Reasoning With Ontologies for Non-Player Character Decision-Making in Games

Sylvain Lapeyrade
Université Clermont Auvergne, CNRS, LIMOS
Clermont-Ferrand, France
sylvain.lapeyrade@uca.fr

ABSTRACT

In most games, the decision-making of non-player characters (NPCs) is usually constructed using variants of state machines, behaviour trees, utility-based AI or planning. These methods are relatively simple to implement, but have drawbacks in that it can be difficult to create complex non-hard-coded behaviour for many agents and to maintain the algorithms, especially when scaling up. We show that logic programming can be used to overcome these limitations. This new approach is intuitive because game designers usually think of their games with rules that closely resemble logic rules. A methodology is introduced to design both general and modular behaviour using hierarchical and scalable ontologies. Moreover, we show that this approach can be seamlessly combined with the well-founded semantics (WFS) to solve the problem of representation and reasoning despite the lack of NPC knowledge.

KEYWORDS

Logic Programming, Ontologies, Reasoning, Game Artificial Intelligence, Well-Founded Semantics

ACM Reference Format:
Sylvain Lapeyrade. 2022. Reasoning With Ontologies for Non-Player Character Decision-Making in Games. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Auckland, New Zealand, May 9–13, 2022*, IFAAMAS, 3 pages.

1 EXTENDED ABSTRACT

In video games, non-player characters (NPCs) often have a very basic decision-making process which leads to behaviours with a low level of credibility and therefore a less intense experience for the player. This manifests itself in overly simple or repetitive behaviour and inconsistent actions. This problem can be observed with NPCs in commercial games, virtual assistants in educational or serious games, or with virtual agents in simulations. The game artificial intelligence (AI) research community has made ongoing efforts to try to identify the reasons for this lack and to improve the credibility of NPCs [7, 13].

The basic decision-making regarding NPC behaviour is partly explained by game development policies. The video game industry faces strict resource constraints, game designers want to control the game experience and the player does not always want believable NPC behaviour.

The most popular methods for decision-making in game AI, according to [7, 10, 13] include *Finite-State Machine* (FSM), *Behaviour Trees* (BT), *Utility Based AI* and *Action Planning techniques*. Despite its increasing popularity *Machine Learning* is not widely used for decision-making in game AI.

1.0.1 Is Logic Programming Too Slow? Prolog [2], one of the most popular logic programming languages, is sometimes associated with being slow. The slowness is linked to the fact that we do not limit the length of the reasoning as in e.g. GOAP, and to the complexity of the processing required, particularly during the *unification phase*. The *unification* is the elementary operation that Prolog performs to make two terms identical and see if one can replace the other, this is called a *substitution*.

The *tabled resolution* can also speed up the processing as it avoids re-evaluating predicates indefinitely and prevents *left recursion*, which is a goal calling a variant of itself recursively [8]. In addition, many optimisation efforts have been made to speed up the results in parallel with the constant improvement of the computing hardware, which means that the results are calculated faster than a few decades ago.

1.0.2 Is Declarative Programming Too Complicated? Prolog is a declarative programming language. This means that the programmer can express the rules of the game in a declarative way as opposed to imperative languages such as C, Python or Java where the rules are written in a procedural way. Imperative languages are undoubtedly more popular than declarative ones, especially among game developers where the most popular game engines Unity [11] and Unreal Engines [3] use respectively C# and C++ for scripting. But the same can be said for action languages such as STRIPS or PDDL which are used for planning. With a proper methodology and a beginner-friendly interface, such as a game engine plugin, game designers should be able to use logic programming without having to be a Prolog expert.

1.0.3 Rules-Based Systems. The AI approach to modelling NPC behaviour that we have found most similar in principle to the way logic programming works are *rule-based systems*. They are described by Millington as "the most complex non-learning decision makers [covered in his book] [...] a formidable programming task that can support incredible sophistication of behaviour. It can support more advanced AI than any seen in current-generation games" [7].

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Auckland, New Zealand. © 2022 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

As exposed in the last section, Millington also points out that the main weaknesses of rule-based systems are the difficulty of writing good rules, known as the *knowledge acquisition* problem. This makes them more difficult to use compared to *behaviour trees* or *state machines* that can be directly created from popular game engines. This is why, despite some attempts, including such as [5], they are not very common. The issue does not seem to be the quality of the approach but the handling of the algorithm.

1.1 Advantages of Using Logic Programming

The main advantages of using logic programming over simple *rule-based systems* is the utilisation of *backward chaining* [8] and knowledge inferences through an inference engine. The use of an appropriate methodology for constructing ontologies may also address the problem of *knowledge acquisition* described in the last paragraph. Logic-based AI also allows for a very easy and complete explanation of the results, unlike e.g. learning-based techniques, which can be very useful in explaining the AI's behaviour to the player.

1.1.1 Knowledge Inference. An inference engine like Prolog can also deduct new facts from the facts already in the knowledge base. It works by combining available data and inference rules to extract more data until a specified goal is reached. Every fact does not have to be known as inference rules can be used to derive them. This again emphasises the need to have the most precise rules and ontologies possible.

1.1.2 Planning with Backward Chaining. An inference engine like Prolog can do planning using *backward chaining*. *Backward chaining* aims, via a deep first search algorithm, to find the conditions necessary to fulfil the conditions of a given goal [9]. 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. However, these behaviours are framed by the rules declared by the game designer and, if the rules are well defined, should not result in inconsistent actions.

1.2 Hierarchical Ontologies

We build our ontologies to be as general as possible in order to generate behaviours rather than the game designer having to think about every possible situation when writing the rules. Our ontologies are organised as hierarchical packages, like in OOP and its principle of *encapsulation*. Only specific parts of the ontology are accessible from other ontologies. This is to have generic and modular ontologies and to help the developer to know how to use the different ontologies.

1.3 Representing False and Unknown Facts

Prolog is based on the *Close-World Assumption* (CWA). The CWA is the presumption that a **true** statement is also *known* (i.e.present in the knowledge base or derivable from the knowledge base) to be **true**. Conversely, a statement that is *not known* to be **true** is considered as **false**. Therefore, any statement of which we have no knowledge, or which cannot be proven, is evaluated as **false**. This is known as the *Negation As Failure* (NAF) inference rule [8]. Furthermore, in logic programming, one cannot assert **false** facts, or rules that lead to **false** facts. This inability to differentiate between facts that are **false** because they are factually **false** and facts that are **false** because they are unknown is restrictive for the modelling of agents' knowledge and for the quality of their reasoning.

1.3.1 Well-Founded Semantics. In order to deal with negation, two main semantics are used, the *Well-Founded Semantics* (WFS) [12] and the *Stable Model Semantics* [4] at the basis of *Answer Set Programming* (ASP) [6].

The Stable Model Semantics makes it possible to model simply the *logical negation* (i.e. to indicate explicitly that a statement is **false**) but generate multiple models for each query. A similar piece of work to what we are trying to do has been undertaken in [1], where an ASP framework has been integrated in Unity to use rule-based systems and planning.

However, the WFS was preferred because it only generates one model and introduces a third truth value for **undefined** values [8]. One contribution of this work is to use this third truth value to represent the absence of knowledge.

1.3.2 A Third Truth Value. The WFS allows to manage, without leading to inconsistency, the cases where facts are known to be **true** and **false** at the same time. This type of case is particularly possible when the NPC receives conflicting information from several different sources. The WFS provides the possibility for *Well-Founded Partial Models* [12] to circumvent the presence of contradictions and proceeds to derive as many two-valued facts as possible, although some of the consequences may remain **undefined**. This indefiniteness can be used to represent the unknown facts according to some deduction rules. The NPC can also choose to give more credence to one source than the other and choose its truth value, or choose to remain uncertain and take the **undefined** truth value for this fact.

1.4 Conclusion

After noting the reasons for the lack of credibility of AI behaviour in video games. We also noted the absence of logic programming among the behaviour modelling techniques. We have explained the main reasons for this absence and we propose a new approach with solutions. We are currently working on a prototype game under Unity with SWI-Prolog to test our approach.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Francesco Calimeri, Stefano Germano, Giovambattista Ianni, Francesco Pacenza, Simona Perri, and Jessica Zangari. 2018. Integrating Rule-Based AI Tools into Mainstream Game Development. In *RuleML+RR*.

[2] Alain Colmerauer and Philippe Roussel. 1996. The birth of Prolog. In *History of programming languages—II*. 331–367.

[3] Epic Games. 2022. Unreal Engine: The most powerful real-time 3D creation tool. https://www.unrealengine.com/ Accessed may 2022.

[4] Michael Gelfond and Vladimir Lifschitz. 1988. The Stable Model Semantics For Logic Programming. MIT Press, 1070–1080.

[5] Wright Ian and Marshall James. 2000. RC++: a rule-based language for game AI. In *Proceedings of the First International Conference on Intelligent Games and Simulation (GAME-ON 2000)*. SCS Europe BVBA. Other page information: - Conference Proceedings/Title of Journal: Proceedings of the First International Conference on Intelligent Games and Simulation (GAME-ON 2000) Other identifier: 2000441.

[6] Vladimir Lifschitz. 2019. *Answer set programming*. Springer Berlin.

[7] Ian Millington. 2019. *AI for games* (third edition ed.). Taylor & Francis, a CRC title, Boca Raton.

[8] Ulf Nilsson and Jan Maluszynski. 1995. *Logic, Programming, and PROLOG* (2nd ed.). John Wiley & Sons, Inc., New York, NY, USA.

[9] Stuart J. Russell and Peter Norvig. 2021. *Artificial intelligence: a modern approach* (fourth edition ed.). Pearson, Hoboken.

[10] Andrey Simonov, Aleksandr S. Zagarskikh, and Victor Fedorov. 2019. Applying Behavior characteristics to decision-making process to create believable game AI. *Procedia Computer Science* (2019).

[11] Unity Technologies. 2022. Unity Real-Time Development Platform — 3D, 2D, VR & AR Engine. https://unity.com/ Accessed may 2022.

[12] Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. 1991. The well-founded semantics for general logic programs. *J. ACM* 38, 3 (July 1991), 619–649. https://doi.org/10.1145/116825.116838

[13] Georgios N. Yannakakis and Julian Togelius. 2018. *Artificial Intelligence and Games* (1st ed. 2018 ed.). Springer International Publishing : Imprint: Springer, Cham. https://doi.org/10.1007/978-3-319-63519-4