Information Processing Letters 162 (2020) 105993

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Lengths of words accepted by nondeterministic finite automata

Aaron Potechin [a,1], Jeffrey Shallit [b,*,2]

[a] Department of Computer Science, University of Chicago, Chicago, IL 60637, United States
[b] School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada

ARTICLE INFO

Article history:
Received 14 February 2018
Received in revised form 19 June 2020
Accepted 22 June 2020
Available online 2 July 2020
Communicated by Marek Chrobak

Keywords:
Finite automaton
Formal languages
Theory of computation
Computational complexity
Strong exponential-time hypothesis

ABSTRACT

We consider two natural problems about nondeterministic finite automata (NFA). First, given an NFA M of n states, and a length ℓ, does M accept a word of length ℓ? We show that the classic problem of triangle-free graph recognition reduces to this problem, and give an $O(n^{\omega}(\log n)^{1+\epsilon}\log \ell)$-time algorithm to solve it, where ω is the optimal exponent for matrix multiplication. Second, provided $L(M)$ is finite, we consider the problem of listing the lengths of *all* words accepted by M. Although this problem seems like it might be significantly harder, we show that in the unary case this problem can be solved in $O(n^{\omega}(\log n)^{2+\epsilon})$ time. Finally, we give a connection between NFA acceptance and the strong exponential-time hypothesis.

© 2020 Elsevier B.V. All rights reserved.

1. Introduction

A nondeterministic finite automaton (NFA) $A = (Q, \Sigma, \delta, q_0, F)$ consists of a finite, nonempty set of states $Q = \{q_0, q_1, \ldots, q_{n-1}\}$, an input alphabet Σ, an initial state q_0, a set $F \subseteq Q$ of final states, and a transition function $\delta : Q \times \Sigma \to 2^Q$. This transition function is then extended in the usual way to the domain $Q \times \Sigma^*$. The language accepted by an NFA A is defined to be

$$L(A) = \{x \in \Sigma^* \ : \ \delta(q_0, x) \cap F \neq \emptyset\}.$$

Our NFA's do not have ϵ-transitions. The *transition diagram* of an NFA A is the directed graph $G = G(A)$ with source q_0, sink vertices given by F, and directed edge from p to q labeled a if $q \in \delta(p, a)$. An NFA is *unary* if its input alphabet Σ consists of a single letter. For more information about the model, the reader can consult, for example, [4].

Without loss of generality, we can assume all NFA's under discussion are *initially connected* (i.e., every state is reachable from the start state q_0) and that a final state is reachable from every state. Note that both properties are testable for an NFA A in time linear in the number of edges in its transition diagram.

An NFA A is *acyclic* if its transition diagram has no cycles, or, equivalently, if $L(A)$ is finite. Note that if an n-state NFA is acyclic, then $L(A) \subseteq (\Sigma \cup \{\epsilon\})^{n-1}$.

In this note, we consider the following natural problem about NFA's:

NFA LENGTH ACCEPTANCE

Instance: An n-state NFA $A = (Q, \Sigma, \delta, q_0, F)$ and a length ℓ.

Question: Does A accept a word of length ℓ?

Proposition 1. NFA LENGTH ACCEPTANCE *can be solved in* $O(n^{\omega}(\log n)^{1+\epsilon}(\log \ell))$ *time.*

* Corresponding author.
E-mail addresses: potechin@uchicago.edu (A. Potechin), shallit@uwaterloo.ca (J. Shallit).
[1] This work was supported by the Simons Collaboration for Algorithms and Geometry and by the NSF under agreement No. CCF-1412958.
[2] Supported in part by NSERC Grant 2018-04118.

https://doi.org/10.1016/j.ipl.2020.105993
0020-0190/© 2020 Elsevier B.V. All rights reserved.

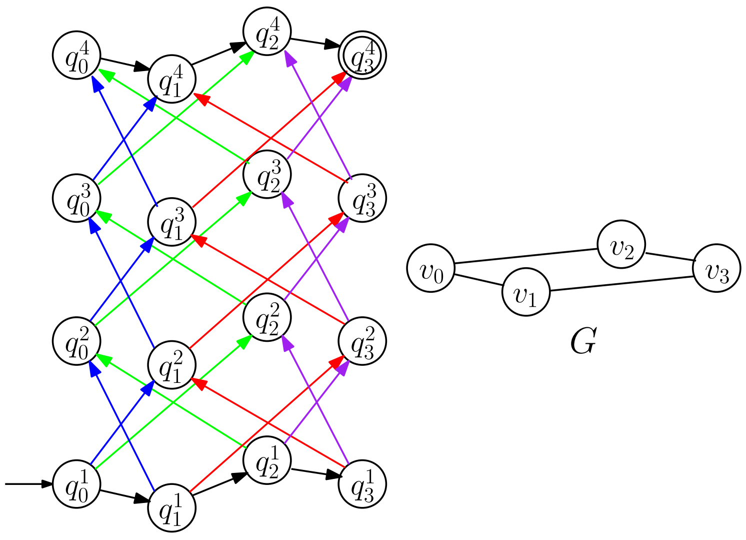

Fig. 1. The reduction when G is a square. Since a square has no triangle, there is no path of length 6 from q_0^1 to q_3^4. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Proof. First, we create a boolean adjacency matrix $M = M(A)$ with a 1 in row i and column j if there is a letter a such that $q_j \in \delta(q_i, a)$. Then standard results on path algebras (see, e.g., [8, Theorem 3.8.3]) imply that A accepts a word of length ℓ if and only if $(M^\ell)_{0,j} = 1$ for some j such that $q_j \in F$. A single Boolean matrix multiplication can be carried out using the usual matrix multiplication algorithms modulo $n+1$, and then converting each element that is at least one to 1. This involves arithmetic on integers of $\log n$ bits (which can be done in $(\log n)^{1+\epsilon}$ time). Raising M to the ℓ power can be done using the usual "binary method of exponentiation" (see, e.g., [7, §4.6.3]) with $\log \ell$ matrix multiplications. □

In the next section, we prove a lower bound on the complexity of this problem, by reducing from the classic problem of triangle-free graph recognition. The same reduction works even if our NFA is restricted to be over a unary alphabet, and even if it is required to be acyclic. In Section 3 we discuss the problem of listing all the elements of $L(A)$ when A is a unary acyclic NFA.

2. A lower bound

In this section, we show that the classic problem of determining whether an undirected graph is triangle-free reduces to `NFA LENGTH ACCEPTANCE` in linear time.

Let G be an undirected graph on n vertices, say $v_0, v_1, \ldots, v_{n-1}$. We assume, without loss of generality, that G has no self-loops. We create a unary acyclic NFA A as follows. The construction consists of four layers, numbered from 1 to 4, with each layer having n states, each corresponding to one of G's vertices. State i in layer j is denoted q_i^j. In the bottom layer (layer 1), we let q_0^1 be the initial state of A and we add a transition from q_i^1 to q_{i+1}^1 for $0 \le i \le n-2$, giving us a linearly-connected chain of states. Next, we add transitions from layer 1 to layer 2, layer 2 to layer 3, and layer 3 to layer 4 as follows: if G has an edge from v_i to v_k, then A has a transition from q_i^j to q_k^{j+1} for $j = 1, 2, 3$. Finally, the top layer (layer 4) has transitions from q_i^4 to q_{i+1}^4 for $0 \le i \le n-2$. The unique final state of A is q_{n-1}^4. See Fig. 1. A similar idea was used in [1].

We claim that a^{n+2} is accepted by A if and only if there exists a triangle in G. To see this, suppose a word a^r is accepted by A. Then an accepting path must

- Begin at the initial state q_0^1;
- Follow i edges in layer 1, ending at q_i^1;
- Transit to layer 2, arriving at some state q_j^2;
- Transit to layer 3, arriving at some state q_k^3;
- Transit to layer 4, arriving at some state q_l^4; and
- Follow $n-1-l$ edges in layer 4, ending at q_{n-1}^4.

The total length of this path is then $r = i+3+n-1-l$. But $r = n+2$ if and only if $i = l$. Then G has edges (v_i, v_j), (v_j, v_k), and (v_k, v_i) and so has the triangle (v_i, v_j, v_k).

For the other direction, suppose G has the triangle (v_i, v_j, v_k). Then there are edges (v_i, v_j), (v_j, v_k), and (v_k, v_i). We construct an acceptance path for a^{n+2} as follows:

- Take the path of transitions from q_0^1 to q_i^1. This path has length i;
- Take the transition from q_i^1 to q_j^2;
- Take the transition from q_j^2 to q_k^3;
- Take the transition from q_k^3 to q_i^4; and
- Take the path of transitions from q_i^4 to q_{n-1}^4. This path has length $n-1-i$.

The accepted word has length $i+3+n-1-i = n+2$. Starting with a graph G of n vertices and m edges, this construction produces a unary acyclic NFA with $4n$ vertices and $3m+2n$ edges.

We have thus proved:

Theorem 2. *There is a linear-time reduction from* TRIANGLE-FREE GRAPH *to* NFA LENGTH ACCEPTANCE.

The fastest general algorithm for TRIANGLE-FREE GRAPH currently known runs in $O(n^{\omega}(\log n)^{1+\epsilon})$ time [6,2,3]. It consists of computing M^3, where M is G's adjacency matrix, and checking if the diagonal contains a 1. This shows that finding a significantly faster algorithm for NFA LENGTH ACCEPTANCE would give a significantly faster algorithm for TRIANGLE-FREE GRAPH.

3. Unary acyclic NFA enumeration

In this section we consider a related problem, which we call UNARY ACYCLIC NFA ENUMERATION:

Instance: a unary acyclic n-state NFA A.

Problem: to enumerate (list) the elements of $L(A)$.

At first glance, this problem seems like it might be harder than NFA LENGTH ACCEPTANCE, since it requires checking the lengths of all possible accepted words, rather than a single word. Nevertheless, we give a $O(n^{\omega}(\log n)^{2+\epsilon})$-time algorithm for the problem. Since the same argument giving a linear-time reduction from TRIANGLE-FREE GRAPH to NFA LENGTH ACCEPTANCE works for reducing TRIANGLE-FREE GRAPH to UNARY ACYCLIC NFA ENUMERATION, it is unlikely we can greatly improve our algorithm, unless a significant advance is made.

The naive approach to solving UNARY ACYCLIC NFA ENUMERATION is to maintain a list L of the states of A (represented, say, as a bit vector) and update this list as we read additional symbols of input. If A has n states, then the longest word accepted is of length $\leq n-1$. To update L after reading each new symbol potentially requires a union of n sets, each with at most n elements. Thus the total running time is $O(n^3)$.

We consider a different approach. Suppose $A = (Q, \Sigma, \delta, q_0, F)$ has n states, labeled $q_0, q_1, \ldots, q_{n-1}$. We create a new NFA $A' = (Q', \{a\}, \delta', q'_0, F)$, as follows. Let 2^k be the smallest power of 2 that is $\geq n$. Define $Q' = Q \cup \{p_0, p_1, \ldots, p_{2^k-1}\}$. Let $q'_0 = p_0$ be the new initial state, and, in addition to the transitions already present in A, define δ' by adding additional transitions from p_i to p_{i+1} for $0 \leq i < p_{2^k-1}$, and from p_{2^k-1} to q_0. Let M' be the adjacency matrix of A'.

Now A accepts a word of length i if and only if there is a path of length i from q_0 to a final state of A, if and only if there is a path of length 2^k from p_i to a final state of A'. Thus we can compute all words accepted by A with a single exponentiation of M' to the appropriate power.

We now compute M'^{2^k} using exactly k Boolean matrix multiplications, through repeated squaring. To determine if a^i is accepted by A, it suffices to check the entry corresponding to the row for p_i and the columns for the final states of A'. We do this for each possible length, 0 through $n-1$, and so the total cost is $O(n^{\omega}(\log n)^{2+\epsilon} + n^2)$.

We have proved

Theorem 3. *If M is a unary NFA that accepts a finite language L, we can enumerate the elements of L in $O(n^{\omega}(\log n)^{2+\epsilon})$ bit operations, where ω is the optimal exponent for matrix multiplication.*

This result previously appeared in [8, §3.8].

4. Hardness of NFA acceptance

In this section, we consider the following decision problem:

NFA ACCEPTANCE

Input: An NFA M of total size m (counting states and transitions) and an input x of length ℓ.

Question: Does M accept x?

The obvious algorithm for this problem keeps track of the current set of states and updates it for each new input letter read; it runs in $O(\ell m)$ time.

In this section, we show that in the case when the NFA is sparse (i.e., m is not much larger than n, the number of states of the NFA), significantly improving this algorithm would disprove the strong exponential time hypothesis (SETH) [5]. However, this does not rule out an improvement when the NFA is dense, and we leave it as an open problem to either find a significant improvement to this algorithm, or show why such an improvement is unlikely.

Recall the following decision problem (e.g., [11]):

ORTHOGONAL VECTORS

Input: Two lists $(v_i)_{1 \leq i \leq n}$ and $(w_i)_{1 \leq i \leq n}$ of boolean vectors of dimension d.

Question: do there exist i, j such that the boolean product $v_i \cdot w_j = 0$?

Theorem 4. ORTHOGONAL VECTORS *reduces in linear time and log space to acyclic* NFA ACCEPTANCE.

Proof. The idea is to create an NFA M that accepts the input $(0w_1 0)(0w_2 0)\cdots(0w_n 0)$ if and only if there exist i, j such that $v_i \cdot w_j = 0$. We do this by using nondeterminism to "guess" the appropriate v_i and w_j, and then compute the dot product with a small gadget. See Fig. 3.

The NFA is built out of some simple NFA gadgets M_i, one for each v_i. On input w, the NFA M_i accepts iff $v_i \cdot w = 0$. If the vectors are of length d, this can be done with $d+1$ states. This is illustrated in Fig. 2 for $w = [1, 0, 0, 1]$.

The NFA M has the following layers:

1. In the bottom layer, path of length $(n-1)(d+2)$, where we assign the state name a_j to the $(j-1)(d+2)$-th state on this path. The start state is a_1. The boxes labeled "Path to skip w_i" just read n symbols, no matter what they are.
2. A distinguished state x that replaces the start state of each of the gadgets M_i, $1 \leq i \leq n$.
3. The gadgets M_i, $1 \leq i \leq n$, except for their final states.
4. A distinguished state y that replaces the final state for each gadget.

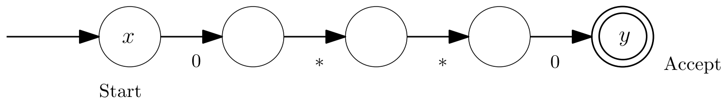

Fig. 2. The gadget for testing whether $v \cdot w$ is 0, when $w \in \{0,1\}^4$ and $v = [1,0,0,1]$. Here $*$ means that the NFA can take this edge regardless of what the input bit is.

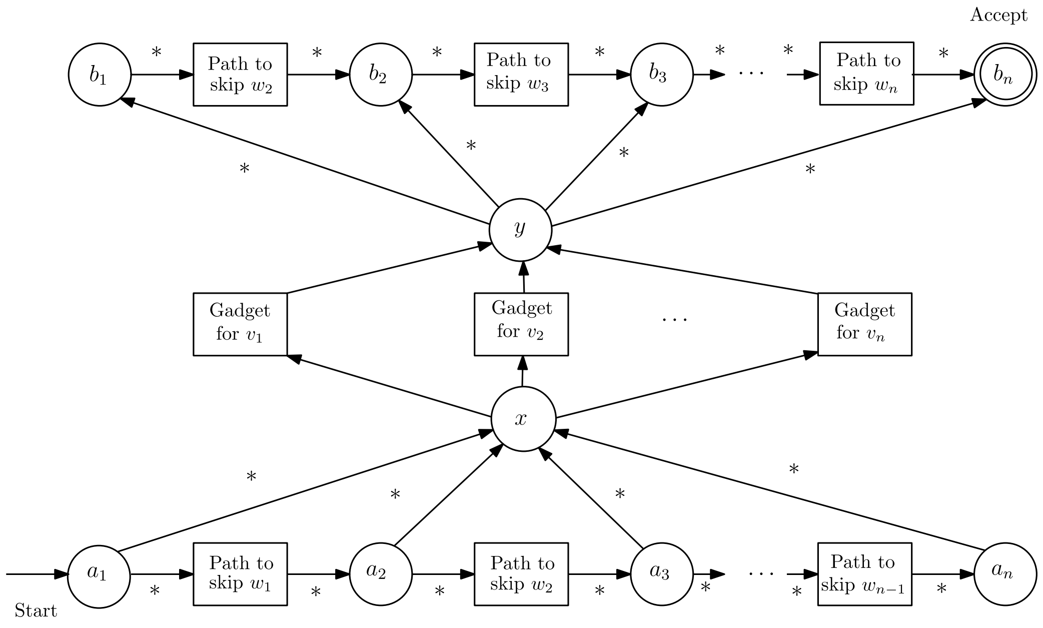

Fig. 3. Structure of the acyclic NFA solving orthogonal vectors.

5. In the top layer, a path of length $(n-1)(d+2)$, where we assign the label b_j to the $(j-1)(d+2)$-th state on this path. The final state is b_n.

The transitions for M are as follows: except for transitions within the gadgets M_j, all of these transitions can be made regardless of the input, indicated again by $*$.

1. For the path containing the states a_j, at each a_j we either choose to transition to x, which means that we read w_j from the input, or we can transition to the next state of the path (unless we are at a_n, in which case we can only transition to x). At each other state in the path, we can only transition to the next state of the path.
2. From x we can transition to the start state of any gadget M_i, which means that we will check $v_i \cdot w_j$.
3. We have the transitions for each gadget M_i, except that if we would transition to the final state of M_i, we instead transition to y.
4. From y we can transition to any b_j. This flexibility allows the NFA to end up at the final state after the correct number of steps.
5. For the path containing the states b_j, at each state we can only transition to the next state in the path.

The total number of states and transitions are both $O(dn)$.

We now argue briefly that the construction is correct. Suppose there are some i, j such that $v_i \cdot w_j = 0$. Then in the bottom layer we read $(0w_1 0)(0w_2 0)\cdots(0w_{j-1}0)$. We then read a 0 to get to x, and successfully traverse the gadget corresponding to v_i to get to y. Then in the top layer we read $(0w_{j+1}0)\cdots(0w_n 0)$ and accept.

On the other hand, if the NFA accepts $(0w_1 0)\cdots(0w_n 0)$, then this can only happen by a path traversing the bottom layer, transiting to x, successfully exiting the gadget corresponding to some v_i and exiting at y, and then traversing the top layer and ending at y. This path forces $v_i \cdot w_j = 0$ for some i, j. □

Corollary 5. *If there is an algorithm for* NFA ACCEPTANCE *that runs in $O(n^{2-\epsilon})$ time, then* SETH *is false.*

Proof. Such an algorithm would imply an algorithm for ORTHOGONAL VECTORS that runs in the same time bound. We then use a result of Williams (e.g., [9] or [10, Thm. 1, p. 22]). □

5. Conclusion

In this paper, we analyzed the complexity of the acyclic NFA acceptance problem, which asks if an acyclic NFA accepts a given input. In the case where the NFA is unary, we showed that there is an algorithm using matrix multiplication that runs in $\tilde{O}(n^\omega)$ time, which in fact enumerates all input lengths that are accepted. We also showed that we can reduce the triangle detection problem to unary acyclic NFA acceptance, so significantly improving on this algorithm for unary acyclic NFA acceptance would imply a breakthrough for triangle detection. In the general case, we show that significantly improving the trivial $O(nm)$ algorithm (where n is the input length and m is the number of edges in the NFA) when m is $O(n)$ would imply that the strong exponential time hypothesis (SETH) is false.

That said, there are a number of open questions remaining. First, what bounds can we show for acyclic NFA acceptance when the NFA is dense? In particular, can we prove a $\tilde{\Omega}(n^3)$ lower bound under some assumption? Second, can we reduce acyclic unary NFA acceptance and acyclic NFA acceptance to other problems?

Declaration of competing interest

We declare that we have no conflicts of interest.

References

[1] A. Abboud, V.V. Williams, Popular conjectures imply strong lower bounds for dynamic problems, preprint, available at https://arxiv.org/abs/1402.0054, 2014.

[2] N. Alon, R. Yuster, U. Zwick, Color-coding, J. Assoc. Comput. Mach. 42 (1995) 844–856.

[3] N. Alon, R. Yuster, U. Zwick, Finding and counting given length cycles, Algorithmica 17 (1997) 209–223.

[4] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 1979.

[5] R. Impagliazzo, R. Paturi, The complexity of k-SAT, in: Proc. 14th IEEE Conf. on Computational Complexity, IEEE Computer Society, 1999, pp. 237–240.

[6] A. Itai, M. Rodeh, Finding a minimum circuit in a graph, SIAM J. Comput. 7 (1978) 413–423.

[7] D.E. Knuth, The Art of Computer Programming. Volume 2: Seminumerical Algorithms, 2nd edition, Addison-Wesley, 1981.

[8] J. Shallit, A Second Course in Formal Languages and Automata Theory, Cambridge University Press, 2009.

[9] R. Williams, A new algorithm for optimal 2-constraint satisfaction and its implications, Theor. Comput. Sci. 348 (2005) 357–365.

[10] V. Williams, Hardness of easy problems: basing hardness on popular conjectures such as the strong exponential time hypothesis, in: T. Husfeldt, I. Kanj (Eds.), Proc. 10th Int'l. Symp. Parameterized and Exact Computation (IPEC 2015), LIPICS Schloss Dagstuhl, 2015, pp. 16–29, Leibniz Int'l Proceedings in Informatics.

[11] R. Williams, H. Yu, Finding orthogonal vectors in discrete structures, in: Proc. 25th ACM-SIAM Symp. Discrete Algorithms (SODA), 2014, pp. 1867–1877.