Artificial Intelligence Research and Development
T. Alsinet et al. (Eds.)

doi:10.3233/FAIA240443

LLM Reasoner and Automated Planner: A New NPC Approach

Israel PUERTA MERINO [a,1], and Jordi SABATER-MIR [a]
[a] *IIIA, CSIC, Campus UAB, E-08193. Bellaterra, Catalonia (Spain)*

Abstract. In domains requiring intelligent agents to emulate plausible human-like behavior, such as formative simulations, traditional techniques like behavior trees encounter significant challenges. Large Language Models (LLMs), despite not always yielding optimal solutions, usually offer plausible and human-like responses to a given problem. In this paper, we exploit this capability and propose a novel architecture that integrates an LLM for decision-making with a classical automated planner that can generate sound plans for that decision. The combination aims to equip an agent with the ability to make decisions in various situations, even if they were not anticipated during the design phase.

Keywords. Intelligent agents, Serious Games, Formative agent-based simulations.

1. Introduction

Intelligent agents are required to emulate plausible human-like behaviour in multiple domains, such as serious games or formative simulations. Within this context, Non-Playable Characters (NPCs) usually face the challenge of making context-based decisions that must appear plausible and coherent for the human observer. Classical approaches to this problem, such as behaviour trees, encounter limitations. These methods require developers to anticipate and manually specify the actions and decision-making flow for every potential scenario, a task that is tedious, prone to errors and often hindered by the complexity of the simulated world. In response to these limitations, it would be desirable to have an architecture capable of making contextually appropriate decisions without exhaustive scenario specification.

Large Language Models (LLMs) with recent papers such as Generative Agents [1], have demonstrated an interesting potential in decision-making. Despite not always yielding optimal solutions, they usually provide plausible and human-like responses to given problems. Thus, this could be the path to develop a powerful reasoning system for these scenarios where optimal processing is unnecessary but a flexible model capable of giving plausible and human-like responses is desired.

Therefore, to exploit this capability, we propose a novel Agent architecture that integrates an LLM for decision-making with a classical Automated Planning (AP) algorithm to generate sound plans to achieve the decisions. This combination aims to empower an autonomous agent with the flexibility to evolve in any situation, even those unforeseen by the developer, while maintaining a plausible and human-like behavior.

[1] Corresponding Author: Israel Puerta Merino; E-mail: israelpm01@hotmail.com

2. System Description

The intelligent system developed employs an LLM to make the environment-based decision of which goal to follow. Subsequently, it utilizes the AP algorithm to devise an executable plan, a sequential list of specific actions to achieve the selected goal. As depicted in Figure 1, the general architecture of this Intelligent Agent is based on three main modules: the Reasoner module, which utilizes the LLM to generate the goal; the Planner module, which employs AP techniques to generate the plan; and the Interface with the environment. Each module is briefly explained below:

- **Reasoner**. This module features a main data structure, called *memories*, representing the environmental context. These memories comprise the list of the pertinent perceptions the agent has encountered, presented in natural language format for the LLM to comprehend.
  The module receives a list of potential goals to pursue and, leveraging the environment context provided by the memories, determines which one to pursue.
  To enhance the model's capacity to address the reasoning process, we introduce an additional field in the Reasoner's memories, called the *personality traits* field. This field serves as an initial phrase, also expressed in natural language, where aspects such as the agent's role in the environment or its priorities are specified.
- **Planner**. This module features a world-state representation called AP Problem, which employs a data structure compatible with the AP algorithm.
  It receives the selected goal and generates a plan, a sequence of actions to achieve it in the current context (defined on the AP Problem).
- **Interface**. This module receives a new instance of the environment state and utilizes it to generate all the information required by the other modules. It is responsible for processing all the world-state information to generate (1) the possible

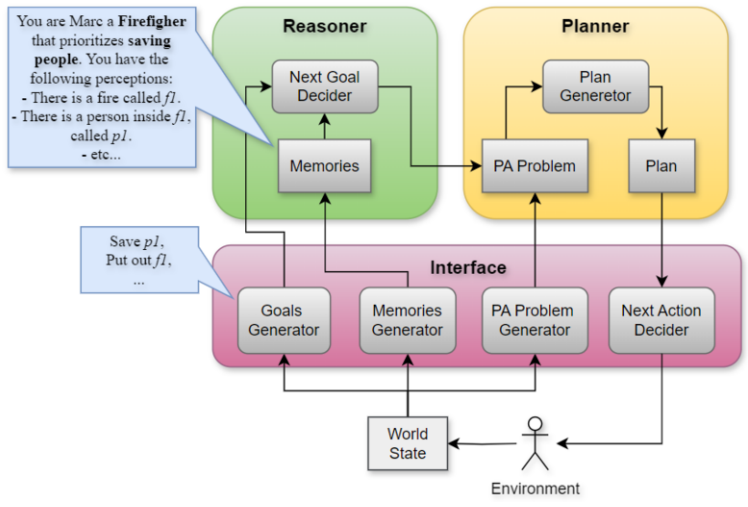

Figure 1. General architecture of the intelligent agent designed and implemented.

goals that the agent could achieve in that specific situation,[2] (2) the new perceptions to add to the Reasoner's memories, and (3) the AP Problem representation of the received world state. Once the plan is generated, the interface sequentially gives the actions to the environment to be executed.

The execution of the system consist on an iterative and reactive process where, on each iteration, the system exhibits the following behavior:

1. The iteration commences when the Interface module receives a new instance of the world-state. It utilizes the world-state information to generate all possible goals.
2. The Interface module generates natural language perceptions of the world-state and compares them with the Reasoner's memories, adding all the new ones. Subsequently, it constructs the AP Problem representation of the new world-state.
3. If any new perception has been added to the memories, the Reasoner module re-evaluates the goal to achieve using the updated memories list and the goals generated by the interface.
4. If the generated AP Problem differs from the previous one or the selected goal has changed, the Planner module generates a new plan.
5. If the plan has changed or the agent has completed the previously selected action, the interface takes the action at the top of the plan, removes it from the list and sends it to the environment for execution.

3. Use Case

To test the presented architecture, we implemented it in a simple use case to illustrate its resulting behaviour. This scenario comprises three main elements: a burning car with a person inside, a fire extinguisher and a safe zone. In this context, the intelligent agent will be introduced, with different personality traits, and we expect them to behave intuitively. For example, if the agent is *a firefighter who prioritizes saving people*, we expect it to save the person inside the car over putting out the fire.

We used five different configurations for the agent: the person inside the car (PI), an external common person (PO), a firefighter who prioritizes saving people (FP), a firefighter who prioritizes putting out fires (FF) and a paramedic (PA). We evaluated the behaviour of these agents in different situations, starting with a single agent and expanding it up to four simultaneously, to observe its performance in a multi-agent context.[3] All the testing scenarios are listed in Table 1.

After conducting the experiments, we have confirmed that the developed intelligent agent exhibits a plausible and human-like behaviour: the agent consistently attempts to rescue the person trapped inside the car or provide as much assistance as possible. We consider this outcome a significant success, demonstrating the agent's ability to au-

[2] The method to generate the possible goals depends on the specific environment and its implementation. In our *Use Case*, we use an independent module for goal generation. This module utilizes a range of general goal classes, which receive the environment state and generate a list of achievable goals accordingly.

[3] For these initial experiments, we implemented the architecture using the Rhymas framework [2] and executed the simulation in Unreal Engine. The AP technology utilized was the Python library Unified Planning. Additionally, we employed the platform LM Studio to execute the LLM, running a small 7B Mistral model due to computational limitations.

	Scenarios				
One Agent	CI	CO	FP	FF	PA
Two Agents	FP and FF	FP and PA	CI and FP		
Four Agents	CI, FP, FF and PA				

Table 1. Summary of all the scenarios used to test the behaviour of the developed architecture.

tonomously select and pursue goals based on the specific context without specifying explicit decision-making instructions.

Ideally, users should be able to control the reasoning logic and priorities of the NPC by specifying individual characteristics in its personality traits. This mechanism generally works well, observing changes in the NPC's decision tendencies based on their personality. However, full control over the agent's decisions remains elusive, so these traits can only generate some propensity. This can be evidenced in the disparity between the two firefighters: the one prioritizing saving people **always** does so, while the one prioritizing putting out fires does so **most** of the time but occasionally decides to save the person instead.

Even so, we have observed that the agent always acts consistently. For example, considering the firefighter whose main duty is to save people in a context where there is only fire, with no one in danger, the agent will try to extinguish the fire, as expected. Thus, we may conclude that the agent acts based on personality traits and some common sense. This latent common sense explains why the agent is not always faithful to its personality traits, and it makes sense because the LLM's decision-making ability is based on what it has learned in its training, which should not be assumed to be less crucial than the prompt provided to make the reasoning.

4. Conclusions

In this work, we have explored the design and implementation of a new intelligent agent proposal, utilizing the decision-making capacity of LLMs to select the desired goals and the power of AP to generate plans to achieve them dynamically. As illustrated in our initial simple experiments, this architecture yields promising results: despite using a limited LLM due to computational constraints, the NPCs consistently exhibit the expected plausible human-like behaviour, indicating a fruitful avenue for further exploration. In future studies, it would be valuable to investigate the agent's performance with more powerful LLMs, which could potentially mitigate the fidelity issues described. Moreover, exploring the combination of this technique with other classical methods could lead to developing a reasoning model that leverages the strengths of both technologies.

References

[1] Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., Bernstein, M. S. Generative agents: Interactive simulacra of human behavior. In: Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology. Association for Computing Machinery; 2023, Oct. p. 1-22, doi: 10.1145/3586183.3606763

[2] Sabater-Mir J., Camps-Ortin I., Cozar-Alier C. An Agent-Based Simulation Framework for Firefighters. In: Proceedings of the 24th International Conference of the Catalan Association for Artificial Intelligence. IOS Press; 2002, Oct. p. 160-163, doi: 10.3233/FAIA220332