

# Integrating the Power of Machine Learning and Model Checking in Safety-Critical Systems

Rong Gu<sup>[0000–0003–0570–6005]</sup>

Mälardalen University, Västerås, Sweden  
rong.gu@mdu.se

## 1 Motivation of Integration

Safety-critical systems, e.g., aircraft and cars, may cause casualties and heavy financial loss when errors occur. When such systems are real-time, safety depends not only on functional correctness but also on timing. Model checking (MC), as a classic technique of formal methods, is well-known for its capability of rigorous analysis of complex applications, such as real-time and concurrent systems. Despite the wide usage of MC, it has some usability barriers for the industry, such as scalability and difficulties in mastering various formalisms for different verification tools. Machine learning (ML) has been proven to be useful and even better than human intelligence in solving complex and single-domain problems, such as AlphaGo beating the best human players in Go, and thus would offer great help in overcoming the challenges of adopting MC in safety-critical systems.

## 2 Our Experience and Ongoing Work

**Controller synthesis for autonomous systems.** A classic problem of autonomous systems is the construction of controllers that satisfy the so-called *reach-avoid* property. Intuitively, the reach-avoid property means the system must reach the destination within a time frame and avoid obstacles on the way. We designed a tool called *TAMAA* (UPPAAL-based Mission Planning for Autonomous Agents) [1] that uses MC to generate such controllers within a confined environment that only has static obstacles. The experimental result shows that when the agent number is greater than five, UPPAAL exhausts the memory due to the large state space of the model. To overcome this issue, we proposed *MCRL* (Model Checking + Reinforcement Learning) [2], which uses reinforcement learning to generate controllers without a correctness guarantee. Next, we check the learning result, i.e., a controller that restricts the state space, and iterate learning and checking until we obtain a result that satisfies the reach-avoid property. Further, we developed *MoCReL* (Model-checked Compressed Reinforcement Learning) [3] that can not only check the learning result but also compress it without losing the correctness guarantee, that is, the generated controller takes much less memory space but still satisfies the reach-avoid property.

Along this line of research, we proposed a TRebeca-based (Timed Reactive Objects Language) platform for controller synthesis that guarantees safety and

security together [4]. We model the system as a Markov decision process (MDP) by using TRebeca and explore the state space by using our *Guess-and-then-Check* algorithm for controller generation. Briefly, the algorithm separates the synthesis process into three phases. In phase I (i.e., guessing), we exhaustively explore the transitions belonging to the system such that we do not miss valid system actions that might be correct. However, we accumulate rewards for the transitions belonging to the environment and selectively explore them based on the rewards. Essentially, the higher the reward the faster the environment can guide phase I to a state that terminates the state-space exploration, e.g., states violating the safety property. In phase II, we check the guessed controller against a safety property by exhaustively exploring the environment’s transitions. Since the guessed controller greatly restricts the state space already, phase II does not necessarily have the state-space-explosion problem even when the environment has a great number of transitions. After phase II, we obtain a controller that is guaranteed to be safe. However, it may not be secure in the sense that it may reveal information to unauthorized intruders. Hence, phase III checks the controller against a security property and prunes the transitions that are not secure. Our algorithm splits the exhaustive state-space exploration into different phases, so the state-space-explosion problem is alleviated. Additionally, we adopt learning in the state-space exploration, which helps us to terminate the exploration on the branches that are *doomed* to fail as soon as possible.

**Timing analysis for real-time systems.** Ambiguities and inconsistencies in requirements are two kinds of errors that would have been avoided given the right analytical method. Ambiguity refers to multiple interpretations of the same piece of requirement. Consistency refers to the existence of models that satisfy all the timing requirements. We proposed a conceptual framework that employs LLM for detecting ambiguities in textual requirements and MC for finding counterexamples that cause inconsistencies in timing requirements. As our textual requirements are based on templates, given a proper amount of prompting, LLM would be able to generate formal models from texts directly. Additionally, LLM can also help interpret the counterexamples and generate improved texts for requirements. As the framework is under development in collaboration with our industrial partners, we have confidence that the framework would be state-of-the-art and practically useful in real-life applications.

## References

1. Gu, R., Enoiu, E., Seceleanu, C.: Tamaa: Uppaal-based mission planning for autonomous agents. In: ACM Symposium on Applied Computing (2020)
2. Gu, R., Enoiu, E., Seceleanu, C., Lundqvist, K.: Verifiable and scalable mission-plan synthesis for autonomous agents. In: FMICS. Springer (2020)
3. Gu, R., Jensen, P.G., Seceleanu, C., Enoiu, E., Lundqvist, K.: Correctness-guaranteed strategy synthesis and compression for multi-agent autonomous systems. *Science of Computer Programming* (2022)
4. Gu, R., Moezkarimi, Z., Sirjani, M.: Guess and then check: Safety and security guaranteed construction of cyber-physical systems. Tech. rep. (January 2024)

# Towards Adaptive Reactive Synthesis Modulo Theories (Extended Abstract)

Andoni Rodríguez-Barcenilla and César Sánchez

IMDEA Software Institute, Madrid. Spain

**Abstract.** Reactive synthesis is the process of using temporal logic specifications in *LTL* to generate correct controllers, but its use has been restricted to Boolean specifications. Recently, a Boolean abstraction technique allows to translate  $LTL_{\mathcal{T}}$  specifications that contain literals in theories into equi-realizable *LTL* specifications. However, no synthesis procedure exists yet. In synthesis modulo theories, the system to synthesize receives valuations of environment variables in a first-order theory  $\mathcal{T}$  and outputs valuations of system variables from  $\mathcal{T}$ . In this extended abstract of [5], we address how to synthesize a full controller using a combination of the static Boolean controller obtained from the *Booleanized LTL* specification together with on-the-fly queries to a solver that produces models of a satisfiable existential formulae from  $\mathcal{T}$ . This is the first method that realizes reactive synthesis modulo theories. Additionally, it allows to produce adaptive responses which increases explainability and can improve runtime properties like performance. Our approach is applicable to both *LTL* modulo theories and  $LTL_f$  modulo theories.

## 1 Problem Statement

Reactive synthesis is the problem of automatically producing a system that models a given temporal specification, where the Boolean variables (i.e., atomic propositions) are split into variables controlled by the environment and variables controlled by the system. Realizability is the related decision problem of deciding whether such a system exists. These problems have been widely studied [3], specially in the domain of Linear Temporal Logic (*LTL*) [2]. Realizability corresponds to an infinite game where players alternatively choose the valuations of the Boolean variables they control. A specification is realizable if and only if the system has a strategy such that the specification is satisfied in all plays played according to the strategy. The synthesis process is produced from a winning system strategy. Both reactive synthesis and realizability are decidable for *LTL* [3]. *LTL* modulo theories ( $LTL_{\mathcal{T}}$ ) is the extension of *LTL* where Boolean atomic propositions can be literals from a (multi-sorted) first-order theory  $\mathcal{T}$ . Realizability of  $LTL_{\mathcal{T}}$  specifications is decidable under certain conditions over  $\mathcal{T}$ , shown in [4] using a Boolean abstraction or *Booleanization* method that translates specifications in  $LTL_{\mathcal{T}}$  into equi-realizable *LTL* formulae, which means that the original

specification in  $LTL_{\mathcal{T}}$  is realizable if and only if the produced Boolean LTL specification is realizable, and vice versa. Note that an  $LTL_{\mathcal{T}}$  reactive specification splits the theory variables into environment controlled and system controlled variables that can appear in a single literal, while  $LTL$  Boolean atoms belong fully to either player.

However, to perform  $LTL_{\mathcal{T}}$  reactive synthesis it is not enough to synthesize a controller for the Booleanized specifications because the system will receive and will have to produce values from theory variables. Some previous synthesis methods try to create statically a controller that always produces the same outputs for the same inputs (e.g. [1]) but are incomplete for many  $\mathcal{T}$ , since they need to compute Skolem functions. Thus, we propose a general method that uses *on-the-fly* procedures to dynamically produce outputs as the results of computing models of existential  $\mathcal{T}$  formulae. Concretely, the method we propose statically receives an  $LTL_{\mathcal{T}}$  specification  $\varphi$ , Booleanizes  $\varphi$  using [4] and synthesizes a controller  $S$  using standard methods. Then, dynamically  $S$  is combined with a tool that can produce models of satisfiable  $\mathcal{T}$  formulae (e.g., an SMT solver) which collaborate in tandem at each step of the execution. To guarantee that the reaction is produced at every step, we require that  $\mathcal{T}$  has an efficient procedure to provide models of existential fragments of  $\mathcal{T}$ . Our approach does not guarantee termination using semi-decidable  $\mathcal{T}$ , but still yields a partial solution for such cases. We also use an additional component, called *partitioner*, which discretizes the environment  $\mathcal{T}$ -input providing a suitable input for the Boolean controller (but this is an easy by-product of the Booleanization procedure and can be computed statically). As far as we know, this is the first successful reactive synthesis procedure for  $LTL_{\mathcal{T}}$  specifications. We refer the reader to [5] for full details.

## References

- [1] Andreas Katis, Grigory Fedyukovich, Andrew Gacek, John D. Backes, Arie Gurfinkel, and Michael W. Whalen. Synthesis from assume-guarantee contracts using skolemized proofs of realizability. *CoRR*, abs/1610.05867, 2016.
- [2] Amir Pnueli. The temporal logic of programs. In *Proc. of the 18th IEEE Symp. on Foundations of Computer Science (FOCS'77)*, pages 46–67. IEEE CS Press, 1977.
- [3] Amir Pnueli and Roni Rosner. On the synthesis of an asynchronous reactive module. In *Proc. of the 16th Int'l Colloquium on Automata, Languages and Programming (ICALP'89)*, volume 372 of *LNCS*, pages 652–671. Springer, 1989.
- [4] Andoni Rodriguez and César Sánchez. Boolean abstractions for realizability modulo theories. In *Proc. of the 35th International Conference on Computer Aided Verification (CAV'23)*, volume 13966 of *LNCS*. Springer, Cham, 2023.
- [5] Andoni Rodriguez and César Sánchez. Adaptive Reactive Synthesis for LTL and LTLf Modulo Theories. In *Proc. of the 38th AAAI Conf. on Artificial Intelligence (AAAI 2024)*, 2024.



# Distillation of RL Policies through Bisimilar Latent Models with Formal Guarantees

Florent Delgrange<sup>1,2</sup>, Ann Nowé<sup>1</sup>, and Guillermo A. Pérez<sup>2</sup>

<sup>1</sup> AI Lab, Vrije Universiteit Brussel

<sup>2</sup> University of Antwerp – Flanders Make

While *reinforcement learning* (RL) has been applied to a wide range of challenging domains, from game playing [19] to real-world applications [18, 5, 17, 7, 21, 8, 24], more widespread deployment in the real world is hampered by the lack of guarantees provided with the learned policies. Although there are RL algorithms which have limit-convergence guarantees in the discrete setting [22] (and even in some continuous settings with function approximation, e.g., [20]), these are lost when applying more advanced techniques which make use of general nonlinear function approximators [23] to deal with *Markov decision processes* (MDPs) with general state and action spaces. Those methods are grouped together under the term *deep RL* [19]. In this work, we apply deep RL to *unknown* MDPs with logical specification or discounted-reward objectives, and we consider the challenge of simplifying and verifying RL policies. Our goal is to *enable model checking* [4] by learning an accurate, tractable model of the environment.

**Bisimulation Guarantees.** To recover the formal guarantees, we thus seek a verifiable *discrete latent model* that approximates the unknown environment.

Given the original (continuous, possibly unknown) environment model  $\mathcal{M}$ , a *latent space model* is another (smaller, explicit) MDP  $\bar{\mathcal{M}}$  with state-action space linked to the original one via a *state embedding* and an *action lifting* function, respectively denoted as  $\phi$  and  $\psi$ . Intuitively, an agent may execute a latent policy  $\bar{\pi}$  (defined over the latent spaces) in  $\mathcal{M}$  as follows: at each step of the interaction, the current state  $s$  of  $\mathcal{M}$  is embedded to the latent space via  $\phi(s) = \bar{s}$ , then the agent executes the latent action  $\bar{a}$  prescribed by the policy  $\bar{\pi}$  by lifting it back to the original model via  $\psi$ . Then,  $\mathcal{M}$  transitions to the next state  $s'$  according to its transition function  $\mathbf{P}$ , the original state

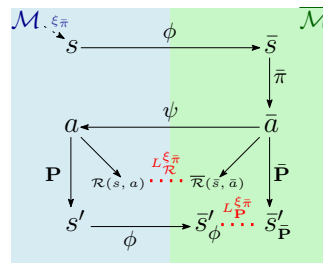


Fig. 1: Execution of  $\bar{\pi}$ .

and this resulting action. The guarantees rely on (i) the *bisimulation pseudo-metric*  $\tilde{d}_{\bar{\pi}}$  [2, 12–14], and (ii) two *local losses*  $L_{\mathbf{P}}$  and  $L_{\mathcal{R}}$  [15]. The former can be interpreted as the *largest behavioral difference* between  $\mathcal{M}$  and  $\bar{\mathcal{M}}$  when  $\bar{\pi}$  is executed. In particular, a zero distance means that the agent behaves the same way in both models. The latter intuitively quantify respectively the expected distance between the original and latent reward functions,  $\mathcal{R}$  and  $\bar{\mathcal{R}}$ , as well as

their transition functions,  $\mathbf{P}$  and  $\bar{\mathbf{P}}$ . We show that these two losses bound  $\tilde{d}_{\bar{\pi}}$ :

$$\mathbb{E}_{s \sim \xi_{\bar{\pi}}} \tilde{d}_{\bar{\pi}}(s, \phi(s)) \leq \frac{L_{\mathcal{R}} + KL_{\mathbf{P}}}{1 - \gamma}; \quad \tilde{d}_{\bar{\pi}}(s_1, s_2) \leq \frac{L_{\mathcal{R}} + KL_{\mathbf{P}}}{1 - \gamma} \left( \frac{1}{\xi_{\bar{\pi}}(s_1)} + \frac{1}{\xi_{\bar{\pi}}(s_2)} \right)$$

where  $\xi_{\bar{\pi}}$  is a *stationary measure* of  $\mathcal{M}$  under  $\bar{\pi}$ ,  $\gamma$  is a discount,  $K$  is a constant, and  $s_1, s_2$  share the same embedding  $\phi(s_1) = \phi(s_2)$ . These inequalities guarantee the *quality of the abstraction* and *representation*: when local losses are small, (i) in average, states and their embedding, and (ii) all states sharing the same discrete representation, are bisimilarly close. We give PAC estimation schemes to compute both the losses and said bounds [11]. Next, we learn a distillation  $\bar{\pi}$  of the RL policy along with  $\bar{\mathcal{M}}$ , where the behaviors of the agent can be formally verified. The bounds offer a confidence metric allowing to lift the guarantees obtained this way back to  $\mathcal{M}$ , when it operates under  $\bar{\pi}$ .

**Generative models.** We learn the latent space either by (i) maximizing a variational lower bound on the log-likelihood of the trajectories generated by the deep RL policy in the original environment [11], or (ii) minimizing the *Wasserstein distance* between those trajectories and those reconstructed in the latent model [10]. The resulting objective functions incorporates respectively (i) variational proxies to, or (ii) directly the local losses. This enables learning a discrete latent model, state embedding and action lifting functions, as well as a distillation  $\bar{\pi}$  of the RL policy. Our algorithm enables efficient learning of the representation, avoiding the *posterior collapse* problem that occurs in variational models [1].

**Experiments.** We trained deep RL policies [19, 16] for various benchmarks [6], which we then distill via our approach. The results reveal that optimizing our latent models allows minimizing the local losses (Fig 2a). Furthermore, this enables the distillation of RL policies into  $\bar{\pi}$ , for which the formal guarantees apply: the original performance is eventually recovered (Fig 2b).

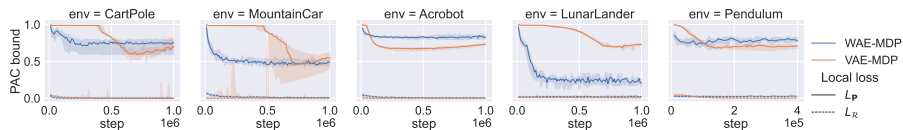


Fig. 2a. PAC bounds: Wasserstein [10] vs. Variational [11] auto-encoded MDP

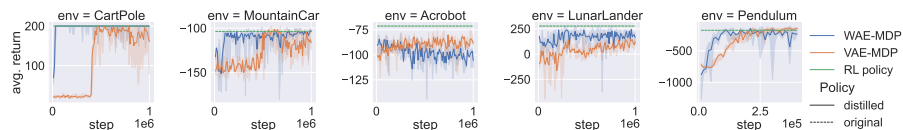


Fig. 2b. Evaluation of the distilled policies

Beyond distillation, our abstraction and representation guarantees have also proved beneficial for policy learning. Notably, we extended our guarantees in the context of learning beliefs in POMDPs, removing the need of introducing recurrent neural networks for deep RL policies [3], and synthesizing controllers in hierarchical scenarios based on deep RL policies [9].

## Acknowledgements

This research received funding from the Flemish Government (AI Research Program) and was supported by the DESCARTES iBOF project. G.A. Perez is also supported by the Belgian FWO “SAILor” project (G030020N).

## References

1. Alemi, A.A., Poole, B., Fischer, I., Dillon, J.V., Sauros, R.A., Murphy, K.: Fixing a broken ELBO. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, July 10-15, 2018. Proceedings of Machine Learning Research, vol. 80, pp. 159–168. PMLR (2018)
2. de Alfaro, L., Henzinger, T.A., Majumdar, R.: Discounting the future in systems theory. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003. Proceedings. Lecture Notes in Computer Science, vol. 2719, pp. 1022–1037. Springer (2003). [https://doi.org/10.1007/3-540-45061-0\\_79](https://doi.org/10.1007/3-540-45061-0_79), [https://doi.org/10.1007/3-540-45061-0\\_79](https://doi.org/10.1007/3-540-45061-0_79)
3. Avalos, R., Delgrange, F., Nowe, A., Perez, G., Roijers, D.M.: The wasserstein believer: Learning belief updates for partially observable environments through reliable latent space models. In: The Twelfth International Conference on Learning Representations (2024), <https://openreview.net/forum?id=KrtGfTGaGe>
4. Baier, C., Katoen, J.: Principles of model checking. MIT Press (2008)
5. Bellemare, M.G., Candido, S., Castro, P.S., Gong, J., Machado, M.C., Moitra, S., Ponda, S.S., Wang, Z.: Autonomous navigation of stratospheric balloons using reinforcement learning. *Nat.* **588**(7836), 77–82 (2020). <https://doi.org/10.1038/S41586-020-2939-8>, <https://doi.org/10.1038/s41586-020-2939-8>
6. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym. *CoRR* **abs/1606.01540** (2016), <http://arxiv.org/abs/1606.01540>
7. Ceusters, G., Rodríguez, R.C., García, A.B., Franke, R., Deconinck, G., Helsen, L., Nowé, A., Messagie, M., Camargo, L.R.: Model-predictive control and reinforcement learning in multi-energy system case studies. *Applied Energy* **303**, 117634 (2021). <https://doi.org/https://doi.org/10.1016/j.apenergy.2021.117634>, <https://www.sciencedirect.com/science/article/pii/S0306261921010011>
8. Degraeve, J., Felici, F., Buchli, J., Neunert, M., Tracey, B.D., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de Las Casas, D., Donner, C., Fritz, L., Galperti, C., Huber, A., Keeling, J., Tsimpoukelli, M., Kay, J., Merle, A., Moret, J., Noury, S., Pesamosca, F., Pfau, D., Sauter, O., Sommariva, C., Coda, S., Duval, B., Fasoli, A., Kohli, P., Kavukcuoglu, K., Hassabis, D., Riedmiller, M.A.: Magnetic control of tokamak plasmas through deep reinforcement learning. *Nat.* **602**(7897), 414–419 (2022). <https://doi.org/10.1038/S41586-021-04301-9>, <https://doi.org/10.1038/s41586-021-04301-9>
9. Delgrange, F., Avni, G., Lukina, A., Schilling, C., Nowé, A., Pérez, G.A.: Synthesis of hierarchical controllers based on deep reinforcement learning policies (2024)

10. Delgrange, F., Nowé, A., Pérez, G.A.: Wasserstein auto-encoded mdps: Formal verification of efficiently distilled RL policies with many-sided guarantees. In: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net (2023), <https://openreview.net/pdf?id=JLLTtEdh1ZY>
11. Delgrange, F., Nowé, A., Pérez, G.A.: Distillation of rl policies with formal guarantees via variational abstraction of markov decision processes. *Proceedings of the AAAI Conference on Artificial Intelligence* **36**(6), 6497–6505 (Jun 2022). <https://doi.org/10.1609/aaai.v36i6.20602>, <https://ojs.aaai.org/index.php/AAAI/article/view/20602>
12. Desharnais, J., Gupta, V., Jagadeesan, R., Panangaden, P.: Metrics for labelled markov processes. *Theor. Comput. Sci.* **318**(3), 323–354 (2004). <https://doi.org/10.1016/J.TCS.2003.09.013>, <https://doi.org/10.1016/j.tcs.2003.09.013>
13. Ferns, N., Panangaden, P., Precup, D.: Metrics for markov decision processes with infinite state spaces. In: UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005. pp. 201–208. AUAI Press (2005), [https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article\\_id=1175&proceeding\\_id=21](https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=1175&proceeding_id=21)
14. Ferns, N., Precup, D., Knight, S.: Bisimulation for markov decision processes through families of functional expressions. In: van Breugel, F., Kashefi, E., Palamidessi, C., Rutten, J. (eds.) *Horizons of the Mind. A Tribute to Prakash Panangaden - Essays Dedicated to Prakash Panangaden on the Occasion of His 60th Birthday*. LNCS, vol. 8464, pp. 319–342. Springer (2014). [https://doi.org/10.1007/978-3-319-06880-0\\_17](https://doi.org/10.1007/978-3-319-06880-0_17), [https://doi.org/10.1007/978-3-319-06880-0\\_17](https://doi.org/10.1007/978-3-319-06880-0_17)
15. Gelada, C., Kumar, S., Buckman, J., Nachum, O., Bellemare, M.G.: Deepmdp: Learning continuous latent space models for representation learning. In: Chaudhuri, K., Salakhutdinov, R. (eds.) *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. *Proceedings of Machine Learning Research*, vol. 97, pp. 2170–2179. PMLR (2019), <http://proceedings.mlr.press/v97/gelada19a.html>
16. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Dy, J.G., Krause, A. (eds.) *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. *Proceedings of Machine Learning Research*, vol. 80, pp. 1856–1865. PMLR (2018), <http://proceedings.mlr.press/v80/haarnoja18b.html>
17. Libin, P.J.K., Moonens, A., Verstraeten, T., Perez-Sanjines, F., Hens, N., Lemey, P., Nowé, A.: Deep reinforcement learning for large-scale epidemic control. In: Dong, Y., Ifrim, G., Mladenic, D., Saunders, C., Hoecke, S.V. (eds.) *Machine Learning and Knowledge Discovery in Databases. Applied Data Science and Demo Track - European Conference, ECML PKDD 2020, Ghent, Belgium, September 14-18, 2020, Proceedings, Part V*. *Lecture Notes in Computer Science*, vol. 12461, pp. 155–170. Springer (2020). [https://doi.org/10.1007/978-3-030-67670-4\\_10](https://doi.org/10.1007/978-3-030-67670-4_10), [https://doi.org/10.1007/978-3-030-67670-4\\_10](https://doi.org/10.1007/978-3-030-67670-4_10)
18. Mason, K., Grijalva, S.: A review of reinforcement learning for autonomous building energy management. *Comput. Electr. Eng.* **78**, 300–312 (2019). <https://doi.org/10.1016/J.COMPELECENG.2019.07.019>, <https://doi.org/10.1016/j.compeleceng.2019.07.019>

19. 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). <https://doi.org/10.1038/nature14236>, <https://doi.org/10.1038/nature14236>
20. Nowe, A.: Synthesis of “safe” fuzzy controllers based on reinforcement learning. Ph.D. thesis, Vrije Universiteit Brussel (1994)
21. Reymond, M., Hayes, C.F., Willem, L., Radulescu, R., Abrams, S., Roijers, D.M., Howley, E., Mannion, P., Hens, N., Nowé, A., Libin, P.: Exploring the pareto front of multi-objective COVID-19 mitigation policies using reinforcement learning. *CoRR* **abs/2204.05027** (2022). <https://doi.org/10.48550/ARXIV.2204.05027>, <https://doi.org/10.48550/arXiv.2204.05027>
22. Tsitsiklis, J.N.: Asynchronous stochastic approximation and q-learning. *Mach. Learn.* **16**(3), 185–202 (1994). <https://doi.org/10.1007/BF00993306>, <https://doi.org/10.1007/BF00993306>
23. Tsitsiklis, J.N., Roy, B.V.: An analysis of temporal-difference learning with function approximation. *IEEE Trans. Autom. Control.* **42**(5), 674–690 (1997). <https://doi.org/10.1109/9.580874>, <https://doi.org/10.1109/9.580874>
24. Wurman, P.R., Barrett, S., Kawamoto, K., MacGlashan, J., Subramanian, K., Walsh, T.J., Capobianco, R., Devlic, A., Eckert, F., Fuchs, F., Gilpin, L., Khandelwal, P., Kompella, V., Lin, H., MacAlpine, P., Oller, D., Seno, T., Sherstan, C., Thomure, M.D., Aghabozorgi, H., Barrett, L., Douglas, R., Whitehead, D., Dürr, P., Stone, P., Spranger, M., Kitano, H.: Out-racing champion gran turismo drivers with deep reinforcement learning. *Nat.* **602**(7896), 223–228 (2022). <https://doi.org/10.1038/S41586-021-04357-7>, <https://doi.org/10.1038/s41586-021-04357-7>

# Validating Streaming JSON Documents with Learned VPAs (Long Abstract)\*

Gaëtan Staquet<sup>1,2</sup> 

<sup>1</sup> University of Mons (UMONS), Mons, Belgium

`gaetan.staquet@umons.ac.be`

<sup>2</sup> University of Antwerp (UAntwerp) – Flanders Make, Antwerp, Belgium

*JavaScript Object Notation* (JSON) has overtaken XML as the de facto standard data-exchange format, in particular for web applications. JSON documents are easier to read for programmers and end users since they only have arrays and objects as structured types. Moreover, in contrast to XML, they do not include named open and end tags for all values, but open and end tags (braces actually) for arrays (ordered collections of values) and objects (unordered collections of key-value pairs) only.

In order to be correctly processed by web services, for instance, a JSON document has to satisfy some structural constraints. These constraints can be encoded in a *JSON schema* [3]. Recently, a standard to formalize JSON schemas has been proposed and (hand-coded) validation tools for such schemas can be found online [3]. Pezoa et al, in [4], observe that the standard of JSON documents is still evolving and that the formal semantics of JSON schemas is also still changing. Furthermore, validation tools seem to make different assumptions about both documents and schemas. The authors of [4] carry out an initial formalization of JSON schemas into formal grammars from which they are able to construct a *batch* validation tool from a given JSON schema.

In this joint work with Véronique Bruyère and Guillermo A. Pérez [1], we present a *streaming* algorithm to *validate* JSON documents against a set of constraints given as a JSON schema. The idea is to represent the set of valid JSON documents via an automaton. As we have to take into account the nesting of objects and arrays, a classical (deterministic) finite automaton is not sufficient. Instead, we require a *stack* to remember that nesting. We show that there always exists a *visibly pushdown automaton* (VPA) that accepts the same set of JSON documents as a JSON schema.

We present a VPA active learning framework to automatically construct a VPA from a JSON schema, that extends TTT [2] by leveraging side information about the language of all JSON documents. In order to be able to learn this VPA in a reasonable time, we fix an order on the keys appearing in objects. That is, in the VPA, objects are now *ordered* collections of key-value pairs. As, in general, a JSON document may not follow this order, we develop the concept of *key graph*, which allows us to efficiently realize the validation no matter the key-value order in the document. The validation algorithm we propose accepts other permutations of keys, via the key graph.

---

\* Gaëtan Staquet is a Research Fellow of the Fonds de la Recherche Scientifique – FNRS.

We also present experimental results of our learning and validation algorithms from JSON schemas used in industrial cases. Finally, we compare the efficiency of validation against the classical validation algorithm, on a number of randomly generated JSON documents.

## References

1. Bruyère, V., Pérez, G.A., Staquet, G.: Validating streaming json documents with learned vpas. In: Sankaranarayanan, S., Sharygina, N. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 271–289. Springer Nature Switzerland, Cham (2023)
2. Malte Isberner: Foundations of active automata learning: an algorithmic perspective. Ph.D. thesis, Technical University Dortmund, Germany (2015), <http://hdl.handle.net/2003/34282>
3. JSON Schema: JSON Schema website. Online (2015), <https://json-schema.org>
4. Felipe Pezoa, Juan L. Reutter, Fernando Suárez, Martín Ugarte, Domagoj Vrgoc: Foundations of JSON Schema. In: Jacqueline Bourdeau, Jim Hendler, Roger Nkambou, Ian Horrocks, Ben Y. Zhao (eds.) Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016. pp. 263–273. ACM (2016). <https://doi.org/10.1145/2872427.2883029>

# Enabling Verification of Deep Neural Networks in Perception Tasks Using Fuzzy Logic and Concept Embeddings

Gesina Schwalbe<sup>[0000-0003-2690-2478]</sup><sup>1</sup>, Christian Wirth<sup>2</sup>, and Ute Schmid<sup>3</sup>

<sup>1</sup> University of Lübeck, Germany [gesina.schwalbe@uni-luebeck.de](mailto:gesina.schwalbe@uni-luebeck.de)

<sup>2</sup> Continental AG, Frankfurt, Germany

<sup>3</sup> Cognitive Systems, University of Bamberg, Bamberg, Germany

**Abstract.** This abstract summarizes our work towards logical consistency checks of convolutional neural networks (CNNs) for computer vision against symbolic prior domain knowledge [14]. The two simple ideas are (i) to treat the object detections as (fuzzy) logical predicates, e.g.,  $\text{Is}_{\text{person}}$ , and (ii) to model additional predicates via concept extraction, e.g., for  $\text{Is}_{\text{bodypart}}$ . The approach poses an interesting direction towards formal verification of CNNs against symbolic constraints.

**Keywords:** Concept Activation Vectors, Fuzzy Logic, Calibration, Verification Testing

## 1 Motivation

In safety critical tasks like perception for autonomous driving, it is necessary to verify that a model complies with given prior domain knowledge [13]. Examples in object detection are (fuzzy) logical relations between symbolic concepts, e.g., anatomical structures like *body part regions are mostly part of person regions*. While DNNs are known to extract a rich set of symbolic features—even if operating on non-symbolic inputs like images [3, 11, 16]—black-box CNNs only disclose information on symbolic concepts in their final outputs (e.g., *where is a person*). These are typically insufficient for verifying complex symbolic relations; and respecting rich semantics during design [5, 8] proves impossible if new constraints may frequently emerge during operation.

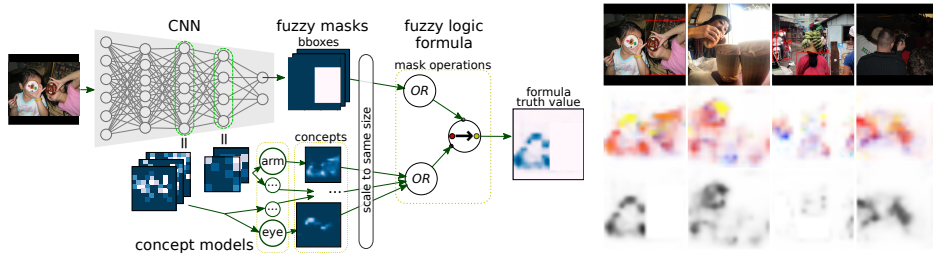
## 2 Idea

The goal is to allow verification testing (and potentially subsequently formal verification) of CNNs against symbolic constraints with symbols not known during training. The three ingredients we use to solve this are:

- (1) If interpreting visual symbols (here called visual semantic concepts) as logical 1-ary predicates operating on image regions like  $\text{Is}_{\text{person}}$ ,  $\text{Is}_{\text{bodypart}}$ , one can formulate the symbolic prior knowledge as logical relations, e.g.,

$$\forall x: \text{Is}_{\text{bodypart}}(x) \rightarrow \text{Is}_{\text{person}}(x).$$





(a) Overall approach: (1) Enrich CNN outputs using concept models (CMs). For inference/grounding: (2) Interpret CNN outputs as same-sized fuzzy truth masks; (3) evaluate fuzzy logic formula (here:  $F(p) = (I_{\text{eye}}(p) \vee \dots \vee I_{\text{arm}}(p)) \rightarrow I_{\text{Person}}(p)$  for pixels  $p$ ).

(b) Example true alarms of a monitor for formula Eq. 1: *top*: EffDet predicted boxes; *mid*: CM outputs (**leg**, **ankle**, **arm**, **wrist**, **eye**); *bottom*: pixel-wise truth values

Depicted photos (Fig. 1b left to right) courtesy [David Quitoriano](#), [j bizzie](#), [CCFoodTravel.com](#), [k.steudel](#) © CC BY 2.0

Fig. 1: Our approach to score per-pixel consistency with symbolic rules.

- (2) Extending a suggestion by Diligenti et al. [2] to object detection, CNNs can be interpreted as a family of (fuzzy) logical 1-ary predicates operating on image regions (here: pixels), such as  $I_{\text{person}}$ . Thus, if all symbols of interest can be allocated to outputs of the CNN, we can verify in how far the formulas evaluate to true when using the CNN predicates—how *logically consistent* the CNN behaves with respect to the rule.
- (3) To add missing symbols to the output of a trained CNN, we utilize concept embedding analysis [12, 11], a method from the field of explainable AI; and to ensure that these additional outputs can properly be interpreted as fuzzy truth values, we adopt techniques from uncertainty calibration for DNNs [6].

Altogether, we provide a simple and scalable framework to compute a *logical consistency score* of a CNN with regards to a given logical rule and input, see Fig. 1. This is shown to uncover a substantial proportion of detection errors for two state-of-the-art object detectors, and benefits from the additional uncertainty calibration and the fuzziness. This could be used, e.g., as self-supervised error indicator at operation-time, for identification of corner cases at test-time, or even for automated corrections of predictions.

*Comparison to Related Work* In contrast to related work by Diligenti et al. [2], our approach is applied to *object detection* instead of mere classification, and works in a *post-hoc* manner, i.e., on *any* CNN-based object detector without changes to training or architecture, other than, e.g., [5, 2].

### 3 Formalization: Fuzzy Logic Rules for Object Detection

Consider an image  $x$ , the domain of pixel positions  $P$ , and an instance or semantic segmentation mask, e.g., for an object class  $\text{cls}$ . Our modeling approach is to interpret such a mask as *per-pixel* truth values  $(I_{\text{cls}}(p, x))_{p \in P}$  that are obtained from grounding the predicate  $I_{\text{cls}}$  to  $x$  and pixels  $p \in P$ , as shown in Fig. 1.

**1-ary logical predicates (concepts):** Fuzzy segmentation masks can originate from the CNN object or concept presence prediction by a post-hoc attached concept model [11]. Note that we here use the fact that CNN outputs are inherently fuzzy, i.e., are not binary truth values but confidences in  $[0, 1]$ .

*Object predictions:* A CNN bounding box prediction  $i$  of class  $cls$  is turned into an instance segmentation mask  $(Is_{cls,i}(p, x))_p$  by setting all pixels inside the detection box to the objectness score. Therefore, we can define per-pixel predicates for object and other concept classes, given a *fixed* DNN.

*Additional concept presence predictions:* This is obtained using concept embedding analysis [3, 12], and allows to classify pixels by concepts. This is done by post-hoc attaching per concept a small linear *concept model* (CM) to the intermediate output of a CNN layer. The CM is trained to accurately predict presence of the concept at each activation map pixel (=values of neurons associated with a common pixel position).

**Other logical predicates:** We also define a predicate `closeBy` for fuzzy proximity between pixels in terms of a squared exponential distance function. For smoothing predicate outputs, we apply a neighborhood condition predicate defined via average pooling.

**Logical connectives:** User supplied rules that use CNN-based predicates can then be grounded to CNN inputs  $x$ . That way one obtains the truth values of these predicates and a fuzzy fulfillment score of the rule (cf. Fig. 1a). We apply continuous t-norm fuzzy logics [10], like Product (P) and Łukasiewicz (Ł) logic, for realizing the rules as mathematical functions. This allows generalization to fuzzy truth values in  $[0, 1]$  of quantifiers ALL ( $\forall$ ), EXISTS ( $\exists$ ); and of logical connectives NOT ( $\neg$ ), AND ( $\wedge$ ), OR ( $\vee$ ), and implication ( $\rightarrow$ ).

**Calibration:** To ensure good calibration of the CMs, i.e., ensure that the predicted confidences coincide with certainties about correctness, we suggest to use the standard measures of a Laplace approximation of a Bayesian neural network [6] and a binary cross-entropy loss.

**Example rules for person detection:** We formalized “body parts indicate persons” (meaning no false negatives due to occlusion) as:

$$F(p, x) := \underbrace{\bigvee_{b \in \text{BodyParts}} Is_b(p, x)}_{IsBodyPart(p, x)} \rightarrow \underbrace{(\exists q \in P : (\bigvee_i Is_{person,i}(q, x)) \wedge \text{closeBy}(p, q))}_{IsPartOfAPerson(p, x)}. \quad (1)$$

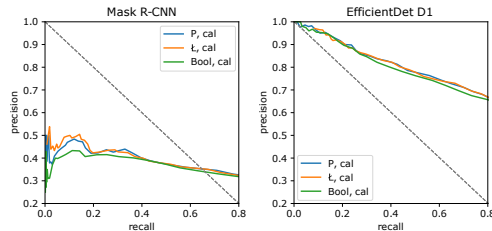
## 4 Experiments and Results

For experimental validation we considered the object detectors Mask R-CNN [4], and EfficientDet D1 [15] (EffDet) trained on MS COCO 2017 [7], with pixel accuracies for  $Is_{person}$  of 0.960 (MR) and 0.930 (EffDet). MS COCO was also used to define the ground truth for the considered body parts eye, arm, wrist, leg, ankle as in [11]. Our main findings for the example rule Eq. 1 were (cf. Tab. 1):

*Calibrated CMs:* Our calibrated CMs showed good performance comparable to the vanilla Dice loss approach from [11], but were on average much better calibrated (Tab. 1b).

Table 1: Results for identifying detector false negatives with the rule from Eq. 1

(a) ROC AUC, and F1 at $\overline{\text{best}}$ and 0.5 threshold					(b) CM avg. expected calibration error (ECE) [9] and set intersection over union (sIoU) [3] at $\overline{\text{best}}$ and 0.5 threshold for MR model.				
Logic	Pixel-level		Image-level		ECE	$\overline{\text{sIoU}}$	sIoU <sub>0.5</sub>		
	AUC	AUC	F1 <sub>0.5</sub>	$\overline{\text{F1}}$					
MR	Bool	0.829	0.619	0.146	0.455	<b>BCE, cal</b> $0.001 \pm 0.001$	<b>0.297 <math>\pm</math> 0.102</b>	0.218 $\pm$ 0.126	
	L	0.833	0.632	0.282	0.462				
	L cal	0.833	0.632	<b>0.295</b>	0.461				
	P	0.833	0.632	0.244	<b>0.466</b>				
	P cal	<b>0.834</b>	<b>0.632</b>	0.243	0.465				
EffDet	Bool	0.840	0.671	0.421	0.749	Dice loss	0.010 $\pm$ 0.008	0.265 $\pm$ 0.079	0.263 $\pm$ 0.081
	L	0.843	0.690	0.557	0.752				
	L cal	<b>0.847</b>	0.695	<b>0.592</b>	<b>0.755</b>				
	P	0.843	0.689	0.500	0.751				
	P cal	0.847	<b>0.696</b>	0.523	0.753				



(c) PR curves (upper left) for different logics

*Error retrieval:* Given the operation-time setting, formula truth values and an alarm threshold are used to identify CNN errors (false negatives in case of Eq. 1), both per-pixel and per-image (Tab. 1a). A substantial amount of respective detection errors could be retrieved that way. On image level, decent precision-recall balances could be found, e.g., for EffDet 10% of erroneous images identified at precision  $\geq 0.95$  (cf. Fig. 1c).

*Advantages of fuzziness and calibration:* Both calibration (cal) and fuzziness slightly but consistently outperform uncalibrated and Boolean (Bool) rules, i.e., standard intersection and union operations on binary masks (Fig. 1c, Tab. 1a). In case the alarm threshold cannot be tuned, fuzziness substantially outperformed Bool, as F1<sub>0.5</sub> scores in Tab. 1a show.

*Other use-cases* Manual qualitative inspection showed that corner cases selected by logical consistency scores prove useful for identification of interesting failure types (cf. Fig. 1b).

## 5 Outlook

Our suggested framework setup proved to be promising. Hence, we would like to see it evaluated on more models, datasets, rules, as well as in an end-to-end fashion for error removal and robustification against outliers, and in user-studies for other use-cases. We hope our work serves as a starting point for rich post-hoc verification of symbolic rules on perception CNNs.

## References

1. Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A.: Network dissection: Quantifying interpretability of deep visual representations. In: Proc. 2017 IEEE Conf.

- Comput. Vision and Pattern Recognition. pp. 3319–3327. IEEE Computer Society (2017). <https://doi.org/10.1109/CVPR.2017.354>
2. Diligenti, M., Gori, M., Saccà, C.: Semantic-based regularization for learning and inference. *Artificial Intelligence* **244**, 143–165 (2017). <https://doi.org/10.1016/j.artint.2015.08.011>
  3. Fong, R., Vedaldi, A.: Net2Vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In: Proc. 2018 IEEE Conf. Comput. Vision and Pattern Recognition. pp. 8730–8738. IEEE Computer Society (2018). <https://doi.org/10.1109/CVPR.2018.00910>
  4. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 2980–2988 (2017). <https://doi.org/10.1109/ICCV.2017.322>
  5. Koh, P.W., Nguyen, T., Tang, Y.S., Musmann, S., Pierson, E., Kim, B., Liang, P.: Concept bottleneck models. In: Proc. 2020 Int. Conf. Machine Learning. pp. 5338–5348. PMLR (2020)
  6. Kristiadi, A., Hein, M., Hennig, P.: Being bayesian, even just a bit, fixes overconfidence in relu networks. In: Proc. 37th Int. Conf. Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event. Proceedings of Machine Learning Research, vol. 119, pp. 5436–5446. PMLR (2020)
  7. Lin, T.Y., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: Proc. 13th European Conf. Computer Vision - Part V. Lecture Notes in Computer Science, vol. 8693, pp. 740–755. Springer International Publishing (2014)
  8. Losch, M., Fritz, M., Schiele, B.: Interpretability beyond classification output: Semantic bottleneck networks. In: Proc. 3rd ACM Comput. Science in Cars Symp. Extended Abstracts (2019)
  9. Naeini, M.P., Cooper, G., Hauskrecht, M.: Obtaining well calibrated probabilities using bayesian binning. Proceedings of the AAAI Conference on Artificial Intelligence **29**(1) (2015). <https://doi.org/10.1609/aaai.v29i1.9602>
  10. Novák, V., Perfilieva, I., Mockor, J.: Mathematical Principles of Fuzzy Logic. The Springer International Series in Engineering and Computer Science, Springer US (1999). <https://doi.org/10.1007/978-1-4615-5217-8>
  11. Schwalbe, G.: Verification of size invariance in DNN activations using concept embeddings. In: Artificial Intelligence Applications and Innovations. pp. 374–386. IFIP Advances in Information and Communication Technology, Springer International Publishing (2021). [https://doi.org/10.1007/978-3-030-79150-6\\_30](https://doi.org/10.1007/978-3-030-79150-6_30)
  12. Schwalbe, G.: Concept Embedding Analysis: A Review. arXiv:2203.13909 (2022)
  13. Schwalbe, G., Schels, M.: A survey on methods for the safety assurance of machine learning based systems. In: Proc. 10th European Congress Embedded Real Time Software and Systems (2020), <https://hal.archives-ouvertes.fr/hal-02442819>
  14. Schwalbe, G., Wirth, C., Schmid, U.: Enabling verification of deep neural networks in perception tasks using fuzzy logic and concept embeddings. CoRR **abs/2201.00572** (2022), <https://arxiv.org/abs/2201.00572>
  15. Tan, M., Pang, R., Le, Q.V.: EfficientDet: Scalable and efficient object detection. In: Proc. 2020 IEEE/CVF Conf. Comput. Vision and Pattern Recognition. pp. 10781–10790 (2020)
  16. Zhang, R., Madumal, P., Miller, T., Ehinger, K.A., Rubinstein, B.I.P.: Invertible concept-based explanations for CNN models with non-negative concept activation vectors. In: Proc. 35th AAAI Conf. Artificial Intelligence. vol. 35, pp. 11682–11690. AAAI Press (2021)

# Learning Event-recording Automata Passively

(extended abstract)

Anirban Majumdar<sup>1</sup>, Sayan Mukherjee<sup>1</sup>, and Jean-François Raskin<sup>1</sup>

Université Libre de Bruxelles, Belgium

*Inferring finite-state automata* for regular languages has been an active area of research for decades. The main goal is, given a regular language  $L$ , to learn a finite-state automaton representing  $L$ . There are primarily two approaches to solve this problem: (i) *active learning*: where the learner is aided by a teacher, who knows  $L$ ; the learner builds an automaton by querying the teacher, and (ii) *passive learning*: where the learner is provided with a labelled set of words, i.e. a set of words that are *positive* and another set of words that are *negative*; the learner tries to infer a finite-state automaton that accepts every positive word and rejects every negative word.

*Event-recording automata* is a subclass of Timed Automata. The latter is a popular choice for modelling real-time systems, and has been studied extensively over the past couple of decades. Timed Automata is more expressive than event-recording automata, however, with the expressive power comes the difficulty of solving some important decision problems. For example, timed automata are not determinizable in general, and further, checking inclusion between two non-deterministic timed automata is undecidable. To recover determinizability, the class of event-recording automata was introduced by Alur et al. in [1]. This class of automata has since been of interest because it is sufficiently expressive while keeping several important decision problems decidable.

*Learning models of timed systems.* Timed Automata extends finite state automata with non-negative real-valued variables, called *clocks*. The transitions of timed automata have two additional attributes over finite-state automata: (i) *guards* – these are expressions over clocks that specify for which values of the clocks the transition is enabled, and (ii) *resets* – these specify which clocks' values will be set back to 0 once the transition is taken. These attributes contribute to the challenges in learning a timed automaton:

1. there is no bound on the number of clocks in a timed automaton. It is difficult to infer how many clocks are necessary to represent a language,
2. an arbitrary number of clocks can be reset on a transition,
3. the learning algorithms need to infer the guards present on the transitions.

To circumvent some of these challenges, several works have tried to learn subclasses of Timed Automata that restrict the number of clocks, for example, [2] studies learning of one-clock deterministic timed automata. In a more recent work, [6] studies the problem for deterministic Timed Automata, however, they reset a new clock on every transition.

In an event-recording automaton, every clock is associated with an event, and in every transition the clock that is associated to the event present in the transition, is reset. Therefore, the first two challenges mentioned above do not arise when learning event-recording automata. This makes learning event-recording automata feasible. The works [4,3,5] propose algorithms for learning event-recording automata in an active learning framework.

*In this work* we propose an algorithm ‘LEAP’, that learns an event-recording automaton from a given sample of positive and negative examples using a passive learning approach. A well known passive learning algorithm for inferring finite-state automata for regular languages is RPNI. LEAP adapts RPNI to infer event-recording automata. As input, we consider finite words over pairs of events and guards, such words are called *guarded words*. A guarded word represents a set of timed words; we call those timed words *concretizations* of the guarded word. Given a set of positive guarded words and a set of negative guarded words, LEAP returns an ERA that *accepts* every concretization of the positive words and *rejects* every concretization of the negative ones.

Similar to RPNI, LEAP first constructs a tree-like ERA based on the positive set of guarded words. It then tries to merge states of the tree, based on a total order defined over guarded words. After every merge, LEAP checks if the new automaton accepts a concretization of any of the negative words; if it does, the latest merge is undone and LEAP tries to merge another pair of states. When no two states remain that can be merged, LEAP terminates.

To check if a merge is allowed, LEAP checks if a concretization of a negative guarded word is accepted by the computed automaton. We show that checking if a guarded word has a concretization that is accepted by a given ERA, is an NP-complete problem. We prove this by showing a reduction from 3SAT. We then propose an SMT-based algorithm for checking this inclusion.

We have a prototype implementation of LEAP. We are yet to implement a total order over guarded words for choosing states to merge. Currently, we use a partial order over guarded words, similar to the regular case, that is known to be sufficient for RPNI to terminate.

Lastly, we prove a completeness result for LEAP – when provided with a ‘good’ input, the algorithm should return a ‘good’ automaton. To get such a guarantee, we restrict ourselves to guarded words of a special kind – where every guard present in a word is a ‘simple constraint’. Given a language  $L$  recognizable by an ERA  $\mathcal{A}$ , we can associate a positive set and a negative set of ‘simple’ guarded words, that, when provided as an input, LEAP returns an ERA  $\mathcal{A}_*$  such that the timed languages of both  $\mathcal{A}$  and  $\mathcal{A}_*$  are the same.



This is still a work in progress.

## References

1. Alur, R., Fix, L., Henzinger, T.A.: Event-clock automata: A determinizable class of timed automata. *Theor. Comput. Sci.* **211**(1-2), 253–273 (1999)

2. An, J., Chen, M., Zhan, B., Zhan, N., Zhang, M.: Learning one-clock timed automata. In: TACAS (1). Lecture Notes in Computer Science, vol. 12078, pp. 444–462. Springer (2020)
3. Grinchtein, O., Jonsson, B., Leucker, M.: Learning of event-recording automata. *Theor. Comput. Sci.* **411**(47), 4029–4054 (2010)
4. Grinchtein, O., Jonsson, B., Pettersson, P.: Inference of event-recording automata using timed decision trees. In: CONCUR. Lecture Notes in Computer Science, vol. 4137, pp. 435–449. Springer (2006)
5. Lin, S., André, É., Dong, J.S., Sun, J., Liu, Y.: An efficient algorithm for learning event-recording automata. In: ATVA. Lecture Notes in Computer Science, vol. 6996, pp. 463–472. Springer (2011)
6. Waga, M.: Active learning of deterministic timed automata with myhill-nerode style characterization. In: CAV (1). Lecture Notes in Computer Science, vol. 13964, pp. 3–26. Springer (2023)

# MILP Formulations for Passive Learning

Simon Dierl<sup>1</sup>  and Falk Howar<sup>1,2</sup> 

<sup>1</sup> TU Dortmund University, Dortmund, Germany

{simon.dierl,falk.howar}@tu-dortmund.de

<sup>2</sup> Fraunhofer ISST, Dortmund, Germany

## 1 Introduction

Learning algorithms derive models of the behavior of black-box systems through observation of their output. We study *automata* learning algorithms that generate FSAs and Moore machines which are human-readable and amenable to automated model checking. *Active* learning algorithms such as  $L^*$  [1] interact with a system and eventually produce a perfect model. *Passive* learning only processes samples of a system’s output and only produces imperfect models. However, passive learning can operate offline and in scenarios not tractable by active learning.

We focus on passive learning algorithms that produce Moore machines that classify observed behavior. The identification of *minimal* automata matching a given set of inputs is NP-complete [3]. Classic algorithms extend the approach used by  $k$ -Tails [2]: a prefix tree of the observed traces is constructed and then minimized by merging states according to an equivalence rule. Recently, *model-based* learning approaches have gained traction. These approaches define correct classification of the given samples as constraints and then invoke a solver to produce a valid model. Various works have employed SAT solving [8, 9], SMT solving [6], and MILP solving [5]. Since MILP solving creates minimized models in a single solver call (e.g., minimal states or minimal fan-out), it is of particular interest in creating small, readable and efficiently analyzable models.

However, the authors of [5] do not propose a single MILP formulation, but solve for multiple objective functions and select the best solution on their *validation* data set. We seek to address this gap by finding a formulation that a) is fast to solve and b) yields automata with good prediction power on unknown samples.

## 2 MILP Formulations

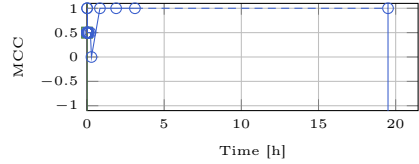
We base our approach on Shvo et al.’s [5]. The learning problem is encoded by fixing a maximum automaton size and solving i) a mapping from prefix tree to automaton state, ii) a matching transition function, and iii) the automaton’s output function. The solver then attempts to minimize the number of live states. Based on this, we investigate potential optimizations and changes to the model:

1. fixing the output function to certain states ( $q_0$  outputs  $o_0$  etc.),
2. optimizing for minimal fan-out after minimizing states,
3. instead of using constraints, optimizing for minimal errors first, and
4. use of symmetry-braking constraints such as BFS-based ones [7].





(a) fixed acceptance &amp; minimized errors



(b) minimized FO &amp; BFS symmetry

**Fig. 1.** Prediction power of each intermediate solution for two formulations.

### 3 Preliminary Results

Preliminary results on are varied. As illustrated in Fig. 1, formulation choice can impact solving time by factor of ca.  $10^4$ . However, there appears to be no clear “silver bullet” formulation for prediction power according to preliminary experiments.

Next steps are the selection of some formulations with an acceptable trade-off between prediction power and solving time and a comprehensive comparison to SAT-based learning with DFA-Inductor [9], Shvo et al.’s DISC [5], and the classic RPNI [4] algorithm on a wide selection of data.

**Acknowledgments.** The authors would like to thank the Federal Government and the Heads of Government of the Länder, as well as the Joint Science Conference (GWK), for their funding and support within the framework of the NFDI4Ing consortium. This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – projects [495857894 \(STING\)](#) and [442146713 \(NFDI4Ing\)](#).

### References

1. Angluin, D.: Learning regular sets from queries and counterexamples. *Inf. Comput.* **75**(2) (1987)
2. Biermann, A.W., Feldman, J.A.: On the synthesis of finite-state machines from samples of their behavior. *IEEE Trans. Comput.* **C-21**(6) (1972)
3. Karp, R.M.: Reducibility among combinatorial problems. In: *Complexity of Computer Computations* (1972)
4. Oncina, J., García, P.: Inferring regular languages in polynomial updated time. In: *SNRFAI* (1992)
5. Shvo, M., Li, A.C., Toro Icarte, R., McIlraith, S.A.: Interpretable sequence classification via discrete optimization. *Proc. AAAI Conf. Artif. Intell.* **35**(11) (2021)
6. Tappier, M., Aichernig, B.K., Lorber, F.: Timed automata learning via SMT solving. In: *NFM* (2022)
7. Ulyantsev, V., Zakirzyanov, I., Shalyto, A.: BFS-based symmetry breaking predicates for DFA identification. In: *LATA* (2015)
8. Wallner, F., Aichernig, B.K., Burghard, C.: It’s not a feature, it’s a bug: Fault-tolerant model mining from noisy data. In: *ICSE* (2024)
9. Zakirzyanov, I., Morgado, A., Ignatiev, A., Ulyantsev, V., Marques-Silva, J.: Efficient symmetry breaking for SAT-based minimum DFA inference. In: *LATA* (2019)

# Advances in SAYNT: Symbiotic Policy Synthesis in POMDPs\*

Roman Andriushchenko<sup>1</sup>, Alexander Bork (✉)<sup>2</sup>, Milan Češka<sup>1</sup>,  
Sebastian Junges<sup>3</sup>, Joost-Pieter Katoen<sup>2</sup>, and Filip Macák<sup>1</sup>

<sup>1</sup> Brno University of Technology, Brno, Czech Republic

<sup>2</sup> RWTH Aachen University, Aachen, Germany

alexander.bork@cs.rwth-aachen.de

<sup>3</sup> Radboud University, Nijmegen, The Netherlands

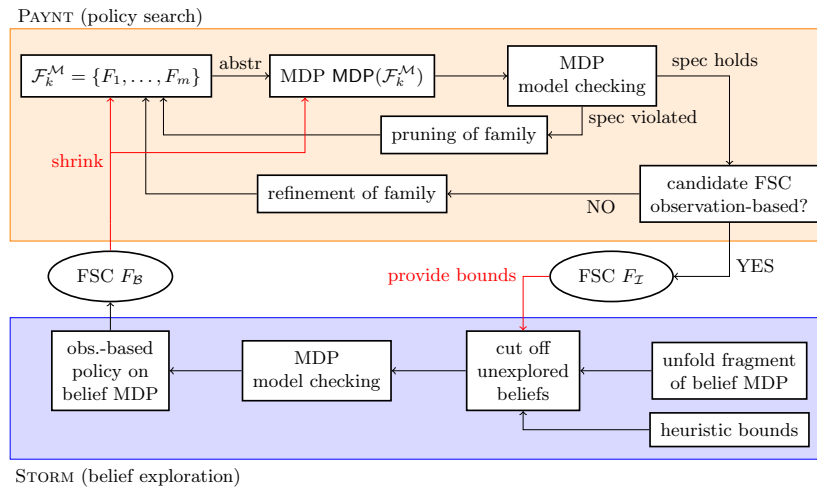


Fig. 1: Schematic depiction of the symbiotic approach

A formidable synthesis challenge is to find a decision-making policy that satisfies temporal constraints even in the presence of stochastic noise. *Markov decision processes (MDPs)* [10] are a prominent model to reason about such policies under stochastic uncertainty. A major shortcoming of MDPs is the assumption that the policy can always depend on the precise state of a system.

*Partially observable MDPs (POMDPs)* overcome this by requiring policies to only depend on *observable* state information, but policy synthesis for POMDPs under specifications such as *the probability to reach the exit is larger than 50%* is an undecidable problem [8]. Nevertheless, in recent years, a variety of approaches have been successfully applied to tackle a range of challenging benchmarks for

\* This work has been supported by the Czech Science Foundation grant GA23-06963S (VESCAA), the ERC AdG Grant 787914 (FRAPPANT) and the DFG RTG 2236/2 (UnRAVeL).

POMDP policy synthesis. From a user perspective, however, it is hard to pick the right approach without detailed knowledge of the underlying methods.

We present SAYNT [2], a symbiotic anytime algorithm tightly integrating two modern, orthogonal methods for synthesising finite-state controllers (FSCs) for policies, represented as deterministic Mealy machines [1]. SAYNT integrates the belief exploration approach as implemented in the STORM model checker [6] with the inductive policy search implemented in PAYNT [4].

In essence, SAYNT relies on the fact that a policy found via one approach can boost the other approach. The key observation is that such a policy is beneficial even when it is sub-optimal in terms of the objective at hand. Fig. 1 sketches the symbiotic approach.

The key idea of belief exploration is that each sequence of observations and actions induces a *belief*—a distribution over POMDP states that reflects the probability to be in a state conditioned on the observations. POMDP policies can decide optimally solely based on the belief [11]. The evolution of beliefs can be captured by a fully observable, yet possibly infinite *belief MDP*. The practical approach that is part of SAYNT is to unfold a finite fragment of this belief MDP and make its frontier absorbing. The resulting MDP approximates the actual belief MDP and can be analysed with STORM’s MDP model checkers [5]. The accuracy of the approximation is improved by policies provided by the policy search part of the integrated approach. These policies improve the values used to approximate the frontier of the belief exploration (see lower part of Fig. 1).

Orthogonally, policy search approaches search a (finite) space of policies [7,9] and evaluate these policies by verifying the induced Markov chain. SAYNT uses the policy-search method implemented in PAYNT that explores spaces of finite-state controllers [3]. PAYNT uses a combination of abstraction-refinement, counterexamples (to prune sets of policies), and increasing a controller’s memory to search large policy spaces, see the upper part of Fig. 1.

In an experimental evaluation [2], we have shown that the symbiosis offers a powerful push-button, anytime synthesis algorithm producing, in the given time, superior and/or more succinct FSCs compared to other state-of-the-art methods.

In addition to already published work, we present recent advancements in the symbiotic policy synthesis approach.

## References

1. Amato, C., Bonet, B., Zilberstein, S.: Finite-state controllers based on Mealy machines for centralized and decentralized POMDPs. In: AAAI. pp. 1052–1058. AAAI Press (2010)
2. Andriushchenko, R., Bork, A., Češka, M., Junges, S., Katoen, J.P., Macák, F.: Search and Explore: Symbiotic policy synthesis in POMDPs. arXiv preprint arXiv:2305.14149 (2023)
3. Andriushchenko, R., Češka, M., Junges, S., Katoen, J.P.: Inductive synthesis of finite-state controllers for POMDPs. In: UAI. vol. 180, pp. 85–95. PMRL (2022)
4. Andriushchenko, R., Češka, M., Junges, S., Katoen, J.P., Stupinský, Š.: PAYNT: a tool for inductive synthesis of probabilistic programs. In: CAV. LNCS, vol. 12759, pp. 856–869. Springer (2021)

5. Bork, A., Katoen, J.P., Quatmann, T.: Under-approximating expected total rewards in POMDPs. In: TACAS (2). LNCS, vol. 13244, pp. 22–40. Springer (2022)
6. Dehnert, C., Junges, S., Katoen, J.P., Volk, M.: A Storm is coming: A modern probabilistic model checker. In: CAV. LNCS, vol. 10427, pp. 592–600. Springer (2017)
7. Hansen, E.A.: Solving pomdps by searching in policy space. In: UAI. pp. 211–219. Morgan Kaufmann (1998)
8. Madani, O., Hanks, S., Condon, A.: On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence* **147**(1), 5–34 (2003)
9. Meuleau, N., Kim, K., Kaelbling, L.P., Cassandra, A.R.: Solving pomdps by searching the space of finite policies. In: UAI. pp. 417–426. Morgan Kaufmann (1999)
10. Puterman, M.L.: Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons (1994)
11. Smallwood, R.D., Sondik, E.J.: The optimal control of partially observable Markov processes over a finite horizon. *Oper. Res.* **21**(5), 1071–1088 (1973)

# Predicate Decision Diagrams for Explainable Policy Representation

Debraj Chakraborty<sup>1</sup>, Clemens Dubslaff<sup>2</sup>, Sudeep Kanav<sup>1</sup>, Jan Křetínský<sup>1</sup>, and Christoph Weinhuber<sup>3</sup>

<sup>1</sup> Masaryk University, Brno, Czechia

<sup>2</sup> Eindhoven University of Technology, The Netherlands

<sup>3</sup> Technical University of Munich, Germany

**Abstract.** Safety-critical controllers of complex systems are hard to construct manually. Automatic approaches such as synthesis or learning provide a tempting alternative. However, one of the fundamental hurdles on the way to adoption is the lack of explainability of the constructed controllers. Fortunately, explainable representation of the controllers can often be mined from them. To this end, decision-tree (DT) learning has been prevalently used, due to a number of advantages over other popular representation structures such as binary decision diagrams (BDD). In this work, we introduce *predicate decision diagrams* (PDD), an extension of BDD with predicates as variables. By embracing some of the DT concepts, PDD eliminates BDD’s disadvantages on explainability. As a result, we provide an algorithm for efficient construction of PDD as an explainable representation of controllers. This brings BDD-based methods, otherwise popular in verification, back in game.

**Keywords:** Binary decision diagram · Decision tree · Learning · Explainability · Decision making and control · Strategy synthesis.

**Controllers** of dynamic systems are hard to construct for a number of reasons: (i) the systems—be it software or cyber-physical systems—are complex due to phenomena such as concurrency, hybrid dynamics, uncertainty, and their sizes grow steadily; (ii) they are often safety-critical, with a number of complex specifications to adhere to. Consequently, manually crafting the controllers is very error-prone. *Automated controller synthesis* or *learning* provide a tempting alternative, where controllers are constructed algorithmically and their correctness is ensured with human participation limited only to the specification process. Since the problem of constructing a controller can be transformed into constructing a winning strategy (a.k.a. policy) in a game played by the controller and its environment, there are numerous approaches solving the problem.

**Explainability** of the automatically constructed controllers (or rather its lack) poses, however, a fundamental obstacle to practical use of automated construction of controllers. Indeed, an unintelligible controller can hardly be accepted by an engineer, who is supposed to maintain the system, adapt its implementation,

provide arguments in the certification process, not to speak of legally binding explainability of learnt controllers [1–3]. Unfortunately, most of the synthesis and learning approaches compute controllers in the shape of huge tables of state-action pairs, describing which actions can be played in each state, or of cryptic black boxes. Such representations are typically so huge that they are utterly incomprehensible. An explainable representation should be (i) small enough to be contained in the human working memory, and (ii) in a simple enough formalism so that the decisions can be read off in a way that relates them to the real states of the system.

**Decision trees** (DT), e.g. [14], have a very specific, even unique position among machine-learning models due to their interpretability. As a result, they have become the predominant formalism for explainable representations of controllers [8, 9, 4, 7, 12, 5, 6, 13] over the past decade. Figure 1 shows an example of policy representation using DTs.

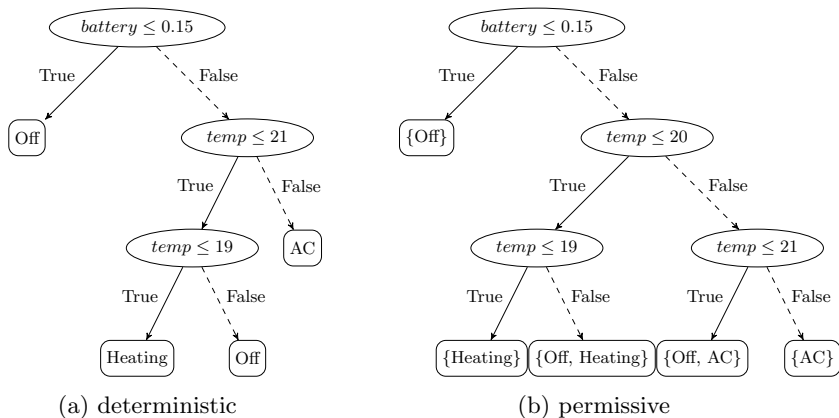


Fig. 1: An example of how a decision tree can represent a policy. (a) shows a deterministic policy, (b) a permissive one with multiple actions at some states.

In contrast, binary decision diagrams (BDD) [10] are very popular to represent large objects compactly, exploiting bit-wise representation with automatic reduction mechanisms. However, they have not been used much for policy representation on the output. The reasons are due to several advantages of DT:

- ❶ DT can use more general predicates when arriving at a decision, while BDD only use a 0/1 values on each bit of the encoding. Since deciding one predicate may depend on a number of bits, DT may be smaller. Since richer predicates may encode domain knowledge easier, DT can be more explainable.
- ❷ DT can vary the order of predicates examined on each branch. This gives more flexibility and could lead to substantial space savings in DT (but practically disables BDD’s merging isomorphic subgraphs as they rarely occur).

- ③ DT learning allows “don’t care” values to be set in any way so that the learnt DT is the smallest. In contrast, BDD natively represent only fully specified sets, with no heuristic how to resolve the “don’t cares” resulting typically in large representation.

We show that despite all these disadvantages BDDs can be a competitive alternative to DT. To this end, we introduce **predicate decision diagrams (PDD)**, which on the one hand, can be seen as regular BDD over a certain special bitstring representation, and thus inherit all the BDD advantages; on the other hand, they emulate DT, profiting from their explainability and smaller size. In particular:

- ① We collect the predicates mined by the DT representation of a given controller and use them as “variables” in the PDD. This eliminates the issue ①.
- ② We apply sifting (variable reordering techniques for BDD) and merging isomorphic subgraphs to improve the size of the PDD, alleviating the issue ②.
- ③ We use Coudert and Madre’s method [11] to restrict the support of BDD on care sets, compressing BDDs further and fixing issue ③.

Notably, due to ① and ②, branches can arise in the PDD that are contradictory, e.g.  $temp \leq 19 \wedge \neg temp \leq 20$ . Detecting and eliminating them further reduces the size. Interestingly and fortunately, finding good variable ordering and eliminating contradictory branches commutes, which eases the search significantly.

**Our contribution** can be summarized as follows:

- We introduce *PDD*, a BDD structure allowing for both explainable and small representation of controllers.
- We design an algorithm effectively synthesizing PDD representations and prove its optimality properties, in particular the reasoning modulo DT theory for branch consistency.
- We experimentally demonstrate that we almost close the gap between BDD and DT, in half of the cases matching the size of DT or even outrunning it, while retaining the same level of explainability as DT. In particular, we reduce the gap between BDD and DT sizes on average by 98%; the median being match between the two; and due to several extreme savings, PDD are on average 15% percent *smaller* than DT!
- In conclusion, we present a viable alternative to DT, breaking their decade-long hegemony for explainable representation and bringing the traditional BDD-based tools back in the game.

## References

1. Ethics guidelines for trustworthy AI - european commission, directorate-general for communications networks, content and technology (2019), <https://data.europa.eu/doi/10.2759/177365>
2. Four principles of explainable artificial intelligence - (U.S.) national institute of standards and technology (NIST) (2020), <https://doi.org/10.6028/NIST.IR.8312-draft>

3. Proposal for a regulation laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts, com(2021) 206 final - european commission (2021), <https://ec.europa.eu/transparency/regdoc/rep/1/2021/EN/COM-2021-206-F1-EN-MAIN-PART-1.PDF>
4. Ashok, P., Brázdil, T., Chatterjee, K., Křetínský, J., Lampert, C.H., Toman, V.: Strategy representation by decision trees with linear classifiers. In: QEST. Lecture Notes in Computer Science, vol. 11785, pp. 109–128. Springer (2019)
5. Ashok, P., Jackermeier, M., Jagtap, P., Křetínský, J., Weininger, M., Zamani, M.: dtcontrol: decision tree learning algorithms for controller representation. In: Ames, A.D., Seshia, S.A., Deshmukh, J. (eds.) HSCC '20: 23rd ACM International Conference on Hybrid Systems: Computation and Control, Sydney, New South Wales, Australia, April 21–24, 2020. pp. 30:1–30:2. ACM (2020). <https://doi.org/10.1145/3365365.3383468>, <https://doi.org/10.1145/3365365.3383468>
6. Ashok, P., Jackermeier, M., Křetínský, J., Weinhuber, C., Weininger, M., Yadav, M.: dtcontrol 2.0: Explainable strategy representation via decision tree learning steered by experts. In: TACAS (2). Lecture Notes in Computer Science, vol. 12652, pp. 326–345. Springer (2021)
7. Ashok, P., Křetínský, J., Larsen, K.G., Coënt, A.L., Taankvist, J.H., Weininger, M.: SOS: safe, optimal and small strategies for hybrid Markov decision processes. In: QEST. Lecture Notes in Computer Science, vol. 11785, pp. 147–164. Springer (2019)
8. Brázdil, T., Chatterjee, K., Chmelik, M., Fellner, A., Křetínský, J.: Counterexample explanation by learning small strategies in Markov decision processes. In: CAV (1). Lecture Notes in Computer Science, vol. 9206, pp. 158–177. Springer (2015)
9. Brázdil, T., Chatterjee, K., Křetínský, J., Toman, V.: Strategy representation by decision trees in reactive synthesis. In: TACAS (1). Lecture Notes in Computer Science, vol. 10805, pp. 385–407. Springer (2018)
10. Bryant, R.E.: Graph-Based Algorithms for Boolean Function Manipulation. IEEE Transactions on Computers **35**, 677–691 (1986)
11. Coudert, O., Madre, J.: A unified framework for the formal verification of sequential circuits. In: 1990 IEEE International Conference on Computer-Aided Design. Digest of Technical Papers. pp. 126–129 (1990). <https://doi.org/10.1109/ICCAD.1990.129859>
12. Garg, P., Neider, D., Madhusudan, P., Roth, D.: Learning invariants using decision trees and implication counterexamples. In: POPL. pp. 499–512. ACM (2016)
13. Jüngermann, F., Křetínský, J., Weininger, M.: Algebraically explainable controllers: decision trees and support vector machines join forces. Int. J. Softw. Tools Technol. Transf. **25**(3), 249–266 (2023)
14. Mitchell, T.M.: Machine learning. McGraw Hill series in computer science, McGraw-Hill (1997)