RAIRO-Theor. Inf. Appl. **42** (2008) 335–360
DOI: 10.1051/ita:2007038

Available online at:
www.rairo-ita.org

STATE COMPLEXITY OF CYCLIC SHIFT *

Galina Jirásková [1] and Alexander Okhotin [2]

Abstract. The cyclic shift of a language L, defined as $\text{SHIFT}(L) = \{vu \mid uv \in L\}$, 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) 1373–1375], where a high lower bound for partial DFAs using a growing alphabet was given. We improve this result by using a fixed 4-letter alphabet, obtaining a lower bound $(n-1)! \cdot 2^{(n-1)(n-2)}$, which shows that the state complexity of cyclic shift is $2^{n^2+n\log n-O(n)}$ for alphabets with at least 4 letters. For 2- and 3-letter alphabets, we prove $2^{\Theta(n^2)}$ state complexity. We also establish a tight $2n^2+1$ lower bound for the nondeterministic state complexity of this operation using a binary alphabet.

Mathematics Subject Classification. 68Q45, 68Q19.

1. Introduction

Cyclic shift is a unary operation on formal languages defined as $\text{SHIFT}(L) = \{vu \mid uv \in L\}$ 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 [11], 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 [19]

Keywords and phrases. Finite automata, descriptional complexity, cyclic shift.

* *This paper has been presented at the Seventh Workshop on Descriptional Complexity of Formal Systems (DCFS 2005) held in Como, Italy on June 30–July 2, 2005.*

[1] Mathematical Institute, Slovak Academy of Sciences, Grešákova 6, 040 01 Košice, Slovakia; jiraskov@saske.sk
Supported by the VEGA Grants no. 2/3164/23 and no. 2/6089/26.

[2] Academy of Finland *and* Department of Mathematics, University of Turku, Turku 20014, Finland; alexander.okhotin@utu.fi
Supported by the Academy of Finland under grants 206039 and 118540.

Article published by EDP Sciences

is based upon the representation of a context-free language as a homomorphic image of a Dyck language intersected with a regular set; Maslov [18] uses pushdown automata; Hopcroft and Ullman ([11], Exercise 6.4(c)) directly transform a context-free grammar to another grammar generating its cyclic shift. In contrast, it is easy to prove that neither linear context-free nor deterministic context-free languages are closed under this operation.

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 [17]. 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. In particular, Maslov obtains tight upper bounds on the state complexity of union, intersection, concatenation and star, as well as asymptotic 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 [2,3] and Yu *et al.* [23]. 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's results by at most 1 state [17,23]. The new study of the state complexity got a considerable following. State complexity of numerous operations was investigated: in particular, Câmpeanu *et al.* [5] determined the state complexity of shuffle, Domaratzki [6] studied proportional removals, while Salomaa *et al.* [20] investigated the reversal (mirror image) in different cases. A recent trend is the study of the state complexity of combinations of basic operations, initiated by Salomaa *et al.* [21]. The state complexity of various operations with respect to nondeterministic finite automata was researched in the works of Holzer and Kutrib [10] and Jirásková [16]. Many authors also considered the special cases of one-letter alphabets and of finite languages. Comprehensive surveys of the field were given by Yu [22] and by Hromkovič [14].

In the recent research, many operations on DFAs, such as concatenation [23], star [23], shuffle [5] and reversal [20], were found to be quite hard, in the sense that their state complexity is exponential in a linear function of the sizes of the original DFAs. In this context it is interesting to observe that the earlier studied cyclic shift operation is, in fact, significantly harder. According to Maslov [17], 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. This implies a $(n-3)^{n-3} \cdot 2^{(n-3)^2}$ lower bound for DFAs with a complete transition function. Unfortunately, no proof of this fact was published due to space constraints [17].

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 of $(n-1)! \cdot 2^{(n-1)(n-2)}$ using a fixed four-letter alphabet

and DFAs with a complete transition function. Then we extend our construction to obtain a lower bound of $2^{n^2/9+o(n^2)}$ 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 cyclic shift

The cyclic shift of a language L is defined as $\textsc{shift}(L) = \{vu \mid uv \in L\}$. In this section, we recall the construction of an automaton accepting the cyclic shift of a given regular language presented by Maslov [17]. Using this construction we get upper bounds on the state complexity and the nondeterministic state complexity of cyclic shift.

Let us fix the types of finite automata we consider and the notation we use. We define a *deterministic finite automaton* (DFA) as a quintuple $(Q, \Sigma, \delta, q_0, F)$, in which Q is a finite set of states, Σ is an input alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of accepting states. We consider only complete DFAs, that is, the transition function is total. A *nondeterministic finite automaton* (NFA) is a quintuple $(Q, \Sigma, \delta, q_0, F)$ with nondeterministic transition function $\delta : Q \times \Sigma \to 2^Q$; this transition function can also be defined as $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ to allow for epsilon transitions, but since such transitions can be eliminated without increasing the number of states, the difference between these automata is inessential to us. Another variant of an NFA, on the contrary, has to be considered separately: these are NFAs with multiple initial states defined as $(Q, \Sigma, \delta, Q_0, F)$, where $Q_0 \subseteq Q$ is a set of initial states.

The *(deterministic) state complexity* of a regular language L is the least number of states in any DFA accepting L. The *nondeterministic state complexity* of a language L is similarly defined as the least number of states in any NFA with a single initial state accepting L.

Let $A = (Q, \Sigma, \delta, q_0, F)$ be an n-state finite automaton, deterministic or nondeterministic, 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 the automaton A is deterministic, then so are the 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 $w = vu$ so that $uv \in L(A)$. Consider the middle state in the accepting computation of A on uv, reached after consuming u, and let us reformulate the condition as follows: there exists a state $q_i \in Q$, such that the computation of A on u ends in state q_i, while v is accepted by A starting from q_i. The former is equivalent to $u \in L(C_i)$, the latter means $v \in L(B_i)$. 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 [17])**.** *Let A be a finite automaton with n states and, as described above, construct finite automata B_i and C_i $(0 \leqslant i \leqslant n-1)$, which are deterministic if A is deterministic. Then*

$$\textsc{shift}(L(A)) = \bigcup_{i=0}^{n-1} L(B_i)L(C_i).$$

Since the deterministic and nondeterministic state complexity of union and concatenation is known, this representation gives upper bounds on the deterministic and nondeterministic state complexity of cyclic shift. First, let us introduce the notation for the corresponding state complexity functions.

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 an $f_k(n)$-state DFA. Similarly, define $g_k(n)$ to be the least number, such that the cyclic shift of the language recognized by any n-state NFA over a k-letter alphabet can be recognized by a $g_k(n)$-state NFA.

Let $f(n)$ and $g(n)$ be the corresponding numbers for an arbitrary alphabet, that is, $f(n) = \max_k f_k(n)$, $g(n) = \max_k g_k(n)$. The functions $f(n)$ and $g(n)$ are the state complexity and the nondeterministic state complexity of cyclic shift, respectively.

Theorem 1. *Let $f_k(n)$, $g_k(n)$, $f(n)$, $g(n)$ be as defined above, let $n \geqslant 1$. Then*

$$n = f_1(n) \leqslant f_2(n) \leqslant \ldots \leqslant f_k(n) \leqslant \ldots \leqslant f_{n^n}(n) = f(n) \leqslant (n2^n - 2^{n-1})^n,$$

and

$$n = g_1(n) \leqslant g_2(n) \leqslant \ldots \leqslant g_k(n) \leqslant \ldots \leqslant g_{2^{n^2}}(n) = g(n) \leqslant 2n^2 + 1.$$

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 [17,23]. Using the representation from Lemma 1 we get an upper bound of $(n2^n - 2^{n-1})^n$ 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 be 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 every 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 each symbol corresponds to one of n^n different transition tables, and if there are more than n^n symbols, some of them must have identical transition tables. Any such coincident pairs will coincide in the cyclic shift of the language, hence the equality of $f_{n^n}(n)$ to $f(n)$.

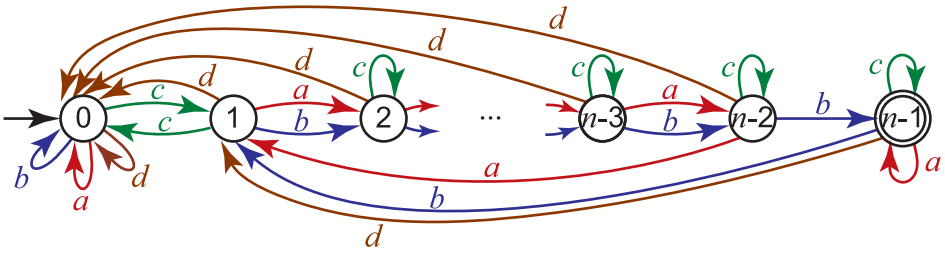


FIGURE 1. A DFA that requires at least $(n-1)! \cdot 2^{(n-1)(n-2)}$ states for its cyclic shift.

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} letters with distinct transition tables. □

3. Deterministic state complexity

In order to determine the asymptotics of the deterministic state complexity of 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 A_n has n states and every deterministic finite automaton recognizing $\text{SHIFT}(L(A_n))$ 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, while the entire automaton is given in Figure 1.

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

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

In order to argue that the minimal DFA accepting $\text{SHIFT}(L(A_n))$ must have many states, we consider a $2n^2$-state NFA with multiple initial states recognizing this language and show that the subset construction applied to this NFA yields many pairwise inequivalent subsets.

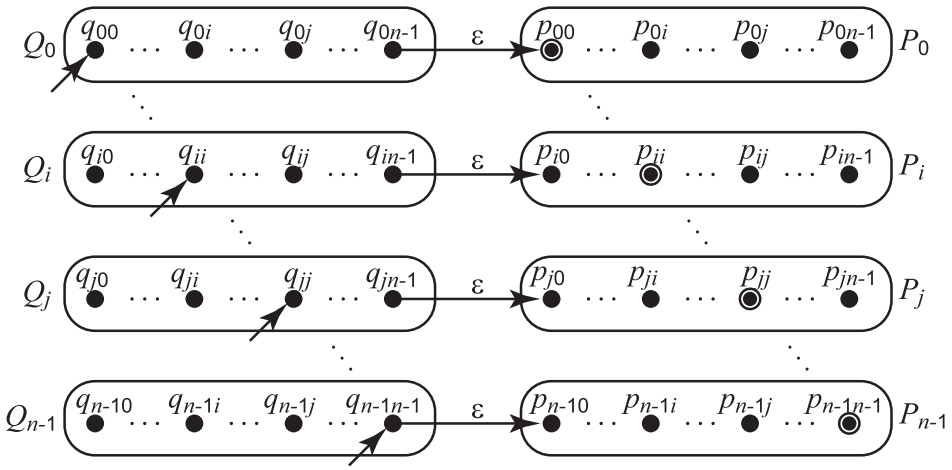


FIGURE 2. Construction of an NFA for SHIFT$(L(A_n))$.

The NFA is constructed generally according to the representation given by Lemma 1. The states of this NFA shall be named using double subscripts, and we shall always omit the comma between these subscripts: for example, a name q_{in-1} means subscripts i and $n-1$. The set of states of the NFA is $Q_0 \cup P_0 \cup Q_1 \cup P_1 \cup \cdots \cup Q_{n-1} \cup P_{n-1}$, where $Q_i = \{q_{i0}, \ldots, q_{in-1}\}$ and $P_i = \{p_{i0}, \ldots, p_{in-1}\}$ $(0 \leqslant i \leqslant n-1)$ are $2n$ copies of Q. 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 initial states are $\{q_{00}, q_{11}, \ldots, q_{n-1n-1}\}$, while the set of accepting states is $\{p_{00}, p_{11}, \ldots, p_{n-1n-1}\}$.

This construction of an NFA is illustrated in Figure 2. 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.

3.2. Reachable subsets

Having defined an NFA for SHIFT$(L(A_n))$, see Figure 2, let us consider the DFA obtained out of this NFA using the subset construction. We shall refer to the states of this DFA as "subsets" (of the set of states of the original NFA), and we shall be concerned with the reachability and inequivalence of these subsets.

By *reachability* we mean reachability from the *initial subset*, which is $\{q_{00}, q_{11}, \ldots, q_{n-1n-1}, p_{n-10}\}$ (state p_{n-10} is there because of an ε-transition from q_{n-1n-1}). We first prove the reachability of $(n-1)! \cdot 2^{(n-1)(n-2)}$ different subsets via strings over the symbols $\{a, b, c\}$.

These symbols have one thing in common: the transitions by each of them form a permutation of the set of states. This ensures the following property.

Lemma 2. *For every string $w \in \{a,b,c\}^*$, the subset reached from the initial subset $\{q_{00}, q_{11}, \ldots, q_{n-1n-1}, p_{n-10}\}$ by w is of the form $\{q_{0k_0}, q_{1k_1}, \ldots, q_{n-1k_{n-1}}\} \cup P$, where $(k_0, \ldots, k_{n-1})$ is a permutation of $(0, \ldots, n-1)$ and $P \subseteq \{p_{ij} \mid 0 \leqslant i \leqslant n-1, 0 \leqslant j \leqslant n-1, j \neq k_i\}$.*

Proof. The proof is by induction on the length of w. The basis holds, since the initial subset is of the required form. For the induction step, consider a string ws, where $w \in \{a,b,c\}^*$ and $s \in \{a,b,c\}$. By the induction hypothesis, the subset S_w reached from the initial subset by w is of the above form for some permutation $(k_0, \ldots, k_{n-1})$ and for some set P. The subsequent transition by s leads to the subset $S_{ws} = \{q_{0\delta(k_0,s)}, \ldots, q_{n-1\delta(k_{n-1},s)}\} \cup P'$ for some $P' \subseteq \{p_{ij} \mid 0 \leqslant i \leqslant n-1, 0 \leqslant j \leqslant n-1\}$. The vector $(\delta(k_0,s), \ldots, \delta(k_{n-1},s))$ is a permutation as a composition of two permutations, namely $(k_0, \ldots, k_{n-1})$ and the transition table by s. It remains to prove that $p_{i\delta(k_i,s)} \notin P'$ for every i $(0 \leqslant i \leqslant n-1)$.

Consider how $p_{i\delta(k_i,s)}$ could appear in P'. If $p_{i\delta(k_i,s)} = p_{i\delta(j,s)}$ for some $p_{ij} \in P$, then, since the transitions by s form a permutation, $k_i = j$. Therefore, $p_{ik_i} \in P$, which, by the induction hypothesis, cannot be the case.

The other seemingly possible way to obtain $p_{i\delta(k_i,s)}$ is by an ε-transition from q_{in-1}. Then q_{in-1} must belong to S_{ws}, that is, $q_{i\delta(k_i,s)} = q_{in-1}$, and thus $\delta(k_i,s) = n-1$. On the other hand, the ε-transition from q_{in-1} goes to p_{i0}, and hence $p_{i\delta(k_i,s)} = p_{i0}$, which implies $\delta(k_i,s) = 0 \neq n-1$. The contradiction obtained proves this case impossible. □

This property relies only upon all symbols being permutations. Using the specific form of the transition tables for a, b and c, let us strengthen this property for a limited class of accessing strings, in which the symbols c occur in pairs only.

Lemma 3. *Assume $w \in \{a,b,cc\}^*$. Then the subset reachable from the initial subset via w contains q_{00} and is disjoint from P_0, i.e., it is of the form $\{q_{00}, q_{1k_1}, \ldots, q_{n-1k_{n-1}}\} \cup P$, where $(k_1, \ldots, k_{n-1})$ is a permutation of $(1, \ldots, n-1)$ and $P \subseteq \{p_{ij} \mid 1 \leqslant i \leqslant n-1, 0 \leqslant j \leqslant n-1, j \neq k_i\}$.*

Proof. The induction is straightforward: q_{00} is in the initial subset, the transitions by a and b leave q_{00} as it is, while the transitions by cc lead q_{00} to q_{01} and back to q_{00}. The state q_{0n-1} is thus never visited, and therefore none of the states from P_0 can appear in the subset reached by w. □

Lemma 3 gives a necessary form of subsets reachable *via* $\{a,b,cc\}^*$. The main result of this subsection is that all subsets of this form that contain the states $p_{10}, p_{20}, \ldots, p_{n-10}$ are reachable. Let us refer to these subsets as *target subsets*:

Definition 2. For every permutation $k = (k_1, \ldots, k_{n-1})$ of $(1, \ldots, n-1)$ and for every set $P \subseteq \{p_{ij} \mid 1 \leqslant i \leqslant n-1, 1 \leqslant j \leqslant n-1, j \neq k_i\}$, the subset

$$\mathcal{S}(k,P) = \{q_{00}, q_{1k_1}, q_{2k_2}, \ldots, q_{n-1k_{n-1}}\} \cup \{p_{10}, p_{20}, \ldots, p_{n-10}\} \cup P,$$

is called a target subset.

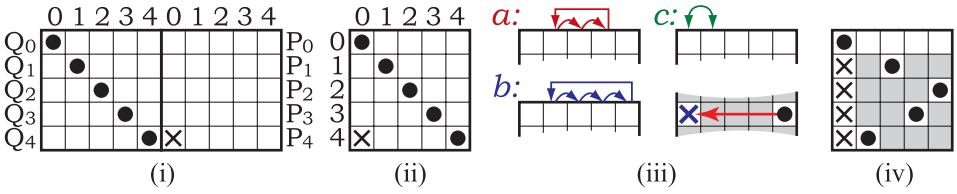


FIGURE 3. Subsets represented as diagrams.

Denote composition of permutations by $k \circ \ell = (k_{\ell_1}, \ldots, k_{\ell_{n-1}})$, and let $I = (1, \ldots, n-1)$ be the identity permutation. Define an application of a permutation k to a subset $P \subseteq \{p_{ij} \mid 1 \leqslant i, j \leqslant n-1\}$ as $kP = \{p_{ik_j} \mid p_{ij} \in P\}$.

Lemma 4. *Every target subset $\mathcal{S}(k, P)$ is reachable from the initial subset via some string in $\{a, b, cc\}^*$.*

There are $(n-1)! \cdot 2^{(n-1)(n-2)}$ target subsets, since $(n-1)!$ is the number of permutations of $n-1$ elements, and, for each permutation, P can be an arbitrary subset of a set of cardinality $(n-1)(n-2)$. The proof of Lemma 4 is given in the rest of this subsection.

Finding an accessing string for any target subset can be regarded as a combinatorial puzzle. Our task is to reach any given target subset–position from the fixed start position, which is essentially the same problem as reaching a fixed end position from an arbitrarily chosen initial position, as in the well-known *fifteen puzzle* and *Rubik's cube*. In accordance to this puzzle analogy, let us represent our subsets in the form of *diagrams*.

Consider the $2n^2$-state NFA in Figure 2, the subsets of which we consider. Following the arrangement of states in that figure, a subset can be simply represented as an $n \times 2n$ matrix of bits, such as the one shown in Figure 3i that corresponds to the initial subset $\{q_{00}, q_{11}, q_{22}, q_{33}, q_{44}, p_{40}\}$ in the case $n = 5$. The elements q_{ij} will always be shown with circles, while crosses are used for p_{ij}.

These diagrams can be much improved, if we notice that, according to Lemma 2, for any subset reachable *via* strings in $\{a, b, c\}^*$, a state q_{ij}'s being in this subset necessarily implies that p_{ij} is not there. In other words, each circle in the left part of the matrix implies an empty square in the corresponding place in the right part. This allows us to overlap the left and the right $n \times n$ submatrices to form $n \times n$ diagrams, such as the one for the initial subset shown in Figure 3ii. This is the form of diagrams we shall use to illustrate all the following constructions.

The rules of our combinatorial puzzle can be stated as follows. Given the initial position shown in Figure 3ii, one can proceed using *moves* of the following three types corresponding to the first three symbols of the alphabet:

- the move a cyclically rotates columns $1, 2, \ldots, n-2$, without touching columns 0 and $n-1$;
- b rotates columns $1, 2, \ldots, n-2, n-1$, leaving column 0 intact;
- c swaps columns 0 and 1, leaving the rest of the columns as they are.

After every move, the circle in the rightmost column $n-1$ sets a cross in the leftmost position in the same row. These rules are shown in Figure 3iii. The task is to find a sequence of moves leading to every position, such as in Figure 3iv, for every permutation of circles (except the fixed circle in the upper left corner, corresponding to q_{00}) and for every combination of crosses and empty squares in the grey area.

Let us now construct a few sequences of moves (that is, strings in $\{a, b, cc\}^*$) that do useful transformations of positions. The solution for the puzzle could then be obtained as a combination of these sequences.

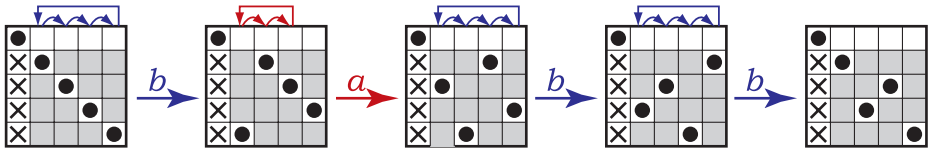
The first type of sequences implements an arbitrary permutation of the columns $1, \ldots, n-1$ without touching the column 0. This can be done using a and b only.

Lemma 5 (Sequence for a permutation)**.** *For every permutation $k = (k_1, \ldots, k_{n-1})$ of $\{1, \ldots, n-1\}$ there exists a string $u \in \{a, b\}^*$, such that every target subset $\mathcal{S}(\ell, P)$ goes to the target subset $\mathcal{S}(k \circ \ell, kP)$ upon reading u.*

Proof. Let us start from the particular case when the given permutation swaps $m-1 \leftrightarrow m$ for some m $(2 \leqslant m \leqslant n-1)$, leaving the other states as they are.

Claim I. *Suppose $k_{m-1} = m$, $k_m = m-1$, and $k_i = i$ for all $i \notin \{m, m-1\}$. Then this permutation is implemented by $u_m = b^{n-m-1}ab^{m-1}$.*

Let us take $n = 5$ and $m = 3$ and see how the string $u_3 = babb$ implements the permutation $2 \leftrightarrow 3$:



The prefix b^{n-m-1} moves the circles to be swapped into columns $n-2$ and $n-1$, then the symbol a swaps these circles, and finally b^{m-1} rotates the columns back to their original order.

To prove Claim I formally, consider the computation of the original DFA A_n on the string u_m starting from each of its states:

- 0 remains in 0, since $\delta(0, a) = \delta(0, b) = 0$;
- each j (with $0 < j < m-1$) goes to $j + n - m - 1 < n - 2$ by b^{n-m-1}, then to $j + n - m$ by a, to $n-1$ by b^{m-j-1}, then to 1 by b, and to j by b^{j-1};
- $m-1$ goes to $n-2$ by b^{n-m-1}, to 1 by a, and then to m by b^{m-1};
- m goes to $n-1$ by b^{n-m-1}, remains in $n-1$ by a, and proceeds to $m-1$ by b^{m-1};
- every $j > m$ goes to $n-1$ by b^{n-1-j}, to 1 by b, and to $j-m$ by b^{j-m-1}, then to $j-m+1$ by a, and finally to j by b^{m-1}.

Let i and i' be the numbers, such that $\ell_i = m-1$ and $\ell_{i'} = m$. Now, for each state of the NFA in the subset

$$\mathcal{S}(\ell, P) = \{q_{00}, q_{1\ell_1}, \ldots, \underline{q_{im-1}}, \ldots, \underline{q_{i'm}}, \ldots, q_{n-1\ell_{n-1}}\} \cup \{p_{10}, \ldots, p_{n-10}\} \cup P,$$

we can tell where it gets by $b^{n-m-1}ab^{m-1}$. Of the two underlined states, q_{im-1} goes to q_{im} and $q_{i'm}$ goes to $\{q_{i'm-1}, p_{i0}\}$. The state q_{00} goes to q_{00}. Each q_{ij} $(0 < j < m-1)$ goes to $\{q_{ij}, p_{i0}\}$, and each q_{ij} $(j > m)$ goes to $\{q_{ij}, p_{i0}\}$ as well. Similarly, p_{i0} goes to p_{i0} (for all i); p_{im-1} goes to p_{im}; $p_{i'm}$ goes to $p_{i'm-1}$; p_{ij} $(j \neq m, m-1)$ remains in p_{ij}. Assembling these states together, we obtain the subset reached from $\mathcal{S}(\ell, P)$ *via* $b^{n-m-1}ab^{m-1}$, and it is easy to check that it is exactly

$$\{q_{00}, q_{1\ell_1}, \ldots, \underline{q_{im}}, \ldots, \underline{q_{i'm-1}}, \ldots, q_{n-1\ell_{n-1}}\} \cup \{p_{10}, \ldots, p_{n-10}\} \cup \{p_{ik_j} \mid p_{ij} \in P\} = \mathcal{S}(k \circ \ell, kP).$$

This completes the proof of Claim I.

Claim II. *Let $k = (k_1, \ldots, k_{n-1})$ and $k' = (k'_1, \ldots, k'_{n-1})$ be permutations, and assume that a string w implements k and a string w' implements k' as defined in the statement of the lemma. Then $k' \circ k$ is implemented by the string ww'.*

By assumption, the automaton moves from a target subset $\mathcal{S}(\ell, P)$ to $\mathcal{S}(k \circ \ell, kP)$ by w, and from there to $\mathcal{S}(k' \circ (k \circ \ell), k'(kP))$ by w'. Composition of permutations is associative, hence $k' \circ (k \circ \ell) = (k' \circ k) \circ \ell$. It is easy to see that $k'(kP) = \{p_{ik'_{k_j}} \mid p_{ij} \in P\} = (k' \circ k)P$. Therefore, $\mathcal{S}(\ell, P)$ goes to $\mathcal{S}((k' \circ k) \circ \ell, (k' \circ k)P)$ by ww', which proves Claim II.

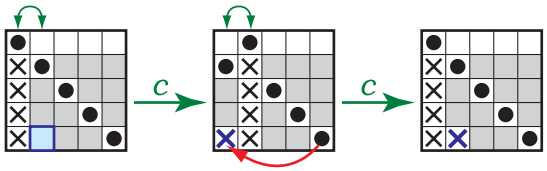
Let us state the following well-known fact without a proof:

Claim III. *Every permutation can be represented as a composition of permutations $k^{(m)}$ of the form $m-1 \leftrightarrow m$.*

Now the lemma can be proved in the case of an arbitrary permutation k. Following Claim III, let $m_1, \ldots, m_t$ $(t \geqslant 0,\ 2 \leqslant m_i \leqslant n-1)$ be the numbers, such that $k = k^{(m_t)} \circ \ldots \circ k^{(m_1)}$. According to Claim I, each $k^{(m)}$ is implemented by $u_m = b^{n-m-1}ab^{m-1}$, and therefore, by Claim II, the concatenation $w = u_{m_1} \ldots u_{m_\ell}$ implements k. This completes the proof. □

Lemma 6 (Sequence for 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 target subset $\mathcal{S}(I, P)$ goes to the target subset $\mathcal{S}(I, P \cup \{p_{ij}\})$ upon reading w_{ij}.*

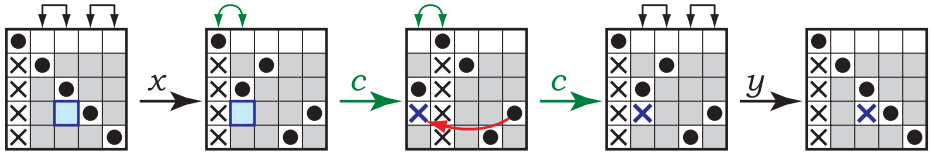
Proof. First consider a special case of $i = n-1$, $j = 1$. This square can be filled by doing the move c twice (that is, the bit p_{n-11} can be set via the string $w_{n-1,1} = cc$) as follows:



The first c exchanges columns 0 and 1, moving the empty square from column 1 to column 0, where it is immediately filled, because the circle in its line is in position $n-1$, or, in other words, q_{n-1n-1} is in $\mathcal{S}(I,P)$. The second c restores the order of columns, moving the filled square back to its original position in column 1. No other squares are altered.

The above construction depends upon the particular configuration in row n − 1: the bit to be set is in column 1 and the circle is in the last column. This configuration can be reproduced for any other bit by permuting columns $1,\ldots,n-1$. Let p_{ij} $(1 \leqslant i \leqslant n-1,\ 1 \leqslant j \leqslant n-1,\ i \neq j)$ be an arbitrary bit that needs to be set. Consider a permutation $k=(k_1,\ldots,k_{n-1})$, such that $k_i = n-1$ and $k_j = 1$ (that is, i is mapped to $n-1$ and j is mapped to 1), while the rest of the elements are set arbitrarily. Denote the inverse of k by k^{-1}. Let x be the string given by Lemma 5 for k and let y be the string given by Lemma 5 for k^{-1}. Define w_{ij} as $xccy$.

Let us see how such a string sets the bit p_{32} in the case $n=5$. The first permutation $\{3 \leftrightarrow 4, 2 \leftrightarrow 1\}$ moves the empty square and the circle in line 3 to the same position as in the above example. Then cc has the same effect as in that example. Afterwards, an application of the inverse permutation $\{3 \leftrightarrow 4, 2 \leftrightarrow 1\}$ (which in this case coincides with the first permutation) restores the order of the columns.



Now let us formally prove that $\mathcal{S}(I,P)$ goes to $\mathcal{S}(I,P\cup\{p_{ij}\})$ by $xccy$. By Lemma 5, $\mathcal{S}(I,P)$ goes to

$$\mathcal{S}(k,kP)=\{q_{00},q_{1k_1},\ldots,q_{in-1},\ldots,q_{j1},\ldots,q_{n-1k_{n-1}}\}\cup\{p_{10},p_{20},\ldots,p_{n-10}\}\cup \cup\{p_{sk_t}\mid p_{st}\in P\}$$

by the string x. Then, by the first c $(0 \leftrightarrow 1)$, we proceed to

$$\{q_{01},q_{1k_1},\ldots,q_{in-1},\ldots,q_{j0},\ldots,q_{n-1k_{n-1}}\}\cup\{p_{11},p_{21},\ldots,p_{n-11}\}\cup \cup\{p_{s\delta(k_t,c)}\mid p_{st}\in P\}\cup\{p_{i0}\},$$

and next, by the second c $(0 \leftrightarrow 1)$, to

$$\{q_{00},q_{1k_1},\ldots,q_{in-1},\ldots,q_{j1},\ldots,q_{n-1k_{n-1}}\}\cup\{p_{10},p_{20},\ldots,p_{n-10}\}\cup \cup\{\underbrace{p_{s\delta(\delta(k_t,c),c)}}_{p_{sk_t}}\mid p_{st}\in P\}\cup\{p_{i1}\}=\mathcal{S}(k,kP\cup\{p_{i1}\}).$$

The last string y reverts the permutation: according to Lemma 5, $\mathcal{S}(k,kP\cup\{p_{i1}\})$ goes to

$$\mathcal{S}(k^{-1}\circ k,k^{-1}kP\cup k^{-1}\{p_{i1}\})=\mathcal{S}(I,P\cup\{p_{ij}\}),$$

and this completes the proof of the lemma. □

Now we have sufficient tools to prove the reachability of $(n-1)! \cdot 2^{(n-1)(n-2)}$ subsets.

Proof of Lemma 4. Starting from the initial subset

$$\{q_{00}, q_{11}, q_{22}, \ldots, q_{n-1n-1}\} \cup \{p_{n-10}\},$$

let us first reach the target subset

$$\mathcal{S}(I, \varnothing) = \{q_{00}, q_{11}, q_{22}, \ldots, q_{n-1n-1}\} \cup \{p_{10}, p_{20}, \ldots, p_{n-10}\},$$

which can be done via the string b^{n-1}.

Let k^{-1} be the inverse of the permutation $k = (k_1, \ldots, k_{n-1})$, let $P' = k^{-1}P$. We now proceed with setting the bits in P' one by one. Consider every $p_{ij} \in P'$. Then $p_{ik_j} \in P$ and, since $\mathcal{S}(k, P)$ is a target subset, $k_j \neq k_i$. Hence $j \neq i$, which makes Lemma 6 applicable to all target subsets $\mathcal{S}(I, \widetilde{P}')$ with $\widetilde{P}' \subseteq P'$. Let w_{ij} be the string given by Lemma 6 for each p_{ij}. *Via* the concatenation of all such w_{ij} we reach the subset

$$\mathcal{S}(I, P') = \{q_{00}, q_{11}, q_{22}, \ldots, q_{n-1n-1}\} \cup \{p_{10}, p_{20}, \ldots, p_{n-10}\} \cup k^{-1}P.$$

Finally, let x be the string given by Lemma 5 for k. By x, we reach

$$\mathcal{S}(k, kP') = \mathcal{S}(k, P) = \{q_{00}, q_{1k_1}, q_{2k_2}, \ldots, q_{n-1k_{n-1}}\} \cup \{p_{10}, p_{20}, \ldots, p_{n-10}\} \cup P,$$

and the proof of the lemma is complete. □

3.3. Inequivalence of subsets

We now prove pairwise inequivalence of all target subsets, which were shown to be reachable in Lemma 4. 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 p_{ij}. The same will be done for the states q_{ij} $(1 \leqslant i, j \leqslant n-1)$. Then, the inequivalence of the subsets will follow immediately.

Lemma 7. *For every i and j $(1 \leqslant i \leqslant n-1, 1 \leqslant j \leqslant n-1)$, the string $b^{n-1-j}db^{i-1}$ is accepted by the NFA from 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. 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 accepting state p_{ii} by the string b^{i-1}. This case is illustrated in Figure 4a.

Let us see how the NFA rejects this string from the rest of the states mentioned in the lemma. As shown in Figure 4b, every state $p_{i\ell}$ $(0 \leqslant \ell \leqslant n-1)$ with $\ell \neq j$ goes to a state $p_{i\ell'}$ with $\ell' \neq n-1$ by the string b^{n-1-j}, then to state p_{i0} by d, and remains in non-accepting state p_{i0} upon reading b^{i-1}. Similarly, for every $1 \leqslant k \leqslant n-1$ and $0 \leqslant \ell \leqslant n-1$, where $\ell \neq j$, state $p_{k\ell}$ goes to p_{k0} by $b^{n-1-j}db^{i-1}$.

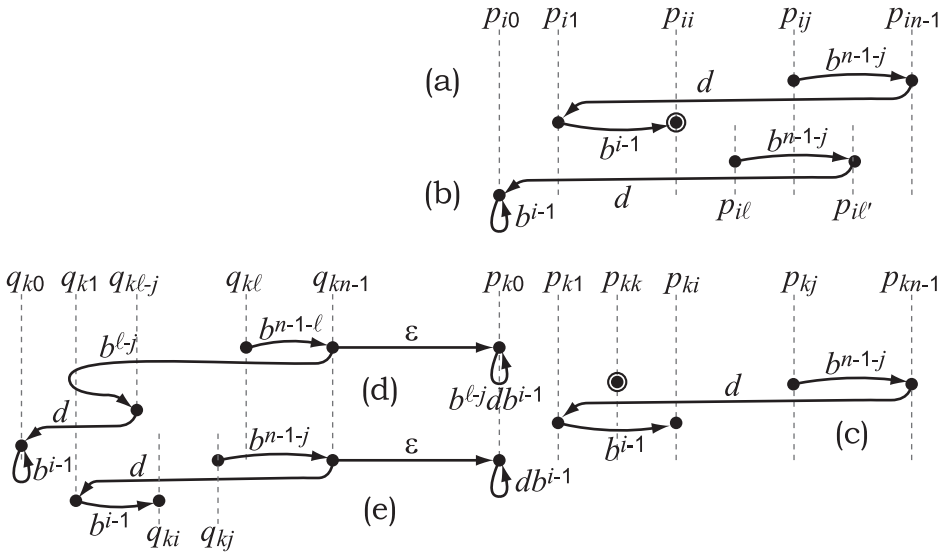


FIGURE 4. Acceptance of $b^{n-1-j}db^{i-1}$ from p_{ij} and only from p_{ij}.

The case of a state p_{kj}, where $1 \leqslant k \leqslant n-1$ and $k \neq i$, is illustrated in Figure 4c: similarly to Figure 4a, p_{kj} goes to p_{ki} by $b^{n-1-j}db^{i-1}$, but unlike state p_{ii}, p_{ki} is not an accepting state.

Let $1 \leqslant k \leqslant n-1$. Every state $q_{k\ell}$, where $1 \leqslant j < \ell \leqslant n-1$, goes to state q_{kn-1} by the string $b^{n-1-\ell}$; from this point, one can either go to p_{k0} by the epsilon transition and remain there until the remainder of the string is consumed, or one can proceed to $q_{k\ell-j}$ by $b^{\ell-j}$ (consider that $\ell - j < n-1$) then to q_{k0} by d, and read the rest of the string (b^{i-1}) while remaining in this state. This is shown in Figure 4d, and since neither p_{k0} nor q_{k0} is accepting, the input is rejected. The case of states $q_{k\ell}$, where $1 \leqslant \ell < j$, is similar to that, except that p_{k0} cannot be reached: state $q_{k\ell}$ goes to $q_{kn-1-j+\ell}$ by b^{n-1-j} (note that $n-1-j+\ell < n-1$), then to q_{k0} by d and remains in non-accepting state q_{k0} upon reading b^{i-1}. For $\ell = j$, the computation proceeds as illustrated in Figure 4e: q_{kj} goes to q_{kn-1} by b^{n-1-j} and then either proceeds by an epsilon transition to p_{k0}, where the rest of the string is consumed, or by d to q_{k1}, and then to q_{ki} by b^{i-1} (and, if $i = n-1$, then also to p_{k0}). Again, neither q_{ki} nor p_{k0} is accepting.

Finally, upon reading the string $b^{n-1-j}db^{i-1}$, state q_{00} remains in q_{00}. □

Lemma 8. *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 state q_{ij}, but is not accepted from any other state in*

$$\{q_{00}\} \cup \{q_{ij} \mid 1 \leqslant i \leqslant n-1, 1 \leqslant j \leqslant n-1\} \cup \{p_{ij} \mid 1 \leqslant i \leqslant n-1, 0 \leqslant j \leqslant n-1\}.$$

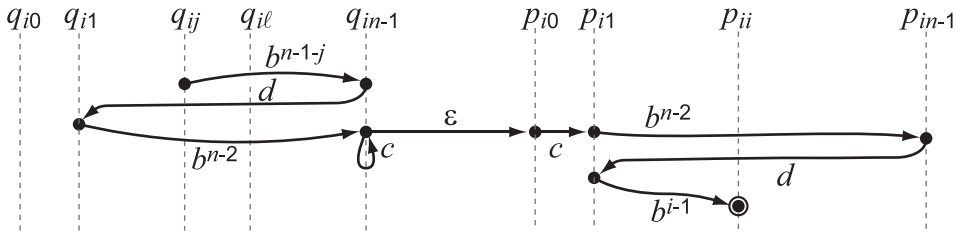


FIGURE 5. Acceptance of $b^{n-1-j}db^{n-2}ccb^{n-2}db^{i-1}$ from q_{ij}.

Proof. Let us first show how one can reach an accepting state from state q_{ij} by the given string. As illustrated in Figure 5, one can go from q_{ij} to q_{in-1} by b^{n-1-j}, then to q_{i1} by d and again to q_{in-1} by b^{n-2}. Then one can choose to remain in q_{in-1} by the first c, then move to p_{i0} using the ε-transition and proceed to p_{i1} by the second c. The rest of the computation is deterministic: state p_{i1} goes to state p_{in-1} by the string b^{n-2}, then to p_{i1} by d and, finally, to accepting state p_{ii} by b^{i-1}.

Consider any state q_{kj}, where $k > 0$ and $k \neq i$, and let us prove that each computation from this state by the string $b^{n-1-j}db^{n-2}ccb^{n-2}db^{i-1}$ is rejecting. Since all transitions by d in the DFA A_n go either to state 0 or to state 1, the NFA, after reading the string $b^{n-1-j}db^{n-2}ccb^{n-2}d$, must be in one of states $\{q_{k0}, q_{k1}, p_{k0}, p_{k1}\}$. Then, after reading b^{i-1}, it proceeds to one of $\{q_{k0}, q_{ki}, p_{k0}, p_{ki}\}$. None of these states is accepting for $k \neq i$.

Now consider any state $q_{k\ell}$, where $1 \leqslant k \leqslant n-1$ and $\ell \neq j$. As demonstrated in Figure 6a, from $q_{k\ell}$ one may go to q_{k0} or to p_{k0} by the string $b^{n-1-j}d$ (p_{k0} is possible when $\ell > j$), and each of these two states goes to itself by the remaining suffix $b^{n-2}ccb^{n-2}db^{i-1}$. Neither of these states is accepting.

The computation from every state $p_{k\ell}$ ($1 \leqslant k \leqslant n-1$, $0 \leqslant \ell \leqslant n-1$) is deterministic. By $b^{n-1-j}d$, state $p_{k\ell}$ may go either to p_{k1} (if $\ell = j$, see Fig. 6b) or to p_{k0} (if $\ell \neq j$, as in Fig. 6c). Then, in the first case, p_{k1} goes to state p_{kn-1} by the string $b^{n-2}cc$, then to p_{kn-2} by b^{n-2}, and finally to p_{k0} by db^{i-1}; in the second case, p_{k0} goes to itself by $b^{n-2}ccb^{n-2}db^{i-1}$.

Finally, state q_{00} goes to itself upon reading $b^{n-1-j}db^{n-2}ccb^{n-2}db^{i-1}$. □

It can now be established that all $(n-1)! \cdot 2^{(n-1)(n-2)}$ subsets shown to be reachable in Lemma 4 are pairwise inequivalent.

Lemma 9. *Let $\mathcal{S}(k, P)$ and $\mathcal{S}(\ell, P')$ be two distinct target subsets. Then the languages recognized by the NFA from $\mathcal{S}(k, P)$ and from $\mathcal{S}(\ell, P')$ are distinct.*

Proof. Recalling the definition of target subsets, we have

$$\mathcal{S}(k, P) = \{q_{00}, q_{1k_1}, q_{2k_2}, \ldots, q_{n-1k_{n-1}}\} \cup \{p_{10}, p_{20}, \ldots, p_{n-10}\} \cup P \quad \text{and}$$
$$\mathcal{S}(\ell, P') = \{q_{00}, q_{1\ell_1}, q_{2\ell_2}, \ldots, q_{n-1\ell_{n-1}}\} \cup \{p_{10}, p_{20}, \ldots, p_{n-10}\} \cup P',$$

where $k = (k_1, \ldots, k_{n-1})$ and $\ell = (\ell_1, \ldots, \ell_{n-1})$ are permutations of $(1, \ldots, n-1)$,

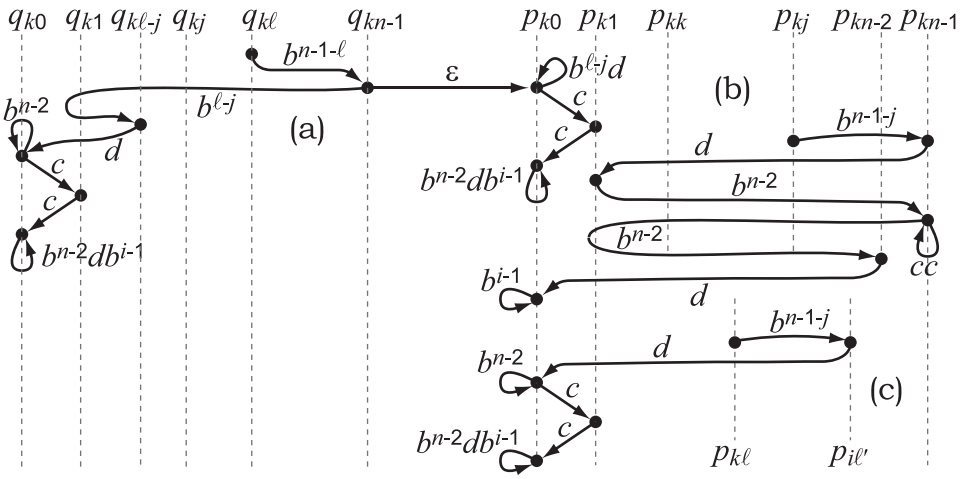


FIGURE 6. Rejection of $b^{n-1-j}db^{n-2}ccb^{n-2}db^{i-1}$ from $q_{k\ell}$, p_{kj}, $p_{k\ell}$.

$P \subseteq \{p_{ij} \mid 1 \leqslant i \leqslant n-1,\ 1 \leqslant j \leqslant n-1,\ j \neq k_i\}$ and $P' \subseteq \{p_{ij} \mid 1 \leqslant i \leqslant n-1,\ 1 \leqslant j \leqslant n-1,\ j \neq \ell_i\}$. Since $\mathcal{S}(k,P) \neq \mathcal{S}(\ell,P')$, we have either $k \neq \ell$, or $k = \ell$ and $P \neq P'$.

Suppose $(k_1,\ldots,k_{n-1}) \neq (\ell_1,\ldots,\ell_{n-1})$ and let t be a number, such that $k_t \neq \ell_t$. State q_{tk_t} is in $\mathcal{S}(k,P) \setminus \mathcal{S}(\ell,P')$, and therefore, by Lemma 8, the string $b^{n-1-k_t}db^{n-2}ccb^{n-2}db^{t-1}$ is accepted from $\mathcal{S}(k,P)$ and is not accepted from $\mathcal{S}(\ell,P')$.

If $k = \ell$, then $P \neq P'$, that is, there exists a state p_{ij} with $i > 0$ and $j > 0$, such that $p_{ij} \in P \,\Delta\, P'$. Without loss of generality, assume $p_{ij} \in P \setminus P'$. Then $p_{ij} \in \mathcal{S}(k,P) \setminus \mathcal{S}(\ell,P')$, and, according to Lemma 7, the string $b^{n-1-j}db^{i-1}$ is accepted from exactly one of these subsets. □

Hence we have shown the following result.

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

Let us now apply Stirling's approximation of the factorial to estimate this function (all logarithms are base two):

$$\begin{aligned}(n-1)! &\geqslant \sqrt{2\pi}(n-1)^{n-\frac{1}{2}}e^{-n+1} \geqslant (n-1)^{n-1}e^{-n} = 2^{(n-1)\log(n-1)-n\log e}\\ &= 2^{n\log n - n\log e - \log(n-1) + n\log(1-\frac{1}{n})} \geqslant 2^{n\log n - n\log e - 3\log n}.\end{aligned}$$

Therefore, we have the following lower bound on the state complexity of cyclic shift for a k-letter alphabet ($k \geqslant 4$):

$$f_k(n) \geqslant (n-1)! \cdot 2^{(n-1)(n-2)} \geqslant 2^{n^2 + n\log n - (3+\log e)n - 3\log n}.$$

On the other hand, the upper bound for f_k given in Theorem 1 can be transformed to the same exponential form:

$$f_k(n) \leqslant (n2^n - 2^{n-1})^n = n^n 2^{n^2} \left(1 - \frac{1}{2n}\right)^n \leqslant n^n 2^{n^2} = 2^{n^2 + n\log n},$$

where the second inequality relies upon an easily established fact that $(1-\frac{1}{2n})^n < 1$ for all $n \geqslant 1$. Altogether we obtain the following bounds on the state complexity of the cyclic shift, which hold for every alphabet of at least four letters:

Corollary 1. *For every $k \geqslant 4$,*

$$2^{n^2+n\log n-(3+\log e)n-3\log n} \leqslant f_k(n) \leqslant 2^{n^2+n\log n}$$

holds for all $n \geqslant 3$, and therefore $f_k(n) = 2^{n^2+n\log n-O(n)}$.

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 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 10. *Let $\Sigma_k = \{c_1, \ldots, c_k\}$ and let $h : \Sigma_k^* \to \{a,b\}^*$ be a homomorphism defined by $h(c_i) = a^{i-1}b$ (for all $1 \leqslant i \leqslant k-1$) and $h(c_k) = a^{k-1}$. Then, for every language L over Σ_k, $h^{-1}(\textsc{shift}(h(L))) = \textsc{shift}(L)$.*

Let us note that not every code satisfies the statement of the lemma. For instance, if $h(c_1) = aa, h(c_2) = ab, h(c_3) = ba$, then $h^{-1}(\textsc{shift}(h(\{c_1c_2\}))) = h^{-1}(\textsc{shift}(\{aaab\})) = h^{-1}(\{aaab, baaa, abaa, aaba\}) = \{c_1c_2, c_3c_1, c_2c_1, c_1c_3\}$. Lemma 10 is applicable only to the given particular homomorphism.

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 $c_{i_1} \cdots c_{i_m} \in h^{-1}(\textsc{shift}(h(L)))$, and assume that at least one of c_{i_j} is not c_k (the case of $c_{i_j} = c_k$ for all j is trivial, since $h(c_k^m) \in a^*$). Then $h(c_{i_1} \cdots c_{i_m}) \in \textsc{shift}(h(L))$, and there exists a factorization $h(c_{i_1} \cdots c_{i_m}) = xy$ (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: $x = h(c_{i_1} \cdots c_{i_\ell})$, $y = h(c_{i_{\ell+1}} \cdots c_{i_m})$. Then $h(c_{i_{\ell+1}} \cdots c_{i_m} c_{i_1} \cdots c_{i_\ell}) \in h(L)$, and hence, since h is a code, $c_{i_{\ell+1}} \cdots c_{i_m} c_{i_1} \cdots c_{i_\ell} \in L$. Therefore, $c_{i_1} \cdots c_{i_\ell} c_{i_{\ell+1}} \cdots c_{i_m} \in \textsc{shift}(L)$.

Now suppose the factorization splits some ℓ-th symbol in two, that is, $x = h(c_{i_1} \cdots c_{i_{\ell-1}})z'$ and $y = z''h(c_{i_{\ell+1}} \cdots c_{i_m})$, where $h(c_{i_\ell}) = z'z''$ for some $z', z'' \in \{a,b\}^+$. Note that $z' = a^t$ $(1 \leqslant t < k-1)$, since all proper prefixes of the images of symbols under h are of this form. So we have $z''h(c_{i_{\ell+1}} \cdots c_{i_m} c_{i_1} \cdots c_{i_{\ell-1}})a^t \in h(L)$, and this 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_k and then with a^t. Since no string in $h(\Sigma_k^*)$ can be of this form, this supposedly problematic case is in fact impossible. □

Next, we use this encoding to compute the cyclic shift of a given DFA over an arbitrary alphabet. The given DFA is first subjected to a homomorphic encoding to the alphabet $\{a,b\}$, then a cyclic shift over $\{a,b\}$ is computed, and finally an inverse homomorphism is used to get back to the original alphabet. According to Lemma 10, the resulting DFA will recognize the cyclic shift of the original DFA.

We present this construction in detail, taking note of the number of states in each automaton constructed. For the homomorphism and for the inverse homomorphism, we construct the states directly and obtain their exact number; when the cyclic shift over $\{a,b\}$ is taken, we use the corresponding state complexity function f_2 as an upper bound for the number of states. Thus the state complexity of cyclic shift over an k-letter alphabet is expressed through f_2.

Lemma 11. *For every n-state DFA A over an k-letter alphabet there exists an $f_2(kn-n)$-state DFA for the language* $\textsc{shift}(L(A))$.

Proof. Let $A = (Q, \{c_1, \ldots, c_k\}, \delta, q_0, F)$ and consider the homomorphism $h : \{c_1, \ldots, c_k\}^* \to \{a,b\}^*$ defined in Lemma 10. Let us construct a new DFA $B = (Q', \{a,b\}, \delta', q_0', F')$ with the set of states $Q' = Q \times \{1, \ldots, k-1\}$, of which $q_0' = (q_0, 1)$ is the initial state and $F = \{(q,1) \mid q \in F\}$ is the set of accepting states, and with the transition function $\delta' : Q' \times \{a,b\} \to Q'$, where, for every $q \in Q$,

$$\begin{aligned} \delta'\big((q,i),a\big) &= (q, i+1) && (1 \leqslant i < k-1), \\ \delta'\big((q,i),b\big) &= \big(\delta(q,c_i),1\big) && (1 \leqslant i \leqslant k-1), \\ \delta'\big((q,k-1),a\big) &= \big(\delta(q,c_k),1\big). \end{aligned}$$

It is easy to prove that this $(k-1)n$-state automaton recognizes the language $h(L(A))$.

Consider the language $\textsc{shift}(L(B)) \subseteq \{a,b\}^*$. By the definition of f_2, there exists a DFA $C = (\widehat{Q}, \{a,b\}, \widehat{\delta}, \widehat{q_0}, \widehat{F})$ that recognizes this language and has at most $f_2(kn-n)$ states. Construct a new DFA $D = (\widehat{Q}, \{c_1, \ldots, c_k\}, \widehat{\delta'}, \widehat{q_0}, \widehat{F})$ with the same set of states, the same initial state and the same accepting states, in which the transition function is defined as $\widehat{\delta'}(q, c_i) = \widehat{\delta}(q, h(c_i))$. Then $L(D) = h^{-1}(L(C)) = h^{-1}(\textsc{shift}(h(L(A))))$, which equals $\textsc{shift}(L(A))$ by Lemma 10. □

Since, under the conditions of Lemma 11, the language $\textsc{shift}(L(A))$ may require up to $f_k(n)$ states, the following relation between the state complexity of cyclic shift over an k-letter alphabet and over a binary alphabet is established.

Theorem 3. *For all $k \geqslant 3$ and $n \geqslant 1$, $f_2(kn-n) \geqslant f_k(n)$.*

For $k=4$, we obtain $f_2(3n) \geqslant f_4(n)$. Now let us use the lower bound expression from Corollary 1, simplifying it as follows:

$$f_4(n) \geqslant 2^{n^2+n\log n-(3+\log e)n-3\log n} \geqslant 2^{n^2} \quad \text{(for all } n \geqslant 31\text{)}.$$

From this we can infer the following lower bound for f_2:

$$f_2(n) \geqslant f_4\left(\frac{n}{3}\right) \geqslant 2^{n^2/9} \quad \text{(for all } n \geqslant 93\text{)}.$$

Using the upper bound from Corollary 1, we can estimate the state complexity of the cyclic shift for binary and ternary alphabets.

Corollary 2. *The following bounds hold:*

$$2^{n^2/9} \leqslant f_2(n) \leqslant f_3(n) \leqslant 2^{n^2+n\log n},$$

where $n \geqslant 93$. Therefore, $f_2(n) = 2^{\Theta(n^2)}$ and $f_3(n) = 2^{\Theta(n^2)}$.

4. Nondeterministic state complexity

We now turn our attention to the nondeterministic state complexity of cyclic shift. Using the representation in Lemma 1 we obtain upper bounds $2n^2$ for NFAs with multiple initial states and $2n^2+1$ for NFAs with a single initial state. The aim of this section is to show that these bounds are tight for all $n \geqslant 2$ and already for a binary alphabet. Since the cyclic shift of every unary and of every 1-state NFA language is the same language, our analysis covers all cases.

To obtain lower bounds on the nondeterministic state complexity of cyclic shift we use the *fooling-set* lower-bound technique known from communication complexity theory [1,13] which has often been used in the study of descriptional complexity of regular languages [2–4,9,12]. After defining a fooling set, we give the lemma due to Birget [2] describing the fooling-set lower-bound technique. For the sake of completeness, we recall its proof here.

Definition 3. A set of pairs of strings $\{(x_i, y_i) \mid i = 1, 2, \ldots, n\}$ is said to be a *fooling set* for a language L if for every i and j in $\{1, 2, \ldots, n\}$,

(I) the string $x_i y_i$ is in the language L; and
(II) if $i \neq j$, then at least one of the strings $x_i y_j$ and $x_j y_i$ is not in L.

Lemma 12 (Birget [2])**.** (Lower-bound argument for nondeterministic state complexity). *Let a set of pairs of strings $\{(x_i, y_i) \mid i = 1, 2, \ldots, n\}$ be a fooling set for a regular language L. Then every NFA for the language L needs at least n states.*

Proof. Let $M = (Q, \Sigma, \delta, Q_0, F)$ be an NFA accepting the language L. Consider any i-th pair (x_i, y_i). Since $x_i y_i \in L$, there is a state q_i in Q_0 and a state p_i in Q

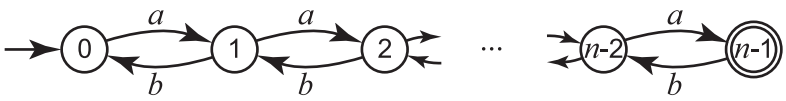


FIGURE 7. An NFA that requires $2n^2 + 1$ states for its cyclic shift.

such that $p_i \in \delta(q_i, x_i)$ and $\delta(p_i, y_i) \cap F \neq \varnothing$. In other words, p_i is a state in an accepting computation of M on $x_i y_i$ that is reached after reading x_i.

Assume that a fixed choice of p_i has been made for every i in $\{1, 2, \ldots, n\}$. We prove that the states $p_1, p_2, \ldots, p_n$ must be pairwise different. Suppose by contradiction that $p_i = p_j$ for some i and j such that $i \neq j$. Then the NFA M accepts both strings $x_i y_j$ and $x_j y_i$. However, this contradicts the assumption that the set $\{(x_i, y_i) \mid i = 1, 2, \ldots, n\}$ is a fooling set for the language L. Hence the NFA M has at least n states. □

The next lemma shows that the nondeterministic state complexity of cyclic shift is at least $2n^2$.

Lemma 13. *For every $n \geqslant 2$, there exists a binary NFA M of n states such that every NFA for the cyclic shift of the language $L(M)$ needs at least $2n^2$ states.*

Proof. Let $n \geqslant 2$ and let $\Sigma = \{a, b\}$. Define an n-state NFA $M = (Q, \Sigma, \delta, q_0, F)$, where $Q = \{0, 1, \ldots, n-1\}$, $q_0 = 0$, $F = \{n-1\}$, and for each q in Q,

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

The NFA M is shown in Figure 7. Note that M is partially deterministic. This automaton computes the difference between the number of a's and the number of b's in the input strings. The automaton M recognizes the language of all strings w over Σ, such that this difference is between 0 and $n-1$ for every prefix of w and it reaches the value $n-1$ for the entire w. Denote this language by L.

To prove the lemma we are going to describe a fooling set for the language $\text{SHIFT}(L)$ of size $2n^2$.

For every $k = 0, 1, \ldots, n-1$, consider the following sets of pairs of strings:

$$\mathcal{A}_k = \{(b^k a^{n-1} b^{n-1} a^i, b^i a^{2n-2} b^{n-1-k}) \mid i = 0, 1, \ldots, n-1\}$$

and

$$\mathcal{B}_k = \{(b^k a^{n-1} b^{n-1} a^{2n-2} b^{n-j}, a^{n-j} b^{n-1-k}) \mid j = 1, 2, \ldots, n\}.$$

Let $\mathcal{F} = \mathcal{A}_0 \cup \mathcal{B}_0 \cup \mathcal{A}_1 \cup \mathcal{B}_1 \cup \cdots \cup \mathcal{A}_{n-1} \cup \mathcal{B}_{n-1}$. We will prove that the set $\mathcal{F}$ is a fooling set for the language $\text{SHIFT}(L)$. We need to show that (I) and (II) in Definition 3 hold.

To prove (3), note that for all $k = 0, 1, \ldots, n-1$, all $i = 0, 1, \ldots, n-1$, and all $j = 1, 2, \ldots, n$, the strings

$$b^k a^{n-1} b^{n-1} a^i b^i a^{2n-2} b^{n-1-k} \text{ and } b^k a^{n-1} b^{n-1} a^{2n-2} b^{n-j} a^{n-j} b^{n-1-k}$$

are in the language $\text{SHIFT}(L)$ since their cyclic shifts

$$a^{n-1} b^{n-1-k} b^k a^{n-1} b^{n-1} a^i b^i a^{n-1} \text{ and } a^{n-1} b^{n-j} a^{n-j} b^{n-1-k} b^k a^{n-1} b^{n-1} a^{n-1}$$

are accepted by the NFA M and so are in the language L.

To prove (3), we have four cases to consider:

(i) Let $(b^k a^{n-1} b^{n-1} a^i, b^i a^{2n-2} b^{n-1-k})$ and $(b^k a^{n-1} b^{n-1} a^j, b^j a^{2n-2} b^{n-1-k})$, where $0 \leqslant i < j \leqslant n-1$, be two different pairs in $\mathcal{A}_k$. Then the string

$$b^k a^{n-1} b^{n-1} a^i b^j a^{2n-2} b^{n-1-k}$$

is not in $\text{SHIFT}(L)$ because the only cyclic shift of this string that could be in L is $a^{n-1} b^{n-1-k} b^k a^{n-1} b^{n-1} a^i b^j a^{n-1}$, which is not accepted by the NFA M since $i < j$.

(ii) Let $(b^k a^{n-1} b^{n-1} a^{2n-2} b^{n-i}, a^{n-i} b^{n-1-k})$ and $(b^k a^{n-1} b^{n-1} a^{2n-2} b^{n-j}, a^{n-j} b^{n-1-k})$, where $1 \leqslant i < j \leqslant n$, be two different pairs in $\mathcal{B}_k$. Then the string

$$b^k a^{n-1} b^{n-1} a^{2n-2} b^{n-j} a^{n-i} b^{n-1-k}$$

is not in $\text{SHIFT}(L)$ because the only its cyclic shift that could be in L is $a^{n-1} b^{n-j} a^{n-i} b^{n-1-k} b^k a^{n-1} b^{n-1} a^{n-1}$, which is not accepted by the NFA M since $n - j < n - i$.

(iii) Let $(b^k a^{n-1} b^{n-1} a^i, b^i a^{2n-2} b^{n-1-k})$ be a pair in $\mathcal{A}_k$ and $(b^k a^{n-1} b^{n-1} a^{2n-2} b^{n-j}, a^{n-j} b^{n-1-k})$ a pair in $\mathcal{B}_k$. Then the string

$$b^k a^{n-1} b^{n-1} a^{2n-2} b^{n-j} b^i a^{2n-2} b^{n-1-k}$$

is not in $\text{SHIFT}(L)$ since each of its cyclic shifts contains the string a^{2n-2} as a substring.

(iv) Let $(b^k u, v b^{n-1-k})$ be a pair in $\mathcal{A}_k \cup \mathcal{B}_k$ and $(b^\ell x, y b^{n-1-\ell})$ a pair in $\mathcal{A}_\ell \cup \mathcal{B}_\ell$, where $0 \leqslant k < \ell \leqslant n-1$. Then the string

$$b^\ell x v b^{n-1-k}$$

is not in $\text{SHIFT}(L)$, since each of its cyclic shifts not ending with b contains the string $b^{n-1+\ell-k}$, where $n - 1 + \ell - k > n - 1$, as a substring.

Hence the set $\mathcal{F}$ is a fooling set for the language $\text{SHIFT}(L)$ of size $2n^2$. By Lemma 12, every NFA for the language $\text{SHIFT}(L)$ needs at least $2n^2$ states. □

In the following lemma we consider NFAs with a single initial state and show that in this case one more state is necessary.

Lemma 14. *For every $n \geqslant 2$, there exists a binary NFA M of n states such that every NFA with a single initial state for the cyclic shift of the language $L(M)$ needs at least $2n^2 + 1$ states.*

Proof. Let M be the NFA described in the proof of Lemma 13 and let L be the language accepted by this NFA. Let N be any NFA with a single initial state for the language $\textsc{shift}(L)$ and let q_0 be its initial state. Recall that the set $\mathcal{F}$ containing pairs

$$(b^k a^{n-1} b^{n-1} a^i, b^i a^{2n-2} b^{n-1-k}) \text{ and } (b^k a^{n-1} b^{n-1} a^{2n-2} b^{n-j}, a^{n-j} b^{n-1-k}),$$

where $k = 0, 1, \ldots, n-1$, $i = 0, 1, \ldots, n-1$, and $j = 1, 2, \ldots, n$, is a fooling set for the language $\textsc{shift}(L)$.

Let q_{ki} $(0 \leqslant k \leqslant n-1, 0 \leqslant i \leqslant n-1)$ be the state in an accepting computation of the NFA N on the string $b^k a^{n-1} b^{n-1} a^i b^i a^{2n-2} b^{n-1-k}$ that is reached after reading $b^k a^{n-1} b^{n-1} a^i$. Similarly, let p_{kj} $(0 \leqslant k \leqslant n-1, 1 \leqslant j \leqslant n)$ be the state in an accepting computation of the NFA N on the string $b^k a^{n-1} b^{n-1} a^{2n-2} b^{n-j} a^{n-j} b^{n-1-k}$ that is reached after reading $b^k a^{n-1} b^{n-1} a^{2n-2} b^{n-j}$. Since $\mathcal{F}$ is a fooling set for $\textsc{shift}(L)$, the states q_{ki} and p_{kj} must be pairwise different. Let us prove that the initial state q_0 must be different from each of these states.

We first show that the initial state q_0 is different from each state q_{ki} and p_{kj} with $k > 0$. Suppose that $q_0 = q_{ki}$ for some $k > 0$ and some i. Note that the string $a^{2n-2} b^{n-1}$ is in $\textsc{shift}(L)$ and so must be accepted by the NFA N from the initial state q_{ki}. But then the string $b^k a^{n-1} b^{n-1} a^i a^{2n-2} b^{n-1}$ is also accepted by N. However, this string is not in $\textsc{shift}(L)$ since each of its cyclic shifts not ending with b contains a substring b^{n-1+k}, where $n - 1 + k > n - 1$, and the automaton M in Figure 7 cannot read any such string; on the other hand, no string ending with b is accepted by M. If $q_0 = p_{kj}$ for some $k > 0$ and some j, then the string $b^k a^{n-1} b^{n-1} a^{2n-2} b^{n-j} a^{2n-2} b^{n-1}$, which is not in $\textsc{shift}(L)$ for the same reason, would be accepted. In both cases, we obtain a contradiction.

It remains to prove that the initial state q_0 is different from each state q_{0i} and p_{0j}. Suppose that $q_0 = q_{0i}$ for some i. The string $b a^{2n-2} b^{n-2}$ is in $\textsc{shift}(L)$ and so must be accepted from state q_{0i}. Then the string $a^{n-1} b^{n-1} a^i b a^{2n-2} b^{n-2}$ is also accepted by the NFA N. However, this string is not in $\textsc{shift}(L)$, because the only cyclic shift of this string that could be in L is $a^{n-1} b^{n-2} a^{n-1} b^{n-1} a^i b a^{n-1}$ (any other shift contains a substring a^n, which M cannot read), but M gets to state $n - 1$ after reading $a^{n-1} b^{n-2} a^{n-2}$, where it cannot read the next a, and hence this shift is not in L. If $q_0 = p_{0j}$ for some j, then the string $a^{n-1} b^{n-1} a^{2n-2} b^{n-j} b a^{2n-2} b^{n-2}$, which is not in $\textsc{shift}(L)$ because each of its cyclic shifts contains a substring a^{2n-2}, would be accepted. So, we again have a contradiction and the lemma follows. □

Hence we have shown that the nondeterministic state complexity of cyclic shift is $2n^2 + 1$ if $n \geqslant 2$. The witness languages are defined over a binary alphabet. Taking into account our earlier observation that $\textsc{shift}(L(M)) = L(M)$ for every 1-state NFA M and for every M over a unary alphabet, we can further conclude that for $n = 1$ the nondeterministic state complexity is 1, while in the case of a

unary alphabet the corresponding nondeterministic state complexity function is $g_1(n) = n$. Our results are stated in the following theorem.

Theorem 4. *The nondeterministic state complexity g_k of cyclic shift for a k-letter alphabet is*

$$g_1(n) = n \qquad \text{and} \qquad g_k(n) = \begin{cases} 1, & \text{if } n = 1 \\ 2n^2 + 1, & \text{if } n \geqslant 2 \end{cases} \qquad (\text{for every } k \geqslant 2).$$

5. Calculations

We have determined the nondeterministic state complexity of cyclic shift exactly, and the worst-case automaton constructed in Lemmata 13–14 has a visible structure. On the other hand, our results in the deterministic case 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 state complexity of cyclic shift are like.

These hardest DFAs can be found using an exhaustive search over all the automata over a given k-letter alphabet with a given number of states n. For every DFA, an NFA for its cyclic shift has to be constructed and determinized, and the result has to be minimized. The greatest number of states in the resulting automata, denoted $f_k(n)$, can thus be calculated.

Another number that can be effectively calculated is the function $f(n) = \sup_{k \geqslant 1} f_k(n)$, that is, the state complexity of the cyclic shift over all alphabets. This function is well-defined according to Theorem 1, which states that $f(n) = f_{n^n}(n)$. Note that the transition table over n states and n^n letters is unique up to a permutation of letters. The automata differ only in whether the initial state is accepting, and in the number of accepting states among the non-initial states. The case of all accepting states is trivial, which leaves us with as few as $2n - 2$ automata to consider, and the greatest number of states for their cyclic shift gives the value of $f(n)$.

Let us report the results of our calculations 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, f_6 correspond to alphabets of size 2, 3, 4, 5, and 6, respectively. The column $f(n)$ gives the hardest result over all alphabets, computed by considering $2n - 2$ automata as described above. Our theoretical upper bound $(n2^n - 2^{n-1})^n$ is included for comparison in the rightmost column. Among the values in Table 1, our lower bound $(n-1)! \cdot 2^{(n-1)(n-2)}$ is applicable to $f_4(3) = 702$ and $f_4(4) = 1\,087\,620$, which are accordingly proved to be at least 8 and 384, respectively.

An interesting thing to note is that the state complexity of cyclic shift depends on the size of the alphabet, while for all previously studied common operations the worst case automata are defined over a binary alphabet [10,16,23].

The actual hardest automata responsible for the values in the table are provided in Figures 8–10 (we omit 8 hardest automata for $f_5(3)$ and 16 hardest automata

TABLE 1. State complexity of the cyclic shift: calculated values.

n	$f_2(n)$	$f_3(n)$	$f_4(n)$	$f_5(n)$	$f_6(n)$	$f(n)$	upper bound
1	1	1	1	1	1	1	1
2	5	5	5	5	5	5	36
3	108	511	702	805	832	845	8000
4	20 237	550 283	1 087 620			1 349 340	9 834 496
5	56 817 428						61 917 364 224

for $f_6(3)$). It is important to note that these are *all* automata with the respective number of states and over the respective alphabets (modulo permutation of letters) whose cyclic shift requires this many states. The next hardest automaton in each case requires fewer states than the number in the table minus one: for instance, for the alphabet $\{a, b\}$ and for 3 states the next reachable value below 108 is 89, and there exist no 3-state automata over this alphabet whose cyclic shift requires between 90 and 107 states. For 4 states and the binary alphabet the next value below 20 237 is 19 718.

Such intermediate unreachable values of descriptional complexity are known in the literature as *magic numbers* [8,15,24], and, in contrast, it has recently been shown that there are no such numbers for union and intersection of DFAs [12]. The existence of magic numbers for cyclic shift of DFAs over a fixed alphabet, which we observed in our calculations, is in fact very easy to establish theoretically. Note that the state complexity function of cyclic shift, $2^{\Theta(n^2)}$, grows faster than the number of DFAs with n states, which is $2^{O(n \log n)}$ [7], and therefore, for every fixed alphabet of k letters, the values of the state complexity of cyclic shift for n-state automata are dispersed across the range between 1 and $f_k(n)$, with large gaps between some values. Let us note in passing that if we consider magic numbers for cyclic shift of DFAs over all alphabets, then the above counting argument is no longer applicable, and the question of the existence of such magic numbers remains open.

Returning to the hardest automata in Figures 8–10, it is difficult to understand what makes them the hardest. It is no easier to explain the numbers in the sequences: why 108? why 20 237? 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 [2,5,16,17,20,22], and the hardest automata have been determined. It would be very interesting to obtain similar results for cyclic shift.

6. CONCLUSION

With its $2^{n^2+n \log n-O(n)}$ state complexity, 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

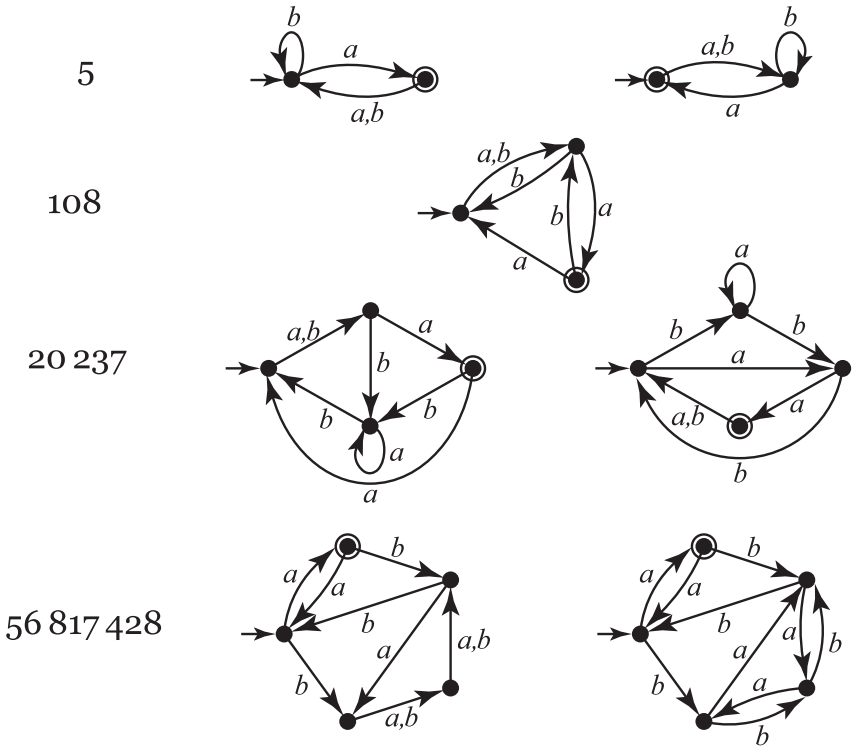


FIGURE 8. The hardest DFAs over $\{a, b\}$, and how many states their cyclic shift requires.

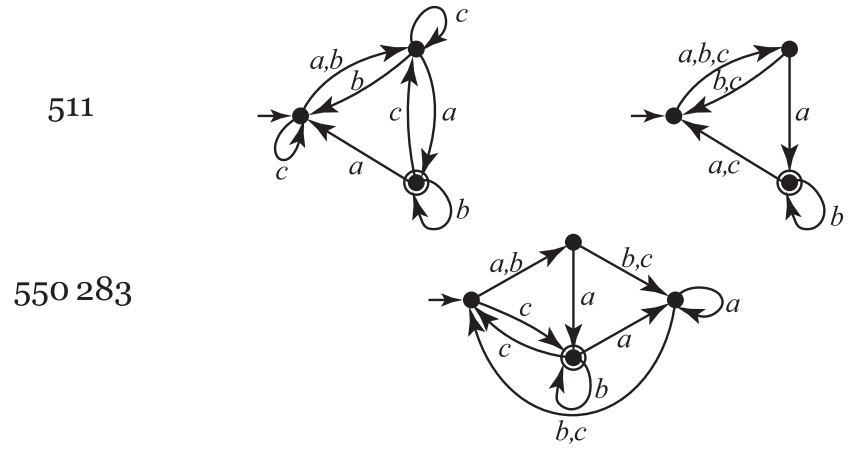


FIGURE 9. The hardest three-symbol DFAs.

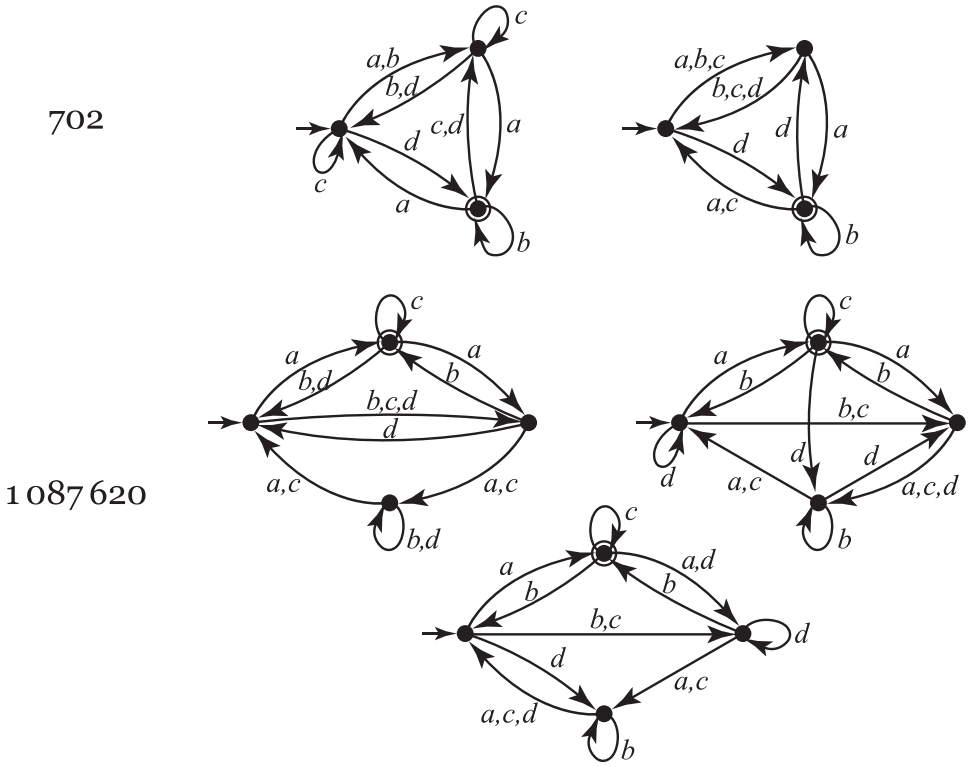


FIGURE 10. The hardest four-symbol DFAs.

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 nondeterminsitic 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 with a certain precision, nothing else is known about this integer sequence and about the hardest automata corresponding to its elements. Only once the form of these automata is explained, one could say that the cyclic shift of regular languages is entirely understood. Investigating what makes the hardest automata the hardest is left as a challenging research problem.

Acknowledgements. We are grateful to Vladimir Zakharov for proposing the study, and to Michael Domaratzki and Kai Salomaa for discussions at the early stage of our research. We are indebted to Jozef Jirásek, Michal Kunc, Vladimir Prus and Alexei Rybak for lending machine time for the calculation of $f_2(5)$ and $f_4(4)$. We wish to thank an anonymous referee for careful reading and for the good advice to use the notation $\mathcal{S}(k, P)$ in Section 3.

References

[1] A.V. Aho, J.D. Ullman and M. Yannakakis, On notions of information transfer in VLSI circuits, in *Proceedings of 15th ACM STOC*, ACM (1983) 133–139.

[2] J.-C. Birget, Intersection and union of regular languages and state complexity. *Inform. Process. Lett.* **43** (1992) 185–190.

[3] J.-C. Birget, Partial orders on words, minimal elements of regular languages, and state complexity. *Theor. Comput. Sci.* **119** (1993) 267–291.

[4] J.-C. Birget, The state complexity of $\overline{\Sigma^*\overline{L}}$ and its connection with temporal logic. *Inform. Process. Lett.* **58** (1996) 185–188.

[5] C. Câmpeanu, K. Salomaa and S. Yu, Tight lower bound for the state complexity of shuffle of regular languages. *J. Autom. Lang. Comb.* **7** (2002) 303–310.

[6] M. Domaratzki, State complexity and proportional removals. *J. Autom. Lang. Comb.* **7** (2002) 455–468.

[7] M. Domaratzki, D. Kisman and J. Shallit, On the number of distinct languages accepted by finite automata with n states. *J. Autom. Lang. Comb.* **7** (2002) 469–486.

[8] V. Geffert, Magic numbers in the state hierarchy of finite automata, *Mathematical Foundations of Computer Science*, MFCS 2006, Stará Lesná, Slovakia, August 28–September 1, 2006, Springer, Berlin. *Lect. Notes Comput. Sci.* **4162** (2006) 312–423.

[9] I. Glaister and J. Shallit, A lower bound technique for the size of nondeterministic finite automata. *Inform. Process. Lett.* **59** (1996) 75–77.

[10] M. Holzer and M. Kutrib, Nondeterministic descriptional complexity of regular languages. *Int. J. Found. Comput. Sci.* **14** (2003) 1087–1102.

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

[12] M. Hricko, G. Jirásková and A. Szabari, Union and intersection of regular languages and descriptional complexity, in *Proceedings of DCFS 2005*, Como, Italy, June 30–July 2, 2005, 170–181.

[13] J. Hromkovič, *Communication Complexity and Parallel Computing*. Springer-Verlag, Berlin, Heidelberg (1997).

[14] J. Hromkovič, Descriptional complexity of finite automata: concepts and open problems. *J. Autom. Lang. Comb.* **7** (2002) 519–531.

[15] K. Iwama, A. Matsuura and M. Paterson, A family of NFAs which need $2^n - \alpha$ deterministic states. *Theor. Comput. Sci.* **301** (2003), 451–462.

[16] G. Jirásková, State complexity of some operations on binary regular languages. *Theor. Comput. Sci.* **330** (2005) 287–298.

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

[18] A.N. Maslov, Cyclic shift operation for languages. *Probl. Inf. Transm.* **9** (1973) 333–338.

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

[20] A. Salomaa, D. Wood and S. Yu, On the state complexity of reversals of regular languages. *Theor. Comput. Sci.* **320** (2004) 315–329.

[21] A. Salomaa, K. Salomaa and S. Yu, State complexity of combined operations. *Theor. Comput. Sci.* **383** (2007) 140–152.

[22] S. Yu, State complexity: recent results and open problems. *Fund. Inform.* **64** (2005) 471–480.

[23] S. Yu, Q. Zhuang and K. Salomaa, The state complexity of some basic operations on regular languages. *Theor. Comput. Sci.* **125** (1994) 315–328.

[24] L. van Zijl, Magic numbers for symmetric difference NFAs. *Int. J. Found. Comput. Sci.* **16** (2005) 1027–1038.

Communicated by J. Hromkovic.
Received December 6, 2006. Accepted August 17, 2007.