InnateCoder: Learning Programmatic Options with Foundation Models

Rubens O. Moraes[1], Quazi A. Sadmine[2,3], Hendrik Baier[4,5], and Levi H. S. Lelis[2,3]

[1] Departamento de Informática, Universidade Federal de Viçosa
[2] Department of Computing Science, University of Alberta
[3] Alberta Machine Intelligence Institute (Amii)
[4] Information Systems, Eindhoven University of Technology
[5] Centrum Wiskunde & Informatica, Amsterdam
h.j.s.baier@tue.nl

This is an "encore abstract" submission based on a paper of the same name published at IJCAI 2025 [3] and its extended arXiv version [4].

Motivation. Deep reinforcement learning (DRL) has shown remarkable progress across domains, from games to robotics. However, a fundamental limitation remains: Outside of transfer learning settings, agents typically start from scratch, without built-in skills or prior guidance. Standard DRL systems initialize with random neural network weights, forcing agents to painstakingly learn even the most basic behaviors through interaction. This *tabula rasa* approach suffers from sample inefficiency, as enormous amounts of experience are required before meaningful progress is achieved. Guided by advances in large foundation models, this paper introduces **InnateCoder** (IC), a system that leverages programmatic representations of policies [8] and the human knowledge encoded in foundation models (FMs) to provide agents with "innate skills" before training begins.

Method. The central idea behind InnateCoder is to extract *programmatic options* - temporally extended behaviors — from code generated by an FM. Given a natural-language task description and the grammar of a domain-specific language (DSL), the FM produces candidate programs. These programs may not themselves solve the task zero-shot, but they often contain useful subroutines encoding reusable behavioral fragments. IC decomposes programs into a diverse library of candidate options, which are routines the agent can call to control behavior over multiple steps [7]. This equips the agent with domain-relevant skills before any environment interaction. IC's workflow consists of three stages.

1. Option learning. The FM is prompted with the task description and DSL, and generates programmatic candidate policies. InnateCoder extracts subtrees of these program's abstract syntax trees, filters them for behavioral diversity, and stores them as options that serve as the agent's innate skill set.
2. Semantic space induction. Traditional program synthesis in RL relies on *syntax space* exploration [1], where neighboring programs often represent equivalent behaviors, wasting search effort. InnateCoder instead defines a *semantic space*: new candidate policies are created by replacing sub-trees of existing programs with options from the FM-generated library, creating neighbors that are

more likely to encode genuinely distinct behaviors. **3. Policy search**. A stochastic hill-climbing algorithm starts from a random program and greedily explores both syntax and semantic neighborhoods. The search aims at a programmatic policy that maximizes expected return under the environment's reward function.

This approach departs from prior work in two significant ways. First, options are discovered zero-shot from a foundation model before training begins, rather than through interaction with the environment. Second, semantic search guided by these options enables efficient exploration of vast policy spaces containing thousands of candidate behaviors — well beyond the reach of conventional option-learning methods.

Results. We evaluate InnateCoder in two challenging benchmarks: MicroRTS [5], a real-time strategy game requiring long-term planning across resource collection, building, and combat; and Karel the Robot [6], a program synthesis and RL benchmark with diverse grid-world tasks. Across both domains, IC demonstrates superior sample efficiency compared to stochastic hill-climbing (SHC) in syntax space and to baselines that learn options either from randomly sampled programs, or online from experience [2]. In MicroRTS, IC achieves higher win rates in fewer games. In Karel, IC outperforms SHC and matches or exceeds the baselines on tasks such as DoorKey, Harvester, and Snake. A key insight is that the FM baseline — directly executing the best programs produced by the foundation model — performed poorly. However, the subroutines extracted from these weak programs provided crucial building blocks for InnateCoder, supporting the hypothesis that FMs are valuable as providers of partial skills, even when their direct solutions fail. Furthermore, results were robust across different FM families, including GPT-4o and Llama-3.1, and competitive with handcrafted and DRL policies from previous MicroRTS competition winners. Even smaller models such as GPT-3.5 produced competitive results, demonstrating accessibility for groups without access to the most powerful FMs. Because all FM queries occur during preprocessing, the approach remains computationally lightweight.

In summary, InnateCoder delivers three contributions. First, it introduces zero-shot option generation from FMs, equipping agents with domain-relevant skills before training. Second, it enables semantic search via programmatic options, yielding substantial improvements in sample efficiency over syntax-based search. Third, it produces policies that are competitive with or superior to state-of-the-art programmatic and neural methods, while remaining efficient and lightweight. By utilizing the general knowledge and coding abilities of foundation models for reinforcement learning, InnateCoder offers a practical and scalable framework for accelerating the learning of effective policies.

Acknowledgments. This research was supported by Canada's NSERC, the CIFAR AI Chairs program, and Brazil's CAPES. It has also received funding from the Dutch research programme NWA ORC 2020 (NWA.1389.20.251), and from the EU's Horizon Europe programme (101120406).

Disclosure of Interests. The authors have no competing interests to declare.

References

1. Carvalho, T.H., Tjhia, K., Lelis, L.H.S.: Reclaiming the source of programmatic policies: Programmatic versus latent spaces. In: The Twelfth International Conference on Learning Representations (2024)
2. Moraes, R.O., Lelis, L.H.S.: Searching for programmatic policies in semantic spaces. In: Proceedings of the International Joint Conference on Artificial Intelligence (2024). https://doi.org/10.24963/ijcai.2024/662, https://doi.org/10.24963/ijcai.2024/662
3. Moraes, R.O., Sadmine, Q.A., Baier, H., Lelis, L.H.S.: Innatecoder: Learning programmatic options with foundation models. In: 34th International Joint Conference on Artificial Intelligence (IJCAI 2025). Accepted. (2025)
4. Moraes, R.O., Sadmine, Q.A., Baier, H., Lelis, L.H.S.: Innatecoder: Learning programmatic options with foundational models (extended version) (2025), https://arxiv.org/abs/2505.12508
5. Ontañón, S.: Combinatorial multi-armed bandits for real-time strategy games. Journal of Artificial Intelligence Research **58**, 665–702 (2017)
6. Pattis, R.E.: Karel the robot: a gentle introduction to the art of programming. John Wiley & Sons (1994)
7. Sutton, R.S., Precup, D., Singh, S.: Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial intelligence **112**(1-2), 181–211 (1999)
8. Trivedi, D., Zhang, J., Sun, S., Lim, J.J.: Learning to synthesize programs as interpretable and generalizable policies. In: Advances in Neural Information Processing Systems. pp. 25146–25163 (2021)