International Journal of Foundations of Computer Science
© World Scientific Publishing Company

NONDETERMINISTIC DESCRIPTIONAL COMPLEXITY OF REGULAR LANGUAGES

MARKUS HOLZER
Institut für Informatik, Technische Universität München
Boltzmannstraße 3, D-85748 Garching bei München, Germany
holzer@informatik.tu-muenchen.de

and

MARTIN KUTRIB
Institut für Informatik, Universität Giessen
Arndtstraße 2, D-35392 Giessen, Germany
kutrib@informatik.uni-giessen.de

Received (received date)
Revised (revised date)
Communicated by Editor's name

ABSTRACT

We investigate the descriptional complexity of operations on finite and infinite regular languages over unary and arbitrary alphabets. The languages are represented by nondeterministic finite automata (NFA). In particular, we consider Boolean operations, catenation operations – concatenation, iteration, λ-free iteration – and the reversal. Most of the shown bounds are tight in the exact number of states, i.e. the number is sufficient and necessary in the worst case. Otherwise tight bounds in the order of magnitude are shown.

Keywords: state complexity, language operations, nondeterministic finite automata.

1. Introduction

Finite automata are used in several applications and implementations in software engineering, programming languages and other practical areas in computer science. They are one of the first and most intensely investigated computational models. Nevertheless, some challenging problems of finite automata are still open. An important example is the question how many states are sufficient and necessary to simulate two-way nondeterministic finite automata with two-way deterministic finite automata. The problem has been raised in [19] and partially solved in [22]. A lower bound and an interesting connection with the open problem whether DLOGSPACE equals NLOGSPACE or not is given in [1].

Since regular languages have many representations in the world of finite automata it is natural to investigate the succinctness of their representation by different types of automata in order to optimize the space requirements. It is well known that nondeterministic finite automata (NFA) can offer exponential saving in space compared with deterministic finite automata (DFA), but the problem to convert a given DFA to an equivalent minimal NFA is PSPACE-complete [10]. Since minimization of NFAs is also PSPACE-complete, conversions from nondeterministic to deterministic variants are of particular interest. Concerning the number of states asymptotically tight bounds are $O(n^n)$ for the two-way DFA to one-way DFA conversion, $O(2^{n^2})$ for the two-way NFA to one-way DFA conversion, and 2^n for the one-way NFA to one-way DFA conversion, for example. For finite languages over a k-letter alphabet the NFA to DFA conversion has been solved in [20] with a tight bound of $O(k^{\frac{n}{\log_2 k+1}})$. A valuable source for further results and references is [3].

Related to these questions are the costs (in terms of states) of operations on regular languages with regard to their representing devices. For example, converting a given NFA to an equivalent DFA gives an upper bound for the NFA state complexity of complementation. In recent years results for many operations have been obtained. DFAs state-of-the-art surveys can be found in [26, 27].

When certain problems are computationally hard in general, a natural question concerns simpler versions. To this regard promising research has been done for unary languages. It turned out that this particular case is essentially different from the general case. For example, the minimization of NFAs becomes NP-complete instead of PSPACE-complete [9, 23]. The problem of evaluating the costs of unary automata simulations has been raised in [22]. In [7] it has been shown that the unary NFA to DFA conversion takes $e^{\Theta(\sqrt{n\cdot\ln(n)})}$ states, the NFA to two-way DFA conversion has been solved with a bound of $O(n^2)$ states, and the costs of the unary two-way to one-way DFA conversion reduces to $e^{\Theta(\sqrt{n\cdot\ln(n)})}$. Several more results can be found in [14, 15].

State complexity results concerning operations on unary regular languages represented by DFAs are covered by the surveys [26, 27]. Estimations of the average state complexity are shown in [17].

Here we investigate the costs of operations on finite and infinite regular languages over unary and arbitrary alphabets represented by NFAs. In particular, we consider Boolean operations, catenation operations and the reversal. Most of the bounds are tight in the exact number of states, i.e. the number is sufficient and necessary in the worst case. Otherwise tight bounds in the order of magnitude are shown. The technical depth of our results varies from immediate to more subtle extensions to previous work. Indeed the technique to prove minimality for DFAs is not directly applicable to the case of NFAs. Therefore, we mostly have to start from scratch or to use counting arguments to prove our results on NFA minimality with respect to the number of states.

In the next section we define the basic notions and present a preliminary result. Section 3 is devoted to the study of infinite languages. Operations on NFAs accepting finite languages are considered in Section 4.

2. Preliminaries

We denote the powerset of a set S by 2^S. The empty word is denoted by λ, the reversal of a word w by w^R, and for the length of w we write $|w|$. For the number of occurrences of a symbol a in w we use the notation $\#_a(w)$. By $\gcd(x_1,\ldots,x_k)$ we denote the *greatest common divisor* of the integers $x_1,\ldots,x_k$, and by $\operatorname{lcm}(x_1,\ldots,x_n)$ their *least common multiple.* If two numbers x and y are relatively prime (i.e. $\gcd(x,y)=1$) we write $x \perp y$.

A *nondeterministic finite automaton* is a system $\mathcal{A} = \langle S, A, \delta, s_0, F\rangle$, where S is the finite set of *internal states*, A is the finite set of *input symbols*, $s_0 \in S$ is the *initial state*, $F \subseteq S$ is the set of *accepting states*, and $\delta : S \times A \to 2^S$ is the *transition function.* The set of rejecting states is implicitly given by the partitioning, i.e. $S \setminus F$.

An NFA is called *unary* if its set of input symbols is a singleton. In this case we use $A = \{a\}$ throughout the paper. If not otherwise stated we assume that the NFAs are always *reduced.* This means that there are no unreachable states and that from any state an accepting state can be reached. An NFA is said to be *minimal* if its number of states is minimal with respect to the accepted language. Since every n-state NFA with λ-transitions can be transformed to an equivalent n-state NFA without λ-transitions [8] for state complexity issues there is no difference between the absence and presence of λ-transitions. For convenience, we consider NFAs without λ-transitions only.

As usual the transition function δ is extended to a function $\Delta : S \times A^* \to 2^S$ reflecting sequences of inputs: $\Delta(s,\lambda) = \{s\}$ and $\Delta(s, wa) = \bigcup_{s' \in \Delta(s,w)} \delta(s', a)$ for $s \in S$, $a \in A$, and $w \in A^*$. In the sequel we always denote the extension of a given δ by Δ.

Let $\mathcal{A} = \langle S, A, \delta, s_0, F\rangle$ be an NFA, then a word $w \in A^*$ is *accepted* by $\mathcal{A}$ if $\Delta(s_0, w) \cap F \neq \emptyset$. The *language accepted* by $\mathcal{A}$ is $L(\mathcal{A}) = \{w \in A^* \mid w \text{ is accepted by } \mathcal{A}\}$.

The following preliminary result is a key tool in the following sections, and can be proved by a simple pumping argument.

Lemma 1 *Let $n \geq 1$ be an arbitrary integer. Then $n+1$ resp. n states are sufficient and necessary in the worst case for an* NFA *to accept the language $\{a^n\}^+$ resp. $\{a^n\}^*$.*

3. Operations on Infinite Languages

3.1. Boolean Operations

We start our investigations with Boolean operations. In the case when the finite automaton is deterministic it is well-known that in the worst case the Boolean operations union, intersection and complementation have a state complexity of $m \cdot n$, $m \cdot n$ and m, respectively. However, the state complexity of NFA operations is essentially different. At first we consider the union for arbitrary alphabets.

Theorem 1 *For any integers $m, n \geq 1$ let $\mathcal{A}$ be an m-state and $\mathcal{B}$ be an n-state* NFA*. Then $m+n+1$ states are sufficient and necessary in the worst case for an* NFA *to accept the language $L(\mathcal{A}) \cup L(\mathcal{B})$.*

Proof. In order to construct an $(m+n+1)$-state NFA for the language $L(\mathcal{A}) \cup L(\mathcal{B})$ we simply use a new initial state and connect it to the states of $\mathcal{A}$ and $\mathcal{B}$ that are reached after the first state transition.

Now we are going to show that $m+n+1$ states are necessary in the worst case. Let $\mathcal{A}$ be an m-state NFA that accepts the language $\{a^m\}^*$ and $\mathcal{B}$ an n-state NFA that accepts $\{b^n\}^*$.

Let $\mathcal{C}$ be an NFA for the language $L(\mathcal{A}) \cup L(\mathcal{B})$. In order to reject the inputs a^i, $1 \leq i \leq m-1$, but to accept the input a^m the NFA $\mathcal{C}$ needs at least $m-1$ non-accepting states $s_1, \ldots, s_{m-1}$ from each of which an accepting state is reachable. Similarly, $\mathcal{C}$ needs at least $n-1$ states $s'_1, \ldots, s'_{n-1}$ for processing the inputs b^i, $1 \leq i \leq n-1$.

Denote by P_a resp. P_b the sets of states that are reachable by inputs of the form a^j resp. b^j for $j \geq 1$. None of the accepting states may be reachable from the states in $P_a \cap P_b$. Otherwise words of the form $a^j b^k$ or $b^j a^k$ would be accepted.

It follows that neither the s_i nor the s'_i may belong to the intersection $P_a \cap P_b$. But, trivially, they do belong to P_a resp. to P_b. Now consider all inputs $\{a^m\}^+$. There must exist an accepting state $s_m \in P_a$ which is reached for infinitely many of them. If one of the states $s_1, \ldots, s_{m-1}$ is reachable from s_m, then s_m may not belong to $P_a \cap P_b$ because otherwise inputs of the form $a^j b^k$ would be accepted. If none of the states $s_1, \ldots, s_{m-1}$ is reachable from s_m, then there must exist another state $s_{m+1} \in P_a$ which is different from $s_1, \ldots, s_{m-1}$. Otherwise there must exist $i_0 \geq 1$ such that $\mathcal{C}$ is in some state $s \in \{s_1, \ldots, s_{m-1}\}$ after processing the input $a^{i_0 \cdot m}$. But from state s an accepting state is reachable by processing input a^{j_0} for some $1 \leq j_0 \leq m-1$. This implies that $a^{i_0 \cdot m + j_0}$ would be accepted. The same holds for states $s'_n \in P_b$ and $s'_{n+1} \in P_b$ for the inputs $\{b^n\}^+$. It follows either s_m (s'_n) does not belong to $P_a \cap P_b$ or there exists another state s_{m+1} (s'_{n+1}). In any case there exist at least two states different from $s_1, \ldots, s_{m-1}, s'_1, \ldots, s'_{n-1}$.

Finally, the initial state s_0 must be an accepting state since $\lambda \in L(\mathcal{A}) \cup L(\mathcal{B})$, but s_0 cannot be part of a loop since otherwise inputs of the form $a^j b^k$ or $b^j a^k$ would be accepted. Altogether, $P_a \cup P_b$ must contain at least $m+n$ different states that are not equal to the initial state. □

Now we turn to the union of unary NFAs. Due to the lack of suited tools and methods such as unique minimization etc. the proof of the lower bound refers specifically to the structures of the witness automata.

Theorem 2 *For any integers $m, n \geq 1$ let $\mathcal{A}$ be a unary m-state and $\mathcal{B}$ be a unary n-state* NFA*. Then $m+n+1$ states are sufficient for an* NFA *to accept the language $L(\mathcal{A}) \cup L(\mathcal{B})$. If neither m is a multiple of n nor n is a multiple of m, then there exist a unary m-state* NFA *$\mathcal{A}$ and a unary n-state* NFA *$\mathcal{B}$ (with the same input symbol) such that any* NFA *accepting $L(\mathcal{A}) \cup L(\mathcal{B})$ needs at least $m+n+1$ states.*

Proof. The upper bound follows immediately from the construction given in the proof of Theorem 1. In order to prove the lower bound without loss of generality we may assume $m > n$. Since m is not a multiple of n we obtain $n > \gcd(m, n)$.

Let $\mathcal{A}$ be an m-state NFA that accepts the language $\{a^m\}^*$ and $\mathcal{B}$ be an n-state NFA that accepts $\{a^n\}^*$. Let $\mathcal{C}$ be an NFA with initial state s_0 for the language $L(\mathcal{A}) \cup L(\mathcal{B})$. In order to prove that $\mathcal{C}$ has at least $m + n + 1$ states assume that it has at most $m + n$ states.

At first we consider the input a^{m+n} and show that it does not belong to $L(\mathcal{C})$. Since $2m > m + n > m$ the input does not belong to $L(\mathcal{A})$. If it would belong to $L(\mathcal{B})$, then there were a constant $c > 2$ such that $m + n = c \cdot n$. Therefore, $m = (c-1)\cdot n$ and, thus, m would be a multiple of n what contradicts the assumption of the theorem.

The next step is to show that each of the states $s_0 \vdash s_1 \vdash \cdots \vdash s_{m-1} \vdash s_m$ which are passed through when accepting the input a^m either is not in a cycle or is in a cycle of length m. To this end assume contrarily some state s_i is in a cycle of length $x \neq m$, i.e. s_i is reachable from s_i by processing x input symbols. Due to the number of states we may assume $x \leq m + n$.

By running several times through the cycle a^{m+x}, a^{m+2x} and a^{m+3x} are accepted.

We observe $m + x$ is not a multiple of m. If it is not a multiple of n we are done. Otherwise, x cannot be a multiple of n since m is not a multiple of n. Moreover, now we observe $m + 2x$ is not a multiple of n since $m + x$ is and x is not. If $m + 2x$ is not a multiple of of m we are done.

Otherwise we consider $m + 3x$ which now cannot be a multiple of m since $m + 2x$ is and x is not. If $m + 3x$ is not a multiple of n we are done.

Otherwise we summarize the situation: $m + 3x$ and $m + x$ are multiples of n which implies $2x$ is a multiple of n. Since $m + 2x$ is a multiple of m we conclude that $2x$ is a multiple of m, too. Moreover, x is neither a multiple of m nor of n.

From $x \leq m + n < 2m \Longrightarrow 2x < 4m$ we derive $2x \in \{m, 3m\}$. If $2x = m$, then m is a multiple of n, a contradiction. So let $2x = 3m$. Since $2x$ is a multiple of n there exists a constant $c \in \mathbb{N}$ such that $3m = cn$. From $x \leq m + n$ follows $\frac{3}{2}m \leq m + n$. Therefore $\frac{1}{2}m \leq n$ which together with $n < m$ implies $c \in \{4, 5, 6\}$.

It holds $\frac{3}{c}m = n$. On the other hand, $m + x = m + \frac{3}{2}m = \frac{5}{2}m$ is a multiple of n. Therefore $\frac{5}{2}/\frac{3}{c} = \frac{5\cdot c}{6}$ must belong to $\mathbb{N}$ for $c \in \{4, 5, 6\}$. Thus $c = 6$, but in this case $\frac{1}{2}m = n \Longrightarrow m = 2n$ and m is a multiple of n what contradicts the assumption of the theorem.

In order to complete the proof of the theorem now we come back to the sequence of states $s_0 \vdash s_1 \vdash \cdots \vdash s_{m-1} \vdash s_m$ passed through when accepting the input a^m. Correspondingly let $s_0 \vdash s'_1 \vdash \cdots \vdash s'_{n-1} \vdash s'_n$ be an accepting sequence for a^n.

Since $L(\mathcal{C})$ is an infinite language $\mathcal{C}$ must contain some cycle. If a s_i from $s_0, \ldots, s_m$ is in a cycle, then its length is m states. State s_i may not be in the sequence $s_0 \vdash s'_1 \vdash \cdots \vdash s'_n$ because in this case a^{m+n} would be accepted. In order to accept a^n but to reject $a^1, \ldots, a^{n-1}$ in any case, the states $s_0, s'_1, \ldots, s'_n$ must be pairwise different. Altogether this results in at least $n + 1 + m$ states.

If on the other hand a s_j from $s_0, s'_1, \ldots, s'_n$ is in a cycle, then it may not be in $s_0, s_1, \ldots, s_m$. Otherwise the cycle length must be m and a^{m+n} would be accepted.

Concluding as before this case results in at least $m+n+1$ states, too.

Finally, if neither a state from $s_0, s_1, \ldots, s_m$ nor a state from $s'_1, \ldots, s'_n$ is in a cycle, then obviously $s_0, \ldots, s_m$ must be pairwise different, but all states s'_j may or may not appear in $s_0, \ldots, s_m$. This takes at least $m+1$ states. So there remain at most $n-1$ states for a cycle. Therefore, the cycle length x is at most $n-1$. Now we consider an accepting computation for the input a^{mn}. It must have run through the cycle. Running once more through the cycle leads to acceptance of a^{mn+x} which does not belong to $L(\mathcal{C})$.

So in any case we obtain a contradiction to the assumption that $\mathcal{C}$ has at most $m+n$ states. □

Next we are going to prove a tight bound for the intersection.

Theorem 3 *For any integers $m, n \geq 1$ let $\mathcal{A}$ be an m-state and $\mathcal{B}$ be an n-state* NFA*. Then $m \cdot n$ states are sufficient and necessary in the worst case for an* NFA *to accept the language $L(\mathcal{A}) \cap L(\mathcal{B})$.*

Proof. Clearly, the NFA defined by the cross-product of $\mathcal{A}$ and $\mathcal{B}$ accepts the language $L(\mathcal{A}) \cap L(\mathcal{B})$ with $m \cdot n$ states.

As witness languages for the fact that the bound is reached in the worst case define the k-state language $L_k = \{w \in \{a,b\}^* \mid \#_a(w) \equiv 0 \pmod{k}\}$ for all $k \in \mathbb{N}$.

Identically, L'_k is defined to be $\{w \in \{a,b\}^* \mid \#_b(w) \equiv 0 \pmod{k}\}$. It remains to show that an NFA $\mathcal{C}$ that accepts $L_m \cap L'_n$ for $m, n \geq 1$, needs at least $m \cdot n$ states.

Consider the input words $a^i b^j$ and $a^{i'} b^{j'}$ with $0 \leq i, i' \leq m-1$ and $0 \leq j, j' \leq n-1$, and assume $\mathcal{C} = \langle S, A, \delta, s_0, F \rangle$ has less than $m \cdot n$ states. Since there are $m \cdot n$ such words, for at least two of them the intersection

$$\{s \in S \mid s \in \Delta(s_0, a^i b^j) \wedge \Delta(s, a^{m-i} b^{n-j}) \cap F \neq \emptyset\} \cap \Delta(s_0, a^{i'} b^{j'})$$

is not empty. This implies $a^{i'} b^{j'} a^{m-i} b^{n-j} \in L_m \cap L'_n$. Since either $i \neq i'$ or $j \neq j'$ it follows $i' + m - i \not\equiv 0 \pmod{m}$ or $j' + n - j \not\equiv 0 \pmod{n}$, a contradiction. □

In the unary case the lower bound requires $m \perp n$. In [18] unary languages are studied whose *deterministic* state complexities are not relatively prime.

Theorem 4 *For any integers $m, n \geq 1$ let $\mathcal{A}$ be a unary m-state and $\mathcal{B}$ be a unary n-state* NFA*. Then $m \cdot n$ states are sufficient for an* NFA *to accept the language $L(\mathcal{A}) \cap L(\mathcal{B})$. If $m \perp n$, then there exist a unary m-state* NFA *$\mathcal{A}$ and a unary n-state* NFA *$\mathcal{B}$ (with the same input symbol) such that any* NFA *accepting $L(\mathcal{A}) \cap L(\mathcal{B})$ needs at least $m \cdot n$ states.*

Proof. As witness languages consider $L(\mathcal{A}) = \{a^m\}^*$ and $L(\mathcal{B}) = \{a^n\}^*$. Since $m \perp n$ the intersection $L(\mathcal{A}) \cap L(\mathcal{B})$ is $\{a^{m \cdot n}\}^*$. Due to Lemma 1 any NFA accepting the intersection needs at least $m \cdot n$ states. □

The complementation of NFA languages is an expensive task at any rate. It is well known [16] that 2^n is the tight upper bound on the number of states necessary for a deterministic finite automaton to accept an (infinite) n-state NFA language. Since the complementation operation on deterministic finite automata neither increases nor decreases the number of states (simply exchange accepting and rejecting states) we obtain an upper bound for the state complexity of the complementation on NFAs.

Corollary 1 *For any integer $n \geq 1$ the complement of an n-state* NFA *language is accepted by a 2^n-state* NFA.

Unfortunately, this expensive upper bound is tight. Birget [2, 4] showed for an input alphabet of size four that, for any integer $n \geq 1$, there exists an n-state NFA $\mathcal{A}$ such that any NFA that accepts the complement of $L(\mathcal{A})$ needs at least 2^n states. The question whether we can achieve a tight bound over a smaller alphabet is currently open. But using a two-letter alphabet we can prove a tight bound in the order of magnitude.

Theorem 5 *For any integer $n > 2$ there exists an n-state* NFA *$\mathcal{A}$ such that any* NFA *that accepts the complement of $L(\mathcal{A})$ needs at least 2^{n-2} states.*

Proof. For $k \geq 0$ let $L_k = \{a,b\}^* a\{a,b\}^k b\{a,b\}^*$. It is clear that L_k is accepted by a $(k+3)$-state NFA. Intuitively, $\mathcal{A}$ has to guess the position of an input symbol a which is followed by k arbitrary input symbols and a symbol b.

In order to accept the complement of L_k an NFA $\mathcal{B} = \langle S', \{a,b\}, \delta', s'_0, F' \rangle$ has to verify that the input has no substring $a\{a,b\}^k b$. Therefore, after reading a symbol a the NFA $\mathcal{B}$ must be able to remember the next k input symbols. Altogether this needs 2^{k+1} states (cf. Figure 1).

More formally, we consider the input words of length $k+1$. Observe that for each of these words w the concatenation ww belongs to the complement of L_k. Let $S(w)$ be $\{s \in S' \mid s \in \Delta'(s'_0, w) \wedge \Delta'(s,w) \cap F' \neq \emptyset\}$, and v, v' be two arbitrary different words from $\{a,b\}^{k+1}$. Assume $S(v) \cap S(v') \neq \emptyset$. It follows $\Delta'(s'_0, vv') \cap F' \neq \emptyset$ and $\Delta'(s'_0, v'v) \cap F' \neq \emptyset$ and, therefore, vv' and $v'v$ are accepted by $\mathcal{B}$.

But this is a contradiction since there exists a position $1 \leq p \leq k+1$ at which v has a symbol a and v' a symbol b or vice versa. Thus either vv' or $v'v$ is of the form $x_1 \cdots x_{p-1} a x_{p+1} \cdots x_{k+1} y_1 \cdots y_{p-1} b y_{p+1} \cdots y_{k+1}$ and, therefore, belongs to L_k. From the contradiction follows $S(v) \cap S(v') = \emptyset$. Since there exist 2^{k+1} words in $\{a,b\}^{k+1}$ the state set S' has to contain at least 2^{k+1} states. □

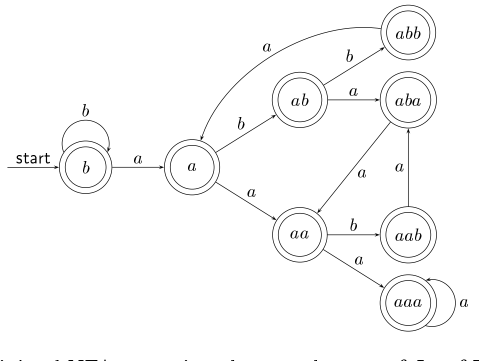

Fig. 1. A minimal NFA accepting the complement of L_2 of Theorem 5.

For complementation of unary NFAs a crucial role is played by the function $F(n) = \max\{\operatorname{lcm}(x_1, \ldots, x_k) \mid x_1, \ldots, x_k \in \mathbb{N} \wedge x_1 + \cdots + x_k = n\}$ which gives the maximal order of the cyclic subgroups of the symmetric group of n symbols. For example, the first seven values of F are $F(1) = 1$, $F(2) = 2$, $F(3) = 3$, $F(4) = 4$, $F(5) = 6$, $F(6) = 6$, $F(7) = 12$ due to the sums $1 = 1$, $2 = 2$, $3 = 3$, $4 = 4$, $5 = 2 + 3$, $6 = 1 + 2 + 3$ (or $6 = 6$) and $7 = 3 + 4$.

Since F depends on the irregular distribution of the prime numbers we cannot expect to express $F(n)$ explicitly by n. The function itself has been investigated by Landau [11, 12] who has proved the asymptotic growth rate $\lim_{n\to\infty} \frac{\ln(F(n))}{\sqrt{n\cdot\ln(n)}} = 1$. A bound immediately derived from Landau's result is: $\ln(F(n)) \in \Theta(\sqrt{n \cdot \ln(n)})$. For our purposes the implied rough estimation $F(n) \in e^{\Theta(\sqrt{n\cdot\ln(n)})}$ suffices. Shallit and Ellul [21] pointed out that the bound $F(n) \in O(e^{\sqrt{n\cdot\ln(n)}})$ which is claimed in [7] is not correct. They deduced finer bounds from a result in [24] where the currently best known approximation for F has been proved. Nevertheless, in [7] it has been shown that for any unary n-state NFA there exists an equivalent $O(F(n))$-state deterministic finite automaton.

Corollary 2 *For any integer $n \geq 1$ the complement of a unary n-state* NFA *language is accepted by an $O(F(n))$-state* NFA*.*

As for the regular case the expensive upper bound is tight in the order of magnitude.

Theorem 6 *For any integer $n > 1$ there exists a unary n-state* NFA *$\mathcal{A}$ such that any* NFA *accepting the complement of $L(\mathcal{A})$ needs at least $\Omega(F(n))$ states.*

Proof. Let $x_1, \ldots, x_k \in \mathbb{N}$ be integers such that $x_1 + \cdots + x_k = n - 1$ and $\mathrm{lcm}(x_1, \ldots, x_k) = F(n-1)$. Now define for $1 \leq i \leq k$ the languages $L_i = \{a^{x_i}\}^*$ and consider the union of their complements: $L = \overline{L_1} \cup \overline{L_2} \cup \cdots \cup \overline{L_k}$.

Since $\overline{L_i}$ is acceptable by a x_i-state NFA $\mathcal{A}_i$, the language L is acceptable by an NFA $\mathcal{A}$ with at most $1 + x_1 + \cdots + x_k = n$ states. To this end we introduce a new initial state and connect it nondeterministically to the states of the $\mathcal{A}_i$ that are reached after their first state transition. The complement of L is $\overline{L} = \{a^{\mathrm{lcm}(x_1,\ldots,x_k)}\}^*$. Therefore, the complement of the n-state language L needs $\mathrm{lcm}(x_1, \ldots, x_k) = F(n-1)$ states. Since $F(n)$ is of order $e^{\Theta(\sqrt{n\cdot\ln(n)})}$ it follows $F(n-1)$ is of order $\Omega(F(n))$. □

3.2. Catenation Operations

Now we turn to the catenation operations. In particular, tight bounds for concatenation, iteration and λ-free iteration will be shown. Roughly speaking, in terms of state complexity these are efficient operations for NFAs. Again, this is essentially different when deterministic finite automata come to play. For example, for arbitrary alphabets in [28] a bound of $(2m-1) \cdot 2^{n-1}$ states has been shown for the DFA-concatenation, and in [25] a bound of $2^{n-1} + 2^{n-2}$ states for the iteration.

Theorem 7 *For any integers $m, n \geq 1$ let $\mathcal{A}$ be an m-state* NFA *and $\mathcal{B}$ be an n-state* NFA*. Then $m + n$ states are sufficient and necessary in the worst case for an* NFA *to accept the language $L(\mathcal{A})L(\mathcal{B})$.*

Proof. The upper bound is due to the observation that in $\mathcal{C}$ one has simply to connect the accepting states in $\mathcal{A}$ with the states in $\mathcal{B}$ that follow the initial state.

The upper bound is reached for the concatenation of the languages $L(\mathcal{A}) = \{a^m\}^*$ and $L(\mathcal{B}) = \{b^n\}^*$. The remaining proof follows the idea of the proof of Theorem 1. □

In the unary case the lower bound of the concatenation misses the upper bound by one state. It is currently an open question how to close the gap by more sophisticated constructions or witness languages.

Theorem 8 *For any integers $m, n > 1$ let $\mathcal{A}$ be a unary m-state* NFA *and $\mathcal{B}$ be a unary n-state* NFA*. Then $m + n$ states are sufficient for an* NFA *to accept the language $L(\mathcal{A})L(\mathcal{B})$. Moreover, there exist a unary m-state* NFA *$\mathcal{A}$ and a unary n-state* NFA *$\mathcal{B}$ such that any* NFA *$\mathcal{C}$ accepting $L(\mathcal{A})L(\mathcal{B})$ needs at least $m + n - 1$ states.*

Proof. The upper bound has been shown in the proof of Theorem 7.

Let $L(\mathcal{A})$ be the m-state language $\{a^k \mid k \equiv m - 1 \pmod m\}$ and $L(\mathcal{B})$ be the n-state language $\{a^k \mid k \equiv n - 1 \pmod n\}$.

The shortest word in $L(\mathcal{A})$ respectively $L(\mathcal{B})$ is a^{m-1} respectively a^{n-1}. Therefore the shortest word in $L(\mathcal{C})$ is a^{m+n-2}. Assume contrarily to the assertion $\mathcal{C}$ has at most $m + n - 2$ states. Let $\mathcal{C}$ accept the input a^{m+n-2} by running through the state sequence $s_0 \vdash s_1 \vdash \cdots \vdash s_{m+n-2}$ where all states except s_{m+n-2} are non-accepting. Due to the assumption at least one of the non-accepting states s_i must appear at least twice in the sequence. This implies that there exists an accepting computation that does not run through the cycle $s_i \vdash \cdots \vdash s_i$. So an input whose length is at most $m + n - 3$ would be accepted, a contradiction. □

The constructions yielding the upper bounds for the iteration and λ-free iteration are similar. The trivial difference between both operations concerns the empty word only. Moreover, the difference does not appear for languages containing the empty word. Nevertheless, in the worst case the difference costs one state.

Theorem 9 *For any integer $n > 2$ let $\mathcal{A}$ be a unary or non-unary n-state* NFA*. Then $n + 1$ resp. n states are sufficient and necessary in the worst case for an* NFA *to accept the language $L(\mathcal{A})^*$ resp. $L(\mathcal{A})^+$.*

Proof. Let $\mathcal{A} = \langle S_A, A_A, \delta_A, s_{0,A}, F_A\rangle$ be an n-state NFA. Then the transition function of an n-state NFA $\mathcal{C} = \langle S, A, \delta, s_0, F\rangle$ that accepts the language $L(\mathcal{A})^+$ is for $s \in S$ and $a \in A$ defined as follows: $\delta(s, a) = \delta_A(s, a)$ if $s \notin F_A$, and $\delta(s, a) = \delta_A(s, a) \cup \delta_A(s_{0,A}, a)$ if $s \in F_A$. The other components remain unchanged, i.e., $S = S_A$, $s_0 = s_{0,A}$, and $F = F_A$.

If the empty word belongs to $L(\mathcal{A})$ then the construction works fine for $L(\mathcal{A})^*$ also. Otherwise an additional state has to be added: Let $s_0' \notin S_A$ and define $S = S_A \cup \{s_0'\}$, $s_0 = s_0'$, $F = F_A \cup \{s_0'\}$, and for $s \in S$ and $a \in A$: $\delta(s, a) = \delta_A(s, a)$ if $s \notin F_A \cup \{s_0'\}$, $\delta(s, a) = \delta_A(s, a) \cup \delta_A(s_{0,A}, a)$ if $s \in F_A$, and $\delta(s, a) = \delta_A(s_{0,A}, a)$ if $s = s_0'$.

In order to prove the tightness of the bounds for any $n > 2$ consider the n-state language $L = \{a^k \mid k \equiv n - 1 \pmod n\}$. At first we show that $n + 1$ states are necessary for $\mathcal{C} = \langle S, \{a\}, \delta, s_0, F\rangle$ to accept $L(\mathcal{A})^*$.

Contrarily, assume $\mathcal{C}$ has at most n states. We consider words of the form a^i with $0 \leq i$. The shortest four words belonging to $L(\mathcal{A})^*$ are λ, a^{n-1}, a^{2n-2}, and a^{2n-1}. It follows $s_0 \in F$. Moreover, for a^{n-1} there must exist a path $s_0 \vdash s_1 \vdash \cdots \vdash s_{n-2} \vdash s_n$ where $s_n \in F$ and $s_1, \ldots, s_{n-2}$ are different non-accepting states. Thus, $\mathcal{C}$ has at least $n-2$ non-accepting states.

Assume for a moment F to be a singleton. Then $s_0 = s_n$ and for $1 \leq i \leq n-3$ the state s_0 must not belong to $\delta(s_i, a)$. Processing the input a^{2n-1} the NFA cannot enter s_0 after $2n-2$ time steps. Since $a \notin L(\mathcal{A})^*$ the state s_0 must not belong to $\delta(s_0, a)$.

On the other hand, $\mathcal{C}$ cannot enter one of the states $s_1, \ldots, s_{n-3}$ since there is no transition to s_0. We conclude that $\mathcal{C}$ is either in state s_{n-2} or in an additional non-accepting state s_{n-1}. Since there is no transition such that $s_{n-2} \in \delta(s_{n-2}, a)$ in both cases there exists a path of length n from s_0 to s_0. But a^n does not belong to $L(\mathcal{A})^*$ and we have a contradiction to the assumption $|F| = 1$.

Due to our assumption $|S| \leq n$ we now have $|F| = 2$ and $|S| - |F| = n-2$. Let us recall the accepting sequence of states for the input a^{n-1}: $s_0 \vdash s_1 \vdash \cdots \vdash s_{n-2} \vdash s_n$. Both s_0 and s_n must be accepting states. Assume $s_n \neq s_0$. Since a^{2n-2} belongs to $L(\mathcal{A})^*$ there must be a possible transition $s_0 \vdash s_1$ or $s_n \vdash s_1$. Thus, a^{2n-2} is accepted by s_n. In order to accept a^{2n-1} there must be a corresponding transition from s_n to s_n or from s_n to s_0. In both cases the input a^n would be accepted. Therefore $s_n = s_0$.

By the same argumentation the necessity of a transition for the input symbol a from s_0 to s_0 or from s_0 to s_n follows. This implies that a is accepted. From the contradiction follows $|S| > n$.

As an immediate consequence we obtain the tightness of the bound for $L(\mathcal{A})^+$. In this case $s_0 \in F$ is not required. Thus, just one accepting state is necessary.

In order to prove the result for non-unary NFAs one has simply to exchange the witness language L by $\{w \in \{a, b\}^* \mid \#_a(w) \equiv n-1 \pmod{n}\}$. □

3.3. Reversal

The next operation under consideration is the reversal. The bounds for unary NFAs are trivial. For general deterministic automata one may expect that the state complexity is linear. But it is not. A tight bound of 2^n states for the reversal has been shown in [13]. From the following efficient bound for NFAs it follows once more that nondeterminism is a powerful concept.

Theorem 10 *For any integer $n > 3$ let $\mathcal{A}$ be an n-state* NFA. *Then $n+1$ states are sufficient and necessary in the worst case for an* NFA *to accept the language $L(\mathcal{A})^R$.*

Proof. Basically, the idea is to reverse the directions of the transitions. This works fine for NFAs whose set of accepting states is a singleton. In general we are concerned with more than one accepting state and have to add a new initial state. So we obtain an $(n+1)$-state NFA.

The language $L_k = a^k\{a^{k+1}\}^*(\{b\}^* \cup \{c\}^*)$ for $k \geq 1$, may serve as an example for the fact that the bound is reached. The $(k+3)$-state NFA $\mathcal{A}$ that accepts L_k and the $(k+4)$-state NFA $\mathcal{C}$ that accepts L_k^R are depicted in Figure 2.

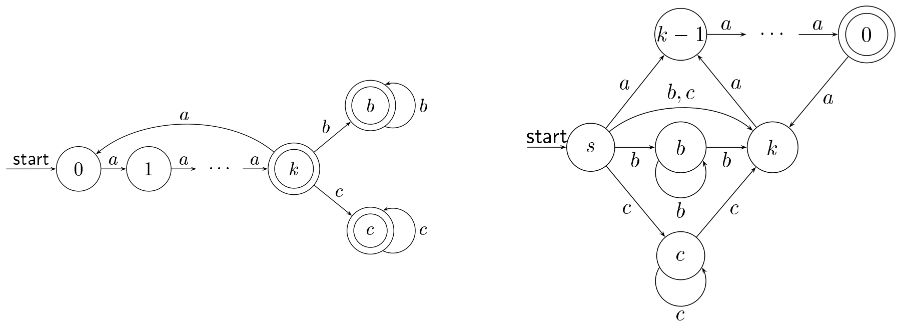

Fig. 2. A $(k+3)$-state and a $(k+4)$-state NFA accepting L_k and L_k^R of Theorem 10.

The necessity of $k+4$ states can be seen as follows. Since accepted inputs may begin with an arbitrary number of b's or c's we need two states s_b and s_c to process them. This cannot be done by the initial state because the loops would lead to acceptance of words with prefixes of the form b^*c^* or c^*b^*.

Obviously, a loop of $k+1$ states is needed in order to verify the suffix $\{a^{k+1}\}^*a^k$. If one in this sequence would be equal to s_b (s_c), then it would have a loop for b's (c's) and, hence, inputs of the form $c^*a^*b^*a^k$ $(b^*a^*c^*a^k)$ would be accepted. For similar reasons the new initial state cannot be within a loop. Altogether it follows that $\mathcal{C}$ needs at least $k+4$ states what proves the tightness of the bound. $\square$

Table 1. NFA and DFA state complexities for infinite languages.

Infinite Languages				
	NFA		DFA	
	general	unary	general	unary
$\cup$	$m+n+1$	$m+n+1$	mn	mn
$\sim$	$O(2^n)$	$e^{\Theta(\sqrt{n\cdot\ln(n)})}$	n	n
$\cap$	mn	mn	mn	mn
R	$n+1$	n	2^n	n
$\cdot$	$m+n$	$O(m+n)$	$(2m-1)2^{n-1}$	mn
$*$	$n+1$	$n+1$	$2^{n-1}+2^{n-2}$	$(n-1)^2+1$
$+$	n	n		

4. Operations on Finite Languages

4.1. Boolean Operations and Catenation

The situation for finite languages is easier since essentially the structure of the corresponding NFAs is simpler. When we are concerned with finite languages over arbitrary alphabets we may assume without loss of generality that minimal NFAs not accepting the empty word have only one accepting state. Since they do not contain any cycles they do contain at least one accepting (sink) state for which the transition function is not defined. Now a given minimal NFA with more than one accepting state is modified such that a sink state becomes the only accepting state. To this end simply the transition function has to be extended. If the finite language contains the empty word, then in addition the initial state is a second accepting one.

For upper bounds in the finite deterministic case see [6]. Here the state complexity of the union can be reduced by three states compared with the general case.

Theorem 11 *For any integers $m, n \geq 2$ let $\mathcal{A}$ be an m-state* NFA *and $\mathcal{B}$ be an n-state* NFA*. If $L(\mathcal{A})$ and $L(\mathcal{B})$ are finite, then $m+n-2$ states are sufficient and necessary in the worst case for an* NFA *$\mathcal{C}$ to accept the language $L(\mathcal{A}) \cup L(\mathcal{B})$.*

Proof. We can adapt the proof of Theorem 1 as follows. Since NFAs for finite languages do not contain any cycles, for the construction of the NFA $\mathcal{C}$ we do not need a new initial state (this saves one state). Moreover, we can merge both initial states (this saves the second one) and both accepting sink states (this saves the third one). Now the construction of $\mathcal{C}$ is straightforward.

The finite languages $\{a^m\}$ and $\{b^n\}$ are witnesses for the necessity of the number of states for the union in the worst case. An NFA that accepts the language $\{a^m\}$ needs at least $m+1$ states. By the same argumentation as in the proof of Theorem 1 and merged initial and sink states we obtain at least $(m+1)+(n+1)-2$ states for an NFA that accepts $\{a^m\} \cup \{b^n\}$. □

In case of finite language concatenations one state can be saved.

Theorem 12 *For any integers $m, n \geq 1$ let $\mathcal{A}$ be an m-state* NFA *and $\mathcal{B}$ be an n-state* NFA*. If $L(\mathcal{A})$ and $L(\mathcal{B})$ are finite, then $m+n-1$ states are sufficient and necessary in the worst case for an* NFA *to accept the language $L(\mathcal{A})L(\mathcal{B})$.*

Proof. Since for finite languages $\mathcal{A}$ and $\mathcal{B}$ must not contain any cycles the initial state of $\mathcal{B}$ is not reachable, after the construction of Theorem 7 it can be deleted what yields an upper bound of $m+n-1$ states.

As witnesses for the tightness consider the languages $\{a^{m-1}\}$ and $\{b^{n-1}\}$. They are accepted by m-state resp. n-state NFAs. Clearly, any NFA for the concatenation needs at least $m+n-1$ states. □

Now we turn to operations on finite unary languages. In [20] it is stated that every finite unary n-state NFA language is acceptable by some complete deterministic finite automaton with at most $n+1$ states. A little bit more sophisticated, one observes that the minimum NFA for a finite unary language L has $n+1$ states if the longest word in L is of length n. Otherwise the NFA would run through a cycle when accepting a^n and, thus, L would be infinite. Now we can always construct a minimum NFA $\mathcal{A} = \langle S, \{a\}, \delta, s_0, F\rangle$ for L as follows: $S = \{s_0, s_1, \ldots, s_n\}$, $\delta(s_i, a) = \{s_{i+1}\}$ for $0 \leq i \leq n-1$, and $F = \{s_i \mid a^i \in L\}$.

From the construction it follows conversely that a minimum $(n+1)$-state NFA for a non-empty finite unary language accepts the input a^n. An immediate consequence is that we have only to consider the longest words in the languages in order to obtain the state complexity of operations that preserve the finiteness.

Theorem 13 *For any integers $m, n \geq 1$ let $\mathcal{A}$ be a unary m-state* NFA *and $\mathcal{B}$ be a unary n-state* NFA*. If $L(\mathcal{A})$ and $L(\mathcal{B})$ are finite, then* $\max(m, n)$, $\min(m, n)$ *respectively $m + n - 1$ states are sufficient and necessary in the worst case for an* NFA *to accept the language $L(\mathcal{A}) \cup L(\mathcal{B})$, $L(\mathcal{A}) \cap L(\mathcal{B})$ respectively $L(\mathcal{A})L(\mathcal{B})$.*

The situation for the complementation of finite languages over an ℓ-letter alphabet, $\ell \geq 2$, is quite different from the general case, since the upper bound of the transformation to a deterministic finite automaton is different. In [20] it has been shown that $O(\ell^{\frac{n}{\log_2 \ell + 1}})$ states are an upper bound for deterministic finite automata accepting a finite n-state NFA language.

Corollary 3 *For any integers $\ell, n > 1$ the complement of a finite n-state* NFA *language over an ℓ-letter alphabet is accepted by an $O(\ell^{\frac{n}{\log_2 \ell + 1}})$-state* NFA*.*

Note, that for $\ell = 2$ the upper bound is $O(2^{\frac{n}{2}})$. A slight modification of the proof of Theorem 5 yields:

Theorem 14 *For any integers $\ell > 1$ and $n > 2$ there exists a finite n-state* NFA *language L over an ℓ-letter alphabet such that any* NFA *that accepts the complement of L needs at least $\Omega(\ell^{\frac{n}{2 \cdot \log_2 \ell}})$ states.*

Proof. For $\ell > 1$ let $A = \{a_1, \ldots, a_\ell\}$ be an alphabet. Let $k \geq 0$ be an integer. A finite language L_k is defined by $A^j a_1 A^k y$, where $0 \leq j \leq k$ and $y \in A \setminus \{a_1\}$. The NFA depicted in Figure 3 accepts L_k with $2k + 3$ states.

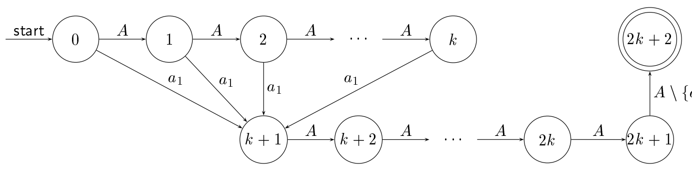

Fig. 3. A $(2k + 3)$-state NFA accepting L_k of Theorem 14.

An NFA $\mathcal{B}$ for the complement works similar to the corresponding NFA in the proof of Theorem 5. It need not remember $k+1$ input symbols exactly, but whether a symbol has been a_1 or not. Since previously we argued with words of finite lengths it follows immediately that $\mathcal{B}$ needs at least 2^{k+1} states. Additionally the length of the prefix A^j has to be tracked. For this purpose the state set has to be doubled such that we have a lower bound of 2^{k+2} states. Transforming $2 = \ell^{\log_\ell 2} = \ell^{\frac{1}{\log_2 \ell}}$, for $n = 2k + 4 > 2k + 3$ we obtain the lower bound $\ell^{\frac{k+2}{\log_2 \ell}} \in \Omega(\ell^{\frac{n}{2 \cdot \log_2 \ell}})$. □

The complementation applied to finite languages yield infinite languages. So in general for the lower bounds we cannot argue with the simple chain structure as before.

Theorem 15 *For any integer $n \geq 1$ let $\mathcal{A}$ be an n-state* NFA*. If $L(\mathcal{A})$ is finite, then $n + 1$ states are sufficient and necessary in the worst case for an* NFA *$\mathcal{C}$ to accept the complement of $L(\mathcal{A})$.*

Proof. Without loss of generality we may assume that $\mathcal{A}$ has the simple chain structure with states from s_0 to s_{n-1} as mentioned before. By interchanging accepting and non-accepting states we obtain an NFA that processes all inputs up to a length $n-1$ as required. But all longer words a^k, $k \geq n$, are belonging to the complement of $L(\mathcal{A})$. So it suffices to add a new accepting state s_n and two transitions from s_{n-1} to s_n and from s_n to s_n, in order to complete the construction of $\mathcal{C}$.

The tightness of the bound can be seen for the n-state NFA language $L = \{a^k \mid 0 \leq k \leq n-1\}$. Since a^n is the shortest word belonging to the complement of L it follows that $\mathcal{C}$ has at least $n+1$ states. □

4.2. Kleene Operations and Reversal

The state complexity for the iterations in the finite language case is as for infinite languages if the iteration is λ-free. If not the costs are reduced by two states. The following result is for both unary and arbitrary languages.

Theorem 16 *For any integer $n > 1$ let $\mathcal{A}$ be a unary or non-unary n-state* NFA. *If $L(\mathcal{A})$ is finite, then $n-1$ resp. n states are sufficient and necessary in the worst case for an* NFA *to accept the language $L(\mathcal{A})^*$ resp. $L(\mathcal{A})^+$.*

Proof. For the upper bounds we can adapt the construction of Theorem 9. The accepting states are connected to the states following the initial state. That is all for λ-free iterations.

For iterations we have to provide acceptance of the empty word. The following two observations let us save two states compared with infinite languages. First, the initial state is never reached again after initial time. Second, since the underlying language is finite and the accepting automaton is reduced there must exist an accepting state s_f for which the state transition is not defined. We can take s_f as new initial state and delete the old initial state what altogether leads to an $(n-1)$-state NFA for the iteration.

The bound for the λ-free iteration is reached for the language $L = \{a^{n-1}\}$ which requires n states. For the acceptance of $L^+ = \{a^{n-1}\}^+$ at least n states are necessary.

The bound for the iteration is reached for the language $L = \{a^n\}$ that requires $n+1$ states. Clearly, in order to accept $\{a^n\}^*$ at least n states are necessary. □

A proof of a tight bound for the reversal of finite languages can be found in [5]. It is of order $O(2^{\frac{n}{2}})$ for a two-letter alphabet. From the following efficient bounds for NFAs it follows once more that nondeterminism is a powerful concept. Moreover, the fact that NFAs for finite languages do not have any cycle leads again to the possibility of saving one state compared with the infinite case.

Theorem 17 *For any integer $n \geq 1$ let $\mathcal{A}$ be an n-state* NFA. *If $L(\mathcal{A})$ is finite, then n states are sufficient and necessary in the worst case for an* NFA *to accept the language $L(\mathcal{A})^R$.*

Proof. Assume without loss of generality that $\mathcal{A}$ has only one accepting sink state. By the construction of the proof of Theorem 10 we obtain an $(n+1)$-state NFA that has an unreachable state. It is the unique former accepting sink state. The bound follows if the state is deleted.

Let for $n \geq 1$ the language L_n defined to be $\{a,b\}^{n-1}$. Trivially, L_n is accepted by an n-state NFA. Since $L_n = L_n^R$ the assertion follows. □

The bound for the reversal of finite NFA languages is in some sense strong. It is sufficient and reached for all finite languages.

Table 2. NFA and DFA state complexities for finite languages (ℓ is the size of the input alphabet, t is the number of accepting states of the 'left' automaton).

	Finite Languages			
	NFA		DFA	
	general	unary	general	unary
$\cup$	$m+n-2$	$\max\{m,n\}$	$O(mn)$	$\max\{m,n\}$
$\sim$	$O(\ell^{\frac{n}{\log_2 \ell+1}})$	$n+1$	n	n
$\cap$	$O(mn)$	$\min\{m,n\}$	$O(mn)$	$\min\{m,n\}$
R	n	n	$O(2^{\frac{n}{2}})$	n
$\cdot$	$m+n-1$	$m+n-1$	$O(mn^{t-1}+n^t)$	$m+n-2$
$*$	$n-1$	$n-1$	$2^{n-3}+2^{n-4}$	$n^2-7n+13$
$+$	n	n		

Acknowledgements

We thank Jean-Camille Birget, Keith Ellul, and Jeffrey O. Shallit for fruitful discussions on the subject.

References

1. P. Berman, and A. Lingas, "On the complexity of regular languages in terms of finite automata," Technical Report 304, Polish Academy of Sciences, 1977.
2. J.-C. Birget, "Partial orders on words, minimal elements of regular languages and state complexity," *Theoret. Comput. Sci.* **119** (1993) 267–291.
3. J.-C. Birget, "State-complexity of finite-state devices, state compressibility and incompressibility," *Math. Systems Theory* **26** (1993) 237–269.
4. J.-C. Birget, Personal communication, October 2002.
5. C. Câmpeanu, K. Čulik, K. Salomaa and S. Yu, "State complexity of basic operations on finite languages," *Proc. 4th International Workshop on Implementing Automata (WIA '99)*, LNCS 2214, 2001, pp. 60–70.
6. C. Câmpeanu, N. Sântean and S. Yu, "Minimal cover-automata for finite languages," *Theoret. Comput. Sci.* **267** (2001) 3–16.
7. M. Chrobak, "Finite automata and unary languages," *Theoret. Comput. Sci.* **47** (1986) 149–158.
8. J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Language, and Computation* (Addison-Wesley, Reading, Massachusetts, 1979).
9. H. B. Hunt, D. J. Rosenkrantz and T. G. Szymanski, "On the equivalence, containment, and covering problems for the regular and context-free languages," *J. Comput. System Sci.* **12** (1976) 222–268.
10. T. Jiang and B. Ravikumar, "Minimal NFA problems are hard," *SIAM J. Comput.* **22** (1993) 1117–1141.
11. E. Landau, "Über die Maximalordnung der Permutationen gegebenen Grades," *Archiv der Math. und Phys.* **3** (1903) 92–103.

12. E. Landau, *Handbuch der Lehre von der Verteilung der Primzahlen.* (Teubner, Leipzig, 1909).

13. E. Leiss, "Succinct representation of regular languages by Boolean automata," *Theoret. Comput. Sci.* **13** (1981) 323–330.

14. C. Mereghetti and G. Pighizzini, "Two-way automata simulations and unary languages," *J. Aut. Lang. and Comb.* **5** (2000) 287–300.

15. C. Mereghetti and G. Pighizzini, "Optimal simulations between unary automata," *SIAM J. Comput.* **30** (2001) 1976–1992.

16. A. R. Meyer and M. J. Fischer, "Economy of description by automata, grammars, and formal systems," *Proc. 12th Annual IEEE Symposium on Switching and Automata Theory*, 1971, pp. 188–191.

17. C. Nicaud, "Average state complexity of operations on unary automata," *Proc. Mathematical Foundations of Computer Science (MFCS '99)*, LNCS 1672, 1999, pp. 231–240.

18. G. Pighizzini and J. O. Shallit, "Unary language operations, state complexity and Jacobsthal's function," *Int. J. Found. Comput. Sci.* **13** (2002) 145–159.

19. W. J. Sakoda and M. Sipser, "Nondeterminism and the size of two way finite automata," *Proc. 10th Annual ACM Symposium on Theory of Computing (STOC '78)*, 1978, pp. 275–286.

20. K. Salomaa and S. Yu, "NFA to DFA transformation for finite languages over arbitrary alphabets," *J. Aut., Lang. and Comb.* **2** (1997) 177–186.

21. J. O. Shallit and K. Ellul, Personal communication, October 2002.

22. S. Sipser, "Lower bounds on the size of sweeping automata," *J. Comput. System Sci.* **21** (1980) 195–202.

23. L. J. Stockmeyer and A. R. Meyer, "Word problems requiring exponential time," *Proc. 5th Annual ACM Symposium on Theory of Computing (STOC '73)*, 1973, pp. 1–9.

24. M. Szalay, "On the maximal order in S_n and S_n^*," *Acta Arithm.* **37** (1980) 321–331.

25. S. Yu, "Regular languages," in *Handbook of Formal Languages 1*, eds. G. Rozenberg and A. Salomaa (Springer, Berlin, 1997) chapter 2, pp. 41–110.

26. S. Yu, "State complexity of regular languages," *J. Aut. Lang. and Comb.* **6** (2001) 221–234.

27. S. Yu, "State complexity of finite and infinite regular languages," *Bull. EATCS* **76** (2002) 142–152.

28. S. Yu, Q. Zhuang and K. Salomaa, "The state complexities of some basic operations on regular languages," *Theoret. Comput. Sci.* **125** (1994) 315–328.