2024). The llama 3 herd of models. <https://arxiv.org/abs/2407.21783>
- Ha, D., & Schmidhuber, J. (2018). World models. <https://doi.org/10.5281/ZENODO.1207631>
- Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., Sermanet, P., Brown, N., Jackson, T., Luu, L., Levine, S., Hausman, K., & Ichter, B. (2022). Inner monologue: Embodied reasoning through planning with language models. <https://arxiv.org/abs/2207.05608>
- Kambhampati, S. (2024). Can large language models reason and plan? *Annals of the New York Academy of Sciences*, 1534(1), 15–18. <https://doi.org/10.1111/nyas.15125>

- Nair, L., Gizzi, E., & Sinapov, J. (2024). Creative problem solving in large language and vision models – what would it take? <https://arxiv.org/abs/2405.01453>
- Ohlsson, S. (1992). Information-processing explanations of insight and related phenomena. *Advances in the psychology of thinking*, 1. https://www.academia.edu/20712367/Information_processing_explanations_of_insight_and_related_phenomena
- Plaat, A., Wong, A., Verberne, S., Broekens, J., van Stein, N., & Back, T. (2024). Reasoning with large language models, a survey. <https://arxiv.org/abs/2407.11511>
- Qwen, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., ... Qiu, Z. (2025). Qwen2.5 technical report. <https://arxiv.org/abs/2412.15115>
- Sarathy, V. (2018). Real world problem-solving. *Frontiers in human neuroscience*, 12. <https://hrilab.tufts.edu/publications/sarathy2018real.pdf>
- Sarathy, V., Rabb, N., Kasenberg, D., & Scheutz, M. (2024). Using puzzle video games to study cognitive processes in human insight and creative problem-solving. *Proceedings of the 46th Annual Meeting of the Cognitive Science Society*. <https://hrilab.tufts.edu/publications/sarathyetal2024cogsci.pdf>
- Sarathy, V., & Scheutz, M. (2018). Macgyver problems: Ai challenges for testing resourcefulness and creativity. *Advances in Cognitive Systems*, 6. <https://hrilab.tufts.edu/publications/sarathy2018MacGyverACS.pdf>
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. <https://arxiv.org/abs/1707.06347>
- Shukla, Y., Gao, W., Sarathy, V., Velasquez, A., Wright, R., & Sinapov, J. (2023). Lgts: Dynamic task sampling using llm-generated sub-goals for reinforcement learning agents. <https://arxiv.org/abs/2310.09454>
- Silver, T., & Chitnis, R. (2020). Pddl-gym: Gym environments from pddl problems. <https://arxiv.org/abs/2002.06432>
- Smirnov, P., Joublin, F., Ceravola, A., & Gienger, M. (2024). Generating consistent pddl domains with large language models. <https://arxiv.org/abs/2404.07751>
- Stechly, K., Valmeekam, K., & Kambhampati, S. (2024). On the self-verification limitations of large language models on reasoning and planning tasks. <https://arxiv.org/abs/2402.08115>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. <https://arxiv.org/abs/1706.03762>
- Verma, V., Maimone, M. W., Gaines, D. M., Francis, R., Estlin, T. A., Kuhn, S. R., Rabideau, G. R., Chien, S. A., McHenry, M. M., Graser, E. J., Rankin, A. L., & Thiel, E. R. (2023). Autonomous robotics is driving perseverance rover's progress on mars. *Science Robotics*, 8(80). <https://doi.org/10.1126/scirobotics.adi3099>

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2023). Chain-of-thought prompting elicits reasoning in large language models. <https://arxiv.org/abs/2201.11903>

A Prompting Strategies

I used four different prompts for the different combinations of independent variables used in the LLM experiment. I now provide each prompt below as well as the conditions that required their use.

A.1 Without NLWM, Zero-Shot Prompting

[Prompt] You are a planner for an agent in a 2D GridWorld environment. You are given an observation of the environment. You are also given a goal that the agent must achieve.

Your task is to generate a high-level plan that the agent can follow to achieve its goal. Then, generate a low-level action sequence that the agent can execute to achieve its goal.

[Observation] {obs}

[Goal] The agent must move to the same cell as the Goal object

[Plan]

A.2 Without NLWM, CoT Prompting

[Prompt] You are a planner for an agent in a 2D GridWorld environment. You are given an observation of the environment. You are also given a goal that the agent must achieve.

Your task is to generate a high-level plan that the agent can follow to achieve its goal. Then, generate a low-level action sequence that the agent can execute to achieve its goal. Below is an example observation, goal and plan. Use the example as a reference for your own problem solving steps.

[EXAMPLE]

[Observation] The agent is located at (5, 8). A green object is located at (19, 13). A brown border forms an enclosure around the agent as well as all objects except the green object, and the green object at (19, 13) There is an opening in the enclosure at (18, 13).

[Goal] The agent must move to the same cell as the Goal object.

[Plan] First determine that the green object is the Goal object. Align the agent on the horizontal axis ($y=13$) by moving DOWN. Because the agent is at $y=8$, it must move DOWN $13 - 8 = 5$ times. Next, move the agent to reach the goal ($x=19$) by moving RIGHT. Because the agent is at $x=5$, it must move RIGHT $19 - 5 = 14$ times.

[1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[ACTUAL]

[Observation] {obs}

[Goal] The agent must move to the same cell as the Goal object.

[Plan]

A.3 With NLWM, Zero-Shot Prompting

[Prompt] You are a planner for an agent in a 2D GridWorld environment. You are given an observation of the environment. You are also given a goal that the agent must achieve. You are also given a set of transition effects associated with objects in the environment.

Your task is to generate a high-level plan that the agent can follow to achieve its goal. Then, generate a low-level action sequence that the agent can execute to achieve its goal.

[Observation] {obs}

[Goal] The agent must move to the same cell as the Goal object

[Transition Effects] {nlwm}

[Plan]

A.4 With NLWM, CoT Prompting

[Prompt] You are a planner for an agent in a 2D GridWorld environment. You are given an observation of the environment. You are also given a goal that the agent must achieve. You are also given a set of transition effects associated with objects in the environment.

Your task is to generate a high-level plan that the agent can follow to achieve its goal. Then, generate a low-level action sequence that the agent can execute to achieve its goal. Below is an example observation, goal and plan. Use the example as a reference for your own problem solving steps.

[EXAMPLE]

[Observation] The agent is located at (5, 8). A green object is located at (19, 13). A brown border forms an enclosure around the agent as well as all objects except the green object, and the green object at (19, 13) There is an opening in the enclosure at (18, 13).

[Goal] The agent must move to the same cell as the Goal object.

[Transition Effects] - Green: The green object is the Goal that the agent must reach.
- Brown: Brown objects are False Walls. The agent can move through False Walls.

[Plan] First determine that the green object is the Goal object. Align the agent on the horizontal axis ($y=13$) by moving DOWN. Because the agent is at $y=8$, it must move DOWN $13 - 8 = 5$ times. Next, move the agent to reach the goal ($x=19$) by moving RIGHT. Because the agent is at $x=5$, it must move RIGHT $19 - 5 = 14$ times.

[1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[ACTUAL]

[Observation] {obs}

[Goal] The agent must move to the same cell as the Goal object.

[Transition Effects] {nlwm}

[Plan]