Certifying DFA Bounds for Recognition and Separation*

Orna Kupferman, Nir Lavee, and Salomon Sickert

School of Computer Science and Engineering, The Hebrew University, Israel.
orna@cs.huji.ac.il, nir.lavee@mail.huji.ac.il,
salomon.sickert@mail.huji.ac.il

Abstract The automation of decision procedures makes certification essential. We suggest to use determinacy of turn-based two-player games with regular winning conditions in order to generate certificates for the number of states that a deterministic finite automaton (DFA) needs in order to recognize a given language. Given a language L and a bound k, recognizability of L by a DFA with k states is reduced to a game between Prover and Refuter. The interaction along the game then serves as a certificate. Certificates generated by Prover are minimal DFAs. Certificates generated by Refuter are faulty attempts to define the required DFA. We compare the length of offline certificates, which are generated with no interaction between Prover and Refuter, and online certificates, which are based on such an interaction, and are thus shorter. We show that our approach is useful also for certification of separability of regular languages by a DFA of a given size. Unlike DFA minimization, which can be solved in polynomial time, separation is NP-complete, and thus the certification approach is essential. In addition, we prove NP-completeness of a strict version of separation.

1 Introduction

Deterministic finite automata (DFAs) are among the most studied computation models in theoretical computer science. In addition to serving as an abstract mathematical concept, they are often the basis for specification and implementation of finite-state hardware and software designs [**?**]. In particular, the theory of DFAs applies also to deterministic automata of infinite words that recognize *safety* languages, which are characterized by finite forbidden behaviors [**?**,**?**].

A fundamental problem about DFAs is their *minimization*: For $k \geq 1$, we say that a language $L \subseteq \Sigma^*$ is *k-DFA-recognizable* if there is a k-DFA, namely a DFA with at most k states, that recognizes L. In the minimization problem, we are given a DFA $\mathcal{A}$ and a bound $k \geq 1$, and decide whether $L(\mathcal{A})$, namely

* This is the full version of an article with the same title that appears in the ATVA 2021 conference proceedings. The final authenticated publication is available online at http://arxiv.org/abs/2107.01566. Orna Kupferman is supported in part by the Israel Science Foundation, grant No. 2357/19. Salomon Sickert is supported by the Deutsche Forschungsgemeinschaft (DFG) under project number 436811179.

the language of $\mathcal{A}$, is k-DFA-recognizable. DFAs enjoy a clean (and beautiful) theory of canonicity and minimization, based on a *right-congruence* relation: A language $L \subseteq \Sigma^*$ induces a relation $\sim_L \subseteq \Sigma^* \times \Sigma^*$, where for every two words $h_1, h_2 \in \Sigma^*$, we have that $h_1 \sim_L h_2$ iff for all words $t \in \Sigma^*$, we have that $h_1 \cdot t \in L$ iff $h_2 \cdot t \in L$. By the Myhill-Nerode Theorem [**?**,**?**], the language L is k-DFA-recognizable iff the number of equivalence classes of $\sim_L$ is at most k. Moreover, a given DFA $\mathcal{A}$ can be minimized in polynomial time, by a fixed-point algorithm that merges states associated with the same equivalence class of $\sim_{L(\mathcal{A})}$.

Another fundamental problem about DFAs is *separation*: Given DFAs $\mathcal{A}_1$ and $\mathcal{A}_2$, and a bound $k \geq 1$, decide whether there is a k-DFA $\mathcal{A}$ that *separates* $\mathcal{A}_1$ and $\mathcal{A}_2$. That is, $L(\mathcal{A}_1) \subseteq L(\mathcal{A})$ and $L(\mathcal{A}) \cap L(\mathcal{A}_2) = \emptyset$. Finding a separator for $\mathcal{A}_1$ and $\mathcal{A}_2$ is closely related to the *DFA identification* problem. There, given sets $S_1, S_2 \subseteq \Sigma^*$ of positive and negative words, and a bound $k \geq 1$, we seek a k-DFA that accepts all words from S_1 and no word from S_2. DFA identification is NP-complete [**?**], with numerous heuristics and applications [**?**,**?**]. NP-hardness of DFA separation can be obtained by a reduction from DFA identification, but for DFA separation with additional constraints, in particular strict separation, NP-hardness is open [**?**]. Studies of separation include a search for regular separators of general languages [**?**], as well as separation of regular languages by weaker classes of languages, e.g., FO-definable languages [**?**] or piecewise testable languages [**?**].

Let us return to the problem of DFA minimization, and assume we want to *certify* the minimality of a given DFA. That is, we are given a DFA $\mathcal{A}$ and a bound $k \geq 1$, and we seek a proof that $L(\mathcal{A})$ is not k-DFA-recognizable. The need to accompany results of decision procedures by a certificate is not new, and includes certification of a "correct" decision of a model checker [**?**,**?**], reachability certificates in complex multi-agent systems [**?**], and explainable reactive synthesis [**?**]. Certifying that $L(\mathcal{A})$ is not k-DFA-recognizable, we can point to $k+1$ words $h_1, \ldots, h_{k+1} \in \Sigma^*$ that belong to different equivalence classes of the relation $\sim_{L(\mathcal{A})}$, along with an explanation why they indeed belong to different classes, namely words $t_{i,j} \in \Sigma^*$, for all $1 \leq i \neq j \leq k+1$, such that $h_i \cdot t_{i,j}$ and $h_j \cdot t_{i,j}$ do not agree on their membership in $L(\mathcal{A})$.

The above certification process is *offline*: Refuter (that is, the entity proving that $L(\mathcal{A})$ is not k-DFA-recognizable) generates and outputs the certificate without an interaction with Prover (that is, the entity claiming that $L(\mathcal{A})$ is k-DFA-recognizable). In this work we describe an *interactive certification protocol*:[1] Given $\mathcal{A}$ and $k \geq 1$, Refuter and Prover interact, aiming to convince each other about the (non-)existence of a k-DFA for $L(\mathcal{A})$. Our approach offers two advantages over offline certification. First, the length of the certificate is shorter. Second, the interactive protocol can also be used for efficiently certifying bounds

[1] Note that while our certification protocol is interactive, the setting is different from that of an interactive proof system in computational complexity theory. In particular, our Prover and Refuter are both finite-state, they have complementary objectives, and no probability is involved.

on the size of DFA separators. In addition, we solve the open problem of the complexity of deciding strict separation by a k-DFA. We show that it is NP-complete, and so are variants requiring only one side of the separation to be strict.

The underlying idea behind the interactive certification protocol is simple: Consider a language $L \subseteq \Sigma^*$ and a bound $k \geq 1$. We consider a *turn-based two-player game* between Refuter and Prover. In each round in the game, Prover provides a letter from a set $[k] = \{1, 2, \ldots, k\}$ that describes the state space of a DFA for L that Prover claims to exist, and Refuter responds with a letter in $\Sigma \cup \{\#\}$, for a special reset letter $\# \notin \Sigma$. Thus, during the interaction, Prover generates a word $y \in [k]^\omega$ and Refuter generates a word $x \in (\Sigma \cup \{\#\})^\omega$. The word x describes an infinite sequence of words in Σ^*, separated by #'s, and the word y aims to describe runs of a k-DFA on the words in the sequence. Prover wins if the described runs are legal: They all start with the same initial state and follow some transition function, and are consistent with L: There is a way to classify the states in $[k]$ to accepting and rejecting such that Prover responds with an accepting state whenever the word generated by Refuter since the last # is in L. Clearly, if there is a k-DFA for L, then Prover can win by following its runs. Likewise, a winning strategy for Prover induces a k-DFA for L. The key idea behind our contribution is that since the above described game is determined [**?**], Refuter has a winning strategy iff no k-DFA for L exists. Moreover, since the game is regular, this winning strategy induces a finite-state transducer, which we term an (L, k)-*refuter*, and which generates interactive certificates for $L(\mathcal{A})$ not being k-DFA-recognizable.

Consider a language L with index N. Recall that the interaction between Refuter and Prover generates words $x \in (\Sigma \cup \{\#\})^\omega$ and $y \in [k]^\omega$. If $k < N$, Refuter can generate x for which the responses of Prover in y must contain a violation of legality or agreement with L. Once a violation is detected, the interaction terminates and it constitutes a certificate: an *informative bad prefix* [**?**] of the safety language of interactions in which Prover's responses are legal and agree with L. We show that the length of certificates generated by offline refuters is at most $O(k^2 \cdot N)$, whereas interaction reduces the length to $O(k^2 + N)$. We show that both bounds are tight. For separation, we describe a refuter that generates certificates of length at most $O(k^2 \cdot |\Sigma| + k \cdot (N_1 + N_2))$, where N_1 and N_2 are the indices of the separated languages.

Our interactive certification protocol has similarities with the interaction that takes place in *learning* of regular languages [**?**], (see recent survey in [**?**]). There, a Learner is tasked to construct a DFA $\mathcal{A}$ for an unknown regular set L by asking a Teacher queries of two types: Membership ("$w \in L$?") and equivalence ("$L(\mathcal{A}) = L$?"). In our setting, Refuter also wants to "learn" the k-DFA for L that Prover claims to possess, but she needs to learn only a fraction of it from Prover – a fraction that reveals that it does not actually recognize L. This is done with a single type of query ("what is the next state?"), which may give Refuter more information than the information gained in the learning setting.

2 Preliminaries

2.1 Automata

A *deterministic automaton on finite words* (DFA, for short) is $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, F \rangle$, where Q is a finite set of states, $q_0 \in Q$ is an initial state, $\delta : Q \times \Sigma \to Q$ is a partial transition function, and $F \subseteq Q$ is a set of final states. We sometimes refer to δ as a relation $\Delta \subseteq Q \times \Sigma \times Q$, with $\langle q, \sigma, q' \rangle \in \Delta$ iff $\delta(q, \sigma) = q'$. A run of $\mathcal{A}$ on a word $w = w_1 \cdot w_2 \cdots w_m \in \Sigma^*$ is the sequence of states $q_0, q_1, \ldots, q_m$ such that $q_{i+1} = \delta(q_i, w_{i+1})$ for all $0 \leq i < m$. The run is accepting if $q_m \in F$. A word $w \in \Sigma^*$ is accepted by $\mathcal{A}$ if the run of $\mathcal{A}$ on w is accepting. The language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts. We define the *size* of $\mathcal{A}$, denoted $|\mathcal{A}|$, as the number of states that $\mathcal{A}$ has. For a language $L \subseteq \Sigma^*$, we use $comp(L)$ to denote the language complementing L, thus $comp(L) = \Sigma^* \setminus L$.

Consider a language $L \subseteq \Sigma^*$. For two finite words h_1 and h_2, we say that h_1 and h_2 are *right L-indistinguishable*, denoted $h_1 \sim_L h_2$, if for every $t \in \Sigma^*$, we have that $h_1 \cdot t \in L$ iff $h_2 \cdot t \in L$. Thus, $\sim_L$ is the Myhill-Nerode right congruence used for minimizing DFAs. For $h \in \Sigma^*$, let $[h]$ denote the equivalence class of h in $\sim_L$ and let $\langle L \rangle$ denote the set of all equivalence classes. When L is regular, the set $\langle L \rangle$ is finite and we use index(L) to denote $|\langle L \rangle|$. The set $\langle L \rangle$ induces the *residual automaton* of L, defined by $\mathcal{R}_L = \langle \Sigma, \langle L \rangle, \Delta_L, [\epsilon], F \rangle$, with $\langle [h], a, [h \cdot a] \rangle \in \Delta_L$ for all $[h] \in \langle L \rangle$ and $a \in \Sigma$. Also, F contains all classes $[h]$ with $h \in L$. The DFA $\mathcal{R}_L$ is well defined and is the unique minimal DFA for L.

Lemma 1. *Consider a regular language L of index N. For every $1 \leq k \leq N$, there is a set $H_k = \{h_1, \ldots, h_k\}$ of words $h_i \in \Sigma^*$ such that $h_i \not\sim_L h_j$ for all $1 \leq i \neq j \leq k$ and $|h_i| \leq k - 1$ for all $1 \leq i \leq k$.*

Proof. Let H be a set of shortest representatives of the classes in $\langle L \rangle$. If every word $h \in H$ has $|h| \leq k - 1$, we can define H_k as an arbitrary subset of size k of H. Otherwise, there exists $h \in H$ with $|h| \geq k$. Let $[h_1], \ldots, [h_{k+1}]$ be the prefix with $k + 1$ states of a simple path in $\mathcal{R}_L$ from $[\epsilon]$ to $[h]$. For every $1 \leq i \leq k + 1$, we have $|h_i| = i - 1$, and we define $H_k = \{h_1, \ldots, h_k\}$. □

Consider a language $L \subseteq \Sigma^\omega$ of infinite words. Here, the language complementing L is $comp(L) = \Sigma^\omega \setminus L$. A finite word $x \in \Sigma^*$ is a *bad prefix* for L if for every $y \in \Sigma^\omega$, we have that $x \cdot y \notin L$. That is, x is a bad prefix if all its extensions are words not in L. A language $L \subseteq \Sigma^\omega$ is a *safety* language if every word not in L has a bad prefix. A language L is a *co-safety* language if $comp(L)$ is safety. Equivalently, every word $w \in L$ has a *good prefix*, namely a prefix $x \in \Sigma^*$ such that for every $y \in \Sigma^\omega$, we have that $x \cdot y \in L$.

2.2 Transducers and Realizability

Consider two finite alphabets Σ_I and Σ_O. For two words $x = x_1 \cdot x_2 \cdots \in \Sigma_I^\omega$ and $y = y_1 \cdot y_2 \cdots \in \Sigma_O^\omega$, we define $x \oplus y$ as the word in $(\Sigma_I \times \Sigma_O)^\omega$ obtained by merging x and y. Thus, $x \oplus y = (x_1, y_1) \cdot (x_2, y_2) \cdots$.

A (Σ_I/Σ_O)-*transducer* models a finite-state system that generates letters in Σ_O while interacting with an environment that generates letters in Σ_I. Formally, a (Σ_I/Σ_O)-transducer is $\mathcal{T} = \langle \Sigma_I, \Sigma_O, \iota, S, s_0, \rho, \tau \rangle$, where $\iota \in \{sys, env\}$ indicates who initiates the interaction – the system or the environment, S is a set of states, $s_0 \in S$ is an initial state, $\rho : S \times \Sigma_I \to S$ is a transition function, and $\tau : S \to \Sigma_O$ is a labeling function on the states. Consider an input word $x = x_1 \cdot x_2 \cdots \in \Sigma_I^\omega$. The *run* of $\mathcal{T}$ on x is the sequence $s_0, s_1, s_2 \ldots$ such that for all $j \geq 0$, we have that $s_{j+1} = \rho(s_j, x_{j+1})$. The *annotation of x by $\mathcal{T}$*, denoted $\mathcal{T}(x)$, depends on ι. If $\iota = sys$, then $\mathcal{T}(x) = \tau(s_0) \cdot \tau(s_1) \cdot \tau(s_2) \cdots \in \Sigma_O^\omega$. Note that the first letter in $\mathcal{T}(x)$ is the output of $\mathcal{T}$ in s_0. This reflects the fact that the system initiates the interaction. If $\iota = env$, then $\mathcal{T}(x) = \tau(s_1) \cdot \tau(s_2) \cdot \tau(s_3) \cdots \in \Sigma_O^\omega$. Note that now, the output in s_0 is ignored, reflecting the fact that the environment initiates the interaction. Then, the *computation* of $\mathcal{T}$ on x is the word $x \oplus \mathcal{T}(x) \in (\Sigma_I \times \Sigma_O)^\omega$.

We say that a (Σ_I/Σ_O)-transducer is *offline* if its behavior is independent of inputs from the environment. Formally, its transition function ρ satisfies $\rho(s, x) = \rho(s, x')$ for all states $s \in S$ and input letters $x, x' \in \Sigma_I$. Note that an offline transducer has exactly one run, and it annotates all words by the same lasso-shaped word $u \cdot v^\omega$, with $u \in \Sigma_O^*$ and $v \in \Sigma_O^+$. We sometimes refer to general transducers as *online* transducers, to emphasize they are not offline.

Consider a ω-regular language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$. We say that L is (Σ_I/Σ_O)-*realizable by the system* if there exists a (Σ_I/Σ_O)-transducer $\mathcal{T}$ with $\iota = sys$ all whose computations are in L. Thus, for every $x \in \Sigma_I^\omega$, we have that $x \oplus \mathcal{T}(x) \in L$. We then say that $\mathcal{T}$ (Σ_I/Σ_O)-*realizes* L. Then, L is (Σ_O/Σ_I)-*realizable by the environment* if there exists a (Σ_O/Σ_I)-transducer $\mathcal{T}$ with $\iota = env$ all whose computations are in L. When Σ_I and Σ_O are clear from the context, we omit them. When the language L is ω-regular, realizability reduces to deciding a game with a regular winning condition. Then, by determinacy of games and due to the existence of finite-memory winning strategies [**?**], we have the following.

Proposition 1. *For every ω-regular language $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$, exactly one of the following holds.*

1. *L is (Σ_I/Σ_O)-realizable by the system.*
2. *$comp(L)$ is (Σ_O/Σ_I)-realizable by the environment.*

3 Proving and Refuting Bounds on DFAs

Consider a regular language $L \subseteq \Sigma^*$ and a bound $k \geq 1$. We view the problem of deciding whether L can be recognized by a k-DFA as the problem of deciding a turn-based two-player game between Refuter and Prover. In each round in the game, Prover provides a letter from a set $[k] = \{1, 2, \ldots, k\}$ that describes the state space of a DFA for L that Prover claims to exist, and Refuter responds with a letter in $\Sigma \cup \{\#\}$, for a special reset letter $\# \notin \Sigma$. Thus, during the interaction, Prover generates a word $y \in [k]^\omega$ and Refuter generates a word $x \in (\Sigma \cup \{\#\})^\omega$. The word x describes an infinite sequence of words in Σ^*, separated by #'s,

and the word y aims to describe runs of the claimed DFA on the words in the sequence.

Below we formalize this intuition. Let $\Sigma' = \Sigma \cup \{\#\}$, for a letter $\# \notin \Sigma$. Consider a (finite or infinite) word $w = x \oplus y \in (\Sigma' \times [k])^* \cup (\Sigma' \times [k])^\omega$. Let $x = x_1 \cdot x_2 \cdots$ and $y = y_1 \cdot y_2 \cdots$. We say that w is *legal* if the following two conditions hold:

1. For all $1 \leq j < |w|$ with $x_j = \#$, we have $y_{j+1} = y_1$.
2. There exists a function $\delta : [k] \times \Sigma \to [k]$ such that $y_{j+1} = \delta(y_j, x_j)$ for all $1 \leq j < |w|$ with $x_j \in \Sigma$.

The first condition ensures that Prover starts all runs in the same state $y_1 \in [k]$, which serves as the initial state in her claimed DFA. The second condition ensures that there exists a deterministic transition relation that Prover follows in all her transitions.

A word w being legal guarantees that Prover follows some k-DFA. We now add conditions on w in order to guarantee that this DFA recognizes L. Consider a position $1 \leq j < |w|$. Let $\#(j) = \max\{j' : (j' < j \text{ and } x_{j'} = \#) \text{ or } j' = 0\}$ be the last position before j in which Refuter generates the reset letter $\#$ (or 0, if no such position exists). When the interaction is in position j, we examine the word w^j that starts at position $\#(j) + 1$ and ends at position $j - 1$. Thus, $w^j = x_{\#(j)+1} \cdot x_{\#(j)+2} \cdots x_{j-1} \in \Sigma^*$. The run that Prover suggests to w^j is then $y_{\#(j)+1}, y_{\#(j)+2}, \ldots, y_j$, and we say that y *maps* w^j to y_j. When y is clear from the context, we also say that Prover maps w^j to y_j. Note that if j_1 and j_2 are such that $w^{j_1} = w^{j_2}$, then w being legal ensures that w^{j_1} and w^{j_2} are mapped to the same state. Now, we say that $w = x \oplus y \in (\Sigma' \times [k])^* \cup (\Sigma' \times [k])^\omega$ *agrees with* L if there exists a set $F \subseteq [k]$ such that for all $1 \leq j < |w|$, Prover maps w^j to an element in F iff $w^j \in L$.

Remark 1. Note that a word w agrees with L iff w agrees with $comp(L)$. Indeed, our definition of agreement with L only guarantees we can define an acceptance condition on top of the claimed k-DFA for either L and $comp(L)$. Since these DFAs dualize each other, they have the same index, and so it makes sense not to distinguish between them in our study. □

Example 1. Let $\Sigma = \{a, b\}$ and $k = 2$. An interaction between Prover and Refuter may generate the prefix of a computation in $(\{a, b, \#\} \times \{1, 2\})^\omega$ described in Table 1. Note that while w fixes $\delta(1, a)$, $\delta(2, a)$, and $\delta(2, b)$, it does not fix $\delta(1, b)$.

In Figure 1 we describe two possible DFAs induced by w and the two possible choices for $\delta(1, b)$.

Consider the language $L_1 \subseteq \{a, b\}^*$ of all words with an even number of a's. Then, w agrees with L_1, since there is $F = \{1\}$ and all w^j with an even number of a's are mapped to F. However, if we consider the language $L_2 \subseteq \{a, b\}^*$ of all words with an even number of b's, there is no F witnessing that w agrees with L_2. Clearly, any $F \subseteq \{2\}$ is not a witness, since $\epsilon \in L_2$, but $1 \notin F$. Moreover, $F = \{1, 2\}$ cannot be a witness, since $ab \notin L_2$, and $F = \{1\}$ is also ruled out, since $a \in L_2$. Thus w does not agree with L_2. □

$w = x \oplus y =$	$(a,1)$	$(b,2)$	$(\#,2)$	$(a,1)$	$(a,2)$	$(a,1)$	$(b,2)$	$(\#,2)$	$(\#,1)$	$(a,1)$	$(a,2)$
$j =$	1	2	3	4	5	6	7	8	9	10	11
$\#(j) =$	0	0	0	3	3	3	3	3	8	9	9
$w^j =$	ϵ	a	ab	ϵ	a	aa	aaa	$aaab$	ϵ	ϵ	a

Table 1. $x \oplus y$ and its analysis.

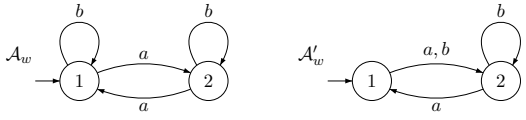

Figure 1. The DFAs $\mathcal{A}_w$ and $\mathcal{A}'_w$ induced by w.

The language $\mathrm{DFA}(L,k) \subseteq (\Sigma' \times [k])^\omega$ of words with correct annotations is then $\mathrm{DFA}(L,k) = \{x \oplus y \in (\Sigma' \times [k])^\omega : x \oplus y \text{ is legal and agrees with } L\}$. Then, $\mathrm{NoDFA}(L,k)$ is the language of words with incorrect annotations, thus $\mathrm{NoDFA}(L,k) = \mathit{comp}(\mathrm{DFA}(L,k))$.

By Proposition 1, we have the following:

Proposition 2. *Consider a language $L \subseteq \Sigma^*$. Exactly one of the following holds:*

- *L can be recognized by a k-DFA, in which case $\mathrm{DFA}(L,k)$ is $(\Sigma'/[k])$-realizable by the system.*
- *L cannot be recognized by a k-DFA, in which case $\mathrm{NoDFA}(L,k)$ is $([k]/\Sigma')$-realizable by the environment.*

By Proposition 2, the language $\mathrm{DFA}(L,k)$ is $(\Sigma'/[k])$-realizable by the system whenever $k \geq \mathrm{index}(L)$. Moreover, as we argue below, a $(\Sigma'/[k])$-transducer $\mathcal{T}$ that realizes $\mathrm{DFA}(L,k)$ induces a k-DFA for L. To see this, consider the word $x \in (\Sigma')^* = w_1 \cdot \# \cdot w_2 \cdots \# \cdot w_{|\Sigma|^k} \cdot \#$ obtained by concatenating all words $w_i \cdot \# \in \Sigma^k \cdot \#$ in some order. Since every transition in a k-DFA is reachable by traversing a word of length at most $k-1$, the computation of $\mathcal{T}$ on x must commit on all the transitions in a transition function $\delta : [k] \times \Sigma \to [k]$, and must also induce a single classification of the states in $[k]$ to accepting and rejecting. Note also that if $k > \mathrm{index}(L)$, the transducer may induce several different DFAs for L.

By Proposition 2, we also have that the language $\mathrm{NoDFA}(L,k)$ is $([k]/\Sigma')$-realizable by the environment whenever $k < \mathrm{index}(L)$. A $([k]/\Sigma')$-transducer that realizes $\mathrm{NoDFA}(L,k)$ is termed an (L,k)-*refuter*.

4 Certifying Bounds on Recognizability

Recall that $\mathrm{DFA}(L,k)$ contains exactly all words that are legal and agree with L. Accordingly, if a word $x \oplus y \in (\Sigma' \times [k])^\omega$ is not in $\mathrm{DFA}(L,k)$, it contains a

violation of legality or agreement with L, and thus has a bad prefix for $\mathrm{DFA}(L,k)$. Formally, we define the language $\mathrm{Violate}(L,k) \subseteq (\Sigma' \times [k])^*$ of words that include a violation of legality or agreement with L as follows.

$$\begin{aligned}\mathrm{Violate}(L,k) = \{x \oplus y : &\text{ there is } j \geq 1 \text{ such that } x_j = \# \text{ and } y_{j+1} \neq y_1, \text{ or}\\ &\text{there are } j_1, j_2 \geq 1 \text{ such that}\\ &y_{j_1} = y_{j_2}, x_{j_1} = x_{j_2}, \text{ and } y_{j_1+1} \neq y_{j_2+1},\\ &\text{or } w^{j_1} \in L, w^{j_2} \notin L \text{ and } y_{j_1} = y_{j_2}\}.\end{aligned}$$

Note that while all the words in $\mathrm{Violate}(L,k)$ are bad prefixes for $\mathrm{DFA}(L,k)$, there are bad prefixes for $\mathrm{DFA}(L,k)$ that are not in $\mathrm{Violate}(L,k)$. For example, if $L = \{a^{2n} : n \geq 0\}$, then the word $(a,1)$ is a bad prefix for $\mathrm{DFA}(L,1)$, as both $(a,1)(a,1)$ and $(a,1)(\#,1)$, which are the only possible extensions of $(a,1)$ by a single letter, are in $\mathrm{Violate}(L,1)$, yet $(a,1)$ itself is not in $\mathrm{Violate}(L,1)$. Formally, using the terminology of [**?**], the language $\mathrm{Violate}(L,k)$ contains all the *informative bad prefixes* of $\mathrm{DFA}(L,k)$, namely these that contain an explanation to the prefix being bad. Since every infinite word not in $\mathrm{DFA}(L,k)$ has a bad prefix in $\mathrm{Violate}(L,k)$, then restricting attention to bad prefixes in $\mathrm{Violate}(L,k)$ is appropriate in the context of certificates. Also, as we discuss in Remark 2, a bad prefix of $\mathrm{DFA}(L,k)$ that is not informative can be made informative by concatenating to it any letter in $\Sigma' \times [k]$.

Remark 2. Surprisingly, extending a bad prefix of $\mathrm{DFA}(L,k)$ by any letter of $\Sigma' \times [k]$ transforms it to an informative bad prefixes, i.e., makes it an element of $\mathrm{Violate}(L,k)$: Let $w = (x_1,y_1)\cdots(x_n,y_n)$ be a bad prefix. In particular, we have $(x_1,y_1)\cdots(x_n,y_n)\cdot(\#,y_{n+1})\cdot(\#,y_1)^\omega \notin \mathrm{DFA}(L,k)$ for all $y_{n+1} \in [k]$. Since continuing a word with $(\#,y_1)$ after a preceding $\#$ does not impact legality or agreement with L, the word $w' = (x_1,y_1)\cdots(x_n,y_n)\cdot(\#,y_{n+1})$ must include a violation of legality or agreement with L and thus $w' \in \mathrm{Violate}(L,k)$. Lastly, since the definition of $\mathrm{Violate}(L,k)$ does not refer to the Σ' component of the last letter read, we can replace $\#$ by any letter of Σ' and thus have shown that any letter of $\Sigma' \times [k]$ transforms a bad prefix to an informative bad prefix. □

Refuting recognizability of L by a k-DFA, we consider two approaches. In the first, we consider the interaction of Prover with an offline (L,k)-refuter. Such a refuter has to generate a word $x \in (\Sigma')^*$ such that for all $y \in [k]^{|x|}$, we have that $x \oplus y \in \mathrm{Violate}(L,k)$. We call x a *universal informative bad prefix* (see [**?**] for a study of bad prefixes for safety languages in an interactive setting). In the second approach, we consider the interaction of Prover with an online (L,k)-refuter. There, the goal is to associate every sequence $y \in [k]^\omega$ that is generated by Prover with a sequence $x \in (\Sigma')^\omega$ such that $x \oplus y$ has a prefix in $\mathrm{Violate}(L,k)$. In Sections 4.1 and 4.2 we compare the two approaches in terms of the length of the certificate (namely the word in $\mathrm{Violate}(L,k)$) that they generate.

4.1 Certification with Offline Refuters

Recall that a word $x \in (\Sigma')^*$ is a *universal informative bad prefix* for $\mathrm{DFA}(L,k)$ if for all $y \in [k]^{|x|}$, we have that $x \oplus y \in \mathrm{Violate}(L,k)$.

Theorem 1. *Consider a regular language $L \subseteq \Sigma^*$ and let $N = \text{index}(L)$. For every $k < N$, the length of a shortest universal informative bad prefix for $\text{DFA}(L, k)$ is at most $O(k^2 \cdot N)$. This bound is tight: There is a family of regular languages $L_1, L_2, \ldots$ such that for every $n \geq 1$, the length of a shortest universal informative bad prefix for $\text{DFA}(L_n, N_n - 1)$ is $\Omega(N_n^3)$, where $N_n = \text{index}(L_n)$.*

Proof. We start with the upper bound and construct, for every $k < N$, a universal informative bad prefix for $\text{DFA}(L, k)$ of length $O(k^2 \cdot N)$.

Let $H = \{h_1, \ldots, h_{k+1}\}$ be representatives of $k + 1$ distinct Myhill-Nerode classes. Since $k < N$, such a set H exists. Moreover, by Lemma 1, we can assume that $|h_i| \leq k$, for all $1 \leq i \leq k+1$. For each pair $\langle h_i, h_j \rangle$, there is a distinguishing tail $t_{i,j}$ of length at most N. Let x be the concatenation of all words of the form $h_i \cdot t_{i,j} \cdot \#$ and $h_j \cdot t_{i,j} \cdot \#$, for all pairs. There are $k \cdot (k+1)$ such words, each of length at most $k + N + 1$, so $|x| \leq (k + N + 1) \cdot k \cdot (k + 1)$, which is $O(k^2 \cdot N)$. Below we prove that x is a universal informative bad prefix.

Let $y \in [k]^{|x|}$. For every $h_i \in H$, the subword $\# \cdot h_i$ appears in x, so y maps h_i to some element in $[k]$. By the pigeonhole principle, there are two distinct words h_i and h_j such that y maps both words to the same element. If $x \oplus y$ is legal, the transitions are consistent, so y maps both $h_i \cdot t_{i,j}$ and $h_j \cdot t_{i,j}$ to the same state. Then, however, as exactly one of $h_i \cdot t_{i,j}$ and $h_j \cdot t_{i,j}$ is in L, there is no $F \subseteq [k]$ that satisfies the condition of agreement with L, and so $x \oplus y \in \text{Violate}(L, k)$. Hence, x is a universal informative bad prefix for $\text{DFA}(L, k)$.

For a matching lower bound, we describe a family of regular languages $L_1, L_2, \ldots$ such that for every $n \geq 1$, the length of a shortest universal informative bad prefix for $\text{DFA}(L_n, N_n - 1)$ is $\Omega(N_n^3)$, where $N_n = \text{index}(L_n)$. For $n \geq 1$, let $\Sigma_n = \{a, b_1, \ldots, b_n\}$ and consider the language $L_n = \{a^n b_i^2 : 1 \leq i \leq n\}$. Let $\mathcal{A}_n$ be a minimal DFA for L_n. For example, $L_3 = \{aaab_1b_1, aaab_2b_2, aaab_3b_3\}$, and the DFA $\mathcal{A}_3$ for L_3 appears in Figure 2.

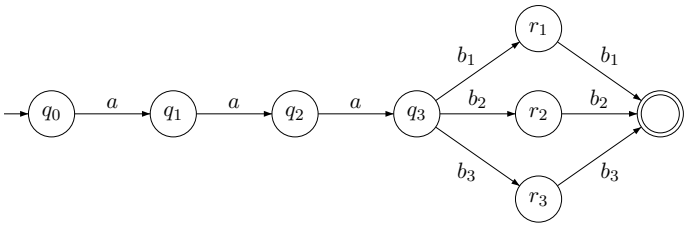

Figure 2. A DFA for L_3.

It is easy to see that $\text{index}(L_n) = N_n = 2n+3$, corresponding to (see Figure 2) $n + 1$ states $q_0, \ldots, q_n$, n states $r_1, \ldots, r_n$, an accepting state, and a rejecting sink, which we omit from the figure.

Let $k = N_n - 1$, and consider some prefix $x \in (\Sigma')^*$. For $1 \leq i \neq j \leq n$, the words $a^n b_i$ and $a^n b_j$ belong to different Myhill-Nerode classes, corresponding to

the states r_i and r_j, respectively. The distinguishing tails are b_i and b_j. We claim that if x does not contain the subword $a^n b_i b_j$ or $a^n b_j b_i$, then there is $y \in [k]^{|x|}$ such that $x \oplus y \notin \text{Violate}(L, k)$, and so x is not a universal informative bad prefix for $\text{DFA}(L, k)$. To see this, consider the word $y \in [k]^{|x|}$ constructed by following the DFA obtained from $\mathcal{A}_n$ by merging the states r_i and r_j. We can choose $x' = \#^\omega$ and $y' \in [k]^\omega$ such that y'_1 is consistent with the transitions in y, and $y'_j = y_1$ for all $j \geq 2$. If $x = y = \epsilon$, we can choose $x' \oplus y' = (\#, 1)^\omega$. Then, $(x \cdot x') \oplus (y \cdot y') \in \text{DFA}(L, k)$, and so $x \oplus y$ is not an informative bad prefix for $\text{DFA}(L, k)$.

Hence, if x is a universal informative bad prefix for $\text{DFA}(L, k)$, then for every $1 \leq i \neq j \leq n$, it contains the subwords $a^n b_i b_j$ or $a^n b_j b_i$, which are of length $n + 2$. There are $n \cdot (n-1)/2$ such subwords and they are disjoint. Therefore, $|x| \geq (n+2) \cdot n \cdot (n-1)/2$, which is $\Omega(N_n^3)$. □

4.2 Certification with Online Refuters

We now consider refuters that take Prover's choices into account when outputting letters. We show that this capability allows an interactive refuter to win in fewer rounds than an offline refuter.

Theorem 2. *Consider a regular language $L \subseteq \Sigma^*$ and let $N = \text{index}(L)$. For every $k < N$, there exists an (L, k)-refuter that generates a word in $\text{Violate}(L, k)$ within $O(k^2 + N)$ rounds. This bound is tight: There is a family of regular languages $L_1, L_2, \ldots$ such that for every $n \geq 1$, every (L, k)-refuter needs at least $\Omega(N_n^2)$ rounds to construct a word in $\text{Violate}(L_n, N_n - 1)$, where $N_n = \text{index}(L_n)$.*

Proof. We start with the upper bound, by describing a winning strategy. As in the offline case, let $H = \{h_1, \ldots, h_{k+1}\}$ be representatives of distinct Myhill-Nerode classes, each of length at most k. Unlike the offline case, where Refuter outputs all pairs of heads and distinguishing tails, here a single pair suffices to achieve the same effect. Refuter starts the interaction by outputting $h_1 \cdot \# \cdots \# \cdot h_{k+1} \cdot \#$. By the pigeonhole principle, there are distinct words h_i and h_j that are mapped by Prover to the same state. Refuter then outputs $h_i \cdot t_{i,j} \cdot \# \cdot h_j \cdot t_{i,j} \cdot \#$. If Prover does not violate the conditions of legality, it maps $h_i \cdot t_{i,j}$ and $h_j \cdot t_{i,j}$ to the same state. Exactly one of them is in L, so there is no $F \subseteq [k]$ that can satisfy agreement with L, and so the generated word is in $\text{Violate}(L, k)$. We now analyze its length. Recall that Refuter first outputs $k+1$ words of length at most k each, separated by #'s, and then two words of length at most $k + N$ each, again separated by #. Thus, the length of the prefix is $k(k+1)+2(k+N)+k+3$, which is $O(k^2 + N)$.

For a matching lower bound, we describe a family of regular languages $L_1, L_2 \ldots$ such that for every $n \geq 1$, every refuter needs at least $\Omega(N_n^2)$ rounds to generate a word in $\text{Violate}(L_n, N_n - 1)$, where $N_n = \text{index}(L_n)$. Consider the DFA $\mathcal{A}_n$ from the offline lower bound, again with $k = N_n - 1$. We claim that $\Omega(N_n^2)$ rounds are required to generate a word in $\text{Violate}(L_n, N_n - 1)$.

Let $x \in (\Sigma')^*$ be the word generated by Refuter. Assume there exists $1 \leq i \leq n$ such that the subword $a^n b_i$ does not appear in x. The state corresponding to $a^n b_i$ is r_i. Hence, Prover can follow the DFA obtained by removing the state r_i from $\mathcal{A}_n$ without violating legality or agreement with L. Therefore, in order to guarantee a generation of a word in Violate$(L_n, N_n - 1)$, Refuter must output all the words $a^n b_1, \ldots, a^n b_n$ in some order. Each of these n words has length $n+1$, and they are disjoint. Their total length is therefore at least $n \cdot (n+1)$, which is $\Omega(N_n^2)$. □

Remark 3. **Fixed alphabet** In the proofs of Theorems 1 and 2, we use languages L_n over an alphabet Σ_n that depends on n. By replacing the letters $b_1, \ldots, b_n$ by words in $\{a, b\}^{\lfloor \log n \rfloor}$, one gets languages over the fixed alphabet $\Sigma = \{a, b\}$ that exhibit the claimed lower bounds for both online and offline refuters. □

4.3 Optimal Survival Strategies for Provers

Assume that L is not k-DFA recognizable. Then, there is an (L, k)-refuter, and Refuter is going to win a game against Prover and generate a word in Violate(L, k). Suppose that Prover aims at prolonging the interaction. It is tempting to think that the following greedy strategy is optimal for such an objective: Prover follows the transitions of $\mathcal{R}_L$. If $k <$ index(L), then Prover may be forced to deviate from $\mathcal{R}_L$ and make a "mistake", namely choose to output one of the k states that have already been exposed. Using this strategy, Prover can prolong the game at least until $k+1$ different states are exposed. The following example shows that this strategy is not necessarily best at prolonging the game as long as possible (no matter how clever the choice when a "mistake" is forced is).

Example 2. For $n \geq 1$, consider the language $L_n = \{w : w_1 = b \text{ or } |w| = n\}$ over $\Sigma = \{a, b\}$.

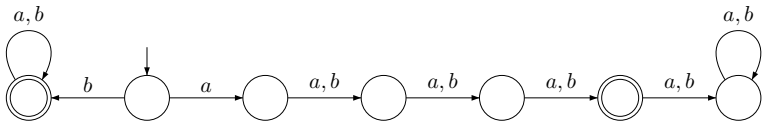

Figure 3. A DFA for L_4.

Denote the canonical DFA for L_n by $\mathcal{A}_n$. For example, $\mathcal{A}_4$ appears in Figure 3. The number of states is $n+3$. Let $k = n+2$. We claim that if Prover follows $\mathcal{A}_n$, then $x = a^{n+1} \cdot \# \cdot b \cdot a$ induces a bad prefix $x \oplus y$ for L_n. The first word, a^{n+1}, forces Prover to expose all $n+2$ states, after which it cannot have an accepting sink. Then, $\# \cdot b$ forces Prover to choose an existing state instead of an accepting sink. To prolong the game, it chooses the only accepting state, and

then the last a ends the game. The number of rounds needed to win against this prover is at most $n + 4$.

Prover can do better than $n + 4$ rounds. Let $L'_n = \{w : w_1 = b \text{ or } (w_1 = a \text{ and } |w| = 0 \bmod n)\}$ and let $\mathcal{B}_n = \mathcal{R}_{L'_n}$ be the minimal DFA for L'_n. For example, $\mathcal{B}_4$ appears in Figure 4. The shortest possible word length on which $\mathcal{A}_n$ and $\mathcal{B}_n$ disagree is $2n$ (for example, $a^{2n} \in L'_n \setminus L_n$), which is better than $n + 4$. It can be shown that an (L_n, k)-refuter can generate a bad prefix for L_n in at most $2n$ rounds against all provers, so $\mathcal{B}_n$ is optimal in that sense. □

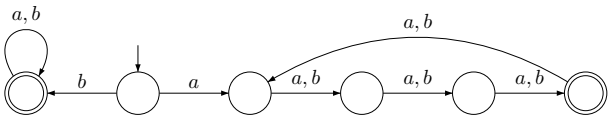

Figure 4. The DFA $\mathcal{B}_4$.

5 Bounds on DFA Separation

Consider three languages $L_1, L_2, L \subseteq \Sigma^*$. We say that L is a *separator* for $\langle L_1, L_2 \rangle$ if $L_1 \subseteq L$ and $L \cap L_2 = \emptyset$. Equivalently, $L_1 \subseteq L \subseteq comp(L_2)$. For $k \geq 1$, we say that a pair of languages $\langle L_1, L_2 \rangle$ is *k-DFA-separable* iff there is a k-DFA $\mathcal{A}$ such that $L(\mathcal{A})$ separates $\langle L_1, L_2 \rangle$. We extend the definition to DFAs and say that two DFAs $\mathcal{A}_1$ and $\mathcal{A}_2$ are separated by a DFA $\mathcal{A}$, if their languages are separated by $L(\mathcal{A})$.

In this section we study refuting and certifying bounds on DFA separation. We first give proofs that deciding (strict and non-strict) k-DFA-separability, is NP-complete. The problem being NP-hard suggests that there is no clean theory of equivalence classes that is the base for offline certification. We continue and describe interactive certification protocol for k-DFA-separability.

5.1 Hardness of Separation

The following Theorem 3 is considered by the literature (e.g., [**?**]) to be a consequence of [**?**]. Since we also investigate the strict-separation case and there is a progression of techniques, we describe below an alternative and explicit proof.

Theorem 3. *Given DFAs $\mathcal{A}_1$ and $\mathcal{A}_2$, and a bound $k \geq 1$, deciding whether $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$ is k-DFA-separable is NP-complete.*

Proof. Membership in NP is easy, as given a candidate separator $\mathcal{A}$ of size k, we can verify that $L(\mathcal{A}_1) \subseteq L(\mathcal{A})$ and $L(\mathcal{A}) \cap L(\mathcal{A}_2) = \emptyset$ in polynomial time. Note that if $k \geq \text{index}(L(\mathcal{A}_1))$, then $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$ is k-DFA-separable by $\mathcal{A}_1$. Thus, we can assume that $k < \text{index}(L(\mathcal{A}_1))$, and so membership in NP applies also for the case k is given in binary.

For NP-hardness, we reduce from the *DFA identification* problem. Recall that there, given sets $S_1, S_2 \subseteq \Sigma^*$ of positive and negative words, and a bound $k \geq 1$, we seek a k-DFA that accepts all words in S_1 and no word in S_2. By [**?**], DFA identification is NP-complete. Given S_1, S_2, and k, our reduction constructs DFAs $\mathcal{A}_1$ and $\mathcal{A}_2$ such that $L(\mathcal{A}_1) = S_1$ and $L(\mathcal{A}_2) = S_2$. Clearly, a k-DFA solves the DFA identification problem for S_1, S_2, and k, iff it solves the k-DFA-separation of $\mathcal{A}_1$ and $\mathcal{A}_2$.

Constructing a DFA $\mathcal{A}_S$ such that $L(\mathcal{A}_S) = S$, for some finite set $S \subseteq \Sigma^*$ can be done in polynomial time, by traversing prefixes of words in S. Formally, we define $\mathcal{A}_S = \langle \Sigma, Q, q_0, \delta, F \rangle$, where $Q = \{w : w \text{ is a prefix of a word in } S\}$, $q_0 = \epsilon$, and for all $w \in Q$ and $\sigma \in \Sigma$, we have that $\delta(w, \sigma) = w \cdot \sigma$ if $w \cdot \sigma \in S$, and $\delta(w, \sigma)$ is undefined otherwise. Finally, $F = S$. It is easy to see that $L(\mathcal{A}_S) = S$ and that $|\mathcal{A}_S| \leq \sum_{w \in S} |w|$. □

Consider three languages $L_1, L_2, L \subseteq \Sigma^*$. We say that L is a *strict separator* for $\langle L_1, L_2 \rangle$ if $L_1 \subset L$, $L \cap L_2 = \emptyset$, and $L \cup L_2 \subset \Sigma^*$. Equivalently, $L_1 \subset L \subset \mathit{comp}(L_2)$. For $k \geq 1$, we say that a pair of languages $\langle L_1, L_2 \rangle$ is *k-DFA-strictly-separable* iff there is a k-DFA $\mathcal{A}$ such that $L(\mathcal{A})$ strictly separates $\langle L_1, L_2 \rangle$. Again, we extend the definition to DFAs.

Theorem 4. *Given DFAs $\mathcal{A}_1$ and $\mathcal{A}_2$, and a bound $k \geq 1$, deciding whether $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$ is k-DFA-strictly-separable is NP-complete.*

Proof. We start with membership in NP. As in the proof of Theorem 3, a witness k-DFA $\mathcal{A}$ can be checked in polynomial time. However, if k is given in binary and greater than $\text{index}(L(\mathcal{A}_1))$ and $\text{index}(L(\mathcal{A}_2))$, we cannot base a separator on $\mathcal{A}_1$ or $\mathcal{A}_2$. We fill this gap by showing that if a DFA strictly separates $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$, then there also exists one that is polynomial in $|\mathcal{A}_1|$ and $|\mathcal{A}_2|$.

Assume that $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$ are strictly separable, and let $T = \mathit{comp}(L(\mathcal{A}_1) \cup L(\mathcal{A}_2))$. Note that $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$ being strictly separable implies that $|T| > 1$. Let $\mathcal{A}_T$ be a minimal DFA for T. Note that $|\mathcal{A}_T| \leq |\mathcal{A}_1| \cdot |\mathcal{A}_2|$. Consider a word $w \in T$ that is accepted along a simple path in $\mathcal{A}_T$. Thus, $|w|$ is polynomial in $|\mathcal{A}_T|$. Consider a DFA $\mathcal{A}_1^w$ with $L(\mathcal{A}_1^w) = L(\mathcal{A}_1) \cup \{w\}$. Note that $|\mathcal{A}_1^w|$ is polynomial in $|\mathcal{A}_1|$ and $|w|$. It is not hard to see that $\mathcal{A}_1^w$ is a strict separator for $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$. Indeed, $L(\mathcal{A}_1^w)$ strictly contains $L(\mathcal{A}_1)$, it is contained in $\mathit{comp}(L(\mathcal{A}_2))$, and as $|T| > 1$, the latter containment is strict. Hence, $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$ are strictly separable by a DFA that is polynomial in $|\mathcal{A}_1|$ and $|\mathcal{A}_2|$.

For NP-hardness, we describe a reduction from k-DFA-separability, proved to be NP-hard in Theorem 3. Consider two DFAs $\mathcal{A}_1$ and $\mathcal{A}_2$ over Σ, and assume that $0 \notin \Sigma$. Assume also that $L(\mathcal{A}_1), L(\mathcal{A}_2) \neq \emptyset$, and that $L(\mathcal{A}_1), L(\mathcal{A}_2)$ are finite, and thus have rejecting sinks. Clearly, k-DFA-separability is NP-hard also in this case. Let $\mathcal{A}_1'$ and $\mathcal{A}_2'$ be DFAs obtained from $\mathcal{A}_1$ and $\mathcal{A}_2$ by extending the alphabet to $\Sigma \cup \{0\}$ and adding a transition labeled 0 from every state to the rejecting sink. Note that $L(\mathcal{A}_1') = L(\mathcal{A}_1)$ and $\mathit{comp}(L(\mathcal{A}_2')) = (\Sigma \cup \{0\})^* \setminus L(\mathcal{A}_2)$. We prove that for every $k \geq 1$, we have that $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$ is k-DFA-separable iff $\langle \mathcal{A}_1', \mathcal{A}_2' \rangle$ is k-DFA-strictly-separable.

Assume that there is a k-DFA $\mathcal{A} = \langle \Sigma, Q, \delta, q_0, F \rangle$ that separates $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$. Let $\mathcal{A}'$ be the k-DFA obtained from $\mathcal{A}$ by extending the alphabet to $\Sigma \cup \{0\}$, and adding a transition labeled 0 from every state to q_0. It is easy to see that $L(\mathcal{A}') = (\Sigma^* \cdot 0)^* \cdot L(\mathcal{A})$, and so $L(\mathcal{A}) \subseteq L(\mathcal{A}')$. Also, whenever $L(\mathcal{A})$ is not empty, this containment is strict. Indeed, each word $w \in L(\mathcal{A})$ induces the word $0 \cdot w \in L(\mathcal{A}') \setminus L(\mathcal{A})$. Hence, as $\emptyset \neq L(\mathcal{A}_1) \subseteq L(\mathcal{A})$, we have that $L(\mathcal{A}'_1) \subset L(\mathcal{A}')$. In addition, as $L(\mathcal{A}) \cap L(\mathcal{A}_2) = \emptyset$, then clearly $L(\mathcal{A}') \cap L(\mathcal{A}'_2) = \emptyset$. Moreover, as $L(\mathcal{A}_2) \neq \emptyset$, there is a word $w \in L(\mathcal{A}_2)$. Then, $w \notin L(\mathcal{A})$ and so $w \cdot 0 \cdot w \notin L(\mathcal{A}')$. In addition, $w \cdot 0 \cdot w \notin L(\mathcal{A}'_2)$. Thus, $L(\mathcal{A}') \cup L(\mathcal{A}'_2) \subset (\Sigma \cup \{0\})^*$, and we are done.

For the other direction, assume there is a k-DFA $\mathcal{A}'$ that strictly separates $\langle \mathcal{A}'_1, \mathcal{A}'_2 \rangle$. Consider the k-DFA $\mathcal{A}$ obtained from $\mathcal{A}'$ by removing all transitions labeled 0 and changing the alphabet to Σ. Every word in $L(\mathcal{A}_1)$ is also in $L(\mathcal{A}')$, and it does not contain 0. So, $L(\mathcal{A}_1) \subseteq L(\mathcal{A})$. Similarly, every word in $L(\mathcal{A}')$ that does not contain 0 and is not in $L(\mathcal{A}'_2)$, is also not in $L(\mathcal{A}_2)$. Therefore, $L(\mathcal{A}) \cap L(\mathcal{A}_2) = \emptyset$. Hence, $\mathcal{A}$ separates $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$, and we are done. □

The reduction described in the proof of Theorem 4 can be used to prove NP-completeness also for *one-sided strict separation* problems. Formally, we have the following, which generalizes Conjecture 1 from [**?**].

Theorem 5. *Given DFAs $\mathcal{A}_1$ and $\mathcal{A}_2$, and a bound $k \geq 1$, the problems of deciding whether there exists a k-DFA $\mathcal{A}$ such that $L(\mathcal{A}_1) \subset L(\mathcal{A}) \subseteq comp(L(\mathcal{A}_2))$ and whether there exists a k-DFA $\mathcal{A}'$ such that $L(\mathcal{A}_1) \subseteq L(\mathcal{A}') \subset comp(L(\mathcal{A}_2))$ are NP-complete.*

Proof. We start with the problem of deciding whether there exists a k-DFA $\mathcal{A}$ such that $L(\mathcal{A}_1) \subseteq L(\mathcal{A}) \subset comp(L(\mathcal{A}_2))$.

Membership in NP is easy, as we can verify each containment in polynomial time. Note that, as in the proof of Theorem 3, if $k \geq \text{index}(L(\mathcal{A}_1))$, then $\mathcal{A}_1$ is a witness. Thus, we can assume that $k < \text{index}(L(\mathcal{A}_1))$, and so membership in NP applies also for the case k is given in binary.

For NP-hardness, we follow the same reduction from k-DFA-separability described in the proof of Theorem 4, and argue it is valid also for our problem. Assume that there is a k-DFA $\mathcal{A} = \langle \Sigma, Q, \delta, q_0, F \rangle$ that separates $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$. Let $\mathcal{A}'$ be the k-DFA obtained from $\mathcal{A}$ by extending the alphabet to $\Sigma \cup \{0\}$, and adding a transition labeled 0 from every state to q_0. It is easy to see that $L(\mathcal{A}') = (\Sigma^* \cdot 0)^* \cdot L(\mathcal{A})$, and so $L(\mathcal{A}) \subseteq L(\mathcal{A}')$. Therefore, we have $L(\mathcal{A}'_1) \subseteq L(\mathcal{A}')$. In addition, as $L(\mathcal{A}) \cap L(\mathcal{A}_2) = \emptyset$, then clearly $L(\mathcal{A}') \cap L(\mathcal{A}'_2) = \emptyset$. Moreover, as $L(\mathcal{A}_2) \neq \emptyset$, there is a word $w \in L(\mathcal{A}_2)$. Then, $w \notin L(\mathcal{A})$ and so $w \cdot 0 \cdot w \notin L(\mathcal{A}')$. In addition, $w \cdot 0 \cdot w \notin L(\mathcal{A}'_2)$. Thus, $L(\mathcal{A}') \cup L(\mathcal{A}'_2) \subset (\Sigma \cup \{0\})^*$, and we are done.

For the other direction, assume there is a k-DFA $\mathcal{A}'$ such that $L(\mathcal{A}_1) \subseteq L(\mathcal{A}) \subset comp(L(\mathcal{A}_2))$. Consider the k-DFA $\mathcal{A}$ obtained from $\mathcal{A}'$ by removing all transitions labeled 0 and changing the alphabet to Σ. Every word in $L(\mathcal{A}_1)$ is also in $L(\mathcal{A}')$, and it does not contain 0. So, $L(\mathcal{A}_1) \subseteq L(\mathcal{A})$. Similarly, every

word in $L(\mathcal{A}')$ that does not contain 0 and is not in $L(\mathcal{A}'_2)$, is also not in $L(\mathcal{A}_2)$. Therefore, $L(\mathcal{A}) \cap L(\mathcal{A}_2) = \emptyset$. Hence, $\mathcal{A}$ separates $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$, and we are done.

Now, for the problem of deciding whether there exists a k-DFA $\mathcal{A}$ such that $L(\mathcal{A}_1) \subset L(\mathcal{A}) \subseteq \mathit{comp}(L(\mathcal{A}_2))$, note that the latter condition is equivalent to $L(\mathcal{A}_2) \subseteq \mathit{comp}(L(\mathcal{A})) \subset \mathit{comp}(L(\mathcal{A}_1))$, which is NP-complete by the above.

5.2 Certifying Bounds on Separation

Consider two regular languages $L_1, L_2 \subseteq \Sigma^*$ and a bound $k \geq 1$. Certifying bounds on separation, we again consider a turn-based two-player game between Prover and Refuter. This time we are interested in whether L_1 and L_2 can be separated by a k-DFA. Consider a word $x \oplus y \in (\Sigma' \times [k])^\omega$. We say that $x \oplus y$ *agrees with* $\langle L_1, L_2 \rangle$ if there exists $F \subseteq [k]$ such that for every $j \geq 1$, if $w^j \in L_1$, then Prover maps w^j to F and if $w^j \in L_2$, then Proven does not map w^j to F.

Accordingly, we define the language $\mathrm{SepDFA}(L_1, L_2, k) \subseteq (\Sigma' \times [k])^\omega$ of words with correct annotations as follows:

$$\mathrm{SepDFA}(L_1, L_2, k) = \{x \oplus y : x \oplus y \text{ is legal and agrees with } \langle L_1, L_2 \rangle\}.$$

Then, $\mathrm{NoSepDFA}(L_1, L_2, k) = \mathit{comp}(\mathrm{SepDFA}(L_1, L_2, k))$ is the language of all words with incorrect annotations.

Proposition 3. *Consider two regular languages $L_1, L_2 \subseteq \Sigma^*$ and $k \geq 1$. Exactly one of the following holds:*

- *$\langle L_1, L_2 \rangle$ is k-DFA-separable, in which case $\mathrm{SepDFA}(L_1, L_2, k)$ is $(\Sigma'/[k])$-realizable by the system.*
- *$\langle L_1, L_2 \rangle$ is not k-DFA-separable, in which case $\mathrm{NoSepDFA}(L_1, L_2, k)$ is $([k]/\Sigma')$-realizable by the environment.*

A transducer that $([k]/\Sigma')$-realizes $\mathrm{NoSepDFA}(L, k)$ is termed an (L_1, L_2, k)-refuter, and we seek refuters that generate short certificates. As has been the case in Section 4, such a certificate is an informative bad prefix for $\mathrm{SepDFA}(L_1, L_2, k)$. Formally, we define the language $\mathrm{Violate}(L_1, L_2, k) \subseteq (\Sigma' \times [k])^*$ of words that include a violation of legality or agreement with L_1 and L_2 as follows.

$$\begin{aligned}\mathrm{Violate}(L_1, L_2, k) = \{x \oplus y : &\text{ there is } j \geq 1 \text{ such that } x_j = \# \text{ and } y_{j+1} \neq y_1, \text{ or}\\ &\text{there are } j_1, j_2 \geq 1 \text{ such that}\\ &y_{j_1} = y_{j_2}, x_{j_1} = x_{j_2}, \text{ and } y_{j_1+1} \neq y_{j_2+1},\\ &\text{or } w^{j_1} \in L_1, w^{j_2} \in L_2, \text{ and } y_{j_1} = y_{j_2}\}.\end{aligned}$$

Before constructing an (L_1, L_2, k)-refuter that generates short certificates, we first need some notations and observations. Let $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, F \rangle$ and $\mathcal{A}' = \langle \Sigma, Q', q'_0, \delta', F' \rangle$ be DFAs. We define the set $F_{\mathcal{A},\mathcal{A}'}$ of states of $\mathcal{A}$ that are reachable by traversing a word in $L(\mathcal{A}')$. Formally, $q \in F_{\mathcal{A},\mathcal{A}'}$ iff there is $w \in L(\mathcal{A}')$ such that $\delta^*(q_0, w) = q$, where δ^* is the extension of δ to words. Note that $F_{\mathcal{A},\mathcal{A}'}$ does not depend on the acceptance condition of $\mathcal{A}$.

Lemma 2. *For every DFAs $\mathcal{A}$ and $\mathcal{A}'$, we have that $L(\mathcal{A}') \subseteq L(\mathcal{A})$ iff $F_{\mathcal{A},\mathcal{A}'} \subseteq F$, and $L(\mathcal{A}) \cap L(\mathcal{A}') = \emptyset$ iff $F_{\mathcal{A},\mathcal{A}'} \subseteq Q \setminus F$.*

Proof. We start with the first claim. If $F_{\mathcal{A},\mathcal{A}'} \subseteq F$, then for every word $w \in L(\mathcal{A}')$, we have that $\delta^*(q_0, w) \in F$, and so $w \in L(\mathcal{A})$ and $L(\mathcal{A}') \subseteq L(\mathcal{A})$. If $F_{\mathcal{A},\mathcal{A}'} \not\subseteq F$, then there exists a word $w \in L(\mathcal{A}')$ such that $\delta^*(q_0, w) \in Q \setminus F$. Then, $w \in L(\mathcal{A}') \setminus L(\mathcal{A})$, and so $L(\mathcal{A}') \not\subseteq L(\mathcal{A})$.

For the second claim, note that $L(\mathcal{A}) \cap L(\mathcal{A}') = \emptyset$ iff $L(\mathcal{A}') \subseteq \mathit{comp}(L(\mathcal{A}))$. Let $\tilde{\mathcal{A}}$ be $\mathcal{A}$ with $Q \setminus F$ being the set of accepting states. By the first claim, we have that $L(\mathcal{A}') \subseteq L(\tilde{\mathcal{A}})$ iff $F_{\tilde{\mathcal{A}},\mathcal{A}'} \subseteq Q \setminus F$. Since $\mathcal{A}$ and $\tilde{\mathcal{A}}$ differ only in the acceptance condition, $F_{\tilde{\mathcal{A}},\mathcal{A}'} = F_{\mathcal{A},\mathcal{A}'}$, and so we are done. □

Lemma 2 implies the following characterization of separability by a DFA with a given structure:

Theorem 6. *Consider DFAs $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}$. Let $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \emptyset \rangle$. For a set $F \subseteq Q$, define $\mathcal{A}_F = \langle \Sigma, Q, q_0, \delta, F \rangle$. Then, $F_{\mathcal{A},\mathcal{A}_1} \cap F_{\mathcal{A},\mathcal{A}_2} = \emptyset$ iff there exists a set $F \subseteq Q$ such that $\mathcal{A}_F$ separates $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$.*

Proof. By Lemma 2, the DFA $\mathcal{A}_F$ is a separator for $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$ iff $F_{\mathcal{A},\mathcal{A}_1} \subseteq F$ and $F_{\mathcal{A},\mathcal{A}_2} \subseteq Q \setminus F$. If $F_{\mathcal{A},\mathcal{A}_1} \cap F_{\mathcal{A},\mathcal{A}_2} = \emptyset$, then $F = F_{\mathcal{A},\mathcal{A}_1}$ satisfies both containments. In the other direction, if there exists a set F that satisfies both containments, then $F_{\mathcal{A},\mathcal{A}_1} \cap F_{\mathcal{A},\mathcal{A}_2} = \emptyset$. □

Consider a DFA $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \emptyset \rangle$. If there is no set F such that $\mathcal{A}_F$ is a separator for $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$, there exists a state $q \in F_{\mathcal{A},\mathcal{A}_1} \cap F_{\mathcal{A},\mathcal{A}_2}$. That is, there are words $w_1 \in L(\mathcal{A}_1)$ and $w_2 \in L(\mathcal{A}_2)$ such that $\delta^*(q_0, w_1) = \delta^*(q_0, w_2) = q$. Note that if Prover follows $\mathcal{A}$, then Refuter can cause the interaction to be a word in $\mathrm{Violate}(L(\mathcal{A}_1), L(\mathcal{A}_2), k)$ by generating $w_1 \cdot \# \cdot w_2 \cdot \#$. Indeed, then the resulting prefix cannot agree with $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$. Accordingly, Refuter's strategy is to first force Prover to commit on the transitions of a k-DFA, and then to generate $w_1 \cdot \# \cdot w_2 \cdot \#$, for the appropriate words w_1 and w_2. Next, we show how Refuter can force Prover to commit on the transitions of a k-DFA.

A legal word $w = x \oplus y$ induces a partial function $\delta_w : [k] \times \Sigma \to [k]$, where for all $j \geq 1$, we have that $y_{j+1} = \delta_w(y_j, x_j)$. Forcing Prover to commit on the transitions of a k-DFA amounts to generating a word w for which δ_w is complete.

Lemma 3. *For every $k \geq 1$, there is a strategy for Refuter that forces Prover to commit on the transitions of a k-DFA in $O(k^2 \cdot |\Sigma|)$ rounds.*

Proof. Refuter maintains a set $S \subseteq [k]$ of discovered states, and a set $\Delta \subseteq [k] \times \Sigma \times [k]$ of discovered transitions. Note that for every discovered state $q \in S$, Refuter can construct a word $w \in \Sigma^*$ that Prover maps to q using transitions in Δ. Initially, the sets S and Δ are empty. Prover starts the interaction outputting an initial state q_0, and Refuter sets $S = \{q_0\}$.

Assume that there is an undiscovered transition from one of the discovered states. That is, there exist $q \in S$ and $\sigma \in \Sigma$ such that $\langle q, \sigma, r \rangle \notin \Delta$ for all $r \in [k]$.

Refuter outputs $w \cdot \sigma \cdot \#$, where w is a word Prover maps to q. Then, Prover answers with a state q', and Refuter adds q' to S, and $\langle q, \sigma, q' \rangle$ to Δ.

Refuter repeats the above process until Δ is complete. Each of the k states has $|\Sigma|$ outgoing transitions. Refuter exposes one new transition in at most $k+1$ rounds: A shortest word w that Prover maps to q has length at most $k-1$, then she outputs the letter σ, and then $\#$. Overall, the number of rounds is at most $k \cdot (k+1) \cdot |\Sigma|$, which is $O(k^2 \cdot |\Sigma|)$. □

Lemma 4. *Let $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, F \rangle$ and $\mathcal{A}' = \langle \Sigma, Q', q_0', \delta', F' \rangle$ be DFAs with N and N' states, respectively. For every state $q \in Q$, if there exists a word $w \in L(\mathcal{A}')$ such that $\delta^*(q_0, w) = q$, then there exists a word $w' \in L(\mathcal{A}')$ such that $\delta^*(q_0, w') = q$ and $|w'| \leq N \cdot N'$.*

Proof. Consider the product DFA $\mathcal{P} = \mathcal{A} \times \mathcal{A}'$. Let $w \in L(\mathcal{A}')$ be such that $\delta^*(q_0, w) = q$. Let $q' = (\delta')^*(q_0', w)$. Then, the state $\langle q, q' \rangle$ of $\mathcal{P}$ is reachable from $\langle q_0, q_0' \rangle$. A simple path from $\langle q_0, q_0' \rangle$ to $\langle q, q' \rangle$ induces the required word w'. □

Theorem 7. *Let $L_1, L_2 \subseteq \Sigma^*$ be regular languages, and let $N_1 = \text{index}(L_1)$ and $N_2 = \text{index}(L_2)$. For every $k \geq 1$, if $\langle L_1, L_2 \rangle$ is not k-DFA-separable, then Refuter can generate a word in $\text{Violate}(L_1, L_2, k)$ in $O(k^2 \cdot |\Sigma| + k \cdot N_1 + k \cdot N_2)$ rounds.*

Proof. As described in Lemma 3, Refuter can force Prover to commit on a k-DFA $\mathcal{A}$ in $O(k^2 \cdot |\Sigma|)$ rounds. Since $\langle L_1, L_2 \rangle$ is not k-DFA-separable, there are words $w_1 \in L_1, w_2 \in L_2$ such that the runs of $\mathcal{A}$ on w_1 and on w_2 both end in the same state. By Lemma 4, there exist such words satisfying $|w_1| \leq k \cdot N_1$ and $|w_2| \leq k \cdot N_2$. Refuter maintains a pair of such words for every k-DFA. After the DFA $\mathcal{A}$ is exposed, Refuter outputs the corresponding string $w_1 \cdot \# \cdot w_2 \cdot \#$, which has length at most $k \cdot N_1 + k \cdot N_2 + 2$. Overall, the interaction requires $O(k^2 \cdot |\Sigma| + k \cdot N_1 + k \cdot N_2)$ rounds. □

Recall that when $L_2 = \mathit{comp}(L_1)$, separation coincides with recognizability, with $N_1 = N_2 = N$. Hence, the $O(N^2)$ lower bound on the length of certificates in Theorem 2, applies also for $(N-1)$-DFA-separation. Our upper bound for $(N-1)$-DFA-separation in Theorem 7 includes an extra $|\Sigma|$ factor, as Refuter first forces Prover to commit on all transitions of the claimed DFA. We conjecture that Refuter can do better and force Prover to only to commit on a relevant part of the claimed DFA; namely one in which we can still point to a state $q \in F_{\mathcal{A},\mathcal{A}_1} \cap F_{\mathcal{A},\mathcal{A}_2}$ that is reachable via two words $w_1 \in L(\mathcal{A}_1)$ and $w_2 \in L(\mathcal{A}_2)$. Thus, rather than forcing Prover to commit on all $|\Sigma|$ successors of each state, Refuter forces Prover to commit only on transitions that reveal new states or reveal the required state q. Then, the prefix of the certificate that is generated in Lemma 3 is only of length $O(N^2)$, making the bound tight. Note that such a lazy exposure of the claimed DFA could be of help also in implementations of algorithms for the DFA identification problem [**?**].

6 Discussion and Directions for Future Research

On the Size of Provers and Refuters. Our study of certification focused on the *length* of certificates. We did not study the *size* of the transducers used by Prover and Refuter in order to generate these certificates. A naive upper bound on the size of such transducers follows from the fact that they are winning strategies in a game played on a deterministic looping automaton for Violate(L, k). Such an automaton has to store in its state space the set of transitions committed by Prover, and is thus exponential in k. The (L, k)-refuter we used for generating short certificates is also exponential in k, as it stores in its state space a mapping from the $k+1$ words in H to $[k]$ (see Theorem 2). On the other hand, it is easy to see that Prover can do with a transducer that is polynomial in k, as she can follow the transitions of $\mathcal{R}_L$.

Interestingly, with a slight change in the setting, we can shift the burden of maintaining the set of transitions committed by Prover from Refuter to Prover. We do this by requiring Prover to reveal new states in her claimed k-DFA in an *ordered* manner: Prover can respond with a state $i \in [k]$ only after she has responded with states $\{1, \ldots, i-1\}$. Formally, we say that $w = x \oplus y \in (\Sigma' \times [k])^* \cup (\Sigma' \times [k])^\omega$, with $x = x_1 \cdot x_2 \cdots$ and $y = y_1 \cdot y_2 \cdots$ is *ordered* iff for all $1 \leq j \leq |w|$ we have $y_j \leq \max\{y_l : 1 \leq l < j\} + 1$. Note that if Prover has a winning strategy in a game on DFA(L, k), she also has a winning strategy in a game in which DFA(L, k) is restricted to ordered words. In such a game, however, Refuter can make use of $\mathcal{R}_L$ and circumvent the maintenance of subsets of transitions, whereas Prover has to maintain a mapping from the states in $\mathcal{R}_L$ to their renaming imposed by the order condition. We leave the analysis of this setting as well as the study of trade-offs between the size of transducers and the length of the certificates to future research.

Infinite words. Our setting considers automata on finite words, and it focuses on the number of states required for recognizing a regular language. In [**?**], we used a similar methodology for refuting the recognizability of ω-regular languages by automata with limited expressive power. For example, deterministic *Büchi* automata (DBAs) are less expressive than their non-deterministic counterpart, and a DBA-refuter generates certificates that a given language cannot be recognized by a DBA. Thus, the setting in [**?**] is of automata on infinite words, and it focuses on expressive power.

Unlike DFAs, which allow polynomial minimization, minimization of DBAs is NP-complete [**?**]. Combining our setting here with the one in [**?**] would enable the certification and refutation of *k-DBA-recognizability*, namely recognizability by a DBA with k states. The NP-hardness of DBA minimization makes this combination very interesting. In particular, there are interesting connections between polynomial certificates and possible membership of DBA minimization in co-NP, as well as connections between size of certificates and succinctness of the different classes of automata.