On the state complexity of the cyclic shift

Galina Jirásková Alexander Okhotin

April 6, 2005

Abstract

The cyclic shift of a language L, defined as $\text{SHIFT}(L) = \{uv \mid vu \in L\}$, is an operation known to preserve regularity. *** We improve this result by obtaining a lower bound $(n-1)! \cdot 2^{(n-1)(n-2)}$ using a fixed 4-letter alphabet, and prove that the state complexity is $2^{n^2+n\log n+\Omega(n)}$ for automata with 4 letters and more, and $2^{\Omega(n^2)}$ for 2- and 3-letter alphabets. We also investigate the NFA state complexity of this operation, proving that it is exactly $2n^2+1$ for any non-singleton alphabet.

1 Introduction

The cyclic shift is a unary operation on formal languages, defined as $\text{SHIFT}(L) = \{uv \mid vu \in L\}$ and known occasionally studied since the 1960s. It obviously preserves the class of regular languages. Another less obvious fact is that the cyclic shift of every context-free language is context-free, and three independent proofs of this result are known: the proof by Oshiba [4] is based upon the representation of a context-free language as a homomorphic image of a Dyck language intersected with a regular set; Maslov [3] uses pushdown automata; Hopcroft and Ullman [1, Exercise 6.4(c)] use context-free grammars.

The state complexity of the cyclic shift has been *** in Maslov's pioneering paper on the state complexity of deterministic finite automata [2], where ***. *** representation $\text{SHIFT}(L(A)) = \bigcup_{i=0}^{n-1} L(B_i) \cdot L(C_i)$ *** connected with k^n.

In the context of the modern state complexity theory. ***

Finite automata are one of the simplest computational models. Despite their simplicity, some challenging problems concerning finite automata are still open. For instance, we recall the question of how many states are sufficient and necessary for two-way deterministic finite automata to simulate two-way nondeterministic finite automata. The importance of this problem is underlined by its relation to the well-known open question whether or not DLOGSPACE equalsNLOGSPACE [**?**, **?**].

Recently, a renewed interest in regular languages and finite automata can be observed. For a discussion, the reader may refer to [**?**, **?**]. Some aspects of this area are now intensively investigated. One such aspect is the state complexity of regular languages and their operations.

The state complexity of a regular language is the number of states of its minimal deterministic finite automaton (DFA). The nondeterministic state complexity of a regular language is the number of states of a minimum state nondeterministic finite automaton (NFA) accepting the language. The state complexity (the nondeterministic state complexity) of an operation on regular languages represented by DFAs (NFAs, respectively) is the number of states that are sufficient and necessary in the worst case for a DFA (an NFA, respectively) to accept the language resulting from the operation.

Some early results on the state complexity of regular languages can be found in [**?**, **?**, **?**]. The state complexity of some operations on regular languages was investigated in [**?**, **?**, **?**]. Yu, Zhuang, and Salomaa [**?**] were the first to systematically study the complexity of regular language operations. Their paper was followed by several articles investigating the state complexity of finite language operations and unary language operations [**?**, **?**, **?**]. The nondeterministic state complexity of regular language operations was studied by Holzer and Kutrib in [**?**, **?**, **?**]. Further results on this topic are presented in [**?**, **?**, **?**] and state-of-the-art surveys for DFAs can be found in [**?**, **?**].

Hard operations. Traditionally, exponential means hard, polynomial means easy. union, intersection, complement, concatenation, star of DFAs.

Something much harder: the cyclic shift operation.

2 Constructing finite automata for the cyclic shift

Three models: (1) DFA: upper bound $(n * 2^n - 2^{n-1})^n$; (2) NFA with a single initial state: upper bound $2n^2 + 1$; (3) NFA with a set of initial states: upper bound $2n^2$.

To show these upper bounds, ***. A string w is in $\textsc{shift}(L(A))$ if and only if it can be factorized as $w = uv$, where $vu \in L(A)$. guess the middle state.

[3] Given an n-state NFA or DFA A with the set of states $Q = \{0, \ldots, n-1\}$, trans func Δ. For all i $(0 \leqslant i \leqslant n-1)$, let $B_i = (\Sigma, Q, \Delta, q_i, F)$ and $C_i = (\Sigma, Q, \Delta, q_0, \{q_i\})$.

If A is a DFA, then B_i and C_i are DFAs as well.

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

Since the state complexity of union and concatenation is known both for DFAs and for NFAs, *** can get an upper bound ***.

Definition 1. *For every $k \geqslant 1$ and $n \geqslant 1$, let $f_k(n)$ be *** DFA ***. Similarly, define $g_k(n)$ *** NFA ***.*

Let $f(n)$ and $g(n)$ be the numbers for an arbitrary alphabet. $f(n) = \max_k f_k(n)$, $g(n) = \max_k g_k(n)$.

Theorem 1. *Let $n \geqslant 1$.*

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

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

n-state NFA to a $2n^2$-state NFA.

Proof. The upper bound for $f(n)$ comes from □

$$(n2^n - 2^{n-1})^n = 2^{n^2 + n \log n + o(1)}.$$

3 Deterministic state complexity

In order to determine the asymptotics of the DFA complexity of the cyclic shift, we shall 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:

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

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

Note that the symbols a, b, c are permutations, **(by the way, is it a common term? should we cite anything?)** while d is not.

In order to argue that the minimal DFA accepting $\text{SHIFT}(L(A_n))$ must have many states, we shall consider an (n^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 $Q_i = \{q_{i0}, \ldots, q_{i,n-1}\}$ and $P_i = \{p_{i0}, \ldots, p_{i,n-1}\}$ $(0 \leqslant i \leqslant n-1)$ be $2n$ copies of Q. Denote $P = P_0 \cup \ldots \cup P_{n-1}$. 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 $q_{i,n-1}$ to $p_{i,0}$ $(0 \leqslant i \leqslant n-1)$. The set of accepting states is $\{p_{00}, p_{11}, \ldots, p_{n-1,n-1}\}$. 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.

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 $\{q_{00}, q_{11}, \ldots, q_{n-1,n-1}, p_{n-1,0}\}$. Let us begin with proving that a large set of subsets is reachable from the initial subset.

3.2 Reachable subsets

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

a: $0 \to 0,\ \ 1 \to 2 \to \ldots \to n-2 \to 1,\ n-1 \to n-1$
b: $0 \to 0,\ \ 1 \to 2 \to \ldots \to n-2 \to n-1 \to 1$
c: $0 \leftrightarrow 1,\ \ 2 \to 2,\ \ 3 \to 3,\ \ \ldots,\ \ n-1 \to n-1$

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: (i) q_{00} goes to q_{00} by u; (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}.*

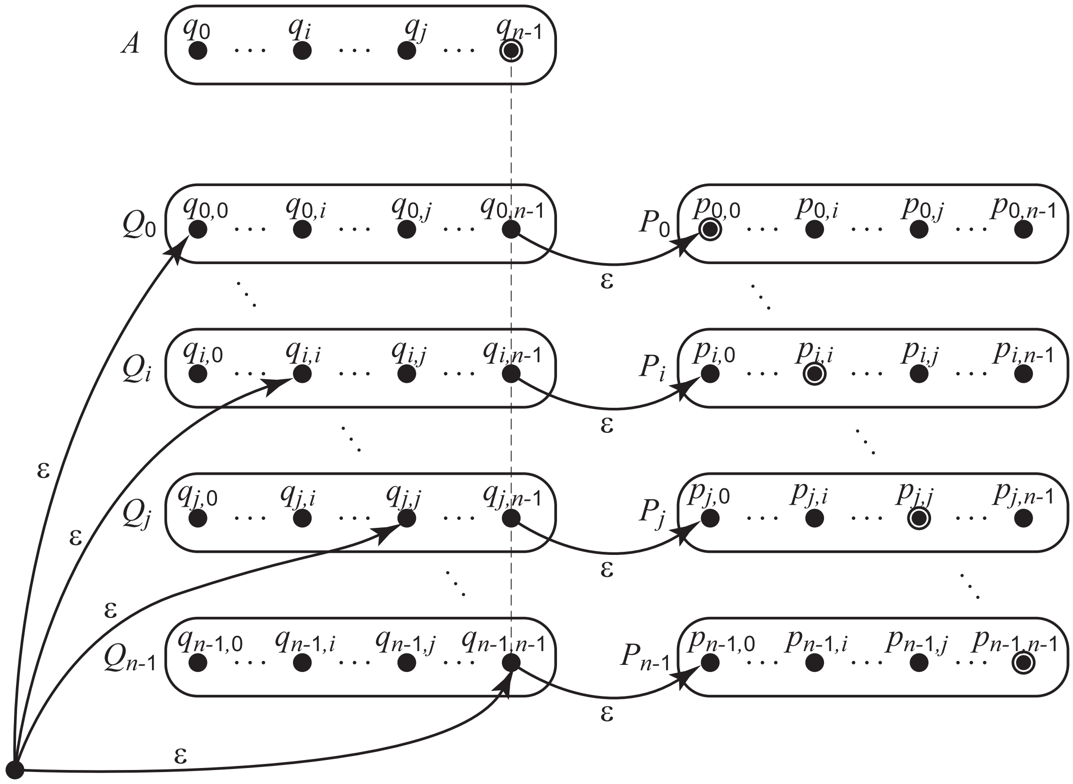

Figure 1: Construction of an NFA for $\textsc{shift}(L(A))$.

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 $i \neq m, m-1$). It is easy to check that this permutation is implemented by $u_i = b^{n-m-1}ab^{m-1}$.

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

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

$$S = \{q_{00}, q_{11}, q_{22}, \ldots, q_{n-1,n-1}\} \cup \{p_{10}, p_{20}, \ldots, p_{n-1,0}\} \cup P' \tag{1}$$

goes to $S \cup \{p_{ij}\}$ upon reading w_{ij}.

Proof. Let x be the string given by the permutation lemma for the permutation $\{i \leftrightarrow n-1, j \leftrightarrow 1\}$, and define w_{ij} as $xccx$. Applying this permutation allows us to reduce the problem to setting the bit p_{i1}, which is done by cc.

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. The state q_{ii} goes to $\{q_{i,n-1}, p_{i0}\}$ by x, then to $\{q_{i,n-1}, p_{i1}, p_{i0}\}$ by c, to $\{q_{i,n-1}, p_{i0}, p_{i1}\}$ 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. All other states q_{kk} ($1 \leqslant k \leqslant n-1$, $k \neq i, j$) remain in $\{q_{kk}\}$ upon reading $xccx$.

Similarly to q_{00}, each state p_{k0} ($1 \leqslant k \leqslant n-1$) goes to p_{k0} by $xccx$. Every $p_{k\ell}$ goes ***. ☐

Lemma 4 (Reachability). *For any $P' \subseteq \{p_{ij} \mid 1 \leqslant i \leqslant n-1,\ 1 \leqslant j \leqslant n-1,\ j \neq i\}$, 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 $\{q_{00}, q_{11}, q_{22}, \ldots, q_{n-1,n-1}\} \cup \{p_{10}, p_{20}, \ldots, p_{n-1,0}\}$ is reachable from the initial subset $\{q_{00}, q_{11}, q_{22}, \ldots, q_{n-1,n-1}\} \cup \{p_{n-1,0}\}$ 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 reachability of $2^{(n-1)(n-2)}$ subsets. In order to multiply this number by $(n-1)!$, 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 $\{p_{ij} \mid 1 \leqslant i \leqslant n-1,\ 1 \leqslant j \leqslant n-1,\ j \neq k_i\}$. Then the subset*

$$\{q_{00}, q_{1k_1}, q_{2k_2}, \ldots, q_{n-1,k_{n-1}}\} \cup \{p_{10}, p_{20}, \ldots, p_{n-1,0}\} \cup P'$$

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

Proof. first, reachability lemma; then, permutation. ☐

3.3 Inequivalence of subsets

We shall now prove that no two of the $(n-1)! \cdot 2^{(n-1)(n-2)}$ states that were shown to be reachable in *** are equivalent. To do this we first associate some special strings with states p_{ij} such that this strings are acepted by the NFA ??? only from state p_{ij}. The same will be done for states q_{ij}.

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^{n-i}.

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^{n-i}. 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_{il} with $l \neq j$ goes to state q_{i0} by the string $b^{n-1-j}d$, and state q_{i0} goes to itself by the string $b^{n-2}ccb^{n-2}db^{i-1}$. Any state q_{kl} 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_{kl} $(1 \leqslant k \leqslant n-1, 0 \leqslant l \leqslant n-1)$ 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 states that were shown to be reachable in ??? are pairwise inequivalent.*

Proof. Let R and S are two different states among these $(n-1)!2^{(n-1)(n-2)}$ reachable states, i.e. R,S do not contain any state of P_0 and any state of $\{q_{0j} \mid j > 0\}$. Then we have two cases:

(i) There is a state p_{ij} with $i > 0$ and $j > 0$ such that p_{ij} is, w.l.o.g., in state R but not in state S. Then, by Lemma 6, the string $b^{n-1-j}db^{n-i}$ is accepted by the DFA starting in state R, and it is not accepted starting in state S.

(ii) There is a state q_{ij} with $i > 0$ and $j > 0$ which is in one of the states R or S but not in both. Then, by Lemma 7, the string $b^{n-1-j}db^{n-2}ccb^{n-2}db^{i-1}$ distinguishes states R and S.

Hence no two different states are equivalent. ☐

Theorem 2. $f_4(n) \geqslant (n-1)! \cdot 2^{(n-1)(n-2)} = 2^{n^2+n\log n-(3+\log_2 e)n+o(n)}$.

Proof. ***

Stirling's approximation: $n! \geqslant \sqrt{2\pi}n^{n+1/2}e^{-n}$.

we have $(n-1)! \geqslant \sqrt{2\pi}(n-1)^{n-1/2}e^{-n+1} = \sqrt{2\pi}2^{(n-1/2)\log_2(n-1)-(n-1)(\log_2 e)} = 2^{n\log_2(n-1)-(\log_2 e)n+o(n)} = 2^{n\log_2 n+n\log_2((n-1)/n)-(\log_2 e)n+o(n)} = 2^{n\log_2 n-(\log_2 e)n+o(n)}$

the whole lower bound: $2^{n^2+n\log n-(3+\log_2 e)n+o(n)}$ ☐

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 f_4 to establish quite a high lower bound for f_2. 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 $L \subseteq \Sigma_m^*$, $h^{-1}(\text{SHIFT}(h(L))) = \text{SHIFT}(L)$.*

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

Conversely, if $c_{i_1} \dots c_{i_\ell} \in h^{-1}(\text{SHIFT}(h(L)))$, then $h(c_{i_1} \dots c_{i_\ell}) \in \text{SHIFT}(h(L))$, and there exists a factorization $h(c_{i_1} \dots c_{i_\ell}) = xy$ (where $x, y \in \{a,b\}^*$), such that $yx \in h(L)$. Let us assume that at least one of c_{i_j} is not a_m (if $c_{i_j} = a_m$ for all j, then $xy = yx = a^{(m-1)\ell}$ ***).

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

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 leftmost b in it, which must be either the last symbol in z'' or the last symbol in some $h(c_{i_{j_0}})$. 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. $\square$

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

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

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 $2^{n^2/(m-1)^2}$ is a lower bound for $f_2(n)$.

Theorem 3. *$2^{n^2/9+o(n^2)}$ is a lower bound for $f_2(n)$. $f_2(n) = 2^{\Omega(n^2)}$.*

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 ??? The next lemma shows that this upper bound is tight for any integer n with $n \geqslant 2$. For $n = 1$, 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.*

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

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

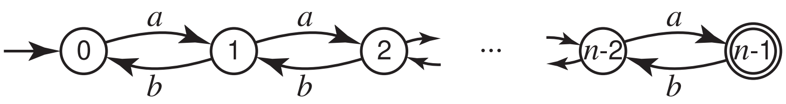

Figure 2: A worst-case NFA that requires $2n^2 + 1$ states for its cyclic shift.

Let L be the language accepted by the NFA M. Let $N = (Q, \Sigma, \delta, q_0, F)$ be any NFA for the language $\text{SHIFT}(L)$. 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 $\text{SHIFT}(L)$, 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 the language $\text{SHIFT}(L)$.

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 differet 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 $\text{SHIFT}(L)$ 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 $\text{SHIFT}(L)$ 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 $\text{SHIFT}(L)$.

(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 the language $\text{SHIFT}(L)$.

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 $x_{lj}y_{ki} = b^l uvb^{n-1-k}$ for some strings u and v. This string is not in the language $\text{SHIFT}(L)$ 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 $\text{SHIFT}(L)$.

(iii) $i \notin \{n-1, 2n-1\}$, and $j = n-1$. Then we have $x_{0i}y_{0j} = a^{n-1}b^{n-1}a^i b^{n-1}a^{2n-2}b^{n-1}$ and for any $k > 0$ $x_{ki}y_{kj} = b^k a^i b^{n-1}a^{2n-2}b^{n-1-k}$, and these strings are not in the language $\text{SHIFT}(L)$ for $i \neq n-1$ (the only possible shift of the latter string that could be in the language L is $a^{n-1}b^{n-1-k}b^k a^i b^{n-1}a^{n-1}$ which is in L only if $i = n-1$).

(iv) $i \notin \{n-1, 2n-1\}$, and $j = 2n-1$. Then for any k, we have $x_{kj}y_{ki} = b^k a^{2n-2}b^{n-1}a^{2n-2-i}b^{n-1-k}$, 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 $a^{n-1}b^{n-1}a^{2n-2-i}b^{n-1-k}b^k a^{n-1}$, which is in L only if $i = n-1$).

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

Hence we have shown that these $2n^2$ pairs of strings satisfy the conditions (1) and (2). Now, we can 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 we have mentioned above, these states are pairwise distinct. To prove the lemma it is sufficiet 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 by contradiction that $q_0 = q_{ki}$ for some $k > 0$ and some i. Then the NFA N also accets the string $x_{ki}a^{2n-2}b^{n-1}$. But $x_{ki} = b^k u$ for a string u, and the string $b^k u a^{2n-2}b^{n-1}$ is not in the language $\text{SHIFT}(L)$ if $k > 0$. So, we have 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 by contradiction 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 4. *For any integer n such that $n \geqslant 2$, $2n^2 + 1$ states are sufficient and necessary in the worst case for an NFA to accept the cyclic shift of an n-state NFA language.* ☐

5 Computations

The NFA complexity of the cyclic shift has been determined 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. Though that is enough to determine the state complexity, it would be interesting to ***, as it was done in the *** state complexity of the more common operations [ref*ref*ref].

It turned out that, though the hardest DFAs with respect to the cyclic shift can be found by an exhaustive search, it is hard to understand what makes them the hardest. The number of states required by these *** is given in Table 1, while the actual hardest automata are illustrated in Figure 3.

n	$f_2(n)$	$f_3(n)$	$f_4(n)$	$f_5(n)$	$f_{n^n}(n)$	$(n-1)! \cdot 2^{(n-1)(n-2)}$	$(n2^n - 2^{n-1})^n$
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.

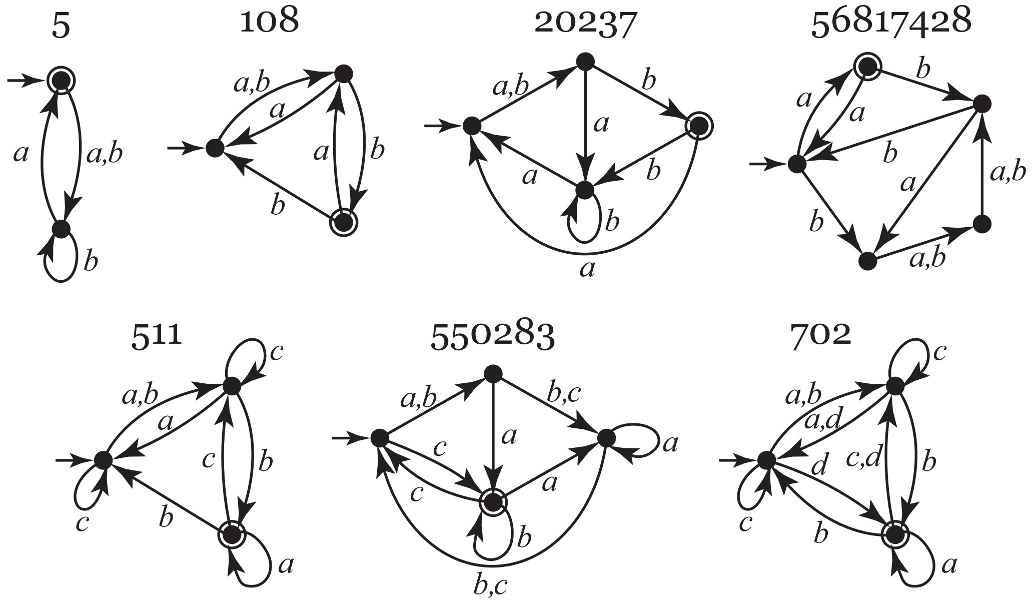

Figure 3: The worst-case DFAs, and how many states their cyclic shift requires.

6 Conclusion

Cyclic shift is hardest known elementary language operation on DFAs. Compare: star of a 5-state language over $\{a, b\}$ requires up to 24 states, cyclic shift, in the worst case, requires 56 million.

NB: the complexity is greater than the number of automata.

At the same time, its NFA state complexity is only quadratic.

further work: improve our estimation of the DFA complexity of the cyclic shift for a two-letter alphabet. explain what are the hardest cases.

References

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

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

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

[4] T. Oshiba, “Closure property of the family of context-free languages under the cyclic shift operation”, *Transactions of IECE*, 55D:4 (1972), 233–237.

Appendix

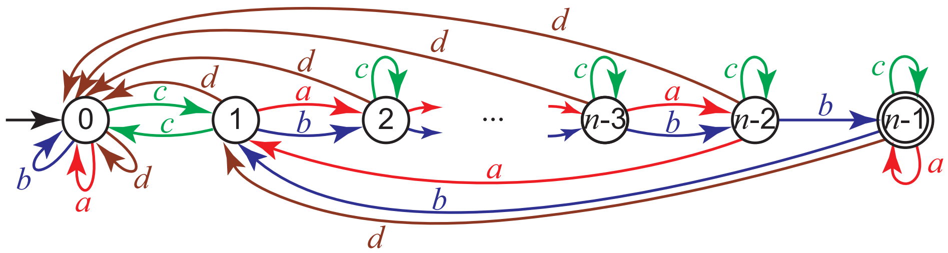

Figure 4: A DFA that requires at least $(n-1)! \cdot 2^{(n-1)(n-2)}$ states for its cyclic shift.