Realizing executable programs from an abstraction of the policy

Senne Deproost
senne.deproost@vub.be

Ann Nowé
ann.nowe@vub.be

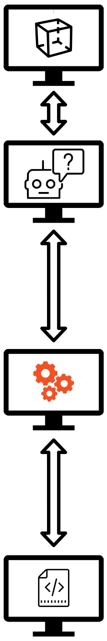
Approaches in literature

- **Template-based parametric search [1]**
  - RL policy guides search towards good parameters within the program
  - Structure of the program is fixed
- **Program latent space [2]**
  - Encode embeddings with variational autoencoder so new programs can be generated
  - Requires set of program execution traces
- **Syntax tree via genetic algorithm [3]**
  - Using evolutionary algorithm to generate the abstract syntax tree of the program
  - Slow convergence

Open gaps

- Automatic state space demarcation
- Deepening statements based on expression-simplicity tradeoff
- Guarantees of safe operation
- Executability guarantee
- Different target languages

PROCESS STEP | **METHOD** | **STAKEHOLDER**



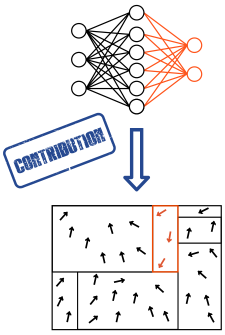
Simulate

Domain knowlege demarcates **operational regions** in the state space. The excluded space should be communicated as well to inform stakeholders downstream.

Multi-domain Experts

Learn

We assume the agent has a critic network to **calculate Q values** so we can use **reward-driven distillation** instead of regular behavioral cloning with a state-action set. This information could be used to aid the construction of the **non-differential programmatic representation** of the policy. Aditional components can be incorporated in the explanation to the user (weights, logits, ...).



ML Engineer

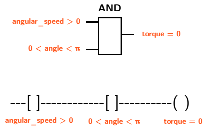
Abstract

The policy of the agent is captured in a **human-readable abstract**, adaptable by an expert. A graph-based representation of state space regions and the actions leading to transition between them would capture the **technical details** to motivate the behaviour. This will not be straightforward explainable.
The control engineer could interpret the routing throughout the nodes and remove undesired transitions where needed. The representation be further simplified by ommiting subrgaphs that lead to quasi-similar behaviour.

Control Engineer

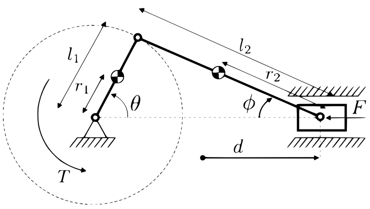
Express

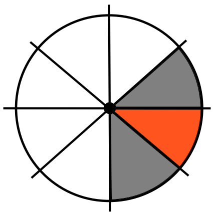
The policy is represented in a **domain-specific language** (DSL) the end user can interact with. This allows for further adaptations by the operators during unforseen situations. Since the programmatic representation was **expressed** from the abstract representation, linked hardware code could be realized as well.



Operators

Slider crank validation





The slider crank setup is a mechanical system where we translate a linear motion into a rotational one. The difficulty comes from the changing goal angular speed as well as the introduction of other forces (e.g. a spring).

To find a good policy, the **demarcation** of the state space should be more detailed in regions where change in behaviour is more crucial. Within, a programmatic policy would need to be more expressive to better mimic a trained RL agent.

Sources

[1] A. Verma, V. Murali, R. Singh, P. Kohli, and S. Chaudhuri, ‘Programmatically Interpretable Reinforcement Learning’, in Proceedings of the 35th International Conference on Machine Learning, PMLR, Jul. 2018, pp. 5045–5054.

[2] D. Hein, S. Udluft, and T. A. Runkler, ‘Interpretable policies for reinforcement learning by genetic programming’, Engineering Applications of Artificial Intelligence, vol. 76, pp. 158–169, Nov. 2018.

[3] D. Trivedi, J. Zhang, S.-H. Sun, and J. J. Lim, ‘Learning to Synthesize Programs as Interpretable and Generalizable Policies’. arXiv, Jan. 31, 2022.

VUB ARTIFICIAL INTELLIGENCE RESEARCH GROUP

Research funded by the CTRLxAI FWO SBO project [S007723N]
Presented at BeNeRL workshop 2024, Amsterdam