Theoretical Computer Science 449 (2012) 106–118

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

State complexity of operations on two-way finite automata over a unary alphabet☆

Michal Kunc [a], Alexander Okhotin [b,*]

[a] Department of Mathematics, Masaryk University, Brno, Czech Republic

[b] Department of Mathematics, University of Turku, Turku FI–20014, Finland

ARTICLE INFO

Keywords:
Finite automata
Two-way automata
Regular languages
Unary languages
State complexity
Landau's function

ABSTRACT

The paper determines the number of states in two-way deterministic finite automata (2DFA) over a one-letter alphabet sufficient and in the worst case necessary to represent the results of basic language-theoretic operations on 2DFAs with a certain number of states. It is proved that (i) intersection of an m-state 2DFA and an n-state 2DFA requires between $m+n$ and $m+n+1$ states; (ii) union of an m-state 2DFA and an n-state 2DFA, between $m+n$ and $2m+n+4$ states; (iii) Kleene star of an n-state 2DFA, $(g(n)+O(n))^2$ states, where $g(n) = e^{(1+o(1))\sqrt{n \ln n}}$ is the maximum value of $\operatorname{lcm}(p_1, \ldots, p_k)$ for $\sum p_i \leqslant n$, known as Landau's function; (iv) k-th power of an n-state 2DFA, between $(k-1)g(n)-k$ and $k(g(n)+n)$ states; (v) concatenation of an m-state 2DFA and an n-state 2DFA, $e^{(1+o(1))\sqrt{(m+n)\ln(m+n)}}$ states. It is furthermore demonstrated that the Kleene star of a two-way nondeterministic automaton (2NFA) with n states requires $\Theta(g(n))$ states in the worst case, its k-th power requires $(k \cdot g(n))^{\Theta(1)}$ states, and the concatenation of an m-state 2NFA and an n-state 2NFA, $e^{\Theta(\sqrt{(m+n)\ln(m+n)})}$ states.

1. Introduction

The study of the number of states in one-way deterministic finite automata (1DFA) needed to represent the results of basic operations on regular languages dates back to Maslov [14], who showed that a concatenation of an m-state 1DFA with an n-state 1DFA is representable by a 1DFA with $m \cdot 2^n - 2^{n-1}$ states, and that this number of states is necessary in the worst case. Similarly, the Kleene star of an n-state 1DFA requires up to $\frac{3}{4}2^n$ states, etc.; this function is known as the *state complexity* of an operation. For the case of a one-letter alphabet $\Sigma = \{a\}$, Yu et al. [21] showed that the complexity of most operations is significantly different from the case of larger alphabets: for instance, the Kleene star requires $(n-1)^2+1$ states. For one-way nondeterministic automata (1NFA), both over unary and larger alphabets, the complexity of all basic operations was determined by Holzer and Kutrib [6]. The first results of this kind for the intermediate family of unambiguous finite automata (1UFA) were recently obtained by Okhotin [17], who showed that complementation of a unary 1UFA requires at least $n^{2-o(1)}$ states and at most $e^{O(\sqrt[3]{n \ln^2 n})}$ states.

State complexity of operations on two-way automata (2DFA and 2NFA) has so far attracted much less attention, probably due to the lack of any general lower bound methods. An attempt to study the complexity of operations on 2DFAs using the constructions and the lower bound techniques developed for 1DFAs was made by Jirásková and Okhotin [7], and, for most

☆ A preliminary version of this paper was presented at the workshop on Descriptional Complexity of Formal Systems (DCFS 2011, Limburg, Germany, 25–27 July 2011), and its extended abstract appeared in the workshop proceedings.

* Corresponding author. Tel.: +358 2 333 5611; fax: +358 2 333 6595.
E-mail addresses: kunc@math.muni.cz (M. Kunc), alexander.okhotin@utu.fi (A. Okhotin).

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.04.010

operations, led to very rough estimations, such as a $\frac{1}{n}2^{\frac{n}{2}-1}$ lower bound and a $2^{O(n^{n+1})}$ upper bound on the state complexity of the Kleene star. However, a few precise results were obtained as well: for instance, the $2n$ state complexity of inverse homomorphisms. These results essentially relied upon an alphabet of exponential size.

This paper is aimed at determining the state complexity of basic operations on 2DFAs in the special case of a one-letter alphabet. The study of unary 2DFAs began with a paper by Chrobak [1], who showed that the language recognized by an n-state 2DFA is ultimately periodic with period $\mathrm{lcm}(p_1, \ldots, p_k)$, for some numbers $p_1, \ldots, p_k \geqslant 1$ with $p_1 + \cdots + p_k \leqslant n$. The authors' [10] recent analysis of 2DFAs over a one-letter alphabet refined this understanding with a precise estimation of the starting point of periodicity, known as the *length of the tail* of the minimal 1DFA, or the *index* of the corresponding monoid: a unary language recognized by an n-state 2DFA has period $\mathrm{lcm}(p_1, \ldots, p_k)$ beginning from ℓ, for some numbers $p_1, \ldots, p_k, \ell \geqslant 1$ with $p_1 + \cdots + p_k + \ell \leqslant n + 1$ [10, Cor. 2]. The greatest value of the least common multiple of partitions of a number n is known as *Landau's function* $g(n)$ and estimated as $g(n) = e^{\sqrt{n \ln n}(1+o(1))}$ [13], and the exact number of states in a 1DFA needed to simulate an n-state unary 2DFA is accordingly expressed as $\max_{1 \leqslant \ell \leqslant n} g(n + 1 - \ell) + \ell$.

Another model considered in this paper is the *two-way nondeterministic finite automata* (2NFA). For a general alphabet, Kapoutsis [8] determined that transforming n-state 2NFAs to 1NFAs requires exactly $\binom{2n}{n+1}$ states in the worst case, while 1DFAs equivalent to n-state 2NFAs require up to exactly $\sum_{i,j \in \{0,\ldots,n-1\}} \binom{n}{i}\binom{n}{j}(2^i-1)^j$ states. In the domain of unary languages, Chrobak's [1] investigation of 2DFAs was continued for their nondeterministic case by Mereghetti and Pighizzini [15], who found that n-state unary 2NFAs require, in the worst case, $g(n) + O(n^2)$ states in equivalent 1DFAs, and by Geffert et al. [4], who established an $n^{O(\log n)}$-state upper bound on the transformation of unary 2NFAs to equivalent 2DFAs. The first result on the state complexity of operations on this model is due to Geffert et al. [5], who sketched a construction of a 2NFA with $O(n^8)$ states for the complement of a given n-state unary 2NFA, which implemented the Immerman–Szelepcsényi inductive counting. For union and intersection, the authors [11] have shown that the union of an m-state 2NFA and an n-state 2NFA requires exactly $m + n$ states in the worst case, while intersection requires between $m + n$ and $m + n + 1$ states[1]; the lower bound proofs used cyclic witness languages, and relied on a number-theoretic lemma with a large computer-generated basis of induction.

This paper begins with investigating the complexity of Boolean operations for 2DFAs. As shown in Section 3, for the *intersection*, the obvious $(m+n+1)$-state upper bound is matched by an almost tight lower bound of $m+n$ states, obtained by intersecting a language with a long period and no tail, with a language with period 2 and a long tail. The *union* of two 2DFAs can be directly represented, as long as one of them halts on every input. The latter condition can be ensured by the authors' [12] upcoming result that every n-state unary 2DFA can be transformed to a *reversible* 2DFA with $2n + 3$ states. This leads to a $(2m + n + 3)$-state upper bound for the union. A lower bound of $m + n$ states is proved similarly to the case of intersection. Admittedly, the lower bounds on the union and intersection of 2DFAs presented in this paper overlap with the authors' [11] recent results on the union and intersection of 2NFAs. However, the results in this paper are established in a different way, specific for 2DFAs; there are far fewer exceptions in their statements, and the proofs contain no machine-generated components.

The next group of results is concerned with concatenation and the derived operations: concatenation closure (Kleene star) and the operation of concatenating k copies of a language: the k-th power. The effect of concatenation and star on unary 1DFAs was first studied by Yu et al. [21], who showed that the worst-case complexity is achieved when the arguments are cyclic languages and the result is a co-finite language; their method was applied for the power operation by Rampersad [19]. Since 2DFAs for co-finite unary languages are as large as 1DFAs, this method can be extended to showing the state complexity of operations on 2DFAs. For the *Kleene star*, discussed in Section 4, the resulting complexity is $(g(n) + O(n))^2$; the next Section 5 shows that the *k-th power* operation has state complexity between $(k-1)g(n) - k$ and $kg(n) + O(kn)$. For the case of *concatenation* of two unary languages given by 2DFAs, its worst-case complexity is explained in terms of a variant of Landau's function, $g(m, n)$, which is in turn found to have growth rate $g(m, n) = e^{\sqrt{(m+n)\ln(m+n)}(1+o(1))}$.

The last subject of this paper, investigated in Section 7, is the state complexity of concatenation, Kleene star and the k-th power for unary 2NFAs. The general idea is to remake the arguments for 2DFAs, presented in Sections 4–6, for the nondeterministic case. This time, the proofs rely on the transformation of an n-state unary 2NFA to a 1DFA with $g(n)+O(n^2)$ states, due to Mereghetti and Pighizzini [15], and use the representation of languages by 1NFAs to apply concatenation and star without increasing the number of states. The differences in the underlying methods lead to estimations of complexity that are not exactly the same as in the deterministic case: the star of an n-state 2NFA is found to require $\Theta(g(n))$ states, the concatenation requires between $\Omega(\sqrt{g(m, n)})$ and $g(m)+g(n)+O((m+n)^2)$ states, and the k-th power needs $(k \cdot g(n))^{\Theta(1)}$ states.

2. Unary 2DFAs and their expressive power

A 1DFA over a unary alphabet is just a directed graph of out-degree 1. It has a unique path from the initial state, which eventually converges to a *cycle*. Zero or more states visited before entering the cycle are called the *tail* of the 1DFA.

Computations of 2DFAs are more complicated and more challenging to understand, even in the unary case. Given an input string w, a 2DFA operates on a tape containing the string $\vdash w \dashv$, where $\vdash$ and $\dashv$ are special symbols known as the

[1] With some exceptions for small values of m or n, where the known lower bound is $m + n - 1$ or $m + n - 2$.

left-end marker and the right-end marker, respectively. According to the standard definition, a 2DFA begins its computation at the left-end marker and accepts at the right-end marker [8]. In this paper, as well as in the authors' recent work [10], the definition is extended to allow acceptance on both sides: this leads to symmetric constructions and allows avoiding some awkward exceptions in the results. Changing the mode of acceptance affects the size of an automaton by at most one state.

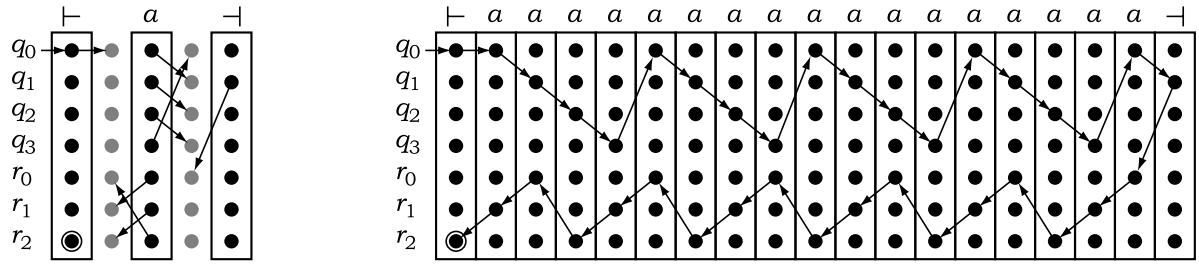
A *2DFA (with two-sided acceptance)* is defined as a sextuple $\mathcal{A} = (\Sigma, Q, q_0, \delta, F_{\vdash}, F_{\dashv})$, in which:

- Σ is a finite alphabet with $\vdash, \dashv \notin \Sigma$,
- Q is a finite set of states,
- $q_0 \in Q$ is the initial state,
- $\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \to Q \times \{-1, +1\}$ is a partially defined transition function, and
- $F_{\vdash}, F_{\dashv} \subseteq Q$ are sets of states accepting on the left-end marker $\vdash$ and on the right-end marker $\dashv$, respectively.

When $\mathcal{A}$ is in a state $q \in Q$ and observes a square of the tape with a symbol $a \in \Sigma \cup \{\vdash, \dashv\}$, the value $\delta(q, a)$ indicates the next state and the direction of motion. The computation of $\mathcal{A}$ on a string $\vdash w \dashv = \vdash a_1 \dots a_\ell \dashv$, with $\ell \geqslant 0$ and $a_1, \dots, a_\ell \in \Sigma$, begins in state q_0, with the head observing $\vdash$. If it eventually reaches an accepting state in $F_{\vdash}$ or in $F_{\dashv}$ while on the corresponding end-marker, the string w is accepted; otherwise, it either encounters an undefined transition or gets into an infinite loop. The set of strings accepted by a 2DFA $\mathcal{A}$ is denoted by $L(\mathcal{A})$.

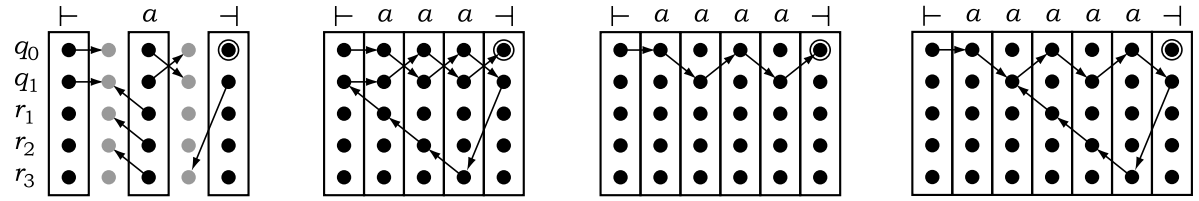
Every 2DFA can be transformed to an equivalent 1DFA, which is often much larger than the original 2DFA. In the case of a one-letter alphabet $\Sigma = \{a\}$, there are essentially two possibilities for a 2DFA to provide a more succinct description than a 1DFA. First, a 2DFA can count divisibility separately for several numbers, while an equivalent 1DFA would need to count modulo the least common multiple of those numbers; this is illustrated in the following example.

Example 1. Let $p_1, \dots, p_k \geqslant 2$ be any integers and denote $p = \operatorname{lcm}(p_1, \dots, p_k)$. Take an arbitrary $f \in \{0, \dots, p-1\}$. Then the language $L = a^f (a^p)^*$ is recognized by a $(p_1 + \cdots + p_k)$-state 2DFA, while the minimal 1DFA for L has p states. For $k = 2$, $p_1 = 4$, $p_2 = 3$ and $f = 5$, this gives a 7-state 2DFA recognizing the language $a^5(a^{12})^*$, which has the set of states $Q = \{q_0, q_1, q_2, q_3, r_0, r_1, r_2\}$, transitions $\delta(q_0, \vdash) = (q_0, +1)$, $\delta(q_i, a) = (q_{(i+1) \bmod 4}, +1)$, $\delta(r_i, a) = (r_{(i+1) \bmod 3}, -1)$, $\delta(q_1, \dashv) = (r_0, -1)$, and $F_{\vdash} = \{r_2\}$, $F_{\dashv} = \varnothing$.



The second advantage of a 2DFA over a 1DFA in terms of succinctness of description is that a 2DFA can recognize a language with tail of length $\ell + 1$ and period p by counting up to ℓ, and then using one of its cycles to distinguish the string a^ℓ, which is accepted in one of the states of the cycle, from longer strings $a^{\ell+ip}$ with $i \geqslant 1$, which are rejected by infinitely looping using this cycle.

Example 2 (*[10]*)**.** Let $p \geqslant 2, f \in \{0, \dots, p-1\}$ and $\ell \geqslant 0$ be any numbers with $\ell \not\equiv f \pmod{p}$. Then the language $L = a^f (a^p)^* \cup \{a^\ell\}$ is recognized by a $(p + \ell)$-state 2DFA with acceptance only on the right-end marker, while the minimal 1DFA recognizing L has $p + \ell + 1$ states. A 2DFA recognizing such a language L with $p = 2, f = 0$ and $\ell = 3$ is illustrated below:



All strings of even length are accepted after one left-to-right traversal in the states $\{q_0, q_1\}$; if the string is found to be of odd length, the automaton then traverses it from right to left, with a countdown using the states r_3, r_2, r_1. Note how the string a^3 is accepted, because the left-end marker is reached at the right moment, and the string a^5 is rejected by re-entering the cycle.

A unary 1DFA is defined by the structure of its transitions, which is completely described by the length of its tail and its cycle, and by the combination of accepting states over this structure. The following result recently established by the authors [10] asserts that for a 1DFA equivalent to an n-state unary 2DFA, its structure cannot be of any other form than a long cycle, obtained as in Example 1, preceded by a tail, as in Example 2.

Theorem A ([10]). *Let $\mathcal{A}$ be an n-state 2DFA over $\Sigma = \{a\}$. Then there exist $k \geqslant 1$ and numbers $p_1, \ldots, p_k \geqslant 1$ and $\ell \geqslant 1$, with $p_1 + \cdots + p_k + \ell \leqslant n + 1$, such that there exists a 1DFA for $L(\mathcal{A})$ with tail of length ℓ and period $p = \operatorname{lcm}(p_1, \ldots, p_k)$.*

This result can also be presented in the following equivalent form.

Theorem B ([10]). *Let $L \subseteq a^*$ be a regular language with the minimal 1DFA with tail of length ℓ and period p. Let $p = p_1 \cdots p_k$, where $p_1, \ldots, p_k$ are powers of distinct primes, be the prime factorization of p (for $p = 1$, assume that $1 = 1$ is a prime factorization). Then, every 2DFA recognizing L must have at least $p_1 + \cdots + p_k + \max(\ell, 1) - 1$ states.*

In particular, this result implies that a language with a long tail requires a 2DFA with as many states as the length of the tail. This will be used in Sections 4–6 to prove lower bounds using co-finite languages.

How large could a least common multiple of numbers $p_1, \ldots, p_k$ be, if the sum of these numbers is bounded by n? As a function of n, this number

$$g(n) = \max\{\operatorname{lcm}(p_1, \ldots, p_k) \mid k \geqslant 1,\ p_1 + \cdots + p_k \leqslant n\}$$

is known as *Landau's function*, as its $e^{(1+o(1))\sqrt{n \ln n}}$ asymptotics was determined by Landau [13] (see also Miller [16] for a more accessible argument). For example, the value $g(12) = 60$ is attained for $k = 3$, $p_1 = 3$, $p_2 = 4$ and $p_3 = 5$. It is worth noting, that the numbers $p_1, \ldots, p_k$, on which the maximum is reached, can be assumed to be powers of distinct primes.[2]

For any positive integer n, the number $g(n)$ is the longest period of a language recognized by an n-state unary 2DFA. In order to take the length of the tail in the corresponding minimal 1DFAs into account, consider the following simple variant of Landau's function,[3] which adds the unused portion of n to the resulting value:

$$g^+(n) = \max_{0 \leqslant \ell < n} g(n - \ell) + \ell = \max\{\operatorname{lcm}(p_1, \ldots, p_k) + \ell \mid k \geqslant 1,\ p_1 + \cdots + p_k + \ell = n\}.$$

For example, $g^+(12) = 60$ and $g^+(13) = 61$, because $g(12) = g(13) = 60$ and $g(11) = 30 \ll 60$. In terms of this function, the 2DFA–1DFA tradeoff is expressed as follows.

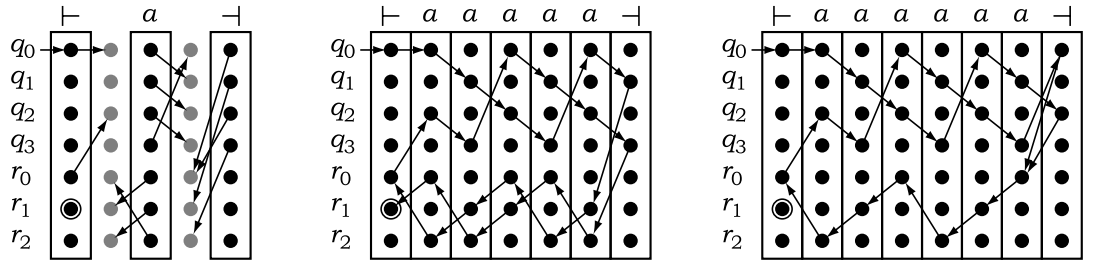
Theorem C ([10]). *Let $n \geqslant 1$. Then, for every unary 2DFA with n states, there exists an equivalent complete 1DFA with $g^+(n) + 1$ states. For $n \geqslant 3$, this number of states is required already for the transformation of 2DFAs to 1NFAs.*

Another related result on the succinctness tradeoffs between deterministic automata over a unary alphabet concerns the *sweeping 2DFAs* [20], in which the head may change its direction of motion only on the end-markers. Chrobak [1] has claimed without a proof that every n-state 2DFA can be transformed to an equivalent sweeping 2DFA with n states. This conjecture was in fact almost correct.

Theorem D ([10]). *Let $n \geqslant 1$. Then for every unary 2DFA with n states, there exists an equivalent sweeping 2DFA with $n + 1$ states. For $n \geqslant 2$, this bound is the best possible.*

All existing knowledge on the capabilities and limitations of unary 2DFAs is, unfortunately, limited to the bounds on the periods and the tails of the languages they recognize, as listed in Theorems A and B. Once a regular language $L \subseteq a^*$ has period $\operatorname{lcm}(p_1, \ldots, p_k)$ and a tail of length ℓ, there are no known methods to determine whether there is a 2DFA with a given number of states $n \geqslant p_1 + \ldots + p_k + \ell - 1$ recognizing this particular language, and if it exists, there are no general methods of constructing it. Consider the following example.

Example 3. The language $L = \{a^4, a^5, a^9, a^{10}, a^{11}\}(a^{12})^*$ is recognized by a 7-state 2DFA with the following transitions:



In particular, this 2DFA accepts the string a^5 after four traversals, and rejects the string a^6 by looping.

As shown by exhaustive calculations, the given automaton is the unique 7-state 2DFA for L, up to permutation of states. Furthermore, the complement of L has no 7-state 2DFA.

2 If two numbers are not relatively prime, one can divide one of them by their common divisor, and once all numbers are relatively prime, each product of powers of different primes can be split into their sum. All these transformations reduce the sum of the numbers, while maintaining the same greatest common multiple $g(n)$.

3 OEIS sequence A039952.

The 2DFA in Example 3 is given to illustrate the sophisticated combinatorics that may occur in the operation of unary 2DFAs. Why does this language have a 7-state 2DFA, while its complement does not have one? The lack of methods for answering this question illustrates the general absence of methods for proving precise state complexity of languages, beyond those given in Theorems A and B.

It should be noted that in this small 7-state example, the computations are still simple enough. Each computation alternates between counting modulo 4 on left-to-right traversals, and modulo 3 on the traversals from right to left, and one can observe the operation of this 2DFA by considering all residues modulo 12. However, tracing the computations of a given automaton does not explain its existence. Even though the number 12 might be small enough to devise a brute-force combinatorial proof that the complement of this language requires more than 7 states, such an approach would be inadequate for automata that count modulo larger numbers and contain multiple cycles, used in various order in different computations. Proving any general characterization of languages, which can be represented by some 2DFA of a given size, is way beyond the current knowledge.

3. Intersection and union

The intersection of an m-state and an n-state 1DFAs can be represented by a 1DFA with mn states [14]. This number of states is necessary already for a unary alphabet, as long as m and n are relatively prime, because, under this assumption, $(a^m)^* \cap (a^n)^* = (a^{mn})^*$. When m and n are not relatively prime, intersection of unary 1DFAs sometimes has a lower state complexity, witnessed by languages of a different form [18].

For 2DFAs, intersection can always be represented with $m + n + 1$ states, and the only known lower bound of $m + n - o(m + n)$ states was established using growing alphabets of enormous size [7]. This paper improves the lower bound to $m + n$ using only unary alphabet. The main idea is to choose one of the languages being intersected to be of the form presented in Example 1, with a period obtained as a prime factorization and with no tail, and let the other 2DFA be of the form as in Example 2, using period 2 and a tail as long as possible. If everything works as planned, the intersection should have both a long period and a long tail.

The period of the first 2DFA will be obtained using the following result:

Theorem E (*Dressler [3]*)**.** *Every number greater than* 9 *is representable as a sum of distinct odd primes.*

Since the constructions of 2DFAs as in Example 1 work for any *powers of primes*, one can relax the assumptions of Theorem E and extend its statement to smaller numbers as follows.

Corollary E.1. *Every number $n \geqslant 3$ with $n \notin \{4, 6\}$ can be represented as a sum of powers of distinct odd primes.*

Proof. If $n \geqslant 10$, such a partition is given by Theorem E. It remains to construct partitions of smaller numbers. Every number $n \in \{3, 5, 7\}$ is an odd prime itself, and is trivially representable as itself. The number $n = 8$ is represented as $3 + 5$, and $n = 9$, as 3^2. □

The almost tight bounds on the complexity of intersection are now established for all values of m and n greater than one, with only two exceptions.

Theorem 1. *For every $m, n \geqslant 2$ with $(m, n) \notin \{(2, 2), (6, 6)\}$, the state complexity of intersection of an m-state 2DFA and an n-state 2DFA over a unary alphabet is at least $m + n$. It is at most $m + n + 1$ for all $m, n \geqslant 1$, for a general alphabet.*

Proof. The construction of an $(m + n + 1)$-state 2DFA for the intersection is straightforward [7]. Given two 2DFAs $\mathcal{A}$ and $\mathcal{B}$ with the sets of states Q and R, respectively, the new 2DFA $\mathcal{C}$ has the set of states $Q \cup \{q_{\leftarrow}\} \cup R$. It begins its computation by simulating $\mathcal{A}$, and if $\mathcal{A}$ accepts on the left-end marker, it proceeds with simulating $\mathcal{B}$, until it accepts or rejects. Whenever $\mathcal{A}$ accepts on the right-end marker, $\mathcal{C}$ gets back to the left-end marker in the special state $q_{\leftarrow}$, and then proceeds with simulating $\mathcal{B}$.

Note that this special state is necessary only if $\mathcal{A}$ may accept on both markers. If all accepting states of $\mathcal{A}$ are on one of the markers, then the simulation of $\mathcal{B}$ may begin directly in an accepting state of $\mathcal{A}$ (when the marker on which $\mathcal{A}$ accepts is the right-end marker $\dashv$, the automaton $\mathcal{B}$ is simulated with all transitions reversed). Hence, $\mathcal{C}$ may use only $m + n$ states in this case.

Turning to the lower bound argument, first assume that the numbers m and n are not both in $\{2, 4, 6\}$. Without loss of generality, let $n \notin \{2, 4, 6\}$, and let $m \geqslant 2$ be any number. Then, by Corollary E.1, there is a partition $n = p_1 + \cdots + p_k$, where $p_1, \ldots, p_k \geqslant 3$ are powers of pairwise distinct odd primes. Consider the languages

$$K = \{a^\ell \mid \ell \equiv m - 1 \pmod 2 \text{ or } \ell = m - 2\} = a^{(m-1) \bmod 2}(a^2)^* \cup \{a^{m-2}\}$$

and

$$L = \{a^\ell \mid \ell \equiv m - 2 \pmod{p_1 \cdots p_k}\} = a^{(m-2) \bmod p_1 \cdots p_k}(a^{p_1 \cdots p_k})^*,$$

which are recognized by an m-state 2DFA as in Example 2, and by an n-state 2DFA from Example 1, respectively. Their intersection is

$$K \cap L = a^j(a^{2 \cdot p_1 \cdots p_k})^* \cup \{a^{m-2}\},$$

where $j \in \{0, 1, \ldots, 2 \cdot p_1 \cdots p_k - 1\}$ is the unique offset with $j \equiv m-1 \pmod 2$ and $j \equiv m-2 \pmod{p_1 \cdots p_k}$, which exists by the Chinese Remainder Theorem. The minimal 1DFA for this language has tail of length $m-1$ and period $p = 2 \cdot p_1 \cdots p_k$. Since the latter is the prime factorization of p, Theorem B is applicable, and it asserts that every 2DFA recognizing this intersection must have at least $2 + p_1 + \cdots + p_k + \max(m-1, 1) - 1 = m + n$ states.

Consider the remaining cases of $(m, n) \in \{(4, 2), (4, 4), (6, 2), (6, 4)\}$, and let the two languages be

$$K = \{a^\ell \mid \ell \equiv m+1 \pmod 3 \text{ or } \ell = m-3\} = a^{(m+1) \bmod 3}(a^3)^* \cup \{a^{m-3}\}$$

and

$$L = \{a^\ell \mid \ell \equiv m-3 \pmod n\} = a^{(m-3) \bmod n}(a^n)^*,$$

with an m-state and an n-state 2DFA as in Example 2 (this time with the cycle of length 3) and Example 1, respectively. The intersection of these languages is

$$K \cap L = \{a^\ell \mid \ell \equiv m+1 \pmod 3 \text{ and } \ell \equiv m-3 \pmod n\} \cup \{a^{m-3}\},$$

The minimal 1DFA for this language has tail of length $m-2$ and period $p = 3 \cdot n$. Since the latter is a prime factorization of p, by Theorem B, every 2DFA recognizing $K \cap L$ has at least $3 + n + \max(m-2, 1) - 1 = m + n$ states. □

Concerning the two exceptions in the statement of Theorem 1, an exhaustive calculation of all 2DFAs over a one-letter alphabet with up to 4 states indicates that an intersection of two 2-state unary 2DFAs is always representable by a 2DFA with 4 states. Furthermore, this bound is reached on the witness languages $L_1 = a(aa)^* \cup \{\varepsilon\}$ and $L_2 = aaa^* \cup \{\varepsilon\}$, each recognized by a 2-state 2DFA, and whose intersection $aaa(aa)^* \cup \{\varepsilon\}$ requires a 4-state 2DFA. As for the case of two 6-state 2DFAs, since the number 6 is not representable as a sum of powers of distinct primes, proving a lower bound of 12 states by the same methods as in Theorem 1 is impossible. All that can be said about the intersection of two 6-state automata is that it requires between 11 and 13 states, where the lower bound follows from Theorem 1 with $m = 6$ and $n = 5$.

Theorem 1 leaves a natural question of whether the actual complexity of intersection is $m+n$ or $m+n+1$. As mentioned in the proof, if one of the 2DFAs to be intersected has all accepting states on the same side, then the state $q_\leftarrow$ is not needed, and the intersection can be recognized with $m+n$ states. Is this extra state ever needed? First of all, there is no known proof that for some language, every 2DFA with a given number of states requires acceptance on both sides. Secondly, even if such proofs existed, it would still be possible that the intersection of two languages with essentially two-sided acceptance could always be represented without adding an extra state, and any counterexample would require another nontrivial proof. The lack of methods for proving statements of this kind has been explained in the comments to Example 3.

Turning to the **union** operation, representing a union of two 2DFAs is straightforward, if one of them halts on every input. Geffert et al. [5] showed that any n-state 2DFA can be transformed to an equivalent $4n$-state 2DFA that halts on every input; since Geffert et al. [5] used a slightly different definition of acceptance in 2DFAs than in this paper, their construction produces $4n + \text{const}$ states under the definition assumed in the present paper. From this fact, Jirásková and Okhotin [7] inferred a $(4m + n + \text{const})$-state upper bound on the complexity of union for 2DFAs, which was accompanied by a lower bound of $m + n - o(m + n)$ states.

For the unary case, there is a stronger result on transforming 2DFAs to *reversible 2DFAs* [9,12], which, in particular, halt on every input. A 2DFA $(\Sigma, Q, q_0, \delta, F_\vdash, F_\dashv)$ is called *reversible*, if

1. for every state $q \in Q$, all transitions leading to q move the head in the same direction $d(q) \in \{-1, +1\}$, with $d(q_0) = +1$ for the initial state, and
2. the transitions from any two different states by the same symbol cannot lead to a single state, that is, whenever $\delta(p, a) = \delta(p', a) = (q, d(q))$, the states p and p' must be the same.

Kondacs and Watrous [9] proved that every 1DFA can be simulated by a reversible 2DFA with twice as many states. Their ideas led to the following reversibility construction for 2DFAs over a unary alphabet.

Theorem F (*Kunc and Okhotin [12]*)**.** *For every n-state 2DFA over a unary alphabet, there exists an equivalent reversible 2DFA with $2n + 3$ states. Every computation of this automaton terminates on the left-end marker, either in a unique accepting state, or in a unique rejecting state. Conversely, for every n, there exists a unary language recognized by an n-state 2DFA, for which every reversible 2DFA requires at least $2n - 2$ states.*

The proof proceeds by first transforming a given n-state 2DFA to an equivalent sweeping 2DFA with $n+1$ states, according to Theorem D, and then applying a new reversibility construction for sweeping 2DFAs, which develops the idea of the construction by Kondacs and Watrous [9] for transforming a 1DFA to a reversible 2DFA, and turns any m-state sweeping 2DFA into a $(2m + 1)$-state reversible 2DFA (at the expense of losing the sweeping property). The combined number of states is $2n + 3$.

Since every reversible 2DFA halts on every input, it can be complemented by exchanging its accepting and rejecting states. Thus, the first part of Theorem F directly implies that the complexity of complementing unary 2DFAs is at most $2n + 3$.

Besides giving an upper bound for the complementation, the reversibility construction in Theorem F also yields an improved upper bound on the state complexity of union for unary 2DFAs, mentioned in the next theorem.

Theorem 2. *For every $m, n \geqslant 2$ with $(m, n) \notin \{(2, 2), (6, 6)\}$, the state complexity of union of an m-state 2DFA and an n-state 2DFA over a unary alphabet is at least $m + n$. For all $m, n \geqslant 1$, it is at most $2m + n + 3$.*

Proof. For the upper bound, consider a pair of unary 2DFAs $\mathcal{A}$ and $\mathcal{B}$, with m and n states, respectively. The new automaton recognizing $L(\mathcal{A}) \cup L(\mathcal{B})$ begins its computation by simulating a $(2m+3)$-state reversible 2DFA $\mathcal{A}'$ recognizing $L(\mathcal{A})$, which exists by Theorem F. If the simulated $\mathcal{A}'$ ends its computation in an accepting configuration, the new automaton accepts. If $\mathcal{A}'$ rejects, it always does so in a configuration known beforehand, from which the new 2DFA proceeds with simulating $\mathcal{B}$, accepting whenever $\mathcal{B}$ accepts.

The lower bound is proved similarly to the proof of Theorem 1. Consider first the case of $n \notin \{2, 4, 6\}$ and arbitrary $m \geqslant 2$, and, according to Corollary E.1, represent n as $n = p_1 + \cdots + p_k$, where $p_1, \ldots, p_k \geqslant 3$ are powers of pairwise distinct odd primes. Consider the language $K = a^{(m-1) \bmod 2}(a^2)^* \cup \{a^{m-2}\}$, recognized by an m-state 2DFA as in Example 2, and another language $L = \{a^\ell \mid \ell \not\equiv m - 2 \pmod{p_1 \cdots p_k}\}$, representable by an n-state 2DFA; note that L is the complement of the language from Example 1, used in the proof of Theorem 1. Their union is

$$K \cup L = \{a^\ell \mid \ell \not\equiv m - 2 \pmod{p_1 \cdots p_k} \text{ or } \ell \not\equiv m \pmod 2\} \cup \{a^{m-2}\}.$$

The minimal 1DFA for this language has tail of length $m - 1$ and period $p = 2 \cdot p_1 \cdots p_k$. Since the latter is a prime factorization of p, Theorem B is applicable, and it asserts that every 2DFA recognizing this union must have at least $2 + p_1 + \cdots + p_k + \max(m - 1, 1) - 1 = m + n$ states.

The remaining cases of $(m, n) \in \{(4, 2),\ (4, 4),\ (6, 2),\ (6, 4)\}$ can be handled in the very same way as for intersection, again using the complement of L. □

Consider the cases $m = n = 2$ and $m = n = 6$ not covered by Theorem 2. According to the calculations, the union of any two 2-state unary 2DFAs is representable by a 2DFA with 3 states, which is necessary in the worst case. For the union of two 6-state 2DFAs, a lower bound of 11 states is given by Theorem 2 for $m = 6$ and $n = 5$.

The gap between the lower bound $m + n$ and the upper bound $2m + n + 3$ is still wide, and narrowing it requires further studies. This gap originates from the lack of knowledge on the exact complexity of transforming a unary 2DFA to an equivalent 2DFA that always halts: this can be done by making the 2DFA reversible, according to Theorem F, and thus using $2n + 3$ states. Though Theorem F provides a $(2n - 2)$-state lower bound on the complexity of transforming a unary 2DFA to a reversible 2DFA, it does not apply to the rest of the constructions. Even if a 2DFA for a language L requires doubling its number of states to recognize L reversibly, this does not rule out having another small irreversible 2DFA for L that halts on every input. Even if there were a proof that 2DFAs for K and L both require doubling (or at least increasing) their number of states to have the property of halting on every input, this would not mean that there is no small 2DFA for their union. All these 2DFAs might be as inexplicable as the one in Example 3, and before such examples could be understood and formally investigated, one can hardly expect any lower bounds on the state complexity of union for 2DFAs.

4. Kleene star

Consider the concatenation closure operation, known as the Kleene star: $L^* = \bigcup_{k=0}^{\infty} L^k$, where L^k denotes a concatenation of k copies of L. For 1DFAs, the state complexity of the Kleene star is $\frac{3}{4}2^n$ for alphabets containing at least two letters [14], but only $(n - 1)^2 + 1$ for the unary alphabet [21]. The latter result is implied by the following two properties:

Theorem G (*Yu, Zhuang and Salomaa [21]*)**.** (i) *Let $\mathcal{A}$ be a unary 1DFA with n states. Then there is a 1DFA for $L(\mathcal{A})^*$ with $(n - 1)^2 + 1$ states.*

(ii) *For every $n \geqslant 2$, the language $L = a^{n-1}(a^n)^*$ has an n-state 1DFA, while the language L^* is co-finite, and its minimal 1DFA has tail of length $(n - 1)^2$ and period 1.*

This result can be "lifted" to 2DFAs, in the following sense. First, the star of a 2DFA can be represented by first converting it to a 1DFA and then applying the construction of Theorem G(i). Secondly, the periodic witness language in Theorem G(ii) can be inflated to $L = a^{g(n)-1}(a^{g(n)})^*$ using a 2DFA with n states, and then the language L^* requires a large 2DFA, because it is co-finite. This leads to the following asymptotically tight bounds:

Theorem 3. (i) *The Kleene star of every n-state unary 2DFA can be represented by a 2DFA with $(g^+(n))^2 + 1$ states.*

(ii) *For every $n \geqslant 2$, the language $L = a^{g(n)-1}(a^{g(n)})^*$ is representable by an n-state 2DFA, but every 2DFA recognizing L^* has at least $(g(n) - 1)^2$ states.*

Proof. To show the upper bound, consider an arbitrary n-state unary 2DFA recognizing a language $L \subseteq a^*$. By Theorem C, there exists a 1DFA with $g^+(n) + 1$ states recognizing the same language. The latter 1DFA can be transformed, according to Theorem G(i), to another 1DFA with $(g^+(n))^2 + 1$ states recognizing the language L^*.

Turning to the lower bound, let $n \geqslant 1$ and $g(n) = p_1 \cdots p_k$, where p_i are powers of distinct primes, and consider the language $L = a^{g(n)-1}(a^{g(n)})^*$. This language is recognized by a 2DFA with $p_1 + \cdots + p_k \leqslant n$ states of the form given in Example 1, which checks that the length of the input string is congruent to $p_i - 1$ modulo each p_i. At the same time, according to Theorem G(ii), the minimal 1DFA for the language L^* has tail of length $(g(n) - 1)^2$ and period 1. Therefore, by Theorem B, every 2DFA for L^* requires at least $(g(n) - 1)^2$ states. □

These bounds are asymptotically equivalent, and the state complexity of Kleene star for unary 2DFAs is hence estimated as $(1+o(1))g(n)^2 = e^{(2+o(1))\sqrt{n \ln n}}$. It could be possible to tighten these bounds further, by first determining the state complexity of star for unary 1DFAs as a function of both their tail and their cycle, along the lines of Pighizzini and Shallit [18, Sect. 5], and then using that result instead of Theorem G(i) in an upper bound proof, as in Theorem 3.

The lower bound can also be improved by transferring the portion of n unused in $g(n)$ to the tail, as in the definition of $g^+(n)$. For example, for $n = 21$, these two functions assume values $g(21) = 420$ and $g^+(21) = 422$, and hence Theorem 3 gives a lower bound of $419^2 = 175561$ states; this lower bound is witnessed by the language $L = a^{419}(a^{420})^*$, recognized by a 19-state 2DFA, whose Kleene star L^* has tail 175561 and period 1. However, there is another language, $L' = a^{421}(a^{420})^*$, recognized by a 21-state 2DFA, for which $(L')^*$ has tail 176401 and period 1. This language witnesses that the state complexity of star of 21-state unary 2DFAs is at least 176401, which is closer to the upper bound of $422^2 + 1 = 178085$ states given in Theorem 3.

However, generalizing these examples to every n, and making any substantial improvements to the bounds of Theorem 3, would require a deeper understanding of the effect of the Kleene star on unary regular languages than is currently attained.

5. Power

Viewing concatenation as a product of languages, one can consider the k-th power of a language, defined as the concatenation of k of its copies, $L^k = L \cdots L$. For 1DFAs over a general alphabet, Domaratzki and Okhotin [2] demonstrated that the k-th power operation requires $\Theta(n2^{(k-1)n})$ states. In the unary case, Rampersad [19] proved that the state complexity of L^k for 1DFAs is exactly $kn - k + 1$. More precisely, the following facts are known.

Theorem H (*Rampersad [19]*). (i) *For every $k \geqslant 1$ and for every unary 1DFA $\mathcal{A}$ with n states (with tail $\ell \geqslant 0$ and period $p \geqslant 1$), the language $L(\mathcal{A})^k$ is representable by a 1DFA with $kn - k + 1$ states (with tail $k\ell + (k-1)p - k + 1$ and period p).*

(ii) *For each $n \geqslant 1$, the language $L = a^{n-1}(a^n)^*$ is recognized by an n-state 1DFA, while, for every $k \geqslant 1$, the minimal 1DFA for L^k has $kn - k + 1$ states, with tail $(k-1)(n-1)$ and period n.*

These properties shall now be lifted to 2DFAs similarly to the case of the Kleene star.

Theorem 4. *Let $k \geqslant 2$. The state complexity of L^k for unary 2DFAs is at least $(k-1)g(n) - k$ and at most $kg^+(n) + 1$.*

Proof. For the lower bound, let $p_1, \ldots, p_m \geqslant 2$ be the numbers from the definition of Landau's function for n, which satisfy $p_1 + \cdots + p_m \leqslant n$ and $\mathrm{lcm}(p_1, \ldots, p_m) = g(n)$. Then there is an n-state 2DFA recognizing the inflated language $L = a^{g(n)-1}(a^{g(n)})^*$. By Theorem H(ii), the minimal 1DFA for L^k has tail of length $(k-1)(g(n)-1)$ and period $g(n)$. Then, by Theorem B, every 2DFA for L^k must have at least $p_1 + \cdots + p_m + (k-1)(g(n)-1) - 1 \geqslant (k-1)g(n) - k$ states.

Upper bound: Let L be recognized by an n-state 2DFA. By Theorem C, there is an equivalent 1DFA with $g^+(n) + 1$ states. Then Theorem H asserts that there is a 1DFA for L^k with $kg^+(n) + 1$ states. □

Unfortunately, the bounds in Theorem 4 do not lead to any asymptotical estimation better than $\Theta(g(n))$. The reason can be traced to the limitations of the lower bound method of Theorems A and B, which estimate the number of states needed to represent a period $g(n)$ only as $\Omega(n)$. On the other hand, in all upper bound arguments, such a period cannot in general be represented using fewer than $g(n)$ states, because it is not known whether this particular language has an efficient representation by an n-state 2DFA (like the language in Example 3). If all that is known about a language is its tail and its period, this kind of gap between the lower and the upper bound is inevitable, at least until the existence of such automata as the one in Example 3 is understood.

6. Concatenation

The state complexity of concatenation for unary 1DFAs is determined by methods much like those used for the star and the power operations. In particular, the lower bound is witnessed by two cyclic languages with a co-finite concatenation.

Theorem I (*Yu, Zhuang and Salomaa [21]*). *For all relatively prime m and n, the languages $K = a^{m-1}(a^m)^*$ and $L = a^{n-1}(a^n)^*$ are recognized by an m-state 1DFA and an n-state 1DFA, respectively, while their concatenation KL is a co-finite language, for which the minimal 1DFA has tail $mn - 1$ and period 1.*

Yu et al. [21] also gave a matching upper bound of mn states in a 1DFA representing a concatenation of an m-state and an n-state 1DFA. The following more refined upper bound also reflects the dependence of tail and cycle lengths of the minimal 1DFA for concatenation on the tails and the cycles of the arguments.

Theorem J (*Pighizzini and Shallit [18]*). *Let $K \subseteq a^*$ be represented by a 1DFA with tail k and period p, and let $L \subseteq a^*$ have a 1DFA with tail ℓ and period q. Then there is a 1DFA recognizing KL with tail $k + \ell + \mathrm{lcm}(p, q) - 1$ and period $\mathrm{lcm}(p, q)$.*

Attempting to lift the lower bound in Theorem I to 2DFAs in the same way as in Sections 4 and 5 leads to the following ultimately unsuccessful argument. Given m and n, one would choose one set of cycle lengths $p_1, \ldots, p_k$ with $p_1 + \cdots + p_k \leqslant m$ and $g(m) = p_1 \cdots p_k$, as well as another set of cycle lengths $q_1, \ldots, q_\ell$ with $q_1 + \cdots + q_\ell \leqslant n$ and $g(n) = q_1 \cdots q_\ell$, and construct a pair of an m-state and an n-state 2DFA with these cycle lengths, which recognize the languages $K = a^{g(m)-1}(a^{g(m)})^*$ and $L = a^{g(n)-1}(a^{g(n)})^*$, expecting to obtain a lower bound $g(m)g(n)$ on the state complexity of their concatenation. However, the last step would not work out, because the numbers $g(m)$ and $g(n)$ are almost never

Table 1
The values of the function $g(m, n) = \max\{\operatorname{lcm}(p_1, \ldots, p_k, q_1, \ldots, q_\ell) \mid p_1 + \cdots + p_k \leqslant m,\ q_1 + \cdots + q_\ell \leqslant n\}$.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	4	6	6	12	15	20	30	30	60	60	84	105
2		2	6	6	10	10	14	30	30	42	42	70	70	90	210
3			6	12	15	15	30	30	60	60	84	105	120	210	210
4				12	20	20	30	60	60	84	84	140	140	210	420
5					30	30	60	60	70	105	140	210	210	420	420
6						30	60	60	70	105	140	210	210	420	420
7							84	105	140	210	210	420	420	420	420
8								120	210	210	420	420	420	420	840
9									210	420	420	420	420	660	840
10										420	420	420	840	840	924
11											420	840	840	1260	1260
12												840	1260	1260	1540
13													1260	1320	2310
14														2310	2520
15															4620

relatively prime, and their greatest common divisor is typically large. The latter, according to Theorem J, would imply that the concatenation KL can be represented by a 1DFA with a much shorter tail than intended, which would in turn diminish the lower bound on a 2DFA for this language.

For this argument to work, for every pair of numbers m, n, one has to find partitions of each of these numbers into powers of distinct primes, so that no primes are shared between the partitions. Landau's function as it is does not provide such partitions, and hence it is necessary to define its variant

$$g(m, n) = \max\{\operatorname{lcm}(p_1, \ldots, p_k, q_1, \ldots, q_\ell) \mid p_1 + \cdots + p_k \leqslant m,\ q_1 + \cdots + q_\ell \leqslant n\} \leqslant g(m + n),$$

where the upper bound is by taking the same numbers $p_1, \ldots, p_k, q_1, \ldots, q_\ell$ within the definition of $g(m + n)$. As in the definition of $g(n)$, one can assume that $p_1, \ldots, p_k, q_1, \ldots, q_\ell$ are relatively prime: there is no loss of generality in this assumption, because for every pair of numbers p, p' in this list, one of them can be divided by $\gcd(p, p')$ without affecting the least common multiple.

For example, the value $g(9, 14) = 660$ is reached on the numbers $p_1 = 2^2$, $p_2 = 5$, $q_1 = 3$ and $p_4 = 11$, and it is less than $g(9 + 14) = 840$. On the other hand, $g(10, 13) = 840 = g(10 + 13) = \operatorname{lcm}(3, 5, 7, 2^3)$, because these four numbers can be arranged into two blocks as $p_1 = 3, p_2 = 7, q_1 = 5, q_2 = 2^3$.

From the calculated values of $g(m, n)$ given in Table 1, it appears that if both m and n are large enough, then $g(m, n)$ becomes equal to $g(m + n)$. For example, if $m + n = 89$, then $g(m, n) = g(89)$, as long as $m, n \geqslant 11$; however, $g(10, 79) = g(88) < g(89)$. But proving or even formulating any related results apparently requires a nontrivial analysis of the combinatorial properties of the partitions in the definition of Landau's function, which is not attempted in this paper.

A close relation between these two functions can be established by showing that $\ln g(m, n)$ is asymptotically equivalent to $\ln g(m + n)$.

Lemma 1. $g(m, n) = e^{(1+o(1))\sqrt{(m+n)\ln(m+n)}}$.

Proof. The argument follows the general outline of Miller's [16] proof of Landau's [13] asymptotics of $g(n)$.

Let π_i denote the i-th prime, and define a variant of Landau's function, in which the cycle lengths are preset to be the first primes:

$$f(n) = \pi_1 \cdots \pi_k,$$

where k is the greatest number with $\pi_1 + \cdots + \pi_k \leqslant n$. Denote the next prime by

$$h(n) = \pi_{k+1}.$$

The proof given by Miller [16] proceeds by showing that $\ln g(n) \sim \ln f(n)$ and $\ln f(n) \sim \sqrt{n \ln n}$. The latter argument also contains an estimation of h as $h(n) \sim \sqrt{n \ln n}$.

Following the same course, consider a variant of $g(m, n)$, in which as many first primes as possible are used in the partition of m, and the subsequent primes are used for n:

$$f(m, n) = \pi_1 \cdots \pi_k \cdot \pi_{k+1} \cdots \pi_\ell,$$

where k is the greatest number with $\pi_1 + \cdots + \pi_k \leqslant m$, and ℓ is the greatest number with $\pi_{k+1} + \cdots + \pi_\ell \leqslant n$.

This function can be tightly bounded as follows:

$$\frac{f(m + n)}{h(m + n)} \leqslant f(m, n) \leqslant f(m + n),$$

where the upper bound holds by definition, and the denominator $h(m+n)$ in the lower bound reflects that at most one prime is unused due to the partition of the number $m+n$ into m and n. To prove the lower bound, let k be the largest integer with $\pi_1+\cdots+\pi_k \leqslant m$, and write $f(m+n)$ as $f(m+n)=\pi_1\cdots\pi_k\cdot\pi_{k+1}\cdots\pi_\ell$. Then $f(m,n)$ equals either $f(m+n)$ or $\pi_1\cdots\pi_k\cdot\pi_{k+1}\cdots\pi_{\ell-1}$, since $\pi_{k+1}+\cdots+\pi_{\ell-1}\leqslant\pi_{k+2}+\cdots+\pi_\ell\leqslant n$. Consequently, $f(m,n)\geqslant\frac{f(m+n)}{\pi_\ell}\geqslant\frac{f(m+n)}{h(m+n)}$.

Turning to the function $g(m,n)$, it can now be estimated as follows:

$$\frac{f(m+n)}{h(m+n)}\leqslant f(m,n)\leqslant g(m,n)\leqslant g(m+n).$$

Since both the upper and the lower bounds are asymptotically $e^{(1+o(1))\sqrt{(m+n)\ln(m+n)}}$ [16], so is $g(m,n)$. □

With this estimation of the function $g(m,n)$, the complexity of concatenation can be expressed in terms of this function, and be in turn estimated.

Theorem 5. *The state complexity of concatenation of an m-state 2DFA and an n-state 2DFA over a unary alphabet is at least $g(m,n)-1$ and at most $2g(m,n)+m+n$.*

Proof. Lower bound: According to the definition of $g(m,n)$, let $p_1,\ldots,p_k\geqslant 2$ and $q_1,\ldots,q_\ell\geqslant 2$ be the relatively prime numbers with $p_1+\cdots+p_k\leqslant m$ and $q_1+\cdots+q_\ell\leqslant n$, for which $g(m,n)=\operatorname{lcm}(p_1,\ldots,p_k,q_1,\ldots,q_\ell)$. The numbers $\operatorname{lcm}(p_1,\ldots,p_k)=p_1\cdots p_k$ and $\operatorname{lcm}(q_1,\ldots,q_\ell)=q_1\cdots q_\ell$ are relatively prime as well.

Consider the languages $K=a^{p_1\cdots p_k-1}(a^{p_1\cdots p_k})^*$ and $L=a^{q_1\cdots q_\ell-1}(a^{q_1\cdots q_\ell})^*$, recognized by an m-state 2DFA and an n-state 2DFA, respectively (both as in Example 1). Since their periods are relatively prime, by Theorem I, the minimal 1DFA for their concatenation KL has tail of length $p_1\cdots p_k\cdot q_1\cdots q_\ell-1=g(m,n)-1$ and period 1. Then, by Theorem B, every 2DFA for this language requires at least $g(m,n)-1$ states.

Upper bound: Given a 2DFA $\mathcal{A}$ with m states, by Theorem A there exists an equivalent 1DFA $\mathcal{A}'$ with tail of length ℓ and period $\operatorname{lcm}(p_1,\ldots,p_k)$, where $k\geqslant 1$ and $p_1,\ldots,p_k,\ell\geqslant 1$ satisfy $p_1+\cdots+p_k+\ell\leqslant m+1$. Similarly, for a 2DFA $\mathcal{B}$ with n states, let $\mathcal{B}'$ be an equivalent 1DFA with tail of length ℓ' and period $\operatorname{lcm}(q_1,\ldots,q_{k'})$, where $k'\geqslant 1$ and $q_1,\ldots,q_{k'},\ell'\geqslant 1$ are such that $q_1+\cdots+q_{k'}+\ell'\leqslant n+1$.

Then, by Theorem J, the language $L(\mathcal{A}')L(\mathcal{B}')$ has a 1DFA with tail of length $\ell+\ell'+\operatorname{lcm}(p_1,\ldots,p_k,q_1,\ldots,q_{k'})-1$ and period $\operatorname{lcm}(p_1,\ldots,p_k,q_1,\ldots,q_{k'})$. The total number of states in this 1DFA is at most

$$\ell+\ell'+2\operatorname{lcm}(p_1,\ldots,p_k,q_1,\ldots,q_{k'})\leqslant m+n+2g(m,n).\quad\square$$

Accordingly, the state complexity of concatenation of an m-state 2DFA and an n-state 2DFA is estimated as $\Theta(g(m,n))=e^{(1+o(1))\sqrt{(m+n)\ln(m+n)}}$.

7. Concatenation and related operations for 2NFAs

The last topic to be investigated in this paper is the state complexity of basic operations for two-way nondeterministic finite automata (2NFA) over a one-letter alphabet. Only Boolean operations were previously considered: Geffert et al. [5] established that an n-state unary 2NFA can be complemented using $O(n^8)$ states, and the authors [11] showed that the complexity of union of an m-state 2NFA and an n-state 2NFA is exactly $m+n$, while intersection requires between $m+n$ and $m+n+1$ states.[4]

A 2NFA is obtained by adding nondeterminism to the definition of a 2DFA. It is defined as a sextuple $\mathcal{A}=(\Sigma,Q,Q_0,\delta,F_{\vdash},F_{\dashv})$, where:

- Σ is the input alphabet with $\vdash,\dashv\notin\Sigma$,
- Q is a finite set of states,
- $Q_0\subseteq Q$ is the set of initial states,
- $\delta\colon Q\times(\Sigma\cup\{\vdash,\dashv\})\to 2^{Q\times\{-1,+1\}}$ is a transition function, which lists possible actions in a given state, when observing a given symbol, and
- $F_{\vdash},F_{\dashv}\subseteq Q$ are sets of accepting states, effective on the left-end marker $\vdash$ and on the right-end marker $\dashv$, respectively.

A computation of $\mathcal{A}$ on a string $\vdash w\dashv=\vdash a_1\ldots a_\ell\dashv$, with $\ell\geqslant 0$ and $a_1,\ldots,a_\ell\in\Sigma$, begins in any state from Q_0, with the head over the left-end marker $\vdash$. At every step, when $\mathcal{A}$ is in a state q and observes a symbol $a\in\Sigma\cup\{\vdash,\dashv\}$, it may choose any element (q',d) from the set $\delta(q,a)$, and then enter the state q' and move the head in the direction d. If, eventually, $\mathcal{A}$ reaches any state from $F_{\vdash}$ while on the left-end marker $\vdash$, or any state from $F_{\dashv}$ while on the right-end marker $\dashv$, the computation is accepting. The language $L(\mathcal{A})$ is the set of input strings, on which there is at least one accepting computation.

The known lower bounds on the state complexity of operations on unary 2NFAs [11] were exclusively based on the period of languages. On the other hand, as evident from Sections 4–6, concatenation and related operations produce languages with long tails, and the desired complexity results require lower bounds on the number of states in 2NFAs based on the length of these tails. Suitable bounds can be inferred from the following known transformation of an n-state unary 2NFA to an equivalent 1DFA, which gives a quadratic upper bound on the length of the tail.

[4] With some exceptions, for which the lower bounds are only $m+n-1$ (these are the cases of m or n equal to 6, as well as finitely many other cases), and with one further exception $m=n=6$ with $(m+n-2)$-state lower bounds.

Theorem K (*Mereghetti and Pighizzini [15, Thm. 3.6]*). *For every $n \geqslant 1$ and for every unary 2NFA $\mathcal{A}$ with n states, there exists an equivalent complete 1DFA with the tail containing at most $5 \cdot n^2 + 1$ states.*

This result directly implies the following lower bound on the number of states in a 2NFA recognizing a language with a tail of a certain length.

Corollary K.1. *Let $L \subseteq a^*$ be a regular language with the tail of length $\ell \geqslant 0$. Then, every 2NFA for L requires at least $\Omega(\sqrt{\ell})$ states.*

The plan is to obtain bounds on the state complexity of the Kleene star, the k-th power and the concatenation for unary 2NFAs, by using this lower bound method in the same way as Theorem B was used in Sections 4–6. The earlier upper bound arguments for 2DFAs actually constructed 1DFAs for the results of the operations; the corresponding arguments for 2NFAs can produce smaller 1NFAs instead. Thus, references to Theorems G and I shall be replaced by the following fact about the complexity of operations on unary 1NFAs.

Theorem L (*Holzer and Kutrib [6, Thms. 8,9]*). (i) *The Kleene star of an n-state unary 1NFA over the alphabet $\{a\}$ is representable by a 1NFA with $n + 1$ states, and this number is necessary in the worst case.*

(ii) *The concatenation of an m-state 1NFA and an n-state 1NFA over the alphabet $\{a\}$ is representable by a 1NFA with $m + n$ states, and at least $m + n - 1$ states are necessary in the worst case.*

Even though the same arguments as used for 2DFAs generally work for 2NFAs, the gap between the lower bound and the upper bound is much wider this time.

Theorem 6. *The Kleene star of every n-state 2NFA over a unary alphabet is representable by a 2NFA with $g(n) + O(n^2)$ states. For every $n \geqslant 1$, the language $L = a^{g(n)-1}(a^{g(n)})^*$ is representable by an n-state 2NFA, but every 2NFA accepting L^* has at least $\Omega(g(n))$ states.*

Proof. The proof follows the one for Theorem 3. The upper bound is obtained by first converting an n-state 2NFA to an equivalent 1DFA with $g(n) + O(n^2)$ states, according to Theorem K. Then, by Theorem L(i), the star of the same language is represented by a 1NFA with one extra state, that is, with $g(n) + O(n^2)$ states.

For the lower bound, take a number $n \geqslant 1$ and consider the numbers $p_1, \ldots, p_k \geqslant 2$ from the definition of Landau's function, for which $g(n) = p_1 \cdots p_k$. Then the language $L = a^{g(n)-1}(a^{g(n)})^*$ is recognized by a 2DFA with n states. By Theorem G(ii), the minimal 1DFA for L^* has tail of length $(g(n) - 1)^2$. To recognize a language with such a tail, a 2NFA needs at least $\Omega(\sqrt{(g(n) - 1)^2}) = \Omega(g(n))$ states by Corollary K.1. □

Thus, the state complexity of the Kleene star operation for 2NFAs is asymptotically estimated as $\Theta(g(n)) = e^{(1+o(1))\sqrt{n \ln n}}$.

The bounds on the concatenation obtained by the same method are much less precise.

Theorem 7. *The state complexity of concatenation of an m-state 2NFA and an n-state 2NFA over a unary alphabet is at least $\Omega(\sqrt{g(m, n)})$ and at most $g(m) + g(n) + O((m + n)^2)$.*

Proof. The proof similarly follows that of Theorem 5.

For the lower bound, given $m, n \geqslant 2$, consider the relatively prime numbers $p_1, \ldots, p_k \geqslant 2$, $q_1, \ldots, q_\ell \geqslant 2$ with $p_1 + \cdots + p_k \leqslant m$ and $q_1 + \cdots + q_\ell \leqslant n$, for which $g(m, n) = \operatorname{lcm}(p_1, \ldots, p_k, q_1, \ldots, q_\ell)$. Let $K = a^{p_1 \cdots p_k - 1}(a^{p_1 \cdots p_k})^*$ and $L = a^{q_1 \cdots q_\ell - 1}(a^{q_1 \cdots q_\ell})^*$; these languages are recognized by an m-state 2DFA and an n-state 2DFA, respectively. They have relatively prime periods, and therefore, by Theorem I, their concatenation KL has tail of length $p_1 \cdots p_k \cdot q_1 \cdots q_\ell - 1 = g(m, n) - 1$. Then, by Corollary K.1, every 2NFA for KL requires at least $\Omega(\sqrt{g(m, n) - 1})$ states.

Turning to the upper bound, let $\mathcal{A}$ and $\mathcal{B}$ be 2NFAs with m and n states, respectively. Then, by Theorem K, there exist equivalent 1DFAs $\mathcal{A}'$ and $\mathcal{B}'$ with $g(m) + O(m^2)$ and $g(n) + O(n^2)$ states, respectively. By Theorem L(ii), their concatenation is representable by a 1NFA with $g(m) + g(n) + O(m^2) + O(n^2)$ states. □

Therefore, the state complexity of concatenation for unary 2NFAs can be roughly estimated as $e^{\Theta(\sqrt{(m+n)\ln(m+n)})}$.

Before approaching the state complexity of the last operation, the k-th power, for 2NFAs, consider its effect on the size of 1NFAs. For 1NFAs over multiple-letter alphabets, the state complexity of L^k was proved to be exactly kn by Domaratzki and Okhotin [2]. The precise complexity in the unary case was left as an open problem, with the following easy bounds implicitly mentioned.

Proposition (*Domaratzki and Okhotin [2, Thm. 3, Clm. 5]*). *The state complexity of L^k for unary 1NFAs is at least $kn - k + 1$ and at most kn, with the lower bound witnessed by the languages $L_n = \{a^{n-1}\}$.*

Theorem 8. *For each $k \geqslant 2$, the state complexity of the k-th power operation for unary 2NFAs is at most $O(k \cdot g(n))$ and at least $\Omega(\sqrt{k \cdot g(n)})$.*

Proof. If a language $L \subseteq a^*$ is recognized by a 2NFA with n states, then, by Theorem K, there is an equivalent 1DFA with $g(n) + O(n^2)$ states. Applying Theorem H(i) to the latter 1DFA gives a 1DFA for the language L^k with $k(g(n) + O(n^2)) - k + 1$ states.

Let $n \geqslant 2$ and consider the numbers $p_1, \ldots, p_k \geqslant 2$ with $g(n) = p_1 \cdots p_k$. The language $L = a^{g(n)-1}(a^{g(n)})^*$ has an n-state 2DFA. By Theorem H(ii), the language L^k has tail of length $(k - 1)(g(n) - 1)$. By Corollary K.1, any 2NFA for the language with such a tail requires at least $\Omega(\sqrt{(k - 1)(g(n) - 1)}) = \Omega(\sqrt{k \cdot g(n)})$ states. □

Therefore, the k-th power operation requires asymptotically $(k \cdot g(n))^{\Theta(1)} = k^{\Theta(1)} \cdot e^{\Theta(\sqrt{n \ln n})}$ states for 2NFAs.

Table 2
State complexity of operations for one-way and two-way automata over a unary alphabet: complementation ($\sim$), intersection ($\cap$), union ($\cup$), concatenation ($\cdot$), Kleene star (*) and k-th power (^{k}). Notation: $g(n)$ is Landau's function, $g(m,n)$ is its two-argument generalization defined in Section 6, and $g^+(n) = \max_{0\leqslant \ell<n} g(n-\ell)+\ell$.

	1DFA	1NFA	2DFA	2NFA
$\sim$	n	$g(n)+O(n^2)$ [6]	$n \leqslant \cdot \leqslant 2n+3$ [12]	$n \leqslant \cdot \leqslant C\cdot n^8$ [5]
$\cap$	mn	mn	$m+n \leqslant \cdot \leqslant m+n+1$	$m+n \leqslant \cdot \leqslant m+n+1$ [11]
$\cup$	mn	$m+n+1$ [6]	$m+n \leqslant \cdot \leqslant 2m+n+4$	$m+n$ [11]
$\cdot$	mn [21]	$m+n-1 \leqslant \cdot \leqslant m+n$ [6]	$g(m,n)-1 \leqslant \cdot \leqslant 2g(m+n)+m+n$	$\Omega(\sqrt{g(m,n)}) \leqslant \cdot \leqslant g(m)+g(n)+O((m+n)^2)$
*	$(n-1)^2+1$ [21]	$n+1$ [6]	$(g(n)-1)^2 \leqslant \cdot \leqslant g^+(n)^2+1$	$\Omega(g(n)) \leqslant \cdot \leqslant g(n)+O(n^2)$
^{k}	$kn-k+1$ [19]	$kn-k+1 \leqslant \cdot \leqslant kn$ [2]	$(k-1)g(n)-k \leqslant \cdot \leqslant kg^+(n)+1$	$\Omega(\sqrt{k\cdot g(n)}) \leqslant \cdot \leqslant k(g(n)+O(n^2))$

Table 3
State complexity of operations for 2DFAs, compared for a unary and for a general alphabet: complementation ($\sim$), intersection ($\cap$), union ($\cup$), concatenation ($\cdot$), Kleene star (*), k-th power (^{k}), reversal (^{R}) and inverse homomorphisms (h^{-1}). Notation: $g(n)$ is Landau's function, $g(m,n)$ is its two-argument generalization, $g^+(n) = \max_{0\leqslant \ell<n} g(n-\ell)+\ell$.

	alphabet $\Sigma=\{a\}$	any alphabet Σ
$\sim$	$n \leqslant \cdot \leqslant 2n+3$ [12]	$n \leqslant \cdot \leqslant 4n+\text{const}$ [5]
$\cap$	$m+n \leqslant \cdot \leqslant m+n+1$	$m+n \leqslant \cdot \leqslant m+n+1$
$\cup$	$m+n \leqslant \cdot \leqslant 2m+n+3$	$m+n \leqslant \cdot \leqslant 4m+n+\text{const}$
$\cdot$	$g(m,n)-1 \leqslant \cdot \leqslant 2g(m+n)+m+n$	$\Omega(\frac{m}{n})+\frac{2^{\Omega(n)}}{\log m} \leqslant \cdot \leqslant 2m^{m+1}\cdot 2^{n^{n+1}}$ [7]
*	$(g(n)-1)^2 \leqslant \cdot \leqslant g^+(n)^2+1$	$\frac{1}{n}2^{\frac{n}{2}-1} \leqslant \cdot \leqslant 2^{O(n^{n+1})}$ [7]
^{k}	$(k-1)g(n)-k \leqslant \cdot \leqslant kg^+(n)+1$	$\frac{1}{n}2^{\frac{n}{2}-1} \leqslant \cdot \leqslant 2^{O(n^{n+1})}$ [7]
^{R}	n	$n+1$ [7]
h^{-1}		$2n$ [7]

8. Summary

The state complexity of operations over a unary alphabet for one-way and two-way finite automata, deterministic and nondeterministic, is compared in the Table 2.

Comparing the new results on unary 2DFAs to the known results for multiple-letter alphabets by Jirásková and Okhotin [7], the new lower bounds for union and for intersection improve over the previous bound $m+n-o(m+n)$, and do so using only a unary alphabet, vs. an alphabet of exponential size. Turning to the concatenation, star and power operations, their state complexity for multiple-letter alphabets is $2^{\Omega(n)}$ [7], while the complexity for a unary alphabet is tied to Landau's function and is therefore smaller. A comparison of the unary case to the general case is given in Table 3.

Acknowledgments

The authors are grateful to the anonymous referees for pointing out several problems with the presentation.

The authors have used computer calculations to build the initial understanding of the state complexity of 2DFAs. These calculations were carried out on the computers of the CSC—IT Center for Science (Espoo, Finland, www.csc.fi).

The first author is supported by the research center Institute for Theoretical Computer Science (ITI), project No. P202/12/G061 of the Czech Science Foundation. The second author is supported by the Academy of Finland under grant 134860.

References

[1] M. Chrobak, Finite automata and unary languages, Theoretical Computer Science 47 (1986) 149–158; 302 (2003) 497–498 (Errata) http://dx.doi.org/10.1016/S0304-3975(03)00136-1.
[2] M. Domaratzki, A. Okhotin, State complexity of power, Theoretical Computer Science 410 (24–25) (2009) 2377–2392. http://dx.doi.org/10.1016/j.tcs.2009.02.025.
[3] R.E. Dressler, A stronger Bertrand's postulate with an application to partitions, Proceedings of the AMS 33 (2) (1972) 226–228. http://www.jstor.org/stable/2038035.
[4] V. Geffert, C. Mereghetti, G. Pighizzini, Converting two-way nondeterministic unary automata into simpler automata, Theoretical Computer Science 295 (1–3) (2003) 189–203. http://dx.doi.org/10.1016/S0304-3975(02)00403-6.
[5] V. Geffert, C. Mereghetti, G. Pighizzini, Complementing two-way finite automata, Information and Computation 205 (8) (2007) 1173–1187. http://dx.doi.org/10.1016/j.ic.2007.01.008.
[6] M. Holzer, M. Kutrib, Nondeterministic descriptional complexity of regular languages, International Journal of Foundations of Computer Science 14 (2003) 1087–1102. http://dx.doi.org/10.1142/S0129054103002199.
[7] G. Jirásková, A. Okhotin, On the state complexity of operations on two-way finite automata, in: Developments in Language Theory, (DLT 2008, Kyoto, Japan, 16–19 September 2008), in: LNCS, vol. 5257, pp. 443–454 http://dx.doi.org/10.1007/978-3-540-85780-8_35 ; full version in progress.

[8] C.A. Kapoutsis, Removing bidirectionality from nondeterministic finite automata, in: Mathematical Foundations of Computer Science, (MFCS, 2005, Gdansk, Poland, 29 August–2 September 2005), in: LNCS, vol. 3618, pp. 544–555, http://dx.doi.org/10.1007/11549345_47.
[9] A. Kondacs, J. Watrous, On the power of quantum finite state automata, in: 38th Annual Symposium on Foundations of Computer Science, FOCS 1997, Miami Beach, Florida, USA, 19–22 October 1997), IEEE, pp. 66–75 http://dx.doi.org/10.1109/SFCS.1997.646094.
[10] M. Kunc, A. Okhotin, Describing periodicity in two-way deterministic finite automata using transformation semigroups, in: Developments in Language Theory (DLT 2011, Milan, Italy, 19–22 July 2011), in: LNCS, vol. 6795, pp. 324–336, http://dx.doi.org/10.1007/978-3-642-22321-1_28.
[11] M. Kunc, A. Okhotin, State complexity of union and intersection for two-way nondeterministic finite automata, Fundamenta Informaticae 110 (1–4) (2011) 231–239. http://dx.doi.org/10.3233/FI-2011-540.
[12] M. Kunc, A. Okhotin, Reversible two-way finite automata over a unary alphabet, TUCS Technical Report 1024, Turku Centre for Computer Science, December 2011.
[13] E. Landau, 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) (1903) 92–103.
[14] A.N. Maslov, Estimates of the number of states of finite automata, Soviet Mathematics Doklady 11 (1970) 1373–1375.
[15] C. Mereghetti, G. Pighizzini, Optimal simulations between unary automata, SIAM Journal on Computing 30 (6) (2001) 1976–1992. http://dx.doi.org/10.1137/S009753979935431X.
[16] W. Miller, The maximum order of an element of a finite symmetric group, American Mathematical Monthly 94 (6) (1987) 497–506. http://dx.doi.org/10.2307/2322839.
[17] A. Okhotin, Unambiguous finite automata over a unary alphabet, Information and Computation 212 (2012) 15–36. http://dx.doi.org/10.1016/j.ic.2012.01.003.
[18] G. Pighizzini, J. Shallit, Unary language operations, state complexity and Jacobsthal's function, International Journal of Foundations of Computer Science 13 (1) (2002) 145–159. http://dx.doi.org/10.1142/S012905410200100X.
[19] N. Rampersad, The state complexity of L^2 and L^k, Information Processing Letters 98 (2006) 231–234. http://dx.doi.org/10.1016/j.ipl.2005.06.011.
[20] M. Sipser, Lower bounds on the size of sweeping automata, in: 11th Annual ACM Symposium on Theory of Computing (STOC 1979, 30 April–2 May 1979, Atlanta, Georgia, USA), pp. 360–364, http://dx.doi.org/10.1145/800135.804429.
[21] S. Yu, Q. Zhuang, K. Salomaa, The state complexity of some basic operations on regular languages, Theoretical Computer Science 125 (1994) 315–328. http://dx.doi.org/10.1016/0304-3975(92)00011-F.