Cyclic Shift on Prefix-Free Languages

Jozef Jirásek [1,*] and Galina Jirásková [2,**]

[1] Institute of Computer Science, Faculty of Science, P.J. Šafárik University, Jesenná 5, 040 01 Košice, Slovakia
jozef.jirasek@upjs.sk

[2] Mathematical Institute, Slovak Academy of Sciences, Grešákova 6, 040 01 Košice, Slovakia
jiraskov@saske.sk

Abstract. We prove that the cyclic shift of a prefix-free language represented by a minimal complete n-state deterministic finite automaton is recognized by a deterministic automaton of at most $(2n-3)^{n-2}$ states. We also show that this bound is tight in the quaternary case, and that it cannot be met by using any smaller alphabet. In the ternary and binary cases, we still get exponential lower bounds.

1 Introduction

Cyclic shift is a unary operation on formal languages defined as $\text{SHIFT}(L) = \{w \mid w = uv \text{ and } vu \in L\}$. The operation preserves regularity since the cyclic shift of a regular language may be expressed as a union of n concatenations [9]. Using such a representation, an upper bound $(n \cdot 2^n - 2^{n-1})^n$ on the state complexity of cyclic shift has been proved already by Maslov [9] in 1970. He also provided a lower bound $(n-2)^{n-2} \cdot 2^{(n-2)(n-2)}$ for incomplete deterministic automata over a growing alphabet of size $2n-2$. It follows that a lower bound for complete deterministic automata over a growing alphabet is $(n-3)^{n-3} \cdot 2^{(n-3)(n-3)}$.

The Maslov's lower bound has been improved by Jirásková and Okhotin [7] by presenting a regular language recognized by a complete n-state deterministic finite automaton, defined over a fixed four-letter input alphabet, that requires at least $(n-1)! \cdot 2^{(n-1)(n-2)}$ deterministic states for its cyclic shift. Nevertheless, the new lower bound does not match the above mentioned upper bound.

In the case of prefix-free regular languages, concatenation is a simple operation. While the state complexity of concatenation is $m \cdot 2^n - 2^{n-1}$ in the general case [9,14], it is only $m+n-2$ if the operands are prefix-free [3,6]. Now a question arises whether such an easy concatenation on prefix-free languages could be used to get the exact value of the state complexity of cyclic shift on this subclass of regular languages. In our paper, we answer this question positively, and prove the tight bound $(2n-3)^{n-2}$ on the state complexity of cyclic shift on prefix-free languages.

* Research supported by grants VEGA 1/0832/12 and APVV-0035-10.

** Research supported by grants VEGA 2/0183/11 and APVV-0035-10.

Our witness languages are defined over a four-letter alphabet. We also prove the optimality of the size of an input alphabet by showing that the upper bound $(2n-3)^{n-2}$ on the state complexity of cyclic shift on prefix-free languages cannot be met by any language defined over a ternary (or any smaller) alphabet. However, in the ternary and binary cases, we still are able to prove exponential lower bounds $(n-2)! \cdot 2^{n-2}$ and $(n-2) \cdot (3^{n-2}-1)+1$, respectively. Our calculations show that these lower bounds can be exceeded.

The study of cyclic (or circular) shift has applications in coding theory. Cyclic codes are block codes, in which the cyclic shift of a codeword always yields another codeword. Thus $L = \textsc{shift}(L)$ for a cyclic code L. It is known that the operation of cyclic shift preserves context-freeness [10,11], and that the cyclic shift of a language described by a regular expression of length n can be described by a regular expression of length $O(n^3)$ [2].

In prefix codes, like variable-length Huffman codes or country calling codes, there is no codeword that is a proper prefix of any other codeword. With such a code, a receiver can identify each codeword without any special marker between words. Motivated by prefix codes, the class of prefix-free regular languages have been recently investigated. It is known that every minimal deterministic automaton recognizing a prefix-free regular language must have exactly one final state, from which all transitions go to a dead state. Using this property, tight bounds on the state complexity of basic operations such as union, intersection, concatenation, star, and reversal have been obtained in [3] and strengthened in [6,8]. The nondeterministic state complexity of basic regular operations has been studied in [4,6], while the complexity of combined operations on prefix-free regular languages has been investigated in [5].

2 Preliminaries

We assume that the reader is familiar with basic concepts of regular languages and finite automata and for unexplained notions we refer to [12,13].

For an alphabet Σ, let Σ^* be the set of all strings over Σ, including the empty string ε. A language is any subset of Σ^*. We denote the power-set of a set X by 2^X. For an integer m, let $[m] = \{0, 1, \ldots, m-1\}$.

A *deterministic finite automaton* (DFA) is a quintuple $M = (Q, \Sigma, \cdot, s, F)$, where Q is a finite non-empty set of states, Σ is an input alphabet, $\cdot : Q \times \Sigma \to Q$ is the transition function, $s \in Q$ is the initial (start) state, and $F \subseteq Q$ is the set of final states. In this paper, all DFAs are assumed to be *complete.* The transition function $\cdot$ is extended to the domain $Q \times \Sigma^*$ in a natural way. The *language accepted by the DFA* M is the set of strings $L(M) = \{w \in \Sigma^* \mid s \cdot w \in F\}$. A state q of M is called a *dead state* if no string is accepted by M from q.

A *nondeterministic finite automaton* (NFA) is a quintuple $M = (Q, \Sigma, \cdot, S, F)$, where $Q, \Sigma,$ and F are defined in the same way as for a DFA, S is the set of initial states, and $\cdot$ is the nondeterministic transition function that maps $Q \times \Sigma$

to 2^Q. The transition function can be naturally extended to the domain $2^Q \times \Sigma^*$. The *language accepted by NFA M* is $L(M) = \{w \in \Sigma^* \mid S \cdot w \cap F \neq \varnothing\}$.

Two automata are *equivalent* if they recognize the same language. A DFA M is *minimal* if every DFA equivalent to M has at least as many states as M. It is well-known that a DFA is minimal if all of its states are reachable and pairwise distinguishable. The *state complexity* of a regular language L, $\text{sc}(L)$, is the number of states in the minimal DFA recognizing the language L.

The *cross-product automaton* [1] for the union of two languages recognized by DFAs $(Q_A, \Sigma, \circ, s_A, F_A)$ and $(Q_B, \Sigma, \bullet, s_B, F_B)$, respectively, is the DFA

$$(Q_A \times Q_B, \Sigma, \cdot, (s_A, s_B), F),$$

where $(p, q) \cdot a = (p \circ a, q \bullet a)$ and $F = (F_A \times Q_B) \cup (Q_A \times F_B)$.

2.1 Prefix-Free Languages

If u, v, w are strings in Σ^* and $w = uv$, then u is a *prefix* of w. If, moreover, $v \neq \varepsilon$, then u is a *proper prefix* of w. A language is *prefix-free* if it does not contain two strings, one of which is a proper prefix of the other.

It is well known that a minimal DFA recognizes a non-empty prefix-free language if and only if it has a dead state and a unique final state, from which all transitions go to the dead state.

3 Cyclic Shift on Prefix-Free Languages

The cyclic shift of a language L is defined as

$$\text{SHIFT}(L) = \{uv \mid vu \in L\}.$$

Assume that the language L is recognized by a DFA A. By definition, a string w is in $\text{SHIFT}(L)$ if it can be partitioned as $w = uv$ so that the string vu is in L. This means that there is a state q, such that the computation of A on the string v ends in the state q, while the string u is accepted by A from the state q. This gives the following result from [9].

Lemma 1 (Maslov [9]). *Let $A = (Q, \Sigma, \cdot, q_0, F)$ with $Q = \{q_0, q_1, \ldots, q_{n-1}\}$ be an n-state DFA. For $i = 0, 1 \ldots, n-1$, let $B_i = (Q, \Sigma, \cdot, q_i, F)$ and $C_i = (Q, \Sigma, \cdot, q_0, \{q_i\})$ be the DFAs that have the same state set and the same transitions as the DFA A, and differ from A only in their initial and final states. Then*

$$\text{SHIFT}(L(A)) = \bigcup_{i=0}^{n-1} L(B_i)\, L(C_i).$$

3.1 Upper Bound for Cyclic Shift on Prefix-Free Languages

Using the above mentioned Maslov's result we now get an upper bound on the number of states of deterministic finite automata recognizing the cyclic shift of prefix-free languages.

Lemma 2 (Upper Bound). *Let $n \geq 3$ and let L be a prefix-free language accepted by a minimal n-state DFA. Then the language* $\textsc{shift}(L)$ *is accepted by a DFA of at most $(2n-3)^{n-2}$ states.*

Proof. Let $A = (Q, \Sigma, \cdot, q_0, \{q_{n-2}\})$ with $Q = \{q_0, q_1, \ldots, q_{n-1}\}$ be a minimal DFA for a prefix-free language L, in which q_{n-1} is the dead state, and q_{n-2} is the sole final state. Then, by Lemma 1, $\textsc{shift}(L) = \cup_{i=0}^{n-1} L(B_i)L(C_i)$, where $B_i = (Q, \Sigma, \cdot, q_i, \{q_{n-2}\})$ and $C_i = (Q, \Sigma, \cdot, q_0, \{q_i\})$. Since q_{n-1} is the dead state of A, and all transitions defined in the unique final state q_{n-2} go to the dead state q_{n-1}, the language $L(B_{n-1})$ is empty and $L(B_{n-2}) = \{\varepsilon\}$. Therefore, the language $L(B_{n-1})L(C_{n-1})$ is empty and

$$L(B_{n-2})L(C_{n-2}) = L(C_{n-2}) = L(B_1) \subseteq L(B_1)L(C_1)$$

since $\varepsilon \in L(C_1)$. Hence $\textsc{shift}(L) = \cup_{i=0}^{n-3} L(B_i)L(C_i)$.

For $i = 0, \ldots, n-3$, the language $L(B_i)L(C_i)$ is accepted by a DFA D_i obtained from the DFAs B_i and C_i as follows. First, since all transitions defined in the unique final state q_{n-2} of B_i go to the dead state, the state q_{n-2} can be merged with the initial state q_0 of C_i. Next, the state q_{n-1} in B_i as well as the states q_{n-1} and q_{n-2} in C_i are all dead, and therefore can be merged into a single dead state. The resulting DFA D_i is deterministic and has $2n-3$ states.

Now the language $\textsc{shift}(L) = \cup_{i=0}^{n-3} L(B_i)L(C_i)$ is accepted by the cross-product automaton $D_0 \times D_1 \times \cdots \times D_{n-3}$ that has at most $(2n-3)^{n-2}$ states. The construction is illustrated in Fig. 1. □

3.2 Lower Bound in Quaternary Case

Throughout this subsection assume that $n \geq 4$ and $\Sigma = \{a, b, c, d\}$. Recall that $[m] = \{0, 1, \ldots, m-1\}$. Our aim is to prove that the upper bound on the state complexity of cyclic shift of prefix-free languages given in the previous lemma is tight in the case of a four-letter alphabet.

To this aim define a quaternary n-state DFA $A = ([m+2], \Sigma, \cdot, 0, \{m\})$, where $m = n-2$. For each state i in $[m]$,

$$\begin{aligned}
i \cdot a &= i + 1 \bmod m, \\
i \cdot b &= \begin{cases} 1, \text{ if } i = 0, \\ 0, \text{ if } i = 1, \\ i, \text{ otherwise}, \end{cases} \\
i \cdot c &= \begin{cases} 0, \text{ if } i \in \{0, 1\}, \\ i, \text{ otherwise}, \end{cases} \\
i \cdot d &= i + 1,
\end{aligned}$$

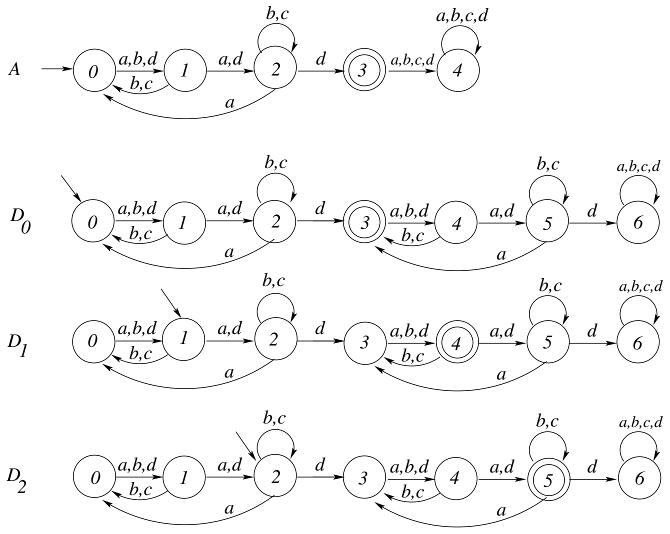

Fig. 1. A five-state DFA A and the resulting DFAs D_i for $L(B_i)L(C_i)$ for $i = 0, 1, 2$

and $m \cdot \sigma = (m+1) \cdot \sigma = m+1$ for each input σ in Σ. The DFA A is depicted in Fig. 2. Since all transitions defined in the unique final state m go to the dead state $m+1$, the language $L(A)$ is prefix-free.

Note that on states in $[m]$, input a causes a great permutation, input b causes a transposition, and input c causes a contraction. Thus, the semigroup of functions of $[m]$ into itself is generated by the inputs a, b, c.

For $i = 0, 1, \ldots, m-1$, construct the DFA $D_i = ([2m+1], \Sigma, \circ, i, \{m+i\})$ accepting the language $L(B_i)L(C_i)$ as described in the proof of the previous lemma. All the automata D_i's have the same transition function $\circ$, defined by $2m \circ \sigma = 2m$ and $i \circ \sigma = (m+i) \circ \sigma = i \cdot \sigma$ for each i in $[m]$ and σ in Σ, and these automata differ only in the initial and final states. Fig. 1 shows the DFAs D_0, D_1, and D_2 corresponding to the DFA A in the case of $m = 3$.

Then the language $\text{SHIFT}(L) = \cup_{i=0}^{m-1} L(D_i)$ is recognized by the cross-product automaton $D_0 \times \cdots \times D_{m-1}$. Our aim is to prove that the cross-product automaton has $(2n-3)^{n-2} = (2m+1)^m$ reachable and pairwise distinguishable states.

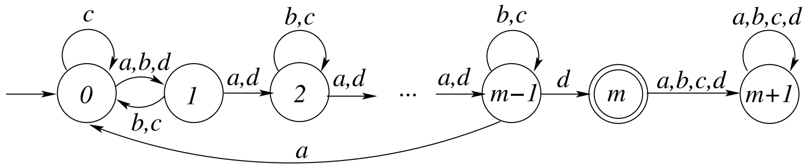

Fig. 2. The quaternary n-state witness DFA A; $m = n - 2$

Let us start with reachability. The state set of the cross-product automaton consists of m-tuples in $[2m+1]^m$, and the initial m-tuple is $(0, 1, \ldots, m-1)$. The next lemma shows that every m-tuple in $[2m+1]^m$ is reachable.

Lemma 3 (Reachability). *Every m-tuple in $[2m+1]^m$ is reachable in the cross-product automaton $D_0 \times D_1 \times \cdots \times D_{m-1}$.*

Proof. Let $(k_0, k_1, \ldots, k_{m-1})$ be an arbitrary but fixed m-tuple in $[2m+1]^m$. We will show that there is a string w that moves the cross-product automaton from its initial state $(0, 1, \ldots, m-1)$ to the state $(k_0, k_1, \ldots, k_{m-1})$.

For the m-tuple $(k_0, k_1, \ldots, k_{m-1})$, consider the two disjoint sets of indices I and J defined by

$$I = \{i \in [m] \mid k_i = 2m\},$$
$$J = \{i \in [m] \mid m \le k_i \le 2m-1\},$$

that is, the i-th component of the m-tuple is the dead state $2m$ of D_i whenever $i \in I$, it is a state in $\{m, m+1, \ldots, 2m-1\}$ whenever $i \in J$, and it is a state in $[m]$ otherwise. Next, define a function $f : [m] \to [m]$ by

$$f(i) = \begin{cases} 1, & \text{if } i \in I, \\ k_i - m, & \text{if } i \in J, \\ k_i, & \text{otherwise.} \end{cases}$$

Since the symbols a, b, c perform the three basic functions on $[m]$ in the DFA A, there is a string v_f over $\{a, b, c\}$ that moves every state i in $[m]$ to $f(i)$ in A.

Finally, for each ℓ in $[m]$ consider the string $u_\ell = a^{m-1-\ell}\, d\, a^\ell$, and define

$$w = (\prod_{i \in I} u_i u_i \prod_{i \in J} u_i) \cdot v_f, \tag{1}$$

where $\prod$ stands for concatenation (in an arbitrary order).

Our goal is to prove that w is the desired string that moves the cross-product automaton from the initial state $(0, 1, \ldots, m-1)$ to the state $(k_0, k_1, \ldots, k_{m-1})$.

First, notice that each D_ℓ goes from its initial state ℓ to the state $m + \ell$ by the string $u_\ell = a^{m-1-\ell} d a^\ell$ and then to the dead state $2m$ by the next u_ℓ since

$$\ell \xrightarrow{a^{m-1-\ell}} m-1 \xrightarrow{d} m \xrightarrow{a^\ell} m+\ell \xrightarrow{a^{m-1-\ell}} 2m-1 \xrightarrow{d} 2m \xrightarrow{a^\ell} 2m.$$

On the other hand, if $j \neq \ell$, then D_j remains in its initial state j upon reading u_ℓ since D_j moves by $a^{m-1-\ell}$ from j to state $(j + m - 1 - \ell) \bmod m$, in which the transition on d is defined the same way as on a, and therefore reading u_ℓ from j with $j \neq \ell$ results in the same state as reading a^m from j:

$$j \xrightarrow{a^{m-1-\ell}} (j + m - 1 - \ell) \bmod m \xrightarrow{d} (j + m - \ell) \bmod m \xrightarrow{a^\ell} j.$$

Now consider the string $\prod_{i \in I} u_i u_i \prod_{i \in J} u_i$, that is, the first part of the string w in (1). Recall that the sets of indices I and J are disjoint, and therefore

- every D_ℓ with $\ell \in I$ goes from ℓ to $2m$ by $\prod_{i\in I} u_i u_i$ and remains in $2m$ upon reading $\prod_{i\in J} u_i$;
- every D_ℓ with $\ell \in J$ remains in its initial state ℓ upon reading $\prod_{i\in I} u_i u_i$ and then goes to $m+\ell$ by $\prod_{i\in J} u_i$;
- every D_ℓ with $\ell \notin I \cup J$ remains in ℓ upon reading $\prod_{i\in I} u_i u_i \prod_{i\in J} u_i$.

It follows that the string $\prod_{i\in I} u_i u_i \prod_{i\in J} u_i$ moves the cross-product automaton from its initial state $(0, 1, \dots, m-1)$ to the state $(k'_0, k'_1, \dots, k'_{m-1})$, where

$$k'_\ell = \begin{cases} 2m, & \text{if } \ell \in I, \\ m+\ell, & \text{if } \ell \in J, \\ \ell, & \text{otherwise.} \end{cases}$$

Then, after reading the second part of the string w in (1), that is the string v_f, which moves every state i in $[m]$ to state $f(i)$ in the DFA A, each dfa D_ℓ with $\ell \in I$ remains in its dead state $2m$, each dfa D_ℓ with $\ell \in J$ goes from $m+\ell$ to $m + f(\ell) = m + (k_\ell - m) = k_\ell$, while each DFA D_ℓ with $\ell \notin I \cup J$ goes from ℓ to $f(\ell) = k_\ell$.

Hence the string $w = (\prod_{i\in I} u_i u_i \prod_{i\in J} u_i) \cdot v_f$ moves the cross-product automaton from its initial state $(0, 1, \dots, m-1)$ to the state $(k_0, k_1, \dots, k_{m-1})$. This proves the lemma. □

The following lemma proves the distinguishability of all the states in the cross-product automaton. Note that only symbols a and d are needed to get this result, which will be used later in the paper when dealing with smaller alphabets.

Lemma 4 (Distinguishability). *Every two distinc states of the cross-product-product automaton $D_0 \times D_1 \times \cdots \times D_{m-1}$ can be distinguished by a string over $\{a, d\}$.*

Proof. Let $(k_0, k_1, \dots, k_{m-1})$ and $(k'_0, k'_1, \dots, k'_{m-1})$ be two distinct m-tuples in $[2m+1]^m$. Then there is an i in $[m]$ with $k_i \neq k'_i$, and without loss of generality we may assume that $k_i \neq 2m$. Set

$$w = d^{2m-1-k_i}\, a\, d^{m-1}\, a\, d^{m-1}\, a^{i+1},$$

and let us show that the string w is accepted by the cross-product automaton from $(k_0, k_1, \dots, k_i, \dots, k_{m-1})$ and rejected from $(k'_0, k'_1, \dots, k'_i, \dots, k'_{m-1})$.

The DFA D_i goes from k_i to the accepting state $m+i$ by the string w since

$$k_i \xrightarrow{d^{2m-1-k_i}} 2m-1 \xrightarrow{a} m \xrightarrow{d^{m-1}} 2m-1 \xrightarrow{a} m \xrightarrow{d^{m-1}} 2m-1 \xrightarrow{a} m \xrightarrow{a^i} m+i.$$

Therefore, the string w is accepted by the cross-product automaton from the state $(k_0, k_1, \dots, k_i, \dots, k_{m-1})$.

On the other hand, let us show that the DFA D_i rejects the string w from each state ℓ different from k_i. If $\ell > k_i$, the D_i moves from ℓ to the dead state $2m$

by w since it is already in $2m$ after reading d^{2m-1-k_i}. If $\ell < k_i$, then D_i moves from ℓ to $\ell' = \ell + 2m - 1 - k_i$ by d^{2m-1-k_i}. If $m \le \ell' < 2m-1$ or $\ell' < m-1$, then D_i moves from ℓ' to the dead state $2m$ by ad^{m-1} or $ad^{m-1}ad^{m-1}$, respectively. If $\ell' = m-1$, the D_i moves from ℓ' to its rejecting state i by $ad^{m-1}ad^{m-1}a^{i+1}$. Hence D_i rejects the string w from each state ℓ with $\ell \neq k_i$.

The transitions in each D_j with $j \neq i$ are the same as in D_i, however, the states $m+i$ and i are rejecting in D_j. Therefore, the DFA D_j rejects the string w from each of its states.

Thus the cross-product automaton rejects w from $(k'_0, k'_1, \ldots, k'_i, \ldots, k'_{m-1})$, which concludes the proof. □

Hence, in the quaternary case, we get a lower bound that matches our upper bound $(2n-3)^{n-2}$ given by Lemma 2. Our next aim is to show that the four-letter alphabet cannot be decreased, that is, to show that the upper bound cannot be met by using any smaller alphabet. On the other hand, we will get still exponential lower bounds in the ternary and binary cases.

3.3 Small Alphabets

Let us start with an upper bound in the ternary case.

Lemma 5. *Let $n \ge 5$. If L is a prefix-free language recognized by a minimal n-state DFA over a ternary input alphabet, then the minimal DFA for* SHIFT(L) *has less than $(2n-3)^{n-2}$ states.*

Proof. Let L be accepted by a minimal n-state DFA A over the alphabet $\{a, b, c\}$. Let $m = n-2$. Let the state set of A be $[m+2]$, with the unique final state m and the dead state $m+1$. Then, since the final state m is reachable in A, there must be a symbol σ in $\{a, b, c\}$ and a state j, from which A goes to m by σ. Without loss of generality, we may assume that $\sigma = c$.

Let $D_0 \times \cdots \times D_{m-1}$ be the cross-product automaton for SHIFT(L) described above. Consider those of its states, in which all the components are less than m, that is, the states in $[m]^m$, and let us show that at least one of them must be unreachable in the cross-product automaton.

For each permutation φ of $[m]$, the state $(\varphi(0), \varphi(1), \ldots, \varphi(m-1))$ may only be reached from the initial state $(0, 1, \ldots, m-1)$ by reading a string w over $\{a, b, c\}$, in which all symbols permute the set $[m]$ in the DFA A. Therefore, no c occurs in w. To reach all such permutation states, the symbols a and b must cause two permutations on $[m]$ generating the group of all permutations on $[m]$ since $m \ge 3$. However, in such a case, no state $(f(0), f(1), \ldots, f(m-1))$ in $[m]^m$, where f is a function from $[m]$ to $[m]$ which is not a permutation, can be reached in the cross-product automaton.

If at least one of the symbols a or b does not cause a permutation on $[m]$, then it is not possible to reach all the states $(\varphi(0), \varphi(1), \ldots, \varphi(m-1))$ where φ is a permutation on $[m]$ and $m \ge 3$. This concludes the proof. □

Now, using a subautomaton of our quaternary witness automaton defined in subsection 3.2 and shown in Fig. 2, we prove an exponential lower bound for the ternary case.

Lemma 6. *For every n with $n \geq 4$, there exists a prefix-free language recognized by an n-state DFA over a ternary alphabet such that every DFA for the language* $\textsc{shift}(L)$ *requires at least* $(n-2)!\,2^{n-2}$ *states.*

Proof. Consider the DFA B obtained from the DFA A in Fig. 2 by considering only the input symbols a, b, d. Since the symbols a and b cause a great permutation and a transposition on $[m]$, respectively, for each permutation φ on $[m]$, there is a string v_φ over $\{a, b\}$ that moves every state i in $[m]$ to the state $\varphi(i)$.

As shown in the proof of Lemma 3, for each set J of $[m]$ and each permutation φ on $[m]$, the state $(k_0, k_1, \ldots, k_{m-1})$ with

$$k_i = \begin{cases} m + \varphi(i), & \text{if } i \in J, \\ \varphi(i), & \text{otherwise} \end{cases}$$

is reached in the cross-product automaton from the initial state $(0, 1, \ldots, m-1)$ by the string

$$\prod_{i \in J} (a^{m-1-i}\, d\, a^i) \cdot v_\varphi.$$

This gives $(n-2)!\,2^{n-2}$ reachable states. All these states are pairwise distinguishable by Lemma 4. $\square$

Let us continue with the binary case. By using another subautomaton of our quaternary witness, the next lemma shows that the lower bound on the state complexity of cyclic shift of prefix-free languages is exponential even in the case of a two-letter alphabet.

Lemma 7. *For every n with $n \geq 4$, there exists a prefix-free language recognized by an n-state DFA over a binary alphabet such that every DFA for the language* $\textsc{shift}(L)$ *requires at least* $(n-2)(3^{n-2}-1)+1$ *states.*

Proof. Consider the DFA C obtained from the DFA A in Fig. 2 by considering only the input symbols a and d. Recall that $m = n - 2$.

There are 3^m possibilities of choosing two disjoint subsets I and J of $[m]$. For each of them, as shown in the proof of Lemma 3, the state $(k_0, k_1, \ldots, k_{m-1})$ with

$$k_i = \begin{cases} 2m, & \text{if } i \in I, \\ m + i, & \text{if } i \in J, \\ i, & \text{otherwise} \end{cases}$$

is reached in the cross-product automaton from the initial state $(0, 1, \ldots, m-1)$ by the string

$$\prod_{i \in I} (u_i\, u_i) \prod_{i \in J} u_i$$

with $u_i = a^{m-1-i}\, d\, a^i$. From every such state $(k_0, k_1, \ldots, k_{m-1})$, except for the state with $I = [m]$, the cross-product automaton moves after reading a^j with j in $[m]$ to the state $(k'_0, k'_1, \ldots, k'_{m-1})$ with

$$k_i = \begin{cases} 2m, & \text{if } i \in I, \\ m + (i+j) \bmod m, & \text{if } i \in J, \\ (i+j) \bmod m, & \text{otherwise.} \end{cases}$$

This gives $(n-2)(3^{n-2}-1)+1$ reachable states. The distinguishability again follows from Lemma 4. □

Recall that the state complexity of a regular language L, $\mathrm{sc}(L)$, is defined as the smallest number of states in any DFA recognizing the language L. Denote by $f_k(n)$ the state complexity function of cyclic shift on prefix-free regular languages over a k-letter alphabet defined by

$$f_k(n) = \max\{\mathrm{sc}(\textsc{shift}(L)) \mid L \subseteq \Sigma^*, |\Sigma| = k, L \text{ is prefix-free, and } \mathrm{sc}(L) = n\}.$$

Using this notation, we can summarize our results in the following theorem.

Theorem 1 (State Complexity). *Let $n \geq 5$ and $f_k(n)$ be the state complexity of cyclic shift on prefix-free regular languages over a k-letter alphabet. Then*

(i) $f_1(n) = n$;
(ii) $f_2(n) \geq (n-2)(3^{n-2}-1)+1$;
(iii) $(n-2)! \cdot 2^{n-2} \leq f_3(n) < (2n-3)^{n-2}$;
(iv) $f_4(n) = f_k(n) = (2n-3)^{n-2}$ *for every k with $k \geq 4$.*

Proof. The equality in (i) holds since the cyclic shift of every unary language is the same language. The lower bound on $f_2(n)$ in (ii) is given by Lemma 7, while the bounds on $f_3(n)$ in (iii) follow from Lemmata 5 and 6. The upper bound on $f_k(n)$ in (iv) is given by Lemma 2, and its tightness for $k = 4$ is proved in Lemmata 3 and 4. Since adding new symbols to the quaternary witness automata does not change the proofs of reachability and distinguishability in the quaternary case, the upper bound is tight for every k with $k \geq 4$. □

Hence the tight bound on the state complexity of cyclic shift on prefix-free languages over an alphabet of at least four symbols is $(2n-3)^{n-2}$. Moreover, the alphabet of size at least four is necessary for the tightness. Using any smaller alphabet, the upper bound $(2n-3)^{n-2}$ cannot be met. However, the lower bounds in the binary and ternary cases are still exponential, namely $(n-2)\cdot(3^{n-2}-1)+1$ and $(n-2)! \cdot 2^{n-2}$, respectively. Our calculations given in Table 1 show that the state complexity of cyclic shift on prefix-free languages in the binary and ternary cases is greater than the above mentioned lower bounds. Its exact value in these two cases remains open. The hardest binary and ternary automata for $n = 4, 5, 6, 7$ are shown in Fig. 3 and Fig. 4, respectively.

Table 1. The state complexity of cyclic shift on prefix-free languages

n	$f_2(n)$	$f_3(n)$	$f_4(n) = (2n-3)^{n-2}$
4	17	25	25
5	121	319	343
6	1709	6193	6561
7	36256	154976	161051

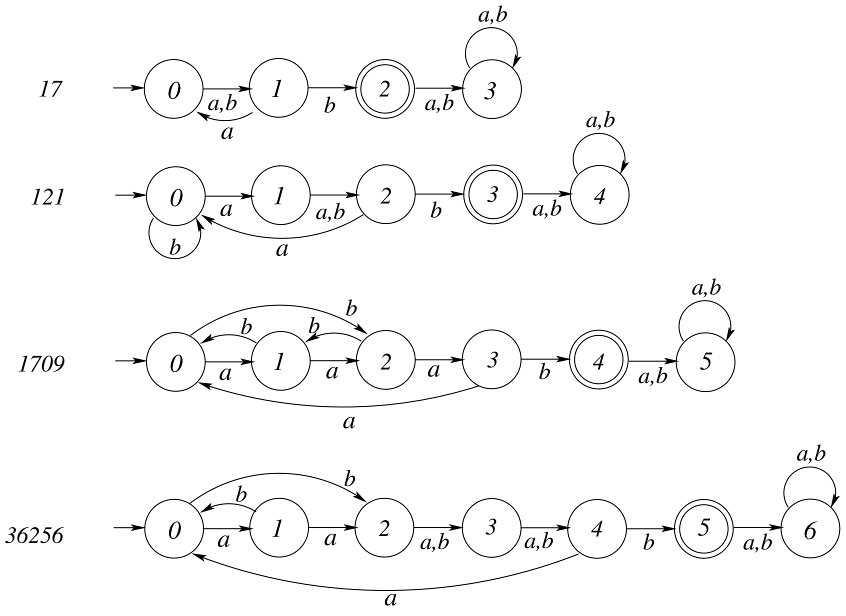

Fig. 3. The hardest binary DFAs; $n = 4, 5, 6, 7$

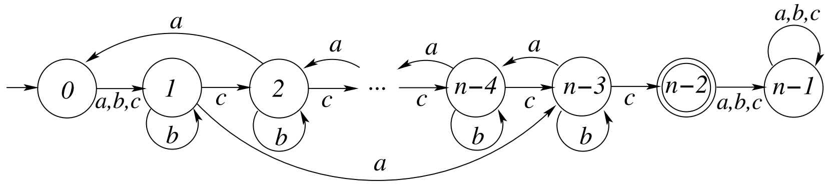

Fig. 4. The hardest ternary DFAs; for $n = 4, 5, 6, 7$

4 Conclusions

We investigated the state complexity of cyclic shift operation in the class of prefix-free regular languages. We obtained the upper bound $(2n-3)^{n-2}$, and we showed that it is tight in the case of a four-letter alphabet. We also proved that this upper bound cannot be met by any prefix-free language defined over a smaller alphabet. In the ternary and binary cases, we were still able to get exponential lower bounds $(n-2)! \cdot 2^{n-2}$ and $(n-2) \cdot (3^{n-2}-1)+1$, respectively. Our calculations showed that these lower bounds can be exceeded.

Notice that for incomplete deterministic finite automata, the tight bound for an alphabet of at least four symbols is $(2n-1)^{n-1}-1$.

The exact values of the state complexity of cyclic shift on binary and ternary prefix-free languages remain open, and are of interest to us. We also conjecture that the state complexity of cyclic shift on prefix-free languages in the binary case is smaller than that in the ternary case.

References

1. Birget, J.-C.: Intersection and union of regular languages and state complexity. Inform. Process. Letters 43, 185–190 (1992)
2. Gruber, H., Holzer, M.: Language operations with regular expressions of polynomial size. Theoret. Comput. Sci. 410, 3281–3289 (2009)
3. Han, Y.-S., Salomaa, K., Wood, D.: Operational state complexity of prefix-free regular languages. In: Automata, Formal Languages, and Related Topics, pp. 99–115. University of Szeged, Hungary (2009)
4. Han, Y.-S., Salomaa, K., Wood, D.: Nondeterministic state complexity of basic operations for prefix-free regular languages. Fund. Inform. 90, 93–106 (2009)
5. Han, Y.-S., Salomaa, K., Yu, S.: State complexity of combined operations for prefix-free regular languages. In: Dediu, A.H., Ionescu, A.M., Martín-Vide, C. (eds.) LATA 2009. LNCS, vol. 5457, pp. 398–409. Springer, Heidelberg (2009)
6. Jirásková, G., Krausová, M.: Complexity in prefix-free regular languages. In: McQuillan, I., Pighizzini, G., Trost, B. (eds.) Proc. 12th DCFS, pp. 236–244. University of Saskatchewan, Saskatoon (2010)
7. Jirásková, G., Okhotin, A.: State complexity of cyclic shift. Theor. Inform. Appl. 42, 335–360 (2008)
8. Krausová, M.: Prefix-free regular languages: Closure properties, difference, and left quotient. In: Kotásek, Z., Bouda, J., Černá, I., Sekanina, L., Vojnar, T., Antoš, D. (eds.) MEMICS 2011. LNCS, vol. 7119, pp. 114–122. Springer, Heidelberg (2012)
9. Maslov, A.N.: Estimates of the number of states of finite automata. Soviet Math. Dokl. 11, 1373–1375 (1970)
10. Maslov, A.N.: The cyclic shift of languages. Problemy Peredači Informacii 9, 81–87 (1973) (Russian)
11. Oshiba, T.: Closure property of the family of context-free languages under the cyclic shift operation. Electron. Commun. Japan 55, 119–122 (1972)
12. Sipser, M.: Introduction to the theory of computation. PWS Publishing Company, Boston (1997)
13. Yu, S.: Regular languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. I, ch. 2, pp. 41–110. Springer, Heidelberg (1997)
14. Yu, S., Zhuang, Q., Salomaa, K.: The state complexity of some basic operations on regular languages. Theoret. Comput. Sci. 125, 315–328 (1994)