ANALYZING ROBUSTNESS OF ANGLUIN'S L* ALGORITHM IN PRESENCE OF NOISE

LINA YE [a], IGOR KHMELNITSKY, SERGE HADDAD [b], BENOÎT BARBOT [c],
BENEDIKT BOLLIG [d], MARTIN LEUCKER [e], DANIEL NEIDER [f,g], AND RAJARSHI ROY [h]

[a] Université Paris-Saclay, CNRS, ENS Paris-Saclay, CentraleSupélec, LMF, France
e-mail address: lina.ye@lmf.cnrs.fr

[b] Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF, France
e-mail address: serge.haddad@lmf.cnrs.fr

[c] Université Paris-Est Créteil, France

[d] Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF, France

[e] Institute for Software Engineering and Programming Languages, Universität zu Lübeck, Germany

[f] TU Dortmund University, Germany

[g] Center for Trustworthy Data Science and Security, University Alliance Ruhr, Germany

[h] Max Planck Institute for Software Systems, Kaiserslautern, Germany

ABSTRACT. Angluin's L* algorithm learns the minimal deterministic finite automaton (DFA) of a regular language using membership and equivalence queries. Its probabilistic approximatively correct (PAC) version substitutes an equivalence query by numerous random membership queries to get a high level confidence to the answer. Thus it can be applied to any kind of device and may be viewed as an algorithm for synthesizing an automaton abstracting the behavior of the device based on observations. Here we are interested on how Angluin's PAC learning algorithm behaves for devices which are obtained from a DFA by introducing some noise. More precisely we study whether Angluin's algorithm reduces the noise and produces a DFA closer to the original one than the noisy device. We propose several ways to introduce the noise: (1) the noisy device inverts the classification of words w.r.t. the DFA with a small probability, (2) the noisy device modifies with a small probability the letters of the word before asking its classification w.r.t. the DFA, (3) the noisy device combines the classification of a word w.r.t. the DFA and its classification w.r.t. a counter automaton, and (4) the noisy DFA is obtained by a random process from two DFA such that the language of the first one is included in the second one. Then when a word is accepted (resp. rejected) by the first (resp. second) one, it is also accepted (resp. rejected) and in the remaining cases, it is accepted with probability 0.5. Our main experimental contributions consist in showing that: (1) Angluin's algorithm behaves well whenever the noisy device is produced by a random process, (2) but poorly with a structured noise, and, that (3) is able to eliminate pathological behaviours specified in a regular way. Theoretically, we show that randomness almost surely yields systems with non-recursively enumerable languages.

Introduction

Discrete-event systems and their languages. Discrete-event systems [CL10] form a large class of dynamic systems that, given some internal state, evolve from one state to another one due to the occurrence of an event. For instance, discrete-event systems can represent a cyber-physical process whose events are triggered by a controller or the environment, or, a business process whose events are triggered by human activities or software executions. Often, the behaviors of such systems are classified as safe (aka correct, representative, etc.) or unsafe. Since a behavior may be identified by its sequence of occurred events, this leads to the notion of a language.

Analysis versus synthesis. There are numerous formalisms to specify (languages of) discrete-event systems. From a designer's perpective, the simpler it is the better its analysis will be. So finite automata and their languages (regular languages) are good candidates for the specification thanks to their simplicity. However, even when the system is specified by an automaton, its implementation may slightly differ due to several reasons (bugs, unplanned human activities, unpredictable environment, etc.). Thus, one generally checks whether the implementation conforms to the specification. However, in many contexts, the system under consideration has already been implemented and the original specification (if any) is lost, as for instance in the framework of process mining [vdA12]. Thus, by observing and interacting with the system, one aims to recover a specification close to the system at hand but that is robust with respect to its pathologic behaviors.

Language learning. The problem of learning a language from finite samples of strings by discovering the corresponding grammar is known as grammatical inference. Its significance was initially stated in [Sol64] and an overview of very first results can be found in [BF72b]. As it may not always be possible to infer a grammar that exactly identifies a language, approximate language learning was introduced in [Wha74], where a grammar is selected from a solution space whose language approximates the target language with a specified degree of accuracy. To provide a deeper insight into language learning, the problem of identifying a (minimal) deterministic finite automaton (DFA) that is consistent with a given sample has attracted substantial attention in the literature since several decades [BF72a, Gol78, Ang87, Val84, JMKO20, FBLJ$^+$21]. An understanding of regular language learning is very valuable for a generalization to other more complex classes of languages. For example, Some researchers adapted learning algorithms for regular languages to learn context-free languages [CE07, Cla10, YC10].

Angluin's L* algorithm. Angluin's L* algorithm learns the minimal DFA of a regular language with two types of queries: membership queries and equivalence queries [Ang87]. Angluin's approach triggered a lot of subsequent research on active automata learning and has numerous applications, such as finding bugs in implementations of security-critical protocols [FJV16, FLP$^+$17, FH17], learning interfaces of classes in software libraries [HIS$^+$12], and inferring interface protocols of legacy software components [ACSvdB20]. One could of course try to adapt it to the synthesis task described above. Since 1987, different improvements of the original Angluin's L* algorithm have been proposed, thus resulting in numerous variants [RS89, KV94, MP91, SG09, BHKL09]. It is fair to say that L*-like algorithms completely dominate the research area of active automata learning. However, for most black box systems, it is often impossible to implement the equivalence query. Thus, its

probabilistic approximatively correct (PAC) version substitutes an equivalence query with a large enough set of random membership queries. Using a PAC framework, one needs to define and evaluate the accuracy of such an approach. Hence, here we are interested in how PAC Angluin's algorithm behaves for devices which are obtained from a DFA by introducing some noise.

Noisy learning. Most learning algorithms in the literature assume the correctness of the training data, including the example data such as attributes as well as classification results. However, sometimes noise-free datasets are not available. [Qui86] carried out an experimental study of the noise effects on the learning performance. The results showed that generally the classification noise had more negative impact than the attribute one, i.e., errors in the values of attributes. [AL87] studied how to compensate for randomly introduced noise and discovered a theorem giving a bound on the smaple size that is sufficient for PAC-identification in the presence of classification noise when the concept classes are finite. Michael Kearns formalized another related learning model from statistical queries by extending Valiant's learning model [Kea98]. One main result shows that any class of functions learnable from this statistical query model is also learnable with classification noise in Valiant's model.

Our contribution. In this paper, we study against which kinds of noise Angluin's algorithm is *robust*. In this work we use the optimized version of this algorithm from [KV94]. So to avoid confusion, we will call it the KV's algorithm. To the best of our knowledge, this is the very first attempt of noise analysis in the automata learning setting. More precisely, we consider the following setting (cf. Figure 1): Assume that a regular device $\mathcal{A}$ is given, typically as a black box. Due to some noise $\mathcal{N}$, the system $\mathcal{A}$ is pertubed resulting in a not necessarily regular system $\mathcal{M}_{\mathcal{N}}$. This one is consulted by the PAC version of KV's algorithm to obtain a regular system $\mathcal{A}_E$. The question studied in this paper is whether $\mathcal{A}_E$ is closer to $\mathcal{A}$ than $\mathcal{M}_{\mathcal{N}}$, or, in other words, to which extent learning via KV's algorithm is robust against the noise $\mathcal{N}$.
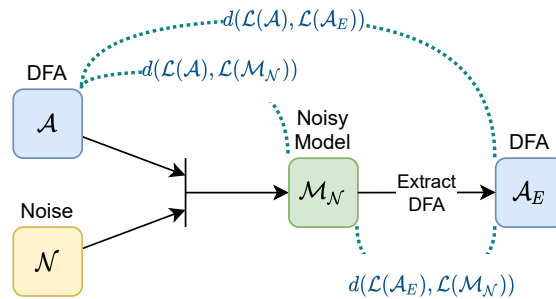

FIGURE 1. The experimental setup and the studied distances

To this end, we introduce four kinds of *noisy devices* obtained from $\mathcal{A}$:

(1) the noisy device is obtained by a random process from a given DFA by inverting the classification of words with a small probability, which corresponds to the classification noise in the classical learning setting;

(2) the noisy device is obtained by a random process that, with a small probability, replaces each letter of a word by one chosen uniformly from the alphabet and then determines its

classification based on the DFA, which corresponds to the attribute noise in the classical setting;

(3) the noisy DFA combines the classification of a word w.r.t. the DFA and its status w.r.t. a counter automaton;

(4) The noisy DFA is obtained by a random process from two DFA such that the language of the first one is included in the second one. Then when a word is accepted (resp. rejected) by the first (resp. second) one, it is also accepted (resp. rejected) and in the remaining cases, it is accepted with probability 0.5.

Our studies are based on the distribution over words that is used for generating words associated with membership queries and defining (and statistically measuring) the *distance* between two devices as the probability that they differ on word acceptance. We have performed experiments over several hundreds random DFA. We have pursued several goals along our experiments, expressed by the following questions:

- What is the threshold (in terms of distance) between pertubating the DFA or producing a device that is no more "similar to" the DFA?

- What is the impact of the nature of noise on the robustness of KV's algorithm?

- What is the impact of the words distribution on the robustness of KV's algorithm?

- How to reduce the size of the extracted DFA by KV's algorithm while keeping as close as possible to the original DFA?

Due to the approximating nature of the PAC version of L^*, we had to consider the question of how to choose the accuracy of the approximate equivalence query to get a good trade-off between accuracy and efficiency. Moreover, since in most cases, KV's algorithm may perform a huge number of refinement rounds before a possible termination, we considered what a "good" number of rounds to stop the algorithm avoiding underfitting and overfitting is.

We experimentally show that w.r.t. the random noise, i.e., the noise is randomly introduced, KV's algorithm behaves quite well, i.e., the learned DFA ($\mathcal{A}_E$) is very often closer to the original one ($\mathcal{A}$) than the noisy random device ($\mathcal{M}_\mathcal{N}$). When the noise is obtained using the counter automaton, KV's algorithm is not robust. Instead, the device $\mathcal{A}_E$ is closer to the noisy device $\mathcal{M}_\mathcal{N}$. Moreover, we establish that the expectation of the length of a random word should be large enough to cover a relevant part of the set of words in order for KV's algorithms to be robust. The size of the extracted DFA can be further reduced by returning an intermediate memorized DFA which is enough close to the DFA returned by the KV's algorithm.

In order to understand why KV's algorithm is robust w.r.t. random noise we have undertaken a theoretical study establishing that almost surely the language of the noisy device ($\mathcal{M}_\mathcal{N}$) for classification noise, i.e., case (1) and, with a further weak assumption on DFA, also for instance noise, i.e., case (2) is not recursively enumerable. Considering non-recursively enumerable languages as unstructured, this means that due to the noise, the (regular) structure of $\mathcal{A}$ vanishes. This is not the case for the counter automaton setting. Altogether, to put it bluntly: the less structure the noisy device has, the better KV's algorithm works.

Organization. In Section 1, we introduce the technical background required for the robustness analysis. In Section 2, we detail the goals and the settings of our analysis. In Section 3,

we provide and discuss the experimental results. In Section 4, we discuss randomness versus structure. Finally in Section 5, we draw our the conclusions and identify future work.

1. Preliminaries

Here we provide the technical background required for the robustness analysis.

Languages. Let Σ be an alphabet, i.e., a nonempty finite set, whose elements are called *letters*. A *word* w over Σ is a finite sequence over Σ, whose length is denoted by $|w|$. The unique word of length 0 is called the *empty word* and denoted by λ. As usual, Σ^* is the set of all words over Σ, and $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$ is the set of words of positive length. A *language* (over Σ) is any set $L \subseteq \Sigma^*$. The symmetric difference of languages $L_1, L_2 \subseteq \Sigma^*$ is defined as $L_1 \Delta L_2 = (L_1 \setminus L_2) \cup (L_2 \setminus L_1)$.

Words distribution and measure of a language. A distribution D over Σ^* is defined by a mapping $\mathbf{Pr}_D$ from Σ^* to $[0,1]$ such that $\sum_{w \in \Sigma^*} \mathbf{Pr}_D(w) = 1$. Let L be a language. Its *probabilistic measure* w.r.t. D, $\mathbf{Pr}_D(L)$ is defined by $\mathbf{Pr}_D(L) = \sum_{w \in L} \mathbf{Pr}_D(w)$, i.e., the sum of the probabilities for all words in L.

Our analysis requires that we are able to efficiently sample a word according to some distribution D. Thus we only consider distributions D_μ with $\mu \in \,]0,1[$, that are defined for a word $w = a_1 \ldots a_n \in \Sigma^*$ by

$$\mathbf{Pr}_{D_\mu}(w) = \mu \left(\frac{1-\mu}{|\Sigma|} \right)^n .$$

To sample a random word according to D_μ in practice, we start with the empty word and iteratively we flip a biased coin with probability $1 - \mu$ to add a letter (and μ to return the current word) and then uniformly select the letter in Σ.

Language distance. Given two languages L_1 and L_2, their distance w.r.t. a distribution D, $d_D(L_1, L_2)$, is defined by $d_D(L_1, L_2) = \mathbf{Pr}_D(L_1 \Delta L_2)$, the sum of the probabilities for all words that are only in L_1 or only in L_2. Computing the distance between languages is in most of the cases impossible. Fortunately whenever the membership problem for L_1 and L_2 is decidable, then using Chernoff-Hoeffding bounds [Hoe63], this distance can be statistically approximated as follows. Let $\alpha, \gamma > 0$ be an error parameter and a confidence level, respectively. Let S be a set of words sampled independently according to D, called a sampling, such that $|S| \geq \frac{\log(2/\gamma)}{2\alpha^2}$. Let $dist = \frac{|S \cap (L_1 \Delta L_2)|}{|S|}$. Then, we have

$$\mathbf{Pr}_D(|d_D(L_1, L_2) - dist| > \alpha) \ < \ \gamma .$$

Since we will not simultaneously discuss about multiple distributions, we omit the subscript D almost everywhere.

Finite Automata. A (complete) deterministic finite automaton (DFA) over Σ is a tuple $\mathcal{A} = (Q, \sigma, q_0, F)$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and $\sigma : Q \times \Sigma \to Q$ is the transition function. The transition function is inductively extended over words by $\sigma(q, \lambda) = q$ and $\sigma(q, wa) = \sigma(\sigma(q, w), a)$. The language of $\mathcal{A}$ is defined as $\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid \sigma(q_0, w) \in F\}$. A language $L \subseteq \Sigma^*$ is called *regular* if $L = \mathcal{L}(\mathcal{A})$ for some DFA $\mathcal{A}$.

KV's algorithm. Given a regular language L, KV's algorithm learns the unique mimimal DFA $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = L$ through two queries:

- membership queries: 'Does a word w belong to L?'
- equivalence queries: 'Does $\mathcal{L}(\mathcal{A}_E) = L$? and if not provide a word $w \in L\Delta\mathcal{L}(\mathcal{A}_E)$'. Here $\mathcal{A}_E$ represents an automaton synthetised by the algorithm based on currently collected membership queries.

The goal of this algorithm is to produce a DFA that recognizes a given regular language $L \subseteq \Sigma^*$. It is worth noting that while Σ is given, L is a priori unknown and can only be accessed through membership queries and equivalence queries. Let us describe in an informal way its behaviour:

(1) Initialize a data structure $Data$ to store useful information from an initial set of membership queries.
(2) Repeat the following steps:
   - synthetize a DFA based on the current $Data$ corresponding to the set of membership queries currently asked (see [KV94] for more details);
   - submit an equivalence query, for which the algorithm can return a pair consisting of a boolean value and a string as a potential counter example: either **true** with empty string or **false** with a counter example w;
   - when not equivalent and using the counter example $w \in L\Delta\mathcal{L}(\mathcal{A}_E)$, update $Data$.

The KV's algorithm ensures that after each round the number of states of the current DFA is increased by one. So the number of rounds is equal to the number of states of the minimal DFA accepting L.

PAC version of KV's algorithm. The KV's algorithm is efficient to learn the minimal DFA when both queries are available and there is no noise in the data. However, as our goal is to analyze the robustness of this algorithm against noise, the latter should be introduced into our data, which results in some noisy device that is not necessarily regular. To handle this, we resort to Probably Approximately Correct (PAC) learning framework, whose goal is to guarantee with high probability that the hypothesis will have low generalization error. Precisely, the learner can learn the concept, i.e., here DFA, given any arbitrary approximation ratio and success probability. The PAC model was later extended to treat noise [Slo95, Dec97, BEK02].

The PAC version of KV's algorithm takes as input an error parameter ε and a confidence level δ, and replaces the equivalence query by a (large enough) number of membership queries '$w \in L\Delta\mathcal{L}(\mathcal{A})$?', where the words are sampled from some distribution D unknown to the algorithm. Thus this algorithm can stop too early when all answers are negative while $L \neq \mathcal{L}(\mathcal{A})$. However due to the large number of such queries which depends on the current round r (i.e., $\lceil \frac{log(1/\delta)+(r+1)\log(2)}{\varepsilon} \rceil$) this algorithm ensures that

$$\mathbf{Pr}_D(d_D(L, \mathcal{L}(\mathcal{A})) > \varepsilon) \ < \ \delta.$$

A key observation is that the PAC version of this algorithm could be used for every language L for which the membership problem is decidable. However, in some cases, L is not necessarily a regular language, the algorithm might never stop and thus our adaptation includes a parameter $maxround$, the upper bound on the number of rounds, that ensures termination. This is formalized by Algorithm 1. Observe that if either the automaton is wrongly considered as equivalent or the maximal round is reached, then $\mathcal{A}_E$ is not necessarily minimal and so the algorithm minimizes $\mathcal{A}_E$ before returning it.

Algorithm 1: The PAC version of KV's algorithm applied on general languages

Input: L, a language unknown to the algorithm
Input: an integer $maxround$ ensuring termination

Data: an integer r, a boolean b, a data structure $Data$ and a DFA $\mathcal{A}_E$
Output: a DFA

`Initialize`$(Data)$
$r \leftarrow 0$

`// The control of` $maxround$ `is unnecessary when` L `is regular`
while $r < maxround$ **do**
    $\mathcal{A}_E \leftarrow$ `Synthetize`$(Data)$
    $(b, w) \leftarrow$ `IsEquivalent`$(\mathcal{A}_E)$
    if b **then return** `Minimize`$(\mathcal{A}_E)$
    `Update`$(Data, w)$
    $r \leftarrow r + 1$
end
return `Minimize`(`Synthesize`$(Data)$)

---

2. Robustness Analysis against Noises

2.1. **Principle and goals of the robustness analysis.**

Principle of the analysis. Figure 1 illustrates the process of our robustness analysis with respect to different types of noises that will be studied in this paper. First we set the qualitative and quantitative nature of the noise ($\mathcal{N}$). Then we generate a set of random DFA ($\mathcal{A}$). Combining $\mathcal{A}$ and $\mathcal{N}$, one gets a noisy model $\mathcal{M}_\mathcal{N}$. More precisely, depending on whether the noise is random or not, $\mathcal{M}_\mathcal{N}$ is either generated off-line (deterministic noise) or on-line (random noise) when a membership query is asked during the execution of KV's algorithm. Finally we compare (1) the distances between $\mathcal{A}$ and $\mathcal{M}_\mathcal{N}$, and (2) between $\mathcal{A}$ and $\mathcal{A}_E$, the automaton returned by the algorithm. The aim of this comparison is to establish whether $\mathcal{A}_E$ is closer to $\mathcal{A}$ than $\mathcal{M}_\mathcal{N}$. In order to get a quantitative measure, we define the *information gain* as:

$$\text{Information gain} = \frac{d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{M}_\mathcal{N}))}{d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}_E))}$$

We consider a low information gain to be in $[0, 0.9)$, a medium information gain to be in $[0.9, 1.5)$, and a high information gain to be in $[1.5, \infty)$. After preliminary experiments, these thresholds seem to appropriately partition the results into three subsets of relevant sizes. The higher the information gain is, the closer $\mathcal{A}_E$ is to $\mathcal{A}$ than $\mathcal{M}_\mathcal{N}$.

In addition, we also evaluate the distance between $\mathcal{A}_E$ and $\mathcal{M}_\mathcal{N}$ in order to study in which cases the algorithm learns in fact the noisy device instead of the original DFA.

Goals of the analysis.

- **Quantitative analysis.** The information gain highly depends on the 'quantity' of the noise, which is also called error rate. So we analyze the information gain depending of the

distance between the original DFA and the noisy device and want to identify a threshold (if any) where the information gain starts to significantly increase.

- **Qualitative analysis.** Another important criterion of the information gain is the 'nature' of the noise. So we analyze the information gain w.r.t. the different noisy devices that we have introduced.

- **Impact of word distribution.** Then, the robustness of the KV's algorithm with respect to word distribution is also analyzed.

In order to perform relevant experiments, one needs to tune two critical parameters of KV's algorithm. Since the running time of the algorithm quadratically depends on the number of rounds (i.e. iterations of the loop), selecting an appropriate *maximal number of rounds* is a critical issue. We vary this maximal number of rounds and analyze how the information gain decreases w.r.t. this number. On the other hand, as an equivalence query is replaced with a set of membership queries whose number depends on the current round and the pair (ε, δ), it is thus interesting to study (1) what is the effect of *accuracy of the approximate equivalence queries*, i.e., the values of (ε, δ) on the ratio of executions that reach the maximal number of rounds and (2) compare the information gain for executions that stop before reaching this maximal number and the same execution when letting it run up to this maximal number.

2.2. **Noise.** A *random language* $R \subseteq \Sigma^*$ is determined by a random process: for each $w \in \Sigma^*$, membership $w \in R$ is determined independently at random, *once and for all*, according to some probability $\mathbf{Pr}(w \in R) \in [0, 1]$. The probability $\mathbf{Pr}(w \in R)$ may depend on some parameters such as w itself and a given DFA.

We now describe the four types of noise that we analyze in this paper. Each type adds noise to a given DFA $\mathcal{A}$ in form of a random language R. For the first two types, *noise with output* and *noise with input*, the probability $\mathbf{Pr}(w \in R)$ of including $w \in \Sigma^*$ in R depends on w itself, $\mathcal{L}(\mathcal{A})$, and some parameter $0 < p < 1$. The third kind of noise, *counter DFA*, is actually *deterministic*, i.e., $\mathbf{Pr}(w \in R) \in \{0, 1\}$ for all $w \in \Sigma^*$. In that case, the given DFA $\mathcal{A}$ determines a unique "noisy" language. Let us be more precise:

DFA with noisy output. Given a DFA $\mathcal{A}$ over the alphabet Σ and $0 < p < 1$, the random language $\mathcal{L}(\mathcal{A}^{\to p})$ flips the classification of words w.r.t. $\mathcal{L}(\mathcal{A})$ with probability p. More formally, for all $w \in \Sigma^*$,

$$\mathbf{Pr}(w \in \mathcal{L}(\mathcal{A}^{\to p})) = (1 - p)\mathbf{1}_{w \in \mathcal{L}(\mathcal{A})} + p\mathbf{1}_{w \notin \mathcal{L}(\mathcal{A})}$$

where $\mathbf{1}_C$ is 1 if condition C holds, and 0 otherwise. Observe that the expected value of the distance $d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}^{\to p}))$ is p. Moreover, in our experiments, we observe that $\left|\frac{d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}^{\to p})) - p}{p}\right| < 5 \cdot 10^{-2}$ for all the generated languages.

DFA with noisy input. Given a DFA $\mathcal{A}$ over the alphabet Σ (with $|\Sigma| > 1$) and $0 < p < 1$, the random language $\mathcal{L}(\mathcal{A}^{\leftarrow p})$ changes every letter of the word with probability p uniformly to another letter and then returns the classification of the new word w.r.t. $\mathcal{L}(\mathcal{A})$. More

formally, for $w = a_1 \ldots a_n \in \Sigma^*$,

$$\mathbf{Pr}(w \in \mathcal{L}(\mathcal{A}^{\leftarrow p})) = \sum_{\substack{w' = b_1 \ldots b_n \in \mathcal{L}(\mathcal{A}) \\ \text{s.t. } |w| = |w'|}} \prod_{1 \leq i \leq n} \left((1 - p)\mathbf{1}_{a_i = b_i} + \frac{p}{|\Sigma| - 1}\mathbf{1}_{a_i \neq b_i} \right).$$

Counter DFA. Let $\mathcal{A}$ be a DFA over the alphabet Σ and $c : \Sigma \cup \{\lambda\} \to \mathbb{Z}$ be a function. We inductively define the function $\bar{c} : \Sigma^* \to \mathbb{Z}$ by

$$\bar{c}(\lambda) = c(\lambda) \text{ and } \bar{c}(wa) = \bar{c}(w) + c(a).$$

The counter language $\mathcal{L}(\mathcal{A}_c)$ is now given as the union of the language of $\mathcal{A}$ and the set of words whose value of function $\bar{c}$ is nonpositive.

$$\mathcal{L}(\mathcal{A}_c) = \mathcal{L}(\mathcal{A}) \cup \{w \in \Sigma^* \mid \bar{c}(w) \leq 0\}.$$

DFA with pathological behaviours. The forth device that we want to learn is a DFA $\mathcal{A}$ viewed as a formal model for a protocol that should be followed by users in some institution (hospital, university, etc.). We only consider $\mathcal{A}$ for which there is a word denoted $w_\mathcal{A}$ such that $w_\mathcal{A} \Sigma^* \cap \mathcal{L}(\mathcal{A}) = \emptyset$. This is a reasonable assumption for a realistic protocol for which after a specific *pathological* sequence of actions (here $w_\mathcal{A}$), one knows that the user cannot succeed.

The observed language generated by the noisy (random) device, denoted here $\mathcal{A}^n$, is obtained as follows:

- $\mathcal{L}(\mathcal{A}^n)$ includes $\mathcal{L}(\mathcal{A})$. The observed language must contain all correct behaviours w.r.t. the protocol;
- For every word $w \in w_\mathcal{A} \Sigma^*$, $w \in \mathcal{L}(\mathcal{A}^n)$ with probability $\frac{1}{2}$. Every pathological behaviour may equally be observed or not;
- For every word $w \notin \mathcal{L}(\mathcal{A}) \uplus w_\mathcal{A} \Sigma^*$, $w \notin \mathcal{L}(\mathcal{A}^n)$. One does not observe any behaviour which is neither correct nor pathological.

3. Experimental Evaluation

In order to empirically evaluate our ideas, we have implemented a prototype and benchmarks in Python, using the NumPy library. They are available on Zenodo[1]. All evaluations were performed on a computer equipped by Intel i5-8250U CPU with 4 cores, 16GB of memory and Ubuntu Linux 18.03.

3.1. **Generating DFAs.** We now describe the settings of the experiments we made with four different types of noises. We choose $\mu = 10^{-2}$ for the parameter of the word distribution so that the average length of a random word is 99. All the statistic distances were computed using the Chernoff-Hoeffding bound [Hoe63] with $\alpha = 5 \cdot 10^{-4}$ as error parameter and $\gamma = 10^{-3}$ as confidence level.

The benchmarks were performed on DFA randomly generated using the following procedure. Let $M_q = 60$ and $M_a = 20$ be two parameters, which impose upper bounds on the number of states and the size of the alphabet, that could be tuned in future experiments. The DFA $\mathcal{A} = (Q, \sigma, q_0, F)$ on Σ is generated as follows:

---

[1]https://doi.org/10.5281/zenodo.8031255

- Uniformly choose $n_q \in [20, M_q]$ and $n_a \in [3, M_a]$;
- Set $Q = [0, n_q]$ and $\Sigma = [0, n_a]$;
- Uniformly choose $n_f \in [0, n_q - 1]$ and let $F = [0, n_f]$;
- Uniformly choose q_0 in Q;
- For all $(q, a) \in Q \times \Sigma$, choose the target state $\sigma(q, a)$ uniformly among all states.

The choice of M_q and M_a was inspired by observing that these values often occur when modeling realistic processes like in business process management.

3.2. **Tunings.** Before launching our experiments, we first tune two key parameters for both efficiency and accuracy purposes: the maximal number of rounds of the algorithm and the value of ε and of δ for the accuracy of the approximate equivalence query. For the sake of simplicity, this tuning is based on experiments over the DFA with the noisy output. The reason is that the expected distance between the DFA and the noisy device is known and can be controlled (p).

Maximal number of rounds. In order to specify a maximal number of rounds that lead to a good performance of the KV's Algorithm, we took a DFA with noisy output $\mathcal{A}^{\rightarrow p}$ for $p \in \{0.005, 0.0025, 0.0015, 0.001\}$. We ran the learning algorithm, stopping every 20 rounds to estimate the distance between the current DFA $\mathcal{A}_E$ to the original DFA $\mathcal{A}$. Figure 2 shows the evolution graphs of $d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}_E))$ w.r.t. the number of rounds according to the different values of p, each of them summarizing five runs on five different DFAs. The vertical axis corresponds to the distance to original DFA $\mathcal{A}$, and the horizontal axis corresponds to the number of rounds. The red line is the distance with $\mathcal{A}^{\rightarrow p}$, and the blue line is the distance with $\mathcal{A}_E$.

We observe that after about 250 rounds, the distance $d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}_E))$ is stabilizing except some rare peaks, which are worth further investigation. Therefore, from now on all the experiments are made with a maximum of 250 rounds. Of course this number depends on the size of $\mathcal{A}$. However, for the variable size that we have chosen (between 10 and 50 states), it seems to be a good choice.

Accuracy of the approximate equivalence query. We have generated thirty-five DFA, and for each of them we generated five $\mathcal{A}^{\rightarrow p}$ with different values of p. Table 1 summarizes our results with different ε and δ for the approximate equivalence query. The rows correspond to the value of the noise p, the columns correspond to the values of ε and δ (where we always choose $\varepsilon = \delta$) and each cell shows the average information gain. From this table, $\varepsilon = \delta = 0.01$ and $\varepsilon = \delta = 0.005$ seem to be optimal values. We decided to fix $\varepsilon = \delta = 0.005$ for all our experiments.

3.3. **Qualitative and Quantitative analysis.** For the four types of noise, we have generated numerous DFA (as described shortly above). Furthermore, for the noisy outputs and inputs given a randomly generated DFA, we have constructed several noisy devices depending on the 'quantity' of noise. By computing the (average) information gain for all these experiments, we have been able to get conclusions about the effect of the nature and the quantity of the noise on the performance of KV's algorithm.
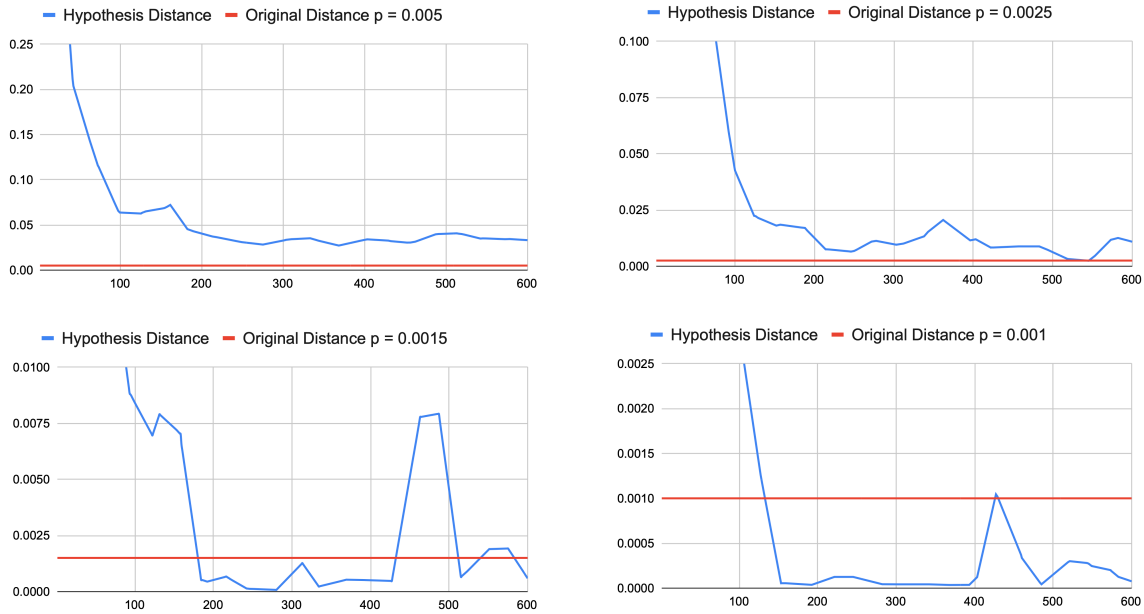
FIGURE 2. Number of rounds analysis

$\varepsilon = \delta$ p	0.05	0.01	0.005	0.001	0.0005
0.01	0.081	0.054	0.047	0.048	0.050
0.005	0.086	0.087	0.072	0.070	0.094
0.0025	0.867	0.292	0.591	0.321	0.748
0.0015	1.401	2.933	3.082	0.980	0.710
0.001	5.334	4.524	3.594	1.811	6.440

TABLE 1. Evaluation of the impact of ε and δ.

When KV's algorithm is applied to a device with random noise, a corresponding random language is generated on-the-fly: once membership of a word in the target language has been determined (e.g., through a membership query), the corresponding truth value is stored and not changed anymore.

DFA with noisy output. We have generated fifty DFA, and for each such DFA $\mathcal{A}$, we have generated random languages with noisy output $\mathcal{L}(\mathcal{A}^{\to p})$ with five values for p between 0.01 and 0.001. Table 2 summarizes the results. Recall that the expected value of $d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}^{\to p}))$ is p. We have identified a threshold for p around 0.0025: if the noise is higher than 0.0025, the resulting DFA $\mathcal{A}_E$ has a bigger distance to the original one $\mathcal{A}$ than to the corresponding noisy device $\mathcal{A}^{\to p}$, and smaller otherwise. Moreover, once we cross the threshold the robustness of the algorithm increases very quickly. We have also included a

column that represents the standard *deviation* of the random variable $d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}_E))$ to assess that our conclusions are robust w.r.t. the probabilistic feature.

p	$d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}_E))$	$d(\mathcal{L}(\mathcal{A}^{\rightarrow p}), \mathcal{L}(\mathcal{A}_E))$	gain	deviation
0.01	0.12625	0.13320	0.07432	0.04102
0.005	0.04420	0.04827	0.11312	0.03366
0.0025	0.00333	0.00568	0.75031	0.00523
0.0015	0.00027	0.00174	5.52999	0.00047
0.001	0.00006	0.00103	15.75817	0.00007

Table 2. Evaluation of the algorithm w.r.t. the noisy output.

DFA with noisy input. We have generated forty-five random DFA, and for each such DFA $\mathcal{A}$, we have generated random languages with noisy input $\mathcal{L}(\mathcal{A}^{\leftarrow p})$ by choosing $p \in \{10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 5 \cdot 10^{-3}\}$. Contrary to the case of noisy output, p does not correspond to the expected value of $d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}^{\leftarrow p}))$. Thus we need to calculate this distance for every pair of the experiments. Thus, we have gathered the pairs whose distances belong to intervals that are described in the first column of Table 3. The second column of this table reports the number of pairs in the interval while the third one presents the average value of the distance for the set of the corresponding pairs. Again we identify a threshold for $d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}^{\leftarrow p}))$ between 0.001 and 0.005. Once we cross the threshold, the robustness of the algorithm increases very quickly.

Range	#	$d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}^{\leftarrow p}))$	$d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}_E))$	$d(\mathcal{L}\mathcal{A}^{\leftarrow p}), \mathcal{L}(\mathcal{A}_E))$	gain	deviation
$[0.025, 1]$	36	0.04027	0.21513	0.22658	0.18	0.05279
$[0.005, 0.025]$	53	0.00924	0.05416	0.06077	0.17	0.04172
$[0.002, 0.005]$	33	0.00378	0.01260	0.01611	0.30	0.01783
$[0.001, 0.002]$	11	0.00123	0.00030	0.00154	4.1	0.00058
$[0.0005, 0.001]$	25	0.00079	0.00002	0.00082	39.5	0.00007

Table 3. Evaluation of the algorithm w.r.t. the noisy input.

Counter DFA. We have randomly generated the counter function as follows: We have uniformly chosen $c(\lambda)$ in $[0, |\Sigma|]$. Then, for all $a \in \Sigma$, $\mathbf{Pr}(c(a) = -1) = \frac{1}{4}$ and for all $0 \leq i \leq 6$, $\mathbf{Pr}(c(a) = i) = \frac{3}{28}$.

We have generated 160 DFA. For each of them, we have generated a counter automaton (as described before). The results of our experiments are given in Table 4, from which we can see that whatever the quantity of noise, the KV's algorithm is unable to get closer to the original DFA. Moreover the extracted DFA $\mathcal{A}_E$ is very often closer to the counter automaton $\mathcal{A}_c$ than the original DFA $\mathcal{A}$.

Range	#	$d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}_c))$	$d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}_E))$	$d(\mathcal{L}(\mathcal{A}_c), \mathcal{L}(\mathcal{A}_E))$	gain	deviation
$[0.005, 0.025]$	14	0.01238	0.02586	0.02053	0.47886	0.01898
$[0.002, 0.005]$	57	0.00245	0.00396	0.00262	0.61765	0.00298
$[0.001, 0.002]$	22	0.00143	0.00209	0.00121	0.68156	0.00126
$[0.0005, 0.001]$	20	0.00079	0.00108	0.00064	0.72481	0.00065
$[0.0001, 0.0005]$	44	0.00025	0.00035	0.00021	0.71054	0.00021

TABLE 4. Evaluation of the algorithm w.r.t. the 'noisy' counter.

Thus we conjecture that when the noise is 'unstructured' and the quantity is small enough such that the word noise is still meaningful, then KV's algorithm is robust against such noise. On the contrary when the noise is structured, then KV's algorithm 'tries to learn' the noisy device whatever the quantity of noise. In Section 4, we will strengthen this conjecture establishing that in some sense, noise produced by random process implies unstructured noise.

3.4. **DFA with pathological behaviours.** Let $\mathcal{A}^+$ be a DFA such that $\mathcal{L}(\mathcal{A}^+) = \mathcal{L}(\mathcal{A}) \uplus w_{\mathcal{A}}\Sigma^*$. Theoretically, $d(\mathcal{L}(\mathcal{A}^n), \mathcal{L}(\mathcal{A}^+))$ is on average equal to $d(\mathcal{L}(\mathcal{A}^n), \mathcal{L}(\mathcal{A}))$. Thus a priori, the KV's algorithm applied on $\mathcal{A}^n$ should not modify this property since $\mathcal{A}$ and $\mathcal{A}^+$ are DFA. However due to the particular shape of the pathological behaviours, it could be not the case. So the goal of studying this device is to detect the general tendancy: is the output DFA closer to $\mathcal{A}$ or to $\mathcal{A}^+$? In order to be relevant, $\mathbf{Pr}_d(w_{\mathcal{A}}\Sigma^*) = \left(\frac{1-\mu}{|\Sigma|}\right)^{|w_{\mathcal{A}}|}$ should be sufficiently small.

DFA construction. We now explain how to randomly generate a DFA $\mathcal{A}$ such that there exists a word $w_{\mathcal{A}}$ fulfilling $w_{\mathcal{A}}\Sigma^* \cap \mathcal{L}(\mathcal{A}) = \emptyset$. Since we do not care about the letters of $w_{\mathcal{A}}$, we choose $w_{\mathcal{A}} = a^3$ and randomly select $|\Sigma|$ with $5 \leq |\Sigma| \leq 20$ which leads to $\mathbf{Pr}_d(w_{\mathcal{A}}\Sigma^*) \leq \frac{1}{125} \leq 0.008$. As for the other parameters, we adopt the same as described before, for instance the number of states is between 20 and 60.

(1) We first create states $q_0, q_1, q_2 \in Q$ and $q_\perp \in Q \setminus F$ with $\sigma(q_0, a) = q_1$, $\sigma(q_1, a) = q_2$, $\sigma(q_2, a) = q_\perp$, and for all $b \in \Sigma$, $\sigma(q_\perp, b) = q_\perp$;
(2) Then we proceed as in section 3.1 for achieving the construction of the rest of $\mathcal{A}$ but excluding $\{q_0, q_1, q_2, q_\perp\}$ to be the destinations of a transition.

Such a DFA $\mathcal{A}$ is depicted in Figure 3 with $\Sigma = \{a, b\}$, for the sake of simplicity. $\mathcal{A}^+$ is equal to $\mathcal{A}$ except that $q_\perp$ is now accepting.

Experimental results. We have constructed and run about 300 benchmarks, each one being a pair of $(\mathcal{A}, \mathcal{A}^+)$. In this context, the information gain is defined as follows.

$$\text{Information gain} = \frac{d(\mathcal{L}(\mathcal{A}^+), \mathcal{L}(\mathcal{A}^n))}{d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}^n))}$$
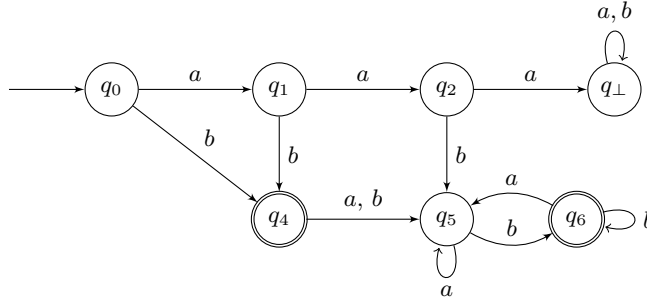
FIGURE 3. A DFA $\mathcal{A}$ where $a^3\Sigma^* \cap \mathcal{L}(\mathcal{A}) = \emptyset$.

The goal of our experiments is to check whether the gain is strictly bigger than 1 or not. Different from Section 2.1, we only consider here a **low** information gain when it is less than one and a **high** one when the information gain otherwise.

Range	#	$d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}^+))$	$d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}^n))$	$d(\mathcal{L}(\mathcal{A}^+)), \mathcal{L}(\mathcal{A}^n))$	gain	deviation
$[0.005, 0.025]$	37	0.00800	0.00291	0.00382	1.31271	0.00010
$[0.002, 0.005]$	61	0.00384	0.00144	0.00177	1.22917	0.00033
$[0.001, 0.002]$	84	0.00141	0.00055	0.00065	1.18182	0.00016
$[0.0005, 0.001]$	59	0.00067	0.00026	0.00033	1.26923	0.00003
$[0.00005, 0.0005]$	59	0.00041	0.00016	0.00019	1.18750	0.00002

TABLE 5. Evaluation of the algorithm w.r.t. the elimination of pathological behaviours.

The experimental results are shown in Table 5, where each row represents the results of all pairs of $\mathcal{A}$ and $\mathcal{A}^+$ whose distance inside the range depicted in the first column. Interestingly, the language of the learned DFA is always closer to the one of $\mathcal{A}$ than the one of $\mathcal{A}^+$, and this for all distances between $\mathcal{A}$ and $\mathcal{A}^+$. However as could be expected, since $\mathcal{A}^+$ is a DFA, the information gain remains close to 1. Hence, our experimentation can be seen as an evidence of capacity of the KV's algorithm to partially eliminate pathological behaviours.

3.5. **Words distribution.** We now discuss the impact of word distribution on the robustness of the KV's algorithm. The parameter μ determines the average length of a random word $(\frac{1}{\mu} - 1)$. Table 6 summarizes experimental results with different values of μ indicated on the first row. The other rows correspond to different values of the noise p for $\mathcal{A}^{\rightarrow p}$. The cells (at the intersection of a pair (p, μ)) contain the (average) information gain, where experiments have been done over twenty-two DFA. Note that the worst and best cases are always eliminated to avoid that the pathological cases perturb the average values. For values of p that matter (i.e., when the gain is greater than 1), there is clear tendency for the gain to first increase w.r.t. μ, reaching a maximum about $\mu = 0.01$ the value that we have chosen and then decrease. A possible explanation would be the following: too short words (i.e., big

μ) does not help to discriminate between languages while too long words (i.e., small μ) lead to overfitting and does not reduce the noise.

p \ μ	0.001	0.005	0.01	0.05	0.1
0.01	0.059	0.067	0.078	0.184	0.317
0.005	0.078	0.130	0.134	0.559	0.966
0.0025	0.165	0.298	0.398	1.246	0.823
0.0015	0.465	0.671	2.267	2.074	1.651
0.001	1.801	10.94	8.907	3.753	2.341

TABLE 6. Analysis of different distributions on Σ^*

3.6. **Reduction of the size of the DFA.** Up to now the extracted DFA is obtained by using as exit condition that either the maximal number of rounds is reached or the current learned automaton is declared equivalent. In practice, since the language of the noisy device is generally not regular, the current automaton of the algorithm is rarely declared equivalent and thus the final $\mathcal{A}_E$ is very often returned after running the maximal number of rounds. In other words, after the maximal number of rounds, chosen 250 here, the (minimal) extracted DFA has normally a larger size than the original one, on average about twice the original one. In view of eliminating over-fitting, we want to reduce the size of the returned DFA. So we proceed as follows.

- Every 10 rounds, we memorize the current automaton in an array **TA**;
- When we exit the while loop, we compute the measure of $\mathcal{L}(\mathcal{A}_E)$, denoted $m = \mathbf{Pr}_D(\mathcal{L}(\mathcal{A}_E))$ and define a threshold which is $c * m$, where we choose $c = 10^{-3}$;
- Then we examine the saved DFAs by increasing size and select the first one whose distance from $\mathcal{A}_E$ is smaller than the threshold and return it (after a minimization);
- Otherwise we return $\mathcal{A}_E$ (after a minimization).

This is formalized by Algorithm 2.

Table 7 shows the experimental results in the following setting, where we randomly generated sixty DFA as before.

- the size of the sixty DFA is randomly selected between 20 and 60;
- we only consider DFA with noisy outputs with the parameter p between 10^{-2} and 10^{-3}.

In this table, $\widehat{\mathcal{A}_E}$ is the DFA returned by Algorithm 2 while $\mathcal{A}_E$ is the DFA returned by Algorithm 1. Similarly, $\widehat{gain}$ and $gain$ are the information gains corresponding to Algorithm 2 and Algorithm 1, respectively. We first observe that the information gains are very close whose better one can equally be $\widehat{gain}$ or $gain$. As long as the information gain is less than one, the size reduction is not significant while as soon as information gain is more than one, the size decreases by about $\frac{1}{3}$. So we conclude that this adaptation is useful when the KV's algorithm performs well, i.e., robust against the introduced noise.

We also performed experimentations with a threshold equal to $0.1m$, $0.01m$, and $0.005m$ not presented in this table. The results establish that this reduces more the size of the DFA

Algorithm 2: Modified KV's algorithm for size reduction of the DFA

Input: L, a language unknown to the algorithm
Input: an integer $maxround$ ensuring termination

Data: integers r, $rfinal$ and $period$, a boolean b, and a data structure $Data$
Output: a DFA

`Initialize`$(Data)$
$r \leftarrow 0$
$rfinal \leftarrow maxround$

// The control of $maxround$ is unnecessary when L is regular
while $r < maxround$ **do**
     $\mathcal{A}_E \leftarrow$ `Synthetize`$(Data)$
     $(b, w) \leftarrow$ `IsEquivalent`$(\mathcal{A}_E)$
     if b **then**
         $rfinal \leftarrow r$
         break
     `Update`$(Data, w)$
     $r \leftarrow r + 1$
     if $r \% period == 0$ **then**
         $\mathbf{TA}[r/period] \leftarrow \mathcal{A}_E$
end
$i \leftarrow 1$
$threshold \leftarrow c * \mathbf{Pr}_D(\mathcal{L}(\mathcal{A}_E))$
repeat
     if $d(\mathcal{L}(TA[i]), \mathcal{L}(\mathcal{A}_E)) \leq threshold$ **then**
         $rfinal \leftarrow r$
         return `Minimize`$(\mathbf{TA}[i])$
     else $i \leftarrow i + 1$
until $i * period > rfinal$
return `Minimize`$(\mathcal{A}_E)$

---

but that the information gain is considerably worse than for Algorithm 1 and most of the time less than one.

p	$|\widehat{\mathcal{A}}_E|$	$|\mathcal{A}_E|$	$|\widehat{\mathcal{A}}_E|/|\mathcal{A}_E|$	$\widehat{gain}$	$gain$
0.01	182.480	182.940	0.99749	0.11326	0.11425
0.005	137.040	141.700	0.96711	0.50633	0.49839
0.0025	68.380	94.500	0.72360	2.64267	2.62994
0.0015	48.840	77.480	0.63036	3.33972	3.44385
0.001	44.600	66.980	0.66587	6.81623	6.49229

TABLE 7. Comparison between Algorithms 1 and 2.

4. Random languages versus structured languages

Recall that in the precedent section, from the experimental results, we conjecture that KV's algorithm is robust, when the noise is random, i.e., unstructured, and its quantity is small enough, such as for DFA with noisy output and with noisy input. This is however not the case for structured counter DFA, for which KV's algorithm learns the noisy device itself instead of the original one whatever the quantity of noise.

In this section, we want to theoretically establish that the main factor of the robustness of the KV's algorithm w.r.t. random noise is that almost surely randomness, in most cases, yields the perturbated language that is unstructured. We consider a language as structured if it can be produced by some general device. Thus we identify the family of structured languages with the family of recursively enumerable languages. More precisely, we show that almost surely DFA with noisy output leads to a language that is not recursively enumerable. We then demonstrate further that with a mild condition, almost surely DFA with noisy input yields also non-recursively enumerable language. As for the counter DFA, by definition, it is clearly recursively enumerable, thus not being studied further.

The following lemma gives a simple means to establish that almost surely a random language is not recursively enumerable.

Lemma 4.1. *Let R be a random language over Σ. Let $(w_n)_{n \in \mathbb{N}}$ be a sequence of words of Σ^*. Let $W_n = \{w_i\}_{i<n}$ and $\rho_n = \max_{W \subseteq W_n} \mathbf{Pr}(R \cap W_n = W)$. Assume that $\lim_{n \to \infty} \rho_n = 0$. Then, for all countable families of languages $\mathcal{F}$, almost surely $R \notin \mathcal{F}$. In particular, almost surely R is not a recursively enumerable language.*

Proof. Let us consider an arbitrary language L. Then, for all n,

$$\mathbf{Pr}(R = L) \leq \mathbf{Pr}(R \cap W_n = L \cap W_n) \leq \rho_n.$$

Thus, $\mathbf{Pr}(R = L) = 0$ and $\mathbf{Pr}(R \in \mathcal{F}) = \sum_{L \in \mathcal{F}} \mathbf{Pr}(R = L) = 0$. ☐

From Lemma 4.1, we immediately obtain that almost surely the noisy output perturbation of any language is not recursively enumerable. The proofs of the two next theorems use the same notations as those given in Lemma 4.1.

Theorem 4.2. *Let L be a language and $0 < p < 1$. Then almost surely $L^{\to p}$ is not a recursively enumerable language.*

Proof. Consider any enumeration $(w_n)_{n \in \mathbb{N}}$ of Σ^* and any $W \subseteq W_n$. The probability that $L^{\to p} \cap W_n$ is equal to W is bounded by $\max(p, 1 - p)^n$. Thus, $\rho_n \leq \max(p, 1 - p)^n$ and $\lim_{n \to \infty} \rho_n = 0$. ☐

We cannot get a similar result for the noisy input perturbation. Indeed consider the language Σ^*, whatever the kind of noise brought to the input, the obtained language is still Σ^*. With the kind of input noise that we study, consider the language that accepts words of odd length (see the automaton $\mathcal{A}'$ of Figure 4). Then the perturbed language with noisy input is unchanged, i.e., $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A}'^{\leftarrow p})$.

However given a DFA $\mathcal{A}$, we can establish a mild condition on $\mathcal{A}$ ensuring that almost surely the random language $\mathcal{L}(\mathcal{A}^{\leftarrow p})$ is not recursively enumerable. To this end, we now recall Markov chains with some important relative notions.

Notation. Let Q be a finite set of states. Then $Dist(Q) = \{\Delta : Q \to \mathbb{Q}_{\geq 0} \mid \sum_{q \in Q} \Delta(q) = 1\}$ is the set of *rational distributions* over Q.

Definition 4.3 (Markov chain). A *finite Markov chain* is a tuple $\mathcal{M} = (Q, P)$ where:
- Q is a finite set of states;

- P is the transition function from Q to $Dist(Q)$ with $P(q)(q')$ also denoted $P(q, q')$.

Definition 4.4 (Irreducibility and Periodicity). Let $\mathcal{M} = (Q, P)$ be a finite Markov chain. Then:
- $\mathcal{M}$ is *irreducible* if for any two states $q, q' \in Q$, there exists $n \in \mathbb{N}$ such that $P^n(q, q') > 0$;

- Assume that $\mathcal{M}$ is irreducible and pick some $q_0 \in Q$. Then the periodicicity of $\mathcal{M}$ denoted $period(\mathcal{M})$ is defined by $period(\mathcal{M}) = \gcd(\{n > 0 \mid P^n(q_0, q_0) > 0\})$ (which is in fact independent of q_0).

Notation. As usual in the context of graphs, a bottom strongly connected component will be denoted by a BSCC.

We are now ready to present a mild condition on a DFA $\mathcal{A}$ such that almost surely the random language $\mathcal{L}(\mathcal{A}^{\leftarrow p})$ is not recursively enumerable.

Definition 4.5 (equal-length-distinguishing DFA). Let $\mathcal{A} = (Q, F, \sigma, q_0)$ be a DFA. We call $\mathcal{A}$ *equal-length-distinguishing* if there exist (possibly identical) BSCC $\mathcal{C}, \mathcal{C}'$ of $\mathcal{A}$, $q_1 \in \mathcal{C} \cap F$, $q_1' \in \mathcal{C}' \setminus F$, and $w, w' \in \Sigma^*$ such that we have $q_1 = \sigma(q_0, w)$, $q_1' = \sigma(q_0, w')$, and $|w| = |w'|$.

Theorem 4.6. *Let Σ be an alphabet with $|\Sigma| > 1$. Let $\mathcal{A} = (Q, \sigma, q_0, F)$ be a DFA over Σ, $0 < p < 1$ and $\mathcal{C}, \mathcal{C}'$ some BSCC of $\mathcal{A}$ (possibly equal). Assume that $\mathcal{A}$ is equal-length-distinguishing. Then almost surely $\mathcal{L}(\mathcal{A}^{\leftarrow p})$ is not a recursively enumerable language.*

Proof. Let us denote $\ell = |w|$ and let $a \in \Sigma$. We build a Markov chain $\mathcal{M}$ from $\mathcal{C}$ as follows: every transition $q \xrightarrow{a} q'$ has probability $1 - p$ and for all $b \neq a$, every transition $q \xrightarrow{b} q'$ has probability $\frac{p}{|\Sigma|-1}$. We proceed similarly from $\mathcal{C}'$ to build $\mathcal{M}'$. We denote $m = period(\mathcal{M})$ and $m' = period(\mathcal{M}')$ in the following.

Let us denote α_n (resp. α_n') the probability in $\mathcal{M}$ (resp. $\mathcal{M}'$) that starting from q_1 (resp. q_1'), the current state at time n is q_1 (resp. q_1'). For the sake of simplicity, we reuse q_1 and q_1' as in Definition 4.5. Since $\mathcal{M}$ and $\mathcal{M}'$ are irreducible with respectively periodicity m and m', $\lim_{n\to\infty} \alpha_{mn}$ (resp. $\lim_{n\to\infty} \alpha_{m'n}'$) exists and is positive. Let us denote α (resp. α') this limit. There exists n_0 such that for all $n \geq n_0$, $\alpha_{mn} \geq \frac{\alpha}{2}$ and $\alpha_{m'n}' \geq \frac{\alpha'}{2}$.

Define $w_n = wa^{mm'(n+n_0)}$ for all $n \in \mathbb{N}$. The probability that w_n is accepted by $\mathcal{L}(\mathcal{A})^{\leftarrow p}$ is lower bounded by the probability that the prefix w is unchanged (thus reaching q_1) and that after $mm'(n + n_0)$ steps the current state in $\mathcal{M}$ is q_1. Recall that q_1 is a final state. So a lower bound is: $\min(p, 1 - p)^\ell \frac{\alpha}{2}$.

The probability that w_n is rejected by $\mathcal{L}(\mathcal{A})^{\leftarrow p}$ is lower bounded by the probability that the prefix w is changed into w' (thus reaching q_1') and that after $mm'(n + n_0)$ steps the current state in $\mathcal{M}'$ is q_1'. Note that q_1' is not a final state. So a lower bound is: $\min(p, 1 - p)^\ell \frac{\alpha'}{2}$.

Let $W \subseteq W_n$. The probability that $L^{\leftarrow p} \cap W_n$ is equal to W is upper bounded by:

$$\left(1 - \min(p, 1 - p)^\ell \frac{\min(\alpha, \alpha')}{2}\right)^n$$

Thus $\rho_n \leq \left(1 - \min(p, 1 - p)^\ell \frac{\min(\alpha, \alpha')}{2}\right)^n$ and $\lim_{n\to\infty} \rho_n = 0$. $\qquad\square$

The DFA $\mathcal{A}$ of Figure 4 that represents the formula 'a Until b' of temporal logic LTL is equal-length-distinguishing. The corresponding pair of states consists of the accepting

state and the leftmost one, both constituting a BSCC with only one of them being a final state. The DFA $\mathcal{A}'$ on the right part of Figure 4 is not equal-length-distinguishing. The whole DFA is a BSCC with only one final state. There does not exist a pair of $w, w' \in \Sigma^*$ satisfying the condition of Definition 4.5. Given a DFA $\mathcal{A} = (Q, \sigma, q_0, F)$, checking whether it is equal-length-distinguishing can be done in quadratic time with the following procedure.

(1) construct a new graph G from $\mathcal{A}$
   - the set of vertices is $Q \times Q$;
   - there is an edge $(q_1, q_2) \to (q_1', q_2')$ in G if there are some transitions $q_1 \xrightarrow{a_1} q_1'$ and $q_2 \xrightarrow{a_2} q_2'$ in $\mathcal{A}$.
(2) check whether there exists at least one vertex (q_1, q_2) in some BSCC of G with $q_1 \in F$ and $q_2 \notin F$, such that it is reachable from (q_0, q_0) in G. The existence of such a vertex implies that $\mathcal{A}$ is equal-length-distinguishing.

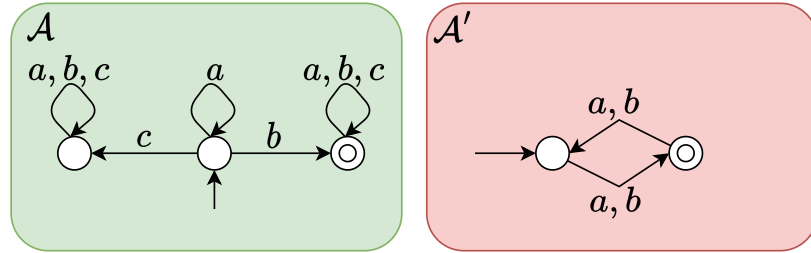The omitted correctness proof of this procedure is straightforward.

FIGURE 4. Two DFA

Since being equal-length-distinguishing is a sufficient condition for ensuring that almost surely $\mathcal{L}(\mathcal{A}^{\leftarrow p})$ is not a recursively enumerable language, we want to investigate whether it is necessary. The next proposition shows a particular case when it is the case.

Proposition 4.7. *Let Σ be an alphabet with $|\Sigma| > 1$. Let $\mathcal{A} = (Q, \sigma, q_0, F)$ be a DFA that is not equal-length-distinguishing and such that every circuit of $\mathcal{A}$ belongs to a BSCC. Then, for every sampling L' of $\mathcal{L}(\mathcal{A}^{\leftarrow p})$, L' is regular.*

Proof. Pick some $n_0 \in \mathbb{N}$ such that for all w with $|w| \geq n_0$ and $q_0 \xrightarrow{w} q$ implies that q belongs to some BSCC. Observe now that, since $\mathcal{A}$ is not equal-length-distinguishing, for words w, w' with $|w| = |w'| \geq n_0$, $w \in L$ iff $w' \in L$. Thus, for every sampling L' of $\mathcal{L}(\mathcal{A}^{\leftarrow p})$, $L' = (L' \cap \Sigma^{<n_0}) \cup (L \cap \Sigma^{\geq n_0})$ implying that L' is regular. $\qed$
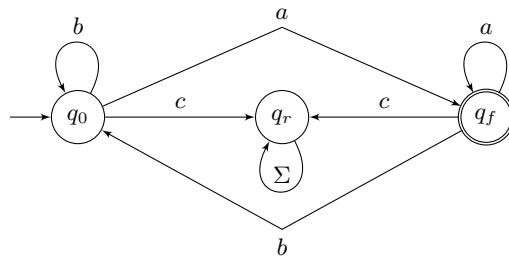

FIGURE 5. A DFA $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) = (a+b)^* a$

However in general this condition is not necessary. Observe that we establish the next proposition using a generalization of Lemma 4.1.

Proposition 4.8. *Let $\mathcal{A}$ be the DFA of Figure 5. Then, $\mathcal{A}$ is not equal-length-distinguishing while almost surely $\mathcal{L}(\mathcal{A}^{\leftarrow \frac{2}{3}})$ is not recursively enumerable.*

Proof. There is a single BSCC with a single state $\{q_r\}$. So $\mathcal{A}$ is not equal-length-distinguishing. Let $w \neq \lambda$ be a word with $|w| = n$ and denote $\tilde{w}$ the random word obtained by the noisy perturbation. Observe that every letter of $\tilde{w}$ is uniformly distributed over Σ. So the probability that $\tilde{w}$ does not contain a c is $(\frac{2}{3})^n$ and the conditional probability that $\tilde{w}$ belongs to $\mathcal{L}(\mathcal{A}^{\leftarrow \frac{2}{3}})$ knowing that it does not contain a c is $\frac{1}{2}$.

Fix some $0 < \rho < 1$. The probability that for all words $w \in \Sigma^n$, $\tilde{w}$ contains a c is equal to $(1 - (\frac{2}{3})^n)^{3^n} \leq e^{-2^n}$. Pick an increasing sequence $(n_k)_{k \in \mathbb{N}}$ such $\sum_{k \in \mathbb{N}} e^{-2^{n_k}} \leq 1 - \rho$. Then with probability at least ρ, for all k, there is a word $w_k \in \Sigma^{n_k}$ such that $\tilde{w}_k$ does not contain a c. Letting ρ go to 1, almost surely there is an infinite number of words w such that $\tilde{w} \in (a + b)^+$.

Let us consider an arbitrary language L' and $(w_n)_{n \in \mathbb{N}}$ be an enumeration of Σ^+. Then almost surely there is an infinite number of w_n such that $\tilde{w}_n$ belong to $(a + b)^+$. Recall that for such a word, the probability that it belongs to $\mathcal{L}(\mathcal{A}^{\leftarrow \frac{2}{3}})$ is equal to $\frac{1}{2}$. Let W_n be the *random* set of the first n^{th} such words. Then for all n, $\mathbf{Pr}(L' = \mathcal{L}(\mathcal{A}^{\leftarrow \frac{2}{3}})) \leq \mathbf{Pr}(L' \cap W_n = \mathcal{L}(\mathcal{A}^{\leftarrow \frac{2}{3}}) \cap W_n) = 2^{-n}$.

Thus $\mathbf{Pr}(L' = \mathcal{L}(\mathcal{A}^{\leftarrow \frac{2}{3}})) = 0$ and $\mathbf{Pr}(\mathcal{L}(\mathcal{A}^{\leftarrow \frac{2}{3}}) \in \mathcal{F}) = \sum_{L' \in \mathcal{F}} \mathbf{Pr}(L' = \mathcal{L}(\mathcal{A}^{\leftarrow \frac{2}{3}})) = 0$ for $\mathcal{F}$ a countable family of languages. $\square$

To show the soundness of the structural criterion described in Theorem 4.6 with experiments and comparisons, we have refined our experiments on DFA with noisy inputs by partitioning the randomly generated DFA, depending on whether they are equal-length-distinguishing or not.

We have chosen $|\Sigma| = 3$ since with greater size, it was difficult to generate DFAs that do not satisfy the hypotheses. Tables 8 and 9 summarize these experiments. The last rows of the tables (where the information gain is greater than one) confirm our conjecture. More precisely, for any equal-length-distinguishing DFA $\mathcal{A}$, as almost surely $\mathcal{L}(\mathcal{A}^{\leftarrow p})$ is not a recursively enumerable language, then beyond a certain threshold, the robustness of the algorithm increases quickly which is not the case in Table 9.

Range	#	$d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}^{\leftarrow p}))$	$d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}_E))$	$d(\mathcal{L}(\mathcal{A}^{\leftarrow p}), \mathcal{L}(\mathcal{A}_E))$	gain	deviation
$[0.005, 0.025]$	85	0.01114	0.03604	0.04345	0.30902	0.05162
$[0.002, 0.005]$	81	0.00338	0.00421	0.00747	0.80443	0.02793
$[0.001, 0.002]$	25	0.00142	0.00035	0.00174	4.09784	0.00062
$[0.0005, 0.001]$	16	0.00071	0.00006	0.00077	11.08439	0.00006

TABLE 8. Experiments on equal-length-distinguishing DFA

Range	#	$d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}^{\leftarrow p}))$	$d(\mathcal{L}(\mathcal{A}), \mathcal{L}(\mathcal{A}_E))$	$d(\mathcal{L}(\mathcal{A}^{\leftarrow p}), \mathcal{L}(\mathcal{A}_E))$	gain	deviation
$[0.005, 0.025]$	36	0.01089	0.02598	0.03410	0.41905	0.06152
$[0.002, 0.005]$	49	0.00308	0.00387	0.00646	0.79628	0.03763
$[0.001, 0.002]$	35	0.00136	0.00057	0.00182	2.39863	0.00072
$[0.0005, 0.001]$	36	0.00075	0.00063	0.00130	1.18583	0.00005

TABLE 9. Experiments on non equal-length-distinguishing DFA

5. CONCLUSION

We have studied how the PAC-version of KV's algorithm behaves for devices which are obtained from a DFA by introducing noise. More precisely, we have investigated whether KV's algorithm reduces the noise producing a DFA closer to the original one than the noisy device. We have considered four kinds of noise either being random or structured. We have shown that, on average, KV's algorithm behaves well for random noise but not for structured one. We have completed our study by establishing that almost surely the random noisy devices produce a non recursively enumerable language confirming the relevance of the structural criterion for robustness of KV's algorithm.

There are several directions for future work. In the short run, we want investigate whether our results are specific to the KV's algorithm or valid for all variants of the Angluin's algorithm. In another direction, KV's algorithm has no information about the original DFA. It would be interesting to introduce a priori knowledge and design more efficient algorithms. For instance, the algorithm could take as input the maximal size of the original DFA or a regular language that is a superset of the original language. In our setting the noise resulted in a noisy device which, once obtained, answers membership queries deterministically. A different form of noise to be studied would be that the answer to a query is randomly noisy meaning that for the same repeated query, different answers could occur.

Finally the language inference capacity of recurrent neural networks (RNN) especially on DFA has been demonstrated by recent work [WGY18, MY18]. So an interesting subsequent work would be to examine whether the KV's algorithm reduces the noise introduced by such RNNs.

REFERENCES

[ACSvdB20] Kousar Aslam, Loek Cleophas, Ramon Schiffelers, and Mark van den Brand. Interface protocol inference to aid understanding legacy software components. *Softw. Syst. Model.*, 19(6):1519–1540, nov 2020. `doi:10.1007/s10270-020-00809-2`.

[AL87] Dana Angluin and Philip D. Laird. Learning from noisy examples. *Mach. Learn.*, 2(4):343–370, 1987. `doi:10.1023/A:1022873112823`.

[Ang87] Dana Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987. `doi:10.1016/0890-5401(87)90052-6`.

[BEK02] Nader H. Bshouty, Nadav Eiron, and Eyal Kushilevitz. PAC learning with nasty noise. *Theoretical Computer Science*, 288(2):255–275, 2002. Algorithmic Learning Theory. `doi:10.1016/S0304-3975(01)00403-0`.

[BF72a] A. W. Biermann and J. A. Feldman. On the synthesis of finite-state machines from samples of their behavior. *IEEE Trans. Comput.*, 21(6):592–597, jun 1972. `doi:10.1109/TC.1972.5009015`.

[BF72b]   Alan W. Biermann and Jerome A. Feldman. A survey of results in grammatical inference. In
            S. Watanabe, editor, *Frontiers of Pattern Recognition*, pages 31–54. Academic Press, New York,
            1972. `doi:10.1016/B978-0-12-737140-5.50007-5`.

[BHKL09]   Benedikt Bollig, Peter Habermehl, Carsten Kern, and Martin Leucker. Angluin-style learning of
            nfa. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, IJCAI'09,
            page 1004–1009. Morgan Kaufmann Publishers Inc., 2009. `doi:10.5555/1661445.1661605`.

[CE07]   Alexander Clark and Rémi Eyraud. Polynomial identification in the limit of substitutable
            context-free languages. *Journal of Machine Learning Research*, 8:1725–1745, dec 2007. `doi:`
            `10.5555/1314498.1314556`.

[CL10]   Christos G. Cassandras and Stephane Lafortune. *Introduction to Discrete Event Systems*.
            Springer Publishing Company, Incorporated, 2010. `doi:10.1007/978-0-387-68612-7`.

[Cla10]   Alexander Clark. Distributional learning of some context-free languages with a minimally
            adequate teacher. In *Grammatical Inference: Theoretical Results and Applications*, ICGI'10,
            page 24–37. Springer-Verlag, 2010. `doi:10.5555/1886263.1886269`.

[Dec97]   Scott E. Decatur. Pac learning with constant-partition classification noise and applications to
            decision tree induction. In *Proceedings of the Fourteenth International Conference on Machine
            Learning*, ICML '97, page 83–91, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers
            Inc.

[FBLJ⁺21]   Daniel Furelos-Blanco, Mark Law, Anders Jonsson, Krysia Broda, and Alessandra Russo.
            Induction and exploitation of subgoal automata for reinforcement learning. *J. Artif. Int. Res.*,
            70:1031–1116, may 2021. `doi:10.1613/jair.1.12372`.

[FH17]   Paul Fiterau-Brostean and Falk Howar. Learning-based testing the sliding window behavior
            of TCP implementations. In Laure Petrucci, Cristina Seceleanu, and Ana Cavalcanti, editors,
            *Critical Systems: Formal Methods and Automated Verification - FMICS-AVoCS 2017, Turin,
            Italy, September 18-20, 2017, Proceedings*, volume 10471 of *Lecture Notes in Computer Science*,
            pages 185–200. Springer, 2017. `doi:10.1007/978-3-319-67113-0_12`.

[FJV16]   Paul Fiterau-Brostean, Ramon Janssen, and Frits W. Vaandrager. Combining model learning
            and model checking to analyze TCP implementations. In Swarat Chaudhuri and Azadeh Farzan,
            editors, *28th International Conference on Computer Aided Verification, CAV 2016, Toronto,
            ON, Canada, July 17-23, 2016, Part II*, volume 9780 of *Lecture Notes in Computer Science*,
            pages 454–471. Springer, 2016. `doi:10.1007/978-3-319-41540-6_25`.

[FLP⁺17]   Paul Fiterau-Brostean, Toon Lenaerts, Erik Poll, Joeri de Ruiter, Frits W. Vaandrager, and
            Patrick Verleg. Model learning and model checking of SSH implementations. In Hakan Erdogmus
            and Klaus Havelund, editors, *Proceedings of the 24th ACM SIGSOFT International SPIN
            Symposium on Model Checking of Software, Santa Barbara, CA, USA, July 10-14, 2017*, pages
            142–151. ACM, 2017. `doi:10.1145/3092282.3092289`.

[Gol78]   E Mark Gold. Complexity of automaton identification from given data. *Information and Control*,
            37(3):302 – 320, 1978. `doi:10.1016/S0019-9958(78)90562-4`.

[HIS⁺12]   Falk Howar, Malte Isberner, Bernhard Steffen, Oliver Bauer, and Bengt Jonsson. Inferring
            semantic interfaces of data structures. In Tiziana Margaria and Bernhard Steffen, editors, *ISoLA
            2012, Heraklion, Crete, Greece, October 15-18, 2012, Part I*, volume 7609 of *Lecture Notes in
            Computer Science*, pages 554–571. Springer, 2012. `doi:10.1007/978-3-642-34026-0_41`.

[Hoe63]   Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of
            the American Statistical Association*, 58(301):13–30, 1963. `doi:10.2307/2282952`.

[JMKO20]   Natasha Yogananda Jeppu, Thomas Melham, Daniel Kroening, and John O'Leary. Learning
            concise models from long execution traces. In *Proceedings of the 57th ACM/EDAC/IEEE Design
            Automation Conference*, DAC '20. IEEE Press, 2020. `doi:10.5555/3437539.3437631`.

[Kea98]   Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–
            1006, 1998. `doi:10.1145/293347.293351`.

[KV94]   Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*.
            MIT Press, 1994. `doi:10.7551/mitpress/3897.001.0001`.

[MP91]   Oded Maler and Amir Pnueli. On the learnability of infinitary regular sets. In *Proceedings of
            the Fourth Annual Workshop on Computational Learning Theory*, COLT '91, page 128–138.
            Morgan Kaufmann Publishers Inc., 1991. `doi:10.5555/114836.114848`.

[MY18]   Franz Mayr and Sergio Yovine. Regular inference on artificial neural networks. In Andreas
            Holzinger, Peter Kieseberg, A Min Tjoa, and Edgar R. Weippl, editors, *Proceedings of Interna-*
            tional Cross-Domain Conference for Machine Learning and Knowledge Extraction, CD-MAKE
            2018, Hamburg, Germany, August 27-30, 2018, volume 11015 of *Lecture Notes in Computer*
            Science, pages 350–369. Springer, 2018. `doi:10.1007/978-3-319-99740-7_25`.

[Qui86]   J. R. Quinlan. The effect of noise on concept learning. In *Machine Learning, An Artificial*
            Intelligence Approach Volume II, chapter 6, pages 149–166. Morgan Kaufmann, 1986.

[RS89]   R. L. Rivest and R. E. Schapire. Inference of finite automata using homing sequences. In
            Proceedings of the twenty-first annual ACM symposium on Theory of computing, STOC '89,
            page 411–420. Association for Computing Machinery, 1989. `doi:10.1145/73007.73047`.

[SG09]   Muzammil Shahbaz and Roland Groz. Inferring mealy machines. In *Proceedings of the 2nd*
            World Congress on Formal Methods, FM '09, page 207–222. Springer-Verlag, 2009. `doi:10.`
            `1007/978-3-642-05089-3_14`.

[Slo95]   Robert H. Sloan. Four types of noise in data for pac learning. *Inf. Process. Lett.*, 54(3):157–162,
            may 1995. `doi:10.1016/0020-0190(95)00016-6`.

[Sol64]   Ray J. Solomonoff. A formal theory of inductive inference. *Inf. Control.*, 7(1, 2):1–22, 224–254,
            1964. `doi:10.1016/S0019-9958(64)90223-2`.

[Val84]   Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. `doi:`
            `10.1145/1968.1972`.

[vdA12]   Wil M. P. van der Aalst. Process mining. *CACM*, 55(8):76–83, 2012. `doi:10.1145/2240236.`
            `2240257`.

[WGY18]   Gail Weiss, Yoav Goldberg, and Eran Yahav. Extracting automata from recurrent neural
            networks using queries and counterexamples. In Jennifer Dy and Andreas Krause, editors,
            Proceedings of the 35th International Conference on Machine Learning, volume 80, pages
            5247–5256. PMLR, 10–15 Jul 2018. `doi:10.48550/arXiv.1711.09576`.

[Wha74]   R. M. Wharton. Approximate language identification. *Information and Control*, 26(3):236 –
            255, 1974. `doi:10.1016/S0019-9958(74)91369-2`.

[YC10]   Ryo Yoshinaka and Alexander Clark. Polynomial time learning of some multiple context-free
            languages with a minimally adequate teacher. In *Proceedings of the 15th and 16th International*
            Conference on Formal Grammar, FG'10/FG'11, page 192–207, Berlin, Heidelberg, 2010. Springer-
            Verlag. `doi:10.1007/978-3-642-32024-8_13`.