Demo: Generate Emergent NPC Behaviours With Symbolic Reasoning

Sylvain Lapeyrade

Demo: Generate Emergent NPC Behaviours With Symbolic Reasoning

Sylvain Lapeyrade

Université Clermont Auvergne, CNRS, LIMOS, France

Abstract

We propose to use ontologies and declarative symbolic reasoning to generate emergent behaviours of Non-Player Characters (NPCs). The objective is that the game designer only needs to specify the rules of the game and its components in a declarative way, as he would naturally do in a traditional board game. The logic reasoner will then deduce the NPC behaviours that comply with the game designer's rules without the game designer having to manually specify all the game possibilities by hand. We illustrate this approach on a prototype of the revisited Wumpus World game made on the Unity game engine with a Prolog environment. This approach is combined with the Well-Founded Semantics (WFS) to solve the problem of representation and reasoning despite the lack of NPC knowledge.

Keywords

Demo, Symbolic Reasoning, Unity, Prolog, Ontologies, Well-Founded Semantics,

1. Introduction

Classical Artificial Intelligence (AI) with reasoning and symbolic representation are quite present in Game AI, especially in Procedural Content Generation (PCG) [1, 2, 3] and General Game Playing (GGP) with Game Description Langages (GDL) [4, 5, 6]. However, there are very few examples of games that use it to design Non-Player Character (NPC) behaviour. Instead, developers prefer to use their own ad hoc techniques or classic NPC AI techniques such as *Finite-State Machine* (FSM), *Behaviour Trees* (BT), *Utility Based AI, Action Planning techniques* [7, 8, 9].

However, reasoning and symbolic representation allow game designers to state the rules and facts of their games as they would naturally do in a classic board game. They do not have to think exhaustively about the possible game situations but can instead describe *what is* in the game (i.e. facts), *what is possible to do* and *what is not possible to do* in the game i.e. rules. This will allow the game designer to generate NPC behaviours that they would not necessarily have thought of, but which nevertheless formally respect the rules of the game that they have listed in their design. The player's game experience can then seem less linear and can be more tailored to their specific game situation. Logic-based AI also allows for an easy and complete explanation of results, which can be very useful in explaining the AI's behaviour to the player.

The Joint Workshop Proceedings of the 2022 Conference on Artificial Intelligence and Interactive Digital Entertainment

sylvain.lapeyrade@uca.fr (S. Lapeyrade)

https://sylvainlapeyrade.github.io (S. Lapeyrade)

0000-0002-0984-6243 (S. Lapeyrade)

2. Related Work

Even if they are not popular at the moment, there are some examples of approaches using symbolic reasoning in combination with a game engine, here all the works use Unity[1].

MKULTRA [10] focus on the natural language processing (NLP) capabilities of logic programming to enable deeper player-NPC interaction. It uses UnityProlog[2], a custom Prolog engine made by the author.

Possible future integration between games engines and Multiagent Systems (MAS) is discussed in [11] as well as results of previous work implementing Belief-Desire-Intention (BDI) agents using the tuProlog engine [12].

More recently, the UnityIIS framework [13] allows symbolic reasoning for planning and rational decision-making using Answer Set programming (ASP) and the Ontology Web language (OWL).

Another recent framework, VEsNA [14] uses DialogFlow[3] and JaCaMO [15] to manage virtual environments with cognitive agents able to support decision-making.

3. Symbolic Reasoning

We use ontologies so that agents can represent their knowledge about the world and make reasoning as the knowledge will have semantics. Our ontologies are organised as hierarchical packages, like in Object-oriented Programming (OOP) and its principle of *encapsulation*. Only specific parts of the ontology are accessible from other ontologies. This is to have generic and modular

[1] Unity Platform: https://unity.com/

[2] UnityProlog: https://github.com/ianhorswill/UnityProlog

[3] Google DialogFlow: https://cloud.google.com/dialogflow

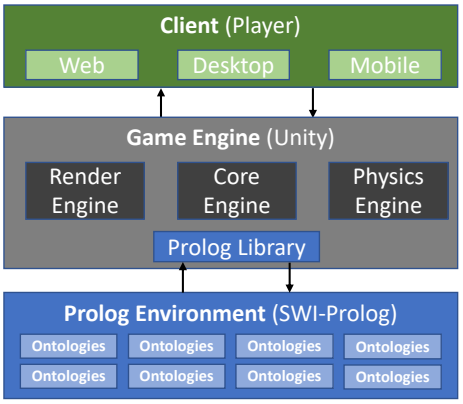

Figure 1: Our architecture combining Unity and Prolog.

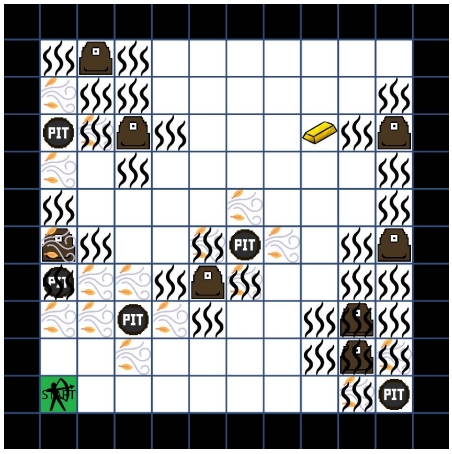

Figure 2: Screenshot from the game.

ontologies and to help the developer to know how to use them.

In order to deal with negative facts, two main semantics are used, the *Well-Founded Semantics* (WFS) [16] and the *Stable Model Semantics* [17] at the basis of *Answer Set Programming* (ASP) [18]. The Stable Model Semantics makes it possible to model simply indicates explicitly that a statement is false but generate multiple models for each query. However, the WFS was preferred because it only generates one model and introduces a third truth value for undefined values [19].

Prolog can do planning using *backward chaining* to find the conditions necessary to fulfil the conditions of a given goal [20]. By giving the inference engine the goal that the agent is trying to reach, it will be able to return the set of sub-goals (e.g. actions) to achieve the main goal and thus lead to an intelligent action sequence. This is very powerful since the sequences are not hard-coded by the game designers and potential sequences not imagined by the game designer may emerge.

4. Prototype

Figure 1 shows how a logic programming development environment is integrated with a game engine. Unity was chosen as the game engine and SWI-Prolog [21] as Prolog environment, as it notably supports the WFS and provides interfaces with C#[4] which can be used with Unity and C++[5] which can be used with the Unreal Engine.

To use the Prolog interface from within Unity, we simply import the interface DLL file into Unity Plugins and call the interface functions in a C# script. We personally decided to separate the code that interacts directly with Prolog in a separate file. This is to make the code more modular, and to be able to integrate the interface into another existing game. Placing the code for the script that interacts with Prolog in a library would allow a game designer to use the interface with minimal knowledge of Prolog.

The idea of the game for the prototype of our approach comes from the reference artificial intelligence textbook *Artificial Intelligence: a Modern Approach* [20]. The authors use the example of the game Wumpus World to show the use of a Knowledge-Base agent. In the game, an agent must explore a cave, room by room, in order to collect gold and return to the cave entrance, all the while avoiding pits and monsters called wumpus. To avoid pits and wumpus while exploring, the agent must use clues surrounding the rooms containing them, and deduce where they are. This mechanic is similar to Minesweeper[6], where the player must deduce where the bombs are from the numbers surrounding the unexplored rooms.

As the basic Wumpus World game is rather simple and does not allow for the use of very complex behaviour, we have extended the game to include more elements such as agent characteristics, personalities, states, different character types, etc. The aim is to create more possible game situations, so that the prologue environment can be used to generate more different behaviour. The aim is to create more possible game situations, so that the Prolog environment can be used to generate more different behaviours. We also generate the cave procedurally with a seed, so that it can be as large as we want it to be and still be interesting to explore. Figure 2 shows a screenshot of the game with a medium-sized cave, the agent in the starting position, bottom left, and the gold to be recovered, top right.

[4]C# Interface: https://github.com/SWI-Prolog/contrib-swiplcs
[5]C++ Interface: https://github.com/SWI-Prolog/packages-cpp

[6]Minesweeper: https://w.wiki/5ZRi

5. Conclusion

Our demo is still being improved, we want to make the possible game situations more complex despite the basic game example being quite simple to get out of the stereotypical behaviours and show that the generated behaviours can be very qualitative. We are currently in a research collaboration with a game studio to create a commercial video game with NPC AI based on symbolic reasoning. The studio's developers have no experience in declarative logic programming, so we will be able to see how well they master the approach and correct any difficulties they encounter.

Acknowledgments

This research was funded by the French National Research Agency (ANR) and the European Regional Economic Development Fund (FEDER). Jan Wielemaker is acknowledged for his helped with the use of the WFS.

References

[1] A. M. Smith, M. Mateas, Answer set programming for procedural content generation: A design space approach, IEEE Transactions on Computational Intelligence and AI in Games 3 (2011) 187–200. doi:10.1109/TCIAIG.2011.2158545.

[2] J. Dormans, Adventures in level design: Generating missions and spaces for action adventure games, in: Proceedings of the 2010 Workshop on Procedural Content Generation in Games, PCGames '10, 2010, pp. 1–8. doi:10.1145/1814256.1814257.

[3] G. Smith, J. Whitehead, M. Mateas, Tanagra: A mixed-initiative level design tool, in: Proceedings of the Fifth International Conference on the Foundations of Digital Games, FDG '10, 2010, p. 209–216. doi:10.1145/1822348.1822376.

[4] M. Genesereth, N. Love, B. Pell, General game playing: Overview of the aaai competition, AI magazine 26 (2005) 62–62.

[5] É. Piette, D. J. N. J. Soemers, M. Stephenson, C. F. Sironi, M. H. M. Winands, C. Browne, Ludii – the ludemic general game system, in: Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020), volume 325 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2020, pp. 411–418.

[6] D. Perez-Liebana, J. Liu, A. Khalifa, R. D. Gaina, J. Togelius, S. M. Lucas, General video game ai: A multitrack framework for evaluating agents, games, and content generation algorithms, IEEE Transactions on Games 11 (2019) 195–214.

[7] G. N. Yannakakis, J. Togelius, Artificial Intelligence and Games, 1st ed. 2018 ed., Springer International Publishing : Imprint: Springer, 2018. doi:10.1007/978-3-319-63519-4.

[8] I. Millington, AI for games, third edition ed., Taylor & Francis, a CRC title, 2019.

[9] A. Simonov, A. S. Zagarskikh, V. Fedorov, Applying behavior characteristics to decision-making process to create believable game ai, Procedia Computer Science (2019).

[10] I. Horswill, Mkultra (demo), Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment 11 (2015) 223–225.

[11] S. Mariani, A. Omicini, Game engines to model MAS: A research roadmap, in: C. Santoro, F. Messina, M. D. Benedetti (Eds.), Proceedings of the 17th Workshop "From Objects to Agents", volume 1664, CEUR-WS.org, 2016, pp. 106–111.

[12] E. Denti, A. Omicini, A. Ricci, Multi-paradigm java–prolog integration in tuprolog, Science of Computer Programming 57 (2005) 217–250. doi:10.1016/j.scico.2005.02.001.

[13] A. Brännström, J. C. Nieves, UnityIIS: Interactive Intelligent Systems in Unity, 2021. URL: https://git.io/JMpzr.

[14] A. Gatti, V. Mascardi, Towards vesna, a framework for managing virtual environments via natural language agents, Electronic Proceedings in Theoretical Computer Science 362 (2022) 65–80. doi:10.4204/EPTCS.362.8.

[15] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, A. Santi, Multi-agent oriented programming with jacamo, Science of Computer Programming 78 (2013) 747–761.

[16] A. Van Gelder, K. A. Ross, J. S. Schlipf, The well-founded semantics for general logic programs, Journal of the ACM 38 (1991) 619–649. doi:10.1145/116825.116838.

[17] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: R. Kowalski, Bowen, Kenneth (Eds.), Proceedings of International Logic Programming Conference and Symposium, MIT Press, 1988, pp. 1070–1080.

[18] V. Lifschitz, Answer set programming, Springer Berlin, 2019.

[19] U. Nilsson, J. Maluszynski, Logic, Programming, and PROLOG, 2nd ed., John Wiley & Sons, Inc., 1995.

[20] S. J. Russell, P. Norvig, Artificial intelligence: a modern approach, fourth edition ed., Pearson, 2021.

[21] J. Wielemaker, T. Schrijvers, M. Triska, T. Lager, Swi-prolog, Theory and Practice of Logic Programming 12 (2012) 67–96. doi:10.1017/S1471068411000494.