On the Length of Shortest Strings Accepted by Two-Way Finite Automata

Egor Dobronravov, Nikita Dobronravov, and Alexander Okhotin(✉)

St. Petersburg State University, 7/9 Universitetskaya nab.,
Saint Petersburg 199034, Russia
yegordobronravov@mail.ru, dobronravov1999@mail.ru,
alexander.okhotin@spbu.ru

Abstract. Given a two-way finite automaton recognizing a non-empty language, consider the length of the shortest string it accepts, and, for each $n \geqslant 1$, let $f(n)$ be the maximum of these lengths over all n-state automata. It is proved that for n-state two-way finite automata, whether deterministic or nondeterministic, this number is at least $\Omega(8^{n/5})$ and less than $\binom{2n}{n+1}$, with the lower bound reached over an alphabet of size $\Theta(n)$. Furthermore, for deterministic automata and for a fixed alphabet of size $m \geqslant 1$, the length of the shortest string is at least $e^{(1+o(1))\sqrt{mn(\log n - \log m)}}$.

1 Introduction

For a one-way nondeterministic finite automaton (1NFA) with n states recognizing a non-empty language, the length of the shortest string it accepts is the length of the shortest path to an accepting state in the transition graph, and is accordingly at most $n-1$. For other kinds of automata, the question of finding the exact length of the shortest string in the worst case is much more involved, and has been a subject of some research. Ellul et al. [4] proved that the greatest length of the shortest string *not* accepted by an n-state 1NFA is exponential in n. The length of the shortest string in the intersection of an m-state and an n-state deterministic automata (1DFA), as shown by Alpoge et al. [1], can be up to $mn-1$ for relatively prime m, n. Chistikov et al. [2] investigated the length of the shortest string for counter automata. For the intersection of a language defined by a formal grammar of a given size and a regular language, the length of a shortest string was estimated by Pierre [11].

This paper investigates the length of a shortest string accepted by a *two-way finite automaton*. A simple upper bound on this length follows from the work of Kapoutsis [7], who proved that every n-state 2NFA can be simulated by an 1NFA with $\binom{2n}{n+1}$ states; this binomial coefficient is of the order $\frac{1}{\sqrt{\pi n}}4^n$. Therefore, the shortest string accepted by an n-state 2NFA is of length at most $\binom{2n}{n+1} - 1$.

Supported by Russian Science Foundation, project 18-11-00100.

P. Hofman and M. Skrzypczak (Eds.): DLT 2019, LNCS 11647, pp. 88–99, 2019.
https://doi.org/10.1007/978-3-030-24886-4_6

Kapoutsis [7] also proved that this transformation of two-way automata to one-way automata is optimal in the worst case, that is, for every n, there is a language L_n recognized by an n-state 2DFA, but by no 1NFA with fewer than $\binom{2n}{n+1}$ states. However, since all strings in this language are of length 4, this example does not imply any lower bound on the length of the shortest string.

In this paper, the greatest length of the shortest string is determined up to a constant factor in the exponent, as $2^{\Theta(n)}$. First, there is a simple construction of an n-state 2DFA with a shortest string of length ca. $2^{n/2}$. This construction is then improved to obtain n-state 2DFA with shortest strings of length ca. $8^{n/5}$. In both cases, the size of the alphabet is exponential in n. For a fixed alphabet of size m, a series of n-state automata with shortest strings of length $e^{(1+o(1))\sqrt{mn \ln \frac{n}{m}}}$ is constructed.

2 Two-Way Finite Automata

Definition 1. *A* nondeterministic two-way finite automaton *(2NFA) is a quintuple $\mathcal{A} = (\Sigma, Q, q_0, \delta, F)$, in which:*

- *Σ is a finite alphabet, which is extended with a left end-marker $\vdash \notin \Sigma$, and a right end-marker $\dashv \notin \Sigma$;*
- *Q is a finite set of states;*
- *$Q_0 \in Q$ is the set of initial states;*
- *$\delta \colon Q \times (\Sigma \cup \{\vdash, \dashv\}) \to 2^{Q \times \{-1,+1\}}$ is the transition function, which lists possible transitions in a certain state while observing a certain tape symbol;*
- *$F \subseteq Q$ is the set of accepting states, effective at the right end-marker $\dashv$.*

Given an input string $w \in \Sigma^*$, a 2NFA operates on a read-only tape containing this string enclosed within end-markers ($\vdash w \dashv$). A 2NFA begins its computation in any initial state with the head observing the left end-marker ($\vdash$). At every step of the computation, when A is in a state $q \in Q$ and observes a square of the tape containing a symbol $a \in \Sigma \cup \{\vdash, \dashv\}$, the transition function specifies a set $\delta(q, a) \subseteq Q \times \{-1, +1\}$ of all the allowed actions, each being a pair of the next state and the direction of head's motion. If $\delta(q, a)$ contains multiple elements, then multiple continuations are possible, and, accordingly, a 2NFA may have multiple computations on the same input string. If the automaton eventually reaches an accepting state while at the right end-marker ($\dashv$), then this is an accepting computation.

The set of strings, on which there is at least one accepting computation*, is the language recognized by the 2NFA, denoted by $L(\mathcal{A})$.*

Other types of finite automata are obtained by restricting 2NFA. An automaton is *deterministic* (2DFA), if there is at most one possible action in each configuration, that is, if $|\delta(q, a)| \leqslant 1$ for all q and a.

A two-way automaton (2DFA for 2NFA) is called *sweeping* [12], if it can change its direction of motion only at the end-markers, and thus operates in alternating left-to-right sweeps and right-to-left sweeps. More precisely, the set of states Q is split into two disjoint subsets of *right-bound states* Q_{+1} and *left-bound states* Q_{-1}, so that all transitions in Q_d, except the transition on end-markers, move the head in the direction d.

For a less restrictive notion of a *direction-determinate automaton* [9], it is only required that every state $q \in Q$ can be entered by transitions from a single direction $d(q) \in \{-1, +1\}$. Every sweeping automaton is direction-determinate, but not vice versa.

An automaton is *one-way* (1NFA or 1NFA), if all its transitions move its head to the right, so that the automaton makes a single left-to-right pass, accepting or rejecting in the end. In one-way automata, the end-markers are of no use and are usually omitted from the definition.

3 Upper Bound

An upper bound on the length of a shortest string accepted by a two-way finite automaton follows from the known transformation of two-way automata to 1NFA, which is optimal for alphabets of unbounded size.

Theorem A (Kapoutsis [7]). *For every n-state 2NFA over an alphabet Σ, there exists a 1NFA with $\binom{2n}{n+1}$ states, which recognizes the same language. Conversely, for every n, there is such an alphabet Σ_n of size $\Theta(n^n)$, and such a language $L_n \subseteq \Sigma_n^*$ recognized by an n-state 2DFA, that every 1NFA recognizing L_n must have at least $\binom{2n}{n+1}$ states.*

Taking into account that the length of a shortest string accepted by a k-state 1NFA is at most $k-1$, this has the following immediate consequence.

Corollary 1. *For every n-state 2NFA, the length of the shortest string it accepts is at most $\binom{2n}{n+1} - 1$.*

For direction-determinate automata, the method of Kapoutsis [7] can be adapted to produce fewer states. As proved by Geffert and Okhotin [6], the following transformation is optimal for alphabets with three or more symbols.

Theorem B (Geffert and Okhotin [6]). *For every n-state direction-determinate 2NFA over an alphabet Σ, there is a 1NFA with $\binom{n}{\lfloor n/2 \rfloor}$ states that recognizes the same language. Conversely, for every n, there exists a language L_n over a fixed 3-symbol alphabet, recognized by an n-state sweeping 2DFA, with the property that every 1NFA recognizing L_n has at least $\binom{n}{\lfloor n/2 \rfloor}$ states.*

Corollary 2. *The length of the shortest string accepted by an n-state direction-determinate 2NFA is at most $\binom{n}{\lfloor n/2 \rfloor}$.*

Using Stirling's approximation, these binomial coefficients are estimated as $\binom{2n}{n+1} = (1+o(1))\frac{1}{\sqrt{\pi n}}4^n$ and as $\binom{n}{\lfloor n/2 \rfloor} = (1+o(1))\sqrt{\frac{2}{\pi n}}2^n$, respectively.

This upper bound is the same for 2NFA and for 2DFA. In fact, as indicated by the following simple result, the length of shortest strings for 2NFA is the same as for 2DFA with the same number of states. However, there may still be some differences in the size of the alphabet necessary to achieve that bound.

Lemma 1. *For every n-state 2NFA (sweeping 2NFA) over an alphabet Σ, there exists an n-state 2DFA (sweeping 2DFA, respectively) over some alphabet Γ, which has the same length of the shortest accepted string. The number of symbols in Γ is at most $(2n)^n$ times the size of Σ.*

Proof. For every symbol $a \in \Sigma$, consider the 2NFA's transitions by that symbol. At each of the n states, there can be up to $2n$ possible transitions. For every choice of these transitions, let Γ contain a new symbol, which is a marked with that choice. The 2DFA's transitions by the marked symbols are defined to act deterministically according to that choice.

Then, for every string w accepted by the 2NFA, the new 2DFA accepts some string w' of the same length, with each symbol marked with the choices made by the 2NFA on the corresponding symbol of w. Conversely, for each string accepted by the 2DFA, the original 2NFA accepts the same string without markings. □

In view of this observation, the rest of this paper concentrates on the case of deterministic two-way automata.

4 Simple Lower Bound

The following two-way automata have long shortest accepted strings.

Lemma 2. *For every odd number $m \geqslant 1$, there exists a $2m$-state sweeping 2DFA, defined over an m-symbol alphabet, which recognizes a singleton language $\{w\}$, with $|w| = 2^m - 1$.*

Proof. Let the alphabet be $\Sigma = \{a_1, \ldots, a_m\}$. The automaton shall make m passes over the string. At each i-th pass, with $i \in \{1, \ldots, m\}$, the automaton marks whether it has encountered any symbol a_i since the last symbol with a number $i+1$ or greater. When a symbol with a number $i+1$ or greater is read, the automaton makes sure that it encountered exactly one a_i since the previous such symbol, and resets the mark. The same check is done in the end of each pass, upon seeing one of the end-markers.

The states are of the form $q_j^{(i)}$, with $i \in \{1, \ldots, m\}$ and $j \in \{0, 1\}$, which indicates making the i-th pass, having seen ($j = 1$) or not seen ($j = 0$) any symbol a_i.

$$Q = \{\, q_j^{(i)} \mid i \in \{1, \ldots, m\},\ j \in \{0, 1\} \,\}$$

The initial state is $q_0^{(1)}$, in which the automaton begins the first pass.

$$\delta(q_0^{(1)}, \vdash) = (q_0^{(1)}, +1)$$

For each i-th pass, let $d_i \in \{+1, -1\}$ be the direction of this pass, with $d_i = +1$ for odd i, and $d_i = -1$ for even i. The transitions at the i-th pass set the mark upon seeing the corresponding symbol a_i.

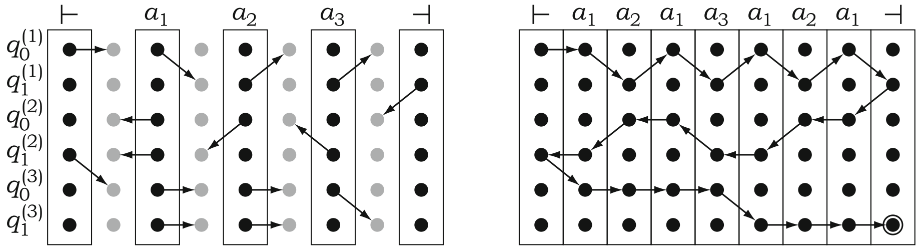

Fig. 1. A 6-state 2DFA with a shortest string of length $2^3 - 1 = 7$, constructed as in Lemma 2, with $m = 3$.

$$\delta(q_0^{(i)}, a_i) = (q_1^{(i)}, d_i), \qquad \text{for } i \in \{1, \ldots, m\},$$

For any of the symbols $a_{i+1}, \ldots, a_m$, the mark of having seen a_i is checked and then reset to false.

$$\delta(q_1^{(i)}, a_t) = (q_0^{(i)}, d_i), \qquad \text{for } i \in \{1, \ldots, m\}$$

All other input symbols are ignored, that is, the automaton passes them without changing its state.

$$\delta(q_j^{(i)}, a_t) = (q_j^{(i)}, d_i), \qquad \text{for } i \in \{1, \ldots, m\},\ j \in \{0, 1\},\ t \in \{1, \ldots, i-1\}$$

On each end-marker, the mark of having seen a_i is again checked, and then the automaton switches from the i-th pass to the $(i+1)$-th.

$$\begin{aligned} \delta(q_1^{(i)}, \dashv) &= (q_0^{(i+1)}, -1), && \text{for all odd } i < m \\ \delta(q_1^{(i)}, \vdash) &= (q_0^{(i+1)}, -1), && \text{for all even } i < m \end{aligned}$$

The last pass leads the automaton to the right end-marker (⊣), where it accepts in the state $q_1^{(m)}$. These transitions are illustrated in Fig. 1 for $m = 3$, along with the shortest accepted string of length 7.

Actually, the automaton always accepts a unique string. The following strings $w_0, w_1, \ldots, w_m$ are defined, with w_3 illustrated in Fig. 1(right).

$$\begin{aligned} w_0 &= \varepsilon \\ w_i &= w_{i-1} a_i w_{i-1} && (1 \leqslant i \leqslant m) \end{aligned}$$

The length of each w_i is $2^i - 1$, and one can verify that the last string w_m is accepted by tracing the automaton's computation. The goal is now to prove that w_m is the unique accepted string.

Claim. Let w be any string accepted by the automaton, let $i \in \{0, 1, \dots, m\}$, and assume that the tape $\vdash w \dashv$ has a substring $¢u\$$, where $¢ \in \{\vdash, a_{i+1}, \dots, a_m\}$, $\$ \in \{\dashv, a_{i+1}, \dots, a_m\}$ and $u \in \{a_1, \dots, a_i\}^*$. Then, $u = w_i$.

The claim is proved by induction on i, from 0 to m.

The base case $i = 0$ is trivial: the string u is defined over an empty alphabet, and therefore must be ε.

For the induction step for i, consider the i-th pass in the automaton's computation, as it passes through the substring $¢u\$$, with the direction of traversal determined by the parity of i. As it enters the substring u from one side, the state is reset to $q_0^{(i)}$, and the automaton must emerge on the other side in the state $q_1^{(i)}$. For this to happen, u must contain exactly one instance of a_i, and therefore, $u = u_0 a_i u_1$, for some substrings $u_0, u_1 \in \{a_1, \dots, a_{i-1}\}^*$. By the induction hypothesis for the substrings u_0 and for u_1, which are delimited by appropriate symbols, both are equal to w_{i-1}. Therefore, $u = w_{i-1} a_i w_{i-1} = w_i$.

For $i = m$, the above claim asserts that every accepted string must be w_m. □

For each number n, Lemma 2 gives an n-state 2DFA with the shortest accepted string of length $2^{\lfloor \frac{n}{2} \rfloor} \approx 1.414^n$. Together with the upper bound $\binom{2n}{n+1} - 1$ given in Corollary 1, this shows that the maximal length of the shortest string for n-state 2DFA and 2NFA is between $(\sqrt{2})^n$ and 4^n. The question is, what is the precise base?

An easy improvement to this construction is given by *counting to 3* rather than to 2; then, the shortest string is of length $3^{\lfloor \frac{n-1}{3} \rfloor} \approx 1.442^n$. A construction that further improves this lower bound is presented in the next section.

5 Improved Lower Bound

A proposed improvement to the lower bound in Lemma 2 is based on the following example of an automaton with a shortest string of length 7, which has as few as 5 states, cf. 6 states in the automaton provided by Lemma 2.

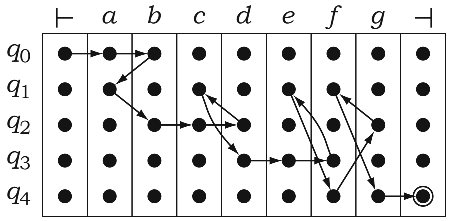

Fig. 2. A 5-state 2DFA with a shortest string of length 7, presented in Example 1.

Example 1. Let $\mathcal{A}$ be a 2DFA over the alphabet $\Sigma = \{a, b, c, d, e, f, g\}$, with the states $Q = \{q_0, q_1, q_2, q_3, q_4\}$, where q_0 is initial and q_4 is accepting, and with the following transitions: $\delta(q_0, \vdash) = (q_0, +1)$, $\delta(q_0, a) = (q_0, +1)$, $\delta(q_1, a) = (q_2, +1)$, $\delta(q_0, b) = (q_1, -1)$, $\delta(q_2, b) = (q_2, +1)$, $\delta(q_1, c) = (q_3, +1)$, $\delta(q_2, c) = (q_2, +1)$, $\delta(q_2, d) = (q_1, -1)$, $\delta(q_3, d) = (q_3, +1)$, $\delta(q_1, e) = (q_4, +1)$, $\delta(q_3, e) = (q_3, +1)$, $\delta(q_1, f) = (q_4, +1)$, $\delta(q_3, f) = (q_1, -1)$, $\delta(q_4, f) = (q_2, +1)$, $\delta(q_2, g) = (q_1, -1)$, $\delta(q_4, g) = (q_4, +1)$. Then, $\mathcal{A}$ is direction-determinate, with $d(q_1) = -1$ and $d(q) = +1$ in all other states.

The shortest string accepted by $\mathcal{A}$ is $w = abcdefg$, as illustrated in Fig. 2. To see that w is indeed the shortest string accepted by $\mathcal{A}$, it is sufficient to transform this automaton to the minimal equivalent partial 1DFA, which is presented in Fig. 3. The shortest string is clearly visible in the figure.

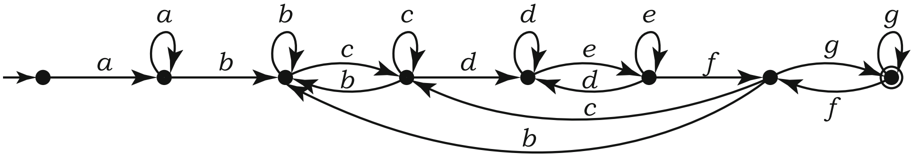

Fig. 3. The minimal 1DFA recognizing the same language as the 2DFA in Fig. 2.

The following lemma iteratively applies this example to construct arbitrarily large direction-determinate 2DFA with shortest accepted strings of length greater than in Lemma 2.

Lemma 3. *Let $\mathcal{A} = (\Sigma, Q, q_0, \delta, F)$ be a k-state direction-determinate 2DFA over some alphabet Σ, in which, for every state $q \in Q$, at most one of the following conditions may hold: (i) $\delta(q_0, \vdash) = q$; (ii) $q \in F$; (iii) $\delta(q, \vdash)$ is defined and $q \neq q_0$; (iv) $\delta(q, \dashv)$ is defined. Let $\ell - 1$ be the length of the shortest string accepted by $\mathcal{A}$. Then, for every odd number $m \geqslant 3$, there exists a km-state direction-determinate 2DFA $\mathcal{B}_m$, defined over an alphabet of size $m \cdot |\Sigma|$, which has the shortest accepted string of length $\ell^m - 1$.*

The construction in Lemma 3 actually generalizes that of Lemma 2, and before presenting it in the general case, it is useful to see how the earlier given construction fits the statement of Lemma 3. Let the base automaton $\mathcal{A}$ be a 2-state partial 1DFA recognizing the language $\{a\}$ over a one-symbol alphabet $\Sigma = \{a\}$. Then, during each i-th pass, the automaton in Lemma 2 simulates the base 1DFA on the symbols a_i, ignoring any symbols $\{a_1, \ldots, a_{i-1}\}$ encountered. A separate instance of the base 1DFA is executed for each block delimited by two symbols in $\{a_{i+1}, \ldots, a_m, \vdash, \dashv\}$.

It remains to extend the same construction to an arbitrary base automaton.

Proof. Let $\mathcal{A}$ be the given k-state direction-determinate 2DFA over an alphabet Σ. Let $Q = Q_{+1} \cup Q_{-1}$ be $\mathcal{A}$'s set of states, where the states in each Q_d are enterable only in the direction d. For $q \in Q_d$, denote by $d(q) = d$ the direction in which q is enterable. Let $F \subseteq Q$ be the set of accepting states of $\mathcal{A}$, effective at the right end-marker ($\dashv$).

The alphabet Ω of the new 2DFA $\mathcal{B}$ consists of symbols of the form $a^{(i)}$, where $a \in \Sigma$ is an input symbol of $\mathcal{A}$, and $i \in \{1, \ldots, m\}$.

$$\Sigma_i = \{\, a^{(i)} \mid a \in \Sigma \,\} \qquad (1 \leqslant i \leqslant m)$$

$$\Omega = \bigcup_{i=1}^{m} \Sigma_i$$

The new automaton's computation is organized into m passes. At each i-th pass, with $i \in \{1, \ldots, m\}$, the automaton $\mathcal{B}$ interprets its tape as $\vdash u_0 \#_1 u_1 \#_2 \ldots \#_z u_z \dashv$, where the separators $\#_1, \ldots, \#_z$ are any symbols of the form $a^{(j)}$, with $j > i$, and the substrings $u_0, \ldots, u_z$ are defined over the alphabet $\Sigma_1 \cup \ldots \cup \Sigma_i$. The substrings are processed one by one, from left to right for odd i, and from right to left for even i. For each string u_j, the automaton $\mathcal{B}$ simulates the computation of $\mathcal{A}$ on that string (if i is odd) or on its reverse (if i is even), taking into account only symbols $a^{(i)}$, with $a \in \Sigma$. All other symbols $a^{(j)}$, with $j < i$ and $a \in \Sigma$, are ignored by passing over them without changing the state or the direction; this is possible, because $\mathcal{A}$ is direction-determinate.

Each separator $\$_i$ acts both as a right delimiter for u_{i-1} and as a left delimiter for u_i. Conditions (i–iv) ensure that whenever $\mathcal{B}$ visits such a separator, it can always tell whether it is currently simulating a computation of $\mathcal{A}$ on u_{i-1} or on u_i.

The states of $\mathcal{B}$ are of the form $q^{(i)}$, which means simulating $\mathcal{A}$ in a state $q \in Q$ at the i-th pass.

$$Q = \{\, q^{(i)} \mid q \in Q,\ i \in \{1, \ldots, m\} \,\}$$

At odd-numbered passes, the substrings $u_0, \ldots, u_z$ are processed from left to right, and from right to left at even-numbered passes. Let $d(i)$ be the general direction of traversal at the i-th pass, defined by $d(i) = +1$ for odd i and $d(i) = -1$ for even i. The automaton $\mathcal{B}$ shall be direction-determinate with $d(q^{(i)}) = d(q) \cdot d(i)$.

Let q_0 be the initial state of $\mathcal{A}$, in which it makes the initial transition $\delta(q_0, \vdash) = r$. Then, the initial state of $\mathcal{B}$ is $q_0^{(1)}$, with the following initial transition.

$$\delta'(q_0^{(1)}, \vdash) = (r^{(1)}, +1)$$

At every i-th pass, with $i \in \{1, \ldots, m\}$, each $\mathcal{A}$'s transition $\delta(q, a) = (r, d(r))$, with $a \in \Sigma$ and $q, r \in Q$, is implemented by the following transition in $\mathcal{B}$.

$$\delta'(q^{(i)}, a^{(i)}) = (r^{(i)}, d(r) \cdot d(i))$$

Note that $\mathcal{A}$'s direction "+1" becomes "the direction of the i-th pass", whereas $\mathcal{A}$'s direction "−1" now goes in the opposite direction to the i-th pass' direction.

Each lesser symbol $a^{(j)}$, with $j < i$ and $a \in \Sigma$, is ignored by continuing in the same direction. This is where the direction-determinacy of $\mathcal{A}$ becomes essential.

$$\delta'(q^{(i)}, a^{(j)}) = (q^{(i)}, d(q) \cdot d(i)), \qquad \text{where } q \in Q,\ j < i,\ a \in \Sigma$$

Next, let $¢_i$ and $\$_i$ be the end-markers at which the i-th pass begins and ends, respectively ($¢_i = \vdash$ and $\$_i = \dashv$ for i odd, and vice versa for i even). For each $\mathcal{A}$'s transition $\delta(q, \dashv) = (r, -1)$ turning at the right end-marker, with $d(q) = +1$ and $d(r) = -1$, the new automaton executes the same turn on any separator symbols.

$$\delta'(q^{(i)}, s) = (r^{(i)}, -d(i)), \qquad \text{for } s \in \{\$_i\} \cup \Sigma_{i+1} \cup \ldots \cup \Sigma_m$$

Each turn at the left end-marker, $\delta(q, \vdash) = (r, +1)$, with $q \neq q_0$, $d(q) = -1$ and $d(r) = +1$, is implemented similarly.

$$\delta'(q^{(i)}, s) = (r^{(i)}, d(i)), \qquad \text{for } s \in \{¢_i\} \cup \Sigma_{i+1} \cup \ldots \cup \Sigma_m$$

When $\mathcal{A}$ is about to accept at its right end-marker ($\dashv$) in a state $q \in F$, the simulating automaton $\mathcal{B}$ proceeds through a separator symbol to the next block, implementing the initial transition $\delta(q_0, \vdash) = r$ for the next block without actually entering q_0.

$$\delta'(q^{(i)}, s) = (r^{(i)}, d(i)), \qquad \text{for } s \in \Sigma_{i+1} \cup \ldots \cup \Sigma_m$$

At the end of the i-th pass, when $\mathcal{A}$'s accepting state $q \in F$ is reached while at the appropriate end-marker, $\mathcal{B}$ proceeds to the next pass by simulating the transition $\delta(q_0, \vdash) = r$ (as long as i is less than m).

$$\delta'(q^{(i)}, \$_i) = (r^{(i+1)}, d(i+1))$$

If that happens for $i = m$, the automaton $\mathcal{B}$ accepts instead.

Since the $\mathcal{B}$ proceeds to the next pass only at the end-markers, in order to accept a string, it needs to make $\frac{m-1}{2}$ left-to-right passes and $\frac{m+1}{2}$ right-to-left passes over the string, with each i-th pass made in the states from Q_i. The moment when the automaton enters any state from Q_{i+1}, it is said to have *completed the i-th pass;* the m-th pass is completed upon acceptance.

For every $i \in \{1, \ldots, m\}$, let $h_i \colon \Omega^* \to \Sigma^*$ be a homomorphism defined by $h_i(a_i) = a$ and $h_i(a_j) = \varepsilon$ for $j \neq i$: this is a projection to Σ_i.

Claim 1. *Let $w \in \Omega^*$ be any string, let $i_0 \in \{0, \ldots, m\}$. Then, $\mathcal{B}$ completes the i_0-th pass in its computation on w if and only if, for each $i \in \{1, 2, \ldots, i_0\}$, for the partition $w = u_0 \#_1 u_1 \#_2 \ldots \#_k u_k$, with $u_0, \ldots, u_k \in (\Sigma_1 \cup \ldots \cup \Sigma_i)^*$ and $\#_1, \ldots, \#_k \in \Sigma_{i+1} \cup \ldots \cup \Sigma_m$, each string $h_i(u_j)$ (the projection of u_j to Σ_i) is in $L(\mathcal{A})$ if i is odd, and in $L(\mathcal{A})^R$ if i is even.*

The proof is by induction on i. For every next i-th pass, with i odd, it is proved that $\mathcal{B}$ first simulates the computation of $\mathcal{A}$ on u_0; then, upon acceptance, on u_1; and so on until u_k. Each transition of $\mathcal{A}$ is simulated by a series of steps of $\mathcal{B}$: first implementing the original transition, and then skipping any intermediate symbols in $\Sigma_1 \cup \ldots \cup \Sigma_{i-1}$. A direct correspondence between these computations is established. The details of the proof are omitted due to space constraints.

It follows that $\mathcal{B}$ completes the last m-th pass if and only if the condition in Claim 1 is satisfied for all i.

With Claim 1 established, the language recognized by $\mathcal{B}$ can be described by the following formulae.

$$\begin{aligned}
L_0 &= \{\varepsilon\} \\
L_i &= \bigcup_{a_1 \ldots a_z \in L(\mathcal{A})} L_{i-1} a_1^{(i)} L_{i-1} a_2^{(i)} L_{i-1} \ldots a_z^{(i)} L_{i-1}, && \text{for odd } i > 1 \\
L_i &= \bigcup_{a_1 \ldots a_z \in L(\mathcal{A})} L_{i-1} a_z^{(i)} L_{i-1} a_{z-1}^{(i)} L_{i-1} \ldots a_1^{(i)} L_{i-1}, && \text{for even } i > 1 \\
L(\mathcal{B}) &= L_m
\end{aligned}$$

Claim 2. *A string w is accepted by $\mathcal{B}$ if and only if, for all $i \in \{0, 1, \ldots, m\}$ and for every substring $¢v\$$ of the tape $\vdash w \dashv$, with $¢ \in \{\vdash\} \cup \Sigma_{i+1} \cup \ldots \cup \Sigma_m$, $\$ \in \{\dashv\} \cup \Sigma_{i+1} \cup \ldots \cup \Sigma_m$ and $v \in (\Sigma_1 \cup \ldots \cup \Sigma_i)^*$, the string v is in L_i.*

First assume that $\mathcal{B}$ accepts w. The condition is proved by induction on i. The base case, $i = 0$, is trivial: the string v is defined over an empty alphabet and hence must be ε, and ε is in L_0.

For the induction step, let i be odd and let $¢v\$$ be a substring of the given form, with $v \in (\Sigma_1 \cup \ldots \cup \Sigma_i)^*$. Consider all occurrences of symbols from Σ_i in v, so that $v = v_0 a_1^{(i)} v_1 a_2^{(i)} v_2 \ldots a_z^{(i)} v_z$, with $v_0, \ldots, v_z \in (\Sigma_1 \cup \ldots \cup \Sigma_{i-1})^*$. By the induction hypothesis for each substring v_j, it belongs to L_{i-1}. On the other hand, since $\mathcal{B}$ completes its i-th pass, by Claim 1, the projection $h(v) = a_1 \ldots a_z$ is accepted by $\mathcal{A}$. This proves that v belongs to the language $L_{i-1} a_1^{(i)} L_{i-1} \ldots a_z^{(i)} L_{i-1}$ for some string $a_1 \ldots a_z \in L(\mathcal{A})$, confirming that v is in L_i. The proof for the case of even i is symmetric.

Conversely, let the condition in Claim 2 hold; the goal is to prove that the computation of $\mathcal{B}$ on w successfully completes all its m passes. For every subsequent i-th pass, let $w = u_0 \#_1 u_1 \#_2 \ldots \#_k u_k$, where $u_0, \ldots, u_k \in (\Sigma_1 \cup \ldots \cup \Sigma_i)^*$ and $\#_1, \ldots, \#_k \in \Sigma_{i+1} \cup \ldots \cup \Sigma_m$. The condition asserts that each string u_j is in L_i. Since the h_i-projection of every string in L_i is a string accepted by $\mathcal{A}$, it is known that $h_i(u_0), \ldots, h_i(u_k) \in L(\mathcal{A})$. Then, by Claim 1, $\mathcal{B}$ completes the i-th pass. The case of even i is again symmetric. This completes the proof of Claim 2.

Since the length of the shortest string in each L_i is $\ell^i - 1$, the shortest string accepted by $\mathcal{B}$ is of length $\ell^m - 1$, as claimed. □

Substituting the 5-state automaton from Example 1 into Lemma 3 yields the following lower bound.

Theorem 1. *For every $n \geqslant 1$, there is an n-state direction-determinate 2DFA over an alphabet of size $7\lceil \frac{n}{5} \rceil$, with the shortest accepted string of length $8^{\lfloor \frac{n}{5} \rfloor} - 1$.*

6 Small Alphabets

The constructions in Lemmata 2 and 3 depend on using an alphabet of linear size. As it is often observed in the state complexity research, the assumption that the alphabet grows with the number of states is not always realistic, and the case of a fixed alphabet at least deserves a separate investigation.

For a unary alphabet, the expressive power of two-way automata is known quite well [3,5,8], and the maximal length of a shortest string can be determined precisely.

The ability of 2DFAs to count in unary notation is described by the following function, known as *Landau's function* [10].

$$g(n) = \max\{\, \mathrm{lcm}(p_1, \ldots, p_k) \mid k \geqslant 1,\ p_1 + \ldots + p_k \leqslant n \,\} = e^{(1+o(1))\sqrt{n \ln n}}$$

The value $g(n)$ is known as the maximum order of an element in the group of permutations of n objects.

Theorem C. *For every $n \geqslant 2$, there is an n-state sweeping 2DFA recognizing the language $a^{g(n-1)-1}(a^{g(n-1)})^*$.*

Using this sweeping 2DFA as the base automaton in Lemma 3 leads to the following consequence.

Corollary 3. *Let Σ be a fixed m-symbol alphabet, with $m \geqslant 1$. Then, for every number $n \geqslant 1$, there exists an n-state 2DFA over the alphabet Σ, with the shortest accepted string of length $g\big(\lfloor \frac{n}{m} \rfloor - 1\big)^m - 1$, which is of the order $e^{(1+o(1))\sqrt{mn \ln \frac{n}{m}}}$.*

An interesting question is whether an exponential lower bound of the form $2^{\Omega(n)}$ can be obtained using a fixed alphabet.

7 On Improving the Estimation

The longest length of a shortest string in an n-state 2DFA is now known to be between 1.515^n and 4^n. The question is, what is the exact value?

An obvious way of improving the lower bound is to find a better base automaton for Lemma 3 than the one in Example 1. Any k-state direction-determinate automaton with the shortest accepting string of length $\ell - 1$ would improve over the existing construction if $\sqrt[k]{\ell} > \sqrt[5]{8}$. On the other hand, the method of Lemma 3 might have its limitations, and some entirely new methods might yield better lower bounds.

Turning to the upper bounds, perhaps the bounds in Corollaries 1 and 2 could be improved by analyzing the constructions in the corresponding Theorems A and B. One could also try improving the upper bound for small alphabets by investigating two-way transformation semigroups of Kunc and Okhotin [8].

References

1. Alpoge, L., Ang, T., Schaeffer, L., Shallit, J.: Decidability and shortest strings in formal languages. In: Holzer, M., Kutrib, M., Pighizzini, G. (eds.) DCFS 2011. LNCS, vol. 6808, pp. 55–67. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22600-7_5
2. Chistikov, D., Czerwinski, W., Hofman, P., Pilipczuk, M., Wehar, M.: Shortest paths in one-counter systems. In: Jacobs, B., Löding, C. (eds.) FoSSaCS 2016. LNCS, vol. 9634. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49630-5_27
3. Chrobak, M.: Finite automata and unary languages. Theor. Comput. Sci. **47**, pp. 149–158 (1986). https://doi.org/10.1016/0304-3975(86)90142-8. Errata: vol. 302, pp. 497–498 (2003).https://doi.org/10.1016/S0304-3975(03)00136-1
4. Ellul, K., Krawetz, B., Shallit, J., Wang, M.-W.: Regular expressions: new results and open problems. J. Autom. Lang. Comb. **10**(4), 407–437 (2005). https://doi.org/10.25596/jalc-2005-407
5. Geffert, V., Mereghetti, C., Pighizzini, G.: Converting two-way nondeterministic unary automata into simpler automata. Theor. Comput. Sci. **295**(1–3), 189–203 (2003). https://doi.org/10.1016/S0304-3975(02)00403-6
6. Geffert, V., Okhotin, A.: One-way simulation of two-way finite automata over small alphabets. In: NCMA 2013, Umeå, Sweden, 13–14 August 2013
7. Kapoutsis, C.A.: Removing bidirectionality from nondeterministic finite automata. In: Jędrzejowicz, J., Szepietowski, A. (eds.) MFCS 2005. LNCS, vol. 3618. Springer, Heidelberg (2005). https://doi.org/10.1007/11549345_47
8. Kunc, M., Okhotin, A.: Describing periodicity in two-way deterministic finite automata using transformation semigroups. In: Mauri, G., Leporati, A. (eds.) DLT 2011. LNCS, vol. 6795. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22321-1_28
9. Kunc, M., Okhotin, A.: Reversibility of computations in graph-walking automata. In: Chatterjee, K., Sgall, J. (eds.) MFCS 2013. LNCS, vol. 8087, pp. 595–606. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40313-2_53
10. Landau, E.: Über die Maximalordnung der Permutationen gegebenen Grades (On the maximal order of permutations of a given degree). Archiv der Mathematik und Physik, Ser. **3**(5), 92–103 (1903)
11. Pierre, L.: Rational indexes of generators of the cone of context-free languages. Theor. Comput. Sci. **95**(2), 279–305 (1992). https://doi.org/10.1016/0304-3975(92)90269-L
12. Sipser, M.: Lower bounds on the size of sweeping automata. In: STOC, pp. 360–364 (1979). https://doi.org/10.1145/800135.804429