# Non-Player Character Decision-Making With Prolog and Ontologies

Sylvain Lapeyrade, Christophe Rey

# Non-Player Character Decision-Making With Prolog and Ontologies

Sylvain Lapeyrade
Université Clermont Auvergne, CNRS,
Clermont Auvergne INP, Mines Saint-Etienne, LIMOS
sylvain.lapeyrade@uca.fr

Christophe Rey
Université Clermont Auvergne, CNRS,
Clermont Auvergne INP, Mines Saint-Etienne, LIMOS
christophe.rey@uca.fr

***Abstract*—This paper proposes a new approach to non-player character (NPC) decision-making in games using a Prolog and ontologies declarative approach, addressing limitations of traditional techniques such as flexibility and maintainability. We implement a more complex Unity version of the Wumpus World to show our method's effectiveness and its potential to facilitate game AI design and prototyping. Our approach can achieve the same results as the classical methods without having to hard-code all the transitions between each possible game situation.**

***Index Terms*—declarative, ontologies, behavior, NPC, games**

## I. Introduction

Artificial intelligence's (AI) growth in gaming has driven demand for advanced, realistic non-player characters (NPCs), making their human-like development a key research focus in game AI. Traditionally, NPC decision-making has been developed using hand-crafted conditional statements, state machines, or behavior trees [1]. However, these methods can be time-consuming and difficult to maintain as game complexity increases. This sometimes leads to generated behaviors that lack the adaptability and flexibility required for more dynamic game scenarios. Indeed, the difficulty of creating relevant NPCs increases with the number of actions they can perform. Game AI designers must then anticipate a wide range of specific possible cases to offer the best possible game experience. Coding every possible situation can lead to difficult-to-maintain "spaghetti code" [2].

We propose a declarative alternative approach where the decision-making is obtained by reasoning with Prolog rules and domain ontologies to describe NPCs' characteristics and preferences. The obtained methodology and the modeled knowledge, especially the ontologies, are reusable for other games. We have developed a slightly more complex version of the Wumpus World game from [3] in the Unity game engine to show that our approach can achieve the same results as the classical methods without having to hard-code all the transitions between each possible game situation. Although declarative programming takes some time to learn, it offers a different and expressive way to imagine and iterate games as well as quickly prototyping and testing new behaviors before implementing them in more complex environments. The proof of concept code and a demo video are available: https://anonymous.4open.science/r/Wumpus_World/.

This research was funded by the French National Research Agency (ANR) and the European Regional Economic Development Fund (FEDER).

979-8-3503-2277-4/23/©2023 European Union

## II. Related Work

Prolog-like reasoners are incorporated into research projects like MKULTRA [4] and commercial games like Project Highrise and City of Gangsters using custom C# reasoner engines [5] for improved compatibility and performance, albeit with limited control and expressiveness compared to Prolog. The ThinkEngine framework is developed by [6] to interface declarative Answer Set Programming (ASP) modules to a game engine via their EmbASP library. UnityIIS [7] uses symbolic reasoning for planning and rational decision-making using ASP and the Ontology Web Language (OWL). A new Wumpus World frame and first-order logic solution were recently implemented in Flora-2 [8].

## III. Game Concept and AI Design

### A. Game Concept

To introduce our approach, we have developed a proof-of-concept game in Unity, using C#, based on the classic logic game Wumpus World. The game is a grid-based environment

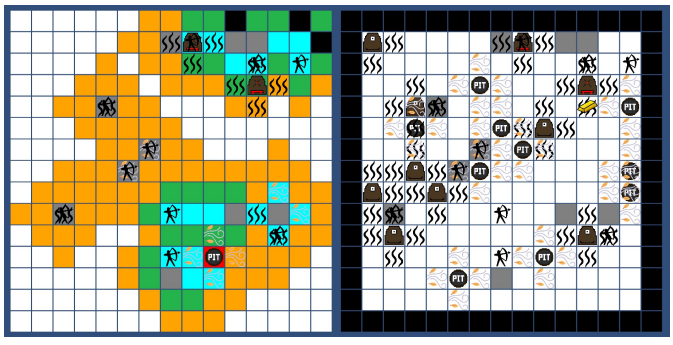

Fig. 1. Screenshot of the proof-of-concept game with dual cave views: agents' perspective on the left and all elements on the right. Colored cells distinguish elements: black for walls, gray for cave entrances, green for safe areas, red for danger, orange for unknown, and blue for visited areas. Brown cyclops monsters symbolize wumpuses, with adjacent cells having stenches. Cells with pits are labeled accordingly and are surrounded by breezes.

where an agent, controlled by AI, explore a cave to find gold, and then exit the cave. All this while avoiding dangerous cells with a pit or a monster, a.k.a. a wumpus. If the agent goes into a dangerous cell, he dies and loses the game. The cells adjacent to dangerous cells are surrounded by a breeze for pits and a stench for wumpuses, indicating the presence of danger. The agent moves one cell at a time and must deduce where the danger is to avoid it; this mechanic is similar to the classic Minesweeper logic puzzle game. When the agent has deduced the precise location of a wumpus, he can shoot an arrow in its direction to kill it, thus making the cell safe. However, the agent only has a limited number of arrows, so he must use them wisely. Some cells in the caves are walls that the agent cannot cross. In our more complex version, the cave is generated randomly and procedurally. It is possible to have multiple agents at the same time in the same cave. Finally, agents now have personality traits and objectives to generate actions. A view of an ongoing game is shown in Figure I.

### B. AI Design

Our decision-making AI is designed as shown in figure 2. It is orchestrated by Prolog rules executed by a SWI-Prolog environment. Communication between Unity and SWI-Prolog is ensured through a socket interface. We can generate different game environments by choosing a random seed. Besides this logic-based AI, the game prototype also allows to choose between three other AI approaches for the agent with a single click: AI based on hand-made conditional statements, finite state machines and behavior trees. Each AI is coded in a different module, while using the same shared modules, making each approach modular and allowing for the easy addition of new AI types. Like the logic-based AI, all other AI approaches produce a list of objectives based on the personality of the agent and the present elements of the game. Then, based on the objectives generated, each AI can produce a set of possible actions. Each action has a utility value based on the agent's personality meaning that the agent will choose the action to execute, among all available, according to its personality. In this setting, AI approaches can be fairly compared.

## IV. Results

### A. Qualitative

In all tested game environments, all four AI approaches follow the same decision-making steps and lead to the same result. It is therefore interesting to compare what each approach requires of the AI designer to work. Not surprisingly, logic-based AI benefits from its high declarative power: designing rules and ontologies can be more intuitive for someone knowing both Prolog and C# than modifying the C# code associated with the other three approaches. The second great advantage is that ontologies are widely reusable in other contexts since they contain general domain such as personality traits or actions. By comparison, the other three approaches contains the same knowledge, however deeply intricated into the C# code. We argue that it limits reusability. Our methodology enables the simple addition of reasoning steps for more complex decision-making. For instance, we can infer NPC emotions from an ontology, then determine objectives based on those emotions.

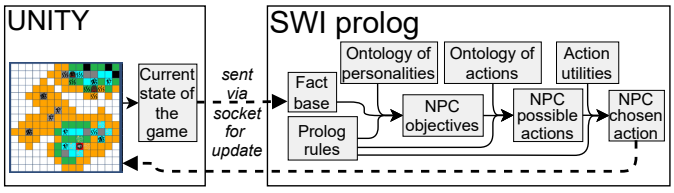


Fig. 2. Prolog rules and ontologies are used for decision-making. First, They infer NPC's objectives through facts, rules, and a personality trait ontology, then determines potential actions via rules and an action ontology. Finally, it select the highest utility action. Ontologies, expressed in a concise OWL sub-language, consist of Prolog concept hierarchies along orchestration rules.

### B. Performance

Our logic-based AI is currently slower than the other three approaches: it takes between 10 to 20 ms per turn, while the others are from 0 to 3 ms per turn. The difference is not due to reasoning in Prolog, which is comparatively as fast as the other methods, but to socket communications and updates of the Prolog fact base. In addition to the socket, we used the official SWI-Prolog C# SwiPLCs interface: https://github.com/SWI-Prolog/contrib-swiplcs, but it is deprecated and not fully functional. A tighter integration of Prolog into Unity like in [5] could be considered but it loses SWI-Prolog features like the tabled resolution and the Well-Founded Semantics.

## V. Conclusion and Future Work

We are looking into generalizing the approach to more expressive ontologies such as OWL ontologies. We are also working on improving performances by optimizing communications and fact base updates. We continue to explore other AI methods like ASP or Flora-2 as an alternative to Prolog, or machine learning and Large Language Models (LLM) to generate actions from descriptions of game rules and situation. Finally, we would like to test whether our approach can be generalized to perform planning, e.g. with ontologies in classical Goal Oriented Action Planning (GOAP).

## References


[1] G. N. Yannakakis and J. Togelius, *Artificial Intelligence and Games*, 1st ed. Springer International Publishing : Imprint: Springer, 2018.

[2] J. Gillberg, "Ai behavior editing and debugging in 'tom clancy's the division'," 2016, game Developer Conference.

[3] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*, 4th ed. Pearson, 2021.

[4] I. Horswill, "Mkultra (demo)," *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2015.

[5] ——, "Logic programming in commercial games: Experiences and lessons learned," 2023, game Developer Conference.

[6] D. Angilica, G. Ianni, and F. Pacenza, "Declarative ai design in unity using answer set programming," in *2022 IEEE Conference on Games (CoG)*, 2022, pp. 417–424.

[7] A. Brännström and J. C. Nieves, "A framework for developing interactive intelligent systems in unity," in *EMAS'22: 10th International Workshop on Engineering Multi-Agent Systems AAMAS*, vol. 10, 2022.

[8] S. Mehdipour Ataee, "A frame and first-order logic solution for the wumpus world: Implemented in flora-2," *Expert Systems with Applications*, vol. 220, p. 119717, 2023.