# On the Fine Grained Complexity of Finite Automata Non-emptiness of Intersection

Mateus de Oliveira Oliveira[1] and Michael Wehar[2](✉)

[1] University of Bergen, Bergen, Norway
mateus.oliveira@uib.no
[2] Swarthmore College, Swarthmore, PA, USA
mwehar1@swarthmore.edu

**Abstract.** We study the fine grained complexity of the DFA non-emptiness of intersection problem parameterized by the number $k$ of input automata ($k$-DFA-NEI). More specifically, we are given a list $\langle \mathcal{A}_1, ..., \mathcal{A}_k \rangle$ of DFA's over a common alphabet $\Sigma$, and the goal is to determine whether $\bigcap_{i=1}^{k} \mathcal{L}(\mathcal{A}_i) \neq \emptyset$. This problem can be solved in time $O(n^k)$ by applying the classic Rabin-Scott product construction. In this work, we show that the existence of algorithms solving $k$-DFA-NEI in time slightly faster than $O(n^k)$ would imply the existence of deterministic sub-exponential time algorithms for the simulation of nondeterministic linear space bounded computations. This consequence strengthens the existing conditional lower bounds for $k$-DFA-NEI and implies new non-uniform circuit lower bounds.

**Keywords:** Finite automata · Intersection non-emptiness · Fine grained complexity · Parameterized complexity

## 1 Introduction

### 1.1 History

In the DFA non-emptiness of intersection problem (DFA-NEI), the input consists of a list $\langle \mathcal{A}_1, ..., \mathcal{A}_k \rangle$ of DFA's over a common alphabet $\Sigma$, and the goal is to determine whether the intersection of the languages $\mathcal{L}(\mathcal{A}_1), ..., \mathcal{L}(\mathcal{A}_k)$ is non-empty. When no restriction is imposed on the input, DFA-NEI is a PSPACE-complete problem [18]. Nevertheless, the classic Rabin-Scott product construction for finite automata yields a simple algorithm that solves DFA-NEI in time $O(n^k)$ where $n$ is the number of states and $k$ is the number of input automata. Therefore, for a fixed number of input automata, the problem can be solved in polynomial time.

In this work, we study the fine grained complexity of DFA-NEI parameterized by the number of input automata $k$. For clarity, we refer to this parameterized version as $k$-DFA-NEI. Interestingly, Rabin and Scott's six-decades-old

Mateus de Oliveira Oliveira acknowledges support from the Bergen Research Foundation and from the Research Council of Norway (project number 288761).

time $O(n^k)$ algorithm for $k$-DFA-NEI remains unimproved, and in particular, time $O(n^2)$ is still the best we can get for deciding non-emptiness of intersection for two DFA's.

Kasai and Iwata [17] are believed to be the first to provide conditional lower bounds for $k$-DFA-NEI. They showed that $k$-DFA-NEI requires deterministic time $\Omega(n^{(k-2)/2})$ under the conjecture that NSPACE$[k \cdot \log n] \not\subseteq$ DTIME$[n^{k-\varepsilon}]$ for all $\varepsilon > 0$.

Almost two decades later, Karakostas, Lipton, and Viglas showed that faster algorithms for certain variants of $k$-DFA-NEI would imply both faster algorithms for certain NP-hard problems and new complexity class separations [16]. In particular, they showed that an algorithm solving $k$-DFA-NEI in time $n^{o(k)}$ would have two consequences. First, this would imply that the well studied SUBSET SUM problem can be solved in time $O(2^{\varepsilon \cdot n})$ for every $\varepsilon > 0$. Second, this would imply that NTIME$[n] \subseteq$ DTIME$[2^{o(n)}]$. They also showed some remarkable consequences of the existence of algorithms solving $k$-DFA-NEI in time $s \cdot r^{o(k)}$ where $s$ is the number of states in the largest input automaton and $r$ is the number of states in the second largest input automaton. In particular, such an algorithm would imply that NSPACE$[O(\log s)] \subset$ DTIME$[s^{1+\epsilon}]$ for all $\epsilon > 0$, which would further imply that $P \neq NL$. Additionally, by padding, we would also have NSPACE$[s] \subseteq$ DTIME$[2^{o(s)}]$. It is worth noting that this last result strongly requires that the runtime has only a marginal dependence on the size $s$ of the largest automaton. Further, this last result is in a similar spirit as conditional lower bounds for weighted satisfiability problems from [8–10].

It was shown by Fernau and Krebs [12], and independently in [30], that an algorithm solving $k$-DFA-NEI in time $n^{o(k)}$ would contradict the celebrated exponential time hypothesis (ETH). Using a refinement of the proof technique introduced in [16], it was shown in [29,30] that if $k$-DFA-NEI can be solved in time $n^{o(k)}$, then $P \neq NL$. Additional results on the parameterized complexity of non-emptiness of intersection for DFA's are presented in [17,19,26] and results on the fine grained complexity of non-emptiness of intersection specifically for two and three DFA's are presented in [23].

### 1.2 Our Results

**Finer Simulations for Nondeterministic Linear Space.** Our first result (Theorem 1) provides a finer reduction from the problem of simulating a nondeterministic space bounded Turing machine to $k$-DFA-NEI. The following two corollaries of Theorem 1 fill in some gaps in the literature related to non-emptiness of intersection. In this work, NSPACE$[n]$ denotes the class of functions computable by 2-tape Turing Machines over a binary alphabet using at most $n$ bits on its work tape.

(Corollary 1.1) If we can solve $k$-DFA-NEI in time $n^{o(k)}$, then NSPACE$[n] \subseteq$ DTIME$[2^{o(n)}]$ [28].[1]

[1] This work was not formally published.

(Corollary 1.2) If there exists $k \geq 2$ and $\varepsilon > 0$ such that $k$-DFA-NEI can be solved in time $O(n^{k-\varepsilon})$, then NSPACE$[n + o(n)] \subseteq$ DTIME$[2^{(1-\delta)n}]$ for some $\delta > 0$ [30].

As mentioned in the first part of the introduction, the conclusion that NSPACE$[n] \subseteq$ DTIME$[2^{o(n)}]$ can be obtained from the results in [16] under the assumption that there exists an algorithm for $k$-DFA-NEI running in time $s \cdot r^{o(k)}$ where $s$ is the size of the largest automaton and $r$ is the size of the second largest automaton. Corollary 1.1 relaxes this assumption to the existence of an algorithm running in time $n^{o(k)}$, with no regard to the way in which the sizes of the input automata compare with each other. We observe that the same assumption as ours was shown in [16] to imply that NTIME$[n] \subseteq$ DTIME$[2^{o(n)}]$. Therefore, we improve the consequence in [16] from NTIME$[n] \subseteq$ DTIME$[2^{o(n)}]$ to NSPACE$[n] \subseteq$ DTIME$[2^{o(n)}]$.

Corollary 1.2 states that for each $k > 1$, any additive constant improvement on the running time of the Rabin-Scott algorithm for $k$-DFA-NEI would imply the existence of faster than state-of-the art algorithms for the simulation of nondeterministic linear space bounded computations. In particular, an algorithm solving non-emptiness of intersection for two DFA's in time $O(n^{2-\varepsilon})$, for some $\varepsilon > 0$, would imply that NSPACE$[n + o(n)] \subseteq$ DTIME$[2^{(1-\delta)n}]$ for some $\delta > 0$.

**Contradicting Stronger Versions of ETH and SETH.** In the satisfiability problem for Boolean formulas (SAT), we are given a Boolean formula. The goal is to determine if there exists an assignment that satisfies the formula. It is common to restrict the inputs for SAT to formulas in conjunctive normal form (CNF-SAT). Further, it is common to restrict the inputs for SAT to formulas in conjunctive normal form with clause width at most $k$ ($k$-CNF-SAT) for some fixed number $k$.

The *Exponential Time Hypothesis* (ETH) asserts that for some $\varepsilon > 0$, 3-CNF-SAT cannot be solved in time $(1 + \varepsilon)^n$ [14]. The *strong exponential time hypothesis* (SETH) asserts that for every $\varepsilon > 0$, there is a large enough integer $k$ such that $k$-CNF-SAT cannot be solved in time $(2 - \varepsilon)^n$ [7,14,15]. ETH has been used to rule out the existence of subexponential algorithms for many decision problems [14], parameterized problems [8,20], approximation problems [22], and counting problems [11]. On the other hand, SETH has been useful in establishing tight lower bounds for many problems in $P$ such as Edit Distance [3], $k$-Dominating Set [24], $k$-DFA-NEI [30], and many other problems [2,27,33].

Our next results state that slightly faster algorithms for $k$-DFA-NEI would contradict much stronger versions of ETH and SETH. First, we show that if there exists $k \geq 2$ such that $k$-DFA-NEI can be solved in time $O(n^{k-\varepsilon})$ for some $\varepsilon > 0$, then satisfiability for $n$-variable Boolean formulas of size $2^{o(n)}$ can be solved in time $O(2^{(1-\delta)n})$ for some $\delta > 0$ (Corollary 2). The inexistence of such fast algorithms for satisfiability for $n$-variable Boolean formulas of subexponential size is a safer assumption than SETH. Going further, we show that if $k$-DFA-NEI can be solved in time $n^{o(k)}$, then satisfiability for $n$-input fan-in-2 Boolean circuits of depth $O(n)$ and size $2^{o(n)}$ can be solved in time $2^{o(n)}$

(Corollary 3). We note that this consequence is stronger than the existence of algorithms solving CNF-SAT in sub-exponential time. Indeed, CNF formulas of polynomial size are a very weak model of computation, which are unable, for instance, to compute the parity of their input bits [13]. On the other hand, circuits of linear depth can already simulate complicated cryptographic primitives. Therefore, the inexistence of satisfiability algorithms for such circuits is a safer assumption than ETH.

**Non Uniform Circuit Lower Bounds.** Finally, from the results mentioned above together with results obtained within the context of Williams' *algorithms versus lower bounds* framework [1,31,32] (as well as [4]), we infer that faster algorithms for $k$-DFA-NEI would imply non-uniform circuit lower bounds that are sharper than what is currently known. In particular, an algorithm running in time $n^{o(k)}$ for $k$-DFA-NEI would imply that there are problems in $E^{NP}$ that cannot be solved by non-uniform fan-in-2 Boolean circuits of linear depth and sub-exponential size (Corollary 4). We note that currently it is still open whether every problem in $E^{NP}$ can be solved by non-uniform fan-in-2 Boolean circuits of linear size. Additionally, we show that an algorithm running in time $O(n^{2-\varepsilon})$ for 2-DFA-NEI would imply that there are problems in $E^{NP}$ that cannot be solved by non-uniform Boolean formulas of sub-exponential size (Corollary 5).

Further, we have that even polylogarithmic improvements for the running time of algorithms solving 2-DFA-NEI would imply interesting lower bounds. More specifically, if 2-DFA-NEI can be solved in time $O(n^2/\log^c n)$ for every $c > 0$, then there are functions that can be computed in NTIME$[2^{O(n)}]$ but not by non-uniform $\mathrm{NC}^1$ circuits.

Analogous conditional non-uniform circuit lower bounds have been obtained in [1] under the assumptions that the Edit Distance problem can be computed in time $O(n^{2-\varepsilon})$ for some $\varepsilon > 0$ and in time $O(n^2/\log^c n)$ for every $c \geq 1$. It is worth noting that Theorem 5 which establishes conditional lower bounds for fan-in-2 Boolean circuits of linear depth and sub-exponential size is not explicitly stated in [1] and no parallel to the associated conditional lower bound for $k$-DFA-NEI is given for Edit Distance.

## 2 Reducing Acceptance in NSPACE[$n$] to DFA-NEI

In this section we provide a reduction from the problem of simulating 2-tape Turing machines to DFA-NEI. For any $k$, the reduction in Theorem 1 outputs $k$ DFA's each with at most $O(m^2 \cdot n \cdot \sigma^{1+c} \cdot 2^{\frac{\sigma}{k}})$ states where $m$ denotes the number of states in the Turing machine, $n$ denotes the input string length, $\sigma$ denotes the amount of space on the binary work tape, and $c$ denotes the maximum number of occurrences of a special delimiter symbol # that can simultaneously appear on the work tape during the computation. The parameter $c$ is a constant associated with the Turing machine and is independent of the parameters $n$ and $\sigma$.

**2-tape Turing Machines:** A 2-tape Turing machine with binary alphabet is a machine with a two-way read-only input tape and a two-way binary work tape.

More formally, it is a tuple $M = (Q, \{0,1\}, c, q_0, F, \delta)$ where $Q$ is a set of states, $c$ is the maximum number of occurrences of special delimiter symbol $\#$, $q_0 \in Q$ is an initial state, $F$ is a set of final states, and

$$\delta : Q \times (\{0,1\} \cup \{\#\})^2 \to \mathcal{P}(Q \times \{-1,0,1\}^2 \times (\{0,1\} \cup \{\#\}))$$

is a partial transition function that assigns to each triple

$$(q, b_1, b_2) \in Q \times (\{0,1\} \cup \{\#\})^2,$$

a set of tuples

$$\delta(q, b_1, b_2) \subseteq Q \times \{-1,0,1\}^2 \times (\{0,1\} \cup \{\#\}).$$

We say that a tuple

$$(q, d, d', w) \in Q \times \{-1,0,1\}^2 \times (\{0,1\} \cup \{\#\})$$

is an instruction that sets the machine to state $q$, moves the input head from position $p$ to position $p + d$, moves the work head from position $p'$ to position $p' + d'$, and writes symbol $w$ at position $p'$ on the work tape. The transition function $\delta$ specifies that if the machine $M$ is currently at state $q$, reading symbol $b_1$ on the input tape and symbol $b_2$ on the work tape, then the next instruction of the machine must be an element of the set $\delta(q, b_1, b_2)$.

**Configurations:** A *space-$\sigma$ configuration* for $M$ on input $x \in \{0,1\}^*$ is a tuple

$$(q, h, h', y) \in Q \times [|x|] \times [\sigma] \times (\{0,1\} \cup \{\#\})^\sigma$$

where intuitively, $q \in Q$ is the current state of $M$, $h \in [|x|]$ is the position of $M$'s input tape head, $h' \in [\sigma]$ is the position of $M$'s work tape head, and $y \in (\{0,1\} \cup \{\#\})^\sigma$ is the binary string (containing at most $c$ special delimiter symbols) corresponding to the first $\sigma$ symbols on the work tape of $M$.

**Configuration Sequences:** A *space-$\sigma$ configuration sequence* for $M$ on input $x \in \{0,1\}^*$ is a sequence of the form

$$\begin{aligned} S \equiv (q_0, h_0, h'_0, y_0) &\xrightarrow{(q_1,d_1,d'_1,r_1,r'_1,w_1)} (q_1, h_1, h'_1, y_1) \\ &\xrightarrow{(q_2,d_2,d'_2,r_2,r'_2,w_2)} (q_2, h_2, h'_2, y_2) \\ &\dots \\ &\xrightarrow{(q_k,d_k,d'_k,r_k,r'_k,w_k)} (q_k, h_k, h'_k, y_k) \end{aligned}$$

satisfying the following conditions.

1. For each $i \in \{0, 1, ..., k\}$, $(q_i, h_i, h'_i, y_i)$ is a space-$\sigma$ configuration for $M$ on $x$.
2. $q_0$ is the initial state of $M$, $y_0 = 0^\sigma$, meaning that the work tape is initialized with zeros, and $h_0 = h'_0 = 1$, meaning that the input tape head and work tape head are in the first position of their respective tapes.

3. For each $i \in \{1, ..., k\}$, $(q_i, d_i, d'_i, w_i) \in \delta(q_{i-1}, x[h_{i-1}], y_{i-1}[h'_{i-1}])$, meaning the state of the machine at time $i$, the directions taken by both heads at time $i$, and the symbol written on the work tape at time $i$ are compatible with the transition function $\delta$, and depend only on the state at time $i-1$ and on the symbols that are read at time $i-1$.
4. For each $i \in \{1, ..., k\}$, $h_i = h_{i-1} + d_i$, $h'_i = h'_{i-1} + d'_i$, $r_i = x[h_{i-1}]$, $r'_i = y_{i-1}[h'_{i-1}]$, and $y_i$ is obtained from $y_{i-1}$ by substituting $w_i$ for the symbol $y_{i-1}[h'_{i-1}]$, and by leaving all other symbols untouched. Intuitively, this means that the configuration at time $i$ is obtained from the configuration at time $i-1$ by the application of the transition $(q_i, d_i, d'_i, w_i)$.
5. For each $i \in \{0, 1, ..., k\}$, $y_i$ contains at most $c$ occurrences of the special delimiter symbol #.

We say that the sequence

$$I \equiv (q_1, d_1, d'_1, r_1, r'_1, w_1)(q_2, d_2, d'_2, r_2, r'_2, w_2)...(q_k, d_k, d'_k, r_k, r'_k, w_k)$$

that induces a configuration sequence $S$ as above is a *space-$\sigma$ instruction sequence* for $M$ on input $x$. We say that $I$ is accepting if $q_k \in F$.

*Remark 1.* As suggested in [17], the technique provided in the proof of Proposition 3 from [25] can be applied to remove the special delimiter symbol from the work tape by increasing the Turing machine's state and space complexities. However, without a formal proof in the literature of the stated result from [17] and because it is not required in our work, we decided to refrain from using it.

**Theorem 1.** *Let a nondeterministic $m$-state 2-tape Turing machine $M$ with binary tape alphabet (other than at most $c$ occurrences of symbol #) and an input string $x$ of length $n$ be given. If $M$ uses at most $\sigma$ symbols on the work tape, then for every $k$, we can efficiently compute $k$ DFA's $\langle \mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_k \rangle$ each with a binary alphabet and $O(m^2 \cdot n \cdot \sigma^{1+c} \cdot 2^{\frac{\sigma}{k}})$ states such that $M$ accepts $x$ if and only if $\bigcap_{i=1}^{k} \mathcal{L}(\mathcal{A}_i) \neq \emptyset$.*

*Proof.* The Turing machine $M$ accepts $x$ if and only if there exists an accepting space-$\sigma$ instruction sequence for $M$ on $x$. We build $k$ DFA's that read in a binary string and collectively determine whether the string encodes an accepting space-$\sigma$ instruction sequence for $M$ on $x$.

Consider splitting the work tape of $M$ into $k$ equal sized blocks each consisting of $\frac{\sigma}{k}$ work tape cells. A block-$i$ space-$\sigma$ configuration for $M$ on input $x$ consists of the state, input tape head, work tape head, the contents of the work tape from position $lbound_i := (i-1) \cdot \frac{\sigma}{k} + 1$ to position $rbound_i := i \cdot \frac{\sigma}{k}$, and all $c$ positions of the occurrences of special delimiter symbol #. We construct $k$ DFA's $\langle \mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_k \rangle$ where each DFA $\mathcal{A}_i$ keeps track of the current block-$i$ space-$\sigma$ configuration for $M$ on input $x$. The DFA's read in space-$\sigma$ instructions one at a time and transition accordingly where each instruction is encoded as a unique bit string of length $O(\log(m))$.

The start state of DFA $\mathcal{A}_i$ represents the block-$i$ space-$\sigma$ configuration $(q_0, 1, 1, 0^{\frac{\sigma}{k}})$ where $q_0$ is the start state of $M$. Further, a state representing a block-$i$ space-$\sigma$ configuration $(q_j, h_j, h'_j, contents_j)$ is accepting if $q_j$ is a final state of $M$. Suppose that the DFA $\mathcal{A}_i$ is currently at a state representing a block-$i$ space-$\sigma$ configuration $(q_j, h_j, h'_j, contents_j)$ and reads in a space-$\sigma$ instruction $(q, d, d', r, r', w)$. The DFA $\mathcal{A}_i$ transitions to a state representing a block-$i$ space-$\sigma$ configuration $(q_{j+1}, h_{j+1}, h'_{j+1}, contents_{j+1})$ if:

1. $(q, d, d', w) \in \delta(q_j, r, r')$ and $q = q_{j+1}$
2. $h_{j+1} = h_j + d$ and $h'_{j+1} = h'_j + d'$
3. $1 \leq h_j, h_{j+1} \leq n$ and $1 \leq h'_j, h'_{j+1} \leq \sigma$
4. $r = x[h_j]$
5. if $lbound_i \leq h'_j \leq rbound_i$, then $r' = contents_j[h'_j - lbound_i + 1]$ and $w = contents_{j+1}[h'_j - lbound_i + 1]$

Collectively the DFA's determine whether the input string encodes an accepting space-$\sigma$ instruction sequence for $M$ on $x$. Therefore, the Turing machine $M$ accepts $x$ if and only if there exists an accepting space-$\sigma$ instruction sequence for $M$ on $x$ if and only if $\bigcap_{i=1}^{k} \mathcal{L}(\mathcal{A}_i) \neq \emptyset$. Further, the DFA's each have at most $O(m^2 \cdot n \cdot \sigma^{1+c} \cdot 2^{\frac{\sigma}{k}})$ states because there are $O(m)$ space-$\sigma$ instructions and $O(m \cdot n \cdot \sigma^{1+c} \cdot 2^{\frac{\sigma}{k}})$ block-$i$ space-$\sigma$ configurations. □

*Remark 2.* The preceding simulation is sufficient for our purposes. However, we suggest that it could be refined by having only one DFA keep track of the Turing machine's tape heads. When $\sigma = O(\log(n))$, such a refinement could be used to obtain a tighter connection between $k$-DFA-NEI and nondeterministic logspace.

**Corollary 1.** *We obtain the following directly from the preceding theorem:*

1. *If we can solve* $k$-DFA-NEI *in time* $n^{o(k)}$*, then* NSPACE$[n] \subseteq$ DTIME$[2^{o(n)}]$.
2. *If there exists* $k \geq 2$ *and* $\varepsilon > 0$ *such that* $k$-DFA-NEI *can be solved in time* $O(n^{k-\varepsilon})$*, then* NSPACE$[n + o(n)] \subseteq$ DTIME$[2^{(1-\delta)n}]$ *for some* $\delta > 0$.

## 3 Non-emptiness of Intersection and Conditional Lower Bounds

In this section we apply results obtained in Theorem 1 to show that even a slight improvement in running time of the classic algorithm for non-emptiness of intersection for finite automata would yield faster than state of the art algorithms for satisfiability for Boolean formulas and Boolean circuits. Therefore, this result implies that the impossibility of obtaining better algorithms for non-emptiness of intersection for $k$ finite automata can be based on assumptions that are safer than the exponential time hypothesis (ETH). An analogous result is proven with respect to non-emptiness of intersection for a constant number of finite automata (say two). We will show that the existence of algorithms that are faster than time $O(n^{2-\varepsilon})$ for non-emptiness of intersection for 2 DFA's would contradict assumptions that are safer than the strong exponential time hypothesis (SETH). We note that the endeavour of basing lower bounds for algorithms in P on assumptions that are safer than ETH or SETH has been pursued before [1]. In this work, we obtain lower bounds using assumptions like those used in [1], with the advantage that our reductions are simpler. Therefore, we believe that the techniques employed here provide a cleaner framework that can potentially be used to strengthen the analysis of the fine grained complexity of other algorithmic problems in P.

Finally, by applying Williams' *algorithms versus lower bounds* framework, we are able to show that faster algorithms for non-emptiness of intersection for finite automata would also imply non-uniform circuit lower bounds that are much better than those that are currently known.

## 3.1 Satisfiability for Boolean Formulas

**Lemma 1.** *Satisfiability for $n$-variable Boolean formulas of size $s$ is solvable by a nondeterministic 2-tape Turing machine with binary alphabet using at most $n + O(\log(s))$ bits and a fixed number of delimiter symbol # occurrences on the work tape.*[2]

*Proof.* The machine uses $n$ tape cells to guess an assignment $x \in \{0,1\}^n$ to the input variables. Subsequently, using $O(\log s)$ work tape cells, the machine evaluates the Boolean formula from the input tape on the guessed assignment from the work tape. This evaluation problem is referred to as the Boolean formula value problem (BFVP) and has been shown to be solvable in space $O(\log s)$ on formulas of size $s$ in [6,21]. Storing both the assignment and the formula evaluation on the same work tape, it will be necessary to use a fixed number of occurrences of the delimiter symbol # as left/right tape markers, a delimiter between assignment and formula evaluation, and markers for remembering the position in the formula evaluation and the assignment when the work tape head moves from formula evaluation to assignment or vice versa. □

By combining Theorem 1 with Lemma 1, we obtain the following.

**Theorem 2.** *If there exists $k \geq 2$ and $\varepsilon > 0$ such that $k$-DFA-NEI can be solved in time $O(n^{k-\varepsilon})$, then satisfiability for $n$-variable Boolean formulas of size $s$ is solvable in time $\text{poly}(s) \cdot 2^{n(1-\delta)}$ for some $\delta > 0$.*

*Proof.* Suppose that there exists $k \geq 2$ and $\varepsilon > 0$ such that $k$-DFA-NEI can be solved in time $O(n^{k-\varepsilon})$. Let a $n$-variable Boolean formula $\phi$ of size $s$ be given. By Theorem 1 and Lemma 1, we can reduce satisfiability for $\phi$ to non-emptiness of intersection for $k$ DFA's each with $\text{poly}(s) \cdot 2^{\frac{n}{k}}$ states. Therefore, by assumption, we can determine whether $\phi$ has a satisfying assignment in time $s^{O(k)} \cdot 2^{\frac{k-\varepsilon}{k} \cdot n}$. It follows that satisfiability for $n$-variable Boolean formulas of size $s$ is solvable in time $\text{poly}(s) \cdot 2^{n(1-\delta)}$ for some $\delta > 0$. □

[2] In addition, both QBF and satisfiability for nondeterministic branching programs are solvable by nondeterministic 2-tape Turing machines with binary alphabet using at most $n + O(\log(s))$ bits and a fixed number of delimiter symbol # occurrences.

It was shown in [30] that if there exists some $k \geq 2$ and $\varepsilon > 0$ such that $k$-DFA-NEI can be solved in time $O(n^{k-\varepsilon})$, then SETH is false. The following corollary improves this result by showing a much stronger consequence. The corollary follows directly from Theorem 2.

**Corollary 2.** *If there exists $k \geq 2$ and $\varepsilon > 0$ such that $k$-DFA-NEI can be solved in time $O(n^{k-\varepsilon})$, then satisfiability for $n$-variable Boolean formulas of size $2^{o(n)}$ can be solved in time $O(2^{n(1-\delta)})$ for some $\delta > 0$.*

Note that while CNF's of bounded width and polynomial size are a very weak computational model, Boolean formulas of sub-exponential size can already simulate any circuit in the class NC. Therefore, the consequence of Corollary 2 would contradict the NC-SETH hypothesis, a more robust version of SETH which states that satisfiability for circuits of polynomial size and polylogarithmic depth cannot be solved in time $O(2^{n(1-\delta)})$ for any $\delta > 0$ [1]. In the next subsection we show that the existence of an algorithm running in time $n^{o(k)}$ for $k$-DFA-NEI would imply faster satisfiability algorithms for even larger classes of circuits.

## 3.2 Satisfiability for Boolean Circuits

In the CIRCUIT VALUE problem (CV), we are given a $n$-input fan-in-2 Boolean circuit $C$ and a string $x \in \{0,1\}^n$. The goal is to determine whether the circuit $C(x)$, obtained by initializing the input variables of $C$ according to $x$, evaluates to 1. Let the size of $C$ denote the number of gates of $C$. The next lemma, which is a classic result in complexity theory [5], states that the circuit value problem for circuits of depth $d$ and size $s$ can be solved in space $O(d) + O(\log s)$ on a 2-tape Turing machine.

**Lemma 2 (Borodin [5]).** *There is a deterministic 2-tape Turing machine $M$ with binary alphabet, that takes as input a pair $\langle C, x\rangle$ where $x$ is a string in $\{0,1\}^n$ and $C$ is a $n$-input fan-in-2 Boolean circuit of depth $d$ and size $s$, and determines, using at most $O(d)+O(\log s)$ work tape cells, whether $C(x)$ evaluates to 1.*

In the satisfiability problem for Boolean circuits, we are given a $n$-input fan-in-2 Boolean circuit $C$. The goal is to determine whether there exists a string $x \in \{0,1\}^n$ such that $C(x)$ evaluates to 1. As a consequence of Lemma 2, we have that satisfiability for Boolean circuits can be decided by a nondeterministic 2-tape Turing machine using at most $n + O(d) + O(\log s)$ tape cells.

**Lemma 3.** *There is a nondeterministic 2-tape Turing machine $M$ with binary alphabet and a fixed number of delimiter symbol # occurrences, that takes as input a $n$-input fan-in-2 Boolean circuit $C$ of depth $d$ and size $s$, and determines, using at most $n + O(d) + O(\log s)$ tape cells, whether $C$ is satisfiable.*

By combining Theorem 1 with Lemma 3, we obtain the following.

**Theorem 3.** *If* $k$-DFA-NEI *can be solved in time* $O(n^{f(k)})$*, then satisfiability for* $n$*-input fan-in-*2 *Boolean circuits of depth* $d$ *and size* $s$ *can be solved in time* $s^{O(f(k))} \cdot 2^{\frac{f(k)}{k} \cdot (n+O(d))}$ *where* $k$ *is allowed to depend on* $n$*,* $d$*, and* $s$*.*

*Proof.* Suppose that $k$-DFA-NEI can be solved in time $O(n^{f(k)})$. Let a $n$-input fan-in-2 Boolean circuit $C$ of depth $d$ and size $s$ be given. By Theorem 1 and Lemma 3, for any $k$, we can reduce satisfiability for $C$ to non-emptiness of intersection for $k$ DFA's each with poly$(s) \cdot 2^{\frac{n+O(d)}{k}}$ states. Therefore, by assumption, we can determine whether $C$ has a satisfying assignment in time $s^{O(f(k))} \cdot 2^{\frac{f(k)}{k} \cdot (n+O(d))}$. It follows that satisfiability for $n$-input fan-in-2 Boolean circuits of depth $d$ and size $s$ is solvable in the desired time. □

It was shown in [12,30] that if $k$-DFA-NEI can be solved in time $n^{o(k)}$, then ETH is false. The following corollary improves this result by showing a much stronger consequence. The corollary follows directly from Theorem 3.

**Corollary 3.** *If* $k$-DFA-NEI *can be solved in time* $n^{o(k)}$*, then satisfiability for fan-in-*2 *Boolean circuits of depth* $O(n)$ *and size* $2^{o(n)}$ *can be solved in time* $2^{o(n)}$*.*

Note that the consequence in the preceding corollary is stronger than ETH because circuits of linear depth and sub-exponential size are a stronger computational model than formulas in conjunctive normal form.

## 3.3 Circuit Lower Bounds

Corollaries 2 and 3 lead us to the following questions.

- What are the barriers (beyond ETH) to solving satisfiability for fan-in-2 Boolean circuits of depth $O(n)$ and size $2^{o(n)}$ more efficiently?
- What are the barriers (beyond SETH) to solving satisfiability for Boolean formulas of size $2^{o(n)}$ more efficiently?
- What are the barriers to solving satisfiability only slightly faster for Boolean formulas of polynomial size?

Below, we investigate the preceding questions and reference recent works [1,4,31,32] that connect satisfiability problems to non-uniform circuit complexity lower bounds. From these connections, we observe that faster algorithms for $k$-DFA-NEI would imply new non-uniform circuit complexity lower bounds.

**Barriers Beyond ETH.** The following is a slightly modified restatement of a technical theorem from [1] (with related results in [4,31,32]) that connects circuit satisfiability to non-uniform circuit complexity lower bounds for $E^{NP}$. Recall that $E^{NP}$ is the class of functions that can be computed by Turing machines that operate in time $2^{O(n)}$ with the help of an NP oracle. Note that the strings that are passed to each call of the oracle may have size up to $2^{O(n)}$.

**Theorem 4 (Theorem 8 in [1]).** *Let $S(n)$ be a time constructible and monotone non-decreasing function such that $n \leq S(n) \leq 2^{o(n)}$. Let $\mathcal{C}$ be a class of circuits. Suppose there is a SAT algorithm for $n$-input circuits which are arbitrary functions of three $O(S(n))$-size circuits from $\mathcal{C}$, that runs in time $O(2^n/(n^{10} \cdot S(n)))$. Then $E^{NP}$ does not have $S(n)$-size circuits from $\mathcal{C}$.*

Notice that if we take three circuits of linear depth and sub-exponential size and combine their outputs using a constant number of additional gates, then the resulting circuit still has linear depth and sub-exponential size. Therefore, a satisfiability algorithm running in time $2^{o(n)}$ for $n$-input fan-in-2 Boolean circuits of linear depth and sub-exponential size would imply that there are functions in $E^{NP}$ that cannot be computed by such circuits. This would be a significant consequence in complexity theory since to date it is not even known whether all functions in $E^{NP}$ can be computed by non-uniform circuits of linear size. In view of our discussion, Theorem 5 directly follows from Theorem 4, but is not explicitly stated in [1].

**Theorem 5.** *If satisfiability for $n$-input fan-in-2 Boolean circuits of depth $O(n)$ and size $2^{o(n)}$ is solvable in time $O(2^{(1-\delta)n})$ for some $\delta > 0$, then $E^{NP}$ does not have non-uniform fan-in-2 Boolean circuits of $O(n)$ depth and $2^{o(n)}$ size.*

By combining the preceding theorem with Corollary 3, we obtain the following.

**Corollary 4.** *If $k$-*DFA-NEI *can be solved in time $n^{o(k)}$, then $E^{NP}$ does not have non-uniform fan-in-2 Boolean circuits of $O(n)$ depth and $2^{o(n)}$ size.*

**Barriers Beyond SETH.** Next, we look at another known result that connects formula satisfiability to non-uniform formula complexity lower bounds for $E^{NP}$. The following theorem is similar to Corollary 1.1 in [1] and directly follows from Theorem 4.

**Theorem 6.** *If satisfiability for $n$-variable Boolean formulas of size $2^{o(n)}$ is solvable in time $O(2^{(1-\delta)n})$ for some $\delta > 0$, then $E^{NP}$ does not have non-uniform Boolean formulas of size $2^{o(n)}$.*

By combining the preceding theorem with Corollary 2, we obtain the following.

**Corollary 5.** *If there exists $k \geq 2$ and $\varepsilon > 0$ such that $k$-*DFA-NEI *can be solved in time $O(n^{k-\varepsilon})$, then $E^{NP}$ does not have non-uniform Boolean formulas of size $2^{o(n)}$.*

**Barriers to Slightly Faster Algorithms.** Finally, we investigate the possible consequences of polylogarithmic improvements to the running time of algorithms for $k$-DFA-NEI, and in particular for 2-DFA-NEI. The following is a restatement of Theorem 7 in [1] (related to Theorem 1.3 in [31]).

**Theorem 7 (Theorem 7 in [1]).** *Suppose that there is a satisfiability algorithm for bounded fan-in formulas of size $n^r$ running in time $O(2^n/n^r)$, for each constant $r > 0$. Then* NTIME$[2^{O(n)}]$ *is not contained in non-uniform* NC$^1$.

By combining the preceding theorem with Theorem 1 and Lemma 1, we obtain the following.

**Corollary 6.** *If $k$-*DFA-NEI *can be solved in time $O(n^k/(\log n)^c)$ for every $c > 0$, then* NTIME$[2^{O(n)}]$ *is not contained in non-uniform* NC$^1$.

*Proof.* Suppose that there exists $k \geq 2$ such that $k$-DFA-NEI can be solved in time $O(n^k/(\log n)^c)$ for every $c > 0$. By combining Theorem 1 and Lemma 1, it follows that for all $r > 0$, satisfiability for $n$-variable Boolean formulas of size $n^r$ can be reduced to intersecting $k$ DFA's with at most $poly(n) \cdot 2^{\frac{n}{k}}$ states where $k$ and $r$ are treated as constants. Therefore, satisfiability for $n$-variable Boolean formulas of size $n^r$ can be solved in time

$$\frac{(poly(n) \cdot 2^{\frac{n}{k}})^k}{\log^c(poly(n) \cdot 2^{\frac{n}{k}})} \leq \frac{poly(n) \cdot 2^n}{(O(\log n) + \frac{n}{k})^c} \leq O(\frac{n^d \cdot 2^n}{n^c})$$

for all $c > 0$ and some constant $d$ that is independent of $c$. If we set $c = d \cdot r$, then we have that satisfiability for Boolean formulas of size $n^r$ can be solved in time $O(2^n/n^r)$. It follows that satisfiability for Boolean formulas of size $n^r$ can be solved in time $O(2^n/n^r)$ for all $r > 0$. Moreover, by Theorem 7, it follows that NTIME$[2^{O(n)}]$ does not have non-uniform NC$^1$ circuits. $\square$

Notice that if we set $k = 2$ in the preceding corollary, then it follows that if 2-DFA-NEI can be solved in time $O(n^2/(\log n)^c)$ for every $c > 0$, then NTIME$[2^{O(n)}]$ is not contained in non-uniform NC$^1$.

## 4 Conclusion

We analyzed the fine grained complexity of the non-emptiness of intersection problem parameterized by the number of input DFA's ($k$-DFA-NEI). Despite the fact that this problem has been studied for at least six decades, the fastest known algorithm for $k$-DFA-NEI is still the $O(n^k)$ time algorithm obtained by a direct application of the classic Rabin-Scott product construction.

The lack of progress in the task of developing a faster algorithm for $k$-DFA-NEI motivated the search for evidence supporting the possibility that substantially faster algorithms for this problem do not exist. In this work, we have simplified and strengthened several previous conditional lower bounds for $k$-DFA-NEI under a unified perspective. In particular, we have shown that if $k$-DFA-NEI can be solved in time $n^{o(k)}$ then NSPACE$[n] \subseteq$ DTIME$[2^{o(n)}]$. Additionally, we have shown that solving non-emptiness of intersection for two DFA's in time $O(n^{2-\varepsilon})$ for some $\varepsilon > 0$ would imply that NSPACE$[n + o(n)] \subseteq$ DTIME$[2^{(1-\delta)n}]$ for some $\delta > 0$. Further, we have unveiled several connections between $k$-DFA-NEI and non-uniform circuit complexity theory. In particular, we have shown that even improving non-emptiness of intersection for two DFA's by a $\log^c n$ factor for every $c > 0$ would imply non-uniform NC$^1$ circuit lower bounds.

## References

1. Abboud, A., Hansen, T.D., Williams, V.V., Williams, R.: Simulating branching programs with edit distance and friends: or: a polylog shaved is a lower bound made. In: Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC 2016), pp. 375–388. ACM (2016)
2. Abboud, A., Williams, V.V.: Popular conjectures imply strong lower bounds for dynamic problems. In: Proceedings of the 55th Annual Symposium on Foundations of Computer Science (FOCS 2014), pp. 434–443. IEEE (2014)
3. Backurs, A., Indyk, P.: Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In: Proceedigs of the 47th Annual ACM Symposium on Theory of Computing (STOC 2015), pp. 51–58. ACM (2015)
4. Ben-Sasson, E., Viola, E.: Short PCPs with projection queries. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) ICALP 2014. LNCS, vol. 8572, pp. 163–173. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43948-7_14
5. Borodin, A.: On relating time and space to size and depth. SIAM J. Comput. **6**(4), 733–744 (1977)
6. Buss, S.R.: Algorithms for boolean formula evaluation and for tree contraction. In: Arithmetic, Proof Theory and Computational Complexity, pp. 96–115. Oxford University Press (1993)
7. Calabro, C., Impagliazzo, R., Paturi, R.: The complexity of satisfiability of small depth circuits. In: Chen, J., Fomin, F.V. (eds.) IWPEC 2009. LNCS, vol. 5917, pp. 75–85. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-11269-0_6
8. Chen, J., et al.: Tight lower bounds for certain parameterized NP-hard problems. Inf. Comput. **201**(2), 216–231 (2005)
9. Chen, J., Huang, X., Kanj, I.A., Xia, G.: $W$-Hardness under linear FPT-reductions: structural properties and further applications. In: Wang, L. (ed.) COCOON 2005. LNCS, vol. 3595, pp. 975–984. Springer, Heidelberg (2005). https://doi.org/10.1007/11533719_98
10. Chen, J., Huang, X., Kanj, I.A., Xia, G.: Strong computational lower bounds via parameterized complexity. J. Comput. Syst. Sci. **72**(8), 1346–1367 (2006)
11. Dell, H., Husfeldt, T., Marx, D., Taslaman, N., Wahlén, M.: Exponential time complexity of the permanent and the Tutte polynomial. ACM Trans. Algorithms (TALG) **10**(4), 1–32 (2014)
12. Fernau, H., Krebs, A.: Problems on finite automata and the exponential time hypothesis. Algorithms **10**(1), 24 (2017)
13. Furst, M., Saxe, J.B., Sipser, M.: Parity, circuits, and the polynomial-time hierarchy. Math. Syst. Theory **17**(1), 13–27 (1984)
14. Impagliazzo, R., Paturi, R.: On the complexity of $k$-SAT. J. Comput. Syst. Sci. **62**, 367–375 (2001)
15. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? J. Comput. Syst. Sci. **63**, 512–530 (2001)
16. Karakostas, G., Lipton, R.J., Viglas, A.: On the complexity of intersecting finite state automata and NL versus NP. Theor. Comput. Sci. **302**(1), 257–274 (2003)
17. Kasai, T., Iwata, S.: Gradually intractable problems and nondeterministic log-space lower bounds. Math. Syst. Theory **18**(1), 153–170 (1985). https://doi.org/10.1007/BF01699467
18. Kozen, D.: Lower bounds for natural proof systems. In: Proceedings of the 8th Annual Symposium on Foundations of Computer Science (FOCS 1977), vol. 77, pp. 254–266 (1977)

19. Lange, K.-J., Rossmanith, P.: The emptiness problem for intersections of regular languages. In: Havel, I.M., Koubek, V. (eds.) MFCS 1992. LNCS, vol. 629, pp. 346–354. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-55808-X_33
20. Lokshtanov, D., Marx, D., Saurabh, S.: Slightly superexponential parameterized problems. In: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011), pp. 760–776 (2011)
21. Lynch, N.: Log space recognition and translation of parenthesis languages. J. ACM **24**(4), 583–590 (1977)
22. Marx, D.: On the optimality of planar and geometric approximation schemes. In: Proceedings of the 48th Annual Symposium on Foundations of Computer Science (FOCS 2007), pp. 338–348. IEEE (2007)
23. de Oliveira Oliveira, M., Wehar, M.: Intersection non-emptiness and hardness within polynomial time. In: Hoshi, M., Seki, S. (eds.) DLT 2018. LNCS, vol. 11088, pp. 282–290. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98654-8_23
24. Pătraşcu, M., Williams, R.: On the possibility of faster SAT algorithms. In: Proceedings of the 21st Annual Symposium on Discrete Algorithms (SODA 2010), pp. 1065–1075. SIAM (2010)
25. Seiferas, J.I.: Relating refined space complexity classes. J. Comput. Syst. Sci. **14**(1), 100–129 (1977)
26. Todd Wareham, H.: The parameterized complexity of intersection and composition operations on sets of finite-state automata. In: Yu, S., Păun, A. (eds.) CIAA 2000. LNCS, vol. 2088, pp. 302–310. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44674-5_26
27. Wang, J.R., Williams, R.R.: Exact algorithms and strong exponential time hypothesis. In: Encyclopedia of Algorithms, pp. 657–661 (2016)
28. Wehar, M.: Intersection emptiness for finite automata. Honors thesis, Carnegie Mellon University (2012)
29. Wehar, M.: Hardness results for intersection non-emptiness. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) ICALP 2014. LNCS, vol. 8573, pp. 354–362. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43951-7_30
30. Wehar, M.: On the complexity of intersection non-emptiness problems. Ph.D. thesis, University at Buffalo (2016)
31. Williams, R.: Non-uniform ACC circuit lower bounds. In: IEEE 26th Annual Conference on Computational Complexity (CCC 2011), pp. 115–125, June 2011
32. Williams, R.: Improving exhaustive search implies superpolynomial lower bounds. SIAM J. Comput. **42**(3), 1218–1244 (2013)
33. Williams, V.V.: Hardness of easy problems: basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In: Proceedings of the 10th International Symposium on Parameterized and Exact Computation (IPEC 2015), LIPICs, vol. 43, pp. 17–29 (2015)