On the Square of Regular Languages*

Kristína Čevorová [1], Galina Jirásková [2], and Ivana Krajňáková [3]

[1] Mathematical Institute, Slovak Academy of Sciences
Štefánikova 49, 814 73 Bratislava, Slovakia
cevorova@mat.savba.sk
[2] Mathematical Institute, Slovak Academy of Sciences
Grešákova 6, 040 01 Košice, Slovakia
jiraskov@saske.sk
[3] Institute of Computer Science, Faculty of Science, P.J. Šafárik University
Jesenná 5, 040 01 Košice, Slovakia
ivana.krajnakova.vt@gmail.com

Abstract. We show that the upper bound $(n-k) \cdot 2^n + k \cdot 2^{n-1}$ on the state complexity of the square of a regular language recognized by an n-state deterministic finite automaton with k final states is tight in the ternary case for every k with $1 \leq k \leq n-2$. Using this result, we are able to define a language that is hard for the square operation on languages accepted by alternating finite automata. In the unary case, the known upper bound for square is $2n-1$, and we prove that each value in the range from 1 to $2n-1$ may be attained by the state complexity of the square of a unary language with state complexity n whenever $n \geq 5$.

1 Introduction

Square is an operation on formal languages which is defined as $L^2 = L \cdot L = \{uv \mid u \in L \text{ and } v \in L\}$. It is known that if a regular language L is recognized by an n-state deterministic finite automaton (DFA), then the language L^2 is recognized by a DFA of at most $n \cdot 2^n - 2^{n-1}$ states [11]. This upper bound follows from the upper bound $m \cdot 2^n - 2^{n-1}$ on the state complexity of the concatenation $K \cdot L = \{uv \mid u \in K \text{ and } v \in L\}$ of languages K and L recognized by m-state and n-state DFAs, respectively [9, 14]; here, the state complexity of a regular language is the smallest number of states in any DFA recognizing this language.

Yu *et al.* [14] proved that the upper bound for concatenation is tight in the ternary case by describing languages over a three-letter alphabet that meet this upper bound for their concatenation. The binary witnesses have been presented already in [9], however no proof has been given here. The tightness of this upper bound in the binary case is proved in [5].

In [14] it is shown that the upper bound $m \cdot 2^n - 2^{n-1}$ for concatenation cannot be met if the first language is accepted by an m-state DFA that has more than one final state. In such a case, the upper bound is $(m-k) \cdot 2^n + k \cdot 2^{n-1}$, where k is the number of final states in the DFA for the first language [14].

* Research supported by grant APVV-0035-10.

M. Holzer and M. Kutrib (Eds.): CIAA 2014, LNCS 8587, pp. 136–147, 2014.

The tightness of these bounds has been studied in [4], where binary witnesses are described for every k with $1 \leq k \leq n-1$. Later these results have been useful for defining languages that are hard for concatenation of languages accepted by alternating finite automata (AFAs). The known upper bound for alternating finite automata is $2^m + n + 1$ [3], and the authors of [3] wrote: "... we show that $2^m + n + 1$ states suffice for an AFA to accept the concatenation of two languages accepted by AFA with m and n states, respectively. We conjecture that this number is actually necessary in the worst case, but have no proof."

This open problem is almost solved in [6] by taking binary languages K^R and L^R accepted by 2^m-state and 2^n-state DFAs, respectively, both with half of states final, that meet the upper bound for concatenation in [4]. Then, as shown in [6], the languages K and L are accepted by m-state and n-state AFAs, respectively, and every AFA for the language $K \cdot L$ requires at least 2^m+n states.

Motivated by the same problem for the square operation on alternating finite automata, we study this operation in more detail in this paper. The upper bound $n \cdot 2^n - 2^{n-1}$ on the state complexity of the square of a language recognized by an n-state DFA is known to be tight in the binary case. Rampersad [11] described a language over a binary alphabet recognized by an n-state DFA with one final state whose square meets this upper bound.

As in the case of concatenation, this upper bound cannot be met by a language accepted by an n-state DFA that has more than one final state. Here, the upper bound for concatenation gives the upper bound $(n-k) \cdot 2^n + k \cdot 2^{n-1}$ on the state complexity of the square of a language recognized by an n-state DFA with k final states. In the first part of our paper, we show that these upper bounds are tight in the ternary case for every k with $1 \leq k \leq n-2$. We are not able to prove the tightness in the case of $k = n-1$, and we conjecture that in this case, the upper bound cannot be met. The binary case remains open as well.

Using these results, we are able to describe a language L accepted by an n-state AFA such that every AFA for the language L^2 needs at least $2^n + n$ states. This is smaller just but one than the upper bound $2^n + n + 1$ which follows from the known upper bound $2^m + n + 1$ for concatenation of AFA languages [3].

In the second part of the paper, we study the square operation on unary regular languages. In the unary case, the known upper bound on the state complexity of the square of a language recognized by an n-state unary DFA is $2n - 1$ [11]. We are interested in the question which values in the range from 1 to $2n-1$ may be attained by the state complexity of the square of a unary language with state complexity n. We prove that for every n with $n \geq 5$, the hierarchy of possible complexities is contiguous with no gaps in it. For every n and α with $n \geq 5$ and $1 \leq \alpha \leq 2n-1$, we are able to define a unary language L with state complexity n such that the state complexity of the language L^2 is α. This is in contrast to the results for the star of unary languages [2], where there are at least two gaps of length n of values in the range from 1 to $(n-1)^2 + 1$ that cannot be attained by the star of any unary language with state complexity n.

We first recall some basic definitions; for further details, the reader may refer refer to [12, 13].

A *nondeterministic finite automaton* (NFA) is a quintuple $A = (Q, \Sigma, \delta, I, F)$, where Q is a finite set of states, Σ is a finite alphabet, $\delta\colon Q \times \Sigma \to 2^Q$ is the transition function which is extended to the domain $2^Q \times \Sigma^*$ in the natural way, $I \subseteq Q$ is the set of initial states, and $F \subseteq Q$ is the set of final states. The language accepted by A is the set $L(A) = \{w \in \Sigma^* \mid \delta(I, w) \cap F \neq \emptyset\}$. An NFA A is *deterministic* (and complete) if $|I| = 1$ and $|\delta(q, a)| = 1$ for each q in Q and each a in Σ. In such a case, we write $q \cdot a = q'$ instead of $\delta(q, a) = \{q'\}$.

The state complexity of a regular language L, $\mathrm{sc}(L)$, is the number of states in the minimal DFA for L. It is well known that a DFA is minimal if all its states are reachable from its initial state, and no two of its states are equivalent.

The *concatenation* of two languages K and L is the language $K \cdot L = \{uv \mid u \in K \text{ and } v \in L\}$. The square of a language L is the language $L^2 = L \cdot L$.

The *reverse* of a string w is defined by $\varepsilon^R = \varepsilon$ and $(wa)^R = aw^R$ for a string w and a symbol a. The reverse of a language L is the language $L^R = \{w^R \mid w \in L\}$.

A language is called *unary* (*binary*, *ternary*) if it is defined over an alphabet containing one (two, three, respectively) symbols.

2 Square for Automata with k Final States

In this section we consider languages over an alphabet of at least two symbols. The state complexity of concatenation of regular languages accepted by an m-state and an n-state DFAs is known to be $m \cdot 2^n - 2^{n-1}$ [9, 14]. However, if the first automaton has k final states, then the upper bound for concatenation is $(m-k) \cdot 2^n + k \cdot 2^{n-1}$ [14], and it is known to be tight in the binary case for every k with $1 \leq k \leq n-1$ [4].

It follows that the upper bound on the complexity of square is $n \cdot 2^n - 2^{n-1}$. A binary witness language meeting this bound is presented in [11]. If a language is accepted by an n-state DFA with k final states, then the upper bound is $(n-k) \cdot 2^n + k \cdot 2^{n-1}$. For the sake of completeness, we give a simple alternative proof here.

Lemma 1. *Let $n \geq 2$ and $1 \leq k \leq n-1$. If a language L is accepted by an n-state DFA with k final states, then* $\mathrm{sc}(L^2) \leq (n-k) \cdot 2^n + k \cdot 2^{n-1}$.

Proof. Let L be a language accepted by a DFA $A = (Q, \Sigma, \cdot, 0, F)$, where $Q = \{0, 1, \ldots, n-1\}$ and $|F| = k$. Construct an NFA N for the language L^2 from the DFA A as follows. Take two copies of the DFA A; the states in the first copy are labeled by $q_0, q_1, \ldots, q_{n-1}$, and the states of the second copy are labeled by $0, 1, \ldots, n-1$. For each state q_i and each symbol a, add the transition on a from q_i to the initial state 0 of the second copy whenever $i \cdot a \in F$. The initial state of the NFA N is q_0 if $0 \notin F$, otherwise N has two initial states q_0 and 0. The final states of N are final states in the second copy, thus states in F.

Consider the subset automaton of the NFA N. Each reachable subset of the subset automaton is of the form $\{q_i\} \cup S$, where $S \subseteq \{0, 1, \ldots, n-1\}$. Moreover, if $i \in F$, then S must contain the state 0. It follows that the number of reachable states in the subset automaton is at most $(n-k) \cdot 2^n + k \cdot 2^{n-1}$. □

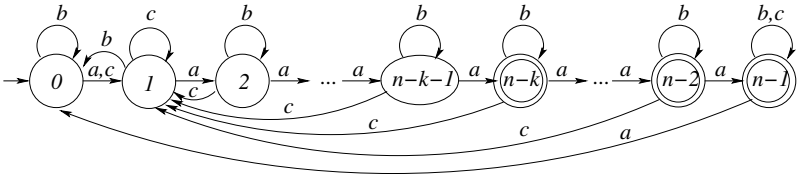

Fig. 1. A DFA A of a language meeting the bound $(n-k)\cdot 2^n + k\cdot 2^{n-1}$ for square

Our next aim is to show that the bounds $(n-k)\cdot 2^n + k\cdot 2^{n-1}$ can be met by languages over a three-letter alphabet assuming that $1 \le k \le n-2$. We are not able to prove the tightness in the case of $k = n-1$, and we conjecture that in this case, the bound $2^n + (n-1)\cdot 2^{n-1}$ cannot be met.

Lemma 2. *Let $n \ge 3$ and $1 \le k \le n-2$. There exists a ternary regular language L accepted by an n-state DFA with k final states and such that* $\mathrm{sc}(L^2) = (n-k)\cdot 2^n + k\cdot 2^{n-1}$.

Proof. Let L be the language accepted by the DFA $A = (Q, \{a,b,c\}, \cdot, 0, F)$ shown in Fig. 1, in which $Q = \{0,1,\ldots,n-1\}$, $F = \{i \mid n-k \le i \le n-1\}$, and

$q\cdot a = (q+1) \bmod n$;

$q\cdot b = q$ if $q \ne 1$ and $1\cdot b = 0$;

$q\cdot c = 1$ if $q \ne n-1$ and $(n-1)\cdot c = n-1$;

notice that the automaton A restricted to the alphabet $\{a,b\}$ and with $k=1$ is the Rampersad's witness automaton meeting the upper bound $n\cdot 2^n - 2^{n-1}$ on the state complexity of the square of regular languages [11].

Construct an NFA N for the language L^2 as described in the proof of Lemma 1. The NFA N is shown in Fig. 2; to keep the figure transparent, we omitted the transitions on c going to states q_1 and 1.

Our goal is to show that the subset automaton corresponding to the NFA N has $(n-k)\cdot 2^n + k\cdot 2^{n-1}$ reachable and pairwise distinguishable states.

To this aim consider the following family of subsets of the states of N:

$$\mathcal{R} = \big\{\{q_i\}\cup S \mid 0 \le i \le n-k-1,\ S \subseteq \{0,1,\ldots,n-1\}\big\}$$
$$\cup\ \big\{\{q_i\}\cup T \mid n-k \le i \le n-1,\ T \subseteq \{0,1,\ldots,n-1\} \text{ and } 0 \in T\big\}.$$

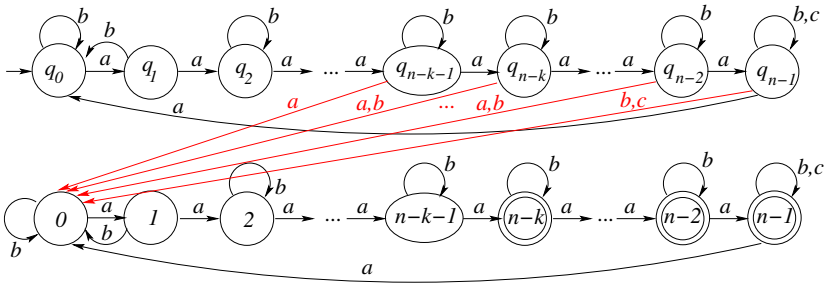

Fig. 2. An NFA N for the language $(L(A))^2$; the transitions on c going to states q_1 and 1 are omitted

The family $\mathcal{R}$ consists of $(n-k)\cdot 2^n + k\cdot 2^{n-1}$ subsets, and we are going to show that all of them are reachable and pairwise distinguishable in the subset automaton of the NFA N.

We prove reachability by induction on the size of subsets. The initial state of the subset automaton is $\{q_0\}$, and the following transitions show that all the subsets in $\mathcal{R}$ of size at most two are reachable:

$$\{q_0\} \xrightarrow{a} \{q_1\} \xrightarrow{a} \cdots \xrightarrow{a} \{q_{n-k-1}\} \xrightarrow{a} \{q_{n-k},0\} \xrightarrow{ab} \{q_{n-k+1},0\} \xrightarrow{ab} \cdots \xrightarrow{ab} \{q_{n-1},0\},$$

$$\{q_{n-1},0\} \xrightarrow{a} \{q_0,1\} \xrightarrow{b} \{q_0,0\},$$

$$\{q_0,1\} \xrightarrow{(ab)^{j-1}} \{q_0,j\} \text{ where } 2\le j\le n-1, \text{ and}$$

$$\{q_0,(j-i) \bmod n\} \xrightarrow{a^i} \{q_i,j\} \text{ where } 1\le i\le n-k-1,\ 0\le j\le n-1.$$

Now let $2\le t\le n$, and assume that each subset in $\mathcal{R}$ of size t is reachable in the subset automaton. Let us show that then also each subset in $\mathcal{R}$ of size $t+1$ is reachable.

To this aim let $S=\{q_i,j_1,j_2,\ldots,j_t\}$, where $0\le j_1<j_2<\cdots<j_t\le n-1$, be a subset in $\mathcal{R}$ of size $t+1$. Consider several cases:

(1) Let $n-k\le i\le n-1$, so $j_1=0$. We show that the set S is reachable by induction on i.

(1a) If $i=n-k$, then S is reached from $\{q_{n-k-1},j_2-1,j_3-1,\ldots,j_t-1\}$ by a, and the latter set is reachable by induction on t.

(1b) Suppose $i>n-k$. If $j_2\ge 2$, then the set S is reached from the set $\{q_{i-1},0,j_2-1,\ldots,j_t-1\}$ by ab. If $j_2=1$, then the set S is reached from the set $\{q_{i-1},n-1,0,j_3-1,\ldots,j_t-1\}$ by a. Both sets containing q_{i-1} are reachable by induction on i.

(2) Let $i=0$. There are four subcases:

(2a) Let $j_1=0$ and $j_2=1$. Take $S'=\{q_{n-1},n-1,0,j_3-1,\ldots,j_t-1\}$. Then S' is reachable as shown in case (1), and it goes to S by a.

(2b) Let $j_1=0$ and $j_2\ge 2$. Take $S'=\{q_{n-1},0,j_2-1,j_3-1,\ldots,j_t-1\}$. Then S' is reachable as shown in case (1), and it goes to S by ab.

(2c) Let $j_1=1$. Take $S'=\{q_{n-1},0,j_2-1,j_3-1,\ldots,j_t-1\}$. Then S' is reachable as shown in case (1), and it goes to S by a.

(2d) Let $j_1\ge 2$. Take $S'=\{q_0,1,j_2-j_1+1,j_3-j_1+1,\ldots,j_t-j_1+1\}$. Then S' is reachable as shown in case (2c), and it goes to S by $(ab)^{j_1-1}$.

(3) Let $1\le i\le n-k-1$. Take $S'=\{q_0,(j_1-i) \bmod n,\ldots,(j_t-i) \bmod n\}$. Then S' is reachable as shown in cases (2a-2d), and it goes to S by a^i.

This proves the reachability of all the subsets in $\mathcal{R}$.

To prove distinguishability, notice that the string c is accepted by the NFA N only from the state $n-1$; remind that state 1 is not final since we have $k\le n-2$. Next, notice that exactly one transition on a goes to each of the states in $\{q_1,q_2,\ldots,q_{n-1},1,2,\ldots,n-1\}$, and exactly one transition on c goes to state 0. It follows that the string $a^{n-1-i}c$ is accepted only from the state i,

where $0 \leq i \leq n-2$, the string $ca^{n-1}c$ is accepted only from the state q_{n-1}, and finally the string $a^{n-1-i}ca^{n-1}c$ is accepted only from the state q_i, where $0 \leq i \leq n-2$. Fig. 3 illustrates this for $n = 5$ and $k = 3$. This means that all the states in the subset automaton of the NFA N are pairwise distinguishable since two distinct subsets must differ in a state q of N, and the string that is accepted only from q distinguishes the two subsets. This proves distinguishability, and concludes the proof. □

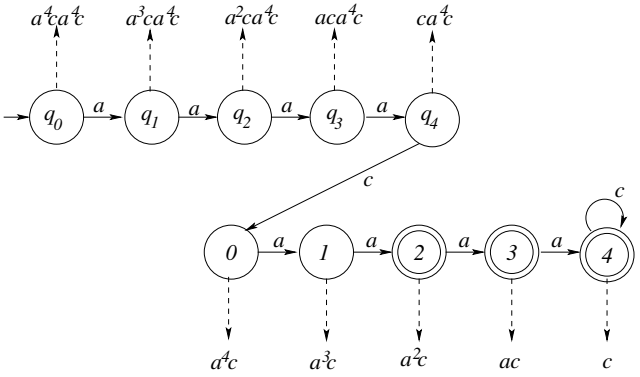

Fig. 3. The strings accepted only from the corresponding states; $n = 5$ and $k = 3$. Notice that exactly one transition on a goes to each of the states in $\{q_1, q_2, \ldots, q_{n-1}, 1, 2, \ldots, n-1\}$, and exactly one transition on c goes to state 0. The unique acceptance of appropriate strings follows from these facts.

As a corollary of the two lemmata above, we get the following result.

Theorem 1 (Square: k Final States). *Let $n \geq 3$ and $1 \leq k \leq n-2$. Let L be a language over an alphabet Σ accepted by an n-state DFA with k final states. Then $\mathrm{sc}(L^2) \leq (n-k) \cdot 2^n + k \cdot 2^{n-1}$, and the bound is tight if $|\Sigma| \geq 3$.* □

2.1 An Application

In this subsection we show how the witness languages described in Lemma 2 can be used to define languages that almost meet the upper bound on the square operation on alternating finite automata.

First, let us give some basic definitions and notations. For details and all unexplained notions, the reader may refer to [1, 3, 6–8, 12, 13].

An *alternating finite automaton* (AFA) is a quintuple $A = (Q, \Sigma, \delta, s, F)$, where Q is a finite non-empty set of states, $Q = \{q_1, \ldots, q_n\}$, Σ is an input alphabet, δ is the transition function that maps $Q \times \Sigma$ into the set $\mathcal{B}_n$ of boolean functions, $s \in Q$ is the initial state, and $F \subseteq Q$ is the set of final states. For example, let $A_1 = (\{q_1, q_2\}, \{a, b\}, \delta, q_1, \{q_2\})$, where transition function δ is given in Table 1.

Table 1. The transition function of the alternating finite automaton A_1

δ	a	b
q_1	$q_1 \wedge q_2$	1
q_2	q_2	$q_1 \vee \overline{q_2}$

The transition function δ is extended to the domain $\mathcal{B}_n \times \Sigma^*$ as follows: For all g in $\mathcal{B}_n$, a in Σ, and w in Σ^*,

$$\begin{aligned}
&\delta(g, \varepsilon) = g;\\
&\text{if } g = g(q_1, \ldots, q_n), \text{ then } \delta(g, a) = g(\delta(q_1, a), \ldots, \delta(q_n, a));\\
&\delta(g, wa) = \delta(\delta(g, w), a).
\end{aligned}$$

Next, let $f = (f_1, \ldots, f_n)$ be the boolean vector with $f_i = 1$ iff $q_i \in F$. The language accepted by the AFA A is the set $L(A) = \{w \in \Sigma^* \mid \delta(s, w)(f) = 1\}$. In our example we have

$$\delta(s, ab) = \delta(q_1, ab) = \delta(\delta(q_1, a), b) = \delta(q_1 \wedge q_2, b) = 1 \wedge (q_1 \vee \overline{q_2}) = q_1 \vee \overline{q_2}.$$

To determine whether $ab \in L(A_1)$, we evaluate $\delta(s, ab)$ at the vector $f = (0, 1)$. We obtain 0, hence $ab \notin L(A_1)$. On the other hand, we have $abb \in L(A_1)$ since $\delta(s, abb) = \delta(q_1 \vee \overline{q_2}, b) = 1 \vee (\overline{q_1} \wedge q_2)$, which gives 1 at $(0, 1)$.

An alternating finite automaton A is nondeterministic (NFA) if $\delta(q_k, a)$ are of the form $\bigvee_{i \in I} q_i$. If $\delta(q_k, a)$ are of the form q_i, then the automaton A is deterministic (DFA).

Recall that the state complexity of a regular language L, $\mathrm{sc}(L)$, is the smallest number of states in any DFA accepting L. Similarly, the alternating state complexity of a language L, in short $\mathrm{asc}(L)$, is defined as the smallest number of states in any AFA for L. The following results are well known.

Lemma 3 ([1, 3, 6, 7]). *If L is accepted by an AFA of n-states, then L^R is accepted by a DFA of 2^n states. If $\mathrm{sc}(L^R) = 2^n$ and the minimal DFA for L^R has 2^{n-1} final states, then $\mathrm{asc}(L) = n$.* □

It follows that $\mathrm{asc}(L) \geq \log(\mathrm{sc}(L^R))$. Using the results given by Lemma 2 and Lemma 3, we get a language that almost meets the upper bound on the complexity of the square operation on AFAs.

Theorem 2 (Square on AFAs). *Let L be a language over an alphabet Σ with $\mathrm{asc}(L) = n$. Then $\mathrm{asc}(L^2) \leq 2^n + n + 1$. The bound $2^n + n$ is met if $|\Sigma| \geq 3$.*

Proof. The upper bound follows from the upper bound $2^m + n + 1$ on the concatenation of AFA languages [3]. Now let L^R be the ternary witness for square from Lemma 2 with 2^n states and 2^{n-1} final states. Then, by Lemma 3, we have $\mathrm{asc}(L) = n$. By Lemma 2, we get

$$\mathrm{sc}((L^2)^R) = \mathrm{sc}((L^R)^2) = 2^{n-1} \cdot 2^{2^n} + 2^{n-1} \cdot 2^{2^n - 1} \geq 2^{n-1} \cdot 2^{2^n}(1 + 1/2).$$

By Lemma 3, we have $\mathrm{asc}(L^2) \geq \lceil \log(2^{n-1} \cdot 2^{2^n}(1 + 1/2)) \rceil = 2^n + n$, which proves the theorem. □

The above result for square complements the results on the complexity of basic operations on AFA languages obtained in [6]. The following table summarizes these results, and compares them to the known results for DFAs [9, 11, 14].

	union	intersection	concatenation	reversal	star	square
AFAs	$m+n+1$	$m+n+1$	$\geq 2^m+n$ $\leq 2^m+n+1$	$\geq 2^n$ $\leq 2^n+1$	$\geq 2^n$ $\leq 2^n+1$	$\geq 2^n+n$ $\leq 2^n+n+1$
DFAs	mn	mn	$m \cdot 2^n - 2^{n-1}$	2^n	$3/4 \cdot 2^n$	$n \cdot 2^n - 2^{n-1}$

3 Square of Languages over Unary Alphabet

A unary alphabet is fundamentally different from the general case. It has a close relation to the number theory – the length of strings is their only property that really matters in complexity questions. From this point of view, unary languages are nothing else than subsets of natural numbers. Instead of writing $a^n \in L$, we will write $n \in L$. The square operation is then, in fact, the sum of two numbers in the language. Let us start with some basic definitions and notations.

For integers i and j with $i \leq j$, let $[i,j] = \{i, i+1, \ldots, j\}$.

A DFA $A = (Q, \{a\}, \delta, q_0, F)$ for a unary language is uniquely given by less information than an arbitrary DFA. Identify states with numbers from the interval $[0, n-1]$ via $q \sim \min\{i \mid \delta(q_0, a^i) = q\}$. Then A is unambiguously given by the number of states n, the set of final numbers F, and the "loop" number $\ell = \delta(q_0, a^n)$. This allows us to freely interchange states and their ordinal numbers and justifies the notation convention used by Nicaud [10], where (n, ℓ, F) denotes a unary automaton with n states, the loop number ℓ, and the set of final states F. Nicaud also provided the following characterization of minimal unary automata.

Theorem 3 ([10, Lemma 1]). *A unary automaton (n, ℓ, F) is minimal if and only if both conditions below are true:*

(1) its loop is minimal, and

(2) states $n-1$ and $\ell-1$ do not have the same finality (that is, exactly one of them is final). □

Finite and cofinite languages are always regular, and if they are unary, then it is easy to determine their state complexity.

Proposition 1. *Let L be a unary language. If L is cofinite, then we have* $\mathrm{sc}(L) = \max\{m \mid m \notin L\} + 2$. *If L is finite, then* $\mathrm{sc}(L) = \max\{m \mid m \in L\} + 2$. □

Proposition 2. *If a language is (co)finite, then also its square is (co)finite.* □

If $\varepsilon \in L$, then every string w in L can be written as εw. This leads to the following observation.

Proposition 3. *If $\varepsilon \in L$, then $L \subseteq L^2$.* □

3.1 Finite Unary Languages

Interestingly, if we consider only finite languages with state complexity n, then we cannot get any other complexity for square but $2n-2$.

Lemma 4. *Let L be a finite unary regular language with $\mathrm{sc}(L) = n$. Then $\mathrm{sc}(L^2) = 2n-2$.*

Proof. By Proposition 1, the greatest number in L is $n-2$. It follows that the greatest member of L^2 is the number $2n-4$. Hence L^2 is also finite and $\mathrm{sc}(L^2) = 2n-4+2 = 2n-2$. □

3.2 General Unary Languages

If we take a unary language with state complexity n, the state complexity of its square will be between 1 and $2n-1$ [11]. But could it be *anywhere* between these two bounds? The next result shows that the answer is yes if $n \geq 5$.

Theorem 4. *Let $n \geq 5$ and $1 \leq \alpha \leq 2n-1$. There exists a unary language L such that $\mathrm{sc}(L) = n$ and $\mathrm{sc}(L^2) = \alpha$.*

Proof. We will provide a witness for every liable combination of n and α. The proof is structured to four main cases depending on α:

1. $\alpha = 2$ (the proof works for $n \geq 6$),
2. $\alpha = 2n-2$ (the proof works for $n \geq 2$),
3. $1 \leq \alpha \leq n-1$ and $\alpha \neq 2$ (the proof works for $n \geq 8$),
4. $n \leq \alpha \leq 2n-1$ and $\alpha \neq 2n-2$ (the proof works for $n \geq 2$).

All witnesses uncovered by these general proofs are part of Table 2 which is an overview of the situation for $n < 5$ and $\alpha \leq n$. If the combination of n and α is liable, one witness is listed, non-existence is indicated by the symbol –.

Table 2. Witnesses for liable combinations of small values of n and α; $2 \leq n \leq 7$ and $1 \leq \alpha \leq n$

α	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$
1	–	$(3,0,\{0,1\})$	$(4,3,\{0,1,3\})$	$(5,4,\{0,1,2,4\})$	$(6,5,\{0,1,2,3,5\})$	$(7,6,\{0,1,2,3,4,6\})$
2	$(2,0,\{0\})$	–	–	$(5,1,\{0,2\})$	$(6,0,\{0,2\})$	$(7,5,\{0,2,6\})$
3		$(3,0,\{0\})$	$(4,2,\{1,2\})$	$(5,0,\{0,2,3\})$	$(6,5,\{0,2,3,5\})$	$(7,6,\{0,2,3,4,6\})$
4			$(4,0,\{0\})$	$(5,0,\{0,3,4\})$	$(6,2,\{0,3,4,5\})$	$(7,0,\{0,3,4,5\})$
5				$(5,0,\{0\})$	$(6,2,\{0,2,5\})$	$(7,6,\{0,1,4,6\})$
6					$(6,0,\{0\})$	$(7,1,\{0,1,3\})$
7						$(7,0,\{0\})$

1. Let $\alpha = 2$ and $n \geq 6$. The construction of witnesses depends on the parity of n. If n is even, then we take the language recognized by the witness DFA $A = (n, 0, F)$, where $F = \{i \in [0, n-1] \mid i \text{ is even and } i \neq n-2\}$. If n is odd, then the witness DFA is $A = (n, n-2, \{i \in [0, n-1] \mid i \text{ is even and } i \neq n-3\})$.

We claim that in both cases $L(A)^2$ is the language of even numbers with the corresponding minimal DFA $(2, 0, \{0\})$. We give the proof for n even; the proof for n odd has only slight technical differences.

We first show that A is minimal. The second condition of Theorem 3 is fulfilled vacantly. The first condition – the minimality of the loop – is satisfied as well: Any equivalent loop must be of even length as not to accept strings of different parity. Since there is exactly one even non-accepting state, it cannot be equivalent with any other state, and the loop is unfactorizable.

Now we show that $L(A)^2$ is the language of all even numbers. Since $L(A)$ contains only even numbers and the sum of two even numbers is even, $L(A)^2$ contains only even numbers. Let us show that $L(A)^2$ contains *all* even numbers.

By Proposition 3, we have $L(A) \subseteq L(A)^2$. All even numbers missing in $L(A)$ are in the form $kn+(n-2)$. But these numbers are in $L(A)^2$, since $kn+(n-2) = (2+kn)+(n-4)$, which is a sum of two numbers in $L(A)$; recall, that $2 \neq n-2$, since $n \geq 6$.

2. Let $\alpha = 2n-2$ and $n \geq 2$. By Lemma 4, every finite language of complexity n is a witness in this case; for example, we can take the language $\{n-2\}$.

From now on, our strategy is based on Proposition 1. All our languages will be cofinite, so their complexity is easily determined by answering the question – how long is the longest string not contained in this language?

3. Let $1 \leq \alpha \leq n-1$, $\alpha \neq 2$, and $n \geq 8$. Technically, this case is further divided to subcases $\alpha = 1$, $\alpha = 3$, $\alpha = 4$, $5 \leq \alpha \leq n$ where $\alpha \neq n-1$, and $\alpha = n-1$. However, the main idea of the construction is always the same, so we provide only the witness automata in Fig. 4, and one exemplary proof in the case of $5 \leq \alpha \leq n$ where $\alpha \neq n-1$. For this case, consider the language L accepted by the unary automaton $A = (n, n-1, F)$, where $F = [\alpha-1, n-3] \cup \{0, 1, n-1\}$.

First, let us show that the numbers greater than $\alpha-2$ are in L^2. Since $\varepsilon \in L$, the language L is a subset of L^2 by Proposition 3. The only number greater than $\alpha-2$ that is not in L is $n-2$. However, we have $n-2 = (n-3)+1$, which is the sum of two numbers in L. Therefore, the number $n-2$ is in L^2. It follows that all numbers greater than $\alpha-2$ are in L^2.

Now let us show that $\alpha-2$ is not in L^2. The only numbers in L that are smaller than $\alpha-1$ are 0 and 1. The sum of any two of them is at most 2, which is less than $\alpha-2$. Thus, by Proposition 1, the state complexity of L^2 is α.

4. $n \leq \alpha \leq 2n-1$ and $\alpha \neq 2n-2$, $n \geq 2$. Consider the unary language $L = \{i \mid i \geq n-1\}$. Then $\mathrm{sc}(L) = n$ and $\mathrm{sc}(L^2) = 2n-1$ since the greatest number that is not in L^2 is $2n-3$. By adding an arbitrary number different than $n-2$ to the language L, we get a language with the same state complexity as L. But the state complexity of the square of the resulting language will be

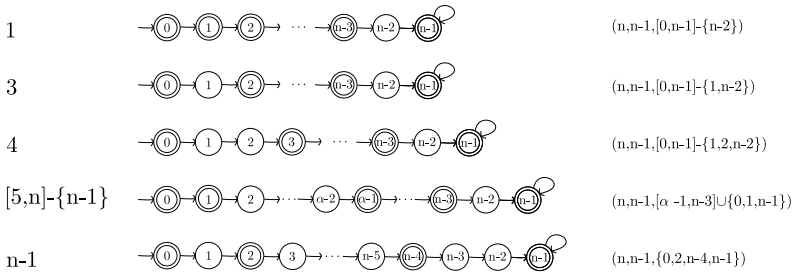

Fig. 4. The construction of witnesses for n and α with $1 \leq \alpha \leq n$

different. Let $m = \alpha - n$. Then $0 \leq m \leq n-1$ and $m \neq n-2$. Let us see what happens, if we add the number m to L.

Let $L_m = L \cup \{m\}$. Then we have $L_m^2 = \{2m\} \cup \{m+i \mid i \geq (n-1)\}$. Since $m \neq n-2$, we have $2m \neq m+n-2$, and therefore the greatest number that is not in L_m^2 is $m+n-2$. It follows that $\mathrm{sc}(L_m^2) = m+n-2+2 = \alpha$. □

4 Conclusions

We considered the square operation on regular languages. In the unary case, the state complexity of square is $2n-1$ [11]. We proved that each value in the range from 1 to $2n-1$ may be attained by the state complexity of the square of a unary language with state complexity n whenever $n \geq 5$.

Next, we studied the square operation on languages over an alphabet of at least two symbols. The known upper bound in this case is $n \cdot 2^n - 2^{n-1}$, and it is known to be tight in the binary case [11]. We investigated the square for languages accepted by automata with more final states. The upper bound on the state complexity of the square of a language accepted by an n-state DFA with k final states is $(n-k) \cdot 2^n + k \cdot 2^{n-1}$. We showed that these upper bounds are tight in the ternary case assuming that $1 \leq k \leq n-2$.

The case of $k = n-1$ remains open, and we conjecture that the upper bound $2^n + (n-1) \cdot 2^{n-1}$ cannot be met in this case. The binary case is open as well.

As an application, we were able to define a ternary language L accepted by an n-state alternating finite automaton such that every alternating finite automaton for the language L^2 requires at least $2^n + n$ states. This is smaller just by one than the known upper bound $2^n + n + 1$ [3]. Our result on the square operation complements the results on the complexity of union, intersection, concatenation, star, and reversal on AFA languages obtained in [6].

References

1. Brzozowski, J., Leiss, E.: On equations for regular languages, finite automata, and sequential networks. Theoret. Comput. Sci. 10, 19–35 (1980)
2. Čevorová, K.: Kleene star on unary regular languages. In: Jurgensen, H., Reis, R. (eds.) DCFS 2013. LNCS, vol. 8031, pp. 277–288. Springer, Heidelberg (2013)

3. Fellah, A., Jürgensen, H., Yu, S.: Constructions for alternating finite automata. Int. J. Comput. Math. 35, 117–132 (1990)
4. Jirásek, J., Jirásková, G., Szabari, A.: State complexity of concatenation and complementation. Internat. J. Found. Comput. Sci. 16, 511–529 (2005)
5. Jirásková, G.: State complexity of some operations on binary regular languages. Theoret. Comput. Sci. 330, 287–298 (2005)
6. Jirásková, G.: Descriptional complexity of operations on alternating and boolean automata. In: Hirsch, E.A., Karhumäki, J., Lepistö, A., Prilutskii, M. (eds.) CSR 2012. LNCS, vol. 7353, pp. 196–204. Springer, Heidelberg (2012)
7. Leiss, E.: Succinct representation of regular languages by boolean automata. Theoret. Comput. Sci. 13, 323–330 (1981)
8. Leiss, E.: On generalized language equations. Theoret. Comput. Sci. 14, 63–77 (1981)
9. Maslov, A.N.: Estimates of the number of states of finite automata. Soviet Math. Doklady 11, 1373–1375 (1970)
10. Nicaud, C.: Average state complexity of operations on unary automata. In: Kutyłowski, M., Wierzbicki, T., Pacholski, L. (eds.) MFCS 1999. LNCS, vol. 1672, pp. 231–240. Springer, Heidelberg (1999)
11. Rampersad, N.: The state complexity of L^2 and L^k. Inform. Process. Lett. 98, 231–234 (2006)
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, 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)