

Towards Verification-Driven Control Reinforcement Learning

Stefan Mitsch, School of Computing, DePaul University

Introduction

The increasing use of machine learning in systems development means that **guarantees about the behavior of learned components must be a vital part of the development process**. In autonomous systems, these types of guarantees are not properties of a neural network in isolation but **require analyzing a neural network's interaction with other components and the operating environment**. Open-loop verification addresses neural networks in isolation and is thus incapable of providing the required guarantees; closed-loop verification tools offer limited generality (e.g., finite time analysis) and include extensive correctness-critical code. Moreover, **separating verification activities from development activities makes cross-fertilization challenging**. We sketch a framework to address these gaps with a logic-based approach that **combines theorem proving with runtime monitoring, neural network verification, and reward shaping** for reinforcement learning.

This material is based upon work supported by the National Science Foundation under Grant No. CCF2427581.

From Partial to Full Specification

- **Analyze scenarios** provide concrete scenarios $P_i(\mathbf{x})$ to fill in the neural network response $Q_i(\mathbf{x}, \mathbf{u})$ from neural network analysis
- **Analyze action space** provide concrete actions $Q_i(\mathbf{x}, \mathbf{u})$ to fill in the scenarios from neural network input regions that elicit these responses, or from control envelope synthesis techniques [KLMP24] that generate the largest safe envelopes

Neural Network Control Envelopes

System Model. Learned controller: function $\mathbf{u} = nn(\mathbf{x})$ from state \mathbf{x} to control output \mathbf{u} ; interacts periodically with latency at most T with a continuous time-space environment, possibly faces adversarial inputs:

$$\begin{aligned} sys \equiv & (\mathbf{u} := nn(\mathbf{x}); \\ & (\mathbf{v} := *)^d; \\ & t := 0; \{ \mathbf{x}' = f(\mathbf{x}, \mathbf{u}, \mathbf{v}), t' = 1 \ \& \ t \leq T \}^* \end{aligned}$$

Control Envelope. A control envelope

$$nn(\mathbf{x}) \equiv \bigcup_i (?P_i(\mathbf{x}); \mathbf{u} := *; ?Q_i(\mathbf{x}, \mathbf{u}))$$

defines expected outputs $Q_i(\mathbf{x}, \mathbf{u})$ in certain input regions $P_i(\mathbf{x})$ (“what do you do when”) or, conversely, the required input to elicit certain output behavior (“when do you do what”) [BM22].

Safety. Safety statements about model sys are expressed in KeYmaera X [FMQ⁺15] formulas of the shape

$$Q(\mathbf{x}) \rightarrow [sys]P(\mathbf{x})$$

which expresses that all runs of the game sys satisfy postcondition $P(\mathbf{x})$ when started in states that satisfy initial conditions $Q(\mathbf{x})$.

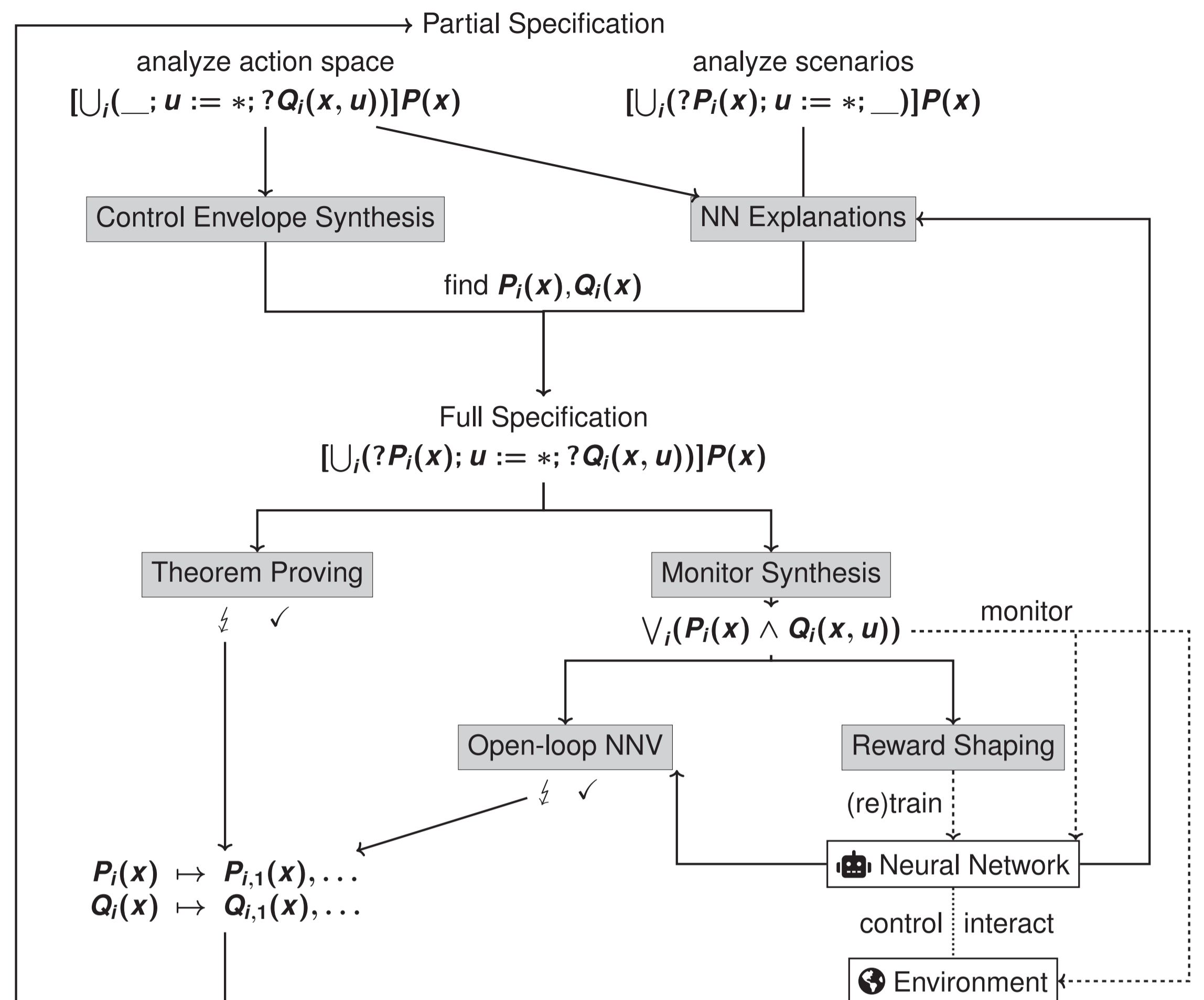
Runtime Verification

The differential equations and nondeterministic alternatives in hybrid programs make them an expressive specification language, but for execution require computationally expensive methods (e.g., online reachability analysis). Much of this computational complexity can be shifted offline through the use of ModelPlex [MP16]: ModelPlex formulas are quantifier- and modality-free, and are thus computationally inexpensive to evaluate from concrete measurements at runtime, which makes them attractive for reinforcement learning and neural network verification.

Reward Shaping and Neural Network Verification

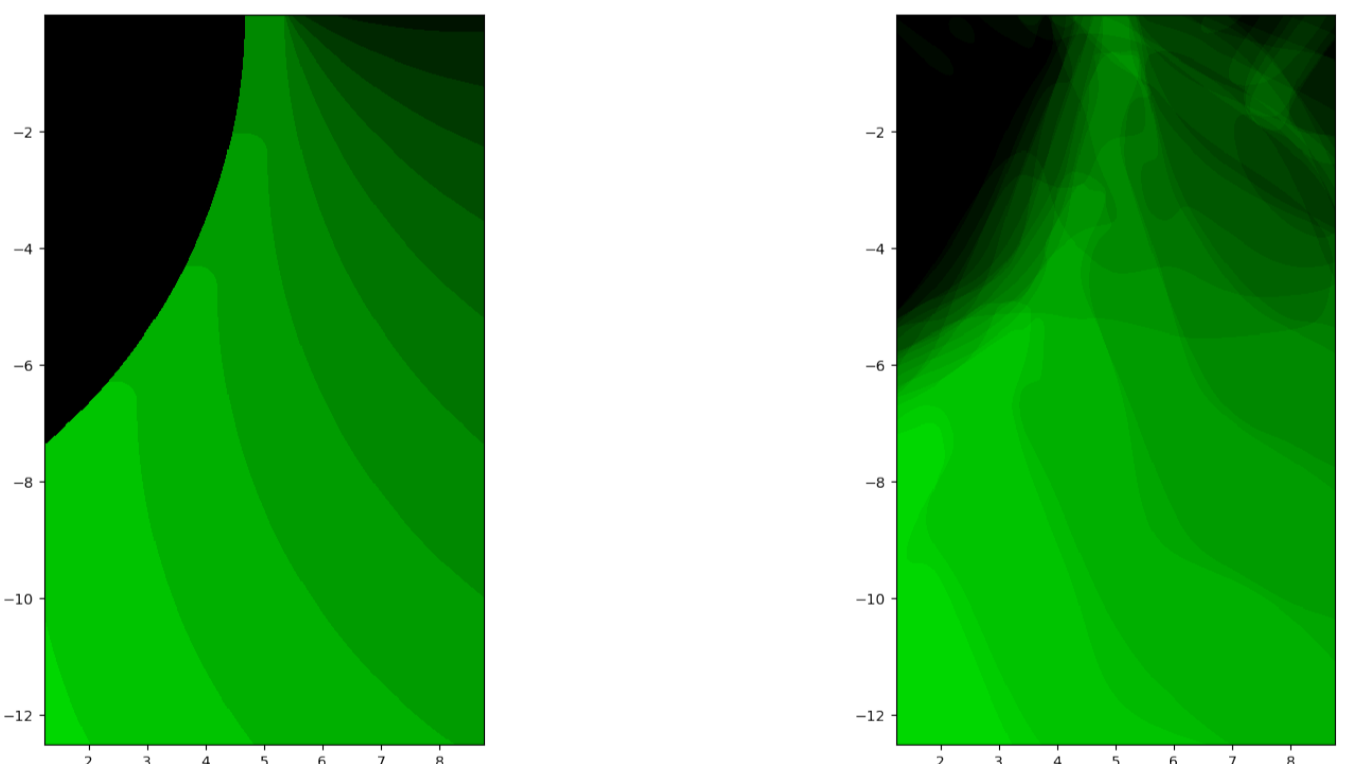
To avoid extensive interference from monitors and fallback control, reward shaping [QM23] turns formal models and monitors into reward function components to steer the training process itself and produce learned controllers that operate safely within the constraints of the formal model. Open-loop neural network verification on the basis of synthesized monitors [TMP24] then proves compliance of the learned controller with the formal specification, or provides counterexample regions that can be combined with the synthesized runtime monitors to shield the system from executing unsafe actions.

Framework



Challenges

- **Extract a Control Envelope from a Neural Network** Finding and certifying control conditions requires integration of neural network analysis, control envelope synthesis, and theorem proving.
- **Sim-to-Real Transfer Monitoring** Detect when the safety-relevant margins of the learned controller become close to unsafety.
- **Critic Assessment** A formal foundation to characterize the difference between the conditions $P_i(\mathbf{x})$ of the formal model and the predictions of the critic paves the way towards automated assessment.



A challenge is determining whether the cause of a detected difference is an optimistic (unsafe) critic or a conservative (safe) control envelope.

- **Liveness Rewards** Proof insights from liveness proofs: a finite strategy of control choices can be used as a starting point for imitation learning, whereas a progress function can be used as a component of the reward function in reinforcement learning (agent gets reward for beating the liveness proof).

References

- [BM22] David Bayani and Stefan Mitsch, *Fanoos: Multi-resolution, multi-strength, interactive explanations for learned systems*, VMCAI, LNCS, vol. 13182, Springer, 2022, pp. 43–68.
- [FMQ⁺15] Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völpl, and André Platzer, *KeYmaera X: an axiomatic tactical theorem prover for hybrid systems*, CADE-25, LNCS, vol. 9195, Springer, 2015, pp. 527–538.
- [KLMP24] Aditi Kabra, Jonathan Laurent, Stefan Mitsch, and André Platzer, *CESAR: control envelope synthesis via angelic refinements*, TACAS, LNCS, vol. 14570, Springer, 2024, pp. 144–164.
- [MP16] Stefan Mitsch and André Platzer, *Modelplex: verified runtime validation of verified cyber-physical system models*, Formal Methods Syst. Des. 49 (2016), no. 1-2, 33–74.
- [QM23] Marian Qian and Stefan Mitsch, *Reward shaping from hybrid systems models in reinforcement learning*, NFM, LNCS, vol. 13903, Springer, 2023, pp. 122–139.
- [TMP24] Samuel Teuber, Stefan Mitsch, and André Platzer, *Provably safe neural network controllers via differential dynamic logic*, NIPS, Curran Associates, Inc., 2024.