I F I G
Research
Report

Institut für Informatik

State Complexity of Basic Operations on Nondeterministic Finite Automata

Markus Holzer Martin Kutrib

IFIG Research Report 0103
April 2001

Institut für Informatik
JLU Gießen
Arndtstraße 2
D-35392 Giessen, Germany
Tel: +49-641-99-32141
Fax: +49-641-99-32149
mail@informatik.uni-giessen.de
www.informatik.uni-giessen.de

Justus-Liebig-
Universität
Giessen

IFIG Research Report

IFIG Research Report 0103, April 2001

State Complexity of Basic Operations on Nondeterministic Finite Automata

Markus Holzer[1]

Institut für Informatik, Technische Universität München

Arcisstraße 21, D-80290 München, Germany

Martin Kutrib[2]

Institut für Informatik, Universität Giessen

Arndtstr. 2, D-35392 Giessen, Germany

Abstract. The state complexities of basic operations on nondeterministic finite automata (NFA) are investigated. In particular, we consider Boolean operations, catenation operations – concatenation, iteration, λ-free iteration – and the reversal on NFAs that accept finite and infinite languages over arbitrary alphabets. 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. For the intersection of finite languages and the complementation tight bounds in the order of magnitude are proved.

It turns out that the state complexities of operations on NFAs and deterministic finite automata (DFA) are quite different. For example, the reversal and concatenation have exponential state complexity on DFAs but linear complexity on NFAs. Conversely, the complementation can be done with linear complexity on DFAs but needs exponentially many states on NFAs.

CR Subject Classification (1998): F.1, F.4.3

[1]E-mail: holzer@informatik.tu-muenchen.de

[2]E-mail: kutrib@informatik.uni-giessen.de

1 Introduction

Motivated by several applications and implementations of finite automata in software engineering, programming languages and other practical areas in computer science, the state complexity of deterministic finite automata has been studied in recent years. For example, the state complexity of the intersection of DFAs has been studied in [15]. A tight bound of 2^n states for the reversal has been shown in [8], whereas catenations and other operations are the main topic of [16]. For the important case of finite languages results have been obtained in [1]. A state-of-the-art survey can be found in [14]. Related to the problem of finding upper bounds for the state complexity is the problem of efficiently simulating nondeterministic automata by deterministic ones. For example, transforming a certain type of NFA to a DFA gives an upper bound for the corresponding NFA state complexity of complementation. Results concerning the simulation problems have been shown in [3, 9, 10, 12].

As pointed out in [14] there are several good reasons why the size of DFAs is a natural and objective measure for regular languages. On the other hand, the influence of the degree of nondeterminism on the power and limitations of certain devices is an important question in descriptional complexity theory. Finite automata with limited nondeterminism have been considered in [7] where an infinite nondeterministic hierarchy of regular languages has been proved. In [4] it is dealt with the quantification of inherent nondeterminism in regular languages, and in [5] with the relation between ambiguity and the amount of nondeterminism.

We expect that examining the state complexity of basic operations on NFAs will enhance the understanding of the relations between nondeterminism, ambiguity and the power of finite automata. In particular, we consider Boolean operations, catenation operations and the reversal on NFAs that accept finite and infinite languages over arbitrary alphabets. 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. For the intersection of finite languages and the complementation tight bounds in the order of magnitude are proved.
It turns out that the state complexity of operations on NFAs and deterministic finite automata are quite different. For example, the reversal and concatenation have exponential state complexity on DFAs but linear complexity on NFAs. Conversely, the complementation can be done with linear complexity on DFAs but needs exponentially many states on NFAs.

In the next section we define the basic notions and present preliminary results. Section 3 is devoted to the study of Boolean operations. Finally, in Section 4 we prove the tight bounds for catenation operations, and in Section 5 for the reversal. At the end of Section 5 a table is given that summarizes the presented results and compares them to the corresponding results for DFAs.

2 Preliminaries

We denote the the integers by $\mathbb{Z}$, the positive integers $\{1, 2, \ldots\}$ by $\mathbb{N}$, the set $\mathbb{N} \cup \{0\}$ by $\mathbb{N}_0$ and the powerset of a set S by 2^S. The empty word is denoted by λ and the reversal of a word w by w^R. For the length of w we write $|w|$. The number of occurrences of a symbol a in the word w is denoted by $\#_a(w)$. We use $\subseteq$ for inclusions and $\subset$ if the inclusion is strict. For a function $f : \mathbb{N}_0 \to \mathbb{N}$ we denote its i-fold composition by $f^{[i]}$ for $i \in \mathbb{N}$. As usual we define the set of functions that grow strictly less than f by $o(f) = \{g : \mathbb{N}_0 \to \mathbb{N} \mid \lim_{n\to\infty} \frac{g(n)}{f(n)} = 0\}$. In terms of orders of magnitude f is an upper bound of the set $O(f) = \{g : \mathbb{N}_0 \to \mathbb{N} \mid \exists\, n_0, c \in \mathbb{N} : \forall\, n \geq n_0 : g(n) \leq c \cdot f(n)\}$. Conversely, f is a lower bound of the set $\Omega(f) = \{g : \mathbb{N}_0 \to \mathbb{N} \mid f \in O(g)\}$.

Definition 1 *A* nondeterministic finite automaton (NFA) *is a system* $\mathcal{A} = \langle S, A, \delta, s_0, F\rangle$, *where*

1. S *is the finite set of* internal states,
2. A *is the finite set of* input symbols,
3. $s_0 \in S$ *is the* initial state,
4. $F \subseteq S$ *is the set of* accepting (or final) states, *and*
5. $\delta : S \times A \to 2^S$ *is the* transition function.

The set of rejecting states is implicitly given by the partitioning, i.e. $S \setminus F$.

In some sense the transition function is complete. W.l.o.g. we may require δ to be a total function, since whenever the operation of an NFA is supposed not to be defined, then δ can map to the empty set which, trivially, belongs to 2^S. Thus, in some sense the NFA need not be complete. However, if not otherwise stated throughout the paper we assume that the NFAs are always *reduced*. This means that there are no unreachable states and that from any state a final 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 [6] 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 as follows:

$$\begin{aligned}\Delta(s, \lambda) &= \{s\} \\ \Delta(s, wa) &= \bigcup_{s' \in \Delta(s,w)} \delta(s', a)\end{aligned}$$

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.

Definition 2 *Let* $\mathcal{A} = \langle S, A, \delta, s_0, F\rangle$ *be an* NFA, *then*

1. *a word* $w \in A^*$ *is* accepted *by* $\mathcal{A}$ *if* $\Delta(s_0, w) \cap F \neq \emptyset$, *and*
2. $L(\mathcal{A}) = \{w \in A^* \mid w$ *is accepted by* $\mathcal{A}\}$ *is the* language accepted *by* $\mathcal{A}$.

The next two preliminary results involve NFAs directly. They are key tools in the following sections, and can be proved by a simple pumping argument.

Lemma 3 *Let $p \geq 1$ be an arbitrary integer. Any* NFA *that accepts the language $\{a^p\}^+$ needs at least $p+1$ states.*

Proof. Assume the NFA $\mathcal{A} = \langle S, \{a\}, \delta, s_0, F\rangle$ with $|S| \leq p$ accepts $\{a^p\}^+$. Then given the input a^p an accepting computation of $\mathcal{A}$ runs through a sequence of states $s_0 \vdash s_1 \vdash \cdots \vdash s_{p-1} \vdash s_p$, where $s_i \in S$ for $0 \leq i \leq p$.

Since the input a^p belongs to the language, s_p must be an accepting state, and there are at most $p-1$ non-accepting states in S. Since the empty word must not be accepted we have a contradiction for $p = 1$. For $p > 1$ we conclude that s_0 to s_{p-1} must not be accepting. Thus, at least one of the states in the sequence $s_0, \ldots, s_{p-1}$ appears at least twice. This implies that there exists a cycle whose length $i \geq 1$ does not exceed $p-1$. Therefore, there exists an accepting computation that runs through the cycle once more. But the corresponding input a^{p+i} does not belong to the language $\{a^p\}^+$. $\square$

The $(p+1)$th state is necessary since the initial state has to be a non-accepting one. If we modify the language to $\{a^p\}^*$ then the initial state could be equal to the accepting state.

Corollary 4 *Let $p \geq 1$ be an arbitrary integer. Any* NFA *that accepts the language $\{a^p\}^*$ needs at least p states.*

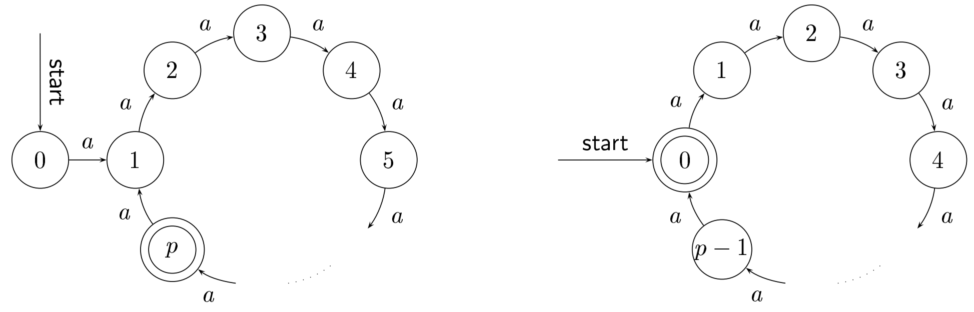

Figure 1: Minimal NFAs accepting $\{a^p\}^+$ and $\{a^p\}^*$.

3 Boolean Operations

We start our investigations with Boolean operations on NFAs that accept languages over arbitrary alphabets. 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. (m and n denote the number of states of the automata on which the operations are performed.) However, the state complexity of NFA operations is essentially different. At first we consider the union.

Theorem 5 *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 *$\mathcal{C}$ 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.

Let $\mathcal{A} = \langle S_A, A_A, \delta_A, s_{0,A}, F_A \rangle$ and $\mathcal{B} = \langle S_B, A_B, \delta_B, s_{0,B}, F_B \rangle$ with $S_A \cap S_B = \emptyset$, then $\mathcal{C} = \langle S, A, \delta, s_0, F \rangle$ is defined as follows:

$$S = S_A \cup S_B \cup \{s_0\} \text{ where } s_0 \notin S_A \cup S_B$$

$$A = A_A \cup A_B$$

$$F = \begin{cases} F_A \cup F_B \text{ if } \lambda \notin L(\mathcal{A}) \cup L(\mathcal{B}) \\ F_A \cup F_B \cup \{s_0\} \text{ otherwise} \end{cases}$$

$$\delta(s,a) = \begin{cases} \delta_A(s_{0,A}, a) \text{ if } a \in A_A \text{ and } a \notin A_B \text{ and } s = s_0 \\ \delta_B(s_{0,B}, a) \text{ if } a \in A_B \text{ and } a \notin A_A \text{ and } s = s_0 \\ \delta_A(s_{0,A}, a) \cup \delta_B(s_{0,B}, a) \text{ if } a \in A_A \cap A_B \text{ and } s = s_0 \\ \delta_A(s,a) \text{ if } a \in A_A \text{ and } s \in S_A \\ \delta_B(s,a) \text{ if } a \in A_B \text{ and } s \in S_B \end{cases}$$

for $s \in S$ and $a \in A$.

During the first transition $\mathcal{C}$ nondeterministically guesses whether the input may belong to $L(\mathcal{A})$ or $L(\mathcal{B})$. Subsequently, $\mathcal{A}$ or $\mathcal{B}$ is simulated. Obviously, $L(\mathcal{C}) = L(\mathcal{A}) \cup L(\mathcal{B})$ and $|S_C| = |S_A| + |S_B| + 1 = m + n + 1$.

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 $\{a^m\}^*$ (cf. Figure 1) and $\mathcal{B}$ an n-state NFA that accepts $\{b^n\}^*$.

Let $\mathcal{C}$ be an NFA for the language $L(\mathcal{A}) \cup L(\mathcal{B})$. Due to the proof of Lemma 3 we observe: 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 $s_1, \dots, s_{m-1}$ from each of which a final state is reachable. Accordingly, $\mathcal{C}$ needs at least $n-1$ states $s'_1, \dots, s'_{n-1}$ for processing the inputs b^i, $1 \leq i \leq n-1$.

Denote by P_a resp. P_b the set of states that are reachable by inputs of the form a^i resp. b^i for $i \geq 1$. None of the final states may be reachable from the states in $P_a \cap P_b$. Otherwise words of the form $a^i b^j$ or $b^i a^j$ 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 words $\{a^m\}^+$. There must exist a final state s_m that accepts infinitely many of them. Thus, s_m is reachable from s_m itself. The same holds for a state s'_n for the words in $\{b^n\}^+$. It follows $s_m \in P_a$ and $s'_n \in P_b$ but $s_m \notin P_a \cap P_b$ and $s'_n \notin P_a \cap P_b$. Finally, the initial state s_0 must be a final state since $\lambda \in L(\mathcal{A}) \cup L(\mathcal{B})$, but $s_0 \neq s_m$ and $s_0 \neq s'_n$ since otherwise $\{a^m\}^i b^n$ or $\{b^n\}^i a^m$ would be accepted for some $i \in \mathbb{N}$. Altogether, $P_a \cup P_b$ must contain at least $m+n$ different states that are not equal to the initial state. □

When we are concerned with finite languages the state complexity of the union can be reduced by three states. For these upper bounds in the deterministic case see [2]. We may assume w.l.o.g. that minimal NFAs for finite languages not containing the empty word have only one final state. Since such NFAs do not contain any cycles they do contain at least one final (sink) state for which the transition function is not defined (otherwise they would not be reduced or would have a cycle). Now a given minimal NFA with more than one final state is modified such that a sink state becomes the only final state. Therefore simply the transition function has to be extended. Alternativ to changing to a former final state now the NFA nondeterministically may change to the sink state. If the finite language contains the empty word, then in addition the initial state is a second final one.

Corollary 6 *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 the previous theorem 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 final 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. Otherwise it would run through cycles. By the same argumentation as in the proof of Theorem 5 and merged initial and sink states we obtain at least $(m+1)+(n+1)-2$ states for an NFA that accepts $\{a^m\} \cup \{b^n\}$ (cf. Figure 2). □

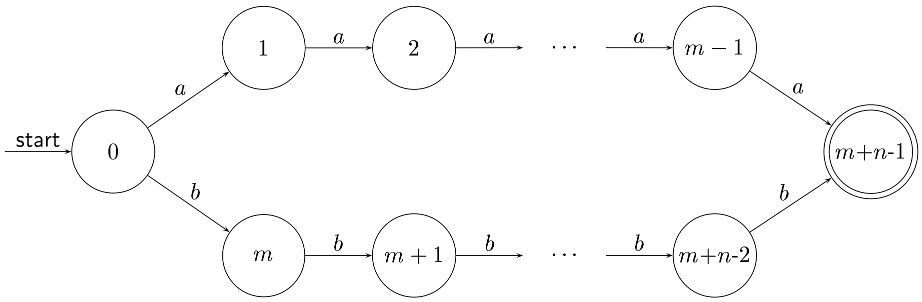

Figure 2: Minimal NFA accepting $\{a^m\} \cup \{b^n\}$.

The complementation on nondeterministic devices is often a difficult problem. In case of regular languages it is an expensive task at any rate. It is well known [11] 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 final and non-final states) we obtain an upper bound for the state complexity of the complementation on NFAs.

Corollary 7 *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 in the order of magnitude. Basically, the idea is to construct an efficiently acceptable language such that nondeterminism cannot do anything for a cheap and efficient acceptance of its complement.

Theorem 8 *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 the following $(k+3)$-state NFA $\mathcal{A} = \langle S, \{a,b\}, \delta, s_0, F\rangle$ (cf. Figure 3).

$$\begin{aligned}
&S = \{s_0, s_1, \cdots, s_{k+2}\}\\
&F = \{s_{k+2}\}\\
&\delta(s_0, a) = \{s_0, s_1\}\\
&\delta(s_0, b) = \{s_0\}\\
&\delta(s_i, a) = \delta(s_i, b) = \{s_{i+1}\}, \quad 1 \leq i \leq k\\
&\delta(s_{k+1}, b) = \{s_{k+2}\}\\
&\delta(s_{k+2}, a) = \delta(s_{k+2}, b) = \{s_{k+2}\}
\end{aligned}$$

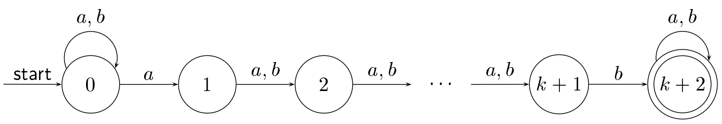

Figure 3: A $(k+3)$-state NFA accepting $\{a,b\}^* a \{a,b\}^k b \{a,b\}^*$.

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 a corresponding 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 in the input $\mathcal{B}$ has to remember the next k input symbols in addition what results in at least 2^{k+1} states (cf. Figure 4).

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 $\mathcal{B}$.

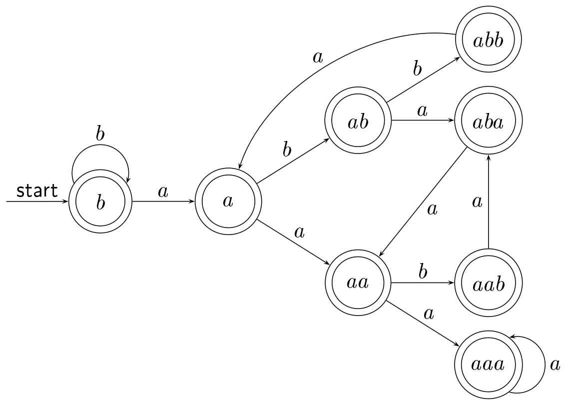

Figure 4: A minimal NFA accepting L_2 of Theorem 8.

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. $\square$

The situation for finite languages over an ℓ-letter alphabet, $\ell \geq 2$, is quite different, since the upper bound of the transformation to a deterministic finite automaton is different. In [12] 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 9 *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 the previous theorem yields:

Theorem 10 *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 5 accepts L_k with $2k+3$ states. (Trivially L_k is also accepted by an NFA with $2k+4$ states.)

An NFA $\mathcal{B}$ for the complement works similar to the corresponding NFA in the previous proof. 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 $2 = \ell^{\log_\ell 2} = \ell^{\frac{1}{\log_2 \ell}}$ we obtain the lower bound $\ell^{\frac{k+2}{\log_2 \ell}} \in \Omega(\ell^{\frac{n}{2 \cdot \log_2 \ell}})$. $\square$

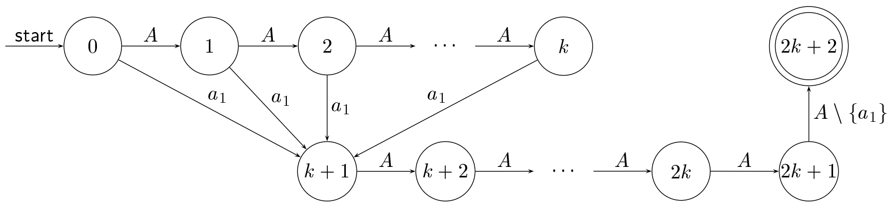

Figure 5: A $(2k+3)$-state NFA accepting L_k of Theorem 10.

Next we are going to prove a tight bound for the remaining Boolean operation, the intersection. The upper bound is obtained by the somehow old-fashioned cross-product construction.

Theorem 11 *For any integers $n, m \geq 1$ 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 $\mathcal{C} = \langle S, A, \delta, s_0, F\rangle$ defined by the cross-product of $\mathcal{A} = \langle S_A, A, \delta_A, s_{0,A}, F_A\rangle$ and $\mathcal{B} = \langle S_B, A, \delta_B, s_{0,B}, F_B\rangle$ accepts the language $L(\mathcal{A}) \cap L(\mathcal{B})$ with $m \cdot n$ states, i.e.,

$$
\begin{aligned}
&S = S_A \times S_B \\
&s_0 = (s_{0,A}, s_{0,B}) \\
&F = F_A \times F_B \\
&\delta((s_1, s_2), a) = (\delta_A(s_1, a), \delta_B(s_2, a)), \qquad \text{for } s_1 \in S_A, s_2 \in S_B, a \in A
\end{aligned}
$$

As witness languages for the fact that the bound is reached in the worst case define $L_k = \{w \in \{a,b\}^* \mid \#_a(w) \equiv 0 \pmod{k}\}$ for all $k \in \mathbb{N}$. An NFA that accepts L_k with k states is depicted in Figure 6.

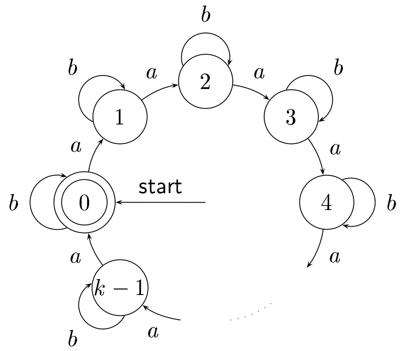

Figure 6: A k-state NFA accepting L_k of Theorem 11.

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 $a^{i'} b^{j'} a^{m-i} b^{n-j} \in L_m \cap L_n$. Contrarily, 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. □

4 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 cheap operations for NFAs. Again, this is essentially different when deterministic finite automata come to play. For example, in [16] a bound of $(2m-1) \cdot 2^{n-1}$ states has been shown for the DFA-concatenation, and in [13] a bound of $2^{n-1} + 2^{n-2}$ states for the iteration.

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. *Then $m + n$ states are sufficient and necessary in the worst case for an* NFA *$\mathcal{C}$ 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 final states in $\mathcal{A}$ with the states in $\mathcal{B}$ that follow the initial state. Let $\mathcal{A} = \langle S_A, A_A, \delta_A, s_{0,A}, F_A \rangle$ and $\mathcal{B} = \langle S_B, A_B, \delta_B, s_{0,B}, F_B \rangle$ with $S_A \cap S_B \neq \emptyset$, then $\mathcal{C} = \langle S, A, \delta, s_0, F \rangle$ is defined according to

$$\begin{aligned}
S &= S_A \cup S_B \\
A &= A_A \cup A_B \\
s_0 &= s_{0,A} \\
F &= F_B \\
\delta(s,a) &= \begin{cases} \delta_A(s,a) \text{ if } s \in S_A \setminus F_A \text{ and } a \in A_A \\ \delta_B(s,a) \text{ if } s \in S_B \text{ and } a \in A_B \\ \delta_A(s,a) \cup \delta_B(s_{0,B},a) \text{ if } s \in F_A \text{ and } a \in A_B \end{cases}
\end{aligned}$$

for $s \in S$ and $a \in A$.

The upper bound is reached for the concatenation of the languages $L(\mathcal{A}) = \{a^m\}^*$ and $L(\mathcal{B}) = \{b^n\}^*$. Let $\mathcal{C}$ be an NFA for the language $L(\mathcal{A})L(\mathcal{B})$. Following the idea of the proof of Theorem 5 we argue that $\mathcal{C}$ needs at least $m-1$ non-accepting states $s_1, \ldots, s_{m-1}$ to reject the inputs a^i for $1 \leq i \leq m-1$, and to accept the input a^m. For the processing of the inputs b^i with $1 \leq i \leq n-1$ and b^m the automaton $\mathcal{C}$ needs $n-1$ non-accepting states $s'_1, \ldots, s'_{n-1}$ in addition. From all these states a final state must be reachable, from which follows that they have to be pairwise different. Otherwise a word $a^i b^j$ with $i \not\equiv 0 \pmod m$ or a word $b^i a^j$ would be accepted. The necessarily final initial state must not be reached by any of the inputs $\{b^n\}^+$ since otherwise words of the form $b^i a^j$ would be accepted. Thus, $\mathcal{C}$ needs at least two final states. Together with the necessary number $(m-1)+(n-1)$ of non-accepting states the assertion follows. □

In case of finite languages the concatenation is one state cheaper.

Lemma 13 *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 *$\mathcal{C}$ 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 the previous theorem. Thus, 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 (cf. Figure 7). □

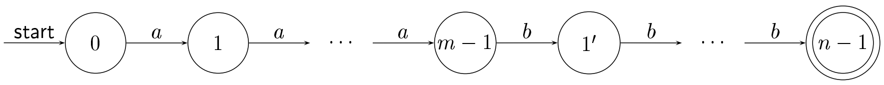

Figure 7: An $(m+n-1)$-state NFA accepting the concatenation $\{a^{m-1}\}\{b^{n-1}\}$ of Lemma 13.

The constructions yielding the upper bounds for the iteration and λ-free iteration are similarly. 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 14 *For any integer $n > 2$ let $\mathcal{A}$ be an 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:

$$\delta(s,a) = \begin{cases} \delta_A(s,a) \text{ if } s \notin F_A \\ \delta_A(s,a) \cup \delta_A(s_{0,A},a) \text{ if } s \in F_A \end{cases}$$

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

$$\begin{aligned} S &= S_A \cup \{s_0'\} \\ s_0 &= s_0' \\ F &= F_A \cup \{s_0'\} \\ \delta(s,a) &= \begin{cases} \delta_A(s,a) \text{ if } s \notin F_A \cup \{s_0'\} \\ \delta_A(s,a) \cup \delta_A(s_{0,A},a) \text{ if } s \in F_A \\ \delta_A(s_{0,A},a) \text{ if } s = s_0' \end{cases} \end{aligned}$$

for $s \in S$ and $a \in A$.

In order to prove the tightness of the bounds for any $n > 2$ let

$$L_n = \{w \in \{a, b\}^* \mid \#_a(w) \equiv n - 1 \pmod{n}\}$$

The language L_n is accepted by an n-state NFA. At first we show that $n+1$ states are necessary for $\mathcal{C} = \langle S, \{a, b\}, \delta, s_0, F\rangle$ 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 $a^1 \notin L(\mathcal{A})^*$ 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^1 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 final state is necessary. $\square$

The state complexity for the iterations in the finite language case is n resp. $n-1$.

Lemma 15 *For any integer $n > 1$ let $\mathcal{A}$ be an 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 14. 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 basing language is finite and the accepting automaton is reduced there must exist a final 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 $L_n = \{w \in \{a,b\}^* \mid |w| = n-1\}$ which requires n states. $L_n^+ = \{\{a,b\}^{n-1}\}^+$ is acceptable with at least n states (cf. Lemma 3).

The bound for the iteration is reached for the simple one-word language $L_n = \{ab^{n-1}\}$ that requires $n+1$ states. Clearly, $\{ab^{n-1}\}^*$ is acceptable with n states. □

5 Reversal

The last operation under consideration is the reversal. For deterministic automata one may expect that the state complexity is linear. But it is not. In [16] for infinite languages a tight bound of 2^n has been shown. A proof of a tight bound for finite languages can be found in [1]. It is of order $O(2^{\frac{n}{2}})$ for a two-letter alphabet. From the following cheap bounds for NFAs it follows once more that nondeterminism is a powerful concept.

Theorem 16 *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 *$\mathcal{C}$ 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 final states is a singleton or whose initial state is not within a loop. In general we are concerned with more than one accepting state and have to add a new initial state as shown below. If, in addition, the old initial state is part of a loop, then its role cannot be played by the new one and we obtain an $(n+1)$-state NFA.

Let $\mathcal{A} = \langle S_A, A, \delta_A, s_{0,A}, F_A\rangle$ be an n-state NFA. Define $\mathcal{C} = \langle S, A, \delta, s_0, F\rangle$ according to

$$S = S_A \cup \{s_0\}, \text{ where } s_0 \notin S_A$$
$$F = \begin{cases} \{s_{0,A}\} \text{ if } \lambda \notin L(\mathcal{A}) \\ \{s_{0,A}, s_0\} \text{ otherwise} \end{cases}$$
$$\delta(s,a) = \begin{cases} \{s' \in S_A \mid s \in \delta_A(s',a)\} \text{ if } s \in S_A \\ \{s' \in S_A \mid \delta_A(s',a) \cap F_A \neq \emptyset\} \text{ if } s = s_0 \end{cases}$$

for $s \in S$, $a \in A$.

Clearly, the $(n+1)$-state NFA $\mathcal{C}$ accepts the reversal of $L(\mathcal{A})$.

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 8.

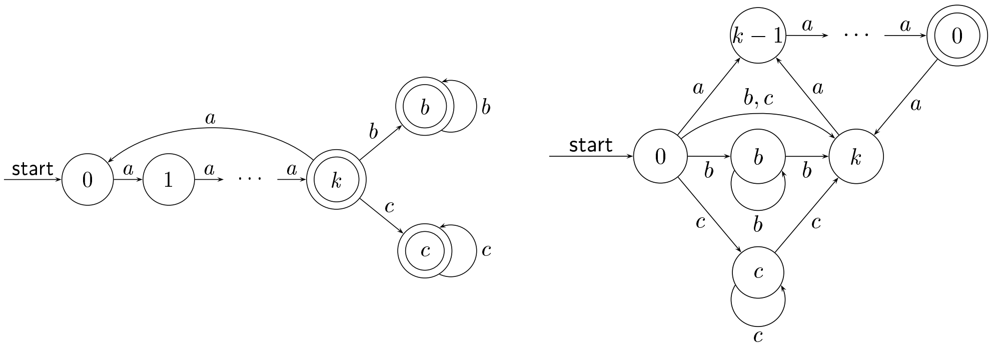

Figure 8: A $(k+3)$-state and a $(k+4)$-state NFA accepting L_k and L_k^R of Theorem 16.

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. □

The fact that NFAs for finite languages do not have any cycle leads once more to the possibility of saving one state compared with the infinite case.

Lemma 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. Recall from the proof of Corollary 6 that for every minimal n-state NFA that accepts a non-empty finite language there exists an equivalent n-state NFA that has only one final state. By the construction of the previous proof we obtain an $(n+1)$-state NFA that has an unreachable state. It is the unique former final 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. It holds also for the empty language.

Finally, Table 1 summarizes the shown state complexity bounds for NFAs.

	NFA		DFA	
	finite	infinite	finite	infinite
$\cup$	$m+n-2$	$m+n+1$	$O(mn)$	mn
$\sim$	$O(\ell^{\frac{n}{\log_2 \ell+1}})$	$O(2^{n-2})$	n	n
$\cap$	$O(mn)$	mn	$O(mn)$	mn
R	n	$n+1$	$O(2^{\frac{n}{2}})$	2^n
$\cdot$	$m+n-1$	$m+n$	$O(mn^{t-1}+n^t)$	$(2m-1)2^{n-1}$
$*$	$n-1$	$n+1$	$2^{n-3}+2^{n-4}$	$2^{n-1}+2^{n-2}$
$+$	n	n		

Table 1: Comparison of the NFA and DFA state complexities (ℓ is the number of states, t is the number of final states of the 'left' automaton.

References

[1] Câmpeanu, C., Čulik, K., Salomaa, K., and Yu, S. *State complexity of basic operations on finite languages*. International Workshop on Implementing Automata, 1999, to appear.

[2] Câmpeanu, C., Čulik, K., Sântean, N., and Yu, S. *Finite languages and cover-automata*. Theoretical Computer Science, to appear.

[3] Chrobak, M. *Finite automata and unary languages*. Theoretical Computer Science 47 (1986), 149–158.

[4] Goldstine, J., Kintala, C., and Wotschke, D. *On measuring nondeterminism in regular languages*. Information and Computation 86 (1990), 179–194.

[5] Goldstine, J., Leung, H., and Wotschke, D. *On the relation between ambiguity and nondeterminism in finite automata*. Information and Computation 100 (1992), 261–270.

[6] Hopcroft, J. E. and Ullman, J. D. *Introduction to Automata Theory, Language, and Computation*. Addison-Wesley, Reading, Massachusetts, 1979.

[7] Kintala, C. M. and Wotschke, D. *Amounts of nondeterminism in finite automata*. Acta Informatica 13 (1980), 199–204.

[8] Leiss, E. *Succinct representation of regular languages by Boolean automata*. Theoretical Computer Science 13 (1981), 323–330.

[9] Mereghetti, C. and Pighizzini, G. *Optimal simulations between unary automata*. Symposium on Theoretical Aspects of Computer Science, LNCS 1373, 1998, pp. 139–149.

[10] Mereghetti, C. and Pighizzini, G. *Unary automata simulations and cyclic languages*. International Workshop on Descriptional Complexity of Automata, Grammars and Related Structures, 1999, pp. 145–153.

[11] Meyer, A. R. and Fischer, M. J. *Economy of description by automata, grammars, and formal systems*. IEEE Symposium on Switching and Automata Theory, 1971, pp. 188–191.

[12] Salomaa, K. and Yu, S. *NFA to DFA transformation for finite languages over arbitrary alphabets*. Journal of Automata, Languages and Combinatorics 2 (1997), 177–186.

[13] Yu, S. *Regular languages*. In Rozenberg, G. and Salomaa, A. (eds.), *Handbook of Formal Languages I*. Springer, 1997, chapter 2, pp. 41–110.

[14] Yu, S. *State complexity of regular languages*. International Workshop on Descriptional Complexity of Automata, Grammars and Related Structures, 1999, pp. 77–88.

[15] Yu, S. and Zhuang, Q. *On the state complexity of intersection of regular languages*. SIGACT News 22.3 (1991), 52–54.

[16] Yu, S., Zhuang, Q., and Salomaa, K. *The state complexities of some basic operations on regular languages*. Theoretical Computer Science 125 (1994), 315–328.