LiVe 2022

6th Workshop on Learning in Verification

Informal proceedings

LiVe'22 was held as a satellite event of ETAPS in Munich on April 2, 2022.

Invited talk:

Armando Tacchella: *There is plenty of room at the bottom: verification (and repair) of small-scale learning models*

Abstract: With the growing popularity of machine learning, the quest for verifying data-driven models is attracting more and more attention, and researchers in automated verification are struggling to meet the scalability and expressivity demands imposed by the size and the complexity of state-of-the-art machine learning architectures. However, there are applications where relatively small-scale learning models are enough to achieve industry-standard performances, yet the issue of checking whether those models are reliable remains challenging. Furthermore, in these domains, verification is just half of the game: providing automated ways to repair models that are found to be faulty is also an important task in practice. In this talk, I will touch upon some research directions that I have pursued in the past decade, commenting the results and providing some connections with related efforts.

Programme:

- Session 1
  - 09:00 - 09:20 Opening
  - 09:20 - 09:40 Stefan Ratschan: **Learning Certificates for Properties of Continuous Dynamical Systems**
  - 09:40 - 10:00 Eirene V. Pandi, Earl T. Barr, Charles Sutton and Andrew D. Gordon: **Type Inference as Optimization**
- Session 2 (MDP+RL)
  - 10:30 - 10:50 Qisong Yang, Thiago D. Simão, Nils Jansen, Simon Tindemans and Matthijs T. J. Spaan: **Training and Transferring Safe Policies in Reinforcement Learning**
  - 10:50 - 11:10 Steven Carr, Nils Jansen, Sebastian Junges and Ufuk Topcu: **Verifiably Safe Reinforcement Learning for POMDPs via Shielding: Probabilistic Type Inference by Optimizing Logical and Natural Constraints**
  - 11:10 - 11:30 Linus Heck, Jip Spel, Sebastian Junges, Joshua Moerman and Joost-Pieter Katoen: **Gradient-Descent for Randomized Controllers under Partial Observability**
  - 11:30 - 11:50 Marnix Suilen, Thiago D. Simão, Nils Jansen and David Parker: **Anytime Learning and Verification of Uncertain Markov Decision Processes**
  - 11:50 - 12:10 Maximilian Weininger: **PAC Guarantees for unknown probabilities through sampling**
  - 12:10 - 12:30 Debraj Chakraborty, Damien Busatto-Gaston, Shibashis Guha, Guillermo A Pérez and Jean-François Raskin: **Safe Learning for Near-Optimal Scheduling**
- Session 3 (NN)
  - 14:00 - 15:00 *Invited talk - Armando Tacchella:* **There is plenty of room at the bottom: verification (and repair) of small-scale learning models**
  - 15:00 - 15:20 Marco Sälzer and Martin Lange: **Complexity and Decidability Results for Verification of Deep Learning Models**
  - 15:20 - 15:40 Christopher Brix and Lisa Pühl: **Binary-Search Tree Exploration in Verification of Neural Networks**
  - 15:40 - 16:00 Igor Khmelnitsky, Daniel Neider, Rajarshi Roy, Xuan Xie, Benoit Bardot, Benedikt Bollig, Alain Finkel, Serge Haddad, Martin Leucker and Lina Ye: **Formal Verification of Neural Networks by Learning Automata Models**
- Session 4 (Properties)
  - 16:30 - 16:50 Vahid Hashemi, Jan Kretinsky, Stefanie Mohr, Emmanouil Seferis: **Gaussian-Based Runtime Detection of Out-of-distribution Inputs for Neural Networks**
  - 16:50 - 17:10 Simon Lutz, Daniel Neider and Rajarshi Roy: **Learning Linear Temporal Formulas from Specification Sketches**
  - 17:10 - 17:30 Kush Grover, Fernando S. Barbosa, Jana Tumova, Jan Kretinsky: **Semantic Abstraction-Guided Motion Planning for scLTL Missions in Unknown Environments**

Learning Certificates for Properties of Continuous Dynamical Systems *Abstract*

Stefan Ratschan

Institute of Computer Science of the Czech Academy of Sciences

Inductive invariants and ranking functions play an important role in the verification of discrete systems, certifying, for example, reachability and termination properties of loops in computer programs. For continuous dynamical systems, similar certificates have been in use since the defense of Aleksandr Lyapunov's thesis in 1893. However, for most of the time since then, it has been a question of engineering ingenuity to come up with such certificates. This has changed in recent years, due to progress in the automatic computation of Lyapunov functions and barrier certificates in the continuous case, and of inductive invariants and ranking functions in the discrete case. It has also become clear that learning (i.e., generalization from special cases) can play an important role in computing such certificates. However, the respective development in the continuous and the discrete case has been largely independent, communication being largely restricted to the area of hybrid dynamical systems.

In the talk we will contribute to this communication[1],

– giving an short overview on the role of certificates for verifying properties of continuous dynamical systems [9, 5, 3], and on algorithms for computing them [6, 7, 10, 12],
– discussing the role of learning in such algorithms [11, 2, 1] and its advantages and downsides that we observed in our work [8], and
– presenting our ongoing work [4] on learning certificates in verification and control synthesis for continuous dynamical systems.

We will conclude the talk with a discussion of open questions in the area, relating the continuous case to the discrete one.

References

1. A. Abate, D. Ahmed, A. Edwards, M. Giacobbe, and A. Peruffo. Fossil: A software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, HSCC '21, New York, NY, USA, 2021. Association for Computing Machinery.
2. A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo. Formal synthesis of Lyapunov neural networks. *IEEE Control Systems Letters*, 5(3):773–778, 2020.

[1] The references included in this abstract should be seen just as illustrative examples, certainly being far from exhaustive.

3. C. Dawson, Z. Qin, S. Gao, and C. Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. In *Conference on Robot Learning*, pages 1724–1735. PMLR, 2022.
4. J. Fejlek and S. Ratschan. Computation of stabilizing and relatively optimal feedback control laws based on demonstrations. *arXiv preprint arXiv:2011.12639*. submitted.
5. H. Han, M. Maghenem, and R. G. Sanfelice. Certifying the LTL formula p until q in hybrid systems. *arXiv preprint arXiv:2106.06455*, 2021.
6. P. Parrilo and S. Lall. Semidefinite programming relaxations and algebraic optimization in control. *European Journal of Control*, 9(2–3), 2003.
7. S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. In R. Alur and G. J. Pappas, editors, *HSCC'04*, number 2993 in LNCS. Springer, 2004.
8. S. Ratschan. Simulation based computation of certificates for safety of dynamical systems. In A. Abate and G. Geeraerts, editors, *Formal Modeling and Analysis of Timed Systems: 15th International Conference, FORMATS 2017*, volume 10419, pages 303–317. Springer International Publishing, 2017.
9. S. Ratschan. Converse theorems for safety and barrier certificates. *IEEE Trans. on Automatic Control*, 63(8):2628–2632, 2018.
10. S. Ratschan and Z. She. Providing a basin of attraction to a target region of polynomial systems by computation of Lyapunov-like functions. *SIAM Journal on Control and Optimization*, 48(7):4377–4394, 2010.
11. H. Ravanbakhsh and S. Sankaranarayanan. Learning control Lyapunov functions from counterexamples and demonstrations. *Autonomous Robots*, pages 1–33, 2018.
12. M. A. B. Sassi, S. Sankaranarayanan, X. Chen, and E. Ábrahám. Linear relaxations of polynomial positivity for polynomial lyapunov function synthesis. *IMA Journal of Mathematical Control and Information*, page dnv003, 2015.

Type Inference as Optimization[⋆]

Eirene V. Pandi[1], Earl T. Barr[2], Andrew D. Gordon[1,3], and Charles Sutton[1]

[1] University of Edinburgh, Edinburgh, UK
[2] University College London, London, UK
[3] Microsoft Research Cambridge, Cambridge, UK

Optionally typed dynamic languages can permit multiple valid type assignments. When this happens, developers can prefer one valid type assignment over another because it better reflects how they think about the program and the problem it solves. Natural type inference (NTI) uses natural language text within source code, such as identifiers, to help choose valid programming language types. A growing body of techniques has been proposed for NTI. These techniques predict types; they seek to return natural type assignments (assignments that reflect developer preferences) while striving for correctness. They are empirically effective, but they are not sound by construction: they do not leverage programming language theory to formalize their algorithms and show correctness and termination. Filling this foundational gap is the purpose of this paper. We are the first to present a detailed algorithm for NTI that is validated with theorems and proofs. Valid type assignments obey logical constraints arising from type rules; natural type assignments obey natural constraints arising from the natural language text associated with a variable and its uses. The core intuition of this work is that logical and natural constraints can interact to speed finding a type valuation that 1. type checks (satisfies the logical constraints) and 2. is most natural. We formulate NTI as a joint optimization problem. To do this, we define a numerical relaxation over boolean logical constraints that give us a condition that we treat as a hard constraint, while simultaneously we minimize distance from natural constraints, which we treat as soft constraints for our optimization problem. Our main result, the first formal proof of soundness for natural type inference, is that our algorithm always terminates, either with an error or with a tuple that is guaranteed to be a type signature for its input.

Contributions: Formal Foundations for Natural Type Inference As the setting for our study, we define an exemplary type inference task as finding a type signature for an untyped function definition within a λ-calculus, whose types are defined by a global set of equations between type names and scalar, record, and function types. The operational semantics and type system satisfy preservation and progress properties.

Our contributions are as follows:

– We present a new algorithmic type system that given an expression yields logical and natural constraints. The algorithm is terminating and the logical

[⋆] This work was supported by Microsoft Research through its PhD Scholarship Programme.

constraints are sound and complete with respect to the declarative type system. Our overall task is finding a type signature for an untyped function definition, is equivalent to satisfying the logical constraint extracted from the function definition.

– We show how to combine a numerical relaxation of the logical constraints with probability distributions over the library of types to form a joint optimization problem. Firstly, we show how to relate the logical semantics and its relaxations. And then we present our key theorem where we shod that the optimizer is guaranteed to terminate with the optimal solution to the natural constraints that satisfies the logical constraints.
– We describe an overall algorithm for natural type inference, building on the algorithmic type system and the constraint satisfaction algorithm. By the correctness theorem, the algorithm always terminates, either with an error, or with a tuple that is guaranteed to be a type signature for its input.

This work is the first to formalize and prove termination and soundness for a natural type inference algorithm. Our specific algorithm deals with ambiguities arising from overloading, dot-notation, and structural equality of type names. It provides formal foundations for OptTyper [1] and shows that the resulting type signatures are sound. Some of our definitions, including logical and natural constraints, the continuous relaxations, and the core optimization problem are based on [1], but all the theorems of this paper are new, as is the formulation of an algorithm for type-checking function definitions in a typed λ-calculus.

As our literature review makes clear, all work in learning-based type inference to date focuses on formalising their method, none states theorems or formally proves its approach to be sound by construction, and all are empirically validated. The present work rises to address this challenge. We have formally developed an inference system, from the ground up, that assigns type names to arbitrary type structures. This type system captures key aspects of type inference in optionally typed languages used in industry, like TypeScript and Python. Crucially, we have validated this system by theorem and proof. This work is the first to formalize and prove termination and soundness for a natural type inference algorithm.

Still, there are limitations that can be addressed in future work. Our algorithm only chooses types from the given library of type definitions. Hence, an input expression will be rejected if it needs a record or function type missing from the library. Another limitation is that we ignore field names when generating natural constraints. We expect it would be straightforward to extend the inference algorithm to augment the given library with type equations defining additional record or function types, as needed, and to take field names into account.

A bigger challenge is to extend natural type inference to features including subtyping, parametric polymorphism, and intersection and union types, important for TypeScript and other languages.

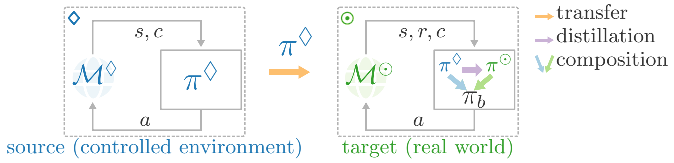
References

1. Pandi, E.V., Barr, E.T., Gordon, A.D., Sutton, C.: Opttyper: Probabilistic type inference by optimising logical and natural constraints. CoRR **abs/2004.00348** (2020)

Training and Transferring Safe Policies in Reinforcement Learning[⋆]

Qisong Yang[⋆⋆1], Thiago D. Simão[⋆⋆2],
Nils Jansen[2], Simon H. Tindemans[1], and Matthijs T. J. Spaan[1]

[1] Delft University of Technology, The Netherlands
{q.yang, s.h.tindemans, m.t.j.spaan}@tudelft.nl
[2] Radboud University, Nijmegen, The Netherlands,
{thiago.simao, nils.jansen}@ru.nl



Despite the numerous achievements of reinforcement learning [RL; 19, 14], safety still prevents the wide adoption of RL [4]. The lack of knowledge about the environment forces standard agents to rely on trial and error strategies. However, this approach is incompatible with safety-critical scenarios [5]. For instance, while operating a power network, an agent trying random actions could cause a blackout, which is strictly unacceptable [12, 18].

Developments in safe RL have allowed learning policies that respect safety constraints expressed by a constrained Markov decision process [CMDP; 3]. For instance, SAC-Lagrangian [6] combines the Soft Actor-Critic [SAC; 7, 8] algorithm with Lagrangian methods to learn a safe policy in an off-policy way. This method solves high-dimensional problems achieving a sample complexity lower than its on-policy counterparts. Unfortunately, it only finds a safe policy at the end of the training process and may be unsafe while learning.

Some knowledge about the safety dynamics can ensure safety during learning. One can precompute unsafe behavior and mask unsafe actions using a so-called shield [2, 10], or start from an initially safe baseline policy and gradually improve its performance while remaining safe [1, 21, 23]. However, this approach may require many interactions with the environment before it finds a reasonable policy [24]. Moreover, reusing a pre-trained policy can be harmful since the agent faces a new trajectory distribution as the policy changes [9]. Therefore, we focus on *how to efficiently solve a task without violating the safety constraints.*

Drawing inspiration from transfer learning [20],we propose to transfer a policy, the safe guide (SaGui, see title figure), from the source task (◊) to the target

[⋆] This research is partially funded by the Netherlands Organisation for Scientific Research (NWO), as part of the grant NWA.1160.18.238: "PrimaVera".
[⋆⋆] Equal contribution.

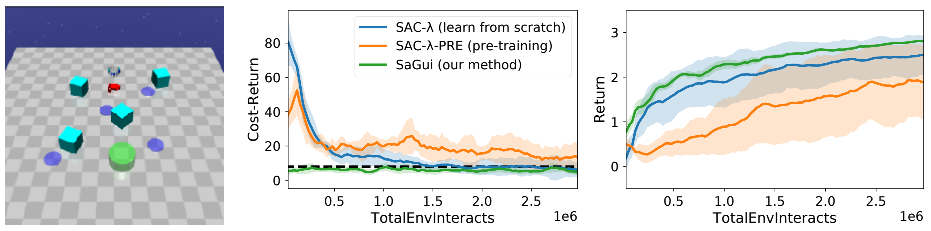


Fig. 1. Empirical analysis on the Safety-Gym Engine [15] showing the safety-related *cost-return* (middle) and task-related *return* (right) of the policy interacting with the target task.

task ($\odot$). This approach has three major steps: *i*) training the SaGui policy and *transferring* it to the target task; *ii*) *distilling* the guide's policy into a student policy, and *iii*) *composing* a behavior policy that balances safe exploration (using the guide) and exploitation (using the student).

To train the SaGui policy, we consider the reward-free constrained RL framework [13], where the agent only observes the cost function, and it does not have access to the reward function. This task-agnostic approach allows us to train a guide even when we do not know the reward of the target task, and this guide can be useful for different reward functions. Inspired by advances in robotics where an agent is trained under strict supervision, we assume the source task is a simulated/controlled environment [16, 22]. Therefore, safety is not required while training the SaGui policy. Nevertheless, we consider this environment provides enough safety information, such that any policy that is safe on the source task is also safe when deployed on the target task [11, 17]. Once the target's reward is revealed, the SaGui policy safely collects the initial trajectories, and we start training the student's policy. To ensure the new policy quickly learns how to act safely, we also employ a policy distillation method, encouraging the student to imitate the SaGui policy.

Inspired by [20] we consider two *safety-transfer metrics* to evaluate this approach: *safety jump-start*, the initial reduction in the expected cost-return of an agent trained using the source knowledge compared to the expected cost-return of an agent learning from scratch; and *safety speed-up*, the difference in the amount of interactions required to reach the safety threshold.

The empirical analysis (Fig. 1) shows that this method provides a large safety jump-start, almost completely preventing the violation of the safety constraints on the target task. It also shows the exploration benefits of SaGui, which allows the agent to solve the target task faster than agents learning with a naive guide.

Some directions for future work include: improving the exploration capabilities of the guide using techniques from reward-free RL; investigating how to compose policies with multiple guides; and evaluating the distillation approach from a curriculum learning perspective in an online setting.

Reference

[1] Achiam, J., Held, D., Tamar, A., Abbeel, P.: Constrained Policy Optimization. In: Proceedings of the 34th International Conference on Machine Learning. pp. 22–31. PMLR (2017)

[2] Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe Reinforcement Learning via Shielding. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence. pp. 2669–2678. AAAI Press (2018)

[3] Altman, E.: Constrained Markov decision processes, vol. 7. CRC Press (1999)

[4] Dulac-Arnold, G., Levine, N., Mankowitz, D.J., Li, J., Paduraru, C., Gowal, S., Hester, T.: Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. Machine Learning **110**(9), 2419–2468 (2021)

[5] García, J., Fernández, F.: A Comprehensive Survey on Safe Reinforcement Learning. The Journal of Machine Learning Research **16**(1), 1437–1480 (2015)

[6] Ha, S., Xu, P., Tan, Z., Levine, S., Tan, J.: Learning to Walk in the Real World with Minimal Human Effort. In: Proceedings of the 2020 Conference on Robot Learning. pp. 1110–1120. PMLR (2020)

[7] Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In: Proceedings of the 35th International Conference on Machine Learning. pp. 1861–1870. PMLR (2018)

[8] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., Levine, S.: Soft Actor-Critic Algorithms and Applications (2018), *arXiv preprint arXiv:1812.05905*

[9] Igl, M., Farquhar, G., Luketina, J., Boehmer, W., Whiteson, S.: Transient Non-stationarity and Generalisation in Deep Reinforcement Learning. In: 9th International Conference on Learning Representations. pp. 1–9. OpenReview.net (2021)

[10] Jansen, N., Könighofer, B., Junges, S., Serban, A., Bloem, R.: Safe Reinforcement Learning Using Probabilistic Shields (Invited Paper). In: 31st International Conference on Concurrency Theory. pp. 1–16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020)

[11] Li, L., Walsh, T.J., Littman, M.L.: Towards a Unified Theory of State Abstraction for MDPs. In: International Symposium on Artificial Intelligence and Mathematics. pp. 1–10. ISAIM (2006)

[12] Marot, A., Donnot, B., Romero, C., Donon, B., Lerousseau, M., Veyrin-Forrer, L., Guyon, I.: Learning to run a power network challenge for training topology controllers. Electric Power Systems Research **189**, 106635 (2020)

[13] Miryoosefi, S., Jin, C.: A Simple Reward-free Approach to Constrained Reinforcement Learning (2021), *arXiv preprint arXiv:2107.05216*

[14] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M.A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra,

D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)

[15] Ray, A., Achiam, J., Amodei, D.: Benchmarking Safe Exploration in Deep Reinforcement Learning (2019), https://cdn.openai.com/safexp-short.pdf

[16] Schuitema, E., Wisse, M., Ramakers, T., Jonker, P.: The design of LEO: A 2D bipedal walking robot for online autonomous Reinforcement Learning. In: International Conference on Intelligent Robots and Systems. pp. 3238–3243. IEEE (2010)

[17] Simão, T.D., Jansen, N., Spaan, M.T.J.: AlwaysSafe: Reinforcement Learning Without Safety Constraint Violations During Training. In: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS). p. 1226–1235. IFAAMAS (2021)

[18] Subramanian, M., Viebahn, J., Tindemans, S.H., Donnot, B., Marot, A.: Exploring grid topology reconfiguration using a simple deep reinforcement learning approach. In: 2021 IEEE Madrid PowerTech. pp. 1–6. IEEE (2021)

[19] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, vol. 2. MIT press (2018)

[20] Taylor, M.E., Stone, P.: Transfer Learning for Reinforcement Learning Domains: A Survey. The Journal of Machine Learning Research **10**(56), 1633–1685 (2009)

[21] Tessler, C., Mankowitz, D.J., Mannor, S.: Reward Constrained Policy Optimization. In: 7th International Conference on Learning Representations. pp. 1–9. OpenReview.net (2019)

[22] Xie, Z., Clary, P., Dao, J., Morais, P., Hurst, J.W., Panne, M.v.d.: Learning Locomotion Skills for Cassie: Iterative Design and Sim-to-Real. In: 3rd Annual Conference on Robot Learning. pp. 317–329. PMLR (2019)

[23] Yang, T., Rosca, J., Narasimhan, K., Ramadge, P.J.: Projection-Based Constrained Policy Optimization. In: 8th International Conference on Learning Representations. pp. 1–10. OpenReview.net (2020)

[24] Zanger, M.A., Daaboul, K., Zöllner, J.M.: Safe Continuous Control with Constrained Model-Based Policy Optimization. In: IEEE/RSJ International Conference on Intelligent Robots and Systems IROS. pp. 3512–3519. IEEE (2021)

Verifiably Safe Reinforcement Learning for POMDPs via Shielding

Steven Carr[1], Nils Jansen[2], Sebastian Junges[2], and Ufuk Topcu[1]

[1] The University of Texas at Austin, USA
[2] Radboud University, Nijmegen, The Netherlands

Reinforcement learning (RL) [30] is a machine learning technique for decision-making in dynamical environments. An *RL agent* explores its environment by taking *actions* and perceiving feedback signals, usually *rewards* and *observations* on the current system state. With success stories such as DeepMind's AlphaGo [27] and AlphaZero [28], RL nowadays reaches into areas such as robotics [15, 1], autonomous driving [26], or healthcare [34]. One of the major limitations for RL agents operating in safety-critical environments is the high cost of failure. Without side-information, an RL agent explores the effects of actions – often selected randomly such as those in state-of-the-art policy-gradient methods [22] – and will thus inevitably select unsafe actions that potentially cause harm to the agent or its environment. Consequently, applications for RL are often restricted to games [19] or assume access to digital twins: high-fidelity simulations of realistic scenarios [31]. This problem of *unsafe exploration* has triggered research on the *safety* of RL [8]. "Safe RL" may refer to (1) changing ("*engineering*") the reward function [17] to encourage the agent to choose safe actions, (2) adding a second ("*constraining*") learning objective by means of a separate cost function [20], or (3) blocking ("*shielding*") unsafe actions at runtime [2].

The typical model to accommodate both uncertainties in actions (due to limitations of actuators) and uncertainties in observations (due to limited sensing) are partially observable Markov decision processes (POMDPs) [14]. We capture safety by reach-avoid specifications, a special case of temporal logic constraints [24].

To provide safety guarantees, we advocate using explicit side-information based on a *partial model of the environment.* In particular, we need to know the graph of a POMDP, while probabilities and rewards may remain unspecified [25]. Under the (necessary) assumption that such a partial model is available, we extract a *shield* that ensures verifiably safe, or *correct*, behavior of an RL agent with respect to the model and formal specifications such as temporal logic constraints [24]. Despite tremendous progress [23, 32, 29], model-based reasoning, and in particular verification, has limitations: Even if a POMDP is completely known, scalability remains a challenge. Already, whether for a POMDP there exists a policy that satisfies a temporal logic specification is undecidable [18]. Computing policies for qualitative reach-avoid specifications is EXPTIME-complete [5], but efficient methods based on satisfiability solvers show good empirical scalability [4, 21, 13].

We contribute the first and effective integration of shields computed via satisfiability solving [13] with various state-of-the-art RL algorithms from Tensorflow [9] and provide an extensive experimental evaluation. We show the following natural effects that arise from such a combination.

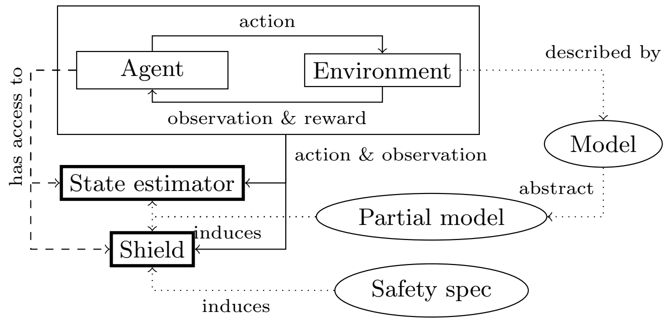


Fig. 1. RL for partial-observation and partial-model information accessible via a state estimator and a shield.

- *Safety during learning:* Exploration is only safe regarding reach-avoid specifications when the RL agent is provided with a shield. Even if the agent has access to the inherent state estimation of a shield, unsafe actions are chosen throughout all RL algorithms.
- *Safety after learning:* A *trained agent* that has been provided with an incentive to adhere to safety still behaves unsafe sometimes. Moreover, typical unwanted tradeoffs in settings with safety and (additional) performance objectives are avoided when (1) safety is (strictly) enforced via shields and (2) the agent focuses on performance.
- *RL convergence:* A shield not only ensures safety, but also significantly reduces the search space and the required amount of data for RL.

Fig. 1 shows the outline of our approach. We demonstrate effects and insights on shielded RL for POMDPs using several typical examples and provide detailed information on RL performance as well as videos showing the exploration and training process. To investigate to what extent more lightweight alternatives to a shield help RL, we experiment with a state estimator. This estimator uses the partial model to track in which states the model may be, based on the observed history. We show that, while the RL agent may indeed benefit from this additional information, the shield provides more safety and faster convergence than relying on just the state estimator. Finally, after learning, we may gently phase out a shield and still preserve the better performance of the shielded RL agent. Then, even an overly protective shield may help to bootstrap an RL agent.

Further related work. Several approaches to safe RL in combination with formal verification exist [10, 16, 2, 12, 7, 3]. These approaches either rely on shielding, or guide the RL agent to satisfy temporal logic constraints. However, none of these approaches take our key problem of partial observability into account. Recent approaches to find safe policies for POMDPs with partial model knowledge do not consider reinforcement learning [6]. Recent deep reinforcement learning approaches for POMDPs, including those that employ recurrent neural networks [11, 33], generate high quality policies with sufficient data but do not guarantee safety.

References

1. Abbeel, P., Coates, A., Quigley, M., Ng, A.Y.: An application of reinforcement learning to aerobatic helicopter flight. In: NIPS. pp. 1–8. MIT Press (2006)
2. Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe reinforcement learning via shielding. In: AAAI. AAAI Press (2018)
3. Bouton, M., Karlsson, J., Nakhaei, A., Fujimura, K., Kochenderfer, M.J., Tumova, J.: Reinforcement learning with probabilistic guarantees for autonomous driving. CoRR **abs/1904.07189** (2019)
4. Chatterjee, K., Chmelik, M., Davies, J.: A symbolic sat-based algorithm for almost-sure reachability with small strategies in pomdps. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). pp. 3225–3232. AAAI Press (2016)
5. Chatterjee, K., Chmelík, M., Gupta, R., Kanodia, A.: Qualitative analysis of pomdps with temporal logic specifications for robotics applications. In: International Conference on Robotics and Automation (ICRA). pp. 325–330. IEEE (2015)
6. Cubuktepe, M., Jansen, N., Junges, S., Marandi, A., Suilen, M., Topcu, U.: Robust finite-state controllers for uncertain pomdps. In: AAAI. pp. 11792–11800. AAAI Press (2021)
7. Fulton, N., Platzer, A.: Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In: AAAI. AAAI Press (2018)
8. Garcıa, J., Fernández, F.: A comprehensive survey on safe reinforcement learning. Journal of Machine Learning Research **16**(1), 1437–1480 (2015)
9. Guadarrama, S., Korattikara, A., Ramirez, O., Castro, P., Holly, E., Fishman, S., Wang, K., Gonina, E., Wu, N., Kokiopoulou, E., Sbaiz, L., Smith, J., Bartók, G., Berent, J., Harris, C., Vanhoucke, V., Brevdo, E.: TF-Agents: A library for reinforcement learning in tensorflow. https://github.com/tensorflow/agents (2018), https://github.com/tensorflow/agents, [Online; accessed 25-June-2019]
10. Hasanbeig, M., Abate, A., Kroening, D.: Cautious reinforcement learning with logical constraints. In: AAMAS. pp. 483–491. International Foundation for Autonomous Agents and Multiagent Systems (2020)
11. Hausknecht, M.J., Stone, P.: Deep recurrent Q-learning for partially observable MDPs. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). pp. 29–37. AAAI Press (2015)
12. Jansen, N., Könighofer, B., Junges, S., Serban, A., Bloem, R.: Safe Reinforcement Learning Using Probabilistic Shields (Invited Paper). In: CONCUR. LIPIcs, vol. 171, pp. 3:1–3:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020)
13. Junges, S., Jansen, N., Seshia, S.A.: Enforcing almost-sure reachability in pomdps. In: CAV (2). LNCS, vol. 12760, pp. 602–625. Springer (2021)
14. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence **101**(1), 99–134 (1998)
15. Kober, J., Bagnell, J.A., Peters, J.: Reinforcement learning in robotics: A survey. Int. J. Robotics Res. **32**(11), 1238–1274 (2013)
16. Könighofer, B., Alshiekh, M., Bloem, R., Humphrey, L., Könighofer, R., Topcu, U., Wang, C.: Shield synthesis. Formal Methods in System Design **51**(2), 332–361 (2017)
17. Laud, A., DeJong, G.: The influence of reward on the speed of reinforcement learning: An analysis of shaping. Tech. rep. (2003)
18. Madani, O., Hanks, S., Condon, A.: On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). pp. 541–548. AAAI Press (1999)

19. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.A.: Playing atari with deep reinforcement learning. CoRR **abs/1312.5602** (2013)
20. Moldovan, T.M., Abbeel, P.: Safe exploration in Markov decision processes. In: ICML. icml.cc / Omnipress (2012)
21. Pandey, B., Rintanen, J.: Planning for partial observability by SAT and graph constraints. In: ICAPS. pp. 190–198. AAAI Press (2018)
22. Peters, J., Schaal, S.: Policy gradient methods for robotics. In: IROS. pp. 2219–2225. IEEE (2006)
23. Pineau, J., Gordon, G., Thrun, S.: Point-based value iteration: An anytime algorithm for pomdps. In: International Joint Conference on Artificial Intelligence (IJCAI). pp. 1025–1032 (2003)
24. Pnueli, A.: The temporal logic of programs. In: Foundations of Computer Science. pp. 46–57. IEEE (1977)
25. Raskin, J., Chatterjee, K., Doyen, L., Henzinger, T.A.: Algorithms for omega-regular games with imperfect information. Log. Methods Comput. Sci. **3**(3) (2007)
26. Sallab, A.E., Abdou, M., Perot, E., Yogamani, S.K.: Deep reinforcement learning framework for autonomous driving. CoRR **abs/1704.02532** (2017)
27. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T.P., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of go with deep neural networks and tree search. Nat. **529**(7587), 484–489 (2016)
28. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T.P., Simonyan, K., Hassabis, D.: Mastering chess and shogi by self-play with a general reinforcement learning algorithm. CoRR **abs/1712.01815** (2017)
29. Silver, D., Veness, J.: Monte-carlo planning in large pomdps. In: Advances in Neural Information Processing Systems (NIPS). pp. 2164–2172. MIT Press (2010)
30. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
31. Tao, F., Zhang, H., Liu, A., Nee, A.Y.C.: Digital twin in industry: State-of-the-art. IEEE Trans. Ind. Informatics **15**(4), 2405–2415 (2019)
32. Walraven, E., Spaan, M.: Accelerated vector pruning for optimal pomdp solvers. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). pp. 3672–3678. AAAI Press (2017)
33. Wierstra, D., Förster, A., Peters, J., Schmidhuber, J.: Solving deep memory pomdps with recurrent policy gradients. In: International Conference on Artificial Neural Networks (ICANN). pp. 697–706. Springer (2007)
34. Yu, C., Liu, J., Nemati, S.: Reinforcement learning in healthcare: A survey. CoRR **abs/1908.08796** (2019)

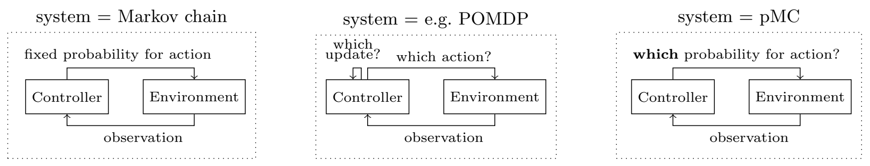
Gradient-Descent for Randomized Controllers under Partial Observability (Extended Abstract)

Linus Heck[1], Jip Spel[1](✉), Sebastian Junges[2], Joshua Moerman[1,3], and Joost-Pieter Katoen[1]

[1] RWTH Aachen University, Aachen, Germany *
[2] Radboud University, Nijmegen, the Netherlands
[3] Open University of the Netherlands, Heerlen, the Netherlands

Publication This is an extended abstract of [5].

Self-stabilizing protocols for distributed systems, and exponential back-off mechanisms in wireless networks, are two of the many examples where Markov chains (MCs) are used to model the probabilistic behaviour in closed-loop systems (Fig. 1(a)). Such systems are typically subject to temporal specifications, e.g., a self-stabilizing protocol should reach a stable configuration in few expected steps. Checking the models against these specifications can be efficiently done using state-of-the-art probabilistic model checking [6, 9].



(a) Verification of closed-loop systems. Memory of the controller is part of the system.

(b) Synthesis of controllers. Memory not fixed and thus not part of the system.

(c) Parameter synthesis for controllers. Memory fixed and part of the system.

Fig. 1. Verification and (syntax-guided) synthesis for controllers

One step beyond verification is the correct-by-construction synthesis of controllers for such systems via Partially Observable Markov Decision Processes (POMDPs) (Fig. 1(b)). In general, the synthesis for partial-information controllers is undecidable. In this work, we restrict the problem to controllers with a fixed memory structure (influencing the number of indistinguishable states) and a fixed set of potential actions that we want to randomize over.

This setting is useful, as in many systems one randomizes on purpose, e.g., in distributed protocols to break symmetry. Note that, the randomization is controllable, but selecting a (near-)optimal way to randomize is non-trivial.

The synthesis task reduces to appropriately randomizing in a system with a fixed topology (Fig. 1(c)). In this context, a controller selects a fixed set of actions (of the POMDP) $\alpha_1, \ldots, \alpha_n$ with probabilities $p_1, \ldots, p_n$. The aim is to synthesize a *realizable* controller, i.e., for indistinguishable states, the controller

* Supported by DFG RTG 2236 "UnRAVeL" and ERC AdG 787914 FRAPPANT.

must take action α_i with the same probability p_i. Synthesizing such controllers can be formally described [7] as feasibility synthesis in *parametric* Markov chains (pMCs), i.e., MCs with symbolic probabilities $p_1, \ldots, p_n$ [3,10]. The goal is to find values $u_1, \ldots, u_n$ for the parameters such that the MC satisfies a given property.

The challenge in applying parameter synthesis is twofold: the problem is ETR-complete [8], yet the number of parameters grows linear in the number of different observations and the number of actions available to the controller. For many real-life applications we must thus deal with thousands of parameters. This scale is out of reach for exact or complete methods [4]. However, heuristic methods have shown some promise [1, 2].

In this presentation, we show a principled way to evaluate gradients in parametric MCs. We characterize gradients as solutions of a linear equation system over the field over rational functions and alternatively as expected rewards of an automaton that is easily derived from the pMC at hand. Using the efficient computation of gradients, we evaluate both classical and adaptive gradient descent methods. Furthermore, we consider the classical gradient descent methods where we only respect the sign of the gradient. Finally, we investigate various methods to deal with restrictions on the parameter space (e.g. parameters should represent probabilities). Using an empirical evaluation, we determine which region restriction and gradient descent method performs best. We show that the resulting method often outperforms state-of-the-art methods QCQP [2] and PSO [1].

References

1. Chen, T., Hahn, E.M., Han, T., Kwiatkowska, M.Z., Qu, H., Zhang, L.: Model repair for Markov decision processes. In: TASE. IEEE (2013)
2. Cubuktepe, M., Jansen, N., Junges, S., Katoen, J.P., Topcu, U.: Synthesis in pMDPs: A tale of 1001 parameters. In: ATVA. LNCS, vol. 11138, pp. 160–176. Springer (2018)
3. Daws, C.: Symbolic and parametric model checking of discrete-time Markov chains. In: ICTAC. LNCS, vol. 3407, pp. 280–294. Springer (2004)
4. Dehnert, C., Junges, S., Jansen, N., Corzilius, F., Volk, M., Bruintjes, H., Katoen, J.P., Ábrahám, E.: Prophesy: A probabilistic parameter synthesis tool. In: CAV. LNCS, vol. 9206. Springer (2015)
5. Heck, L., Spel, J., Junges, S., Moerman, J., Katoen, J.: Gradient-descent for randomized controllers under partial observability. In: VMCAI. LNCS, vol. 13182, pp. 127–150. Springer (2022)
6. Hensel, C., Junges, S., Katoen, J.P., Quatmann, T., Volk, M.: The probabilistic model checker storm. CoRR **abs/2002.07080** (2020)
7. Junges, S., Jansen, N., Wimmer, R., Quatmann, T., Winterer, L., Katoen, J.P., Becker, B.: Finite-state controllers of POMDPs using parameter synthesis. In: UAI. AUAI Press (2018)
8. Junges, S., Katoen, J.P., Pérez, G.A., Winkler, T.: The complexity of reachability in parametric Markov decision processes. J. Comput. Syst. Sci. **119**, 183–210 (2021)
9. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: CAV. LNCS, vol. 6806. Springer (2011)
10. Lanotte, R., Maggiolo-Schettini, A., Troina, A.: Parametric probabilistic transition systems for system design and analysis. Formal Aspects Comput. **19**(1), 93–109 (2007)

Anytime Learning and Verification of Uncertain Markov Decision Processes

Marnix Suilen[1], Thiago D. Simão[1], Nils Jansen[1], and David Parker[2]

[1] Radboud University Nijmegen, The Netherlands
[2] University of Birmingham, UK

Markov decision processes (MDPs) are the standard model to reason about decision-making problems under probabilistic uncertainty. Safety-critical scenarios require assessments of correctness which can, for instance, be described by expected reward or temporal logic [6] specifications. A fundamental requirement for providing such correctness guarantees on MDPs is that probabilities are precisely given. Methods such as variants of model-based reinforcement learning [4] or PAC-learning [7] can learn MDPs by deriving *point estimates* of probabilities from data to satisfy this requirement. This derivation naturally carries the risk of statistical errors. Optimal policies are highly sensitive to small perturbations in transition probabilities, leading to sub-optimal outcomes such as a deterioration in performance [3, 2]. *Uncertain MDPs* (uMDPs) extend MDPs to incorporate such statistical errors by introducing an additional layer of uncertainty via *uncertainty sets* on the transition function [5, 9, 2].

Our approach. We study the problem of learning uncertain MDPs from data. Despite the power of uMDPs to capture broad notions of uncertainty, most work concerns the computation of policies that are robust against uncertainty. In contrast, learning such models is not well-studied yet. We propose an iterative learning method, outlined in Fig. 1, that is able to *adapt to new data* which may be inconsistent with prior assumptions. Furthermore, we recognize the inherent problem of sample-inefficiency and provide a combination of approaches to render the method more *sample efficient*. In particular, we learn intervals of probabilities for individual transitions, without assuming any specific structure for the uncertainty set. This Bayesian *anytime learning approach* employs intervals with linearly updating conjugate priors [8] and can iteratively improve upon a uMDP that approximates the true MDP we wish to learn. The method not only decreases the size of each interval, but also increases it again in case of a so-called *prior-data conflict* where new data suggests the actual probability lies outside the current interval.

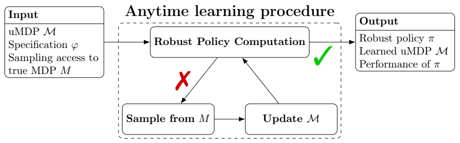
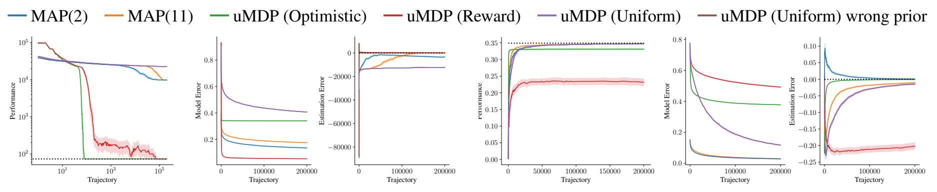


Fig. 1: Outline of the Procedure.

Key features. We summarize the key features of our learning method, and what sets it apart from other methods.

– *An anytime approach.* The ability to iteratively update intervals that are not necessarily subsets of each other allows us to design an *anytime learning*



(a) 30-state chain ($\mathrm{R}_{\mathrm{Min}}(\lozenge T)$). (b) 100-state grid ($\mathbb{P}_{\mathrm{Max}}(\neg\mathsf{trap}\,\mathsf{U}\,\mathsf{safe})$).

Fig. 2: Experimental results on the chain problem and a grid world.

approach. At any time, we may stop the learning and use verification techniques to compute a robust policy for the uMDP the process has yielded thus far. If this policy is not satisfactory, we continue learning towards a new uMDP that more faithfully captures the true MDP by including more data. Thereby, we ensure that the robust policy gets gradually closer to the standard policy for the true MDP, in contrast to e.g. PAC statistical model checking [1], which does not iteratively improve upon an intermediate model.

- *Specification-driven.* Our method features the possibility to learn transitions that only matter for a given specification. In particular, for reachability or expected reward (temporal logic) specifications, which require certain set of target states to be reached, we only learn and update transitions along paths towards these states.
- *Beyond expected reward.* Our method can work with any specification for which robust policies on uMDPs can be computed. In particular, we can work with general LTL specifications, which is intractable for reinforcement learning approaches [10].
- *Flexibility in the type of uncertainty sets.* We learn individual intervals for transition probabilities, instead of complete uncertainty sets. Only when computing a robust policy for the specification are the intervals are restricted to a *specific* type of uncertainty.
- *Efficient exploration policies.* In order to sample trajectories from the true MDP M as efficiently as possible, we propose designated exploration policies that exploit the fact that the intermediate models are uMDPs.

Key experimental results. From our experimental evaluation on several standard benchmarks (see Fig. 2 for an excerpt), we note that our method is effective at learning uMDPs that yield robust policies that are *conservative*, as seen by the *estimation error*: applying the robust policy to the true MDP gives a better performance than expected from the performance on the uMDP. In contrast, a comparison with *maximum a-posteriori* (MAP) estimation (which learns point estimates and derives a standard MDP) shows that MAP-estimation may be misleading by showing a better performance on the learned model than on the true MDP. Furthermore, in some cases, we note that our method is the only one able to converge to an optimal policy within a reasonable number of samples.

References

1. Ashok, P., Kretínský, J., Weininger, M.: PAC statistical model checking for markov decision processes and stochastic games. In: CAV (1). LNCS, vol. 11561, pp. 497–519. Springer (2019)
2. Goyal, V., Grand-Clement, J.: Robust markov decision process: Beyond rectangularity (2020)
3. Mannor, S., Simester, D., Sun, P., Tsitsiklis, J.N.: Bias and variance approximation in value function estimates. Manag. Sci. **53**(2), 308–322 (2007)
4. Moerland, T.M., Broekens, J., Jonker, C.M.: Model-based reinforcement learning: A survey. CoRR **abs/2006.16712** (2020)
5. Nilim, A., Ghaoui, L.E.: Robust control of markov decision processes with uncertain transition matrices. Oper. Res. **53**(5), 780–798 (2005)
6. Pnueli, A.: The Temporal Logic of Programs. In: FOCS. pp. 46–57. IEEE Computer Society (1977). https://doi.org/10.1109/SFCS.1977.32
7. Strehl, A.L., Li, L., Littman, M.L.: Reinforcement learning in finite mdps: PAC analysis. J. Mach. Learn. Res. **10**, 2413–2444 (2009)
8. Walter, G., Augustin, T.: Imprecision and prior-data conflict in generalized Bayesian inference. Journal of Statistical Theory and Practice **3**(1), 255–271 (2009)
9. Wiesemann, W., Kuhn, D., Rustem, B.: Robust markov decision processes. Math. Oper. Res. **38**(1), 153–183 (2013)
10. Yang, C., Littman, M.L., Carbin, M.: Reinforcement learning for general LTL objectives is intractable. CoRR **abs/2111.12679** (2021)

PAC guarantees for unknown probabilities through sampling

Maximilian Weininger

Technical University of Munich, Munich, Germany
maxi.weininger@tum.de

Introduction

Model checking of probabilistic systems is a well-known important topic, see e.g. [BK08, Chapter 10]. However, in real-world applications it often is difficult to know the exact transition probabilities of the considered system. We propose an approach that is able to reduce the case of a system with unknown transition probabilities to a classical model checking problem with known probabilities. Our algorithm provides a guarantee that the result is probably approximately correct (PAC, [Val84]), i.e. that with high probability (more than 1 minus a given tolerance δ), the resulting value is approximately correct (off by at most a given tolerance ε). Moreover, the algorithm is independent of the considered property.

Algorithm

The algorithm repeatedly applies two procedures:

– Infer confidence intervals from simulations.
– Infer best- and worst-case probabilities.

We will first discuss these two procedures and then comment on how to combine them.

Inferring confidence intervals from simulations The aim of the first procedure (from [AKW19]) is to find confidence intervals for the unknown transition probabilities. To this end, it simulates runs of the system and remembers how often every transition occurred. Using the two-sides variant of the Hoeffding bound [Hoe63], it can infer intervals for the transition probabilities that are correct with high probability (1 minus a given tolerance δ). For more details on this step, see [AKW19, Sec. 3.2]. We highlight two interesting details in this procedure which can potentially be improved by learning-based heuristics:

– It is non-trivial to decide when to abort a simulation that cannot reach the target. Several possibilities are discussed in [ADKW20, Sec. 4]. In particular, one can assume that the model structure is known (grey-box) or that the model is completely unknown (black-box).
– We need not have small confidence intervals for all transitions in the model. On the contrary, it suffices to only partially explore the model, using heuristics such as bounded real-time dynamic programming [BCC$^+$14].

Inferring best- and worst-case probabilities Probabilistic systems with intervals on the transition probabilities have been researched for several decades. Most commonly, they are identified by prepending "interval" (e.g. interval Markov chain [JL91,KU02]) or "bounded-parameter" (e.g. bounded-parameter Markov decision process [GLD00]). There are several interesting questions associated with them. For this work, we focus on the question what the "best" or "worst" possible choice of probabilities is for every interval. By doing this, we can specify two probabilistic systems with known probabilities: one that captures the best possible scenario that is possible given our current knowledge, and one for the worst possible instantiation of probabilities. Solving both these systems using classical algorithms, we obtain bounds on the value of the original system.

We infer these best- and worst-case probabilities using the reduction given in [WMK19]. In essence, we introduce a new state for every action of the original system. This state can model every valid instantiation of the intervals. If we assign this new state to ourselves, we obtain the best-case value; assigning it to an adversary yields the worst-case value.

Combining the procedures to obtain a PAC-guarantee Executing these two procedures one after the other, we reduce the problem of model checking a system with unknown probabilities to model checking two systems with known probabilities. With high probability, the intervals we inferred from the simulations are correct, and thus the bounds we obtain from solving these two systems are correct.

However, by executing the procedures once, we have no way to guarantee that the bounds are close to each other. Hence, we have to ensure that we simulate long enough such that the resulting confidence intervals are very small. Then, solving the best- and worst-case system gives bounds that are less than the given precision ε apart. One can naively calculate a number for how many simulations have to be run in order to obtain small enough confidence intervals; however, this number will be astronomically high and infeasible in practice [AKW19]. Instead, we can utilize the fact that many transitions are in fact irrelevant, and that under proper guidance, our simulations will explore only the relevant part of the state space. This can take a form similar to [AKW19, Algo. 7], replacing the guaranteed BVI phase with the inferring the best- and worst-case probabilities.

As a final thought, observe that if one is not interested in a PAC-guarantee, but rather a coarse estimate, applying the procedure once already yields bounds on the value with a probabilistic guarantee. Thus, using a reasonable amount of simulations, one might already obtain satisfactory results.

Acknowledgements

This abstract is about my ongoing work. It builds on concepts that I have developed together with Pranav Ashok, Tobias Meggendorfer and Jan Kretinsky [AKW19,WMK19]. The idea for combining the concepts recently arose as a solution, when we were dealing with the security problem of advanced persistent threats [WGMK21].

References

[ADKW20] Pranav Ashok, Przemyslaw Daca, Jan Kretínský, and Maximilian Weininger. Statistical model checking: Black or white? In *ISoLA (1)*, volume 12476 of *Lecture Notes in Computer Science*, pages 331–349. Springer, 2020.

[AKW19] Pranav Ashok, Jan Kretínský, and Maximilian Weininger. PAC statistical model checking for markov decision processes and stochastic games. In *CAV (1)*, volume 11561 of *Lecture Notes in Computer Science*, pages 497–519. Springer, 2019.

[BCC$^+$14] Tomás Brázdil, Krishnendu Chatterjee, Martin Chmelik, Vojtech Forejt, Jan Kretínský, Marta Z. Kwiatkowska, David Parker, and Mateusz Ujma. Verification of markov decision processes using learning algorithms. In *ATVA*, volume 8837 of *Lecture Notes in Computer Science*, pages 98–114. Springer, 2014.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[GLD00] Robert Givan, Sonia M. Leach, and Thomas L. Dean. Bounded-parameter markov decision processes. *Artif. Intell.*, 122(1-2):71–109, 2000.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[JL91] Bengt Jonsson and Kim Guldstrand Larsen. Specification and refinement of probabilistic processes. In *LICS*, pages 266–277, 1991.

[KU02] Igor Kozine and Lev V. Utkin. Interval-valued finite markov chains. *Reliable Computing*, 8(2):97–113, 2002.

[Val84] Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

[WGMK21] Maximilian Weininger, Kush Grover, Shruti Misra, and Jan Kretínský. Guaranteed trade-offs in dynamic information flow tracking games. In *CDC*, pages 3786–3793. IEEE, 2021.

[WMK19] Maximilian Weininger, Tobias Meggendorfer, and Jan Kretínský. Satisfiability bounds for ω-regular properties in bounded-parameter markov decision processes. In *CDC*, pages 2284–2291. IEEE, 2019.

Safe Learning for Near-Optimal Scheduling

Damien Busatto-Gaston[1], Debraj Chakraborty[1], Shibashis Guha[2], Guillermo A. Pérez[3], and Jean-François Raskin[1]

[1] Université libre de Bruxelles, Belgium
[2] Tata Institute of Fundamental Research, India
[3] University of Antwerp – Flanders Make, Belgium

This is an extended abstract of the work [3].

In this work, we show how to combine synthesis, model-based learning, and online sampling techniques to solve a scheduling problem featuring both hard and soft constraints. We investigate solutions to this problem both from a theoretical and from a more pragmatic point of view. On the theoretical side, we show how safety guarantees (as understood in formal verification) can be combined with guarantees offered by the probably approximately correct (PAC) learning framework [6]. On the pragmatic side, we show how safety guarantees obtained from automatic synthesis can be combined with Monte-Carlo tree search (MCTS) [5] to offer a scalable and practical solution to solve the scheduling problem at hand.

The scheduling problem that we consider is defined as follows. A task system is composed of a set of n preemptible tasks $(\tau_i)_{i \in [n]}$ partitioned into a set F of soft tasks and a set H of hard tasks. Time is assumed to be discrete and measured e.g. in CPU ticks. Each task τ_i generates an infinite number of instances $\tau_{i,j}$, called *jobs*, with $j = 1, 2, \ldots$ Jobs generated by both hard and soft tasks are equipped with deadlines, which are relative to the respective arrival times of the jobs in the system. The computation time requirements of the jobs follow a discrete probability distribution, and are unknown to the scheduler but upper bounded by their relative deadline. Jobs generated by hard tasks must complete before their respective deadlines. For jobs generated by soft tasks, deadline misses result in a penalty/cost. The tasks are assumed to be independent and generated stochastically: the occurrence of a new job of one task does not depend on the occurrences of jobs of other tasks, and both the inter-arrival and computation times of jobs are independent random variables. The scheduling problem consists in finding a *scheduler*, i.e. a function that associates, to all CPU ticks, a task that must run at that moment; in order to: (i) avoid deadline misses by hard tasks; and (ii) minimise the mean cost of deadline misses by soft tasks.

Here, we investigate learning techniques to build algorithms that can schedule safely and optimally a set of hard and soft tasks if only the deadlines and the domains of the distributions describing the tasks of the system are known a priori and not the exact distributions. This is a more realistic assumption. Our motivation was also to investigate the joint application of both synthesis techniques coming from the field of formal verification and learning techniques on an understandable yet challenging setting.

Contributions. First, we show the distributions underlying a task system with only soft tasks are *efficiently* PAC learnable: by executing the task system for a polynomial number of steps, enough samples can be collected to infer ϵ-accurate approximations of the distributions with high probability.

Then, we consider the general case of systems with both hard and soft tasks. Here, safe PAC learning is *not* always possible, and we identify two algorithmically-checkable sufficient conditions for task systems to be safely learnable. These crucially depend on the underlying MDP being a single maximal end-component, as is the case in our setting. Subsequently, we can use *robustness* results on MDPs to compute or learn near-optimal safe strategies from the learnt models.

Third, in order to evaluate the relevance of our algorithms, we present experiments of a prototype implementation. These empirically validate the efficient PAC guarantees. Unfortunately, the learnt models are often too large for the probabilistic model-checking tools. In contrast, the MCTS-based algorithm scales to larger examples: e.g. we learn safe scheduling strategies for systems with more than 10^{20} states. Our experiments also show that a strategy obtained using deep Q-learning [2, 4] by assigning high costs to missing deadlines of hard tasks does not respect safety, even if one learns for a long period of time and the deadline-miss costs of hard tasks are very high (cf. [1]).

References

1. Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe reinforcement learning via shielding. In: AAAI. pp. 2669–2678. AAAI Press (2018)
2. Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A.: Deep reinforcement learning: A brief survey. IEEE Signal Process. Mag. **34**(6), 26–38 (2017)
3. Busatto-Gaston, D., Chakraborty, D., Guha, S., Pérez, G.A., Raskin, J.: Safe learning for near-optimal scheduling. In: Abate, A., Marin, A. (eds.) Quantitative Evaluation of Systems - 18th International Conference, QEST 2021, Paris, France, August 23-27, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12846, pp. 235–254. Springer (2021). https://doi.org/10.1007/978-3-030-85172-9_13
4. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (Feb 2015)
5. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T.P., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016). https://doi.org/10.1038/nature16961
6. Valiant, L.G.: A theory of the learnable. Commun. ACM **27**(11), 1134–1142 (1984)

Complexity and Decidability Results for Verification of Deep Learning Models

Marco Sälzer and Martin Lange

School of Electr. Eng. and Computer Science, University of Kassel, Germany
https://www.uni-kassel.de/eecs/fmv

The reachability problem for neural networks. Deep learning has proved to be very successful in a broad range of applications such as image recognition or natural language processing but also safety-critical applications like autonomous driving, medical or financial applications. Such applications often come with the need for certification of safety properties. To guarantee their validity, one needs to employ methods from the area of formal verification.

Formal verification of deep neural networks (NN) is a relatively new area of research. Within this, most attention is given to efficiently solving the reachability problem NNReach for NN which is defined as follows: given some specification of valid inputs, some specification of valid outputs and a pretrained neural network N, is there a valid input $\boldsymbol{x}$ such that $N(\boldsymbol{x})$ is a valid output? Solving this problem is of high practical relevance. For example, consider a NN used in an image classification setting. A verification algorithm for NNReach could be used to guarantee that there will be no misclassification for some specified set of images.

An obvious question that arises with such a problem is that of its decidability, resp. computational complexity. Katz et al. argued[1] that NNReach is NP-complete [3]. We show that this can be strengthened as follows: NP-hardness of NNReach already holds for very simple specifications over very simple NN, for instance with just one layer or which only use a very limited set of weights and biases. These results imply that there is not much hope for classes of NN and specifications which are of practical relevance and for which NNReach can be solved efficiently.

NNReach Restricted to Simple Neural Networks. We consider classical multi-layer neural networks with ReLU activation only, as these are very common in practice. Furthermore, we consider conjunctions of linear inequalities over the input, respectively output dimensions of a NN as input and output specifications. We call a specification simple if each conjunct is of the form $x \cdot c \leq d$, where x is some variable referring to an input or output dimension and $c, d \in \mathbb{Q}$. Our first

[1] Their argument for the upper bound is flawed, though, using unbounded values as certificates. It merely shows recursive enumerability if anything. However, inclusion in NP can still be shown to hold using the settings of internal ReLU nodes as certificates [4].

major result is that NNReach restricted to NN of minimal depth and output size in combination with simple specifications remains NP-hard.

Theorem 1 ([4]). NNReach *is NP-hard for NN with output dimension one, a single hidden layer and simple specifications.*

Obviously, this result implies that any restriction to shallow NN does not lead to efficiently verifiable classes of neural networks.

Our second major result is that NNReach remains NP-hard even if we restrict it to NN that use just one negative weight/bias, one positive weight/bias and a zero weight/bias and simple specifications.

Theorem 2 ([4]). *Let $c, d \in \mathbb{Q}^{>0}$.* NNReach *restricted to NN that only use $-c, d$ and 0 as weight or bias and simple specifications is NP-hard.*

We can also show that weight or bias 0 is not necessary if we allow for arbitrary specifications. In other words, there is little chance to devise efficient solution to NNReach without imposing extreme restrictions on the use of values for weights and biases to deviate from what occurs in practice.

Similar Reachability Problems for Graph Neural Networks. Motivated by these findings, we discuss investigations of a similar problem for another deep learning model which has gained popularity recently: graph neural networks (GNN), for computing functions over graphs. There are several variations of this model, we consider a general one – the so called Message Passing Neural Networks [1]. To the best of our knowledge, there are no noteworthy results yet on the corresponding reachability problem for GNN verification so far. We start with giving a thorough definition of problems similar to NNReach for the GNN setting. We then discuss first findings for (un-)decidability results.

GNN are commonly used for two cases: node or whole-graph classification. Let G be a graph and v a node in it. A node classifier takes G and v as input and outputs a label, typically some real-valued vector, for v. A whole-graph classifier takes G as input and outputs a label for the complete graph G.

For the node classification case, we define the problem NodeReach: given some specification of valid nodes, some output specification, and a GNN N, is there a graph G with valid node v inside such that $N(G, v)$ is a valid output? For the whole-graph case, we define the problem GraphReach: given some specifications of valid graphs, some output specification of valid outputs, and a GNN N, is there a valid graph G such that $N(G)$ is a valid output?

A first, rather trivial result is that GraphReach is undecidable for specifications in first-order logic (FO). One can construct a straightforward reduction from the satisfiability problem of FO over finite graphs, known to be undecidable [5], to GraphReach. It can also be shown that NodeReach is decidable if we restrict it to graphs of a fixed degree: in this case we can reduce NodeReach to the problem of solving mixed-interger linear programs (MILP). This is even a classic NP-complete problem [2].

References

1. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. Proceedings of Machine Learning Research, vol. 70, pp. 1263–1272. PMLR (2017), http://proceedings.mlr.press/v70/gilmer17a.html
2. Karp, R.M.: Reducibility among Combinatorial Problems, pp. 85–103. Springer US, Boston, MA (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
3. Katz, G., Barrett, C.W., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient SMT solver for verifying deep neural networks. In: Majumdar, R., Kuncak, V. (eds.) Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10426, pp. 97–117. Springer (2017). https://doi.org/10.1007/978-3-319-63387-9_5
4. Sälzer, M., Lange, M.: Reachability is NP-complete even for the simplest neural networks. In: Bell, P.C., Totzke, P., Potapov, I. (eds.) Reachability Problems - 15th International Conference, RP 2021, Liverpool, UK, October 25-27, 2021, Proceedings. Lecture Notes in Computer Science, vol. 13035, pp. 149–164. Springer (2021). https://doi.org/10.1007/978-3-030-89716-1_10, https://doi.org/10.1007/978-3-030-89716-1_10
5. Trakhtenbrot, B.A.: Impossibility of an algorithm for the decision problem in finite classes. Doklady Akademii Nauk SSSR **70**, 569–572 (1950)

Binary-Search Tree Exploration in Verification of Neural Networks

Christopher Brix[1,2][0000−0002−8613−9454] and Lisa Pühl[1,3][0000−0003−4789−9642]

[1] RWTH Aachen University, 52056 Aachen, Germany
[2] brix@cs.rwth-aachen.de
[3] lisa.puehl@rwth-aachen.de

Because neural networks are commonly used in safety-critical applications, it may be necessary to prove e.g. the absence of adversarial examples. This can be done by propagating possible neuron activation values through the network. The non-linearities introduced by the activation functions can be replaced by an overapproximation. Alternatively, the problem is split into one branch per linear region of the activation function. We focus on ReLUs activations, which are commonly used and have two linear regions.

Counterexample-Guided Abstraction Refinement (CEGAR) [3] is the most commonly used technique to decide which splits to add. Here, initially all neurons are overapproximated. Should the verification provide a counterexample that turns out to be spurious, one overapproximation is replaced by a split, adding a new branch. All branches can be explored in parallel. Because each added split limits the values of neurons, computed bounds can only become tighter, and safety properties that have previously been proven do not need to be checked again.

Execution Guided Overapproximation (EGO) [1] explores the tree of possible splits in the opposite direction: First, a split is added for each neuron in the network (`++++` in Fig. 1). Should the constrained sub-problem satisfy all properties of interest, one split on one neuron is replaced by an overapproximation (`+++?`). At some point, the verification might return a spurious counterexample for the first time (`++??` succeeds, `+???` fails). Then, the subtree starting with the sibling of the previous node (`+-??`) is explored by adding all possible splits (`+-++`). Starting at the right-most leaf node, all left children of all its parents can be processed in parallel. Because overapproximations are introduced, previously computed bounds may be weakened and all properties need to be checked again.

Binary-Search Tree Exploration (BiSeTrEx) is the alternative technique proposed by us. CEGAR keeps on adding new splits until the verification succeeds for the first time, whereas EGO removes splits until it starts returning spurious counterexamples. Thus, both techniques can be described as searching for the depth in which the verification result changes. To speed up this process, we propose to add and remove multiple splits simultaneously, in a binary-search-like fashion. After evaluating the root node, for N originally overapproximated nodes, $N/2$ splits are added in one single step. If the verification fails, half of all remaining overapproximated nodes are split until the verification succeeds. If it

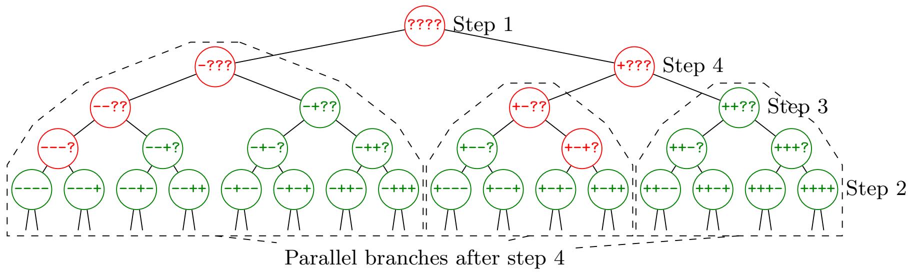


Fig. 1. Visualization of BiSeTrEx, displaying the possible splits for the first four neurons of an exemplary network with eight neurons. A `+` or `-` in the n-th position stands for a performed split of the n-th neuron, a `?` for an overapproximation. Green nodes would result in a successful verification, red nodes in a spurious counterexample. After testing the unconstrained problem with overapproximations (`????`), BiSeTrEx splits on four of the eight neurons (`++++`). In parallel, the subtree starting at `-???` is explored. As the verification succeeds, it next tries to split only two neurons (`++??`) and finally only one neuron (`+???`). This results in a spurious counterexample. Thus, the left and right children of `+???` are explored individually in parallel as well as all left children of parents of `+???`. In this instance, the right sub-tree has already been explored completely, but this may not be the case for larger examples.

succeeds, half of all previously added splits are replaced by an overapproximation. Therefore, BiSeTrEx finds the optimal number of overapproximations in $O(\log N)$ steps, whereas CEGAR and EGO require $O(N)$ iterations. Fig. 1 visualizes a verification process using BiSeTrEx and highlights possible parallelism. Previously computed results can be reused whenever new constraints are added.

[2] states that CEGAR can be sped up beyond the possibilities of EGO by adding timeouts and other heuristics that abort the analysis of nodes with many overapproximations. This is beneficial for abstract domains where performing an overapproximation is slow, such as star sets [4]. The same technique can be applied to BiSeTrEx, combining the best of both worlds for a fast verification process. A prototype implementation of our technique is under development, and we expect to present the first results by the time of the LiVe workshop.

References

1. Bak, S.: Execution-Guided Overapproximation (EGO) for Improving Scalability of Neural Network Verification. In: VNN20 (2020)
2. Bak, S.: nnenum: Verification of ReLU Neural Networks with Optimized Abstraction Refinement. In: NFM 2021. pp. 19–36. Springer (2021)
3. Clarke, E., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-Guided Abstraction Refinement. In: CAV 2000. pp. 154–169. Springer (2000)
4. Tran, H.D., Manzanas Lopez, D., Musau, P., Yang, X., Nguyen, L.V., Xiang, W., Johnson, T.T.: Star-Based Reachability Analysis of Deep Neural Networks. In: Formal Methods – The Next 30 Years. pp. 670–686. Springer (2019)

Formal Verification of Neural Networks by Learning Automata Models

Igor Khmelnitsky[a,b], Daniel Neider[c], Rajarshi Roy[c], Xuan Xie[c], Benoît Barbot[d], Benedikt Bollig[a], Alain Finkel[a,g], Serge Haddad[a,b], Martin Leucker[e], and Lina Ye[a,b,f]

[a] Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF, Gif-sur-Yvette, France
[b] Inria, France
[c] Max Planck Institute for Software Systems, Kaiserslautern, Germany
[d] Université Paris-Est Créteil, France
[e] Institute for Software Engineering and Programming Languages, Universität zu Lübeck, Germany
[f] CentraleSupélec, Université Paris-Saclay, France
[g] Institut Universitaire de France, France

– Extended Abstract –

When developing a program, one of the main fundamental problems in practice is to get it *correct*. Especially for safety-critical systems, lives may depend on the correct functioning of the software, for example, software controlling a car. But also in less safety-critical areas, the verification of systems plays an important economical role. Since its first days, it has been a challenge to get software systems correct, and a plethora of different methods have been developed ranging from debugging and testing to formal verification techniques. More generally, *formal methods* have been developed over the past decades to support the rigorous development of software systems. See [1] for an expert survey on formal methods, giving insight to the past, present, and future of formal methods.

On the other hand, artificial intelligence and as part of it machine learning is currently en vogue for developing software based systems. Especially deep learning methods turned out to be successful when developing applications for speech or image recognition, see for example [2–4], although it seems unclear, why such methods actually perform well in practice [5]. In general, deep learning methods may play one fundamental approach for synthesizing programs [6] rather than programming them manually.

As such, one may come up with the fundamental question of how to verify such networks for being sure that they adhere to what they are suppost to achieve.

Property-Directed Verification of Recurrent Neural Networks A recent approach for verifying properties of recurrent neural networks is given in [7]. It is observed that a recurrent neural network can be understood as an infinite-state machine, whenever it is used as a finite classifier. For such an infinite state system a finite-state automaton may be learned using automata learning techniques such as Angluin's L* [8] as a surrogate model approximating the system at hand. The

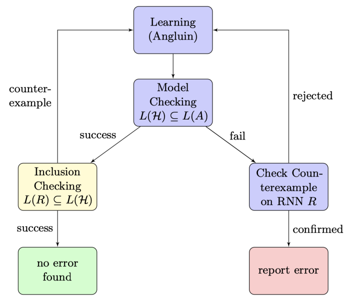

Learning (Angluin)
Model Checking $L(\mathcal{H}) \subseteq L(A)$
counter-example
rejected
success
fail
Inclusion Checking $L(R) \subseteq L(\mathcal{H})$
Check Counterexample on RNN R
success
confirmed
no error found
report error

Fig. 1. Property-directed verification of RNNs

surrogate model may then be used to check whether it meets its specification, for example using model checking techniques.

A surrogate model is a substitute model acting often in a specific role. [7] intertwines the task of verification and of learning the surrogate model in the form that a model is learned driven by the property to verify. While precise answers are obtained for checking the property on the *surrogate model* and an explicit counter example is provided if the underlying RNN does not satisfy the property at hand, a successful verification in terms of the *RNN* is only given up-to a given error probability. The latter is due to the fact that the infinite-state RNN is only statistically compared with the surrogate model. The procedure is sketched in Figure 1, where the specification to check is denoted by a language $L(A)$ of an automaton, the RNN is denoted by R and the surrogate model to be learned is denoted by $\mathcal{H}$ as it acts as hypothesis in the setting of Angluin's L*.

[7] shows that the method is indeed beneficial in identifying both violations of a specification as well as the successful verification of properties (up-to a given error probability).

References

1. Garavel, H., Beek, M.H.t., Pol, J.v.d.: The 2020 expert survey on formal methods. In ter Beek, M.H., Ničković, D., eds.: Formal Methods for Industrial Critical Systems, Cham, Springer International Publishing (2020) 3–69
2. Deng, L., Hinton, G., Kingsbury, B.: New types of deep neural network learning for speech recognition and related applications: an overview. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. (2013) 8599–8603
3. Liu, J., Pan, Y., Li, M., Chen, Z., Tang, L., Lu, C., Wang, J.: Applications of deep learning to mri images: A survey. Big Data Mining and Analytics **1**(1) (2018) 1–18
4. Mahmud, M., Kaiser, M.S., Hussain, A., Vassanelli, S.: Applications of deep learning and reinforcement learning to biological data. IEEE Transactions on Neural Networks and Learning Systems **29**(6) (2018) 2063–2079
5. Sejnowski, T.J.: The unreasonable effectiveness of deep learning in artificial intelligence. CoRR **abs/2002.04806** (2020)

6. Gulwani, S., Polozov, O., Singh, R.: Program synthesis. Foundations and Trends® in Programming Languages **4**(1-2) (2017) 1–119
7. Khmelnitsky, I., Neider, D., Roy, R., Xie, X., Barbot, B., Bollig, B., Finkel, A., Haddad, S., Leucker, M., Ye, L.: Property-directed verification and robustness certification of recurrent neural networks. In Hou, Z., Ganesh, V., eds.: Automated Technology for Verification and Analysis - 19th International Symposium, ATVA 2021, Gold Coast, QLD, Australia, October 18-22, 2021, Proceedings. Volume 12971 of Lecture Notes in Computer Science., Springer (2021) 364–380
8. Angluin, D.: Learning regular sets from queries and counterexamples. Inf. Comput. **75**(2) (1987) 87–106

Gaussian-based runtime detection of out-of-distribution inputs for neural networks

Vahid Hashemi[2], Jan Křetínský[1], Stefanie Mohr[1], and Emmanouil Seferis[1,2]

[1] Technical University of Munich, Germany
[2] AUDI AG, Ingolstadt, Germany

Abstract. Neural Networks (NN) have found their use in various applications, ranging from autonomous cars, medical imaging, to playing computer games. Especially, their use in safety-critical systems led to the question of how one can ensure their safety and reliability. This can be seen as a two-fold area: one offline and the other online. Many recent works have focused on the verification of NN, which is an offline procedure. This means that a NN is trained and verified without ever being applied in its destination (e.g. an autonomous car). This, however, relies on the belief that the developer applying the verification knows what to verify. Looking at the complex task of having a perception system for an autonomous car, it is unlikely that all possible situations are described beforehand and thus verified correctly. This is the main motivation for the Monitoring of NN, which is an online application. During runtime of the NN, e.g. when already applied in a car, the monitor will check the decisions of the NN and raise a flag if it sees a problem. This is used in particular to detect so-called out-of-distribution inputs. All inputs, and especially predicted classes, that the NN has seen before are called in-distribution since they are known. Everything else, that is new to the NN, is called out-of-distribution.
In our paper "Gaussian-based runtime detection of out-of-distribution inputs for neural networks" which was published at "Runtime Verification 21", we introduced a lightweight monitor for NN. It remembers the inner state of the NN on the training data and develops from these safety bounds. If the inner state of a new input lies outside this bound, the monitor will assume that it is an out-of-distribution sample. The inner state of the NN is the value of its neurons. Each neuron will output certain values for each sample in one output class. These values are used to fit a Gaussian distribution for each neuron and each class. Based on the Gaussian, we define intervals, by using the 95-percentile. Any new input to the neural network will then be evaluated for each neuron individually, and if its value lies without the interval, the neuron will raise a flag. If enough neurons in one layer raise their flag, the input will be considered as out-of-distribution.

Learning Linear Temporal Formulas from Specification Sketches

Simon Lutz[1] Daniel Neider[2] **Rajarshi Roy**[2]

Max Planck Institute for Software Systems, Germany
Technical University of Kaiserslautern, Germany

I will present algorithms for learning Linear Temporal Logic (LTL) formulas from *specification sketches*, which are partially specified formulas. The aim of these algorithms is to help engineers to learn suitable specifications for the formal verification of the systems they designed.

Formal verification is a time-tested method of ensuring the safe and reliable operation of systems. Success stories of formal methods include application domains such as communication systems, railway transportation, aerospace, and operating systems to name but a few.

However, virtually all verification techniques assume that the specification required for the verification or design of a system is available in a suitable format, is functionally correct, and expresses precisely the properties the engineers had in mind. These assumptions are often unrealistic in practice. Formalizing system requirements is notoriously difficult and error-prone.

To address this practical issue, we introduce a novel approach to learn formal specifications, which we refer to as *specification sketching*. Inspired by recent advances in program synthesis [3], our new paradigm allows engineers to express their high-level insights about a system in terms of a *specification sketch*, where parts that are difficult or error-prone to formalize can be left out. Moreover, the engineer is asked to provide example executions of the system that the specification should allow or forbid. Based on this additional data, a so-called *sketching algorithm* then fills in the missing low-level details to learn a complete specification.

While specification sketching is conceivable for a wide range of specification languages, in this work, we focus on Linear Temporal Logic (LTL). The rationale for this choice is twofold. First, LTL is popular in academia and widely used in industry, making it the de facto standard for expressing (temporal) properties. Second, LTL is well-understood and enjoys good algorithmic properties.

In the context of specification sketching for LTL, a sketch can leave logical operators or even entire subformulas unspecified, while examples are ultimately-periodic words (words of the form uv^ω, where u, v are finite words). For clearer understanding of the setting, consider the request-response property P expressing that every request p has to be answered eventually by a response q. This property can be formalized by the LTL formula $\varphi \coloneqq \mathbf{G}(p \rightarrow \mathbf{X}\,\mathbf{F}\,q)$, which uses the standard temporal modalities $\mathbf{F}$ ("finally"), $\mathbf{G}$ ("globally"), and $\mathbf{X}$ ("next"). However, assume, for the sake of this example, that an engineer is unsure how to formalize P. In this situation, our sketching paradigm allows the engineer to express high-level insight in the form of a sketch, say $\mathbf{G}(p \rightarrow ?)$, where the question mark indicates which part of the specification is missing. Additionally, the engineer provides two examples: (1) a positive (infinite) trajectory $(\{p\}\{q\})^\omega$, expressing that q is the response that should be used to answer a request, and (2) a negative (infinite) trajectory $\{q\}^\omega$, intended to disallow the system to send responses without requests. Our sketching algorithms then search for a substitution for the question mark such that the completed LTL formula is consistent with the examples (e.g., $\mathbf{X}\,\mathbf{F}\,q$).

It turns out that it is not always possible to find a substitution that is consistent with the given examples. However, I will demonstrate a non-deterministic algorithm that shows that the decision problem of checking whether such a substitution exists is in the complexity class NP. I will further present an effective decision procedure that reduces the decision problem to a satisfiability problem in propositional logic, thus allowing the use of optimized SAT solvers.

I will then present two sketching algorithms for LTL. The first algorithm uses the presented decision procedure of as a sub-routine and transforms the sketching problem into a series of LTL learning problems (i.e., in problems of learning an LTL formula—without syntactic constraints—from positive and negative examples). This transformation allows us to apply a diverse array of learning algorithms for LTL, which have been proposed during the last five years [1, 2]. The second sketching algorithm, on the other hand, extends the LTL learning algorithm by Neider and Gavran [1] and uses a SAT-based technique as an effective means to search for solutions of increasing size.

Finally, I will demonstrate some preliminary experimental results using our prototype implementation—`LTL-sketcher`. In particular, I will emphasize that our algorithms are effective in learning formulas by completing a variety of sketches with different missing information.

The presentation will be based on a work under submission with Simon Lutz and Daniel Neider.

References

[1] D. Neider and I. Gavran. Learning linear temporal properties. In *FMCAD*, pages 1–10. IEEE, 2018.

[2] H. Riener. Exact synthesis of LTL properties from traces. In *FDL*, pages 1–6. IEEE, 2019.

[3] A. Solar-Lezama. Program sketching. *Int. J. Softw. Tools Technol. Transf.*, 15(5-6):475–495, 2013.