An Experimental Evaluation of Conformance Testing Techniques in Active Automata Learning

Bharat Garhewal
Radboud University
Nijmegen, The Netherlands
b.garhewal@ru.nl

Carlos Diego N. Damasceno
Radboud University
Nijmegen, The Netherlands
d.damasceno@cs.ru.nl

***Abstract*—Active automata learning is a technique for dynamically learning finite state machine models of black-box systems. Conformance testing is a well-known bottleneck during learning. While multiple conformance testing techniques (CTTs) for Finite State Machines have been proposed, there is a lack of empirical studies that assess the effects of these CTTs *during* learning. In this work, we compare the performance of eight different CTTs (W, Wp, HSI, H-ADS, H, SPY, SPY-H, I-ADS) while learning 46 models from different communication protocols. Moreover, we propose $\mathtt{APFD}_L$ as a metric for characterizing the efficiency of automata learning experiments in terms of fault detection capacity. This metric allows identifying CTTs with a lower total cost regarding the number of symbols and resets and a higher rate of state discovery during learning. Our results indicate that while the total cost entailed by CTTs in learning tends to be negligible, we found a significant difference in fault detection rate in learning. Nevertheless, the differences in fault detection rates become negligible when CTTs are applied in randomized mode. These findings reveal the positive role that randomness can have in improving learning efficiency, despite compromising test completeness.**

I. Introduction

Active automata learning is frequently used to learn behavioral models of software systems. First introduced by Angluin [1, 2] in 1987, it has since been improved [3, 4] and extended in a variety of ways. It has been used to learn models of many real-world systems [5, 6, 7, 8] (we refer to [9, 10] for surveys). While automata learning is applicable in multiple settings, in this work, we concentrate on *finite state machines* (FSMs) (aka Mealy machines.) Active learning is a dynamic technique, thus it involves interacting with the *system under learning* (SUL) via *output queries* (OQs): sending inputs to the SUL and observing the resulting outputs. By using OQs, learning algorithms construct a hypothesis model of the SUL. Learning also involves the process of conformance testing, whereby we test whether the SUL conforms to a hypothesized model. Conformance testing is also implemented using OQs. However, conformance testing is known to be a bottleneck in the learning process [11, 12, 13], dwarfing the number of inputs sent to the SUL during hypothesis construction.

Conformance testing — the problem of determining if an implementation conforms to a specification — works by constructing a test-suite given (1) a specification and (2) an estimated upper-bound m on the number of states in the SUL. If the specification and the SUL have the same response to the test-suite and the bound m is correct, we conclude that the SUL conforms to the specification, and we say that the test-suite *passes*. While it is known that conformance testing is exponential [14], there have been many *conformance testing techniques* (CTTs) proposed in order to improve the efficiency of testing in practice (i.e., reduce the size of the test-suite necessary to conclude conformance) [15]. Most of these CTTs have the same worst-case complexity, they claim to perform better under specific circumstances, such as the existence of *adaptive distinguishing sequences* [16]. However, in automata learning, to learn an SUL of n states, we may require $n-1$ calls to a conformance tester. Thus, it is important that the SUL fails the test-suite generated by our CTT *as quickly as possible* [17], (called "finding counterexamples faster"). In this work, we wish to experimentally check the performance of different CTTs specifically in the context of automata learning. In this context, we are only concerned with how quickly an SUL fails the test-suite, and not with the size of the test-suite. Testing using randomized test-suites has also been shown to fail quicker [18]. Additionally, while some modern CTTs such as the SPY-(H)-methods [19, 20] have better overall fault detection performance, it is unknown if this is the case in the context of automata learning.

In this work, experimentally evaluate whether CTTs which generate smaller test-suites improve efficiency of active automata learning. We answer do this via an experimental survey of communication protocol models. We run learning experiments for 46 FSMs from the AutomataWiki [21][1] with $L^{\#}$ [22] as the default learning algorithm. We compare eight CTTs (W, Wp, HSI, Hybrid-ADS, I-ADS, SPY, SPY-H, and H) in randomized and non-randomized styles. The SPY, SPY-H, and H methods cannot be used in a randomized fashion, so we skip randomized test-suites for them.

Our results indicate that for these (smaller) communication protocols, the choice of non-randomized CTTs has a minor effect on total cost of learning and average fault detection capacity in favor of I-ADS, with no significant difference between I-ADS and H-ADS or HSI. On the other hand, using randomized CTTs results in both lower cost of learning and

Research supported by NWO TOP project 612.001.852 "Grey-box learning of Interfaces for Refactoring Legacy Software (GIRLS)".

[1]Available at: https://automata.cs.ru.nl.

higher average fault detection capacity for *all* CTTs, with negligible differences between different CTTs. Overall, there is a significant advantage to randomized CTTs when compared to non-randomized CTTs for total cost of learning and fault detection capacity. However, this must be weighed against the fact that randomized CTTs do not offer any kind of completeness guarantee.

Specifically, our contributions are as follows:

1) We introduce $\texttt{APFD}_L$, a new metric to analyze the efficiency of learning by taking into account fault detection capacity and total cost during a learning experiment.
2) We empirically evaluate the efficiency of eight CTTs in the context of active automata learning.
3) We show that lower total cost of learning does not imply a higher fault detection capacity.
4) We show that the efficiency of CTTs can become negligibly different in randomized mode.

The rest of the paper is structured as follows: Section II presents requisite background information and related work, Section III contains the methodology for our experiments, Sections IV and V present and discuss our experimental setup and the results, and Section VI concludes.

II. Background and Related Work

In this section, we first recall the definition of a finite state machine, introduce *active (automata) learning* and some learning algorithms. Next, we describe conformance testing of FSMs in active learning, and briefly discuss different algorithms for conformance testing.

Definition 1 (Finite State Machine): An FSM $\mathcal{M}$ is a tuple $(Q, q_0, I, O, \delta, \lambda)$ where

- Q is a finite set of states, with $q_0 \in Q$ being the *initial* state,
- I and O are finite sets of input and output symbols respectively,
- δ is the total transition function $Q \times I \rightarrow Q$, and
- λ is the total output function $Q \times I \rightarrow O$.

The concatenation of two sequences σ_1, σ_2 is indicated by $\sigma_1 \cdot \sigma_2$. We extend the transition function to an input sequence $i \cdot \sigma$ as follows $\delta^*(q, i \cdot \sigma) = \delta^*(\delta(q, i), \sigma)$. The output function is extended to an input sequence $i \cdot \sigma$ as follows $\lambda^*(q, i \cdot \sigma) = \lambda(q, i) \cdot \lambda^*(\delta(q, i), \sigma)$. For the empty input sequence $\sigma = \epsilon$, we define $\delta^*(q, \sigma) = q$ and $\lambda^*(q, \sigma) = \epsilon$. We write $\lambda(\mathcal{M}, \sigma)$ to indicate $\lambda^*(q_0, \sigma)$. Observe that this definition produces a deterministic FSM. In this work, we only consider deterministic FSMs.

Figure 1 provides a visualization of a modified version of the *Minimally Adequate Teacher* (MAT) framework introduced in [1]. The MAT framework assumes the presence of a *learner* and a *teacher*, where the teacher has complete knowledge of a hidden FSM $\mathcal{M}$, called the System Under Learning (SUL) [2]. The goal of the learner is to learn $\mathcal{M}$ by posing queries to the teacher. These queries may take the form of *output queries* (OQs), which are input sequences σ, and the teacher responds with an output sequence $\lambda^*(\mathcal{M}, \sigma)$ for the query. Each output query has an implicit *reset* symbol at the end of the input sequence, which brings the FSM back to its initial state. By asking OQs, the learner constructs a *hypothesis* $\mathcal{H}$, and poses an *equivalence query* (EQ), to check if the hypothesis is language-equivalent to the SUL: that is, for all input sequences $\sigma \in I^* : \lambda^*(\mathcal{H}, \sigma) = \lambda^*(\mathcal{M}, \sigma)$. If $\mathcal{H}$ and $\mathcal{M}$ are equivalent, the teacher replies "YES" and learning terminates. If not, the teacher replies with a *counterexample*, and learning resumes. Thus, a single active learning experiment may consist of multiple *rounds* of learning and equivalence-checking phases. In practice, the teacher does not know the hidden FSM $\mathcal{M}$ and answers EQs using *conformance testing*.

[2]Also called "Implementation Under Test" or "System Under Test" in testing literature.

A. Learning Algorithms

There are multiple learning algorithms available for learning FSMs: L^*_M [23], Rivest-Schapire [4], TTT [3], DHC [24], and $L^\#$ [22]. Most of the best-performing algorithms (Rivest-Schapire, TTT, $L^\#$) have the same complexity $\mathsf{O}(|I| \cdot |Q|^2 + |Q| \log(m))$, where m is the length of the longest counterexample. In this work, we have chosen to use $L^\#$, which uses a simple observation tree as its primary data structure and prevents duplicate OQs.

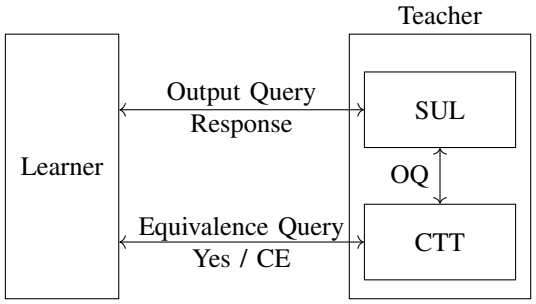

Fig. 1. Variant of the *Minimally Adequate Teacher* framework for $L^\#$.

B. Conformance Testing

Informally, conformance testing is the problem of testing whether an SUL adheres to a specification through OQs. Given a hypothesis $\mathcal{H}$ and under the assumption that the FSM $\mathcal{M}$ representing the SUL has at most m states, a *conformance testing technique* (CTT) can generate an m-complete test-suite:

Definition 2: A test-suite π is m-complete for an FSM $\mathcal{M}$ if for all in-equivalent FSMs $\mathcal{M}'$ with at most m states, there exists a $\sigma \in \pi$ such that $\lambda^*(\mathcal{M}, \sigma) \neq \lambda^*(\mathcal{M}', \sigma)$.

If the outputs of $\mathcal{H}$ and $\mathcal{M}$ agree for all tests, then $\mathcal{M}$ passes the test-suite. In practise, the size of $\mathcal{M}$ is unknown, and we set $m = n + k$, where n is the number of states in $\mathcal{H}$ and $k \in \mathbb{N}$ is the number of 'extra' states. The number of tests in a test-suite is proven to be exponential in the size of k [14].

We now describe the components of a test-suite. Each test-suite typically requires three components:

1) a set of *access sequences* A: an access sequence for a state s is a sequence σ such that $\delta^*(q_0, \sigma) = s$. Thus,

the set A contains an access sequence for each state in the FSM. The empty sequence "ϵ" is used as an access sequence for the initial state;

2) a set of *infix sequences*: $I^{\leq k+1}$; and
3) a *characterization set* W: A characterization set is a set of input sequences such that for any pair of distinct states in the FSM, the set contains at least one input sequence for which the output sequences of the two states disagree.

We note that A and $I^{\leq k+1}$ are fixed over all CTTs. However, some CTTs may impose additional conditions over the characterization set W. Since 1978, there have been multiple CTTs proposed: UIO [25], UIOv [26], W [27, 28], Wp [29], HSI [30, 31, 32, 33], PDS [14, 34, 35], ADS [16, 33], H-ADS [18], SPY [19], SPY-H [20], P [36], I-ADS [33], and H [37, 38]. Figure 2 shows a 'family-tree' of different CTTs. We do not include the UIO and the UIOv methods in the figure as these methods are not m-complete.

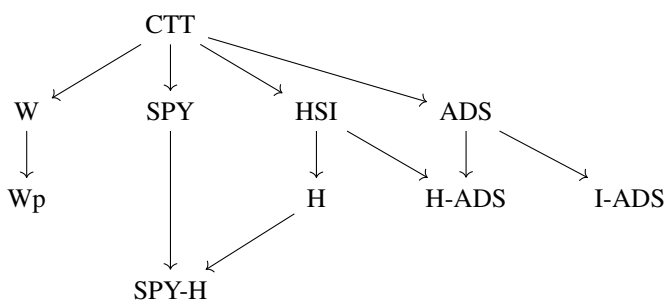

Fig. 2. Relationship of different CTTs: Arrows imply "derivation", that is, H-ADS can be derived (or is inspired) from the HSI and ADS algorithms.

C. Overview of CTTs

We provide a short overview of all the above CTTs here. We have selected eight CTTs: W, Wp, HSI, H-ADS, H, SPY, SPY-H, and I-ADS. The P-method was excluded as we did not find an implementation and the algorithm is very complex. All of these techniques offer the same m-complete guarantee.

The W-method [27, 28] first introduced the notion of a characterization set in conformance testing. However, it does not mandate any particular algorithm for the construction of the set. In our implementation, we have considered the standard Hopcroft partition algorithm for constructing the characterization set. One can also use a simple breadth-first search approach to construct separating sequences[3] for each pair of states; the collection of these sequences is also a characterization set.

The Partial-W (or Wp) method [29] is an improvement of the W-method: it divides the characterization set used by the W-method into (smaller) per-state characterization sets. In other words, for some state s, the Wp-method reduces the characterization set W to the set W_s, consisting of sequences that are necessary to "characterize" state s.

The Harmonic State Identifier (HSI) [30, 31, 33, 32] algorithm is similar to the W/Wp-methods, with the extra requirement that the per-state characterization sets be harmonized. Two characterization sets for states s and t are harmonized if the characterization sets W_s and W_t share a separating sequence for states s and t. Note, the characterization sets created with the HSI algorithm can be used in the W and Wp CTTs as well.

The Preset Distinguishing Sequence (PDS) [14, 34, 35] method attempts to construct a singleton characterization set, but suffers from the drawback that a PDS may not always exist. The Adaptive Distinguishing Sequence (ADS) [16] method attempts to expand the applicability of the PDS method by using an adaptive distinguishing sequence: An ADS is a decision tree where the nodes contain an input symbol and a set of states, with the leaf nodes containing no input symbols and just one state. The sequence of input symbols from the root node to a leaf node uniquely identifies the state at the leaf node. A disadvantage of the ADS-method is that it can only be used for FSMs which admit an ADS; thus, this method cannot be used universally.

The Hybrid-ADS (H-ADS) [39] and I-ADS [33] methods both attempt to modify the ADS-method such that it can be used universally. Both choose similar approaches: while the H-ADS method supplements results of the ADS-method with HSI sequences to identify a state, the I-ADS method creates a set of incomplete ADSs which together uniquely identify a state.

The H, SPY, and SPY-H methods differ from the previous methods in the way they construct their test-suites. The H-method [37, 38] improves on the W-method by selecting separating sequences on-the-fly during construction of a "testing-tree". The SPY-method [19] attempts to reduce "test-branching" by combining shorter test sequences into a single larger test sequence. The SPY-H [20] method combines both approaches, and is claimed to be the best at fault detection in [40].

D. Related Work

Dorofeeva et al. [15] present a survey of the W, Wp, HSI, H, UIOv, UIO, and PDS CTTs on 250 randomly generated FSMs with $k = \{1, 2\}$. Additionally, they also present experimental results for two realistic FSMs. They compare CTTs on the aggregate length of all sequences in the test suite. Note, this metric ignores the number of resets in the test suite. The authors note that while the H-method outperforms the HSI-method in general, the improvement is bigger in realistic FSMs than randomly generated FSMs. However, we note that the approach used in [15] essentially benchmarks a single equivalence query. In active learning, we may require multiple equivalence queries to learn an FSM, and the size of the test suite is not as important as finding a counterexample as quickly as possible. While it is reasonable to assume that a smaller test suite would be better, we do not know how these CTTs

[3] A separating sequence for a pair of distinct states is an input sequence for which the two states have different output responses.

perform over the course of an entire learning experiment. Additionally, their work does not consider more recent CTTs such as H-ADS, SPY, and SPY-H.

Aichernig et al. [41] compare combinations of W, Wp, mutation-coverage, transition-coverage, RandomWords[4], RandomWalk[5], and Randomized-Wp CTTs and three learning algorithms according to the total number of input symbols used for answering output and equivalence queries. Informally, RandomWalk/RandomWords CTTs generate random sequences for testing, while Randomized-Wp generates a random set of infix sequences. Their experiments were performed on 39 benchmark FSMs from the AutomataWiki. They recommend that either the Rivest-Schapire [4] or TTT [3] be used as the learning algorithm and either mutation-based testing or the Randomized-Wp methods be used as CTTs. However, they do not consider newer CTTs in their study.

Thus, to the best of the authors' knowledge, while there exist a multitude of different conformance testing techniques for FSMs, with the exception of [41], there do not appear to be recent studies evaluating the performance of these techniques in the context of automata learning.

III. Methodology

According to Howar et al. [17], efficient model learning requires finding counterexamples fast with a minimal number of symbols and resets. In this section, we introduce a metric that quantifies the efficiency of active learning experiments as a function of the state detection rate and total cost of learning.

A. Average Percentage of Faults Detected During Learning

Our metric builds upon principles from test prioritization, in particular the average percentage of faults detected [42]. Let $\mathtt{TC}$ be the total cost defined by the number of symbols and resets posed in learning and testing a correct model from a SUL with $|S|$ states, where $\mathtt{TC}_i$ indicates the experiment cost until round i. Let $\Delta\mathcal{H}_i$ be the number of states detected between rounds i and $i-1$ of a learning experiment. We define the average percentage of faults detected in a learning experiment ($\mathtt{APFD}_L$) as in Equation (1).

$$\mathtt{APFD}_L = 1 - \frac{\sum_{i=1}^{|S|}(\mathtt{TC}_i \cdot \Delta\mathcal{H}_i)}{\mathtt{TC} \cdot |S|} + \frac{1}{2 \cdot \mathtt{TC}} \tag{1}$$

The $\mathtt{APFD}_L$ is derived from a series of data points indicating the total cost required to discover a fraction of the states of a SUL during a learning experiment. These data points are gathered whenever a round is completed. The term $\frac{1}{2\cdot\mathtt{TC}}$ is necessary to ensure that the area under the curve of $\mathtt{APFD}_L$ is 100%. The $\mathtt{APFD}_L$ value, without $\frac{1}{2\cdot\mathtt{TC}}$, for $\mathtt{TC} = 1$ would be 50%, which is incorrect; adding the corrective $\frac{1}{2\cdot\mathtt{TC}}$ makes it 100%. In Figure 3, we illustrate the percentage of states discovered per round in four experiments performed on TCP[6] [43].

[4]Randomly generated sequences.
[5]Similar to RandomWords.
[6]Transmission Control Protocol.

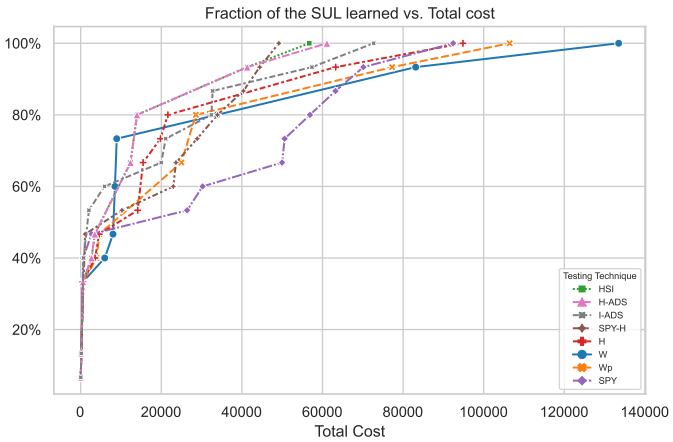

Fig. 3. Example illustrating APFD in learning

As in test prioritization [42], the $\mathtt{APFD}_L$ is a weighted sum of the total costs $\mathtt{TC}_i$ for learning $\Delta\mathcal{H}_i$ states after each round i of a learning experiment is completed. This summation is then normalized by the total cost $\mathtt{TC}$ to learn the SUL and the number of states in the SUL $|S|$. The $\mathtt{APFD}_L$ provides a numeric value ranging from 0 to 100 that quantifies the area across the curve, where higher values imply faster state detection during a learning experiment. In Table I, we report the number of symbols and resets posed during learning ($\mathtt{LSR}$) and testing ($\mathtt{TSR}$), and their sums indicated by the total cost ($\mathtt{TC}$) and the numbers of rounds required by four CTTs in learning the TCP model. The learning efficiency provided by each CTT is indicated by $\mathtt{APFD}_L$.

TABLE I
Cost and Number of Rounds Required per CTT - TCP Model

CTT	LSR	TSR	TC	EQ	$\mathtt{APFD}_L$
HSI	2351	54372	56723	9	96.71%
H-ADS	2351	58751	61102	9	96.65%
I-ADS	2138	70558	72696	11	96.15%
SPY-H	2245	46925	49170	11	95.99%
H	2169	92687	94856	11	95.09%
W	2291	131208	133499	10	94.06%
Wp	2314	104129	106443	9	93.77%
SPY	2218	90207	92425	12	93.12%

In terms of TC, we see that SPY-H resulted in a more efficient learning experiment while using two more EQs as compared to HSI; with the latter having a higher $\mathtt{APFD}_L$. Given that all CTTs produced correct models, this evidence indicates that testing techniques can have very distinct state detection capacities during active learning experiments. In Figure 4, we show a heatmap for the interpolation of the percentage of states learned of the TCP model from the TC for learning. Darker cells indicate higher numbers of symbols and resets to learn a given percentage of states.

As in previous studies [44], the W method entailed the highest TC to completely learn the SUL model while SPY-H had the lowest TC. We see that up to the 90% point, the HSI method had a lower TC than the SPY-H method.

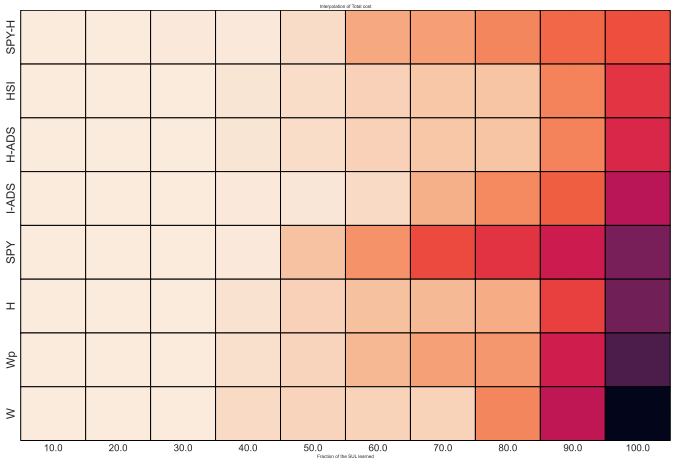

Fig. 4. TC for % of the SUL Learned - TCP model. Darker shades indicate higher TC.

Additionally, up to the 70% point, the W method performed approximately the same as the others. These findings suggest that the superiority of a CTT may be not consistent throughout the learning process. Therefore, tracking state detection during learning experiments becomes relevant to precisely measure the speed at which a CTT finds counterexamples and states are discovered by an active learning algorithm.

IV. Experimental Evaluation

An efficient active learning experiment is characterized by finding counterexamples fast and saving symbols and resets. To quantify the efficiency of an active learning experiment, this work introduces a metric to calculate the weighted cost for discovering states of a SUL. In this section, we present an experiment designed to analyze the performance of several CTTs in active learning experiments. We present the subject systems used as SULs in our experiments, the implementation artifacts, the configuration parameters and modes for conformance testing, and metrics that guided our experiment.

A. Subject Systems

For our experimental evaluation, we have chosen to simulate models learned from real-world communication protocols. We relied on models of various communication protocols available in the AutomataWiki [21]. In Table II, we report characteristics of our subject models in terms of numbers of states and inputs.

TABLE II
Description of the Systems Under Learning

Group	# Models	# States	# Inputs
Bankcards [8]	11	4-9	14-15
SSH [45]	3	17-66	13-22
TCP [43]	6	12-57	10-13
TLS [12]	23	7-15	8-12
Philips models [46]	3	34-58	14-22

In total, we have 46 models describing realistic behavior of protocols used in 4 different domains. In terms of size, the subjects have an average of 10 states and up to 15 inputs symbols in their alphabet. The largest model in our dataset is BitVise, with 66 states and 13 inputs. Figure 5 presents the distribution of benchmark FSMs over the size of their input alphabet and number of states. As we can observe, the majority of our communication protocols are small in size.

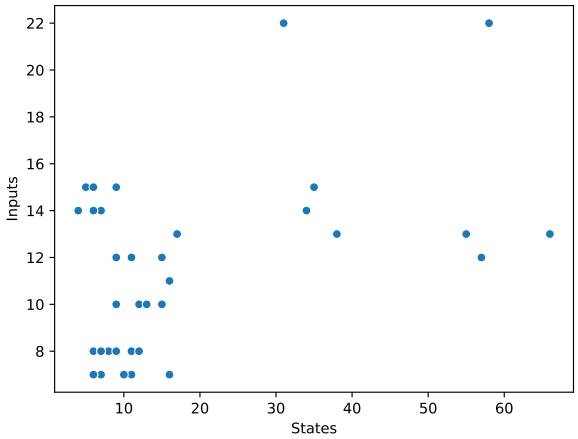

Fig. 5. Scatterplot of the size of the inputs and the number of states in the benchmark FSMs.

B. Model Learning and Testing Implementations

To perform the active learning experiments, we relied on the $L^{\#}$ algorithm [22], the state-of-the-art for observation tree-based active learning. We built our methodology upon an implementation of the $L^{\#}$ algorithm by logging the total cost for learning and testing after each equivalence query and calculating the $\texttt{APFD}_L$. For the sake of reproducibility, the artifacts created in this work are made available online[7].

To approximate an equivalence oracle, we relied on a set of eight different CTTs, ranging from the most classical CTTs (W, Wp) to the more recent ones relying on state identifiers (HSI, H, SPY, SPY-H) and adaptive distinguishing sequences (H-ADS and I-ADS). As being the oldest and most familiar CTT, the W method was selected as baseline for comparisons against the others. All the chosen CTTs generate m-complete test-suites. At the time of writing this work, this is the most extensive empirical study evaluating the efficiency of CTTs in active model learning.

We extended the $L^{\#}$ implementation to support the W, Wp, HSI, H-ADS, and I-ADS conformance testers. All test-suites generated are *maximal*, that is, no test is a prefix of another test. As our implementation of the H-ADS method, we relied on portions of the work done by Soucha [33]. The implementations of the H, SPY, and SPY-H methods are reused from the FSMlib [47]. Given that the above three algorithms use a test-tree to construct their test-suites, the test-suites are maximal by construction. For all learning experiments,

[7]Available on Zenodo, DOI: 10.5281/zenodo.8117699

duplicated OQs were filtered out and counted once with the support of a cache.

C. Parameters for Conformance Testing

Recall that, in conformance testing, we typically do not know the number of states in an implementation. Hence, an upper bound m is often taken as assumption. Similarly, in active learning, a constant number of extra states k is used by CTTs to aid state discovery from an intermediate hypothesis with n states, such that this upper-bound increases whenever a new state is found, i.e., $m = n + k$. Nevertheless, as the cost for conformance testing grows exponentially, choosing a very large value for k may result in incredibly effective but inefficiently large test-suites. Based on results from previous empirical studies [15, 33], we set our fixed values for $k \in \{2, 3\}$.

CTTs in Randomized and Non-Randomized Modes: Underestimating k can cause learning incompleteness while overestimation can cause scalability issues. This trade-off has motivated the investigation of CTT variants where the number of extra states k is randomized, rather than being set as a fixed value. Randomized CTTs have been known to seemingly work better than fixed approaches [41, 18]. Thus, we include both randomized and fixed approaches in our experiments.

Recall that CTTs usually require three components: a set of access sequences A, the set of infix sequences $I^{\leq(k+1)}$, and a characterization set W.

In randomized mode, a CTT takes two parameters: a minimum number of extra states k and an expected random length l. The length of an infix sequence is defined by a function $max(k, exp(l))$, where $exp(l)$ is the number of Bernoulli trials required for success. In this work, the only CTTs which support randomized mode are W, Wp, HSI, H-ADS, and I-ADS. The H, SPY, and SPY-H methods are left out, as they are built upon a finite infix set.

For our randomized experiments, we defined $k \in \{0, 1, 2\}$ and $l \in \{2, 3\}$ for eligible CTTs. Since randomized experiments no longer offer guarantees to m-completeness, we repeat each randomized learning experiment 50 times to produce a sample of measurements of learning costs and hypothesis sizes.

D. Metrics

In practice, active learning experiments stop once the SUL is learned. Since the SUL is unknown, domain knowledge or other stopping criteria are needed to determine if an SUL has been learned. Thus, the most important metric is the overall amount of time taken to finish an experiment. However, running time is a very difficult metric to capture, since experiments must be repeated in isolation to achieve stability of measurement and the results will differ depending on the hardware of the host machine. To address this problem, the aggregate number of symbols and resets in OQs is a popular approximated, time-independent metric for learning efficiency. We measure the costs inherent to the learning and testing phases of our experiments by recording the number of times a SUL is reset and the number of symbols posed in each type of queries. The total cost (TC) is estimated by summing up the individual costs of both OQs and EQs. To derive the $\texttt{APFD}_L$, we also keep track of the hypothesis size during each equivalence query.

To analyze our results, we relied on Python and the Jupyter notebook platform. We used the Mann-Whitney U test to measure the statistical significance ($p < 0.05$) between the total cost of the learning experiments. We complement our statistical analysis with the Vargha-Delaney's $\hat{A}$ effect size to measure the significance of the difference in the total cost of the techniques. To categorize the magnitude of the effect size, we used the intervals between the value of $\hat{A}$ and 0.5 as shown in Table III. Informally, in a comparison between A and B, the value of $\hat{A}$ may range from $[0, 1]$. If the value of $(\hat{A} - 0.5)$ is positive, then "A" is better than "B" (and vice-versa); while the magnitude of the difference indicates the effect size.

TABLE III
SIGNIFICANCE RESULTS

Effect Size	Effect Range
Negligible	$[0.000, 0.147)$
Small	$[0.147, 0.330)$
Medium	$[0.330, 0.474)$
Large	$[0.474, 0.500]$

V. DISCUSSION

We now discuss the results of our experiments for non-randomized, randomized, and finally, a comparison between the best of both the CTT modes. Note that, in all tables, the more performant variant is listed under column "A".

Non-Randomized Experiments

Experiments with $k = 1$ resulted in some learning experiments terminating unsuccessfully (that is, we did not learn the correct FSM). We have thus excluded all learning experiments with $k = 1$ from our analysis. Figure 6 visualizes results from our non-randomized experiments for $k = 2$. We present boxplots of the TC for our eight CTTs: the y-axis is the TC (at log-scale) and the x-axis represents the different CTTs. Outliers are represented with the "$\times$" symbol.

In general, we infer from the plot that the total cost between different CTTs remains roughly the same. As expected, experiments using the W CTT performed the worst amongst all eight CTTs, while the rest seem to perform roughly the same with H-ADS and I-ADS performing marginally better. Table IV contains the statistically significant results. Entries listed under the column 'A' perform better than entries under column 'B' with the corresponding effect size. The complete list including statistically insignificant results is available in the supplementary material.

From Table IV, we find that all comparisons have a 'small' effect on performance, with the exception of I-ADS against W. We note that amongst the newer CTTs, I-ADS performs marginally better than SPY and SPY-H. There does not appear

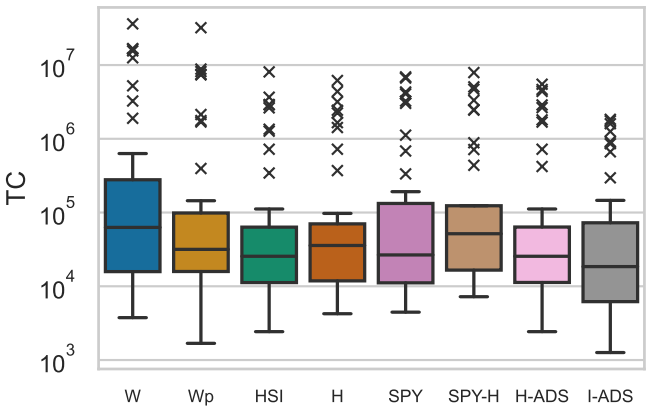

Fig. 6. Total Cost of learning per CTT in Non-Randomized mode for $k = 2$.

TABLE IV
STATISTICALLY SIGNIFICANT EFFECT SIZES FOR TC BETWEEN NON-RANDOMIZED CTTS

A	B	Effect Size	Effect Magnitude
H-ADS	W	0.136	small
HSI	W	0.140	small
I-ADS	SPY	0.120	small
	SPY-H	0.161	small
	W	0.179	medium
	Wp	0.122	small

to be a clear 'winner' between the newer CTTs in terms of total cost of learning: there is no significant difference between the performance of HSI, H-ADS, I-ADS; while SPY-(H) seems to perform *worse* than I-ADS. Note, however, that the size of the effect is small.

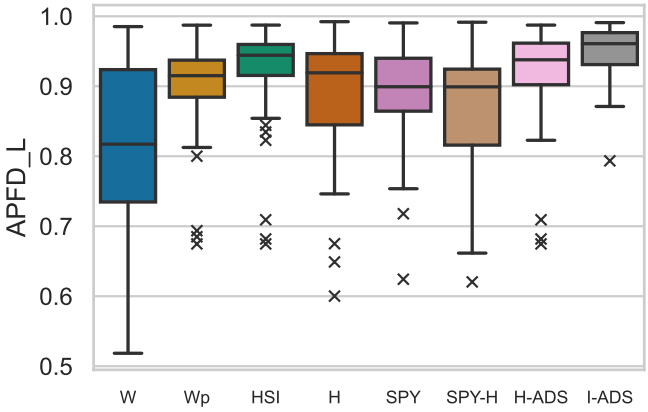

Fig. 7. $\texttt{APFD}_L$ per CTT in Non-Randomized mode for $k = 2$.

The $\texttt{APFD}_L$ plot (Figure 7), however, shows a larger difference. Recall that the $\texttt{APFD}_L$ metric measures efficiency of learning as how fast we manage to learn an FSM if we do not know the actual FSM. In this case, we see larger differences in experiments with W where it clearly performed worse than all the others while I-ADS and HSI seem to perform the best. Table V contains the results for all statistically significant experiments. As expected, the baseline experiments with the W CTT performed the worst, with effect sizes being medium or large. I-ADS seems to be a very clear winner in this comparison – I-ADS performs better than all the other CTTs with small advantages over the H-ADS and HSI experiments and large differences over all the rest. Unexpectedly, the SPY-(H) CTTs do not perform better than any CTT other than the W algorithm. In summary, the $\texttt{APFD}_L$ comparisons indicate a general suggestion to use the I-ADS CTT.

Efficiency of CTTs in Non-Randomized Experiments

The I-ADS CTT offers a minor improvement over W in terms of total cost of learning. $\texttt{APFD}_L$ *indicates that I-ADS, H-ADS, and HSI CTTs perform better than all others CTTs with the baseline W performing the worst.*

TABLE V
STATISTICALLY SIGNIFICANT EFFECT SIZES FOR APFD BETWEEN NON-RANDOMIZED CTTS

A	B	Effect Size	Effect Magnitude
H	W	0.172	medium
H-ADS	SPY	0.120	small
	SPY-H	0.210	medium
	W	0.265	large
	Wp	0.123	small
HSI	H	0.132	small
	SPY	0.137	small
	SPY-H	0.226	medium
	W	0.278	large
	Wp	0.148	small
I-ADS	H	0.249	large
	H-ADS	0.154	small
	HSI	0.140	small
	SPY	0.237	large
	SPY-H	0.324	large
	W	0.348	large
	Wp	0.248	large
SPY	W	0.202	medium
Wp	W	0.213	medium

Randomized Experiments

Our randomized learning experiments were repeated 50 times over parameters $k = \{0, 1, 2\}$ and $l = \{2, 3\}$. While all combinations of experiments finished successfully, in this text, we discuss only the results of the experiments with $k = 2$ and $l = \{2, 3\}$, as comparisons with other parameters indicate at most a negligible difference between the techniques, if any. In this section, we skip the SPY, H, and SPY-H CTTs as those cannot be run in a randomized style.

Figure 8 visualizes the differences in average performance in total cost of all learning experiments. We see no difference between the TC for any CTT for either $l = 2$ or $l = 3$. Statistical analysis between all combinations, listed under Table VI,

show that if there is a statistically significant difference, it is negligible for all combinations. The entries in table VI list CTTs in the format "CTT,l", as k is fixed to 2. In general, the randomized W CTT seems to perform worse in general against all CTTs, with the I-ADS CTT with parameters $(k, l) = (2, 3)$ performing worse against the other techniques. Note, while we have only listed comparisons for $k = 2$, this result also holds for $k = 0$ and $k = 1$ as well.

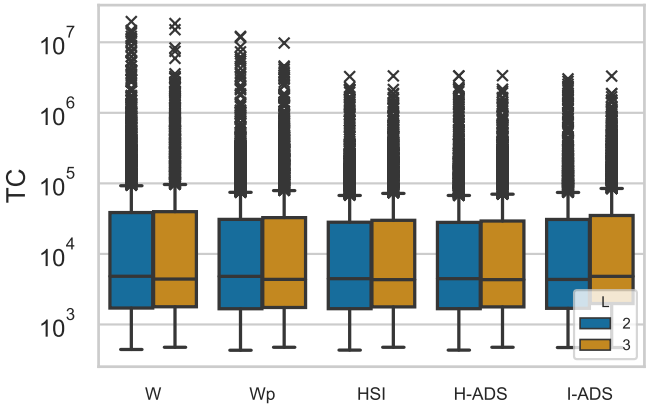

Fig. 8. Total Cost of learning per CTT in Randomized mode.

TABLE VI
STATISTICALLY SIGNIFICANT EFFECT SIZES FOR TC BETWEEN RANDOMIZED CTTS FOR K = 2.

A	B	Effect Size	Effect Magnitude
H-ADS,2	I-ADS,3	0.019	negligible
	W,2	0.022	negligible
	W,3	0.020	negligible
H-ADS,3	I-ADS,3	0.017	negligible
	W,2	0.019	negligible
	W,3	0.018	negligible
HSI,2	I-ADS,3	0.019	negligible
	W,2	0.022	negligible
	W,3	0.020	negligible
HSI,3	I-ADS,3	0.017	negligible
	W,2	0.019	negligible
	W,3	0.018	negligible
I-ADS,2	I-ADS,3	0.018	negligible
	W,2	0.020	negligible
	W,3	0.019	negligible

Next we discuss the results of our $\texttt{APFD}_L$ analysis. Figure 9 visualizes results for the $\texttt{APFD}_L$ metric for the randomized experiments. Again, there is no notable difference visible in the plot. All experiments appear to be very close to the maximum $\texttt{APFD}_L$ of 1.0. Table VII lists the results of our statistical analysis. The entries in table VII list CTTs in the format "CTT,l", as k is fixed to 2. The results of the $\texttt{APFD}_L$ metric seem to follow those of the total cost analysis: the randomized W CTT performs worse in general, alongside I-ADS with $l = 3$. And while there are statistically significant differences between the performance, the differences are negligible.

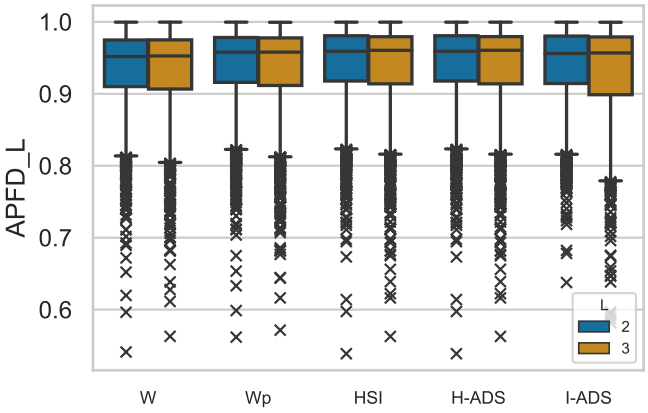

Fig. 9. $\texttt{APFD}_L$ per CTT in Randomized mode.

TABLE VII
STATISTICALLY SIGNIFICANT EFFECT SIZES FOR $\texttt{APFD}_L$ BETWEEN RANDOMIZED CTTS FOR K=2

A	B	Effect Size	Effect Magnitude
H-ADS,2	I-ADS,3	0.027	negligible
	W,2	0.050	negligible
	W,3	0.052	negligible
	Wp,3	0.022	negligible
H-ADS,3	I-ADS,3	0.022	negligible
	W,2	0.045	negligible
	W,3	0.048	negligible
	Wp,3	0.017	negligible
HSI,2	I-ADS,3	0.027	negligible
	W,2	0.050	negligible
	W,3	0.053	negligible
	Wp,3	0.023	negligible
HSI,3	I-ADS,3	0.022	negligible
	W,2	0.045	negligible
	W,3	0.048	negligible
	Wp,3	0.017	negligible
I-ADS,2	I-ADS,3	0.020	negligible
	W,2	0.042	negligible
	W,3	0.044	negligible
I-ADS,3	W,2	0.022	negligible
	W,3	0.024	negligible
Wp,2	W,2	0.035	negligible
	W,3	0.038	negligible
Wp,3	W,2	0.029	negligible
	W,3	0.032	negligible

Efficiency of CTTs in Randomized Experiments

Variation of k, l does not have a noticeable effect on the performance of all CTTs in terms of total cost of learning or $\texttt{APFD}_L$, with all learning experiments finishing successfully. Randomized CTTs have negligible performance differences, if any.

Overall Difference Between the Best of Each CTT Mode

Finally, we compare the best of the randomized and non-randomized approaches against each other. For the randomized

experiments, we have chosen $(k, l) = (2, 2)$. Experimental results for the randomized approach have been averaged across all 50 runs. As the differences between randomized CTTs is negligible, we compare the results of randomized vs non-randomized directly.

Figure 11 visualizes the total cost of learning for both approaches (Non-randomized is marked as "Fixed"). We note that the cost of learning for the randomized CTTs is almost an order of magnitude lower than that of the non-randomized CTTs for almost all CTTs. Statistical analysis of the results indicate that the randomized CTTs perform better with a "medium" effect (Â=0.228).

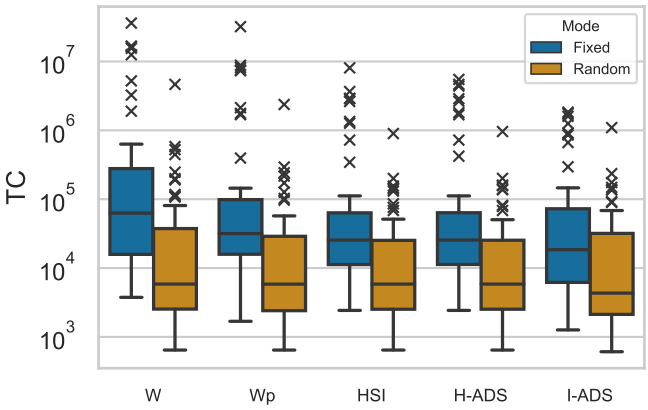

Fig. 10. Total Cost of learning per CTT in the Best of Each Mode. "Fixed" indicates Non-Randomized.

On the other hand, the $\texttt{APFD}_L$ plots in Figure 11 show the same trend: randomized CTTs all have higher $\texttt{APFD}_L$ values, close to the maximum when compared to their non-randomized counterparts. Statistical analysis of the results indicate that the randomized CTTs perform better with a "small" effect (Â=0.132).

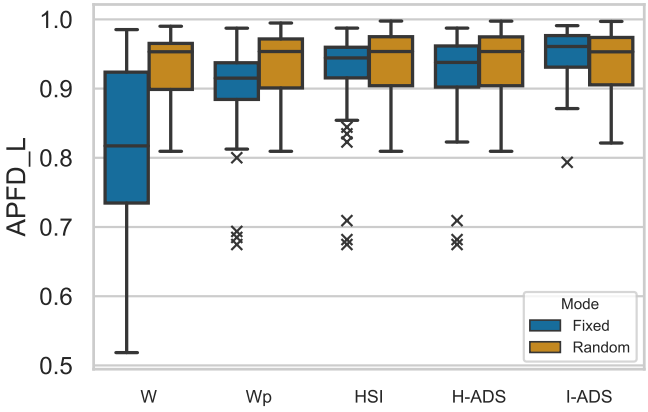

Fig. 11. $\texttt{APFD}_L$ per CTT in the Best of Each Mode. "Fixed" indicates Non-Randomized.

Overall, our results indicate that randomized CTTs in general perform better than non-randomized CTTs both in terms of total cost of learning and efficiency of learning.

Comparison of the Best CTTs of Each Execution Mode

Randomized CTTs reduce total cost of learning by almost an order of magnitude over non-randomized CTTs while improving $\texttt{APFD}_L$ *to almost 1. However, this comes at the cost of loss of* m*-completeness.*

Threats to Validity

1) Learning Alogrithms: In our work, we have only considered the $L^{\#}$ learning algorithm. Different learning algorithms (such as Rivest-Schapire or TTT) may have an influence on our results. However, given that conformance testing is responsible for a majority of all OQs in active learning, and all the state-of-the-art learning algorithms have the same query complexity, we believe that our results will remain valid.

2) Subject Systems: Our subject systems are simulated FSMs of communication protocols, which are relatively small (both in terms of number of states and size of the input alphabet). Experimental evaluation of different types of systems, such as control software might affect the results. The AutomataWiki additionally contains FSMs of software systems from ASML from the Rigorous Examination of Reactive Systems (RERS) challenge in 2019 [48]. While there are 46 FSMs available under this category, we do not include these in our evaluation as these models are extremely sparse: they typically have a more than 50 inputs and at each state of the FSMs, only a few (generally 2-4) inputs do not lead to a sink state. As such, these models are extremely difficult to test, and to the best of our knowledge, state-of-the-art black-box CTTs are unable to test these FSMs in a reasonable amount of time.

3) Size of FSMs: Our simulated FSMs are not large. We have only one FSM of a large industrial component with 3410 states and 80 inputs from [18], thus experimental evaluation of that FSM would not permit us to draw general conclusions.

VI. Conclusions and Future Work

In this paper, we present an empirical evaluation of the efficiency of CTTs in a setting of automata learning. We measured the total cost of learning 46 simulated models of communication protocols and the fault detection capacity entailed by eight different CTTs. To characterize the state detection capacity of model learning, we adapt the APFD efficiency metric from test prioritization research and introduce the notion of average percentage of faults detected in learning ($\texttt{APFD}_L$). $\texttt{APFD}_L$ is derived from a series of data points indicating the partial cost for discovering a fraction of the states of a SUL during a learning experiment.

Our experiments showed a minor difference between the CTTs in terms of total number of symbols and resets. In contrast, the fault detection capacity of the CTTs was found to be significantly distinct, with the I-ADS method as being the most efficient. Nevertheless, these differences become negligible when CTTs are applied in randomized mode and hence, reinforce the positive role of randomness in improving the efficiency of model learning, despite compromising test completeness. As future work, we plan to perform more

extensive evaluations of our empirical methodology with larger models. In particular, the ESM and ASML models in the AutomataWiki should provide an interesting setting for assessing the fault detection of CTTs in very complex and large behavioral models.

References

[1] D. Angluin, "Learning regular sets from queries and counterexamples," *Inf. Comput.*, vol. 75, no. 2, pp. 87–106, 1987.

[2] ——, "Queries and concept learning," *Mach. Learn.*, vol. 2, no. 4, pp. 319–342, 1987.

[3] M. Isberner, F. Howar, and B. 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*, B. Bonakdarpour and S. A. Smolka, Eds. Cham: Springer International Publishing, 2014, pp. 307–322.

[4] R. Rivest and R. Schapire, "Inference of finite automata using homing sequences," *Inf. Comput.*, vol. 103, no. 2, pp. 299–347, 1993.

[5] F. Aarts, H. Kuppens, G. Tretmans, F. Vaandrager, and S. Verwer, "Learning and testing the bounded retransmission protocol," in *Proceedings 11th International Conference on Grammatical Inference (ICGI 2012), September 5-8, 2012. University of Maryland, College Park, USA*, ser. JMLR Workshop and Conference Proceedings, J. Heinz, C. d. l. Higuera, and T. Oates, Eds., vol. 21, 2012, pp. 4–18.

[6] P. Fiterău-Broştean, R. Janssen, and F. Vaandrager, "Learning fragments of the TCP network protocol," in *Proceedings 19th International Workshop on Formal Methods for Industrial Critical Systems (FMICS'14),* Florence, Italy, ser. LNCS, F. Lang and F. Flammini, Eds., vol. 8718. Springer, Sep. 2014, pp. 78–93.

[7] K. Aslam, Y. Luo, R. R. H. Schiffelers, and M. van den Brand, "Interface protocol inference to aid understanding legacy software components," in *Proceedings of MODELS 2018 Workshops*, ser. CEUR Workshop Proceedings, R. Hebig and T. Berger, Eds., vol. 2245. CEUR-WS.org, 2018, pp. 6–11.

[8] F. Aarts, J. de Ruiter, and E. Poll, "Formal models of bank cards for free," in *Software Testing Verification and Validation Workshop, IEEE International Conference on*. Los Alamitos, CA, USA: IEEE Computer Society, 2013, pp. 461–468.

[9] F. Vaandrager, "Model learning," *Communications of the ACM*, vol. 60, no. 2, pp. 86–95, Feb. 2017.

[10] F. Howar and B. Steffen, "Active automata learning in practice," in *Machine Learning for Dynamic Software Analysis: Potentials and Limits: International Dagstuhl Seminar 16172, Dagstuhl Castle, Germany, April 24-27, 2016, Revised Papers*, A. Bennaceur, R. Hähnle, and K. Meinke, Eds. Springer International Publishing, 2018, pp. 123–148.

[11] K. Aslam, L. Cleophas, R. R. H. Schiffelers, and M. van den Brand, "Interface protocol inference to aid understanding legacy software components," *Softw. Syst. Model.*, vol. 19, no. 6, pp. 1519–1540, 2020.

[12] J. de Ruiter and E. Poll, "Protocol state fuzzing of TLS implementations," in *USENIX Security Symp.* USENIX, Aug. 2015, pp. 193–206.

[13] M. Tappler, B. K. Aichernig, and R. Bloem, "Model-based testing iot communication via active automata learning," in *2017 IEEE International conference on software testing, verification and validation (ICST)*. IEEE, 2017, pp. 276–287.

[14] E. Moore, "Gedanken-experiments on sequential machines," in *Automata Studies*, ser. Annals of Mathematics Studies, vol. 34. Princeton University Press, 1956, pp. 129–153.

[15] R. Dorofeeva, K. El-Fakih, S. Maag, A. R. Cavalli, and N. Yevtushenko, "Fsm-based conformance testing methods: A survey annotated with experimental evaluation," *Information & Software Technology*, vol. 52, no. 12, pp. 1286–1297, 2010.

[16] D. Lee and M. Yannakakis, "Testing finite-state machines: State identification and verification," vol. 43, no. 3, pp. 306–320, 1994.

[17] F. Howar, B. Steffen, and M. Merten, "From zulu to rers: Lessons learned in the zulu challenge," 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 4*. Springer, 2010, pp. 687–704.

[18] W. Smeenk, J. Moerman, F. W. Vaandrager, and D. N. Jansen, "Applying automata learning to embedded control software," in *Formal Methods and Software Engineering - 17th International Conference on Formal Engineering Methods, ICFEM 2015, France, 2015, Proceedings*, ser. LNCS, M. J. Butler, S. Conchon, and F. Zaïdi, Eds., vol. 9407. Springer, 2015, pp. 67–83.

[19] A. Simao, A. Petrenko, and N. Yevtushenko, "On reducing test length for fsms with extra states," *Software testing, verification and reliability*, vol. 22, no. 6, pp. 435–454, 2012.

[20] M. Soucha and K. Bogdanov, "Spyh-method: an improvement in testing of finite-state machines," in *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2018, pp. 194–203.

[21] D. Neider, R. Smetsers, F. W. Vaandrager, and H. Kuppens, "Benchmarks for automata learning and conformance testing," in *Models, Mindsets, Meta: The What, the How, and the Why Not? - Essays Dedicated to Bernhard Steffen on the Occasion of His 60th Birthday*, ser. LNCS, T. Margaria, S. Graf, and K. G. Larsen, Eds., vol. 11200. Springer, 2018, pp. 390–416.

[22] F. W. Vaandrager, B. Garhewal, J. Rot, and T. Wißmann, "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*, ser. LNCS, D. Fisman and G. Rosu, Eds., vol. 13243. Springer, 2022, pp. 223–243.

[23] M. Shahbaz and R. Groz, "Inferring Mealy machines," in *FM 2009: Formal Methods, Second World Congress, Eindhoven, The Netherlands, November 2-6, 2009. Proceedings*, ser. LNCS, A. Cavalcanti and D. Dams, Eds., vol. 5850. Springer, 2009, pp. 207–222.

[24] M. Merten, F. Howar, B. Steffen, and T. Margaria, "Automata learning with on-the-fly direct hypothesis construction," in *Leveraging Applications of Formal Methods, Verification, and Validation - International Workshops, SARS 2011 and MLSC 2011. Revised Selected Papers*, ser. Communications in Computer and Information Science, R. Hähnle, J. Knoop, T. Margaria, D. Schreiner, and B. Steffen, Eds., vol. 336. Springer, 2011, pp. 248–260.

[25] K. Sabnani and A. Dahbura, "A protocol test generation procedure," *Computer Networks and ISDN systems*, vol. 15, no. 4, pp. 285–297, 1988.

[26] S. T. Vuong, "The uioυ method for protocol test sequence generation," in *Proc. of 2nd IFIP Int. Workshop on Protocol Test Systems (IWPTS'89)*, 1989, pp. 161–175.

[27] T. Chow, "Testing software design modeled by finite-state machines," vol. 4, no. 3, pp. 178–187, 1978.

[28] M. Vasilevskii, "Failure diagnosis of automata," *Cybernetics*, vol. 9, no. 4, pp. 653–665, 1973.

[29] S. Fujiwara, G. v. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi, "Test selection based on finite state models," vol. 17, no. 6, pp. 591–603, 1991.

[30] A. Petrenko, "Nondeterministic state machine in protocol conformance testing," *Protocol Test Systems*, pp. 363–378, 1994.

[31] A. Petrenko and N. Yevtushenko, "Testing from partial deterministic fsm specifications," *IEEE Transactions on Computers*, vol. 54, no. 9, pp. 1154–1165, 2005.

[32] N. Yevtushenko and A. Petrenko, "Test derivation method for an arbitrary deterministic automaton, automatic control and computer sciences," 1990.

[33] M. Soucha and K. Bogdanov, "State identification sequences from the splitting tree," *Information and Software Technology*, vol. 123, p. 106297, 2020.

[34] A. Gill *et al.*, "Introduction to the theory of finite-state machines," 1962.

[35] G. Gonenc, "A method for the design of fault detection experiments," *IEEE transactions on Computers*, vol. 100, no. 6, pp. 551–558, 1970.

[36] A. Simão and A. Petrenko, "Fault coverage-driven incremental test generation," *The Computer Journal*, vol. 53, no. 9, pp. 1508–1522, 2010.

[37] M. Dorofeeva and I. Koufareva, "Novel modification of the w-method," *Bulletin of the Novosibirsk Computing Center. Series: Computer Science*, vol. 2002, pp. 69–80, 2002.

[38] R. Dorofeeva, K. El-Fakih, and N. Yevtushenko, "An improved fsm based conformance testing method," in *Proc. of the IFIP 25th International Conference on Formal Methods for Networked and Distributed Systems*, pp. 204–218.

[39] W. Smeenk, J. Moerman, F. Vaandrager, and D. N. Jansen, "Applying automata learning to embedded control software," in *Formal Methods and Software Engineering*, M. Butler, S. Conchon, and F. Zaïdi, Eds. Cham: Springer International Publishing, 2015, pp. 67–83.

[40] M. Soucha, "Testing and active learning of resettable finite-state machines," Ph.D. dissertation, University of Sheffield, January 2019. [Online]. Available: https://etheses.whiterose.ac.uk/24370/

[41] B. K. Aichernig, M. Tappler, and F. Wallner, "Benchmarking combinations of learning and testing algorithms for active automata learning," in *Tests and Proofs*, W. Ahrendt and H. Wehrheim, Eds. Cham: Springer International Publishing, 2020, pp. 3–22.

[42] S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Test Case Prioritization: A Family of Empirical Studies," *IEEE Transactions on Software Engineering*, vol. 28, no. 2, pp. 159–182, Feb. 2002, place: Piscataway, NJ, USA Publisher: IEEE Press.

[43] P. Fiterău-Broştean, R. Janssen, and F. Vaandrager, "Combining Model Learning and Model Checking to Analyze TCP Implementations," in *Computer Aided Verification*, ser. LNCS, S. Chaudhuri and A. Farzan, Eds. Cham: Springer International Publishing, 2016, pp. 454–471.

[44] A. T. Endo and A. Simao, "Evaluating test suite characteristics, cost, and effectiveness of FSM-based testing methods," *Information and Software Technology*, vol. 55, no. 6, pp. 1045 – 1062, 2013.

[45] P. Fiterău-Broştean, T. Lenaerts, E. Poll, J. de Ruiter, F. Vaandrager, and P. Verleg, "Model learning and model checking of SSH implementations," in *Proceedings of the 24th ACM SIGSOFT International SPIN Symposium on Model Checking of Software*, ser. SPIN 2017. New York, NY, USA: ACM, 2017, pp. 142–151.

[46] M. Schuts, J. Hooman, and P. Tielemans, "Industrial experience with the migration of legacy models using a dsl," in *Proceedings of the Real World Domain Specific Languages Workshop 2018*, ser. RWDSL2018. New York, NY, USA: Association for Computing Machinery, 2018.

[47] M. Soucha, "FSMlib - A C++ library for handling Finite-State Machines, their testing and learning," Nov. 2021. [Online]. Available: https://github.com/Soucha/FSMlib

[48] M. Jasper, M. Mues, A. Murtovi, M. Schlüter, F. Howar, B. Steffen, M. Schordan, D. Hendriks, R. Schiffelers, H. Kuppens, and F. W. Vaandrager, "Rers 2019: Combining synthesis with real-world models," in *TACAS*, D. Beyer, M. Huisman, F. Kordon, and B. Steffen, Eds. Cham: Springer International Publishing, 2019, pp. 101–115.