



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona



BABA IS BOT

ENEKO SABATE ITURGAIZ

Thesis supervisor

VÍCTOR GIMÉNEZ ÁBALOS (Department of Computer Science)

Thesis co-supervisor

SERGIO ÁLVAREZ NAPAGAO (Department of Computer Science)

Degree

Bachelor's Degree in Informatics Engineering (Computing)

Bachelor's thesis

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Acknowledgements

Me gustaría agradecer a todas las personas que me han apoyado tanto en mis momentos difíciles como en los buenos. Gracias a mi madre, a mi padre, y a toda mi familia por estar siempre ahí. En especial, a Marina, por soportarme en los peores días, y por aportarme ideas valiosas y mucha ayuda. Finalmente, quiero agradecer a Víctor por todo lo que me ha enseñado y, en general, por haber sido un director excepcional.

Abstract

We introduce an LLM-powered embodied agent capable of lifelong learning in the dynamic-rule environment of *Baba Is You*. Its most distinctive contribution lies in a feedback-driven semantic learning mechanism that allows the agent to autonomously refine its understanding of the environment through interaction and contradiction resolution. At the core of this mechanism is a loop that continuously improves the agent’s rule knowledge: (1) a discrepancy between the simulated and actual world is detected, indicating a misunderstanding of a rule (2) the responsible rule is abducted, and its semantics are induced and generalized (3) new goals are generated to test this revised understanding, expecting new discrepancies to show up. This entire reasoning process is powered by a LLM, which serves as the agent’s inference engine, enabling it to analyze discrepancies and infer the underlying domain logic. The agent demonstrates promising performance in the early levels of *Baba Is You*, showcasing robust semantic learning and abstract reasoning capabilities without any human intervention.

Abstract

Presentamos un agente encarnado potenciado por un LLM, capaz de aprendizaje continuo en el entorno de reglas dinámicas de *Baba Is You*. Su contribución más distintiva radica en un mecanismo de aprendizaje semántico guiado por retroalimentación, que permite al agente refinar de forma autónoma su comprensión del entorno mediante la interacción y la resolución de contradicciones. En el núcleo de este mecanismo se encuentra un bucle que mejora continuamente el conocimiento de reglas del agente: (1) se detecta una discrepancia entre el mundo simulado y el real, lo que indica un malentendido de una regla; (2) se abduce la regla responsable y se inducen y generalizan sus propiedades semánticas; (3) se generan nuevos objetivos para poner a prueba esta comprensión revisada, esperando que aparezcan nuevas discrepancias. Todo este proceso de razonamiento está impulsado por un LLM, que actúa como el motor de inferencia del agente, permitiéndole analizar discrepancias e inferir la lógica del dominio subyacente. El agente muestra un rendimiento prometedor en los primeros niveles de *Baba Is You*, demostrando capacidades sólidas de aprendizaje semántico y razonamiento abstracto sin intervención humana.

Abstract

Presentem un agent encarnat impulsat per un LLM, capaç d'aprendre al llarg de tota la vida en l'entorn de regles dinàmiques de *Baba Is You*. La seva contribució més destacada rau en un mecanisme d'aprenentatge semàntic guiat per retroacció, que permet a l'agent refinjar autònomament la seva comprensió de l'entorn mitjançant la interacció i la resolució de contradiccions. Al centre d'aquest mecanisme hi ha un bucle que millora contínuament el coneixement de les regles de l'agent: (1) es detecta una discrepància entre el món simulat i el real, la qual cosa indica una mala interpretació d'una regla; (2) s'abdueix la regla responsable i se n'indueixen i generalitzen les propietats semàntiques; (3) es generen nous objectius per provar aquesta comprensió revisada, esperant que apareguin noves discrepàncies. Tot aquest procés de raonament està impulsat per un LLM, que actua com el motor d'inferència de l'agent, permetent-li analitzar discrepàncies i inferir la lògica subjacent del domini. L'agent demostra un rendiment prometedor en els primers nivells de *Baba Is You*, mostrant capacitats sólides d'aprenentatge semàntic i de raonament abstracte sense cap intervenció humana.

Contents

1	Introduction	9
2	Background	10
2.1	Foundations of Intelligent Agency	10
2.2	Toward General Intelligence	11
2.3	Agentic AI	11
3	Related Work	12
4	Methodology	14
4.1	Infrastructure	15
4.2	Agent Architecture	16
4.3	In depth module description	17
4.3.1	Critic Module	18
4.3.2	Strategist Module	20
5	Experiments	21
5.1	Experimental Setup	22
5.2	Test Environment: Level 5 "Volcano"	22
5.3	Critic Prompt Method Evaluation	23
5.4	Strategist Module Evaluation	24
5.5	System Integration Evaluation	25
6	Results	26
6.1	Critic Module Outcomes	26
6.2	Strategist Module Outcomes	26
6.3	System Integration Outcomes	27
7	Discussion	28
7.1	Prompting Methods and Resource Trade-offs	29
7.2	Limitations of Language Models	29
7.3	Inference Refinement through Module Collaboration	29
8	Conclusion	30
A	Methodology	36
A.1	Critic Module	36
A.1.1	Full Prompt Components	36
A.1.2	Full Prompt	38
A.1.3	Example	44
A.2	Strategist Module	47
A.2.1	Full Prompt Components	47
A.2.2	Full Prompt	48
A.2.3	Example	50

B Results	52
B.1 Critic Module Outcome Examples	52
B.2 Strategist Module Outcome Examples	53
B.3 System Integration Outcome Examples	54
C Time Planning	55
C.1 Task Description	55
C.1.1 Task Definition	55
C.1.2 Resources	57
C.1.3 Task Summary	58
C.2 Task Estimation	58
C.3 Risk Management	59
D Budget	60
D.1 Costs	60
D.1.1 Staff Costs	60
D.1.2 Software Costs	61
D.1.3 Generic Costs	62
D.1.4 Deviation Costs	63
D.2 Management Control	64
E Sustainability	65
E.1 Self-Assesment	65
E.2 Economic Dimension	66
E.3 Environmental Dimension	66
E.4 Social Dimension	66

List of Figures

1	The modules comprising the <i>Voyager</i> agent	13
2	Overview of the testing infrastructure setup	15
3	Agent Architecture diagram	16
4	Context Diagram of the <i>Critic</i> module	18
5	Illustrated Critic prompt usage examples	19
6	Context Diagram of the <i>Strategist</i> module (create test goals)	20
7	Illustrated Strategist prompt usage example	21
8	A step-by-step solution to the level <i>Volcano</i> of <i>Baba Is You</i> , from top to bottom and left to right	23
9	Comparison of <i>Critic</i> prompting methods.	27
10	Evaluation of <i>Strategist</i> 's performance.	28
11	Gantt chart detailing the project timeline, task durations and their dependencies (Authored by me)	59

List of Tables

1	Integration Evaluation Outcomes over 20 runs	28
2	Task Information Summary (Authored by me)	58
3	Overall Project Costs (Authored by me)	60
4	Staff Costs per Task and Subtask (Authored by me)	61
5	Generic Costs Proration (Authored by me)	63
6	Deviation Costs Breakdown (Authored by me)	63

1 Introduction

Unlike traditional Artificial Intelligence (AI) systems, which are based on pre-defined algorithms, AI agents are autonomous entities that have the capacity to perceive, reason and act within an environment, with the potential to adapt and learn from it. Consequently, these agents, along with a specialized toolset of technologies for memory management, planning and environment interaction, can appropriately replace users or other systems by planning and utilizing available tools. Agentic AI represents an exciting new research direction with significant implications for real-world applications across numerous industries [22, 24].

While AI agents have demonstrated impressive capabilities, current state-of-the-art techniques still struggle when faced with complex, abstract environments. In this regard, classical approaches such as Reinforcement Learning (RL) [23, 3] and Search/Planning algorithms [4, 46] have yielded considerable success, often with little to no human intervention. However, these methods typically rely on predefined action and state spaces or reward functions [47], which constrain their applicability to more general or abstract scenarios. Furthermore, these approaches hinder interpretability and traceability of results [50, 27, 44], as well as lack of generalization across tasks [50, 2, 42, 14].

We argue that meaningful progress toward lifelong learning and adaptive behavior comes from mimicking human cognition. In this context, cognitive architectures have proven to be excellent blueprints for structuring and storing information within agents [37]. However, a critical bottleneck remains: how to autonomously learn the logic of an environment, rather than rely on handcrafted models or manually defined abstractions. Traditional search-based methods can be effective if the environment is meticulously modeled in advance, but this raises a fundamental question: can we truly consider the agent autonomous if every complexity of the task has already been modelled by humans?

This study is centered on the videogame *Baba Is You*, leveraging its rule-based system as a foundation for exploring novel approaches to autonomous agent learning. In the game, the rules are represented as movable blocks of text, and players solve puzzles by rearranging these blocks to dynamically alter the rules that dictate the logic of the environment. Videogames, in general, are known to be powerful domains for such research [30, 34], and *Baba Is You* offers a particularly enticing challenge due to its mutable logic systems and meta-reasoning requirements. Unlike with traditional games, its constantly shifting rules forces agents to perform meta-reasoning, testing their ability to adapt dynamically. Several attempts to solve the game have been made; however, due to the undecidable nature and high combinatorial complexity of its mechanics [16], current general-purpose models fail dramatically when trying to address it [11].

This project aims to develop an agent architecture that can tackle some of the intricacies of *Baba Is You*, in order to export the methodology of learning the game’s abstract and complex action semantics to real world problems. We

do this by leveraging understanding of a human’s mental states when trying to solve these puzzles for the first time. When a human firstly plays the game, they have no idea about the current active rules or even how to beat the level, except for the assumptions one can make from interpreting the natural language in the text blocks. Later, when the player experiments with the world, those assumptions are either confirmed or rectified. Our architecture follows the same premise: the amount of explicit information (procedural knowledge) is minimized, the focus is on the agent learning autonomously from its interactions with the environment. Broadly, the agent implements these 3 capabilities to successfully tackle the problem: (1) propose suitable goals based on the current environment and the assumptions/knowledge about it; (2) make hypothesis about how a certain interaction will affect the environment; (3) detect contradictions between hypothesis and perceptions, reason and synthesize into new hypothesis.

2 Background

2.1 Foundations of Intelligent Agency

The concept of *autonomous agents* in AI inherits from the concept of *rational agents*, defined in the discipline of economics as ‘one who makes choices to maximize their utility given its preferences and knowledge’[40]. This idea has evolved into the design of systems that can perceive their environment, reason about it, and act to achieve specific goals. In real-world environments, however, settings are often complex, unpredictable, and too vast to be fully anticipated or hard-coded by human designers.

The necessity for agents to act autonomously within these complex domains has highlighted the importance for systems capable of abstraction, adaptation and continuous learning. An agent with such characteristics does not only act, but also evaluates the consequences of their actions, revises internal assumptions and refines strategies over time. This becomes essential in the context of *lifelong learning*, where agents operating over long periods of time can, with minimal human intervention, continuously acquire and refine knowledge.

These challenges have motivated the development of cognitive architectures [25, 26], which aim to model not only behavior but also the underlying processes of human-like cognition. These architectures offer a principled way to integrate perception, memory, decision-making, and learning within a unified framework, with human cognition as inspiration. By mimicking the structure and mechanisms of the human mind, these architectures intrinsically promote modularization and human-like reasoning, which have, at the same time, incited exploration on systems with meta-reasoning or reflective abilities[13]: agents that can autonomously revise their performance and based on that self-correct their strategies.

These advancements lay the foundation for the pursuit of more general forms of intelligence.

2.2 Toward General Intelligence

This trajectory intersects with the ideas around Artificial General Intelligence (AGI)[17]: AI systems that have a degree of self-understanding and autonomy, with the ability of solving new problems, learning and adapting to new situations in the process. Unlike narrow AI, which are designed to excel in specific domains, AGI aspires to adapt to novel situations with minimal prior training. However, the concept of AGI is still high-level, and there is ongoing disagreement among researchers regarding the definition of intelligence itself. In the view of DeepMind and OpenAI, intelligence is described as the ability to achieve goals in a wide range of environments[39, 35], often pushing for large-scale models and RL approaches. In contrast, other researchers like François Chollet—creator of the Abstract and Reasoning Corpus for Artificial General Intelligence (ARC-AGI)—are critical of the latter perspective. Chollet argues that intelligence should be defined as the efficiency in acquiring new skills across unfamiliar environments, emphasizing abstraction, reasoning, and generalization from limited data[7].

ARC-AGI was introduced as a benchmark to create a meaningful and fair comparison between artificial intelligence and human intelligence[8, 9]. In this regard, ARC-AGI focuses on *fluid intelligence*—the ability to reason, solve novel problems, and adapt to new situations—rather than *crystallized intelligence*, which relies on deduction from prior knowledge and skills. This is achieved by limiting the *crystallized intelligence* into *core knowledge priors*, that knowledge or intuitions that are either present at birth or acquired very early in human development[43]. For instance, Gestalt principles of perception and trajectory completion are forms of knowledge we acquire early in life through reasoning and interaction with the physical world [21, 48]. This highlights the importance of enabling AI systems to actively interact with their environment, allowing them to acquire new knowledge that, in turn, enhances their reasoning capabilities—fostering a virtuous cycle of learning and understanding.

This setup forces AI systems to generalize from limited examples, interact with its environment, and bolster its capabilities through the new gathered insights. Thus, due to lack of generalization and adaptability, naive RL and Machine Learning (ML) approaches have demonstrated poor performance in the ARC-AGI benchmark. Current state-of-the-art solutions include heuristic-based and symbolic methods, integrating tools like Large Language Models (LLMs) to enhance behaviors such as skill acquisition. However, these systems’ designs distance from the concept of intelligent agents defined by Chollet; thus, performing badly in the benchmark.

2.3 Agentic AI

Agentic AI refers to the use of AI agents to perform tasks autonomously without human intervention. These architectures are particularly useful in the context of abstract, dynamic, or open-ended environments, where tasks are unpredictable or human-design of every intricacy of an environment is infeasible.

Consequently, these agents are not rule-based/symbolic, but use AI techniques as RL, ML and Computer Vision (CV).

Among these, LLMs have emerged as powerful tools for learning and decision-making within environments. Nevertheless, there is ongoing debate about whether LLMs can truly reason or they merely string together patterns, without any semantic understanding. Language models have been qualified by some as sheer *stochastic parrots*[5], models able to generate language but incapable of actually understanding it, similar to Searle’s *Chinese Room* argument[41]. On their own, they lack memory, goals, representation of the external world, and the ability to intervene—features crucial for performing complex tasks and for engaging in genuine reasoning.

These limitations may be overcome through the embodiment of these tools in a larger architecture enabling interaction. In such systems, the agent provides the body—equipped with perceptors, memory, actuators, and other modules—while the LLM serves as a kind of inference engine, responsible for high-level interpretation, abstraction, and overall-reasoning. Embodied cognition theories argue that intelligent behavior arises not just from internal representations, but from the continuous interaction between an agent’s body and its environment [10, 49]. In this view, perception, action, and reasoning form an integrated loop, where the body provides grounding for learning and facilitates causal understanding through feedback. LLMs on their own, lacking this grounding, remain confined to the associative level of Pearl’s causal hierarchy [38], unable to perform interventions or reason about counterfactuals. However, recent advances have demonstrated that embedding LLMs within embodied agents leads to significant improvements in robustness, adaptability, and reasoning capabilities [12]. This is evident in systems such as Voyager [50] and CRADLE [45], where the key contributions lie primarily in the surrounding architecture rather than the language model itself.

3 Related Work

Efforts to achieve Artificial General Intelligence date back to the beginning of AI research. Foundational work, as P. Hayes’ *Naive Physics Manifesto*[20], emphasized on the need for physical world representations—instead of artificial playgrounds—as a prerequisite for creating systems capable of generalization and adaptability. Although a project with ambitious and promising standards, it was not long before McDermott pointed out the impossibility of such framework in its *A critique of pure reason*[29], arguing that such frameworks underestimated the complexity of real-world cognition. More recently, benchmarks like Abstraction and Reasoning Corpus (ARC)[7], introduced by François Chollet, have reignited the interest on the kinds of reasonings that resemble general intelligence. The ARC-AGI[8] and ARC-AGI 2[9] competitions have showcased some progress; however, even top-performing systems remain well below double-digit accuracy on the latest benchmark. Crucially, these approaches are still largely heuristic-based, far from Chollet’s vision of intelligence grounded in abstraction,

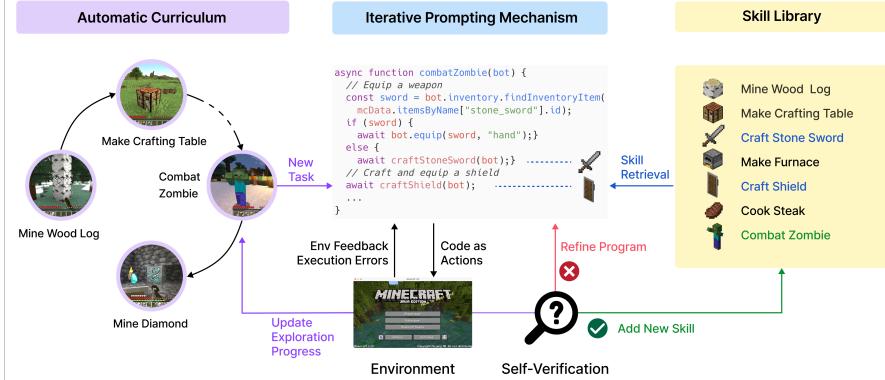


Figure 1: The modules comprising the *Voyager* agent

adaptability, and efficiency. Parallel research, such as Intrinsically Motivated Open-ended Learning (IMOL), aim to endow agents with the capacity to autonomously explore, generate goals, and acquire skills in the absence of explicit goals[36, 15]—capacities that are often labeled as intelligent.

Recent advancements in the capabilities of Large Language Models, including improvements in reasoning, coherence, and accuracy[6, 28], have extended their use within autonomous agents, where LLMs serve as central components for planning, decision-making, and even code generation[51, 52]. One notable development in this context was the Reasoning and Acting (ReAct) framework[53], one of the first agent architectures powered directly by LLMs. ReAct separates the processes of planning and acting, running them in a loop that supports performance revision, effectively creating one of the earliest LLM powered self-improving agent. More recently, *Generative Agents*[37] simulate believable human behavior by combining observation, memory, reflection, and planning in a continuous loop. These agents store experiences in long-term memory, synthesize them into high-level insights, and generate new actions based on reflective reasoning, enabling the emergence of complex, human-like behaviors over time. Building on this idea of long-term autonomy, *Voyager*[50] uses the open-ended world of *Minecraft* as the environment to test its exploration capabilities, where an agent plans goals, learns skills, and improves iteratively through a lifelong learning loop. *Voyager* leverages an adaptive skill library that, with the help of a LLM and feedback from the environment, generates new skills and refines existing ones by writing and updating code. Similarly, *CRADLE*[45] focuses on creating agents that continually self-improve by iteratively generating goals and reflecting on failures to update its knowledge. However, it pushes a step towards generalization, by giving the agent total control over the keyboard and mouse, and having as input screenshots, enabling interaction with any software without the need of Application Programming Interfaces (APIs).

Although somewhat niche, *Baba Is You* has become a valuable testbed in AI due to its intricate and complex mechanics. Some first approaches trying to

solve the game have been search-based, leveraging heuristics with parameters trained using evolutionary algorithms[32]. This work has demonstrated success and efficiency in beating a few simple scenarios; however, the vast rule combinations and its effects make the search space too large for large-scale solution, not to mention the lack of generalization. Other approaches have explored reinforcement learning [19], but they often struggle to converge on effective policies. This is primarily due to the game’s sparse rewards, complex rule dynamics, and heavy reliance on understanding rule semantics—an area where traditional RL systems lack inherent capability. A more recent approach, *Baba Is AI*[11], employs state-of-the-art multi-modal LLMs to plan solutions for simple game scenarios. The method yields insufficient results, as the authors note: "We test three state-of-the-art multi-modal large language models and find that they fail dramatically when generalization requires manipulating and combining the game’s rules". One of the key failures on this last approach can be attributed to lack of embodiment: without a consistent memory and a feedback loop, the LLM on its own has struggles to navigate the game’s intricacies.

4 Methodology

The objective of this work is to design an agent architecture capable of learning environmental semantics, and developing the key components to make it work. Although previous work mastered areas as curriculum planning and skill acquisition (p.e. *Voyager*, *CRADLE*), they relied on complete, LLM-provided (crystallized) knowledge of the domain. This reliance limits their robustness and generalizability, as the logic of the environment is assumed rather than discovered. As a result, any slight change in the domain structure could lead to interpretation failures.

Thus, there is a need for a solution that is capable to learn in unknown or dynamic environments, minimizing given prior knowledge. Such a solution could be valuable in complex scenarios where the underlying structure of the domain must be inferred through exploration, as in puzzle solving and abstract reasoning. Furthermore, dynamic-rule environments specially benefit from this feature: changes in the environment’s logic structure can lead to mutations in the action-space, and an agent with autonomous domain-logic understanding can readapt to these new settings and, when possible, generalize its knowledge across similar scenarios.

We turn to the game *Baba Is You* as the testing environment for our agent’s capabilities. Its dynamic interactive rule system forces agents to discover the effect that these rules cause on their surroundings, and plan accordingly to beat the level. Moreover, due to the interactive nature of the rules, the agent must think meta-logically, modifying the current rules of the environment to reshape the action-space in ways that serve its goals. As a result, it serves as an ideal testbed for emerging autonomous adaptative agents that can reason abstractly.

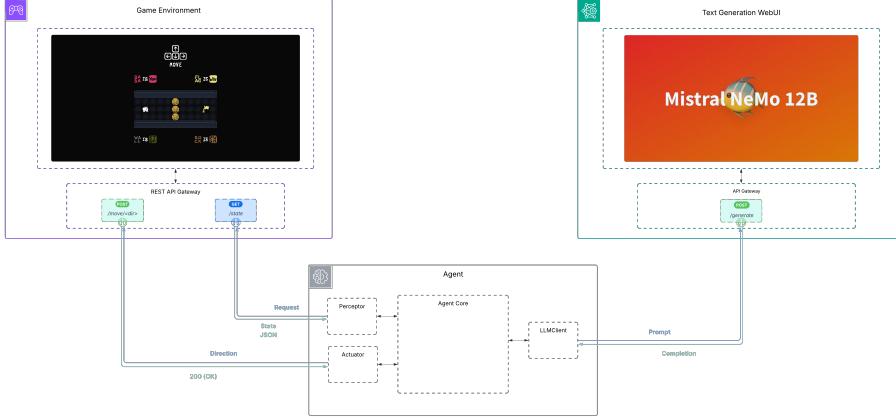


Figure 2: Overview of the testing infrastructure setup

4.1 Infrastructure

The system infrastructure is composed of a set of interconnected, containerized subsystems designed to ensure modularity and replicability. The core components of the system are threefold: the Agent, the LLM Server, and the Game Environment. Both the LLM Server and the Game Environment run in their own isolated *Docker* container to ease deployment and platform independence, facilitating the use of other LLMs and game environments if desired.

The architecture follows a client-server model, with the Agent functioning as the client and both the LLM and Game Environment operating as independent servers. As shown in Figure 2, the Agent communicates with these services via RESTful connections.

- **LLM Server:** This component runs an instance of *Text Generation WebUI (Oobabooga)*¹, which serves a locally hosted language model. The Agent sends requests with prompts to the LLM server’s endpoint, which returns generated text based on the loaded model. This supplies the raw reasoning necessities of the agent.
- **Game Environment Server:** An own custom implementation of a simulated Baba Is You environment², built on C++. On top of the simulation, a server exposes a set of RESTful endpoints that allow the Agent to perceive the game state and perform actions within the environment. This service reduces the workload of the agent by providing seamless perception and actuation.
- **Agent:** The decision-making entity described in detail in Section 4.2. For clarity and modularity, modules responsible for external communications

¹<https://github.com/oobabooga/text-generation-webui>

²<https://github.com/levyy7/BabaIsAPI>

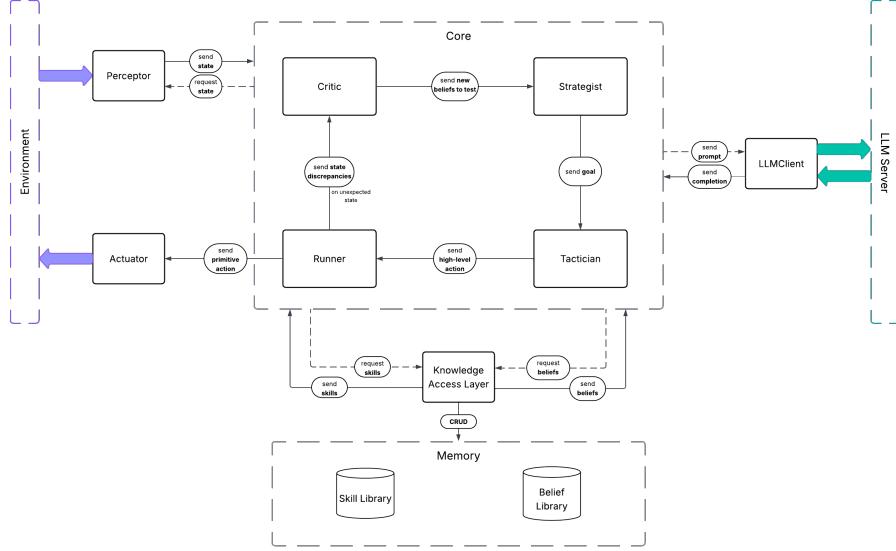


Figure 3: Agent Architecture diagram

with the LLM and the environment are separated from the Agent’s core modules.

4.2 Agent Architecture

The agent architecture is inspired by cognitive models, with most internal modules mirroring cognitive functions or skills within humans. The four most important modules that define the agent’s decision-making and self-revision are what we call *core modules*: *Strategist*, *Tactician*, *Runner* and *Critic*. These modules operate and interact cyclically, emulating the human thought process when solving problems. Supporting them is the Memory, along with a Knowledge Access Layer, which manages and provides structured access to beliefs about rules, skill representations, and related metadata. The architecture also incorporates multiple service clients, enabling the agent to interface with its infrastructure. All system components and their interconnections are illustrated in Figure 3. The execution cycle continuously generates new knowledge entries and refines existing beliefs during runtime, resulting in a lifelong learning process, which is explained in the next paragraphs.

The *Strategist* creates goals to meet the current objectives of the system. Initially, there are no objectives and the *Strategist* defaults to setting the basic goal of "completing the level". As the agent gains new knowledge, mostly from the *Critic*, the *Strategist* generates exploratory goals designed to test and validate these updated beliefs. See section 4.3.2 for implementation-oriented information about the *Strategist* module. When the current goal is set, it is

passed to the *Tactician*.

The *Tactician* takes a goal and creates a list of high-level actions. These actions are structured plans that, when executed altogether, should complete the goal proposed by the *Strategist*. These plans are deliberately structured to include only actions for which the agent has corresponding skill representations. By constraining the planning process in this way, the system guarantees that each high-level action can be effectively translated by the *Runner*.

The *Runner* given a high-level action, creates a list of game-allowed actions (primitive). More specifically, the module sets a "goal state" and creates a list of game actions which, when executing them, should result in the goal state. Take into account that, for converting actions from high-level to primitive, the code associated to this should be updated accordingly with the new understanding of rules. In the first iterations of the agent, this module will fail dramatically to achieve the "goal state". That is why the *Runner* simulates each expected state until reaching the *goal state* and, when encountering a difference between the expected state and the real state, calls the *Critic* to resolve the discrepancy.

The *Critic* takes a discrepancy between two states and updates the beliefs that caused such discrepancy. The rationale behind this module lies in the hypothesis that, if the *Tactician* and *Runner* work correctly, then the only reason why a discrepancy would appear would be due to a misunderstanding of the logic of the world. See Section 4.3.1 for implementation-oriented information about the *Critic* module. Due to level design and technical limitations, a large amount of the updated beliefs will be wrong, or at least will cover the effects of a rule partially.

4.3 In depth module description

The second main objective of this work is to develop and implement the architectural modules that enable semantic learning, modeled after human cognitive processes. Specifically, we focus on designing a mechanism for acquiring and updating the rules of *Baba Is You* in a manner analogous to how humans learn and revise their understanding of new environments. To illustrate this, we emphasize on two of the four core modules in our architecture: the *Strategist* and the *Critic*. These modules are central to semantic inference and adaptation—*Strategist* is responsible for hypothesis and test generation, while *Critic* evaluates inconsistencies and refines internal beliefs when expectations fail to match reality.

These modules are supported by a LLM, which is utilized for tasks requiring natural language understanding, abstract reasoning, and hypothesis generation. Recent advances in LLMs have made them suitable for such cognitively demanding tasks due to their emergent perceived reasoning capabilities. This is particularly relevant in the context of *Baba Is You*, where rules are not just symbolic constructs but are expressed in natural language. Although rule-meaning induction could, in theory, be handled purely through a trial-and-error basis, interpreting and assigning meaning to linguistic expressions, as humans do, profits from all the resources of the domain.

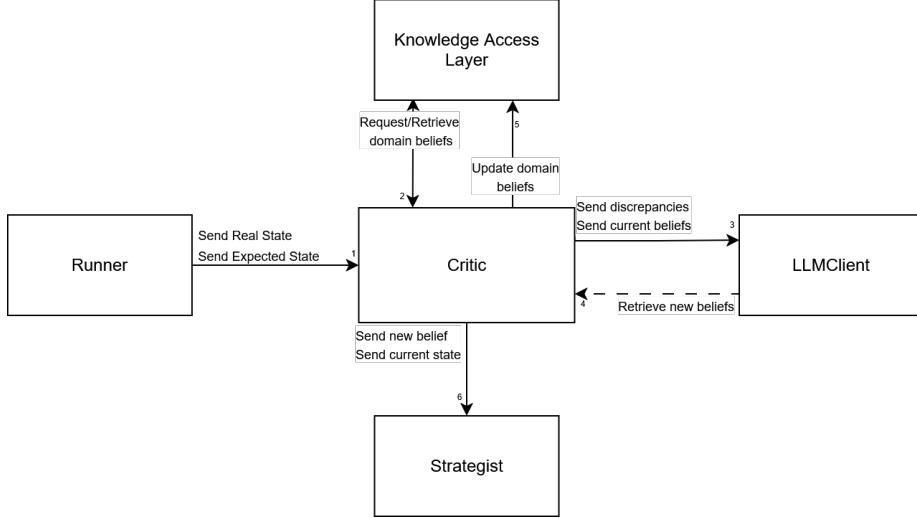


Figure 4: Context Diagram of the *Critic* module

4.3.1 Critic Module

The *Critic* module is the component where the semantic acquisition primarily occurs. It detects discrepancies between expected and actual game states, the agent's current beliefs about the rules, and the performed action. These discrepancies trigger a reasoning process aimed at refining the agent's understanding of the underlying rules. In essence, when an outcome deviates from what was anticipated, the Critic attempts to induce a more accurate semantic interpretation of the relevant rule. This process, along with its context, is illustrated in Figure 4.

As previously mentioned, this reasoning is handled by an LLM, which is responsible for identifying the cause of the discrepancy. However, prompting the LLM effectively is non-trivial, as simply dumping all the information yields poor results. Considerable effort has gone into crafting effective prompt strategies. Broadly, five configurations have been explored:

1. Both the expected and actual states are provided, including a full map representation (as a list of blocks and their positions), the rules existing in both states, the current understanding of these rules of the agent, and the performed action.
2. Builds on (1) by also including the previous state before the action. This way, the LLM grasps better the event at hand.
3. Instead of the full map, only the information directly related to the discrepancy is provided—specifically, the names and positions of the conflicting blocks in the previous, expected, and actual states. Full rule sets, rule-beliefs and the action are still included.

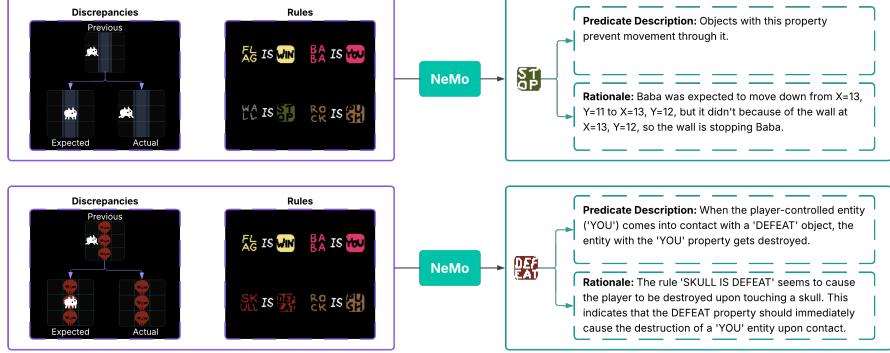


Figure 5: Illustrated Critic prompt usage examples

4. An extension of (3) using prompt chaining[31], where the output of one prompt is fed into the next. This iterative method enables reasoning refinement across multiple steps.
5. Also based on (3), self-reflection[18] adds a final step where the LLM reflects on earlier outputs, synthesizing them into a single, final response. Like prompt chaining, this requires multiple prompts per decision.

Out of these 5, (1) and (2) have been discarded almost immediately due to the confusions and hallucinations the LLM has suffered because of large input lengths and lack of spatial reasoning capabilities. (4) and (5), although being more refined, have been discarded due to their low performance gains while incurring substantial computational costs due to their multi-step prompt pipelines (see Section 6.1). Thus, (3) has been elected as the proposed prompt format for the *Critic* module.

See Figure 5 for visual examples of these prompts. The key elements included in the selected prompt format are:

- **Explanations about rule structures.** Stating that a rule is composed by a subject, verb and predicate, and that the effect of a rule is determined by its predicate.
- **State discrepancies**, specifically: the differences between the state previous to the discrepancy, the expected state and the actual state.
- **Action performed** to reach the state where the discrepancy has occurred.
- **Current beliefs** about the effects that blocks existing in the current level have as rule predicates.
- **Diverse constraints and encouraging behaviors**, such as stating that the output should be in a structured format.

Check A.1.2 for a full example of the prompt and its output.

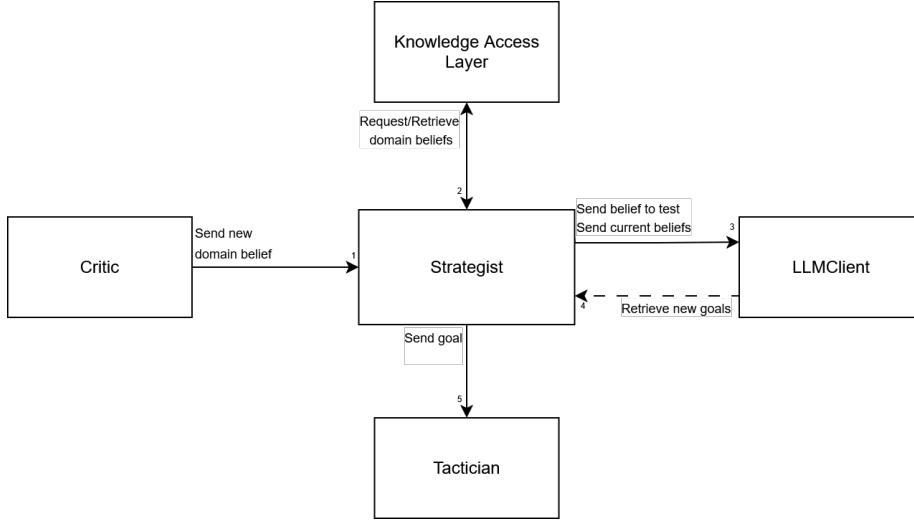


Figure 6: Context Diagram of the *Strategist* module (create test goals)

4.3.2 Strategist Module

The *Strategist* is responsible for generating and setting goals within the agent. The default goal of the system is to "Win the current level". However, whenever the *Critic* induces a new rule-belief, a method in the *Strategist* is called that generates new goals aimed at testing this recently acquired belief. Consequently, if the rule was erroneously induced, these tests will fail, calling again the *Critic* and the *Strategist* afterwards, repeating the process. This process is illustrated at Figure 6. These new goals are placed on a stack-like structure, where until all "test goals" are not resolved, the default goal of completing the level will not be set.

This reasoning is handled by an LLM, which is responsible for generating specific goals to test the provided rule. However, as seen on the *Critic* module, prompting the LLM effectively is non-trivial, as simply dumping all the information yields poor results. Considerable effort has gone into crafting effective prompt strategies. Broadly, three configurations have been explored:

1. The prompt is provided with the map of the state in which the discrepancy occurred, along with the current rules and the rule under evaluation. The map is represented as a dictionary, where each entry corresponds to a block and the values list the positions where that block appears.
2. Similar to (1), but the map is represented as a matrix instead of a dictionary.
3. This version omits the map, providing only rule-beliefs alongside the rule to test.

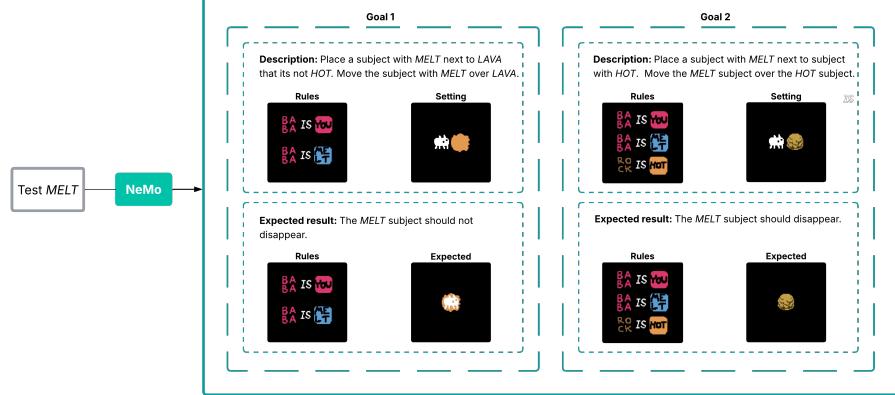


Figure 7: Illustrated Strategist prompt usage example

The LLM struggled with configurations (1) and (2), likely due to limited spatial reasoning and large input length. Prompt (3), on the other hand, performed significantly better, yielding plausible and useful goals. However, since it lacks a specific map context, the language model must create a scenario to test the rule, sometimes resulting in infeasible or unrealistic settings.

See figure 7 for a visual example of this prompt. The key elements included in the selected prompt format are:

- **Explanations about rule structures.** Stating that a rule is composed by a subject, verb and predicate, and that the effect of a rule is determined by its predicate.
- **Belief to test**, which has previously been induced by the *Critic*.
- **Rule beliefs** about all the predicates encountered by the agent.
- **Diverse constraints and encouraging behaviors**, such as stating that the output should be in a structured format.

Check A.2.3 for a full example of the prompt and its output.

5 Experiments

The ultimate goal of this experimentation is to assess whether the developed modules can acquire sufficient semantic understanding of a level’s rules to theoretically complete it. Testing each module individually is a remarkably complex problem, as there is no way to automate or design an experiment whereby a success score is obtained for a module in isolation.

As such, we take a more subjective, qualitative approach by evaluating both *Critic* and *Strategist* using a human-review process (Section 5.3 and 5.4). However, this is not enough to validate the system. Instead, after individual qualitative tests, a quantitative, system-wide integration test is performed to test the end-to-end of the proposed cognitive architecture on a level (Section 5.5).

5.1 Experimental Setup

The main components used in the setup are the LLM and game environments (as shown in Figure 2). We leverage MistralAI’s *Mistral-Nemo-Instruct-2407*[1], deployed locally using Oobabooga’s *Text Generation WebUI*[33]. As for the environment running *Baba Is You*, a custom version of the game is run locally to facilitate information retrieval and agent actuation.

5.2 Test Environment: Level 5 "Volcano"

For the next experiments, we will focus on the level 5 of *Baba Is You, Volcano*. This level has been chosen because it is the first level of the game for which all the potential of the agent can be showcased. Previous levels may also be used in experimentation to the extent that it is necessary for the agent to understand them and play them before level 5. However, these have not been further utilized as they had the rules induced by the *Critic* only in one iteration; thus, giving no chance to showcase the support of the *Strategist* module to the semantics gathering process.

Before tackling this level, the agent is assumed to have already induced the meaning of four key rule components through prior levels: *YOU*, *WIN*, *PUSH*, and *STOP*. In particular, the agent understands that:

- *YOU*: Makes the subject controllable by the player.
- *WIN*: Ends the game in a *WIN* when touching the subject.
- *PUSH*: Makes the subject pushable by the player.
- *STOP*: Makes the subject impassable for the player.

These semantics serve as prior knowledge, enabling the system to focus its hypothesis generation and testing efforts on the unfamiliar properties presented in the current level.

In *Volcano*, the properties *HOT* and *MELT* are introduced. These two rule components interact: if an object with the *MELT* property comes into contact with one possessing the *HOT* property, the former is destroyed. This presents the first instance of interaction between properties in the game, requiring the agent to move beyond surface-level lexical hints and perform semantic induction through hypothesis testing. Furthermore, the lexicon of these new rules offers less immediate semantic clarity compared to earlier rules like *WIN* or *STOP*, which are more intuitively understood. This makes the level a strong candidate



Figure 8: A step-by-step solution to the level *Volcano* of *Baba Is You*, from top to bottom and left to right

for assessing the system’s ability to go beyond lexical assumptions and uncover meaning through structured exploration and reasoning.

Although this level allows multiple solutions, we will use the one portrayed in Figure 8 as the basis for future experiments. This solution is explained in the following steps:

1. Move to the side of the rule *BABA IS YOU*, and push it, in order to be able to access the rest of the map without breaking the rule.
2. Form the rule *LAVA IS MELT*. This in composition with the rule *LAVA IS HOT* will cause the lava in the map to evaporate due to being *HOT* and *MELT* at the same time.
3. Now that lava is not a problem, walk towards the flag to complete the level.

5.3 Critic Prompt Method Evaluation

We explore the most suitable prompting method for the *Critic* module in our current setup. Specifically, we leverage the following prompting methods: 1) Single (regular) prompting 2) Prompt-Chaining[31], where the output of one prompt is used as input for the next (3 prompts total) 3) Self-Reflection prompting[18], where a final LLM call evaluates prior outputs and selects the best one based on reasoning quality (3 prompts total).

After encountering a discrepancy, the *Critic* attempts to abduct the rule responsible for the unexpected behavior and induce its semantics. The outcome

of each test is an updated interpretation of the predicate presumed to have caused the discrepancy.

For this purpose, we evaluate 5 distinct scenarios where discrepancies — that is, an unexpected state in the environment — occur, all five gathered from the first 5 levels of the game. These scenarios align with the different situations a human could come across when trying to solve those levels. Particularly, those 5 discrepancies are:

- The player attempts to walk into a wall while the rule *WALL IS STOP* is active. The agent, unfamiliar with the semantics of *STOP*, expects to pass through the wall. Instead, movement is blocked, indicating a discrepancy between the agent’s prediction and the actual outcome.
- The player moves into an object with the *DEFEAT* property. The agent expects to walk over the skull safely, not knowing the rule’s effect. However, upon contact, the player object is destroyed, signaling a critical misunderstanding.
- The player attempts to push an object that in front of another object with the *PUSH* property. The agent assumes that only the first object will move. In reality, all pushable objects in the chain are pushed together.
- The player object has the property *MELT*, and walks into an object that has the *HOT* property. The agent, knowing neither property, expects to pass unharmed. Instead, the player object is destroyed.
- The player forms the rule *ROCK IS LAVA*. The agent expects the ROCK objects to remain unchanged. Instead, all ROCK objects vanish and are replaced by LAVA.

To evaluate the performance of each prompting method, the *Critic* is executed 10 times per discrepancy per method, totaling 150 runs. The outputs are assessed through a structured human evaluation process. Each iteration is reviewed and scored by a human evaluator on a scale from 0 to 10, based on three key criteria: (1) the accuracy of the induced property semantics, (2) the appropriateness of the rule abduction—*i.e.*, whether the identified rule plausibly explains the discrepancy, and (3) the clarity and simplicity of the explanation.

5.4 Strategist Module Evaluation

The objective of this experiment is to evaluate the overall performance and consistence of the *Strategist* module in our current setup. The first 4 levels generally have easier rules to induce, and the *Critic* has been able to induce the meaning of the rules causing the discrepancies quite easily. In this cases, the tasks of the *Strategist* becomes trivial; thus, we will only leverage cases belonging to the level 5 due to its interesting and challenging scenario.

The specific case selected involves a recurring failure by the *Critic* to accurately induce the semantics of the *MELT* property. In this scenario, the agent

erroneously believes that an object with the *MELT* property will be destroyed upon contact with lava. However, the actual rule logic dictates that objects with *MELT* are destroyed upon contact with any object that has the *HOT* property. This has been chosen as a test case as this is a recurrent output when testing this discrepancy in the *Critic* module.

The expected output of the *Strategist* is a pair of natural language goals designed to test the agent’s belief about the *MELT* property. Additionally, the rationale behind the test and the expected outcomes are included.

To assess the module’s performance, the *Strategist* is run 50 times with the same input scenario. Outputs are evaluated through a structured human-review process. Each iteration receives a score from 0 to 10, based on three key criteria: (1) whether the proposed test could plausibly reveal new, useful information about the target predicate, (2) whether the test aligns with the logic and constraints of the game environment, and (3) whether the test is simple, well-structured, and described in a clear, step-by-step manner.

5.5 System Integration Evaluation

The final experiment evaluates the semantic learning capabilities of the full agent when presented with a new level. For this purpose, we use Level 5 of *Baba Is You, Volcano*, as the testbed. This experiment focuses on assessing the performance and cooperation between the *Critic* and *Strategist* modules—the two central components responsible for rule abduction and semantic induction. The remaining modules, *Tactician* and *Runner*, are simulated for the specific scenario (according to the solution proposed at Section 5.2) and not directly evaluated. If the *Critic* and *Strategist* are able to gather the necessary semantics—particularly for the properties *HOT* and *MELT*—then, in principle, the agent will possess the knowledge required to devise and execute a plan that successfully completes the level.

The experimental procedure proceeds as follows: each run consists of a chain of four module calls—starting with a call to the *Critic*, followed by the *Strategist*, and finally two additional calls to the *Critic* to evaluate the tests proposed by the *Strategist*. If the modules function as intended, the final belief state of the agent should reflect the correct semantics of the involved properties, enabling successful level completion.

We identify three possible outcomes for each run:

1. The *Critic* successfully abducts the correct rule and induces the accurate semantics in the first call, making the intervention of the *Strategist* unnecessary.
2. The *Critic* initially fails to infer the correct semantics, but the *Strategist* generates tests that allow the *Critic* to revise and correct its beliefs.
3. Neither the initial *Critic* call nor the follow-up tests result in correct semantic inference.

To evaluate the effectiveness of the integration between *Critic* and *Strategist*, we execute the process described previously 20 times. Success is defined as the agent arriving at a correct knowledge state regarding the rules and properties necessary to beat the level—i.e., outcomes (1) and (2). Failure corresponds to outcome (3), where the agent does not acquire the required semantic understanding.

6 Results

The results of the aforementioned experiments are organized as follows: In Section 6.1 we show the results of the *Critic* working in isolation, leverage the results of each prompting method and prove that the module is functional. In Section 6.2, similarly, prove that the *Strategist* works. Finally, in Section 6.3 we focus on system integration between the two main modules, testing its cooperation in the frame of the level *Volcano*.

6.1 Critic Module Outcomes

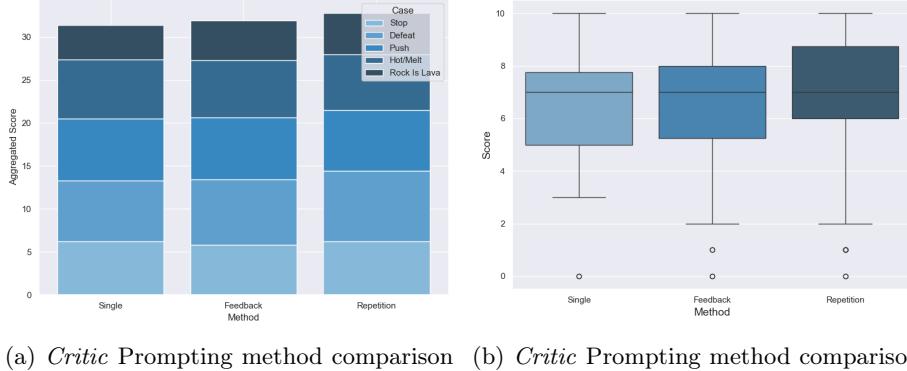
Figure 9a showcases the outcomes of each different prompting method by bar, separating each case with a different color. We can see that the overall performance of the 3 methods is similar, being the feedback and repetition methods slightly better than the single one. Similarly, the separation between cases shows us that there are no big differences between the performance per task. It also shows a similar performance per task, but all methods have had a worse performance regarding case 5 (*Rock Is Lava*).

Figure 9b presents the distribution of scores per prompting method using a boxplot. The average performance of all three methods is comparable, with mean scores around 7. Notably, the feedback and repetition methods show greater variability than the single method, as evidenced by the presence of more extreme scores. Additionally, we observe that different test cases exhibit varying performance, with the most notable outlier being the case of *ROCK IS LAVA*, which scored significantly lower than the others.

To illustrate why this occurred, we analyze some representative outputs from this case (for full examples, see Appendix B.1). An example with a score of 1 stated that forming *ROCK IS LAVA* would turn rocks into lava only when adjacent—an ambiguous and incorrect interpretation. In contrast, a score of 5 described that such a rule would turn all rocks into lava, while an even better scoring output generalized correctly, stating that the subject of the rule would transform into lava.

6.2 Strategist Module Outcomes

Figures 10a and 10b display the performance of the *Strategist* module. Most iterations achieved scores very centered on 5.5. As with the *Critic* module, the



(a) *Critic* Prompting method comparison by task (b) *Critic* Prompting method comparison barplot

Figure 9: Comparison of *Critic* prompting methods.

Strategist shows a lack of consistency, with scores spanning a wide range, from very low (indicating failure) to high (indicating near-perfect outputs).

In order to showcase these occurrences, we focus on concrete examples (for full examples, see Appendix B.2). One rare yet notable failure, which received a score of 1, suggested that a player entity should not pass through another entity with the *MELT* property, nonsensical given the current context. A more recurring case involved the *Strategist* attempting to test whether *LAVA* was responsible for eliminating entities with *MELT*, by placing a lava entity (without any rule assigned) in contact with a subject marked *MELT*, scoring a 6. Finally, the most effective test case—which could best help the *Critic* infer the behavior of *MELT*—involved introducing a subject with the *HOT* property and observing its interaction with a *MELT* entity, correctly leading to its destruction.

6.3 System Integration Outcomes

Table 1 presents the outcome distribution from 20 independent runs of the integrated agent system on the *Volcano* level. We see the system is remarkably successful, as in the majority of cases (14/20) the combination of modules successfully infers the correct meaning of the *MELT* property. Furthermore, in most of those cases (9/14) this is done on the first attempt without need for the *Strategist* module to give feedback to the *Critic*; thus making it highly efficient. No case was observed where the *Strategist* forced a retraction of a correct critic rule. In the remaining 6 cases, neither the initial nor follow-up interactions led to the acquisition of the correct semantics. Overall, the system achieved a success rate of 70% (cases 1 and 2), demonstrating functional coordination between the two modules under evaluation.

We have gathered some example cases to illustrate the cases described above (for full examples, see Appendix B.3). In an example for case (1), the *Critic* correctly inferred the semantics of the *MELT* rule independently, noting that

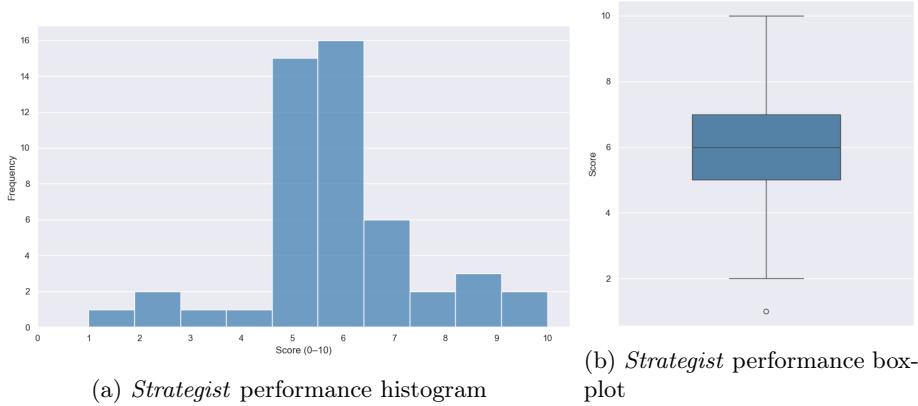


Figure 10: Evaluation of *Strategist*'s performance.

Outcome Type	Description	Count
(1) Immediate Success	<i>Critic</i> infers correct semantics on first attempt	9
(2) Strategist-Assisted Success	<i>Strategist</i> tests enable successful correction	5
(3) Failure	No correct inference achieved	6
Total		20

Table 1: Integration Evaluation Outcomes over 20 runs

an entity with *MELT* is destroyed upon contact with something *HOT*. In a case for (2), an initial incorrect or incomplete induction prompted the activation of the *Strategist*, which generated a targeted scenario where both *HOT* and *MELT* properties were being tested simultaneously. This intervention successfully led to a second reasoning cycle and a correct induction, closely resembling the result of the first case. Lastly, in the third case, both modules failed to fully generalize the rule: the *Critic* produced an overly specific hypothesis, incapable of generalizing that the *HOT* property was the one causing the destruction of the *MELT* entity, instead of lava. Meanwhile, the goal generated by the *Strategist*, though correctly identifying that the current meaning of *MELT* was incorrect, it lacked the specificity or contextual cues needed for the *Critic* to revise its belief adequately in the following iteration.

7 Discussion

The isolated evaluations of the *Critic* and *Strategist* modules have revealed promising capabilities in the previous evaluations, albeit with a notable degree of inconsistency. Despite occasional errors, both modules generally succeeded in understanding rule semantics and proposing meaningful hypothesis-oriented

goals. However, even if the induction has errors, with the capability to intervene and observe action consequences in the environment, this is inconsequential: if it fails at the first few times but succeeds later, it still means that a piece of knowledge was obtained, and the mechanisms are in place to correct the knowledge.

7.1 Prompting Methods and Resource Trade-offs

The comparison between prompting methods indicates that the single prompting strategy performs similarly to the more resource-intensive feedback and repetition approaches. Although the multi-prompting methods occasionally yielded marginally better results, we do not consider the improvements substantial enough to justify their additional computational cost. Furthermore, these methods show greater score variability, which is undesirable due to them being more susceptible to complications during the semantic refinement process. In contrast, more moderate or average outputs are easier to interpret and manage effectively. Based on these findings, the *single* prompting method will be used as the default approach in future iterations.

It is important to note that these results are partly influenced by the choice of language model and the complexity of the test cases. It remains plausible that multi-prompting could yield improved outcomes in more challenging environments or when used with a more powerful LLM.

7.2 Limitations of Language Models

Both modules demonstrated issues commonly associated with LLMs. A recurring problem are hallucinations, that is, when the model uses nonexistent rules/blocks, contradicts itself, or makes assumptions based on invalid game logic. In the case of the *Critic*, inaccuracies often appeared when attempting to infer the actual effects of game rules. These issues, while expected, reveal the need for belief revision and refinement. Nevertheless, in the majority of instances, both the *Critic* and *Strategist* modules succeeded in identifying rule semantics (at least partially) and generating testable goals, showcasing that the language models are suitable for this use.

Some internal tests have been made by replacing the current LLM with more advanced ones, such as Google’s *Gemini* (via its free API). These seemed to produce more accurate and stable outputs, pointing to the quality of the inference engine as a bottleneck for the capacities of the agent.

7.3 Inference Refinement through Module Collaboration

While both *Critic* and *Strategist* have performed well independently, their combined use yields significantly better results. The integration experiment highlights how the *Strategist* can guide and enhance the inference process led by the *Critic*. By providing hypothesis-oriented goals, the *Strategist* helps to boost the inference process toward more accurate rule meanings. This feedback loop

proved highly effective, resulting in a significant improvement over what the *Critic* achieved on its own (see Table 1).

Although the system is only partially complete, having two modules still in development, both *Critic* and *Strategist* sufficiently demonstrate the central contribution of this thesis: semantic acquisition. Additionally, this work supports the use of agent embodiment and, more broadly, feedback loops as mechanisms that can drive emergent improvements in task performance. It also opens the door to add further components in the belief inference process, further improving the quality output.

8 Conclusion

In this work, we introduce the design of a cognitive architecture capable of lifelong learning in complex dynamic-logic environments without any human intervention. We implemented two of the modules of the agent, which have enabled semantic learning in the context of *Baba Is You*. This has been achieved by leveraging a language model, which served as the inference engine embedded within our agent. Although LLMs can't be said to have semantic knowledge themselves, the proposed architecture and embodiment that allows it to interact with the environment appears to enhance them with such semantic acquisition capabilities, albeit with inconsistent results. This setup enabled the agent to identify discrepancies in its environment, abduct the responsible rules, and induce generalized meanings for those rules. This process was encapsulated within a feedback loop designed to iteratively test and refine the agent's rule-beliefs.

The viability of this approach was demonstrated through the experiment conducted on the *Volcano* level of *Baba Is You*. Although this level is relatively simple compared to more advanced stages of the game, our agent exhibited promising semantic learning capabilities. Furthermore, tests suggest that replacing the current LLM with a more powerful alternative could lead to even greater gains in performance and consistency.

Nevertheless, there is still quite work to be done towards having a fully functional agent. Although the agent's high-level design is complete and modularized, the *Tactician* and *Runner* have not yet been developed. These components are not intended to perform learning themselves but rather to execute deductive reasoning: given a correct understanding of the environment and a set of learned skills, they must be able to generate and execute plans to achieve predefined goals. Recent work has shown that such capabilities can be achieved through the use of *Automatic Curriculums* and *Skill libraries*, using LLMs as inference engines[50, 45].

However, we argue that relying on LLMs for deductive planning may not be the most appropriate solution. The stochastic nature of LLMs introduces inconsistency and uncertainty into a problem which can be solved deterministically. Additionally, it adds a layer of resource consumption and unnecessary knowledge which could potentially be avoidable. Other classical symbolic approaches must be revised, which have historically worked fine in deduction. One of such

approaches could be the construction of an ontology governed by formal logic, which would offer a more robust and interpretable framework to work in.

In summary, by demonstrating the feasibility of semantic learning through environmental reasoning and feedback-driven inference in the world of *Baba Is You*, we take an important step toward building agents that can operate effectively and independently in general complex, dynamic worlds.

References

- [1] Mistral NeMo | Mistral AI.
- [2] Francisco Arana-Herrera. Counting problems from the viewpoint of ergodic theory: from primitive integer points to simple closed curves. 2 2022.
- [3] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.
- [4] Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35, 6 2022.
- [5] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? . In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623, Virtual Event Canada, March 2021. ACM.
- [6] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of Artificial General Intelligence: Early experiments with GPT-4, April 2023. arXiv:2303.12712 [cs].
- [7] François Chollet. On the Measure of Intelligence, November 2019. arXiv:1911.01547 [cs].
- [8] François Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. Arc prize 2024: Technical report. *arXiv preprint arXiv:2412.04604*, December 5 2024.
- [9] François Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. Arc-AGI-2: A new challenge for frontier ai reasoning systems. *arXiv preprint arXiv:2505.11831*, May 17 2025.
- [10] Andy Clark. *Being there: putting brain, body, and world together again*. A Bradford book. MIT Press, Cambridge, Mass., 1. paperback ed edition, 2001.
- [11] Nathan Cloos, Meagan Jens, Michelangelo Naim, Yen-Ling Kuo, Ignacio Cases, Andrei Barbu, and Christopher J. Cueva. Baba Is AI: Break the Rules to Beat the Benchmark, July 2024. arXiv:2407.13729 [cs].
- [12] Michael T Cox. Computational Metacognition.
- [13] Michael T Cox. Metacognition in Computation: A Selected History.

- [14] Linxi Fan, Guanzhi Wang, De-An Huang, Zhiding Yu, Li Fei-Fei, Yuke Zhu, and Animashree Anandkumar. Secant: Self-expert cloning for zero-shot generalization of visual policies, 7 2021.
- [15] Sébastien Forestier, Yoan Mollard, and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. In *Advances in neural information processing systems*, pages 1393–1402, 2017.
- [16] Jonathan Geller. Baba is you is undecidable. *IEEE Conference on Computational Intelligence and Games, CIG*, 4 2022.
- [17] Ben Goertzel and Cassio Pennachin, editors. *Artificial general intelligence. Cognitive technologies*. Springer, Berlin ; New York, 2007.
- [18] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. A Survey on LLM-as-a-Judge, March 2025. arXiv:2411.15594 [cs].
- [19] Haoyu Liu, Fan-Yun Sun, Frieda Rong, Kumiko Nakajima, Nicholas Haber, and Shima Salehi. Characterizing Learning Progress of Problem-Solvers Using Puzzle-Solving Log Data. July 2023. Publisher: Zenodo.
- [20] Patrick J. Hayes. The Naive Physics Manifesto. *Expert Systems in the MicroElectronic Age*, 1979. Publisher: Edinburgh University Press.
- [21] Scott P. Johnson, J. Gavin Bremner, Alan Slater, Uschi Mason, Kirsty Foster, and Andrea Cheshire. Infants’ Perception of Object Trajectories. *Child Development*, 74(1):94–108, 2003. _eprint: <https://srcd.onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8624.00523>.
- [22] Sayash Kapoor, Benedikt Stroebel, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. Ai agents that matter. 7 2024.
- [23] Jens Kober, † J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. 2013.
- [24] Naveen Krishnan. Ai agents: Evolution, architecture, and real-world applications. 3 2025.
- [25] John E Laird, Allen Newell, and Paul S Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, 1987.
- [26] Pat Langley, John E Laird, and Scott Rogers. Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2):141–160, 2009.

- [27] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *Proceedings - IEEE International Conference on Robotics and Automation*, 2023-May:9493–9500, 9 2022.
- [28] Joy Mahapatra and Utpal Garain. An Extensive Evaluation of Factual Consistency in Large Language Models for Data-to-Text Generation, November 2024. arXiv:2411.19203 [cs].
- [29] Drew McDermott. A critique of pure reason. *Computational Intelligence*, 3(1):151–160, 1987. _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8640.1987.tb00183.x>.
- [30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. 12 2013.
- [31] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. WebGPT: Browser-assisted question-answering with human feedback.
- [32] Christopher Olson, Lars Wagner, and Alexander Dockhorn. Evolutionary Optimization of Baba Is You Agents. In *2023 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, Chicago, IL, USA, July 2023. IEEE.
- [33] oobabooga. oobabooga/text-generation-webui, June 2025. original-date: 2022-12-21T04:17:37Z.
- [34] OpenAI, :, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olson, Jakub Pachocki, Michael Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. 12 2019.
- [35] OpenAI, Josh Achiam, et al. GPT-4 Technical Report, March 2024. arXiv:2303.08774 [cs].
- [36] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2007.
- [37] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive

simulacra of human behavior. *UIST 2023 - Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 4 2023.

- [38] Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd edition, 2009.
- [39] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A Generalist Agent, November 2022. arXiv:2205.06175 [cs].
- [40] Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice Hall series in artificial intelligence. Pearson, Boston Columbus Indianapolis, third edition, global edition edition, 2016.
- [41] John R Searle. MINDS, BRAINS, AND PROGRAMS. 1980.
- [42] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. *Proceedings of Machine Learning Research*, 164:894–906, 9 2021.
- [43] Elizabeth S. Spelke and Katherine D. Kinzler. Core knowledge. *Developmental Science*, 10(1):89–96, 2007.
- [44] Shao-Hua Sun, Te-Lin Wu, and Joseph J. Lim. Program guided agent. In *International Conference on Learning Representations*, 2020.
- [45] Weihao Tan, Wentao Zhang, Xinrun Xu, Haochong Xia, Ziluo Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, Ruyi An, Molei Qin, Chuqiao Zong, Longtao Zheng, Yujie Wu, Xiaoqiang Chai, Yifei Bi, Tianbao Xie, Pengjie Gu, Xiyun Li, Ceyao Zhang, Long Tian, Chaojie Wang, Xinrun Wang, Börje F. Karlsson, Bo An, Shuicheng Yan, and Zongqing Lu. Cradle: Empowering Foundation Agents Towards General Computer Control, July 2024. arXiv:2403.03186 [cs].
- [46] DeepMind Interactive Agents Team, Josh Abramson, Arun Ahuja, Arthur Brussee, Federico Carnevale, Mary Cassin, Felix Fischer, Petko Georgiev, Alex Goldin, Mansi Gupta, Tim Harley, Felix Hill, Peter C Humphreys, Alden Hung, Jessica Landon, Timothy Lillicrap, Hamza Merzic, Alistair Muldal, Adam Santoro, Guy Scully, Tamara von Glehn, Greg Wayne, Nathaniel Wong, Chen Yan, and Rui Zhu. Creating multimodal interactive agents with imitation and self-supervised learning. 12 2021.
- [47] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

- [48] Peter Ulric Tse. Dynamic Volume Completion and Deformation. *i-Perception*, 8(6):2041669517740368, December 2017.
- [49] Francisco J. Varela, Evan Thompson, and Eleanor Rosch. *The Embodied Mind: Cognitive Science and Human Experience*. MIT Press, 1991.
- [50] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Anima Anandkumar, Ut Austin, and Uw Madison. Voyager: An open-ended embodied agent with large language models. 5 2023.
- [51] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A Survey on Large Language Model based Autonomous Agents. *Frontiers of Computer Science*, 18(6):186345, December 2024. arXiv:2308.11432 [cs].
- [52] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. The Rise and Potential of Large Language Model Based Agents: A Survey, September 2023. arXiv:2309.07864 [cs].
- [53] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models, March 2023. arXiv:2210.03629 [cs].

A Methodology

A.1 Critic Module

A.1.1 Full Prompt Components

- General Context and Objective
 - The AI is assisting in solving turn-based puzzle levels.
 - Objects in the game have no inherent properties; behavior is entirely defined by rules of the form <SUBJECT> IS <PREDICATE>.
 - The goal is to discover which misinterpretation of a rule has caused a discrepancy and update one predicate belief accordingly.

- Methodology Instructions

- Step-by-step reasoning is required to:
 1. Analyze the player position.

2. Compare map discrepancies using existing beliefs and rules.
3. Identify the relevant predicate from the provided list.
4. Update a single belief using a strict JSON format.
 - Only one predicate belief should be updated per output.
 - The output must be in raw JSON format without any markdown or commentary.

- **Rule Format Specification**

- Base rule format: <SUBJECT> IS <PREDICATE>.
- The predicate defines the behavior or transformation applied to the subject.
- Subjects do not act independently unless a rule explicitly enables behavior.

- **Constraints**

- Coordinate origin is at the top-left corner: (0,0).
- Output must be valid JSON, modifying both `description` and `last_rationale`.
- Only one rule predicate may be updated.

- **Input Variables**

- **Action Performed:** The action taken by the player.
- **Player Position:** The player's location after taking the action.
- **Map Discrepancies:** Observed differences between the expected and actual state.
- **Previous Rules:** Rules that were in place before the action.
- **Expected Rules:** Rules that were predicted or assumed by the agent.
- **Actual Rules:** Rules that were truly active during execution.
- **Rule Predicates:** The list of all possible predicates that can be used in the rule expressions.
- **Current Beliefs:** A mapping of predicates to their assumed behaviors and rationales.

(Optional) **Last Iteration Output:** The previously suggested belief update to refine or revise.

(Optional) **Iteration List:** A list of multiple past belief updates for synthesis in repetition prompting.

- **Output Format**

- The output must follow this JSON structure:

```
{
  "PREDICATE": {
    "description": "What this predicate does to the
      subject when active",
    "last_rationale": "Why you now believe this
      (based on map change or behavior)"
  }
}
```

- No additional text or formatting is allowed—only raw JSON is valid.
- An example belief update is provided for reference.

A.1.2 Full Prompt

Listing 1: Full Prompt of the Critic module using the single (regular) method

```
[INST]
You assist an AI agent solving turn-based puzzle levels. In
this game, objects have no inherent properties. All
behavior is controlled by active rules.

Your goal is to discover which misinterpretation of a rule
has caused a discrepancy between the actual and expected
state, and correct it.

Follow these steps:
1. Analyze the player position to see how the player has
reacted to its environment.
2. Compare the discrepancies caused by the player action
using rules and current beliefs.
3. Identify which rule predicate caused the change. Choose
one from the ‘Rule Predicates’ list in the input section.
4. Update exactly one predicate belief, using this
structure:

{{
  "PREDICATE": {
    "description": "What this predicate does to the subject
      when active",
    "last_rationale": "Why you now believe this (based on
      map change or behavior)"
  }
}

---


### Rule Format
- Structure: ‘<SUBJECT> IS <PREDICATE>’
- Predicate defines behavior, like movement or
transformation.
```

- Subject references the block that the predicate affects.

Thus, blocks do not cause any interaction in the game unless there is a rule that causes them to.

```
### Constraints:
- Origin '(0,0)' is top-left of map.
- Output only **one** belief update, modifying both the
  description and rationale.
- Choose the updated predicate from the 'Rule Predicates'
  list below.
- Use **valid raw JSON only** - no markdown, code blocks,
  or extra text.
- Explain your decision-making step by step in
  'last_rationale'.
```

```
### Input:
```

Action Performed:
{action_performed}

Player Position:
{player_position}

Map Discrepancies:
{map_discrepancies}

Previous Rules:
{previous_rules}

Expected Rules:
{expected_rules}

Actual Rules:
{actual_rules}

Rule Predicates:
{rule_predicates}

Current Beliefs:
{current_beliefs}

```
### Output Examples
Use these examples as format guides:
```

```

{{{
    "SLIP": {{
        "description": "Makes the subject keep moving until
                       blocked",
        "last_rationale": "BABA slid without input after first
                           move"
    }}
}}}

---


Your output:
[/INST]

```

Listing 2: Full Prompt of the Critic module using the feedback method

```

[INST]
You assist an AI agent solving turn-based puzzle levels. In
this game, objects have no inherent properties. All
behavior is controlled by active rules.

Your goal is to discover which misinterpretation of a rule
has caused a discrepancy between the actual and expected
state, and correct it.

This prompt is part of a feedback refinement loop. You are
provided with the belief update from the last iteration
in {last_iteration_output}. Your task is to refine or
improve that belief update to better explain the
discrepancy and clarify the predicate's behavior.

Follow these steps:
1. Review ‘Last Iteration Output’ to understand the
   previous belief and rationale.
2. Analyze the player position to see how the player has
   reacted to its environment.
3. Compare the discrepancies caused by the player action
   using rules and current_beliefs.
4. Decide whether to keep the same predicate or switch to a
   better-fitting one from the ‘Rule Predicates’ list.
5. Output exactly one refined predicate belief, using this
   structure:

```

```

{{{
    "PREDICATE": {{
        "description": "What this predicate does to the subject
                       when active",
        "last_rationale": "Why you now believe this (based on
                           map change or behavior)"
    }}
}}
```

```

    }}

---

#### Rule Format
- Structure: '<SUBJECT> IS <PREDICATE>‘
- Predicate defines behavior, like movement or
  transformation.
- Subject references the block that the predicate affects.

Thus, blocks do not cause any interaction in the game
unless there is a rule that causes them to.

---

#### Constraints:
- Origin ‘(0,0)‘ is top-left of map.
- Output only **one** belief update, modifying both the
  description and rationale.
- Choose the updated predicate from the ‘Rule Predicates‘
  list below.
- Use **valid raw JSON only** - no markdown, code blocks,
  or extra text.
- Explain your decision-making step by step in
  ‘last_rationale‘.

---

#### Input:

Action Performed:
{action_performed}

Player Position:
{player_position}

Map Discrepancies:
{map_discrepancies}

Previous Rules:
{previous_rules}

Expected Rules:
{expected_rules}

Actual Rules:
{actual_rules}

Rule Predicates:
{rule_predicates}

```

```

Current Beliefs:
{current_beliefs}

Last Iteration Output:
{last_iteration}
---

### Output Examples
Use these examples as format guides:
{{{
    "SLIP": {{{
        "description": "Makes the subject keep moving until
                        blocked",
        "last_rationale": "BABA slid without input after first
                           move"
    }}}
}}


---


Your output:
[/INST]

```

Listing 3: Full Prompt of the Critic module using the repetition method

```

[INST]
You assist an AI agent solving turn-based puzzle levels. In
this game, objects have no inherent properties. All
behavior is controlled by active rules.

Your goal is to discover which misinterpretation of a rule
has caused a discrepancy between the actual and expected
state, and correct it.

This prompt is part of a repetition prompting process. You
are now given multiple iterations of previous belief
updates in 'Iteration List'. Your task is to synthesize
these into a single, improved belief update that
reflects the most plausible correction based on
accumulated evidence.

Follow these steps:
1. Review 'Iteration List' to understand how the belief
   updates have evolved.
2. Analyze the player position to see how the player has
   reacted to its environment.
3. Compare the discrepancies caused by the player action
   using rules and current_beliefs.
4. Identify which rule predicate caused the change. Choose

```

one from the ‘Rule Predicates’ list in the input section.

5. Update exactly one predicate belief, using this structure:

```
{{  
    "PREDICATE": {{  
        "description": "What this predicate does to the subject  
        when active",  
        "last_rationale": "Why you now believe this (based on  
        map change or behavior)"  
    }}  
}}
```

Rule Format

- Structure: ‘<SUBJECT> IS <PREDICATE>‘
- Predicate defines behavior, like movement or transformation.
- Subject references the block that the predicate affects.

Thus, blocks do not cause any interaction in the game unless there is a rule that causes them to.

Constraints:

- Origin ‘(0,0)’ is top-left of map.
- Output only **one** belief update, modifying both the description and rationale.
- Choose the updated predicate from the ‘Rule Predicates’ list below.
- Use **valid raw JSON only** - no markdown, code blocks, or extra text.
- Explain your decision-making step by step in ‘last_rationale’.

Input:

Action Performed:
{action_performed}

Player Position:
{player_position}

Map Discrepancies:
{map_discrepancies}

```

Previous Rules:
{previous_rules}

Expected Rules:
{expected_rules}

Actual Rules:
{actual_rules}

Rule Predicates:
{rule_predicates}

Current Beliefs:
{current_beliefs}

Iteration List (Previous Belief Updates):
{iteration_list}

---

### Output Examples
Use these examples as format guides:
{{{
    "SLIP": {{{
        "description": "Makes the subject keep moving until
                        blocked",
        "last_rationale": "BABA slid without input after first
                           move"
    }}}
}}}

---

Your output:
[/INST]

```

A.1.3 Example

Listing 4: Full Prompt example of the Critic module, in the context of the system integration experiment

You assist an AI agent solving turn-based puzzle levels. In this game, objects have no inherent properties. All behavior is controlled by active rules.

Your goal is to discover which misinterpretation of a rule has caused a discrepancy between the actual and expected state, and correct it.

```

Follow these steps:
1. Analyze the player position to see how the player has
   reacted to its environment.
2. Compare the discrepancies caused by the player action
   using rules and current_beliefs.
3. Identify which rule predicate caused the change. Choose
   one from the 'Rule Predicates' list in the input section.
4. Update exactly one predicate belief, using this
   structure:

{
  "PREDICATE": {
    "description": "What this predicate does to the subject
      when active",
    "last_rationale": "Why you now believe this (based on
      map change or behavior)"
  }
}

---

#### Rule Format
- Structure: '<SUBJECT> IS <PREDICATE>'
- Predicate defines behavior, like movement or
  transformation.
- Subject references the block that the predicate affects.

Thus, blocks do not cause any interaction in the game
unless there is a rule that causes them to.

---

#### Constraints:
- Origin '(0,0)' is top-left of map.
- Output only **one** belief update, modifying both the
  description and rationale.
- Choose the updated predicate from the 'Rule Predicates'
  list below.
- Use **valid raw JSON only** - no markdown, code blocks,
  or extra text.
- Explain your decision-making step by step in
  'last_rationale'.

---

#### Input:

Action Performed:
MOVE RIGHT

```

```

Player Position:
BABA (Initial Position: X=15 Y=9):
    Expected Position: X=16 Y=9
    Actual Position: destroyed

Map Discrepancies:
Differences at X=16,Y=9:
- previous: ['LAVA']
- expected: ['BABA', 'LAVA']
- actual: ['LAVA']

Previous Rules:
['BABA IS YOU', 'BABA IS MELT', 'FLAG IS WIN', 'LAVA IS HOT', 'ROCK IS PUSH', 'WALL IS STOP']

Expected Rules:
['BABA IS YOU', 'BABA IS MELT', 'FLAG IS WIN', 'LAVA IS HOT', 'ROCK IS PUSH', 'WALL IS STOP']

Actual Rules:
['BABA IS YOU', 'BABA IS MELT', 'FLAG IS WIN', 'LAVA IS HOT', 'ROCK IS PUSH', 'WALL IS STOP']

Rule Predicates:
{'PUSH', 'STOP', 'MELT', 'HOT'}

Current Beliefs:
{'PUSH': {'description': '', 'last_rationale': ''}, 'STOP':
 {'description': '', 'last_rationale': ''}, 'HOT':
 {'description': '', 'last_rationale': ''}, 'MELT':
 {'description': '', 'last_rationale': ''}}


### Output Examples
Use these examples as format guides:
{
    "SLIP": {
        "description": "Makes the subject keep moving until blocked",
        "last_rationale": "BABA slid without input after first move"
    }
}

Your output:
[/INST]

```

A.2 Strategist Module

A.2.1 Full Prompt Components

- **General Context and Objective**

- The AI is assisting in solving turn-based puzzle levels.
- Objects in the game have no inherent properties; all behavior is controlled by active rules.
- The objective is to generate goals to test whether a belief about a rule predicate is correct.

- **Methodology Instructions**

- Step-by-step procedure to construct test goals:
 1. Analyze the belief to be tested. Determine what the predicate is expected to do and which entities it applies to.
 2. Consider possible rule interactions that could lead to unexpected outcomes, especially involving the player (**YOU**).
 3. Design test cases by defining:
 - * A **description** of the test scenario setup and logic.
 - * An **expected_result** that would confirm the belief if observed.
- Exactly two test goals must be generated.
- Output must be raw JSON—no markdown or additional commentary allowed.

- **Rule Format Specification**

- Rules are expressed as: <SUBJECT> IS <PREDICATE>.
- The predicate defines the behavior applied to the subject (e.g., movement, transformation).
- Subjects do not exhibit any behavior without a corresponding active rule.

- **Constraints**

- Do not refer to specific object names; use generic labels like “an entity with MELT”.
- Clarify which predicates are active in the test scenario and how they affect behavior.
- Only two JSON elements must be returned.
- Output must be valid JSON using generic terms and strict structure.

- **Input Variables**

- **Belief to Test:** A single predicate belief that is being examined.
- **Current Beliefs:** A list of current predicate descriptions.

- **Output Format**

- The output must follow this JSON format with two test cases:

```
[
  {
    "description": "Step-by-step reasoning and
                    setup...",
    "expected_result": "Outcome that should happen
                        if the belief is correct"
  },
  {
    "description": "Another goal using different
                    scenario...",
    "expected_result": "Another expected result"
  }
]
```

- No other text, markdown, or comments are allowed.
- Descriptions must use only generic object references and clearly describe active rules.

A.2.2 Full Prompt

Listing 5: Full Prompt Example of the Strategist module test goal generation, in the context of the integration experiment

```
[INST]
You assist an AI agent solving turn-based puzzle levels. In
this game, objects have no inherent properties. All
behavior is controlled by active rules.

Your task is to generate goals with the objective of
testing if a belief about a rule (predicate) is correct.

### Instructions:

1. Analyze the target belief. Identify what the predicate
   is expected to do and which entities it should interact
   with.
2. Consider scenarios where the predicate might behave
   unexpectedly with other rules, especially including the
   player itself.
3. For each goal, specify:
   - A **description** of the setup and reasoning.
   - The **expected_result** if the belief is correct.
```

```

[
  {{
    "description": "...",
    "expected_result": ...
  }}, ...
]

### Rule Format:

The beliefs are a list of predicate descriptions, where predicates cause behaviors when used in rules:

- Structure: '<SUBJECT> IS <PREDICATE>‘
- Predicate defines behavior, like movement or transformation.
- Subject references the block that the predicate affects.

Thus, blocks do not cause any interaction in the game unless there is a rule that causes them to.

### Output Format:

You must return **exactly two** goals in **raw JSON format only**, no markdown or explanation.

The format must be:
[
  {{
    "description": Step-by-step reasoning and setup, using
      **only generic references** like "a subject with
      PREDICATE" or "an entity with YOU".,
    "expected_result": The outcome that should occur if the
      belief is correct.
  }}, ...
]

### Constraints:
- Avoid referencing specific object names.
- Focus on rules and their interactions: refer to objects only through their assigned predicates.
- Mention clearly which predicates are active and their effects.
- Ensure all output is valid JSON with exactly two elements.

---

### Input:

Belief to Test:

```

```

{belief_to_test}

Current Beliefs:
{current_beliefs}

---

Your output:
[/INST]

```

A.2.3 Example

Listing 6: Full Prompt of the Strategist module to generate test cases for a new belief

```
You assist an AI agent solving turn-based puzzle levels. In
this game, objects have no inherent properties. All
behavior is controlled by active rules.
```

```
Your task is to generate goals with the objective of
testing if a belief about a rule (predicate) is correct.
```

```
### Instructions:
```

1. Analyze the target belief. Identify what the predicate is expected to do and which entities it should interact with.
2. Consider scenarios where the predicate might behave unexpectedly with other rules, especially including the player itself.
3. For each goal, specify:
 - A **description** of the setup and reasoning.
 - The **expected_result** if the belief is correct.

```
[
{
  "description": "...",
  "expected_result": "..."
}, ...
]
```

```
### Rule Format:
```

```
The beliefs are a list of predicate descriptions, where
predicates cause behaviors when used in rules:
```

- Structure: '<SUBJECT> IS <PREDICATE>'
- Predicate defines behavior, like movement or transformation.

- Subject references the block that the predicate affects.

Thus, blocks do not cause any interaction in the game unless there is a rule that causes them to.

Output Format:

You must return **exactly two** goals in **raw JSON format only**, no markdown or explanation.

The format must be:

```
[  
  {  
    "description": Step-by-step reasoning and setup, using  
    **only generic references** like "a subject with  
    PREDICATE" or "an entity with YOU".,  
    "expected_result": The outcome that should occur if the  
    belief is correct.  
  }, ...  
]
```

Constraints:

- Avoid referencing specific object names.
- Focus on rules and their interactions: refer to objects only through their assigned predicates.
- Mention clearly which predicates are active and their effects.
- Ensure all output is valid JSON with exactly two elements.

Input:

Belief to Test:

```
{'name': 'MELT', 'description': 'Makes the subject  
disappear on contact with lava', 'last_rationale': "BABA  
moved right onto lava and disappeared when touching it,  
indicating that the 'BABA IS MELT' rule is active and  
causing BABA to be destroyed by lava."}
```

Current Beliefs:

```
{'BABA': {'description': '', 'last_rationale': ''}, 'PUSH':  
  {'description': 'Makes the subject pushable by the  
player', 'last_rationale': ''}, 'FLAG': {'description':  
  '', 'last_rationale': ''}, 'STOP': {'description':  
  'Makes the subject impassable for the player',  
  'last_rationale': "The player should have moved down to  
(13,12) but remained at (13,11), indicating that the  
'WALL IS STOP' rule is preventing BABA from crossing the  
tile."}, 'ROCK': {'description': '', 'last_rationale': ''}
```

```

    ''}, 'SKULL': {'description': '', 'last_rationale': ''},
    'DEFEAT': {'description': 'Removes the subject from the
game', 'last_rationale': "BABA's expected position is
'destroyed', indicating it has been defeated. This is
because the SKULL is present at the expected position,
and the 'SKULL IS DEFEAT' rule is active."}, 'LAVA':
{'description': '', 'last_rationale': ''}, 'HOT':
{'description': '', 'last_rationale': ''}, 'MELT':
{'description': 'Makes the subject disappear on contact
with lava', 'last_rationale': "BABA moved right onto
lava and disappeared when touching it, indicating that
the 'BABA IS MELT' rule is active and causing BABA to be
destroyed by lava."}, 'WIN': {'description': 'Ends the
game in a WIN when touching the subject.',
'last_rationale': 'Makes the subject block win the game
when touched.'}, 'YOU': {'description': 'Makes the
subject controllable by the player.', 'last_rationale':
"The 'YOU' block represents the player's character,
which is the primary agent for interacting with the game
world."}}

```

```

Your output:
[/INST]

```

B Results

B.1 Critic Module Outcome Examples

In this section we will present some *Critic* output examples from the case where *ROCK IS LAVA* is formed, turning all the *ROCKs* in the map to *LAVA*.

Listing 7: Raw Output of the *Critic* module for *ROCK IS LAVA* test case, with a human-score of 1

```

"result": {
    "LAVA": {
        "description": "Turns rock into lava when adjacent",
        "last_rationale": "Rock at (14,5) turned into lava
                        after BABA moved left, indicating that 'LAVA IS
                        HOT' and 'ROCK IS LAVA' rules are active and
                        'ROCK IS PUSH' is not."
    }
}

```

Listing 8: Raw Output of the *Critic* module for *ROCK IS LAVA* test case, with a human-score of 4

```

"result": {
    "LAVA": {
        "description": "Turns rocks into lava",
        "last_rationale": "A rock at (14,5) turned into
                         lava after BABA moved left, indicating that the
                         'ROCK IS LAVA' rule is active."
    }
}

```

Listing 9: Raw Output of the *Critic* module for *ROCK IS LAVA* test case, with a human-score of 10

```

"result": {
    "LAVA": {
        "description": "Turns the subject into lava",
        "last_rationale": "A rock at (14,5) turned into
                         lava after forming 'ROCK IS LAVA', indicating
                         that the 'ROCK IS LAVA' rule is active and
                         transforming rocks."
    }
}

```

B.2 Strategist Module Outcome Examples

In this section we will present some *Strategist* output examples, where the *MELT* rule has been sent to test, with its updated (and incorrect) meaning being "Subject is destroyed when in contact with LAVA".

Listing 10: Raw Output of the *Strategist* module for *MELT* goal generation test case, with a human-score of 1

```

"result": [
    {
        "description": "Create a scenario where a subject
                       with MELT is between the player and a subject
                       with WIN. If the MELT rule is active, the player
                       should not be able to reach the WIN subject,
                       resulting in a game over.",
        "expected_result": "The player is unable to reach
                           the WIN subject, and the game ends."
    },
]

```

Listing 11: Raw Output of the *Strategist* module for *MELT* goal generation test case, with a human-score of 6

```

"result": [
    {
        "description": "Place an entity with MELT on a tile
                       next to lava. Move the entity onto the lava
                       tile."
    },
]

```

```

        "expected_result": "The entity with MELT should
                           disappear upon contact with the lava."
    },
]

```

Listing 12: Raw Output of the *Strategist* module for *MELT* goal generation test case, with a human-score of 10

```

"result": [
{
    "description": "Place a subject with MELT next to a
                   subject with HOT that its not lava. Move the
                   MELT subject over the HOT subject.",
    "expected_result": "The MELT subject disappears."
},
]

```

B.3 System Integration Outcome Examples

The system integration experiment showcases the collaboration of the two main modules developed, *Critic* and *Strategist*, in the context of the level *Volcano*. We have defined 3 different outcomes for each experiment run: (1) The *Critic* is able to induce the meaning of the rule *MELT* in the first run, without the need of the *Strategist* (2) After a wrong/partially correct induction, the *Strategist* aids the *Critic*, resulting in a second and now correct induction (3) The correct meaning could not be induced. We will be giving a example for each case.

Listing 13: Raw Output of the *Critic* module for test case (1)

```

"result": {
    "MELT": {
        "description": "Causes the subject to be destroyed
                       on contact with HOT objects",
        "last_rationale": "BABA was expected to move right
                          onto the LAVA, but was destroyed instead. This
                          indicates that the 'BABA IS MELT' rule is
                          active, causing BABA to be destroyed when in
                          contact with LAVA."
    }
}

```

Listing 14: Raw Output of the *Strategist* module for test case (2), resulting in a new induction cycle with a result similar to (1)

```

"result": [
{
    "description": "Place an entity with MELT next to
                   an entity with HOT. If the belief is correct,
                   the entity should disappear upon contact with
                   HOT."
}
]

```

```

        "expected_result": "Entity with MELT disappears"
    },
]

```

Listing 15: Raw Output of the *Critic+Strategist* modules for test case (3), resulting in an incomplete/incorrect understanding of the *MELT* property

```

"result_critic": {
    "MELT": {
        "description": "Makes the subject dissolve when in
                        contact with LAVA",
        "last_rationale": "BABA was expected to move right
                            onto a LAVA tile but instead dissolved,
                            indicating that the 'MELT' rule is active."
    }
}

"result_strategist": [
{
    "description": "Place a 'MELT' subject next to an
                    entity with 'LAVA'. Then, move the 'MELT'
                    subject to touch the 'LAVA'.",
    "expected_result": "The 'MELT' subject should
                        disappear upon touching the 'LAVA'."
},
]

```

C Time Planning

This project will be ongoing from the 10th of February until the 25th of June—the earliest day in which the thesis defence can occur. Thus, 141 days on which approximately 500 hours of work will be spent. The aim is to dedicate 4 hours on a daily basis (excluding weekends and festivities), although work problems and personal matters could alter this plan.

C.1 Task Description

This section defines the individual tasks and the resources needed for carrying out the project. The project is structured into several major tasks, each broken down into specific subtasks that build on one another. Refer to 2 for a summary of the defined tasks and associated resources.

C.1.1 Task Definition

Project Management (PM) Project Management encompasses the overall planning and coordination of the project. Its subtasks include:

- **Context and Scope (PM-1):** Define the project context, scope, and objectives, ensuring a clear understanding of the problem and its significance.
- **Time Planning (PM-2):** Develop a detailed schedule describing tasks, outlining milestones and planning adverse situation management.
- **Budget and Sustainability (PM-3):** Establish a budget and sustainability plan for managing resources throughout the project.
- **Final Project (PM-4):** Consolidate and refine all project deliverables to meet the required standards.
- **Meetings (PM-5):** Organize regular meetings to monitor progress, address challenges, and adjust plans as necessary.

Testing Environment (TE) The Testing Environment task focuses on creating a simulated version of *Baba Is You* to serve as a platform for evaluating the cognitive architecture. Its subtasks include:

- **Gather and Analyze Existing Environments (TE-1):** Research and assess current game replicas and environments related to *Baba Is You*.
- **Research Game Logic (TE-2):** Investigate the underlying game mechanics and rule systems to replicate essential features.
- **Develop Game (TE-3):** Build a simplified version of the game that supports dynamic rule changes.
- **Design API (TE-4):** Develop an API that facilitates communication between the game environment and the cognitive architecture.

Cognitive Architecture (CA) This task is dedicated to designing and implementing the intelligent agent. It comprises the following subtasks:

- **Research in Cognitive Architectures (CA-1):** Review current cognitive architectures and relevant literature to gather foundational concepts.
- **Research in AI Models (CA-2):** Investigate AI models effective in rule induction and decision-making under uncertainty.
- **Model the Cognitive Architecture (CA-3):** Design a tailored framework that integrates insights from both cognitive architectures and specialized AI models.
- **Implement the Cognitive Architecture (CA-4):** Code the designed architecture, ensuring seamless integration with the testing environment.

Experimentation and Analysis (EA) This task involves validating the system through testing and evaluating its performance. It is broken down into:

- **Implement Experimentation Scripts and Test (EA-1):** Develop and execute scripts that systematically test the integrated system.
- **Analyze Results and Draw Conclusions (EA-2):** Evaluate experimental data to assess the system's effectiveness and derive insights for improvements.

Documentation (DC) Documentation involves compiling a comprehensive record of the entire project: record research findings, development processes, experiment results and analysis in a structured document. The development will require of exhaustive refining and re-writing of the documentation to finally achieve a polished result.

Thesis Defense Preparation (TD) This final task focuses on preparing the presentation and materials for the thesis defense, including the development of a presentation and ensuring a quality talk.

C.1.2 Resources

This section outlines the resources required to successfully execute the project, divided into four categories.

Human Resources The project is supported by a focused team that ensures comprehensive oversight and guidance throughout the research process. The roles are defined as follows:

- **Researcher (R):** Responsible for designing, implementing, experimenting and writing the research tasks and the paper.
- **Thesis Tutor (T):** Provides mentorship and feedback on the project and paper. The thesis tutor is also responsible for guiding the journey of the work, ensuring the researcher does not deviate from the course with unrealistic or useless objectives.
- **Project Management Tutor (GT):** Offers expertise and advice on the project planning. Feedback is also provided after the assignment deliveries with the purpose of refining the work done.

Hardware Resources A personal computer is the primary hardware resource, supplemented by peripherals (e.g., external storage and additional monitors) with all the equipment needed for a reliable internet connection.

Software Resources The software tools utilized in this project include:

- **Development Tools:** Integrated Development Environments (IDEs), version control systems (Git), and code editors.

- **Project Management Tools:** Applications such as TeamGantt for planning and tracking tasks.
- **Documentation Platforms:** Overleaf for preparing the thesis document in LaTeX.
- **Collaboration Tools:** GitHub for repository management and team collaboration.

Material Resources Material resources comprise academic papers, textbooks, technical manuals, and online resources that provide theoretical and practical insights related to cognitive architectures, game design, and artificial intelligence.

C.1.3 Task Summary

The table in 2 summarizes the tasks with their identifiers, estimated time in hours, dependencies, and associated resources, providing a concise overview of the work breakdown structure.

ID	Task	Time(h)	Dependencies	Resources
PM	Project Management	60		
PM-1	Context and Scope	25		R, GT, PC, overleaf, papers, books
PM-2	Time Planning	7.5	PM1	R, GT, PC, overleaf, ganttproject
PM-3	Budget and Sustainability	7.5	PM2	R, GT, PC, overleaf
PM-4	Final Project	10	PM3	R, GT, PC, overleaf
PM-5	Meetings	10		R, T, PC
TE	Testing Environment	80		
TE-1	Gather and analyze existing environments	10		R, PC, github
TE-2	Research game logic	10		R, PC, github
TE-3	Design and implement game	50	TE-1, TE-2	R, PC, IDE, git
TE-4	Design API	10	TE-3	R, PC, IDE, git
CA	Cognitive Architecture	170		
CA-1	Research in Cognitive Architectures	45		R, PC, papers, books
CA-2	Research in AI models	25		R, PC, papers, books
CA-3	Model the Cognitive Architecture	60	CA-1	R, T
CA-4	Implement the Cognitive Architecture	40	CA-2, CA-3	R, T, PC, IDE, git
EA	Experimentation and Analysis	90		
EA-1	Implement experimentation scripts and test	45	TE, CA	R, PC, IDE, git
EA-2	Analyze results and draw conclusions	45	EA-1	R, T, PC
DC	Documentation	70	EA	R, T, PC, overleaf, papers, books
TD	Thesis defense preparation	10	DC	R, PC
Total		500		

Table 2: Task Information Summary (Authored by me)

C.2 Task Estimation

The task estimation process has been carried analyzing the concurrence and duration over-time of each task, based on assumptions from similar and own projects. These estimates are reflected in the Gantt chart in 11, which visually represents the entire project timeline, the concurrence and dependency of the tasks.

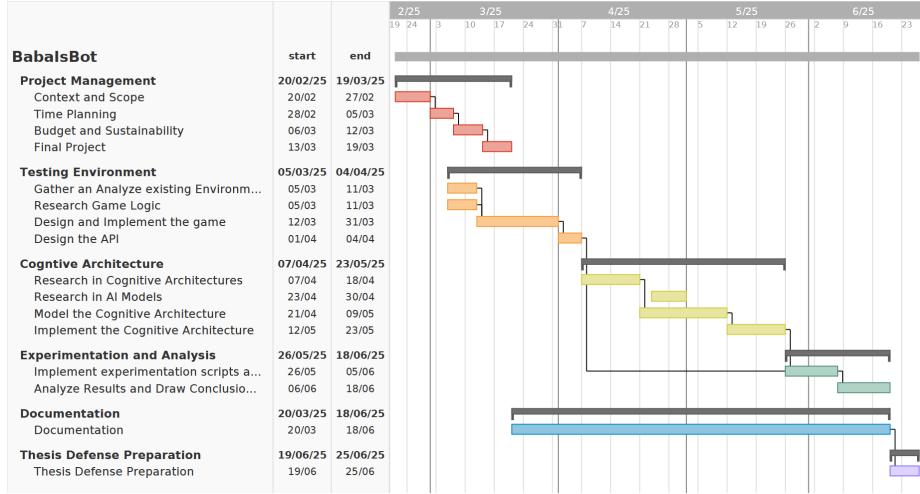


Figure 11: Gantt chart detailing the project timeline, task durations and their dependencies (Authored by me)

Key aspects of the estimation include:

- **Dependencies:** Each task's schedule has been determined by considering its prerequisites and the logical flow of the project. This mapping of dependencies ensures a clear direction of the project and helps in identifying potential bottlenecks.
- **Realistic Concurrence:** The Gantt chart shows overlapping tasks only where resource availability and task dependencies allow. This concurrency is planned to optimize workflow without compromising quality.
- **Consistency and Transparency:** The estimation process aligns with the explanations provided in this document. Each task's time allocation is consistent with its complexity and the required level of detail.

C.3 Risk Management

Effective risk management is crucial to ensure that potential obstacles do not derail the project timeline. The following risks have been identified along with proposed alternative tasks and its anticipated impact on the project:

- **Limited Time:** If the project timeline becomes constrained, tasks will be reprioritized and streamlined. Alternative strategies include reducing non-critical scope and reallocating resources to focus on high-impact areas. Alternatively, if the amount of work is still overwhelming more humble goals will be established.

- **Inexperience in AI Models:** The research and implementation of advanced AI models can be challenging due to a lack of prior expertise. To mitigate this risk, the project's preliminary research could be extended. This approach may extend the initial phase slightly but will ensure that later tasks proceed with a solid understanding of the required AI models.
- **Adversities in Testing Environment Setup:** Technical difficulties in establishing a robust testing environment can delay progress. If significant delays occur, contingency plans such as adopting a simplified prototype or leveraging community resources (e.g., GitHub projects) will be activated.

Overall, the risk management strategy involves continuous monitoring of potential issues and prompt adjustments to the project plan. By planning alternative tasks and reallocating resources as needed, the project is designed to remain on schedule and deliver its objectives successfully.

D Budget

D.1 Costs

The overall project costs have been organized into three main categories: Staff Costs, Generic Costs, and Deviation Costs. These costs have been estimated on the basis of current market data in Barcelona / Spain and personal expenditures:

Cost Category	Total Cost (€)	Notes
Staff Costs	19,444.53	See 4
Generic Costs	930.41	See 5
Deviation Costs	2,537.49	See 6
Grand Total	22,912.44	

Table 3: Overall Project Costs (Authored by me)

D.1.1 Staff Costs

The following table summarizes the different staff roles involved in the project along with a brief description and their respective price per hour:

- **Project Manager (PM):** Oversees and coordinates projects from initiation to completion. They plan schedules, manage resources, monitor progress, and ensure the project meets its goals on time and within budget. Estimated hourly wage: **€20.21 per hour[?]**, approximately in Barcelona, Spain.
- **Software Engineer (SE):** Designs, develops, and maintains software systems. They apply engineering principles to write code, debug programs, and build applications that solve specific problems or meet user

needs. Estimated hourly wage: **€22.87 per hour**[?], approximately in Barcelona, Spain.

- **AI Researcher (AR):** Investigates and models new methods and algorithms in artificial intelligence. They work on advancing AI technologies through research, experimentation, and collaboration, often aiming to solve complex problems with machine learning and data analysis. Estimated hourly wage: **€25 per hour**[?], approximately in Barcelona, Spain.

Each project task and its subtasks are assigned to specific roles, with total prices calculated using the hours dedicated to each task, including the social security extra costs for the company, which are 35% in Spain. The table below lists each task (and subtask) with the assigned role, number of hours, and the total price (calculated as *hours* × *price/hour* × 1.35).

ID	Task	Time (h)	Role	Price (€)
PM	Project Management	60	PM	$60 \times 27.28 \approx 1,637.01$
PM-1	Context and Scope	25	PM	$25 \times 27.28 \approx 682.09$
PM-2	Time Planning	7.5	PM	$7.5 \times 27.28 \approx 204.63$
PM-3	Budget and Sustainability	7.5	PM	$7.5 \times 27.28 \approx 204.63$
PM-4	Final Project	10	PM	$10 \times 27.28 \approx 272.84$
PM-5	Meetings	10	PM	$10 \times 27.28 \approx 272.84$
TE	Testing Environment	80	SE	$80 \times 30.87 \approx 2,469.96$
TE-1	Gather and analyze existing environments	10	SE	$10 \times 30.87 \approx 308.74$
TE-2	Research game logic	10	SE	$10 \times 30.87 \approx 308.74$
TE-3	Design and implement game	50	SE	$50 \times 30.87 \approx 1,543.73$
TE-4	Design API	10	SE	$10 \times 30.87 \approx 308.74$
CA-1	Research in Cognitive Architectures	45	AR	$45 \times 33.75 = 1,518.75$
CA-2	Research in AI models	25	AR	$25 \times 33.75 = 843.75$
CA-3	Model the Cognitive Architecture	60	AR	$60 \times 33.75 = 2,025.00$
CA-4	Implement the Cognitive Architecture	40	SE	$40 \times 30.87 \approx 1,234.98$
EA-1	Implement experimentation scripts and test	45	SE	$45 \times 30.87 \approx 1,389.35$
EA-2	Analyze results and draw conclusions	45	AR	$45 \times 33.75 = 1,518.75$
DC	Documentation	70	AR	$70 \times 33.75 = 2,362.50$
TD	Thesis defense preparation	10	AR	$10 \times 33.75 = 337.50$
Total		480		19,444.53

Table 4: Staff Costs per Task and Subtask (Authored by me)

D.1.2 Software Costs

For this project, the following software tools were used:

- **JetBrains CLion:** A powerful integrated development environment (IDE) for C and C++ that provides advanced code analysis, debugging, and refactoring capabilities.
- **Visual Studio Code (VSCode):** A lightweight, open-source code editor known for its extensive extensions ecosystem, supporting many programming languages and tools.

- **GanttProject:** A free, open-source project management tool that helps in planning and scheduling tasks using Gantt charts.
- **Overleaf:** An online LaTeX editor that enables collaborative writing and real-time document compilation, making it easier to create professional documents.
- **GitHub:** A web-based platform for version control and collaboration that allows for easy sharing, reviewing, and management of code through Git repositories.

All of these software tools are available free of charge—either as open-source products or through free student licenses.

D.1.3 Generic Costs

Generic costs include recurring expenses and equipment amortization over the project period of 141 days (approximately 4.7 months). However, only a fraction of the flat expenses (rent, electricity, and internet) is allocated to the project. Based on an estimated usage of 17% (calculated as $\frac{4\text{hours per day}}{24\text{hours per day}}$), only 17% of these monthly costs is attributed to the project:

- **Rent:** Full monthly rent is 800 €/month. Allocating 17% gives an effective cost of $800 \times 0.17 = 136$ per month.
- **Electricity:** With a full cost of 100 €/month, 17% corresponds to $100 \times 0.17 = 17$ per month.
- **Internet:** For an internet cost of 39 €/month, 17% is $39 \times 0.17 \approx 6.63$ per month.

Equipment amortization is fully allocated to the project and calculated as:

$$\text{Amortization} = \frac{\text{Cost of asset}}{\text{Useful life}}$$

- For the **computer** (2000 € over 5 years):

$$\frac{2000}{5 \text{ years}} \times \frac{1 \text{ year}}{12 \text{ months}} \approx 33.33 / \text{month}$$

- For the **monitor** (300 € over 5 years):

$$\frac{300}{5 \text{ years}} \times \frac{1 \text{ year}}{12 \text{ months}} = 5 / \text{month}$$

Multiplying these monthly costs by 4.7 months yields:

Item	Adjusted Unit Cost (€)	Duration (months)	Total Cost (€)
Rent	136.00	4.7	$136.00 \times 4.7 \approx 639.20$
Electricity	17.00	4.7	$17.00 \times 4.7 \approx 79.90$
Internet	6.63	4.7	$6.63 \times 4.7 \approx 31.16$
Computer Amortization	33.33	4.7	$33.33 \times 4.7 \approx 156.65$
Monitor Amortization	5.00	4.7	$5.00 \times 4.7 = 23.50$
Total Generic Costs			930.41

Table 5: Generic Costs Proration (Authored by me)

D.1.4 Deviation Costs

Deviation costs cover unforeseen expenses during the project. These include:

- **Contingency:** A 10% contingency is applied to the total estimated base cost. In this case, this applies to the staff costs (see 4) and generic costs (see 5). Therefore, the contingency is:

$$10\% \times (\text{Staff Costs} + \text{Generic Costs}) = 10\% \times (19,444.53 + 930.41) = 2,037.49$$

This amount is reserved for unexpected increases in labor costs.

- **Incident Costs:** These are estimated to cover:

- **Integration Challenges:** Estimated at 300.00 € for additional time and resources required for integration.
- **Limited-Time Incidents:** A buffer of 100.00 € for addressing time-sensitive issues.
- **Unpredictable Game Complexity:** An extra allocation of 100.00 € to manage unforeseen complexities.

The total incident costs are:

$$300.00 + 100.00 + 100.00 = 500.00$$

Deviation Item	Estimated Cost (€)	Notes
Contingency (10% of Base Costs)	2,037.49	Based on a total staff cost of 20,374.95 €
Integration Challenges	300.00	Estimated additional cost for integration difficulties
Limited-Time Incidents	100.00	Buffer for time-sensitive issues
Unpredictable Game Complexity	100.00	Extra allocation for unforeseen complexities
Total Incident Costs	500.00	Sum of individual incident costs
Total Deviation Costs	2,537.49	$2,037.49 + 500.00$

Table 6: Deviation Costs Breakdown (Authored by me)

D.2 Management Control

In this section, we describe the procedures and control mechanisms implemented to manage potential budget deviations during project execution. By comparing estimated and real costs on a regular basis, these mechanisms enable early detection and corrective action if deviations occur. The control indicators defined below facilitate detailed monitoring of cost components.

Human Resources Cost Deviation A key indicator is the deviation in human resource costs. This is determined by comparing the estimated cost per hour with the real cost per hour, multiplied by the total hours consumed for each task:

$$\text{Human Resources Deviation} = (\text{Estimated Cost per Hour} - \text{Real Cost per Hour}) \times \text{Total Hours Consumed} \quad (1)$$

This formula allows us to assess the efficiency of each task in the Gantt diagram and adjust allocations as necessary.

Hardware Amortization Deviation For hardware, we monitor deviations in the amortization process. The amortization deviation is calculated by measuring the difference between the estimated usage hours and the real usage hours, then multiplying by the amortization rate (price per hour):

$$\text{Amortization Deviation} = (\text{Estimated Usage Hours} - \text{Real Usage Hours}) \times \text{Price per Hour} \quad (2)$$

This indicator helps ensure that the usage of depreciable assets such as computers and monitors remains within budgeted limits.

Electricity Cost Deviation Although space and transport costs are fixed, electricity represents a variable expense. Its deviation is monitored as:

$$\text{Electricity Cost Deviation} = (\text{Estimated Usage Hours} - \text{Real Usage Hours}) \times \text{Electricity Price per Hour} \quad (3)$$

By tracking this deviation, we can quickly identify any unexpected increases in electricity consumption.

Incidental Cost Deviation Incidental costs are prone to higher variations if the project deviates from the planned schedule. The deviation for these costs is given by:

$$\text{Incidental Cost Deviation} = (\text{Estimated Incidental Hours} - \text{Real Incidental Hours}) \times \text{Total Incidental Cost per Hour} \quad (4)$$

Monitoring this indicator ensures that unplanned expenses are kept under control.

Overall Budget Deviation To obtain a complete overview of the project's cost performance, we sum the individual deviations:

$$\text{Total Cost Deviation} = \text{Human Resources Deviation} + \text{Amortization Deviation} + \text{Electricity Cost Deviation} + \text{Incidental Cost Deviation} \quad (5)$$

By comparing the total estimated cost against the real cost (adjusted for deviations), we can assess whether the contingency margin is sufficient and determine if corrective measures are needed.

E Sustainability

E.1 Self-Assesment

The survey³ helped me understand just how much I still have to learn about sustainability. As a student of computer science and AI, I realized that while I'm confident with technology, my grasp of sustainability's social, environmental, and economic aspects need to grow.

On the social side, the survey showed me that sustainability isn't only about technical achievements—it's also about building systems that support fairness and well-being for all. In AI, this means paying close attention to issues like bias, transparency and the overall impact of my work on society.

When it comes to the environment, the questionnaire made it clear that even technical fields like mine have a role to play. There is a great opportunity to reduce energy use in data centers, create more efficient models and support green technology. By focusing on these areas, my work in AI could help tackle larger issues such as climate change and sustainable resource management, making our technological solutions more environmentally friendly.

Economically, the survey emphasized that sustainable development means balancing growth with responsible resource management. My research could drive innovation and economic value, but it must also take into account long-term stability. Adopting sustainable practices in technology lead to creating economic systems that are both strong and fair.

In summary, the survey was a valuable realization of the areas where I need to expand my knowledge. It motivated me to explore sustainability more deeply and to find ways to integrate these principles into my work.

³https://docs.google.com/forms/d/e/1FAIpQLSfVgBxcxZfh7pB_OVRUNGQmRpFDFlhAskuNcpQBowlRF4-sA/viewform

E.2 Economic Dimension

Reflection on the cost you have estimated for the completion of the project The project cost, estimated at approximately 17,612.1€, reflects a generous allocation for all the project expenditures, while maintaining a competitive and efficient cost profile.

How are currently solved economic issues (costs...) related to the problem that you want to address (state of the art)?, and How will your solution improve economic issues (costs ...) with respect other existing solutions? While my project is aimed at giving a SOTA solution to the problem of solving Baba Is You, the current solutions use brute-force or reinforcement learning techniques which could not be considered SOTA due to their lack of efficiency and generalization. The lack of (SOTA) solutions in this matter makes it impossible to fairly compare the economic issues improvements caused by my solution.

E.3 Environmental Dimension

Have you estimated the environmental impact of the project? Although primarily a digital project, the environmental impact has been assessed by considering energy consumption related to computing resources and equipment usage over the project duration.

Did you plan to minimize its impact, for example, by reusing resources? To reduce environmental impact, the project plans to reuse existing hardware, implement energy-efficient algorithms, and optimize software performance. These measures aim to minimize additional energy consumption and electronic waste.

How is currently solved the problem that you want to address (state of the art)?, and How will your solution improve the environment with respect other existing solutions? Many current solutions do not address environmental concerns by using high time and energy-consuming algorithms such as brute-forcing or reinforcement learning. My solution will try to minimize the environmental effects by using low consumption algorithms.

E.4 Social Dimension

What do you think you will achieve -in terms of personal growth-from doing this project? I hope my knowledge of software development and, especially, of the AI fields touched will improve considerably. Above all, I expect to learn how to do scientific research on a topic of computer science / AI

and all that it involves. In terms of soft skills, organization, project management and discipline are surely going to improve during the project.

How is currently solved the problem that you want to address (state of the art)?, and ... How will your solution improve the quality of life (social dimension) with respect other existing solutions? While my project is aimed at giving a SOTA solution to the problem of solving Baba Is You, the current solutions use brute-force or reinforcement learning techniques which could not be considered SOTA due to their lack of efficiency and generalization. If we focus instead on the general problem of decision-making in complex systems with dynamic rules, currently there are no SOTA solutions that use AI agents with cognitive architectures. The lack of (SOTA) solutions in this matter makes it impossible to compare the quality of life improvements caused by my solution.

Is there a real need for the project? Making a bot to solve a game does not seem really useful, but the particularities of Baba Is You and the proposed SOTA solution makes it valuable as it would (hopefully) expand the knowledge on cognitive architectures and AI agents.