Adversarial Models for Deterministic Finite Automata

Kaixuang Zhang[1(✉)], Qinglong Wang[2], and C. Lee Giles[1]

[1] Information Sciences and Technology, Pennsylvania State University, State College, USA
{kuz22,clg20}@psu.edu

[2] School of Computer Science, McGill University, Montreal, Canada
qinglong.wang@mail.mcgill.ca

Abstract. We investigate a finer-grained understanding of the characteristics of particular deterministic finite automata (DFA). Specifically, we study and identify the transitions of a DFA that are more important for maintaining the correctness of the underlying regular language associated with this DFA. To estimate transition importance, we develop an approach that is similar to the approach widely used to expose the vulnerability of neural networks with the adversarial example problem. In our approach, we propose an adversarial model that reveals the sensitive transitions embedded in a DFA. In addition, we find for a DFA its critical patterns where a pattern is a substring that can be taken as the signature of this DFA. Our defined patterns can be implemented as synchronizing words, which represent the passages from different states to the absorbing state of a DFA. Finally, we validate our study through empirical evaluations by showing that our proposed algorithms can effectively identify important transitions and critical patterns. To our knowledge, this is some of the first work to explore adversarial models for DFAs and is important due to the wide use of DFAs in cyberphysical systems.

Keywords: Deterministic Finite Automata · Transition importance · Critical patterns · Adversarial model

1 Introduction

There has been a great deal of work on the computational power of deterministic finite automata (DFA, level 3 in the Chomsky hierarchy). Although DFA models can be significantly different in terms of the number of states, accepted strings, and complexity [12], the family of DFA models is usually studied as a whole for their computational power and compared with other formal computation models in the Chomsky hierarchy [8]. As such, our understanding of DFA remains at a relatively coarse-grained level. We believe it is still an open question regarding on how to differentiate different DFAs.

Here, we study individual DFAs for their fine-grained characteristics, including transition importance and critical patterns. Specifically, we examine the

C. Goutte and X. Zhu (Eds.): Canadian AI 2020, LNAI 12109, pp. 540–552, 2020.
https://doi.org/10.1007/978-3-030-47358-7_55

importance of transitions by relating this task with the adversarial example problem [10] often seen in deep learning. This problem describes the phenomenon where a model, which generalizes well on clean datasets, is strikingly vulnerable to adversarial samples crafted by slightly perturbing clean samples. Because string identification is an important problem in time series, speech, and other scenarios, this motivates research into understanding complicated learning models such as neural networks. One particular approach is to identify feature-level perturbations that significantly affect a learning model. Similar approaches have been used for examining the sample-level importance in building a learning model [9]. These studies use adversarial examples as a data-driven tool for probing the learning model's vulnerability, hence indirectly gaining an understanding of complicated learning models. In order to directly gain a better understanding of a DFA, we follow a similar approach but study the sensitivity of a DFA through model-level perturbations.

Next, we study critical patterns that can be used for identifying a specific DFA. Specifically, we formally define a critical pattern as a substring, which effectively identifies all strings accepted by a certain DFA. We show that for certain classes of DFA, we can identify these strings statistically by checking the existence of critical patterns embedded in their generated strings without exhaustively searching all possible strings or querying the underlying DFA [1,13]. We then develop an algorithm for finding the critical patterns of a DFA by transforming this task as a DFA synchronizing problem [6]. Last, we provide a theoretical approach for estimating the length of any existing perfect patterns and validate our analysis with empirical results.

We feel that our analysis on DFA models will help in research on the security of cyberphysical systems that are based on working DFAs, e.g., compilers, VLSI design, elevators, and ATMs. This could be especially for the case when the actual state machine is exposed to adversaries and be attacked. It is the intent of this work to open a discussion on these issues.

2 Transition Importance of a DFA

DFAs are one of the simplest automata in the Chomsky hierarchy of phrase structured grammars [4]. More formally, a DFA can be described by a five-tuple $A = \{\Sigma, S, I, F, T\}$, where Σ is the input alphabet (a finite, non-empty set of symbols), S denotes a finite and non-empty set of states, $I \in S$ represents the initial state while $F \subseteq S$ represents the set of accept states, and T is a set of deterministic transition rules. The transition rules of a certain DFA essentially describe how that DFA will process a string as it traverses its states. Throughout this paper all DFAs are complete minimal DFAs. Due to its deterministic nature, it is natural to assume different transitions are equally important for identifying a DFA. However, as will become clear from our analysis, this assumption does not generally hold. Here we illustrate this with the a DFA associated with the

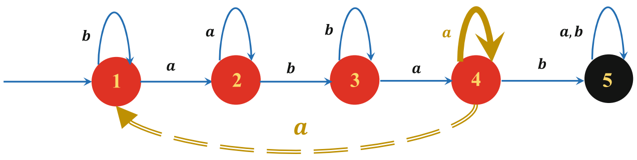


Fig. 1. Example for the Tomita-7 grammar. Red (Black) states are the accept (reject) states (Color figure online).

Tomita-7 grammar[1] shown in Fig. 1. Among all transitions, the cyclic transition (with input a) associated with state-4 is the most important one. This is because by substituting this transition by a transition to state-1 with the same input, we can add significantly more strings to the set of accepted strings.

2.1 Transition Importance Estimation as an Adversarial Model Problem

To estimate the importance of each transition and identify more important ones, we take an approach that is complementary to the approach used to identify sensitive features of a data sample viewed by a deep neural network (DNN) in the context of the adversarial example problem. As such, the transition importance estimated by our approach essentially reflects the sensitivity of a DFA with respect to a transition.

A typical formulation of the adversarial example problem is to maximize a loss function $\mathcal{L}$ with respect to a normal sample x_0 and a model f. Then finding an adversarial sample $\hat{x}$ is conducted by solving the following problem:

$$\hat{x} = \arg\max_{|x - x_0| \leq \epsilon} \mathcal{L}(x, f), \tag{1}$$

where ϵ denotes some predefined constraint on the scale of perturbation. Here we propose to transform the adversarial example problem (Eq. (1)) into the *adversarial model problem*, which considers model-level perturbations. Explicitly, given a model f_0 and a fixed set of string samples X, we try to solve the following problem:

$$\hat{f} = \arg\max_{|f - f_0| \leq \epsilon} \sum_{x \in X} \mathcal{L}(x, f). \tag{2}$$

Eq. (2) describes the problem of perturbing a target model in a constrained manner to cause maximal loss and provides an alternative view of the adversarial

[1] Tomita [11] defined the following grammars with a binary alphabet: (1) a^*, (2) $(ab)^*$, (3) an odd number of consecutive $'a'$s is always followed by an even number of consecutive $'b'$s, (4) any binary string not containing "bbb" as a substring, (5) even number of bs and even number of $'a'$s, (6) the difference between the numbers of $'b'$s and $'a'$s is a multiple of 3, (7) $b^*a^*b^*a^*$. These grammars have been widely used in grammatical inference.

example problem. Specifically, given an ideal mapping f from some functional space $\mathcal{F}$ for a certain learning task, and an arbitrarily small approximation error ϵ, the universal approximation theory [5] states that one can always find a candidate f' in some other functional space $\mathcal{F}'$ (generally taken as a subset of $\mathcal{F}$) satisfying $\|f - f'\| < \epsilon$. Given that DNNs already have very complicated architectures[2], we can only measure the difference between f and f' numerically, although these two functions might be quite different. Since these models built through analytical approaches may not necessarily have actions aligning with our intuition and expectation, we cannot easily, if not impossibly, establish a physical understanding of the gap between f' and f. Furthermore, in practice the gap between these two functions may be amplified by formulating the approximation problem as an optimization problem, and then applying various techniques to solve the latter [2]. These combined effects imply that the root cause of the adversarial example problem lies in both the ambiguity of the theoretical foundation for building a learning model and the imperfection in the practice of applying a learning model. Moreover, it is important to note that our transformation cannot be easily applied to complicated models like DNNs. The function represented by a DNN has too many parameters, including weights, neurons, layers, and all sorts of hyper-parameters. This results in an enormous perturbation space.

On the other hand, for a DFA, the perturbation space is significantly limited to only include its transitions and states. Furthermore, the perturbation of a state can be represented by a set of perturbations applied to the transitions associated with this state. Therefore, in the following, we only consider transition perturbations as they provide a more general description of the adversarial perturbations of a DFA. In addition, we only consider perturbations that make substitution operations on the transitions. This is because for a given DFA, inserting transitions is not allowed since this DFA is already complete and minimal. Also, removing a transition is equivalent to substituting this transition to the transition that connects the current state to an absorbing state, of which the outward transitions all loop back to itself. Our study of the adversarial DFA can be taken as a step in studying the adversarial phenomenon by restricting the underlying models to be physically interpretable and directly investigating the vulnerability of that model.

2.2 Transition Importance

The deterministic property of a DFA enables it to be naturally immune to adversarial examples. However, when the adversarial perturbation is applied to a DFA, it is possible to generate an adversarial DFA, which only differs from the original DFA by a limited number of transitions, yet recognizes a regular grammar that is dramatically different from the one associated with the original DFA. To quantitatively evaluate the difference between two sets of strings accepted by different DFAs, here we introduce the following metric:

[2] Recent research [7] on explaining DNNs have demonstrated the difficulty of analyzing and inspecting these powerful models.

Definition 1 (Intersection over Union (IoU)). *Given two arbitrary DFAs represented by A and $\hat{A}$, and their accepted sets of strings denoted by X and $\hat{X}$, respectively, then*

$$IoU(A, \hat{A}) = \frac{\left|X \cap \hat{X}\right|}{\left|X \cup \hat{X}\right|}. \tag{3}$$

It is easy to notice that the metric IoU is well-defined and lies between 0 and 1. One can apply the L'Hopital's rule to calculate it if both the numerator and the denominator approach infinity. By using the above definition of IoU, we express the adversarial model problem for a DFA as perturbing the transitions of a given DFA to reach a low IoU. Then we have the following theorem.

Theorem 1. *Given a DFA with alphabet $\Sigma = \{a_1, a_2\}$, we use A_1 and A_2 to denote its transition matrices associated with the first and second input symbol. Similarly, let $\hat{A}_1$ and $\hat{A}_2$ denote the transition matrices of perturbed DFA yielding*

$$IoU(A, \hat{A}) = \left(\frac{\sum_{n=1}^{\infty} (\mathbb{1} \otimes p)^{\mathrm{T}} (M_1 \otimes (A_1 + A_2) + M_2 \otimes (\hat{A}_1 + \hat{A}_2))^n (\mathbb{1} \otimes q)}{\sum_{n=1}^{\infty} (p \otimes p)^{\mathrm{T}} (A_1 \otimes \hat{A}_1 + A_2 \otimes \hat{A}_2)^n (q \otimes q)} - 1 \right)^{-1}. \tag{4}$$

where $p \in \mathbb{B}^n$ is a one-hot encoding vector to represent the initial state, and $q \in \mathbb{B}^n$ denotes the set of accept states of a DFA with n states. We also have $\mathbb{1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $M_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, $M_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$, and $\otimes$ denotes the Kronecker product.

Due to space constraints, we only provide a sketch of this proof. For this we construct a new automaton that represents the union of two source DFAs. The initial state vector, accepting state vector, and the adjacency matrix of this constructed automaton are denoted as $\mathbb{1} \otimes p$, $\mathbb{1} \otimes q$, and $M_1 \otimes (A_1 + A_2) + M_2 \otimes (\hat{A}_1 + \hat{A}_2)$, respectively. Similarly, for the DFA that recognizes the intersection of two sets of strings accepted by two DFAs, we denote its initial state vector, accepting state vector, and the adjacency matrix as $p \otimes p$, $q \otimes q$, and $A_1 \otimes \hat{A}_1 + A_2 \otimes \hat{A}_2$, respectively. In order to compute the cardinality of the union set and the intersection set, we need to sum the number of strings for which the length varies from 1 to infinity. Now assume that there are two column vectors s_I and s_E, which represent the set of initial and ending states. Then the number of N-length strings that reach s_E from s_I is $s_I^{\mathrm{T}} P^N s_E$.

Theorem 1 provides directly an explicit formulation for computing our defined IoU. As such, the original adversarial model problem for the DFA can be transformed to an optimization problem. Furthermore, we require that this manipulation only allows one transition substitution to be applied to one of the transition matrices associated with different inputs. The allowed single transition substitution causes the Frobenius norm of the manipulated transition matrix to be changed by $\sqrt{2}$. This also avoids changes to the absorbing states of the source DFA (if they exist), so that any existing absorbing states will not be affected.

In addition, we require the set of accepted states remains the same. Therefore, we have the following optimization problem[3]:

$$\begin{aligned}
\min_{\hat{A}_1, \hat{A}_2 \in \mathcal{T}} & \frac{\sum_{n=1}^{\infty} (p \otimes p)^{\mathrm{T}} (A_1 \otimes \hat{A}_1 + A_2 \otimes \hat{A}_2)^n (q \otimes q)}{\sum_{n=1}^{\infty} (\mathbb{1} \otimes p)^{\mathrm{T}} (M_1 \otimes (A_1 + A_2) + M_2 \otimes (\hat{A}_1 + \hat{A}_2))^n (\mathbb{1} \otimes q)} \\
\text{s.t. } & \left\| \hat{A}_1 - A_1 \right\|_F^2 + \left\| \hat{A}_2 - A_2 \right\|_F^2 = 2; \\
& y(A_1 + A_2) = y(\hat{A}_1 + \hat{A}_2); \\
& (A_1 + A_2) y^{\mathrm{T}} = (\hat{A}_1 + \hat{A}_2) y^{\mathrm{T}}.
\end{aligned} \tag{5}$$

where $y = 0$ when the source DFA does not have an absorbing state and $y = [0, 0, \cdots, 1]$. Otherwise, $\mathcal{T}$ denotes the set of transition matrices which contains exactly one 1 in each row, and $\|\cdot\|_F$ denotes the Frobenius norm.

In practice, it is possible that some additional constraints can be added to the above formulation. Specifically, here we require the perturbed DFA to remain strongly connected and no new absorbing states will be created. Since it is difficult to formulate these constraints in Eq. (5), we manually examine their violations in the obtained solutions. Note that these constraints can be easily checked by analyzing the spectrum of the perturbed transition matrix.

2.3 Evaluation of DFA Transition Importance

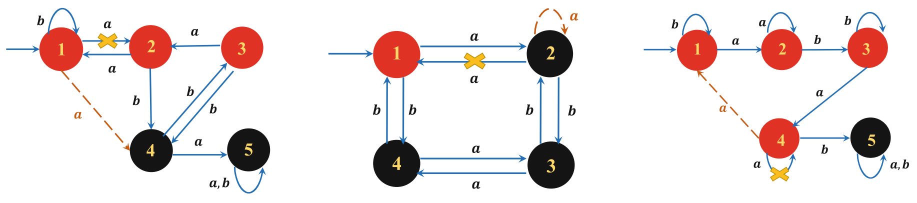
In the following, we use the Tomita grammars to demonstrate our estimation of the transition importance of DFAs. Since this is the first work on studying the adversarial scheme of formal computation models, our evaluation mainly focuses on examining the effectiveness of our proposed approach.

In the experiments, we select the Tomita-3/5/7 grammars as examples. These grammars are selected as they are representative of the exponential, proportional, and polynomial classes [12] of regular grammars with the binary alphabet, respectively (These classes are introduced in Sect. 3). Since it is impossible to sum up to infinity, for our evaluation we fix the maximum length N of binary strings to 20. Also, instead of solving the original objective which takes a quotient form, we apply the symmetry difference of two sets as an alternative. This choice is reasonable since the objective functions capture the same essence of minimizing the cardinality of the intersection and maximizing the cardinality of the union of two sets. Furthermore, as the original problem is formulated as a high-order integer programming problem, which is difficult to solve with existing solvers, we relax the constraints such that $\hat{A}_1$ and $\hat{A}_2$ are constrained as row-wise stochastic matrices as their entries. As such, we determine the final perturbation by selecting the one with the maximal value, which represents the maximal transition probability. We notice that our approximation may not yield the real optimal solution; however, as shown by the results in Table 1, it provides satisfying results in analyzing the transition importance.

[3] The constant number 1 is omitted for simplicity.

Table 1. Optimization results for the Tomita-3/5/7 grammars.

		IoU	
		Value from optimization	Value from randomization
The	3	1.48e-3	0.342
Tomita	5	0.152	0.289
Grammars	7	0.025	0.225



(a) Grammar 3 (b) Grammar 5 (c) Grammar 7

Fig. 2. Illustration of identified important transitions for example DFAs. The marked (with a yellow cross) and dashed lines demonstrate the most sensitive transitions of the original DFAs and the perturbed transitions, respectively. (Color figure online)

The effectiveness of our optimization approach when comparing it with a randomization approach is shown in Table 1. Specifically, for the randomization approach, we randomly select five legitimate perturbations (manually checked according to the constraints described above) and calculate and average the resulting IoU_{rand}. We then compare IoU_{rand} with the IoU_{opt} obtained by our optimization. It is clear that the results provided by the optimization approach are much more desirable. We also provide a visualization of the perturbations generated by our approach for each investigated grammar in Fig. 2.

3 Critical Patterns of DFA

3.1 Different Types of Critical Patterns

Here, we provide a relatively coarse-grained view, in contrast to what we described regarding transition importance, to investigate the characteristics of a DFA. Specifically, we identify critical patterns of a DFA, defined as:

Definition 2 (Absolute and relative patterns of a DFA). *Given the alphabet Σ of a DFA and a data space $X \subseteq \Sigma^*$, X is the union of two disjoint sets, i.e., $X = P \cup N$[4], and we define the following patterns:*

$$\text{Absolute pattern}: \ \hat{m} = \underset{|m|=k}{\arg\max} \left| Pr_{m \sim_f y}(y \in P) - Pr_{m \sim_f y}(y \in N) \right|. \quad (6)$$

[4] For a DFA, P (N) represents the space of strings accepted (rejected) by this DFA.

$$\text{Relative pattern}: \ \hat{m} = \underset{|m|=k}{\arg\max} \left| Pr_{y \in P}(m \sim_f y) - Pr_{y \in N}(m \sim_f y) \right|, \tag{7}$$

where y is a string in X and $m \sim_f y$ indicates that m is a factor (consecutive substring) of y.

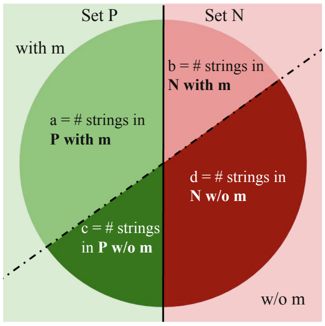

Set P
Set N
with m
b = # strings in **N with m**
a = # strings in **P with m**
d = # strings in **N w/o m**
c = # strings in **P w/o m**
w/o m

Fig. 3. An illustration of the difference between absolute and relative patterns.

Here we focus on the general case where all the strings follow the uniform distribution without using any particular prior knowledge. We illustrate the difference between the absolute and relative patterns with the example in Fig. 3 by splitting the entire data space X into four parts denoted as $\{a, b, c, d\}$. According to Eq. (6), an absolute pattern describes the substring m (has the length of k) such that, among all strings that contain m, it causes the largest discrepancy between the probabilities of a string that belongs to different disjoint sets. Thus, the absolute pattern differentiates strings in $\{a, b\}$, and the objective in Eq. (6) are equal to $\frac{|a-b|}{a+b}$. In contrast, a relative pattern is identified by considering the statistics of the entire data space, with the objective in Eq. (7) equal to $\left|\frac{a}{a+c} - \frac{b}{b+d}\right|$. Note that these two patterns are equivalent to each other under certain circumstances. For example, consider a DFA that rejects any binary string containing "bbb" as a substring[5]. In this case, both the absolute and relative patterns identify the factor "bbb". Here, we are concerned with the absolute pattern since it provides better insight as to the connection between identified patterns and the underlying DFA. In contrast a relative pattern mainly provides a conceptual understanding from a statistic perspective. Furthermore, we introduce the following definition:

Definition 3 (Perfect absolute pattern of a DFA).
Let $\mathcal{A}_p = \{m \mid \max_m \left|Pr_{m \sim_f y}(y \in P) - Pr_{m \sim_f y}(y \in N)\right| = 1\}$, *then the perfect absolute pattern is defined as:*

$$\hat{m} = \underset{m \in \mathcal{A}_p}{\arg\min} |m|. \tag{8}$$

[5] The example DFA is associated with the Tomita-4 grammar.

A perfect absolute pattern describes a substring, which has minimal length among all absolute patterns and perfectly differentiates the strings from different disjoint sets. However, not all DFAs have perfect absolute patterns. Some DFAs, which have a cyclic property, contain recurrent or persistent states that contain both accepting and non-accepting states. This indicates that these DFAs can never determine the label of a string until they finish processing the entire string. These DFAs with a binary alphabet, as previously determined [12], belong to one of three classes, which are then categorized according to the complexity of different DFAs. Specifically, the complexity of a DFA is measured by its entropy value, which essentially reflects how balanced are the sets of strings accepted or rejected by a DFA. As such, a grammar with a higher complexity has a higher entropy value, hence recognizing more balanced string sets. Based on the entropy values of different DFAs, they can be categorized into three basic classes (1) polynomial class, where the number of accepted strings of a certain length is a polynomial function of the length; (2) exponential class, where the number of accepted strings of a certain length takes an exponential form of the length with the base value smaller than 2; (3) proportional class, where the number of accepted strings of a certain length is proportional to number of all binary strings with the same length. Interestingly, except for the proportional class, which contains DFAs with either 0 or 2 absorbing states, DFAs from other classes have exactly one absorbing state. See Wang et al. [12] for more details.

Upon inspection, it is just a random guess for identifying a string accepted by a DFA which has no absorbing state and by only checking its contained factors. Also, determining the pattern of a DFA, which contains two absorbing states, can be taken as performing a random guess twice. As such, we only focus in the following analysis on DFAs belonging to the polynomial and the exponential classes. Importantly, we find that identifying a perfect absolute pattern of a DFA is essentially analogous to designing a *synchronizing word* [6] for the absorbing state of a DFA. Therefore, instead of solving the optimization problem in Eq. (6), we propose a DFA synchronizing word approach and design a metric to evaluate the confidence for determining whether a certain string belongs to a particular class. We show that our metric is highly correlated with the probability in Eq. (6).

3.2 DFA Synchronizing Algorithm

Recall that the synchronizing word (or the reset sequence) is a substring that sends any state of this DFA to the same state. An absorbing state naturally fits this synchronizing scheme. As such, we can set the absorbing state as the state to be synchronized. And since all states will result in an absorbing state when applying the same substring to these states, the label of a string containing the substring can be determined definitely.

However, given a string of fixed length k, there is no guarantee that we can always reach an absorbing state. Thus, we design the following algorithm and metric to evaluate the efficiency of identifying an absolute pattern. More specifically, given a DFA with n states and a predefined length k, we then have its k-order transition matrix A_{Σ}^{k} by multiplying the transition matrix A_{Σ} by itself k times. We focus on the column associated with the absorbing state. This column represents the prefixes coming from all states to this absorbing state. We now choose the most frequent substring m appearing in this column. We denote that number of occurrences as $\hat{n}$ and determine m since an absolute pattern has confidence of $\hat{n}/n$. For the perfect absolute pattern, the confidence is 1, since the substring sends all other states to the absorbing state and it will appear in each entries of the column associated with the absorbing state of the k-order transition matrix. In experiments presented in the latter part of this section, we demonstrate the results of applying this algorithm.

Furthermore, given a DFA with one absorbing state, similar to the Černý's conjecture [3], we can estimate the length of a perfect absolute pattern associated with a DFA by providing a loose upper bound. That is, given a DFA with an absorbing state, we have the following theorem for estimating the minimal length of a synchronizing substring, which leads all states to the absorbing state.

Theorem 2. *The length of a perfect absolute pattern of a DFA with* n *states is at most* $n(n-1)/2$.

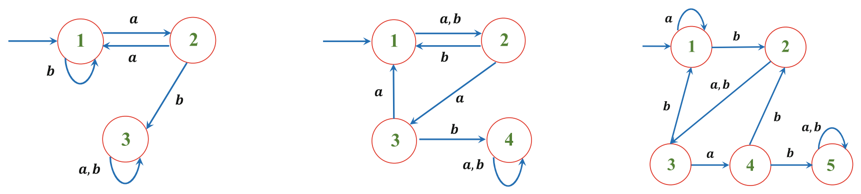
To obtain an upper bound of the length of a perfect absolute pattern, we need to consider the worst case. Specifically, the distance between each state and the absorbing state is $1, 2, \cdots, n-1$, respectively. In addition, at step t, synchronizing the nearest state is the optimal choice. Furthermore, after synchronizing the t-step nearest states, the distances between the rest $n-t$ states and the absorbing state range exactly from $t+1$ to $n-1$ during this iterative process. As such, to synchronize all states in the worst case, we have the length of a synchronizing substring at most $1+2+\cdots+(n-1)$, which is equal to $n(n-1)/2$.

It is straightforward to check that the upper bound in Theorem 2 holds for any size of alphabet. As such, we conjecture that there exists a tighter upper bound, which depends on the number of states and the DFA alphabet size. Next, we provide some examples in order to further investigate the pattern length.

We demonstrate in Fig. 4a and b that when the number of states of a DFA is set to 3 or 4, we can construct a DFA for which the perfect absolute pattern meets the upper bound exactly. Specifically, for the DFAs shown in Fig. 4a and b, their associated patterns are *bab* and *babaab*. However, it is impossible to construct a 5-state DFA, for which the perfect absolute pattern has a length that reaches the upper bound in Theorem 2. More specifically, we have the following result:

Theorem 3. *The length of a absolute pattern of a 5-state DFA is at most 9.*

This theorem can be proved by using combinatoric and enumeration techniques, and is omitted due to space constraints. In Fig. 4c, we construct an example five-state DFA that has a pattern with a length of 9 in the worst case.



(a) Pattern *bab* (b) Pattern *babaab* (c) Pattern *babaabaab*

Fig. 4. DFA examples that illustrate Theorem 2.

Table 2. Transition matrices for example DFAs. $A_\sigma(s_1, s_2)$ represents a transition from state s_1 to s_2 via σ. We only provide the transitions matrices for DFA-3/4/5 and omit the matrices for DFA-1/2 (Tomita-4/7) due to space limit.

	Transition matrix							
DFA 3	$A_a(1,4)$	$A_a(2,3)$	$A_a(3,4)$	$A_a(4,3)$	$A_a(5,4)$	$A_a(6,6)$		
	$A_b(1,2)$	$A_b(2,3)$	$A_b(3,3)$	$A_b(4,5)$	$A_b(5,6)$	$A_b(6,6)$		
DFA 4	$A_a(1,5)$	$A_a(2,4)$	$A_a(3,2)$	$A_a(4,1)$	$A_a(5,6)$	$A_a(6,1)$	$A_a(7,7)$	
	$A_b(1,2)$	$A_b(2,3)$	$A_b(3,2)$	$A_b(4,3)$	$A_b(5,6)$	$A_b(6,7)$	$A_b(7,7)$	
DFA 5	$A_a(1,2)$	$A_a(2,3)$	$A_a(3,4)$	$A_a(4,8)$	$A_a(5,3)$	$A_a(6,8)$	$A_a(7,4)$	$A_a(8,8)$
	$A_b(1,6)$	$A_b(2,1)$	$A_b(3,5)$	$A_b(4,7)$	$A_b(5,2)$	$A_b(6,7)$	$A_b(7,1)$	$A_b(8,8)$

Table 3. Patterns and their corresponding confidence for example DFAs. When several patterns have the same length, we randomly show only one of them.

Length	DFA 1		DFA 2			DFA 3		DFA 4		DFA 5	
	Pattern	Con.	Pattern	Con.	Prob.	Pattern	Con.	Pattern	Con.	Pattern	Con.
2	bb	2/3	ab	3/5	0.674	bb	1/2	ab	3/7	aa	5/8
3	bbb	1	bab	4/5	0.912	abb	1/2	abb	3/7	aaa	7/8
4			abab	1	1.0	abba	1/2	aabb	4/7	aaaa	1
5						baabb	2/3	aaabb	5/7		
6						bbaabb	1	aaaabb	5/7		
7								aaaabbb	5/7		
8								bbaaaabb	1		

3.3 Evaluation of DFA Pattern Identification

In the following experiments, we use the Tomita-4 and Tomita-7 grammars (indexed as DFA-1 and DFA-2), which are representative grammars for the exponential and the polynomial classes [12], respectively, and also use randomly generated other DFAs as shown in Table 2. For all DFAs, we set their starting state as state 1 and their absorbing states as the states with the largest indexes. By applying the algorithm we previously introduced, we obtain and demonstrate in Table 3 identified patterns for all evaluated DFAs.

We observe in Table 3 that our algorithm successfully identified the perfect absolute pattern for the Tomita-4 (DFA-1) grammar. We also find that the length of an identified perfect absolute pattern does not necessarily increase as the number of states increases. Moreover, we observe that the confidence for determining an absolute pattern for all DFAs is non-decreasing as the length of the identified pattern increases. To further understand the relationship between the confidence and the probability introduced in the definition of the absolute pattern, we design the following experiment and use DFA-2 as our demonstrative example. Specifically, we generate 1000 strings for each identified pattern with their lengths less than 15. We then calculate the frequency of the generated strings that appear in both the accepted and rejected sets, respectively. In particular, we use that frequency to approximate the probability by using the law of large numbers. We show in Table 3 that the probability difference and confidence have a positive correlation. Although we do not establish a theoretical relationship between the above mentioned two statistics, we empirically show that it is reasonable to replace the probability with the confidence. As such, we believe these results validate the effectiveness of our algorithm.

4 Conclusion

Here we have defined transition importance and critical patterns for DFAs, which we believe gives insight into understanding and identification of specific DFAs. Specifically, we transformed the widely-accepted adversarial sample scheme to an adversarial model scheme, which reveals the sensitivity of a model with respect to its components. For the case of a DFA, we focus on the components represented by its transitions. In addition, we have designed an effective synchronizing algorithm to find critical patterns of a DFA and studied the upper bound of the length of a perfect absolute pattern. Finally, we empirically validated our algorithms and the practicability of our metric with several grammars. Future work could focus on extending this work to understand more complex models and DFAs used in real applications.

References

1. Angluin, D.: Learning regular sets from queries and counterexamples. Inf. Comput. **75**(2), 87–106 (1987)
2. Bottou, L., Bousquet, O.: The tradeoffs of large scale learning. In: NIPS, pp. 161–168. Curran Associates Inc., (2007)
3. Černỳ, J.: Poznámka k homogénnym experimentom s konečnỳmi automatmi (a note on homogeneous experiments with finite automata). Matematicko-fyzikálny časopis **14**(3), 208–216 (1964)
4. Chomsky, N.: Three models for the description of language. IRE Trans. Inf. Theory **2**(3), 113–124 (1956)
5. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Math. Control Signals Syst. **2**(4), 303–314 (1989). https://doi.org/10.1007/BF02551274

6. Don, H., Zantema, H.: Finding DFAs with maximal shortest synchronizing word length. In: Drewes, F., Martín-Vide, C., Truthe, B. (eds.) LATA 2017. LNCS, vol. 10168, pp. 249–260. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-53733-7_18
7. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM Comput. Surv. **51**(5), 1–42 (2019)
8. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to automata theory, languages, and computation. ACM Sigact News **32**(1), 60–65 (2001)
9. Koh, P.W., Liang, P.: Understanding black-box predictions via influence functions. In: Proceedings of the 34th International Conference on Machine Learning. ICML, pp. 1885–1894 (2017)
10. Serban, A.C., Poll, E.: Adversarial examples - a complete characterisation of the phenomenon. CoRR, abs/1810.01185 (2018)
11. Tomita, M.: Dynamic construction of finite-state automata from examples using hill-climbing. In: Proceedings of the Fourth Annual Conference of the Cognitive Science Society, pp. 105–108 (1982)
12. Wang, Q.: A comparative study of rule extraction for recurrent neural networks. arXiv preprint arXiv:1801.05420 (2018)
13. Weiss, G., Goldberg, Y., Yahav, E.: Extracting automata from recurrent neural networks using queries and counterexamples. In: Proceedings of Machine Learning Research. ICML, vol. 80, pp. 5244–5253. PMLR (2018)