State complexity of cyclic shift

Galina Jirásková*
Mathematical Institute,
Slovak Academy of Sciences,
Grešákova 6, 040 01 Košice, Slovakia
jiraskov@saske.sk

Alexander Okhotin†
Department of Mathematics,
University of Turku
FIN–20014 Turku, Finland
okhotin@cs.utu.fi

February 25, 2010

Abstract

The cyclic shift of a language L, defined as $\textsc{shift}(L) = \{uv \mid vu \in L\}$, is an operation known to preserve both regularity and context-freeness. Its descriptional complexity has been addressed in Maslov's pioneering paper on the state complexity of regular language operations (*Soviet Math. Dokl.*, 11, 1970), where a high lower bound for partial DFAs using a growing alphabet was given. We improve this result by using a fixed four-letter alphabet, obtaining a lower bound $(n-1)! \cdot 2^{(n-1)(n-2)}$, which shows that the state complexity of cyclic shift is $2^{n^2+n\log n+\Omega(n)}$ for alphabets with at least four letters. For a binary alphabet, we prove $2^{\Omega(n^2)}$ state complexity. We also establish a tight $2n^2+1$ upper bound on the nondeterministic state complexity of this operation for any non-unary alphabet.

1 Introduction

Cyclic shift is a unary operation on formal languages defined as $\textsc{shift}(L) = \{uv \mid vu \in L\}$ and occasionally studied since the 1960s. As it can be naturally expected, the cyclic shift of every regular language is regular as well, and proving that is a good exercise in finite automata theory [5, Exercise 3.4(c)]. Another less expected property is that the context-free languages are also closed under cyclic shift. Three independent proofs of this result are known: the proof by Oshiba [10] is based upon the representation of a context-free language as a homomorphic image of a Dyck language intersected with a regular set; Maslov [9] uses pushdown automata; Hopcroft and Ullman [5, Exercise 6.4(c)] directly transform a context-free grammar to another grammar generating its cyclic shift.

The number of states in a deterministic finite automaton (DFA) needed to recognize the cyclic shift of an n-state DFA language has been addressed in Maslov's pioneering paper on the state complexity of operations on regular languages [8]. In that paper, which appeared in 1970 and unfortunately remained unnoticed, the state complexity of quite a few operations on DFAs with partially defined transition functions has been determined.

*Research supported by the VEGA Grant no. 2/3164/23.
†Supported by the Academy of Finland under grant 206039.

In particular, Maslov obtains tight upper bounds on the state complexity of union, intersection, concatenation, and star for a binary alphabet, as well as asymptotical estimations for several less common operations, among them the cyclic shift.

The systematic study of the state complexity of operations on regular languages began only about twenty years later in the works of Birget [1] and Yu, Zhuang and K. Salomaa [12]. Unlike Maslov, who considered DFAs with partially defined transition functions, the contemporary research assumes complete automata; however, the results on the basic operations differ from Maslov results by at most 1 state [9, 12]. The new study of the state complexity got a considerable following. State complexity of numerous operations was investigated: in particular, Câmpeanu, K. Salomaa and Yu [2] determined the state complexity of shuffle, Domaratzki [3] studied proportional removals, while A. Salomaa, Wood and Yu [11] investigated the reversal (mirror image) in various cases. The state complexity of various operations with respect to nondeterministic finite automata was researched in the works of Holzer and Kutrib [4] and Jirásková [7]. Many authors also considered the special cases of one-letter alphabets and of finite languages. Comprehensive surveys of the field were given by Yu [13] and by Hromkovič [6].

In the recent research, many operations on DFAs, such as concatenation [12], star [12], shuffle [2] and reversal [11], were found to be quite hard, in the sense that their the state complexity is an exponent of a linear function. In this context it is interesting to observe that the earlier studied cyclic shift operation is, in fact, significantly harder. According to Maslov [8], there exists a sequence of n-state partial DFAs over a growing alphabet of size $2n-2$, such that their cyclic shift requires at least $(n-2)^{n-2} \cdot 2^{(n-2)^2}$ states. Unfortunately, no proof of that was included due to space constraints [8].

The current interest in different aspects of descriptional complexity of finite automata motivates a return to this unusually hard operation and a closer investigation of its state complexity. After giving fairly simple upper bounds, in Section 3 we achieve a lower bound $(n-1)! \cdot 2^{(n-1)(n-2)}$ using a fixed 4-letter alphabet and DFAs with a complete transition function. Then we extend our construction to obtain a lower bound $2^{n^2/9+o(n^2)}$ for a binary alphabet. We also study the nondeterministic state complexity and, in Section 4, determine it precisely. In Section 5 we report the results of a direct computation of the deterministic state complexity for up to five states.

2 Constructing finite automata for the cyclic shift

The cyclic shift of a language L is defined as $\textsc{shift}(L) = \{uv \mid vu \in L\}$. In this section, we recall the construction of an automaton accepting the cyclic shift of a given regular language presented by Maslov [9]. Using this construction we get upper bounds on the state complexity and the nondeterministic state complexity of cyclic shift.

Let $A = (Q, \Sigma, \delta, q_0, F)$ be an n-state automaton with the set of states $Q = \{q_0, q_1, \ldots, q_{n-1}\}$. For all i $(0 \leqslant i \leqslant n-1)$, let $B_i = (Q, \Sigma, \delta, q_i, F)$ be an automaton with the same states, the same transitions, and the same accepting states as A, and with the initial state q_i. Let $C_i = (Q, \Sigma, \delta, q_0, \{q_i\})$ be an automaton with the same states, the same transitions, and the same initial state as A, and with the only accepting state q_i. Note that if automaton A is deterministic, then so are automata B_i and C_i.

By definition, a string w is in $\textsc{shift}(L(A))$ if and only if it can be factorized as $w = uv$

so that $vu \in L(A)$. Consider the middle state in the accepting computation of A on vu, reached after consuming v, and let us reformulate the condition as follows: there exists a state $q_i \in Q$, such that the computation of A on v ends in the state q_i, while u is accepted by A starting from q_i. The former is equivalent to $v \in L(C_i)$, the latter means $u \in L(B_i)$. This leads to the following representation, in which the union over all i is, in effect, a union over all possible middle states.

Lemma 1 (Maslov [9])**.** *Let automata B_i and C_i ($0 \leqslant i \leqslant n-1$) be constructed as above. Then* $\textsc{shift}(L(A)) = \bigcup_{i=0}^{n-1} L(B_i)L(C_i)$.

Since the state complexity of union and concatenation is known both for DFAs and for NFAs, this representation gives an upper bound for the cyclic shift's state complexity.

Definition 1. *For every $k \geqslant 1$ and $n \geqslant 1$, let $f_k(n)$ be the least number, such that the cyclic shift of the language recognized by any n-state DFA over a k-letter alphabet can be recognized by a $f_k(n)$-state DFA. Similarly, define $g_k(n)$ to be the nondeterministic state complexity of the cyclic shift over a k-letter alphabet. Let $f(n)$ and $g(n)$ be the corresponding numbers for an arbitrary alphabet, i.e., $f(n) = \max_k f_k(n)$, $g(n) = \max_k g_k(n)$.*

Theorem 1. *Let $f_k(n)$, $g_k(n)$, $f(n)$, $g(n)$ be as in the above definition, let $n \geqslant 1$. Then*
$n = f_1(n) \leqslant f_2(n) \leqslant \ldots \leqslant f_k(n) \leqslant f_{n^n}(n) = f(n) \leqslant (n2^n - 2^{n-1})^n$, *and*
$n = g_1(n) \leqslant g_2(n) \leqslant \ldots \leqslant g_k(n) \leqslant g_{2^{n^2}}(n) = g(n) \leqslant 2n^2 + 1$.

Proof. The upper bounds on the state complexity of union and concatenation of an m-state DFA language and an n-state DFA language are known to be mn and $m2^n - 2^{n-1}$, respectively [9, 12]. Using representation in Lemma 1 we get an upper bound $(n2^n - 2^{n-1})^n$ on the state complexity of cyclic shift. Lemma 1 gives also an upper bound on the nondeterministic state complexity of this operation, because each concatenation can be done by $2n$ nondeterministic states, and hence the union of n such concatenations can done by $2n^2+1$ nondeterministic states. Thus we have $f(n) \leqslant (n2^n - 2^{n-1})^n$ and $g(n) \leqslant 2n^2+1$.

The equalities $f_1(n) = g_1(n) = n$ hold because the cyclic shift of any unary language is the same language. For all $k \geqslant 1$, the inequalities $f_k(n) \leqslant f_{k+1}(n)$ and $g_k(n) \leqslant g_{k+1}(n)$ follow by adding dummy letters.

In order to show that $f_{n^n}(n) = f_k(n)$ for all $k \geqslant n^n$, it is sufficient to note that there exist n^n distinct possible transition tables by a symbol, and if there are more letters than n^n, some of them must have identical transitions, and any such coincident pairs coincide in cyclic shift of the language; hence the equality of $f_{n^n}(n)$ to $f(n)$. In the nondeterministic case, by a similar reasoning, the growth must stop at $k = 2^{n^2}$, because the transitions by each letter form a subgraph of the complete graph over n vertices, and therefore there cannot be more than 2^{n^2} distinct letters. □

3 Deterministic state complexity

In order to determine the asymptotics of the DFA complexity of the cyclic shift, we construct an infinite sequence of DFAs $\{A_n\}_{n \geqslant 3}$ over a fixed 4-symbol alphabet $\Sigma = \{a, b, c, d\}$ such that all automata recognizing $\textsc{shift}(L(A_n))$ must have at least $(n-1)! \cdot 2^{(n-1)(n-2)}$ states.

3.1 Hard automata

Each n-th element of the sequence is an automaton A_n with the set of states $Q = \{0, 1, \ldots, n-1\}$, of which 0 is the initial state and $n-1$ is the only accepting state. The transitions by each of the four symbols are defined as follows:

$$\delta(i,a) = \begin{cases} 0 & \text{if } i = 0, \\ i+1 & \text{if } 1 \leqslant i \leqslant n-3, \\ 1 & \text{if } i = n-2, \\ n-1 & \text{if } i = n-1, \end{cases} \qquad \delta(i,b) = \begin{cases} 0 & \text{if } i = 0, \\ i+1 & \text{if } 1 \leqslant i \leqslant n-2, \\ 1 & \text{if } i = n-1, \end{cases}$$

$$\delta(i,c) = \begin{cases} 1 & \text{if } i = 0, \\ 0 & \text{if } i = 1, \\ i & \text{if } i \geqslant 2, \end{cases} \qquad \delta(i,d) = \begin{cases} 0 & \text{if } i \leqslant n-2, \\ 1 & \text{if } i = n-1. \end{cases}$$

In order to argue that the minimal DFA accepting $\textsc{shift}(L(A_n))$ must have many states, we consider a $(2n^2+1)$-state NFA recognizing this language, and show that the subset construction applied to this NFA yields many pairwise inequivalent subsets. This NFA is constructed generally according to the representation given by Lemma 1.

Let $Q_i = \{q_{i0}, \ldots, q_{in-1}\}$ and $P_i = \{p_{i0}, \ldots, p_{in-1}\}$ $(0 \leqslant i \leqslant n-1)$ be $2n$ copies of Q. Let there be a dedicated initial state with ε-transitions to all states q_{ii} $(0 \leqslant i \leqslant n-1)$. The internal transitions within each Q_i and P_i are defined exactly as in the DFA A_n. In addition, there is an ε-transition from each q_{in-1} to p_{i0} $(0 \leqslant i \leqslant n-1)$. The set of accepting states is $\{p_{00}, p_{11}, \ldots, p_{n-1n-1}\}$. This construction of an NFA is illustrated in Figure 1. Note that each pair (Q_i, P_i) represents a concatenation of two DFAs, B_i and C_i in Lemma 1, and the whole automaton recognizes the union of n such concatenations.

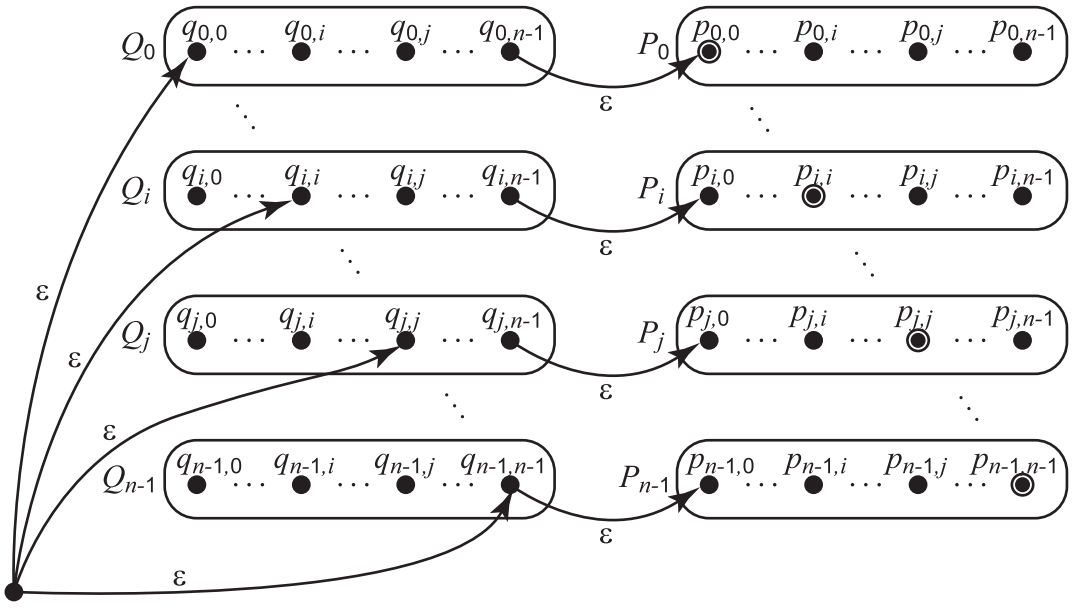

Figure 1: Construction of an NFA for $\textsc{shift}(L(A_n))$.

In the following we discuss reachability and inequivalence of subsets of the set of states of this NFA, which are exactly the states of the DFA obtained out of this NFA using the

subset construction. The *initial subset* is $\{q_{00}, q_{11}, \ldots, q_{n-1n-1}, p_{n-10}\}$. Let us begin with proving that a large set of subsets is reachable from the initial subset.

3.2 Reachable subsets

We first prove the reachability of $(n-1)! \cdot 2^{(n-1)(n-2)}$ subsets via strings over $\{a, b, c\}$.

Lemma 2 (The permutation lemma)**.** *For every permutation $(k_1, \ldots, k_{n-1})$ of $\{1, \ldots, n-1\}$, there exists a string $u \in \{a, b\}^*$ such that, upon reading u, (i) q_{00} goes to q_{00}; (ii) p_{i0} goes to p_{i0} (for all i); (iii) q_{ij} goes to q_{ik_j} and, possibly, also to p_{i0} (for all $1 \leqslant i, j \leqslant n-1$); (iv) p_{ij} goes to p_{ik_j}.*

Proof. First, consider the case when the given permutation swaps $m-1 \leftrightarrow m$ for some m, leaving the other states as they are (formally, $k_{m-1} = m$, $k_m = m-1$, and $k_i = i$ for all $i \neq m, m-1$). This permutation is implemented by $u_m = b^{n-m-1}ab^{m-1}$.

It is well-known that every permutation can be expressed as a composition of such swaps, so a concatenation of several strings u_m defined above implements any given permutation. □

Lemma 3 (Setting a bit)**.** *For every state p_{ij} such that $1 \leqslant i \leqslant n-1$, $1 \leqslant j \leqslant n-1$ and $i \neq j$, there exists a string w_{ij} such that every subset of the form*

$$S = \{q_{00}, q_{11}, q_{22}, \ldots, q_{n-1n-1}\} \cup \{p_{10}, p_{20}, \ldots, p_{n-10}\} \cup P, \tag{1}$$

where $P \subseteq \{p_{km} \mid 1 \leqslant k \leqslant n-1,\ 1 \leqslant m \leqslant n-1, k \neq m\}$, goes to $S \cup \{p_{ij}\}$ upon reading w_{ij}.

Proof. Let x be the string given by the permutation lemma for the permutation $\{i \leftrightarrow n-1, j \leftrightarrow 1\}$ (note that its square is the identity permutation), and define w_{ij} as $xccx$.

The state q_{ii} goes to $\{q_{in-1}, p_{i0}\}$ by x, then to $\{q_{in-1}, p_{i1}, p_{i0}\}$ by c, to $\{q_{in-1}, p_{i0}, p_{i1}\}$ by the second c and proceeds to $\{q_{ii}, p_{i0}, p_{ij}\}$ by x: this is where the bit p_{ij} is being set. Every other state q_{kk} $(1 \leqslant k \leqslant n-1,\ k \neq i)$ proceeds to $\{q_{k\ell}, p_{k0}\}$ by x (where $\ell \neq n-1$, as defined by the permutation), then goes to $\{q_{k\ell'}, p_{k1}\}$ by the first c (where $\ell' = 0$ if $\ell = 1$, and $\ell' = \ell$ otherwise), back to $\{q_{k\ell}, p_{k0}\}$ by the second c, and to $\{q_{kk}, p_{k0}\}$ by x afterwards. The state q_{00} goes to q_{00} by x, then to q_{01} by the first c, back to q_{00} by the second c, and remains in q_{00} upon reading x.

It remains to show that each state p_{km} in (1) goes to itself. Similarly to q_{00}, every p_{k0} $(1 \leqslant k \leqslant n-1)$ goes to p_{k0} by $xccx$. Every $p_{km} \in P$ goes to $p_{k\ell}$ by x (where ℓ is defined by the permutation), to $p_{k\ell'}$ by the first c, to $p_{k\ell}$ by the second c and back to p_{km} by x.

Assembling all these states together, we obtain exactly the subset $S \cup \{p_{ij}\}$. □

Lemma 4 (Reachability)**.** *For any $P \subseteq \{p_{ij} \mid 1 \leqslant i \leqslant n-1,\ 1 \leqslant j \leqslant n-1,\ i \neq j\}$, the subset (1) is reachable from the initial subset via some string in $\{a, b, c\}^*$.*

Proof. Induction on the cardinality of P.

Basis $P = \varnothing$: the subset $\{q_{00}, q_{11}, q_{22}, \ldots, q_{n-1n-1}\} \cup \{p_{10}, p_{20}, \ldots, p_{n-10}\}$ is reachable from the initial subset $\{q_{00}, q_{11}, q_{22}, \ldots, q_{n-1n-1}\} \cup \{p_{n-10}\}$ via the string b^{n-1}.

Induction step: consider a subset of the form (1), and let $p_{ij} \in P$. By the induction hypothesis, the subset $S \setminus \{p_{ij}\}$ is reachable from the initial subset via some string $u \in \{a, b, c\}^*$. By Lemma 3, S is reachable from $S \setminus \{p_{ij}\}$ via w_{ij}. Therefore, S is reachable from the initial subset via uw_{ij}. □

Lemma 4 proves the reachability of $2^{(n-1)(n-2)}$ subsets. In order to multiply this number by a factor of $(n-1)!$, we need to permute the Q_i-components.

Lemma 5 (Improved reachability)**.** *Let $(k_1, \ldots, k_{n-1})$ be any permutation of $(1, \ldots, n-1)$ and let P is be any subset of $\{p_{ij} \mid 1 \leqslant i \leqslant n-1,\ 1 \leqslant j \leqslant n-1,\ i \neq j\}$. Then the subset*

$$\{q_{00}, q_{1k_1}, q_{2k_2}, \ldots, q_{n-1k_{n-1}}\} \cup \{p_{10}, p_{20}, \ldots, p_{n-10}\} \cup \{p_{ik_j} \mid p_{ij} \in P\} \tag{2}$$

is reachable from the initial subset via some string in $\{a, b, c\}^*$.

Proof. First, consider the subset $\{q_{00}, q_{11}, q_{22}, \ldots, q_{n-1n-1}\} \cup \{p_{10}, p_{20}, \ldots, p_{n-10}\} \cup P$, which, by Lemma 4, is reachable from the initial subset via some string u. Let v be the string given by the permutation lemma for the permutation $(k_1, \ldots, k_{n-1})$. The subset (2) is reachable from the initial subset via the string uv. □

3.3 Inequivalence of subsets

We now prove the inequivalence of all subsets that were shown to be reachable in Lemma 5. To do this we first associate a distinct string with each state p_{ij} $(1 \leqslant i, j \leqslant n-1)$ such that this string is accepted by the NFA only from the state p_{ij}. The same will be done for states q_{ij} $(1 \leqslant i, j \leqslant n-1)$. Then, the inequivalence of the subsets will follow immediately.

Lemma 6. *For every i and j $(1 \leqslant i \leqslant n-1,\ 1 \leqslant j \leqslant n-1)$, the string $b^{n-1-j}db^{i-1}$ is accepted by the NFA from the state p_{ij}, but is not accepted from any other state in $\{q_{00}\} \cup \{q_{ij} \mid 1 \leqslant i \leqslant n-1, 1 \leqslant j \leqslant n-1\} \cup \{p_{ij} \mid 1 \leqslant i \leqslant n-1, 0 \leqslant j \leqslant n-1\}$.*

Proof. The state p_{ij} goes to state p_{in-1} by the string b^{n-1-j}, then to state p_{i1} by d, and, finally, to the accepting state p_{ii} by the string b^{i-1}.

On the other hand, the state q_{00} remains in q_{00} upon reading the string $b^{n-1-j}db^{i-1}$. Any state q_{kl} $(1 \leqslant k \leqslant n-1, 1 \leqslant l \leqslant n-1)$ goes either to a state of Q_k or to state p_{k0} by the string $b^{n-1-j}db^{i-1}$. Any state p_{il} with $l \neq j$ goes to state p_{i0} by the string $b^{n-1-j}d$ and then remains in p_{i0} upon reading b^{i-1}. Any state p_{kl} with $k > 0$ and $k \neq i$ goes either to state p_{k0} or to state p_{ki} by the string $b^{n-1-j}db^{i-1}$. □

Lemma 7. *For every i and j $(1 \leqslant i \leqslant n-1,\ 1 \leqslant j \leqslant n-1)$, the string $b^{n-1-j}db^{n-2}ccb^{n-2}db^{i-1}$ is accepted by the NFA from the state q_{ij}, but is not accepted from any other state in $\{q_{00}\} \cup \{q_{ij} \mid 1 \leqslant i \leqslant n-1, 1 \leqslant j \leqslant n-1\} \cup \{p_{ij} \mid 1 \leqslant i \leqslant n-1, 0 \leqslant j \leqslant n-1\}$.*

Proof. The state q_{ij} goes to state q_{in-1} by the string b^{n-1-j}, then to state q_{i1} by d and to state q_{in-1} by the string b^{n-2}. The first c leaves the state q_{in-1} in itself, while the second one moves it to state p_{i1}. State p_{i1} goes to state p_{in-1} by the string b^{n-2}, then to state p_{i1} by d and, finally, to the accepting state p_{ii} by the string b^{i-1}.

On the other hand, the state q_{00} goes to itself by the string $b^{n-1-j}db^{n-2}ccb^{n-2}db^{i-1}$. Any state $q_{i\ell}$ with $\ell \neq j$ may go to q_{i0} or to p_{i0} by the string $b^{n-1-j}d$ (p_{i0} is possible when $\ell \geqslant j$), and each of these two states goes to itself by the remaining string $b^{n-2}ccb^{n-2}db^{i-1}$. Any state $q_{k\ell}$ with $k > 0$ and $k \neq i$ may go to a state in $\{q_{k0}, q_{k1}, p_{k0}, p_{k1}\}$ after reading the second d, and then to a state in $\{q_{k0}, q_{ki}, p_{k0}, p_{ki}\}$ by the string b^{i-1}. Any state $p_{k\ell}$ $(1 \leqslant k \leqslant n-1,\ 0 \leqslant \ell \leqslant n-1)$ may go either to p_{k0} or to p_{k1} after reading the first d.

In the first case, state p_{k0} goes to itself by the string $b^{n-2}ccb^{n-2}db^{i-1}$ In the second case, state p_{k1} goes to state p_{kn-1} by the string $b^{n-2}cc$, then to p_{kn-2} by the string b^{n-2}, and then to p_{k0} by the string db^{i-1}. □

Corollary 1. *All subsets shown to be reachable in Lemma 5 are pairwise inequivalent.*

Proof. Consider two different subsets shown to be reachable in Lemma 5. These two subsets must differ either in a state p_{ij} with $i > 0$ and $j > 0$, or in a state q_{ij} with $i > 0$ and $j > 0$. In the first case, the string $b^{n-1-j}db^{i-1}$ distinguishes the two states by Lemma 6. In the second case, the string $b^{n-1-j}db^{n-2}ccb^{n-2}db^{i-1}$ distinguishes them by Lemma 7. □

Hence we have shown the following result.

Theorem 2. *For all $n \geqslant 3$, there exists a DFA A of n states defined over a four-letter alphabet such that any DFA for the cyclic shift of the language $L(A)$ needs at least $(n-1)! \cdot 2^{(n-1)(n-2)}$ states.*

Using Stirling's approximation, it is not hard to show that $(n-1)! \cdot 2^{(n-1)(n-2)} = 2^{n^2+n\log_2 n-(3+\log_2 e)n+o(n)}$. Recalling the upper bound $(n2^n-2^{n-1})^n$ and its $2^{n^2+n\log n+o(1)}$ asymptotics, we arrive at the following estimation of the state complexity of the cyclic shift:

Corollary 2. *For all $k \geqslant 4$, $f_k(n) = 2^{n^2+n\log n+O(n)}$.*

3.4 The case of a binary alphabet

The lower bound argument above uses four symbols, and reducing the number of symbols in the proof even to three seems to be a challenging task.

Let us use the lower bound for an alphabet of four letters to establish quite a high lower bound for a binary alphabet. The proof is based upon the following method of computing the cyclic shift of any language over any alphabet: the strings in the language are homomorphically encoded using a binary alphabet, the cyclic shift is applied to the encoding, and then the shifted codewords are decoded back to the original alphabet. For a suitable code, the result equals the cyclic shift of the original language.

Lemma 8. *Let $\Sigma_m = \{c_1, \ldots, c_m\}$ and let $h : \Sigma_m^* \to \{a,b\}^*$ be a homomorphism defined by $h(c_i) = a^{i-1}b$ (for all $1 \leqslant i \leqslant m-1$) and $h(c_m) = a^{m-1}$. Then, for every language L over Σ_m, $h^{-1}(\textsc{shift}(h(L))) = \textsc{shift}(L)$.*

Proof. If $w \in \textsc{shift}(L)$, then there exists a factorization $w = uv$, such that $vu \in L$. Then $h(vu) \in h(L)$ and hence $h(uv) \in \textsc{shift}(h(L))$. By the definition of inverse homomorphism, $uv \in h^{-1}(\textsc{shift}(h(L)))$.

Conversely, let $c_{i_1} \ldots c_{i_\ell} \in h^{-1}(\textsc{shift}(h(L)))$, and assume that at least one of c_{i_j} is not c_m (the case of $c_{i_j} = c_m$ for all j is trivial, since $h(c_m^\ell) \in a^*$). Then $h(c_{i_1} \ldots c_{i_\ell}) \in \textsc{shift}(h(L))$, and there exists a factorization $h(c_{i_1} \ldots c_{i_\ell}) = xy$ (where $x, y \in \{a,b\}^*$), such that $yx \in h(L)$.

Suppose the factorization xy splits the codeword on the boundary between two symbols: $x = h(c_{i_1} \ldots c_{i_k})$, $y = (c_{i_{k+1}} \ldots c_{i_\ell})$. Then $h(c_{i_{k+1}} \ldots c_{i_\ell} c_{i_1} \ldots c_{i_k}) \in h(L)$, and hence, since h is a code, $c_{i_{k+1}} \ldots c_{i_\ell} c_{i_1} \ldots c_{i_k} \in L$. Therefore, $c_{i_1} \ldots c_{i_k} c_{i_{k+1}} \ldots c_{i_\ell} \in \textsc{shift}(L)$.

Now suppose the factorization splits some k-th symbol in two, i.e., $x = h(c_{i_1} \dots c_{i_{k-1}})z'$, $y = z''(c_{i_{k+1}} \dots c_{i_\ell})$, where $h(c_{i_k}) = z'z''$ $(z', z'' \in \{a,b\}^+)$. Note that $z' = a^k$ $(1 \leqslant k < m-1)$, since all proper prefixes of the images of symbols under h are of this form. So we have $z''h(c_{i_{k+1}} \dots c_{i_\ell} c_{i_1} \dots c_{i_k})a^k \in h(L)$, and the string is assumed to contain at least one b. Consider the rightmost b in it, which must be either the last symbol in z'' or the last symbol in some $h(c_{i_j})$. In both cases, the string continues with zero or more images of c_m and then with a^k. Since no string in $h(\Sigma_m^*)$ can be of this form, this supposedly problematic case is in fact impossible. □

Lemma 9. *For every n-state DFA A over an m-letter alphabet there exists an $f_2(mn-n)$-state DFA for the language* $\textsc{shift}(L(A))$.

Proof. Define a homomorphism $h : \{c_1, \dots, c_m\}^* \to \{a,b\}^*$ as in Lemma 8. It is easy to construct an $(m-1)n$-state DFA B over $\{a,b\}$ that recognizes $h(L(A))$.

Consider the language $\textsc{shift}(L(B)) \subseteq \{a,b\}^*$. By the definition of f_2, there exists a DFA $C = (Q, \{a,b\}, \delta, q_0, F)$ that recognizes this language and has at most $f_2(mn-n)$ states. Construct a new DFA $D = (Q, \Sigma_m, \delta', q_0, F)$ with the same set of states and the same accepting states, in which the transition function is $\delta'(q, c_i) = \delta(q, h(c_i))$. Then $L(D) = h^{-1}(L(C)) = h^{-1}(\textsc{shift}(h(L(A))))$, which equals $\textsc{shift}(L(A))$ by Lemma 8. □

Corollary 3. *For all $m \geqslant 3$ and $n \geqslant 1$, $f_2(mn-n) \geqslant f_m(n)$.*

This implies that if $f_m(n) = 2^{n^2+o(n^2)}$, then $2^{n^2/(m-1)^2+o(n^2)}$ is a lower bound for $f_2(n)$. Since such a lower bound has been established for $m = 4$ (see Theorem 2), a lower bound $2^{n^2/9+o(n^2)}$ for f_2 follows, and we obtain the following asymptotics:

Proposition 1. *The state complexity of the cyclic shift for binary and ternary alphabets is $2^{\Omega(n^2)}$.*

4 Nondeterministic state complexity

We now turn our attention to the nondeterministic state complexity of the cyclic shift of regular languages. The upper bound $2n^2+1$ follows from Theorem 1. The next lemma shows that this upper bound is tight for any integer n with $n \geqslant 2$. In the case of $n = 1$, the nondeterministic state complexity of the cyclic shift of a 1-state NFA language is 1 because the cyclic shift of any 1-state NFA language is the same language.

Lemma 10. *For any integer n such that $n \geqslant 2$, there exists a binary NFA M of n states such that any NFA for the cyclic shift of the language $L(M)$ needs at least $2n^2+1$ states.*

The proof utilizes a variant of the argument due to Birget [1, Lemma 1], which is sometimes referred as the *fooling set* method. However, a fairly general lemma given by Birget is in our case insufficient to distinguish all $2n^2+1$ states from each other, so we have to construct a more detailed argument that is ultimately based upon the same idea.

Proof. Let n be an arbitrary but fixed integer such that $n \geqslant 2$ and let $\Sigma = \{a,b\}$. Define an n-state NFA $M = (\{0,1,\dots,n-1\}, \Sigma, \delta_M, 0, \{n-1\})$, where for any $p \in \{0,\dots,n-1\}$

$$\delta_M(p,a) = \begin{cases} \{p+1\}, & \text{if } p < n-1, \\ \varnothing, & \text{if } p = n-1. \end{cases} \qquad \delta_M(p,b) = \begin{cases} \varnothing, & \text{if } p = 0, \\ \{p-1\}, & \text{if } p > 0. \end{cases}$$

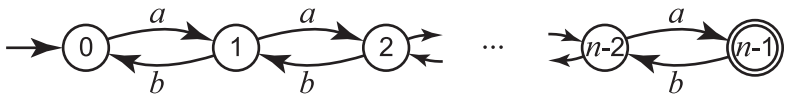

Figure 2: A worst-case NFA that requires $2n^2 + 1$ states for its cyclic shift.

The NFA M is shown in Figure 2. Let L be the language accepted by the NFA M. Let $N = (Q, \Sigma, \delta, q_0, F)$ be any NFA for the language $\textsc{shift}(L)$. We first show that there exist $2n^2$ pairs of strings $(x_i, y_i), i = 1, 2, \ldots, 2n^2$, such that:

(1) for any i, the string $x_i y_i$ is in the language $\textsc{shift}(L)$, and

(2) if $i \neq j$, then at least one of the strings $x_i y_j$ and $x_j y_i$ is not in $\textsc{shift}(L)$.

Then, we can associate a state q_i with each pair (x_i, y_i) such that $q_i \in \delta(q_0, x_i)$ and $\delta(q_i, y_i) \cap F \neq \varnothing$. All states q_i must be pairwise distinct states of the NFA N because otherwise the NFA N would accept both strings $x_i y_j$ and $x_j y_i$ for some $i \neq j$. Finally, we show that the initial state q_0 of the NFA N must be different from any of these $2n^2$ states.

For any $k = 0, 1, \ldots, n-1$, and any $i = 0, 1, \ldots, 2n-1$, define the pair (x_{ki}, y_{ki}) as follows:

$$(x_{ki}, y_{ki}) = \begin{cases} (a^{n-1}b^{n-1}a^i, a^{2n-2-i}b^{n-1}), & \text{if } k = 0 \text{ and } i \notin \{n-1, 2n-1\}, \\ (b^k a^i, a^{2n-2-i}b^{n-1-k}), & \text{if } k > 0 \text{ and } i \notin \{n-1, 2n-1\}, \\ (b^k a^{n-1}, b^{n-1}a^{2n-2}b^{n-1-k}), & \text{if } i = n-1, \\ (b^k a^{2n-2}b^{n-1}, a^{n-1}b^{n-1-k}), & \text{if } i = 2n-1. \end{cases}$$

We have defined $2n^2$ pairs of strings. Now, we will prove that they satisfy the conditions (1) and (2).

To prove (1) we will consider all four cases in the above definition:

(i) $k = 0$ and $i \notin \{n-1, 2n-1\}$. Then $x_{0i}y_{0i} = a^{n-1}b^{n-1}a^{2n-2}b^{n-1}$, and this string is in the language $\textsc{shift}(L)$ because its cyclic shift $a^{n-1}b^{n-1}a^{n-1}b^{n-1}a^{n-1}$ is in the language L.

(ii) $k > 0$ and $i \notin \{n-1, 2n-1\}$. Then $x_{ki}y_{ki} = b^k a^{2n-2}b^{n-1-k}$, and this string is in the language $\textsc{shift}(L)$ because its cyclic shift $a^{n-1}b^{n-1-k}b^k a^{n-1}$ is in the language L.

(iii) $i = n-1$. Then $x_{ki}y_{ki} = b^k a^{n-1}b^{n-1}a^{2n-2}b^{n-1-k}$, which is in the language $\textsc{shift}(L)$.

(iv) $i = 2n-1$. Then $x_{ki}y_{ki} = b^k a^{2n-2}b^{n-1}a^{n-1}b^{n-1-k}$, which is again in $\textsc{shift}(L)$.

To prove (2) we have five cases to consider:

(i) $0 \leqslant k < l \leqslant n-1$. Then for any pairs (x_{ki}, y_{ki}) and (x_{lj}, y_{lj}), we have $x_{lj}y_{ki} = b^l uv b^{n-1-k}$ for some strings u and v. This string is not in the language $\textsc{shift}(L)$ because any its cyclic shift not ending with b contains the string $b^{n-1+l-k}$, where $n-1+l-k > n-1$, as a substring, and so is not in the language L.

(ii) $i, j \notin \{n-1, 2n-1\}$ and $i < j$. Then we have $x_{0j}y_{0i} = a^{n-1}b^{n-1}a^{2n-2+j-i}b^{n-1}$ and for any $k > 0$, $x_{kj}y_{ki} = b^k a^{2n-2+j-i}b^{n-1-k}$, where $2n-2+j-i > 2n-2$, and so these strings are not in the language $\textsc{shift}(L)$.

(iii) $i \notin \{n-1, 2n-1\}$, and $j = n-1$. Then we have $x_{0i}y_{0j} = a^{n-1}b^{n-1}a^ib^{n-1}a^{2n-2}b^{n-1}$ and for any $k > 0$, $x_{ki}y_{kj} = b^ka^ib^{n-1}a^{2n-2}b^{n-1-k}$, and these strings are not in the language $\text{SHIFT}(L)$ for $i \neq n-1$.

(iv) $i \notin \{n-1, 2n-1\}$, and $j = 2n-1$. Then for any k, we have $x_{kj}y_{ki} = b^ka^{2n-2}b^{n-1}a^{2n-2-i}b^{n-1-k}$, which is not in the language $\text{SHIFT}(L)$ for $i \neq n-1$ (the only possible shift of this string that could be in the language L is $a^{n-1}b^{n-1}a^{2n-2-i}b^{n-1-k}b^ka^{n-1}$, which is in L only if $i = n-1$).

(v) $i = n-1$ and $j = 2n-1$. Then for any k, we have $x_{kj}y_{ki} = b^ka^{2n-2}b^{n-1}b^{n-1}a^{2n-2}b^{n-1-k}$. This string is not in the language $\text{SHIFT}(L)$.

We have shown that these $2n^2$ pairs of strings satisfy the conditions (1) and (2). If we had to prove only a $2n^2$ lower bound, it would immediately follow out of this by the lemma of Birget [1]. In order to separate the initial state from the states corresponding to the given pairs, we need to consider these states directly.

Let us associate a state q_{ki} with each pair (x_{ki}, y_{ki}), such that $q_{ki} \in \delta(q_0, x_{ki})$ and $\delta(q_{ki}, y_{ki}) \cap F \neq \varnothing$. As mentioned above, these states are pairwise distinct. To prove the lemma it is sufficient to show that the initial state q_0 is different from any state q_{ki}.

We first show that the initial state q_0 is different from any state q_{ki} with $k > 0$. To do this note that the string $a^{2n-2}b^{n-1}$ is in the language $\text{SHIFT}(L)$, and so must be accepted by the NFA N. Suppose the contrary, that $q_0 = q_{ki}$ for some $k > 0$ and some i. Then the NFA N also accepts the string $x_{ki}a^{2n-2}b^{n-1}$. But $x_{ki} = b^ku$ for a string u, and the string $b^kua^{2n-2}b^{n-1}$ is not in the language $\text{SHIFT}(L)$ if $k > 0$. So, we obtain a contradiction.

It remains to prove that the initial state q_0 is different from any state q_{0i}. To do this note that the string $ba^{2n-2}b^{n-2}$ is in the language $\text{SHIFT}(L)$, and so must be accepted by the NFA N. Suppose that $q_0 = q_{0i}$ for some i. Then the NFA N also accepts the string $x_{0i}ba^{2n-2}b^{n-2}$. But $x_{0i} = a^{n-1}v$ for a string v and the string $a^{n-1}vba^{2n-2}b^{n-2}$ is not in the language $\text{SHIFT}(L)$. So, we again have a contradiction and the lemma follows. □

Hence we have shown the following result.

Theorem 3. *Let $n \geqslant 2, m \geqslant 2$. Then $2n^2 + 1$ states are sufficient and necessary in the worst case for an NFA to recognize the cyclic shift of any n-state NFA language over an m-letter alphabet.*

5 Computations

We have determined the nondeterministic state complexity of cyclic shift exactly, and the worst-case automaton constructed in Lemma 10 has a visible structure. On the other hand, our results for the DFAs are only asymptotic: the automaton constructed for the lower bound argument is not *the* hardest, and the proof does not give any idea of what the hardest DFAs with respect to the cyclic shift are like.

These hardest DFAs can be found using an exhaustive search over all automata over a given alphabet with a given number of states. For every DFA, an NFA for its cyclic shift has to be constructed and determinized, and the result has to be minimized. Let us report the results of our computations for small values of n.

The greatest number of states in the minimal DFAs recognizing the cyclic shift of n-state automata is given in Table 1. The columns f_2, f_3, f_4, f_5 correspond to alphabets of different size, while the column $f(n)$ gives the hardest result over all alphabets. Our lower bound and upper bound are included in the two rightmost columns for comparison.

n	$f_2(n)$	$f_3(n)$	$f_4(n)$	$f_5(n)$	$f(n)$	$(n-1)! \cdot 2^{(n-1)(n-2)}$	$(n2^n - 2^{n-1})^n$
1	1	1	1	1	1	–	1
2	5	5	5	5	5	1	36
3	108	511	702		845	8	8000
4	20237	550283			1349340	384	9834496
5	56817428					98304	61917364224

Table 1: DFA state complexity of the cyclic shift.

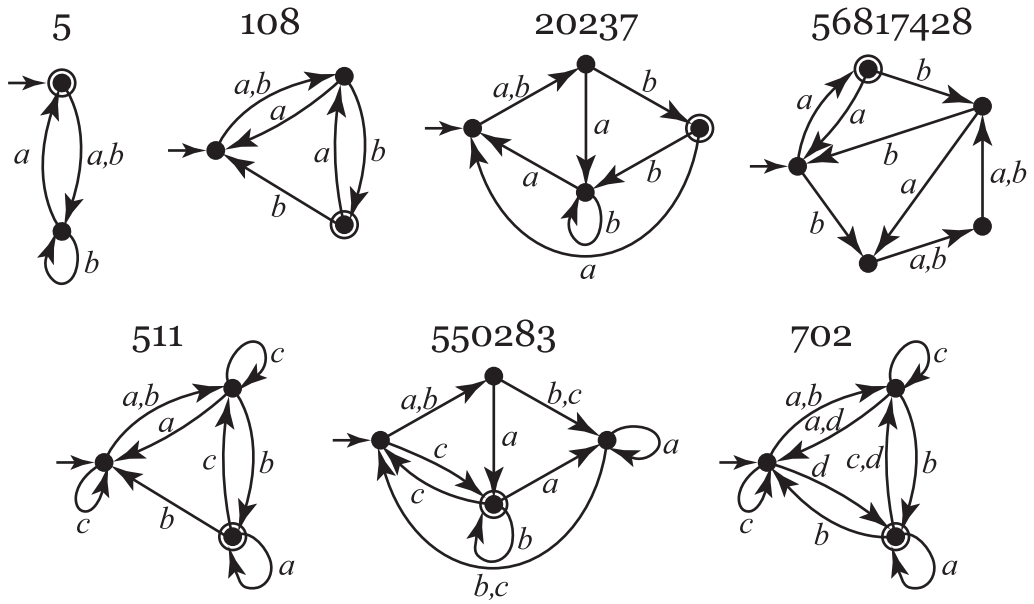

Figure 3: The worst-case DFAs, and how many states their cyclic shift requires.

The actual hardest automata responsible for the values in the table are provided in Figure 3. It is difficult to understand what makes these automata the hardest. It is no easier to explain the numbers in the sequences: why 108? why 20237? For the standard language-theoretic operations, such as the Boolean operations, concatenation, star, etc., the exact values of their state complexity were found to have fairly simple analytical representations [1, 2, 7, 8, 11, 13], and the hardest automata have been constructed. It would be very interesting to obtain similar results for the cyclic shift.

6 Conclusion

With its $2^{n^2+n\log_2 n+O(n)}$ state complexity, the cyclic shift is the hardest known elementary language operation on DFAs. From the point of view of practical computability, the

difference between the cyclic shift and the earlier studied hard operations on DFAs, such as Kleene star, is evident: the star of a 5-state language over $\{a, b\}$ requires at most 24 states, while the cyclic shift, in the worst case, requires 56 million!

In contrast to the hard deterministic case, the NFA state complexity of the cyclic shift has been found to be as low as $2n^2+1$, and an easily understandable worst-case automaton over a binary alphabet has been constructed. Concerning the deterministic state complexity of the cyclic shift, though its order of magnitude has been determined, nothing else is known about this integer sequence, and about the hardest automata corresponding to its elements. Understanding their form is left as a challenging problem to investigate.

References

[1] J.-C. Birget, "Intersection and union of regular languages and state complexity", *Information Processing Letters*, 43:4 (1992), 185–190.

[2] C. Câmpeanu, K. Salomaa, S. Yu, "Tight lower bound for the state complexity of shuffle of regular languages", *J. of Automata, Languages and Combinatorics*, 7 (2002), 303–310.

[3] M. Domaratzki, "State complexity and proportional removals", *J. of Automata, Languages and Combinatorics*, 7 (2002), 455–468.

[4] M. Holzer, M. Kutrib, "Nondeterministic descriptional complexity of regular languages", *International J. of Foundations of Computer Science*, 14 (2003), 1087–1102.

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

[6] J. Hromkovič, "Descriptional complexity of finite automata: concepts and open problems", *J. of Automata, Languages and Combinatorics*, 7 (2002), 519–531.

[7] G. Jirásková, "State complexity of some operations on binary regular languages", *Theoretical Computer Science*, 330 (2005), 287–298.

[8] A. N. Maslov, "Estimates of the number of states of finite automata", *Soviet Mathematics Doklady*, 11 (1970), 1373–1375.

[9] A. N. Maslov, "Cyclic shift operation for languages", *Problems of Information Transmission*, 9:4 (1973), 333–338.

[10] T. Oshiba, "Closure property of the family of context-free languages under the cyclic shift operation", *Transactions of IECE*, 55D:4 (1972), 119–122.

[11] A. Salomaa, D. Wood, S. Yu, "On the state complexity of reversals of regular languages", *Theoretical Computer Science*, 320 (2004), 315–329.

[12] S. Yu, Q. Zhuang, and K. Salomaa, "The state complexity of some basic operations on regular languages", *Theoretical Computer Science*, 125 (1994), 315–328.

[13] S. Yu, "State complexity of regular languages", *J. of Automata, Languages and Combinatorics*, 6 (2001), 221–234.