LearnLib: 10 years later

Markus Frohme[1] (✉), Falk Howar[1,2], and Bernhard Steffen[1]

[1] TU Dortmund University, Dortmund, Germany
<firstname>.<lastname>@cs.tu-dortmund.de
[2] Fraunhofer ISST, Dortmund, Germany

Abstract. In 2015, LearnLib, the open-source framework for active automata learning, received the prestigious CAV artifact award. This paper presents the advancements made since then, highlighting significant additions to LearnLib, including state-of-the-art algorithms, novel learning paradigms, and increasingly expressive models. Our efforts to mature and maintain LearnLib have resulted in its widespread use among researchers and practitioners alike. A key factor in its success is the achieved compositionality which allows users to effortlessly construct thousands of customized learning processes tailored to their specific requirements. This paper illustrates these features through the development of a learning process for the life-long learning of procedural systems. This development can be easily replicated and modified using the latest public release of LearnLib.

Keywords: (Active) Automata Learning · Monitoring · Refinement · Open-Source · Library · Java

1 Introduction

Active automata learning describes the process of inferring an automaton-based model from a hardware or software system by means of actively querying it. What originally started as a theoretical concept for language inference [9] has in recent decades gained a lot of interest from practitioners due to the role of automata learning as the key enabler of practical model-based quality assurance. There exist several success stories that underline its practical relevance [67, 47, 3, 20, 2, 4, 72, 32, 64].

Tools are of paramount importance when applying these theoretical concepts to practical scenarios. Unfortunately, by being mostly research-driven, many tools are no longer developed or even maintained once the main researcher or research group leaves the project. Specifically for active automata learning, we have observed this trend with tools such as *libALF* [16] (last release in 2011), *RALT* [70] (never published, internal use at France Telecom until 2014), *AIDE* [49] (development ceased in 2015), *Sp2Learn* [11] (development ceased in 2016), *Tomte* [1] (latest release in 2016), *SymbolicAutomata* [25] (no major contributions since 2019), or *ROLL* [53] (somewhat regular contributions, but no stable releases). Currently, we are only aware of *AALpy* [59] that is being actively developed and maintained.

This paper highlights the significant advancements of LearnLib a decade after receiving the CAV artifact award in 2015 [45]. These advancements include not only new paradigms for addressing the challenges of practical automata learning and the evolution of automata learning towards more expressive formalisms such as procedural systems, but also enhanced compositionality. We demonstrate how users can now easily construct thousands of customized learning processes tailored to their specific needs. In particular, researchers can easily benchmark their new learning algorithms against the state of the art by simply replacing components in established learning processes, while application developers can seamlessly compose tailored learning processes to evaluate their suitability for specific requirements. This paper showcases this potential through the development of a learning process for the life-long learning of procedural systems.

The open structure of LearnLib has inspired numerous researchers to contribute by providing case studies and algorithms which can now be seamlessly integrated as illustrated in this paper. We aim to further establish LearnLib as a comprehensive resource for automata learning.

LearnLib is written in Java, open-source[3], and released under the Apache 2.0 license[4]. It is deployed to Maven Central, the de-facto standard for various build tools in the Java ecosystem, and therefore can be directly used in many projects. Consequently, the developments presented in this paper can be easily replicated and modified.

Outline We continue in Section 2 with an overview of the current features of LearnLib and highlight the major additions of the last decade. Section 3 showcases the flexibility of LearnLib to easily configure a requirements-based learning process. In Section 4, we summarize our efforts to improve the development process of LearnLib and the impact it had on the research community. Section 5 concludes the paper and gives an outlook on the future of LearnLib.

2 New Features in LearnLib

Active automata learning usually operates within the *minimally adequate teacher* framework as proposed by Angluin [9]. During the exploration phase, a *learning algorithm* (or learner) queries the *system under learning* (SUL) for its behavior through a *membership oracle* to construct a tentative hypothesis model of the SUL. After hypothesis stabilization, an *equivalence oracle* then checks in the verification phase whether the hypothesis model and the SUL are equivalent and, if not, returns a counterexample (a witness for in-equivalence). This counterexample is used by the learner to refine the current hypothesis model, which triggers a subsequent exploration phase. The two phases alternate until the equivalence oracle no longer finds any counterexamples. For details, see [74].

Figure 1 sketches the current features of LearnLib for the involved concepts *learning processes* are composed of. For instantiating a learning process, the user

[3] `https://github.com/LearnLib/learnlib`
[4] `https://www.apache.org/licenses/LICENSE-2.0`

only needs to define the symbolic interactions with the system (`Symbols`) and provide a means to access it (`SUL`). All remaining parts are provided by LearnLib and can be combined depending on the requirements of the user. In the following, we highlight the novel additions to LearnLib.

Active Learning Algorithms Experiments [43] show that one bottleneck of active automata learning typically lies in the performance of the SUL for answering queries. As a result, many learners aim at reducing the number of queries that they pose and the length thereof. We have developed the concept of *lazy partition refinement* [41] which distills the idea of the TTT algorithm [46] to represent hypothesis states with a redundancy-free, prefix-closed set of access sequence while simultaneously distinguishing between them with a redundancy-free, suffix-closed set of discriminators. LearnLib provides two incarnations of this concept in the form of the L^λ algorithm (based on the L^* algorithm [9]) and the TTT^λ algorithm (based on the TTT algorithm [46]).

A different approach to pursue this goal concerns refining the way in which learners communicate with the SUL. The ADT learner [33] and the $L^\#$ learner [78] utilize adaptive distinguishing sequences to separate between hypothesis states. By adaptively deciding which inputs to query next, they can discriminate between more states, thus reducing the number of queries needed. The implementation of the $L^\#$ algorithm was contributed by Ferreira et al. as part of their work in [27].

The amount of system inputs can easily cause performance problems, too. To address this issue, we implemented the concept of *automated alphabet abstraction refinement* (AAAR) [42]. AAAR introduces an orthogonal learning process on the behavioral equivalence classes of the input symbols (as long as such a finite partition exists), incorporating input symbols in the learning process only when really needed. It is worth noting that AAAR is not a single algorithm but a generic concept which can be combined with various of the existing (regular) learning algorithms.

Finally, Bayram [14] extended some of the existing learning algorithms to natively support inferring Moore machines [58]. Being able to record intermediate state outputs can increase the expressiveness of membership queries, boosting the performance of learning processes in case the SUL supports these semantics, too.

Another challenge for practical automata learning is the question whether the chosen formalism is expressive enough to capture the system properties of interest. We extended the scope of LearnLib to support the active inference of context-free (or procedural) systems. The notion of *systems of procedural automata* (SPAs) [35], *systems of behavioral automata* (SBAs) [36], and *systems of procedural Mealy machines* (SPMMs) [34] describe systems that are composed of multiple regular procedures that can mutually call each other. Similar to AAAR, this allows for a generic *meta-learner* that can be parameterized in the concrete learner(s) used for the internal procedures. Furthermore, the existing OP algorithm [40] and TTT algorithm [46] have been extended [44] to support the inference of *visibly push-down automata* (VPAs) [8].

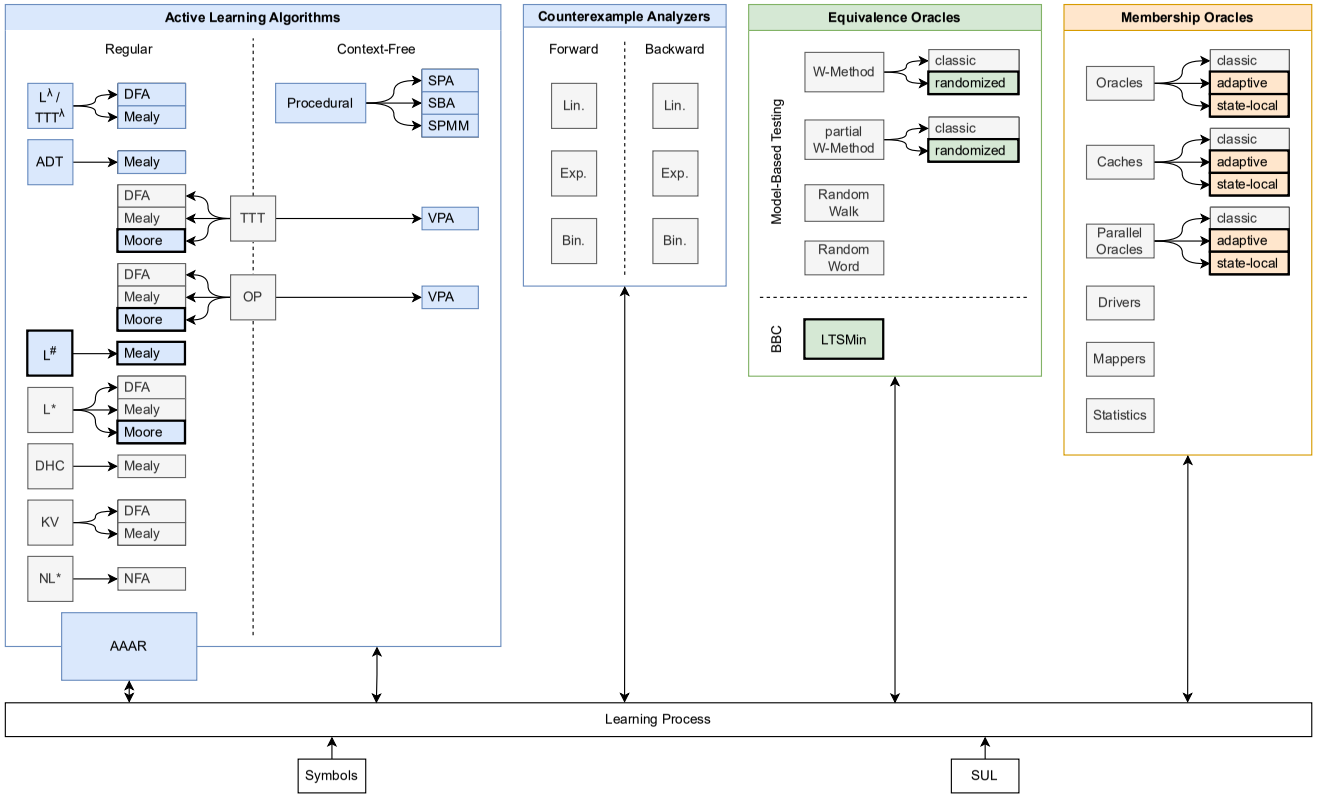

Fig. 1. An overview of the current active automata learning features of LearnLib. Gray nodes represent features from [45] and colored nodes represent new features. Blue marks learning-related concepts, green marks equivalence testing-related concepts, and orange marks membership query-related concepts. Components with bold outlines have been contributed by external community members.

Equivalence Oracles Meijer and Pol [57] integrated the model checker LTSmin [48] into LearnLib and therefore provided a means for *black-box checking* (BBC) [63]. Due to the clever use of LearnLib's modular structure, the authors implement the model checker as a special form of equivalence oracle which seamlessly integrates into the existing hierarchy.

Smetsers et al. [73] discovered that adding fuzzying to methods from model-based testing such as the (partial) W-method [21, 37] is able to boost the performance of counterexample search. The implementations of the authors can be used as drop-in replacements for the classic procedures.

Membership Oracles As an alternative to algorithmic properties for the reduction of membership queries during learning, Geske [38] investigated the concept of state-local alphabets which allow queries to be answered early if non-available input symbols are encountered. This concept is implemented as an independent query filter, making the approach compatible with various existing types of membership oracles.

Another practical means to boost performance concerns the caching and parallelization of queries. Vitorovic [80] introduced a new type of *adaptive membership queries* which model a symbol-by-symbol style communication, contrasting the typical preset queries. His work includes the respective caches and parallel oracles which are implemented in a generic fashion so that they can be shared by adaptive algorithms such as ADT and $L^{\#}$.

Performance Besides algorithmic properties, implementation details such as architectural design decisions or the chosen programming language may also impact (learning) performance. Typically, comparisons of tools (e.g., in [59]) are not conclusive and often depend on the considered use-cases. As a result, we focused more on advancing conceptual features. For example, we improved existing learner implementations by optimizing their batching of queries in order to support a higher degree of parallelization if applicable and we added support for suspending and resuming learning processes to allow for re-using intermediate results in more sophisticated learning setups.

3 Construction of Custom Learning Processes

In this section we illustrate how LearnLib supports the construction of learning processes tailored to specific requirements profiles. The example is included in the artifact and can be used as the basis for experimentation. Additionally, the entire learning process can also be easily implemented with the latest public release of LearnLib (version 0.18.0).

3.1 Requirements

Let us assume that we are interested in controlling a procedural system (systems that may comprise recursion) in a monitoring-based fashion to implement the concept of *life-long learning* [15]. This implies that

– we need a procedural learner and
– monitoring requires us to support
  - prefix-closed semantics as the monitor, ideally, records the subsequent reactions to individual inputs, and
  - extremely long counterexamples since the monitor may detect issues only after days worth of operation.

Immediately, this makes certain options preferable over others.

❶ For modelling prefix-closed procedural systems either SBAs or SPMMs can be used. Since our monitor records individual outputs, we choose SPMMs.
❷ Specific to procedural learners is the choice of a local learner for inferring the involved procedures. We choose the TTT algorithm for this task as it is specifically designed to handle the long counterexamples provided by monitoring.

Another important characteristics of the chosen example is that it comprises two phases:

– a phase for inferring a model for constructing the initial monitor, and
– the monitoring-based life-long learning phase.

In this paper, we focus on the first phase which is interesting because it comes with quite some potential for optimization: we want to speed up the learning process.

❸ Since we can easily spawn multiple local instances of our application, we want to answer queries in parallel.
❹ To reduce the load on our application, we also want to remove duplicate queries via the use of caching.
❺ Furthermore, we have specified a set of critical runs of our system that we want to check during learning to not deploy a knowingly faulty application.

In contrast, the monitoring phase is a purely technical issue. One only has to specify how the monitor can access the current hypothesis model and how monitored traces can be used to refine the hypothesis model.

3.2 Implementation

Listing 1.1 sketches how these requirements can be implemented with LearnLib.

Membership Oracle After the initial setup of symbolic interactions and instantiating our local testing instances, we begin with constructing the main membership oracle that is going to be used by the learner (for exploration) and the equivalence oracle (for verification). LearnLib's design of membership oracles innately supports the processing of query *batches*. Via convenient factory methods, LearnLib provides special *parallel oracles* that split these batches according to a selectable strategy and distribute the (sub-) batches to the provided oracle

```
// setup
var alphabet = ...
var instance1 = ...
var instance2 = ...

// membership oracles
var parallel = ParallelOracleBuilders.newStaticParallelOracle(instance1,
    instance2).create(); ❸
var mqo = MealyCaches.createCache(alphabet, parallel); ❹

// equivalence oracles
var sample = new SampleSetEQOracle<Input,
    Word<Output>>().addAll(requirements); ❺
var wMethod = new WMethodEQOracle<>(mqo);
var eqo = new EQOracleChain<>(sample, wMethod);

// learner
var learner = new SPMMLearner<>(alphabet, Output.ERROR, mqo,
    TTTAdapterMealy::new ❷ ); ❶

// learning loop
learner.startLearning();
var hyp = learner.getHypothesisModel();
DefaultQuery<Input, Word<Output>> cex;
while ((cex = eqo.findCounterExample(hyp, alphabet)) != null) {
    learner.refineHypothesis(cex);
    hyp = learner.getHypothesisModel();
}

// continue to work with the model
startMonitor(learner);
```

Listing 1.1. Construction of the discussed learning process. Encircled numbers reference the respective requirements of the example.

instances. Yet, due to clever use of abstraction, these oracles appear as regular membership oracles to the outside. As a result, we can use the same instance as a delegate for our query cache. Note that we simply re-use a Mealy cache for this purpose because both formalisms are transition output systems. Using a simple decorator pattern, users are able to construct powerful query chains which can easily integrate custom extensions such as query filters (e.g., to only focus on a specific part of the system) or counters (e.g., to measure the workload on the SUL).

Equivalence Oracle In a similar fashion, equivalence oracles can be composed. LearnLib provides out-of-the-box implementations for many of the conventional equivalence tests. The power comes from being able to combine them. In our example, we build an equivalence oracle chain which sequentially asks each of its

inner oracles for counterexamples. First, we use a sampling oracle that simply checks the traces from our previously prepared requirements. Once these traces no longer expose any in-equivalences, we use the W-method for a more methodical test-case generation. Due to the oracle chain, we can abstract all these steps (and potentially more) behind a simple object.

Learner Finally, the learner is set up. As discussed in our requirements, specific to procedural systems is the concept of a *meta-learner* which can be parameterized in its sub-learner(s). Here, we instantiate the SPMM learner and select the TTT algorithm as its procedural learner. The SPMM learner only requires a learner for (regular) Mealy machines, so many other algorithms in LearnLib (or custom ones) can be used, too.

It should be noted that learners can typically be configured with specific counterexample analysis strategies. In our case, however, we can omit this step for two reasons.

- Because SPMMs are context-free, the prefix-closed semantics guarantees that no analysis is required to identify the procedure call responsible for an error, because it is always the same as the procedure call where the error is observed.
- LearnLib uses convention over configuration when reasonable. For the procedural learner, we consider the default binary search-based analysis as adequate.

For other settings, like SPAs or SBAs, LearnLib provides 36 options alone for the (combined) choices of global and procedural counterexample analysis algorithms.

Learning Loop The actual learning process is a simple `while`-loop that alternates between the exploration and verification phase (cf. Section 2). The loop terminates once the equivalence oracle is no longer able to find counterexamples.

Monitoring The learner and the equivalence oracle only communicate via counterexamples. As a result, the very same learner can simply receive counterexamples from the monitor and continue its original learning process. In particular, we do not need to reset any progress but can seamlessly integrate the existing components in the new context. Due to space reasons, we skip the details of the monitor execution. However, from the learning loop, it should be clear how the monitor can access the current hypothesis model and how monitored traces can be used to refine the hypothesis model. The involved workflow is identical to the presented one.

3.3 Takeaways

With LearnLib it is possible to construct complex and intricate learning setups by composing them from simple and easy-to-understand components. The driving

forces behind this ability are the rigorous notion of abstraction and composition. For users, this manifests in two ways.

First, LearnLib provides the building blocks for highly specialized learning setups, all while maintaining compatibility between the involved structures. For example, had we chosen the acceptor-based SBA formalism instead of the transducer-based SPMM formalism, only the output type would have changed from `Word<Output>` to `Boolean`. For the rest, the same notion of composition for membership oracles, equivalence oracles, and (procedural) learners would have been available.

Second, LearnLib establishes functional contracts which allow for highly extensible implementations. In Listing 1.1, each concept ((procedural) learning algorithm, membership oracle, equivalence oracle, etc.) could have been replaced with a custom implementation without impeding the overall workflow.

From a practitioner's perspective this allows for easy building of use-case specific adaptations and optimizations to improve the overall experience when applying automata learning in practice. From a researcher's perspective, LearnLib provides a rich framework in which novel concepts can be implemented and directly evaluated in a multitude of different contexts.

4 Impact on the Community

Over the years, we continuously improved the experience for potential users. On the technical side, we enhanced the build process of LearnLib to include several industry standards (such as static code analysis and test coverage analysis) and transparently offer them in continuous integration and deployment pipelines to enable high-quality community contributions (cf. Section 2). By supporting modern technologies (such as the *Java Platform Module System* (JPMS) for building custom application installers) and improving integrability (e.g., by switching to the logging facade SLF4J and dropping dependencies with proprietary licenses), LearnLib should become much more attractive for professional environments as well.

On the social side, we further integrated with LearnLib's hosting platform GitHub. We replaced previous mailing lists with GitHub's discussion feature[5] which allows for a much more integrated communication about ideas, issues, and pull requests. Judging from the activity, this change seems to have been positively received by the community as well. Furthermore, we open-sourced the LearnLib website[6] to enable external contributors to improve and extend existing documentation, e.g., by referencing their own related projects to help connecting different research groups. Our efforts enable users to profit from LearnLib at different levels:

Tool-Level There are numerous cases where LearnLib is simply used as a tool [72, 68, 29, 31, 76, 10, 6, 5, 7, 12, 65, 22, 82, 39, 52, 66, 24, 18, 56, 55].

[5] `https://github.com/LearnLib/learnlib/discussions`
[6] `https://github.com/LearnLib/learnlib.github.io`

Here, LearnLib provides a service that is either used qualitatively (simply to infer a model) or quantitatively (to compare the performance of different learning algorithms). In these situations, the researchers typically only need to provide an interface to their systems under learning[7] in order to execute their experiments.

Library-Level Other work presents tools that have been built using LearnLib essentially as a library [13, 75, 79, 51, 81, 71, 54, 28, 30, 27, 77, 60, 69]. In this case, LearnLib is used as a library on which the respective tools depend. These situations show how the modularization, the APIs, and the deployment of LearnLib aligns with modern software development.

Framework-Level Cooperation at this level profits most from LearnLib's flexible architecture. Typical here is the extension of LearnLib with custom functionality [57, 23, 50, 17, 26] such as symbol filters or supporting conflicting queries. A particularly fruitful example of this is RALib [19] which is a standalone tool for active learning of register automata. RALib is based on LearnLib and heavily extends its different components with custom learning algorithms and equivalence algorithms (cf. Figure 1). Such extensions, in particular when they are re-integrated again into LearnLib (e.g., [57]), motivate our efforts to increase the flexibility of LearnLib's architecture.

5 Conclusion & Future Work

We have highlighted the advancements of LearnLib over the past decade, aiming to establish it as the central resource for (active) automata learning. Compositionality and extensibility have been the driving forces behind its development, as there cannot be a one-size-fits-all solution in automata learning. Instead, the ability to easily create custom learning processes tailored to specific application profiles is crucial for practical success. It allows researchers to easily benchmark their new learning algorithms against the state of the art by simply replacing components in established learning processes, while application developers can seamlessly compose tailored learning processes to evaluate their suitability for specific requirements. We have illustrated this flexibility by developing a learning process for the life-long learning of procedural systems.

We are continuously working on integrating new technologies. For example, LearnLib has already received initial support for *passive* automata learning via algorithms such as RPNI [61] or OSTIA [62] (contributed by Aleksander Mendoza-Drosik[8]). One of our next goals is to look at the insights gained by RALib [19] to enhance LearnLib's capability to handle data. In the long term, we aim to establish a LearnLib-based benchmark suite as an industry standard for evaluating and comparing active automata learning algorithms.

[7] Sometimes, the definition of this interface is the actual research being conducted.
[8] `https://github.com/aleksander-mendoza`

LearnLib is open-source. We encourage users to replicate and experiment with the concepts presented in this paper and are looking forward to future collaborations which we see as key to the long-term success of LearnLib.

Disclosure of Interests Markus Frohme is funded by Deutsche Forschungsgemeinschaft (DFG), Grant 528775176. The authors are directly involved with development and maintenance of the presented tool. The authors declare they have no financial interests.

References

[1] Fides Aarts. "Tomte : bridging the gap between active learning and real-world systems". PhD thesis. Radboud Universiteit Nijmegen, Netherlands, Oct. 2014. DOI: 2066/130428.

[2] Fides Aarts, Joeri de Ruiter, and Erik Poll. "Formal Models of Bank Cards for Free". In: *Sixth IEEE International Conference on Software Testing, Verification and Validation, ICST 2013 Workshops Proceedings, Luxembourg, Luxembourg, March 18-22, 2013*. IEEE Computer Society, 2013, pp. 461–468. DOI: 10.1109/ICSTW.2013.60.

[3] Fides Aarts, Julien Schmaltz, and Frits W. Vaandrager. "Inference and Abstraction of the Biometric Passport". In: *Leveraging Applications of Formal Methods, Verification, and Validation - 4th International Symposium on Leveraging Applications, ISoLA 2010, Heraklion, Crete, Greece, October 18-21, 2010, Proceedings, Part I*. Ed. by Tiziana Margaria and Bernhard Steffen. Vol. 6415. Lecture Notes in Computer Science. Springer, 2010, pp. 673–686. DOI: 10.1007/978-3-642-16558-0_54.

[4] Fides Aarts et al. "Generating models of infinite-state communication protocols using regular inference with abstraction". In: *Formal Methods Syst. Des.* 46.1 (2015), pp. 1–41. DOI: 10.1007/s10703-014-0216-x.

[5] Bernhard K. Aichernig, Christian Burghard, and Robert Korosec. "Learning-Based Testing of an Industrial Measurement Device". In: *NASA Formal Methods - 11th International Symposium, NFM 2019, Houston, TX, USA, May 7-9, 2019, Proceedings*. Ed. by Julia M. Badger and Kristin Yvonne Rozier. Vol. 11460. Lecture Notes in Computer Science. Springer, 2019, pp. 1–18. DOI: 10.1007/978-3-030-20652-9_1.

[6] Bernhard K. Aichernig and Martin Tappler. "Efficient Active Automata Learning via Mutation Testing". In: *J. Autom. Reason.* 63.4 (2019), pp. 1103–1134. DOI: 10.1007/S10817-018-9486-0.

[7] Bernhard K. Aichernig et al. "Learning a Behavior Model of Hybrid Systems Through Combining Model-Based Testing and Machine Learning". In: *Testing Software and Systems - 31st IFIP WG 6.1 International Conference, ICTSS 2019, Paris, France, October 15-17, 2019, Proceedings*. Ed. by Christophe Gaston, Nikolai Kosmatov, and Pascale Le Gall. Vol. 11812. Lecture Notes in Computer Science. Springer, 2019, pp. 3–21. DOI: 10.1007/978-3-030-31280-0_1.

[8] Rajeev Alur and P. Madhusudan. "Visibly pushdown languages". In: *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*. Ed. by László Babai. ACM, 2004, pp. 202–211. DOI: 10.1145/1007352.1007390.

[9] Dana Angluin. "Learning Regular Sets from Queries and Counterexamples". In: *Information and Computation* 75.2 (1987), pp. 87–106. DOI: 10.1016/0890-5401(87)90052-6.

[10] Paolo Arcaini, Angelo Gargantini, and Elvinia Riccobene. "Regular Expression Learning with Evolutionary Testing and Repair". In: *Testing Software and Systems - 31st IFIP WG 6.1 International Conference, ICTSS 2019, Paris, France, October 15-17, 2019, Proceedings*. Ed. by Christophe Gaston, Nikolai Kosmatov, and Pascale Le Gall. Vol. 11812. Lecture Notes in Computer Science. Springer, 2019, pp. 22–40. DOI: 10.1007/978-3-030-31280-0_2.

[11] Denis Arrivault et al. "Sp2Learn: A Toolbox for the Spectral Learning of Weighted Automata". In: *Proceedings of the 13th International Conference on Grammatical Inference, ICGI 2016, Delft, The Netherlands, October 5-7, 2016*. Ed. by Sicco Verwer, Menno van Zaanen, and Rick Smetsers. Vol. 57. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 105–119. URL: http://proceedings.mlr.press/v57/arrivault16.html.

[12] Kousar Aslam et al. "Interface protocol inference to aid understanding legacy software components". In: *Softw. Syst. Model.* 19.6 (2020), pp. 1519–1540. DOI: 10.1007/S10270-020-00809-2.

[13] Alexander Bainczyk et al. "ALEX: Mixed-Mode Learning of Web Applications at Ease". In: *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications - 7th International Symposium, ISoLA 2016, Imperial, Corfu, Greece, October 10-14, 2016, Proceedings, Part II*. Ed. by Tiziana Margaria and Bernhard Steffen. Vol. 9953. Lecture Notes in Computer Science. 2016, pp. 655–671. DOI: 10.1007/978-3-319-47169-3_51.

[14] Mohamad Bayram. "Moore-basierte Anfragen zur Query-Optimierung beim aktiven Automatenlernen". In german. Bachelor's thesis. TU Dortmund University, 2022.

[15] Antonia Bertolino et al. "Never-stop Learning: Continuous Validation of Learned Models for Evolving Systems through Monitoring". In: *ERCIM News* 2012.88 (2012). URL: http://ercim-news.ercim.eu/en88/special/never-stop-learning-continuous-validation-of-learned-models-for-evolving-systems-through-monitoring.

[16] Benedikt Bollig et al. "libalf: The Automata Learning Framework". In: *Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings*. Ed. by Tayssir Touili, Byron Cook, and Paul B. Jackson. Vol. 6174. Lecture Notes in Computer Science. Springer, 2010, pp. 360–364. DOI: 10.1007/978-3-642-14295-6_32.

[17] Véronique Bruyère, Guillermo A. Pérez, and Gaëtan Staquet. "Learning Realtime One-Counter Automata". In: *Tools and Algorithms for the Con-*

struction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I. Ed. by Dana Fisman and Grigore Rosu. Vol. 13243. Lecture Notes in Computer Science. Springer, 2022, pp. 244–262. DOI: 10.1007/978-3-030-99524-9_13.

[18] Véronique Bruyère, Guillermo A. Pérez, and Gaëtan Staquet. "Validating Streaming JSON Documents with Learned VPAs". In: *Tools and Algorithms for the Construction and Analysis of Systems - 29th International Conference, TACAS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Paris, France, April 22-27, 2023, Proceedings, Part I*. Ed. by Sriram Sankaranarayanan and Natasha Sharygina. Vol. 13993. Lecture Notes in Computer Science. Springer, 2023, pp. 271–289. DOI: 10.1007/978-3-031-30823-9_14.

[19] Sofia Cassel, Falk Howar, and Bengt Jonsson. "RALib: A LearnLib extension for inferring EFSMs". In: *DIFTS* 5 (2015).

[20] Chia Yuan Cho et al. "Inference and analysis of formal models of botnet command and control protocols". In: *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*. Ed. by Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov. ACM, 2010, pp. 426–439. DOI: 10.1145/1866307.1866355.

[21] Tsun S. Chow. "Testing Software Design Modeled by Finite-State Machines". In: *IEEE Transactions on Software Engineering* 4.3 (1978), pp. 178–187. DOI: 10.1109/TSE.1978.231496.

[22] Carlos Diego Nascimento Damasceno, Mohammad Reza Mousavi, and Adenilso da Silva Simão. "Learning by sampling: learning behavioral family models from software product lines". In: *Empir. Softw. Eng.* 26.1 (2021), p. 4. DOI: 10.1007/S10664-020-09912-W.

[23] Carlos Diego Nascimento Damasceno, Mohammad Reza Mousavi, and Adenilso da Silva Simão. "Learning to Reuse: Adaptive Model Learning for Evolving Systems". In: *Integrated Formal Methods - 15th International Conference, IFM 2019, Bergen, Norway, December 2-6, 2019, Proceedings*. Ed. by Wolfgang Ahrendt and Silvia Lizeth Tapia Tarifa. Vol. 11918. Lecture Notes in Computer Science. Springer, 2019, pp. 138–156. DOI: 10.1007/978-3-030-34968-4_8.

[24] Simon Dierl et al. "Learning Symbolic Timed Models from Concrete Timed Data". In: *NASA Formal Methods - 15th International Symposium, NFM 2023, Houston, TX, USA, May 16-18, 2023, Proceedings*. Ed. by Kristin Yvonne Rozier and Swarat Chaudhuri. Vol. 13903. Lecture Notes in Computer Science. Springer, 2023, pp. 104–121. DOI: 10.1007/978-3-031-33170-1_7.

[25] Samuel Drews and Loris D'Antoni. "Learning Symbolic Automata". In: *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala,*

Sweden, April 22-29, 2017, Proceedings, Part I. Ed. by Axel Legay and Tiziana Margaria. Vol. 10205. Lecture Notes in Computer Science. 2017, pp. 173–189. DOI: 10.1007/978-3-662-54577-5_10.

[26] Tiago Ferreira, Gerco van Heerdt, and Alexandra Silva. "Tree-Based Adaptive Model Learning". In: *A Journey from Process Algebra via Timed Automata to Model Learning - Essays Dedicated to Frits Vaandrager on the Occasion of His 60th Birthday*. Ed. by Nils Jansen, Mariëlle Stoelinga, and Petra van den Bos. Vol. 13560. Lecture Notes in Computer Science. Springer, 2022, pp. 164–179. DOI: 10.1007/978-3-031-15629-8_10.

[27] Tiago Ferreira et al. "Conflict-Aware Active Automata Learning". In: *Proceedings of the Fourteenth International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2023, Udine, Italy, 18-20th September 2023*. Ed. by Antonis Achilleos and Dario Della Monica. Vol. 390. EPTCS. 2023, pp. 150–167. DOI: 10.4204/EPTCS.390.10.

[28] Tiago Ferreira et al. "Prognosis: closed-box analysis of network protocol implementations". In: *ACM SIGCOMM 2021 Conference, Virtual Event, USA, August 23-27, 2021*. Ed. by Fernando A. Kuipers and Matthew C. Caesar. ACM, 2021, pp. 762–774. DOI: 10.1145/3452296.3472938.

[29] Paul Fiterau-Brostean, Ramon Janssen, and Frits W. Vaandrager. "Combining Model Learning and Model Checking to Analyze TCP Implementations". In: *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II*. Ed. by Swarat Chaudhuri and Azadeh Farzan. Vol. 9780. Lecture Notes in Computer Science. Springer, 2016, pp. 454–471. DOI: 10.1007/978-3-319-41540-6_25.

[30] Paul Fiterau-Brostean et al. "DTLS-Fuzzer: A DTLS Protocol State Fuzzer". In: *15th IEEE Conference on Software Testing, Verification and Validation, ICST 2022, Valencia, Spain, April 4-14, 2022*. IEEE, 2022, pp. 456–458. DOI: 10.1109/ICST53961.2022.00051.

[31] Paul Fiterau-Brostean et al. "Model learning and model checking of SSH implementations". In: *Proceedings of the 24th ACM SIGSOFT International SPIN Symposium on Model Checking of Software, Santa Barbara, CA, USA, July 10-14, 2017*. Ed. by Hakan Erdogmus and Klaus Havelund. ACM, 2017, pp. 142–151. DOI: 10.1145/3092282.3092289.

[32] Paul Fiterău-Broştean and Falk Howar. "Learning-Based Testing the Sliding Window Behavior of TCP Implementations". In: *Critical Systems: Formal Methods and Automated Verification - Joint 22nd International Workshop on Formal Methods for Industrial Critical Systems - and - 17th International Workshop on Automated Verification of Critical Systems, FMICS-AVoCS 2017, Turin, Italy, September 18-20, 2017, Proceedings*. Ed. by Laure Petrucci, Cristina Seceleanu, and Ana Cavalcanti. Vol. 10471. Lecture Notes in Computer Science. Springer, 2017, pp. 185–200. DOI: 10.1007/978-3-319-67113-0_12.

[33] Markus Frohme. "Active Automata Learning with Adaptive Distinguishing Sequences". In: *CoRR* abs/1902.01139 (2019). arXiv: 1902.01139.

[34] Markus Frohme. "Model-based quality assurance of intrumented context-free systems". PhD thesis. Technical University of Dortmund, Germany, 2023. DOI: 10.17877/DE290R-24032.

[35] Markus Frohme and Bernhard Steffen. "Compositional learning of mutually recursive procedural systems". In: *Int. J. Softw. Tools Technol. Transf.* 23.4 (2021), pp. 521–543. DOI: 10.1007/S10009-021-00634-Y.

[36] Markus Frohme and Bernhard Steffen. "From Languages to Behaviors and Back". In: *A Journey from Process Algebra via Timed Automata to Model Learning - Essays Dedicated to Frits Vaandrager on the Occasion of His 60th Birthday*. Ed. by Nils Jansen, Mariëlle Stoelinga, and Petra van den Bos. Vol. 13560. Lecture Notes in Computer Science. Springer, 2022, pp. 180–200. DOI: 10.1007/978-3-031-15629-8_11.

[37] Susumu Fujiwara et al. "Test Selection Based on Finite State Models". In: *IEEE Transactions on Software Engineering* 17.6 (1991), pp. 591–603. DOI: 10.1109/32.87284.

[38] Maren Geske. "Implementation and performance evaluation of an active learning algorithm for visible state-local alphabets". Master's thesis. TU Dortmund University, 2018.

[39] Dennis Hendriks and Kousar Aslam. "A Systematic Approach for Interfacing Component-Based Software with an Active Automata Learning Tool". In: *Leveraging Applications of Formal Methods, Verification and Validation. Software Engineering - 11th International Symposium, ISoLA 2022, Rhodes, Greece, October 22-30, 2022, Proceedings, Part II*. Ed. by Tiziana Margaria and Bernhard Steffen. Vol. 13702. Lecture Notes in Computer Science. Springer, 2022, pp. 216–236. DOI: 10.1007/978-3-031-19756-7_13.

[40] Falk Howar. "Active Learning of Interface Programs". PhD thesis. TU Dortmund University, 2012. DOI: 10.17877/DE290R-4817.

[41] Falk Howar and Bernhard Steffen. "Active Automata Learning as Black-Box Search and Lazy Partition Refinement". In: *A Journey from Process Algebra via Timed Automata to Model Learning - Essays Dedicated to Frits Vaandrager on the Occasion of His 60th Birthday*. Ed. by Nils Jansen, Mariëlle Stoelinga, and Petra van den Bos. Vol. 13560. Lecture Notes in Computer Science. Springer, 2022, pp. 321–338. DOI: 10.1007/978-3-031-15629-8_17.

[42] Falk Howar, Bernhard Steffen, and Maik Merten. "Automata Learning with Automated Alphabet Abstraction Refinement". In: *Verification, Model Checking, and Abstract Interpretation - 12th International Conference, VMCAI 2011, Austin, TX, USA, January 23-25, 2011. Proceedings*. Ed. by Ranjit Jhala and David A. Schmidt. Vol. 6538. Lecture Notes in Computer Science. Springer, 2011, pp. 263–277. DOI: 10.1007/978-3-642-18275-4_19.

[43] Falk Howar et al. "The Teachers' Crowd: The Impact of Distributed Oracles on Active Automata Learning". In: *Leveraging Applications of Formal Methods, Verification, and Validation - International Workshops, SARS 2011 and MLSC 2011, Held Under the Auspices of ISoLA 2011 in Vienna,*

Austria, October 17-18, 2011. Revised Selected Papers. Ed. by Reiner Hähnle et al. Vol. 336. Communications in Computer and Information Science. Springer, 2011, pp. 232–247. DOI: 10.1007/978-3-642-34781-8_18.

[44] Malte Isberner. "Foundations of active automata learning: an algorithmic perspective". PhD thesis. Technical University Dortmund, Germany, 2015. URL: https://hdl.handle.net/2003/34282.

[45] Malte Isberner, Falk Howar, and Bernhard Steffen. "The Open-Source LearnLib - A Framework for Active Automata Learning". In: *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I*. Ed. by Daniel Kroening and Corina S. Pasareanu. Vol. 9206. Lecture Notes in Computer Science. Springer, 2015, pp. 487–495. DOI: 10.1007/978-3-319-21690-4_32.

[46] Malte Isberner, Falk Howar, and Bernhard Steffen. "The TTT Algorithm: A Redundancy-Free Approach to Active Automata Learning". In: *Runtime Verification - 5th International Conference, RV 2014, Toronto, ON, Canada, September 22-25, 2014. Proceedings*. Ed. by Borzoo Bonakdarpour and Scott A. Smolka. Vol. 8734. Lecture Notes in Computer Science. Springer, 2014, pp. 307–322. DOI: 10.1007/978-3-319-11164-3_26.

[47] Valérie Issarny et al. "CONNECT Challenges: Towards Emergent Connectors for Eternal Networked Systems". In: *14th IEEE International Conference on Engineering of Complex Computer Systems, ICECCS 2009, Potsdam, Germany, 2-4 June 2009*. IEEE Computer Society, 2009, pp. 154–161. DOI: 10.1109/ICECCS.2009.44.

[48] Gijs Kant et al. "LTSmin: High-Performance Language-Independent Model Checking". In: *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*. Ed. by Christel Baier and Cesare Tinelli. Vol. 9035. Lecture Notes in Computer Science. Springer, 2015, pp. 692–707. DOI: 10.1007/978-3-662-46681-0_61.

[49] Ali Khalili and Armando Tacchella. "Learning Nondeterministic Mealy Machines". In: *Proceedings of the 12th International Conference on Grammatical Inference, ICGI 2014, Kyoto, Japan, September 17-19, 2014*. Ed. by Alexander Clark, Makoto Kanazawa, and Ryo Yoshinaka. Vol. 34. JMLR Workshop and Conference Proceedings. JMLR.org, 2014, pp. 109–123. URL: http://proceedings.mlr.press/v34/khalili14a.html.

[50] Paul Kogel, Verena Klös, and Sabine Glesner. "TTT/ik: Learning Accurate Mealy Automata Efficiently with an Imprecise Symbol Filter". In: *Formal Methods and Software Engineering - 23rd International Conference on Formal Engineering Methods, ICFEM 2022, Madrid, Spain, October 24-27, 2022, Proceedings*. Ed. by Adrián Riesco and Min Zhang. Vol. 13478. Lecture Notes in Computer Science. Springer, 2022, pp. 227–243. DOI: 10.1007/978-3-031-17244-1_14.

[51] Martin Kölbl, Stefan Leue, and Thomas Wies. "TarTar: A Timed Automata Repair Tool". In: *Computer Aided Verification - 32nd International*

Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part I. Ed. by Shuvendu K. Lahiri and Chao Wang. Vol. 12224. Lecture Notes in Computer Science. Springer, 2020, pp. 529–540. DOI: 10.1007/978-3-030-53288-8_25.

[52] Eric Lesiuta, Victor Bandur, and Mark Lawford. "SLIME: State Learning in the Middle of Everything for Tool-Assisted Vulnerability Detection". In: *Computer Security. ESORICS 2022 International Workshops - CyberICPS 2022, SECPRE 2022, SPOSE 2022, CPS4CIP 2022, CDT&SECOMANE 2022, EIS 2022, and SecAssure 2022, Copenhagen, Denmark, September 26-30, 2022, Revised Selected Papers*. Ed. by Sokratis K. Katsikas et al. Vol. 13785. Lecture Notes in Computer Science. Springer, 2022, pp. 686–704. DOI: 10.1007/978-3-031-25460-4_39.

[53] Yong Li et al. "A Novel Learning Algorithm for Büchi Automata Based on Family of DFAs and Classification Trees". In: *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part I*. Ed. by Axel Legay and Tiziana Margaria. Vol. 10205. Lecture Notes in Computer Science. 2017, pp. 208–226. DOI: 10.1007/978-3-662-54577-5_12.

[54] Tiziana Margaria and Alexander Schieweck. "Towards Engineering Digital Twins by Active Behaviour Mining". In: *Model Checking, Synthesis, and Learning - Essays Dedicated to Bengt Jonsson on The Occasion of His 60th Birthday*. Ed. by Ernst-Rüdiger Olderog, Bernhard Steffen, and Wang Yi. Vol. 13030. Lecture Notes in Computer Science. Springer, 2021, pp. 138–163. DOI: 10.1007/978-3-030-91384-7_8.

[55] Stefan Marksteiner, Peter Priller, and Markus Wolf. "Approaches for Automating Cybersecurity Testing of Connected Vehicles". In: *Intelligent Secure Trustable Things*. Ed. by Michael Karner et al. Cham: Springer Nature Switzerland, 2024, pp. 219–234. DOI: 10.1007/978-3-031-54049-3_13.

[56] Stefan Marksteiner, Marjan Sirjani, and Mikael Sjödin. "Using Automata Learning for Compliance Evaluation of Communication Protocols on an NFC Handshake Example". In: *Engineering of Computer-Based Systems - 8th International Conference,ECBS 2023, Västerås, Sweden, October 16-18, 2023, Proceedings*. Ed. by Jan Kofron, Tiziana Margaria, and Cristina Seceleanu. Vol. 14390. Lecture Notes in Computer Science. Springer, 2023, pp. 170–190. DOI: 10.1007/978-3-031-49252-5_13.

[57] Jeroen Meijer and Jaco van de Pol. "Sound Black-Box Checking in the LearnLib". In: *NASA Formal Methods - 10th International Symposium, NFM 2018, Newport News, VA, USA, April 17-19, 2018, Proceedings*. Ed. by Aaron Dutle, César A. Muñoz, and Anthony Narkawicz. Vol. 10811. Lecture Notes in Computer Science. Springer, 2018, pp. 349–366. DOI: 10.1007/978-3-319-77935-5_24.

[58] Edward F. Moore. "Gedanken-Experiments on Sequential Machines". In: *Automata Studies*. Ed. by C. E. Shannon and J. McCarthy. Princeton: Prince-

ton University Press, 1956, pp. 129–154. DOI: doi:10.1515/9781400882618-006.

[59] Edi Muskardin et al. "AALpy: An Active Automata Learning Library". In: *Automated Technology for Verification and Analysis - 19th International Symposium, ATVA 2021, Gold Coast, QLD, Australia, October 18-22, 2021, Proceedings*. Ed. by Zhe Hou and Vijay Ganesh. Vol. 12971. Lecture Notes in Computer Science. Springer, 2021, pp. 67–73. DOI: 10.1007/978-3-030-88885-5_5.

[60] Thomas Neele and Matteo Sammartino. "Compositional Automata Learning of Synchronous Systems". In: *Fundamental Approaches to Software Engineering - 26th International Conference, FASE 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2023, Paris, France, April 22-27, 2023, Proceedings*. Ed. by Leen Lambers and Sebastián Uchitel. Vol. 13991. Lecture Notes in Computer Science. Springer, 2023, pp. 47–66. DOI: 10.1007/978-3-031-30826-0_3.

[61] José Oncina and Pedro García. "Inferring regular languages in polynomial update time". In: *World Scientific* (Jan. 1992), pp. 49–61. DOI: 10.1142/9789812797902_0004.

[62] José Oncina, Pedro García, and Enrique Vidal. "Learning Subsequential Transducers for Pattern Recognition Interpretation Tasks". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 15.5 (1993), pp. 448–458. DOI: 10.1109/34.211465.

[63] Doron Peled, Moshe Y. Vardi, and Mihalis Yannakakis. "Black Box Checking". In: *Formal Methods for Protocol Engineering and Distributed Systems: FORTE XII / PSTV XIX. IFIP Advances in Information and Communication Technology*. Ed. by Jianping Wu, Samuel T. Chanson, and Qiang Gao. Boston, MA: Springer US, 1999, pp. 225–240. DOI: 10.1007/978-0-387-35578-8_13.

[64] Andrea Pferscher and Bernhard K. Aichernig. "Fingerprinting Bluetooth Low Energy Devices via Active Automata Learning". In: *Formal Methods - 24th International Symposium, FM 2021, Virtual Event, November 20-26, 2021, Proceedings*. Ed. by Marieke Huisman, Corina S. Pasareanu, and Naijun Zhan. Vol. 13047. Lecture Notes in Computer Science. Springer, 2021, pp. 524–542. DOI: 10.1007/978-3-030-90870-6_28.

[65] Andrea Pferscher and Bernhard K. Aichernig. "Learning Abstracted Nondeterministic Finite State Machines". In: *Testing Software and Systems - 32nd IFIP WG 6.1 International Conference, ICTSS 2020, Naples, Italy, December 9-11, 2020, Proceedings*. Ed. by Valentina Casola, Alessandra De Benedictis, and Massimiliano Rak. Vol. 12543. Lecture Notes in Computer Science. Springer, 2020, pp. 52–69. DOI: 10.1007/978-3-030-64881-7_4.

[66] Swantje Plambeck, Lutz Schammer, and Görschwin Fey. "On the Viability of Decision Trees for Learning Models of Systems". In: *27th Asia and South Pacific Design Automation Conference, ASP-DAC 2022, Taipei, Taiwan, January 17-20, 2022*. IEEE, 2022, pp. 696–701. DOI: 10.1109/ASP-DAC52403.2022.9712579.

[67] Harald Raffelt, Bernhard Steffen, and Tiziana Margaria. "Dynamic Testing Via Automata Learning". In: *Hardware and Software: Verification and Testing, Third International Haifa Verification Conference, HVC 2007, Haifa, Israel, October 23-25, 2007, Proceedings*. Ed. by Karen Yorav. Vol. 4899. Lecture Notes in Computer Science. Springer, 2007, pp. 136–152. DOI: 10.1007/978-3-540-77966-7_13.

[68] Joeri de Ruiter and Erik Poll. "Protocol State Fuzzing of TLS Implementations". In: *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*. Ed. by Jaeyeon Jung and Thorsten Holz. USENIX Association, 2015, pp. 193–206. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/de-ruiter.

[69] Ocan Sankur. "Timed Automata Verification and Synthesis via Finite Automata Learning". In: *Tools and Algorithms for the Construction and Analysis of Systems - 29th International Conference, TACAS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Paris, France, April 22-27, 2023, Proceedings, Part II*. Ed. by Sriram Sankaranarayanan and Natasha Sharygina. Vol. 13994. Lecture Notes in Computer Science. Springer, 2023, pp. 329–349. DOI: 10.1007/978-3-031-30820-8_21.

[70] Muhammad Muzammil Shahbaz. "Reverse Engineering Enhanced State Models of Black Box Software Components to support Integration Testing". PhD thesis. Institut Polytechnique de Grenoble, France, Dec. 2008.

[71] Junya Shijubo, Masaki Waga, and Kohei Suenaga. "Efficient Black-Box Checking via Model Checking with Strengthened Specifications". In: *Runtime Verification - 21st International Conference, RV 2021, Virtual Event, October 11-14, 2021, Proceedings*. Ed. by Lu Feng and Dana Fisman. Vol. 12974. Lecture Notes in Computer Science. Springer, 2021, pp. 100–120. DOI: 10.1007/978-3-030-88494-9_6.

[72] Wouter Smeenk et al. "Applying Automata Learning to Embedded Control Software". In: *Formal Methods and Software Engineering - 17th International Conference on Formal Engineering Methods, ICFEM 2015, Paris, France, November 3-5, 2015, Proceedings*. Ed. by Michael J. Butler, Sylvain Conchon, and Fatiha Zaïdi. Vol. 9407. Lecture Notes in Computer Science. Springer, 2015, pp. 67–83. DOI: 10.1007/978-3-319-25423-4_5.

[73] Rick Smetsers et al. "Complementing Model Learning with Mutation-Based Fuzzing". In: *CoRR* abs/1611.02429 (2016). arXiv: 1611.02429.

[74] Bernhard Steffen, Falk Howar, and Maik Merten. "Introduction to Active Automata Learning from a Practical Perspective". In: *Formal Methods for Eternal Networked Software Systems - 11th International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM 2011, Bertinoro, Italy, June 13-18, 2011. Advanced Lectures*. Ed. by Marco Bernardo and Valérie Issarny. Vol. 6659. Lecture Notes in Computer Science. Springer, 2011, pp. 256–296. DOI: 10.1007/978-3-642-21455-4_8.

[75] Chris McMahon Stone, Tom Chothia, and Joeri de Ruiter. "Extending Automated Protocol State Learning for the 802.11 4-Way Handshake". In: *Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I*. Ed. by Javier López, Jianying Zhou, and Miguel Soriano. Vol. 11098. Lecture Notes in Computer Science. Springer, 2018, pp. 325–345. DOI: 10.1007/978-3-319-99073-6_16.

[76] Martin Tappler, Bernhard K. Aichernig, and Roderick Bloem. "Model-Based Testing IoT Communication via Active Automata Learning". In: *2017 IEEE International Conference on Software Testing, Verification and Validation, ICST 2017, Tokyo, Japan, March 13-17, 2017*. IEEE Computer Society, 2017, pp. 276–287. DOI: 10.1109/ICST.2017.32.

[77] Frits W. Vaandrager, Masoud Ebrahimi, and Roderick Bloem. "Learning Mealy machines with one timer". In: *Inf. Comput.* 295.Part B (2023), p. 105013. DOI: 10.1016/J.IC.2023.105013.

[78] Frits W. Vaandrager et al. "A New Approach for Active Automata Learning Based on Apartness". In: *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I*. Ed. by Dana Fisman and Grigore Rosu. Vol. 13243. Lecture Notes in Computer Science. Springer, 2022, pp. 223–243. DOI: 10.1007/978-3-030-99524-9_12.

[79] Pepe Vila et al. "CacheQuery: learning replacement policies from hardware caches". In: *Proceedings of the 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2020, London, UK, June 15-20, 2020*. Ed. by Alastair F. Donaldson and Emina Torlak. ACM, 2020, pp. 519–532. DOI: 10.1145/3385412.3386008.

[80] Leon Vitorovic. "Query-Parallelisierung des ADT Learners in der LearnLib". In german. Bachelor's thesis. TU Dortmund University, 2024.

[81] Masaki Waga. "Falsification of cyber-physical systems with robustness-guided black-box checking". In: *HSCC '20: 23rd ACM International Conference on Hybrid Systems: Computation and Control, Sydney, New South Wales, Australia, April 21-24, 2020*. Ed. by Aaron D. Ames, Sanjit A. Seshia, and Jyotirmoy Deshmukh. ACM, 2020, 11:1–11:13. DOI: 10.1145/3365365.3382193.

[82] Masaki Waga et al. "Dynamic Shielding for Reinforcement Learning in Black-Box Environments". In: *Automated Technology for Verification and Analysis - 20th International Symposium, ATVA 2022, Virtual Event, October 25-28, 2022, Proceedings*. Ed. by Ahmed Bouajjani, Lukás Holík, and Zhilin Wu. Vol. 13505. Lecture Notes in Computer Science. Springer, 2022, pp. 25–41. DOI: 10.1007/978-3-031-19992-9_2.