Learning Linear-time Temporal Logic

Internship Proposal

Adrien Pommellet
adrien [at] lrde.epita.fr

January 25, 2024

Topics of the proposal: passive learning; linear-time temporal logic; SAT solving; C++ programming

Passive learning is the act of computing a theoretical model of a system from a given set of data, without being able to acquire further information by actively querying said system. The input data may have been gathered through monitoring, collecting executions and outputs of systems. Automata and logic formulas tend to be the most common models, as they allow one to better express the behaviour and properties of systems of complex or even entirely opaque design.

Linear-time Temporal Logic LTL [4] remains one of the most widely used formalisms for specifying temporal properties of reactive systems. It applies to finite or infinite execution traces, and for that reason fits the passive learning framework very well: a LTL formula is a concise way to distinguish between correct and incorrect executions. The LTL learning problem, however, is anything but trivial: even simple fragments on finite traces are NP-complete [2], and consequently recent algorithms tend to leverage SAT solvers [3].

Due to performance issues, it is not at the moment possible to learn *minimal* LTL formulas on large samples. The immediate purpose of this project is therefore to improve upon the learning process by computing a compact intermediate representation of the original sample based on Kripke structures. Optimizing the SAT encoding of LTL's semantics and topology-guided parallel SAT solving [5] also belong to our areas of interest.

This internship requires both theoretical and practical skills. On the one hand, despite recent results pertaining to the passive learning problem for ω-automata [1], no such algorithm for state-based Kripke structures exists yet to our knowledge. On the other hand, we intend on writing a C++ program and employ state-of-the-art SAT solvers to outperform existing approaches. For that reason, some C++ experience is strongly recommended; elementary knowledge of model-checking theory is helpful but not mandatory.

References

[1] León Bohn and Christof Löding. Constructing Deterministic ω-Automata from Examples by an Extension of the RPNI Algorithm. In *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*.

[2] Nathanaël Fijalkow and Guillaume Lagarde. The complexity of learning linear temporal formulas from examples. In Jane Chandlee, Rémi Eyraud, Jeff Heinz, Adam Jardine, and Menno van Zaanen, editors, *Proceedings of the 15th International Conference on Grammatical Inference, 23-27 August 2021, Virtual Event*, volume 153 of *Proceedings of Machine Learning Research*, pages 237–250. PMLR, 2021.

[3] Daniel Neider and Ivan Gavran. Learning linear temporal properties. In Nikolaj S. Bjørner and Arie Gurfinkel, editors, *2018 Formal Methods in Computer Aided Design, FMCAD 2018, Austin, TX, USA, October 30 - November 2, 2018*, pages 1–10. IEEE, 2018.

[4] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57, 1977.

[5] Heinz Riener. Exact synthesis of ltl properties from traces. In *2019 Forum for Specification and Design Languages (FDL)*, pages 1–6, 2019.