Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2025)

From Explainability to Interpretability: A Case Study on Game AI Using Program Synthesis

Manuel Eberhardinger[1,2]
[1]Institute of Applied Artificial Intelligence
Stuttgart Media University, Germany
[2]Ruhr-University Bochum, Germany
eberhardinger@hdm-stuttgart.de

Abstract

Many recent advances in artificial intelligence remain difficult to apply in real games due to their black-box nature. These systems often lack transparency and control, making them difficult to integrate into games where player experience and narrative coherence are important. Unpredictable or inexplicable agent behavior can confuse players and frustrate developers. My dissertation explores how program synthesis can address this issue by generating interpretable, controllable representations of agent behavior. Instead of relying on black-box neural network policies, symbolic programs are extracted or generated that capture agent logic in a readable and editable form. Several methods are explored: synthesizing functional programs to imitate and explain game agents, adapting logical program policies to multi-agent settings, and evaluating large language models for code generation across domains such as simplified Atari games, *Baba is You*, and also tabletop games. An open research question is whether the created conceptual program libraries are transferable across different game domains. My research aims to bridge the gap between AI capabilities and game development needs by making agent behavior transparent, explainable, and adaptable for developers.

1 Introduction

While artificial intelligence (AI) research has made tremendous progress, from agents that can handle complex games (Silver et al. 2018) to large language models that generate code (Austin et al. 2021) or human-like narratives (Tian et al. 2024), much of this work remains difficult to integrate into actual games (Cakmak et al. 2024). One key reason is the lack of control and transparency (Zhu et al. 2023). Many of today's AI systems behave like black-boxes: they make decisions that are often effective, but rarely explainable, controllable, or trustworthy. For game developers, this poses a serious problem, since the integration of black-box AI models could lead to uncontrollable behavior of game agents that confuse the player. In interactive digital entertainment, AI is not just about performance, but also about game experience, narrative coherence, and design intent. If game developers and designers cannot understand or influence what the AI system is doing, the result is a game that feels inconsistent, frustrating, or out of sync with the game world.

In my dissertation, I aim to address the aforementioned problems by exploring how program synthesis can be used to extract interpretable, controllable representations of black-box agent behavior or programs are directly generated from natural language descriptions. Rather than treating AI systems as immutable neural networks or evolved artifacts, my aim is to convert their logic into symbolic programs that are readable, editable, and transferable to other domains or agents. These programmatic representations help developers answer critical questions: Why did the agent make that move? What concept is it using? Can I tweak its logic for a new level or mechanic?

As part of my dissertation, I have developed several different methods using different programming concepts, e.g., functional programming based on Lisp for explaining policies of game agents (Eberhardinger, Maucher, and Maghsudi 2023a,b; Eberhardinger et al. 2024b), adapted the Logical Program Policies (Silver et al. 2020) framework for a simple multi-agent reinforcement learning (MARL) problem (Eberhardinger, Maucher, and Maghsudi 2022), and evaluated large language models (LLM) for synthesizing Python and Java code applied on several domains, such as playing miniature versions of Atari games, solving puzzles in *Baba is You* or finding heuristics for tabletop games in the TAG framework (Gaina et al. 2020; Eberhardinger et al. 2024a, 2025).

There are, however, still some limitations that need to be addressed in my thesis. In addition to the inherent interpretability of programs, the transferability of concepts in programs is also of particular interest to me. Programs can be analyzed and restructured so that parts of the program can be extracted into a new function. In this way, a library of programs can be created that represent the concepts of a game agent (Eberhardinger, Maucher, and Maghsudi 2023a). There is currently no assessment of the transferability of these concepts from one domain to another, but as software engineers use different frameworks in their daily work, the transferability of programs found in automatically created libraries should also be possible. Furthermore, I am currently working on an approach to generate cognitive-plausible explanations for board game agents, which are also represented as programs.

In the end, my thesis proposes several methods based on program synthesis and different programming concepts to

create explanations or generate fully interpretable programmatic policies for game agents. Program synthesis and code generation are used interchangeably in this work.

As I am in the final phase of my dissertation, I am particularly interested in participating in Phase Two of the AIIDE Doctoral Consortium about career opportunities in academia or industry. I look forward to engaging with the community to receive feedback on my published research, as well as constructive input on the direction of my ongoing and future work.

2 Related Work

Program synthesis is the task of finding a program for a given specification (Gulwani, Polozov, and Singh 2017), such as a natural language description, input-output examples, or logical constraints. Gulwani, Polozov, and Singh (2017) give a brief overview of search-based program synthesis such as brute-force enumerative search, stochastic search or constraint solving. Ellis et al. (2021) introduced a framework, called DreamCoder, which divides program search into three different phases which are used in an iterative search algorithm:

1. *Wake*: Use the enumerative search in the first iteration and the neural-guided search in the following iterations to solve the given tasks
2. *Sleep - Abstraction*: Analyze found programs to extract functions into a library which grows in each iteration
3. *Sleep - Dreaming*: Train a neural network on an automatically generated data set of randomly sampled programs executed in the domain and on the solved tasks from 1. After training repeat from the *Wake*-phase with neural-guided search.

We extend this approach in (Eberhardinger, Maucher, and Maghsudi 2023a,b) so that the framework can be used with RL environments.

Another well-known approach is genetic programming (Koza 1992), which is based on evolutionary search optimizing a fitness function by searching over a population of programs on which crossover and mutations are applied. We combine genetic programming with library learning based on Stitch (Bowers et al. 2023), an improved version of the DreamCoder library learning module, in our follow-up work (Eberhardinger et al. 2024b).

Logical programs are also used in different forms for Game AI. Cropper, Evans, and Law (2020) use inductive logic programming to recover game rules from game traces for over 50 games. Another method to create logical programs are Logical Program Policies (LPP) (Silver et al. 2020), which we briefly explain in section 3.1. Silver et al. (2020) showed that LPP can be used for five simple strategy games, being much more resource efficient than neural networks.

Recently, LLMs have demonstrated impressive capabilities in writing code from natural language descriptions (Austin et al. 2021), which opens many possibilities to apply these methods on games. (Liu et al. 2025) uses LLMs, similarly to us, in a hill-climbing approach on a simple RL environment, we in contrast use game-related environments (Eberhardinger et al. 2025).

3 Previous Work

3.1 Logical Program Policies for Multi-Agent RL

In previous work (Eberhardinger, Maucher, and Maghsudi 2022), we adapted the LPP framework from single-agent reinforcement learning (RL) to the MARL setting. We demonstrated its feasibility in the Level-Based Foraging environment (Christianos, Schäfer, and Albrecht 2020), where multiple agents must cooperate to collect fruits. In this environment, both agents and fruits are assigned levels: agents can collect a fruit alone if its level is equal to or higher than the fruit's; otherwise, cooperation with other agents is required.

LPPs are extracted from decision trees composed of logical combinations of small feature detector programs, which are generated using a probabilistic context-free grammar (Silver et al. 2020). These logical combination of programs make agent decisions interpretable and transparent. Rather than relying on a single program per agent, multiple candidate programs are sampled, and the best ones are selected based on their approximate reward in the training environment. This approach enables the resulting policies to generalize across environments without retraining, allowing them to be reused in new scenarios such as different number of agents or observation sizes while maintaining interpretability.

3.2 Learning of Generalizable and Interpretable Knowledge in Grid-Based RL Environments

In this work (Eberhardinger, Maucher, and Maghsudi 2023b,a), we introduced a framework, based on DreamCoder (Ellis et al. 2021), for learning reusable and interpretable knowledge that can reason about agent behavior in grid-based reinforcement learning environments. A first theoretical version of the idea was proposed in Ashlock et al. (2023). The framework is evaluated on a maze navigation task (Parker-Holder et al. 2022) and two simplified Atari-style games (Young and Tian 2019), *Asterix* and *Space Invaders*, demonstrating its ability to generalize across domains. We compare multiple program synthesis approaches, including enumerative search, neural-guided search, and a fine-tuned language model, both with and without learning a library of functions, to assess their effectiveness in capturing agent behavior. Furthermore, we analyze the functions extracted within the synthesized program libraries to uncover the underlying concepts the agents have learned, highlighting the potential of the framework to produce modular and interpretable representations of behavior. We also show explanations of programs found by visualizing the grid cells that are checked in the program.

In follow-up work, we showed that the use of genetic programming significantly improves run time by achieving comparable performance (Eberhardinger et al. 2024b). We also analyzed how integrating library learning into the genetic programming framework improves performance in the beginning but hinders the algorithm in the long run. The libraries found resembled the libraries of the previous work

(Eberhardinger, Maucher, and Maghsudi 2023b,a). Due to time constraints and scaling problems, this approach was only evaluated with the maze runner agent at the time of writing the paper; for the dissertation, the other two Atari environments should also be included.

3.3 From Code to Play: Benchmarking Program Search for Games using LLMs

In this work (Eberhardinger et al. 2024a, 2025), we introduced an easy-to-use, extensible framework for evaluating the current capabilities of LLMs in synthesizing game-related code. The framework is based on a hill-climbing algorithm where an LLM is used for mutation of programs as well as generating the initial program. For this, we integrated five different LLM providers, resulting in a total of 11 models from OpenAI, Anthropic, Mistral, Llama and Google. For Python code generation, the framework includes a diverse set of tasks: five miniature versions of Atari games with symbolic input representations (Young and Tian 2019), ten levels from *Baba is You* to test different game mechanics (Charity and Togelius 2022), an *Asteroids*-inspired vehicle-driving environment, and procedural content generation through maze generation. For Java, the framework uses 12 tabletop games from the TAG framework (Gaina et al. 2020). In total, LLMs are evaluated across 29 distinct game-related tasks, providing a broad benchmark for assessing their effectiveness in generating playable or usable program code. Our findings suggest that models still have difficulty creating programs for more complicated games and especially for TAG, where many user-defined classes are present.

4 Current & Future Work

4.1 Cognitive-Plausible Explanations for Board Game Agents

Currently, we work on an approach to create cognitive-plausible explanations for board game agents. Board game agents have achieved superhuman performance in various domains (Silver et al. 2018), but their decision-making process is incomprehensible for even experienced players. From a cognitive science perspective, these agents often diverge significantly from how humans approach game play, particularly in terms of how knowledge is represented and applied (Mańdziuk 2011; Mańdziuk 2012). To address this, we propose a method for explaining board game agents through cognitively plausible programmatic representations of their behavior. These programmatic representations can then be converted into natural language by an LLM so that they can be understood by everyone.

Building on the idea that game knowledge should be represented both pattern-based and hierarchically, our approach learns feature programs using genetic programming. These programs capture spatial relationships between grid cells, serving as interpretable components that reason about the state of the game. A decision tree is then trained on the activation patterns of these features, forming a hierarchical structure that connects low-level spatial patterns with high-level action decisions. This resembles how humans might decompose visual and strategic information in games. Nevertheless, it is not clear if this is human-like game playing, but for most average players it seems like an appropriate way to play board games. Professional players are, however, an exception to this and most likely not use spatial relationships between features, but rather their intuition.

Our method produces understandable explanations in Tic-Tac-Toe, but applying the method to more complex games presents significant challenges, which we are still trying to address.

4.2 Transferability of Program Libraries

An relevant open research question to my thesis, is to what degree automatically extracted programming libraries are transferable between different game domains. In most games, there are similar game concepts and mechanics, if the functions can be correctly abstracted from the game domain, it should be possible to reuse them in other games. For example, in almost all games it is necessary to locate game objects, so this function could be effectively used in *Space Invaders* to locate aliens but also in *Asterix* to locate where enemies are, but only if no game-related concepts or objects are used in the function.

The same applies to board games. Board games have a lot in common, e.g. *Tic-Tac-Toe* and *Connect 4*, both are turn-based games with two pieces on a grid board. In both games, the goal is to get three or four pieces in line in order to win. Since the board of *Tic-Tac-Toe* is smaller and only three pieces of the same player need to be in a line, the features used in *Tic-Tac-Toe* should be transferable to *Connect 4* and could be a good starting point for extending the library to game concepts of *Connect 4*.

To answer this research questions, program libraries should be automatically extracted based on (Eberhardinger, Maucher, and Maghsudi 2023a) and it can be calculated how many of the functions contained in a library of a given game can be used in other games, and how different libraries affect the performance of program search. This could be used as a proxy to find out which games share similar game concepts.

5 Conclusion

Program synthesis in the context of games is still an emerging research area, but it is gaining increasing attention due to its potential to create interpretable and controllable game assets. In my dissertation work I explored different program synthesis algorithms on games and showed that library learning is a useful tool to improve these algorithms. In the beginning, the focus was on explainability, since finding interpretable programmatic policies for game agents in a custom domain-specific language was challenging, due to the combinatorial explosion of the search space and, thus, only parts of the game policy could be explained. Using LLMs, however, made this feasible in Python for the simplified Atari games. In the future, I believe that there is a lot of potential to apply program synthesis on games. As research progresses, many of the early limitations I encountered, e.g., scalability and search efficiency, can be addressed.

References

Ashlock, D.; Maghsudi, S.; Liebana, D. P.; Spronck, P.; and Eberhardinger, M. 2023. Human-Game AI Interaction: Report from Dagstuhl Seminar 22251.

Austin, J.; Odena, A.; Nye, M.; Bosma, M.; Michalewski, H.; Dohan, D.; Jiang, E.; Cai, C.; Terry, M.; Le Quoc; and Sutton, C. 2021. Program Synthesis with Large Language Models. Jacob and Augustus contributed equally.

Bowers, M.; Olausson, T. X.; Wong, L.; Grand, G.; Tenenbaum, J. B.; Ellis, K.; and Solar-Lezama, A. 2023. Top-Down Synthesis for Library Learning. *Proceedings of the ACM on Programming Languages*, 7(POPL): 1182–1213.

Cakmak, D.; Maghsudi, S.; Liebana, D. P.; and Spronck, P. 2024. Computational Creativity for Game Development (Dagstuhl Seminar 24261). *Dagstuhl Reports*, 14(6): 130–214.

Charity, M.; and Togelius, J. K. A. I. 2022. Competition: Solving puzzle levels in a dynamically changing mechanic space. In *2022 IEEE Conference On Games (CoG)*, 570–575.

Christianos, F.; Schäfer, L.; and Albrecht, S. 2020. Shared experience actor-critic for multi-agent reinforcement learning. *Advances in neural information processing systems*, 33: 10707–10717.

Cropper, A.; Evans, R.; and Law, M. 2020. Inductive General Game Playing. *Machine Learning*, 109(7): 1393–1434.

Eberhardinger, M.; Cakmak, D.; Dockhorn, A.; Gaina, R. D.; Goodman, J.; Hoover, A. K.; Lucas, S. M.; Maghsudi, S.; and Liebana, D. P. 2024a. LLM-based Program Search for Games. *Computational Creativity for Game Development*, 156.

Eberhardinger, M.; Goodman, J.; Dockhorn, A.; Perez-Liebana, D.; Gaina, R. D.; Çakmak, D.; Maghsudi, S.; and Lucas, S. 2025. From Code to Play: Benchmarking Program Search for Games Using Large Language Models. *arXiv preprint arXiv:2412.04057*.

Eberhardinger, M.; Maucher, J.; and Maghsudi, S. 2022. Imitation Learning of Logical Program Policies for Multi-Agent Reinforcement Learning. In *2nd International Workshop on Explainable and Interpretable Machine Learning (XI-ML)*.

Eberhardinger, M.; Maucher, J.; and Maghsudi, S. 2023a. Learning of generalizable and interpretable knowledge in grid-based reinforcement learning environments. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 19, 203–214.

Eberhardinger, M.; Maucher, J.; and Maghsudi, S. 2023b. Towards Explainable Decision Making with Neural Program Synthesis and Library Learning. In *17th International Workshop on Neural-Symbolic Learning and Reasoning*.

Eberhardinger, M.; Rupp, F.; Maucher, J.; and Maghsudi, S. 2024b. Unveiling the decision-making process in reinforcement learning with genetic programming. In *International Conference on Swarm Intelligence*, 349–365. Springer Nature Singapore Singapore.

Ellis, K.; Wong, C.; Nye, M.; Sablé-Meyer, M.; Morales, L.; Hewitt, L.; Cary, L.; Solar-Lezama, A.; and Tenenbaum, J. B. 2021. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd acm sigplan international conference on programming language design and implementation*, 835–850.

Gaina, R. D.; Balla, M.; Dockhorn, A.; Montoliu, R.; and Liebana, D. P. 2020. TAG: A Tabletop Games Framework. In *AIIDE Workshops*.

Gulwani, S.; Polozov, O.; and Singh, R. 2017. Program synthesis. *Foundations and Trends® in Programming Languages*, 4(1-2): 1–119.

Koza, J. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. A Bradford book. Bradford. ISBN 9780262111706.

Liu, M.; Yu, C.-H.; Lee, W.-H.; Hung, C.-W.; Chen, Y.-C.; and Sun, S.-H. 2025. Synthesizing Programmatic Reinforcement Learning Policies with Large Language Model Guided Search. In *The Thirteenth International Conference on Learning Representations*.

Mańdziuk, J. 2012. Human-like intuitive playing in board games. In *Neural Information Processing: 19th International Conference, ICONIP 2012, 2012, Proceedings, Part II 19*, 282–289. Springer.

Mańdziuk, J. 2011. Towards Cognitively Plausible Game Playing Systems. *IEEE Computational Intelligence Magazine*, 6(2).

Parker-Holder, J.; Jiang, M.; Dennis, M.; Samvelyan, M.; Foerster, J.; Grefenstette, E.; and Rocktäschel, T. 2022. Evolving curricula with regret-based environment design. In *International Conference on Machine Learning*, 17473–17498. PMLR.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; and Guez et al., A. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419): 1140–1144.

Silver, T.; Allen, K. R.; Lew, A. K.; Kaelbling, L. P.; and Tenenbaum, J. 2020. Few-shot bayesian imitation learning with logical program policies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 10251–10258.

Tian, Y.; Huang, T.; Liu, M.; Jiang, D.; Spangher, A.; Chen, M.; May, J.; and Peng, N. 2024. Are Large Language Models Capable of Generating Human-Level Narratives? In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 17659–17681. Miami, Florida, USA: Association for Computational Linguistics.

Young, K.; and Tian, T. 2019. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*.

Zhu, J.; Awiszus, M.; Cook, M.; Dockhorn, A.; Eberhardinger, M.; Loiacono, D.; Lucas, S. M.; Matran-Fernandez, A.; Liebana, D. P.; Thompson, T.; et al. 2023. Explainable AI for Games. *Human-Game AI Interaction*, 73.