Journal of Automata, Languages and Combinatorics **XX** (201Y) Z, xxx–yyy
© Otto-von-Guericke-Universität Magdeburg

ON THE NUMBER OF ACCEPTING STATES OF FINITE AUTOMATA

Jürgen Dassow
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
PSF 4120, D-39016 Magdeburg, Germany
e-mail: `dassow@iws.cs.uni-magdeburg.de`

ABSTRACT
In this paper, we start a systematic study of the number of accepting states. For a regular language L, we define the complexity $\mathtt{asc}(L)$ as the minimal number of accepting states necessary to accept L by deterministic finite automata. With respect to nondeterministic automata, the corresponding measure is $\mathtt{nasc}(L)$. We prove that, for any non-negative integer n, there is a regular language L such that $\mathtt{asc}(L) = n$, whereas we have $\mathtt{nasc}(R) \leq 2$ for any regular language R. Moreover, for a k-ary regularity preserving operation $\circ$ on languages, we define $g_\circ^{\mathtt{asc}}(n_1, n_2, \ldots, n_k)$ as the set of all integers r such that there are k regular languages L_i, $1 \leq i \leq k$, such that $\mathtt{asc}(L_i) = n_i$ for $1 \leq i \leq k$ and $\mathtt{asc}(\circ(L_1, L_2, \ldots, L_k)) = r$. We determine this set for the operations complement, union, product, and Kleene closure and give a partial result for set difference.

1. Introduction

State complexity is a fundamental part of automata theory. In the last 25 years, its importance draws from many applications, e. g. in natural language and speech processing, software engineering etc. where systems/automata with a large number of states are used for the description of natural languages or the behaviour of certain software systems. Thus one is interested in questions as minimization, construction of relatively small finite automata from other devices, etc.

We mention some of the classical known facts. For any natural number n, there is a regular language R such that the acceptance of R by deterministic or nondeterministic finite automata requires an automaton with at least n states. There is an algorithm which, for a regular language R with state complexity $\mathtt{sc}(R) = n$, constructs a deterministic finite automaton which has n states and accepts R. By the classical power-set-construction, any nondeterministic finite automaton $\mathcal{A}$ with n states can be transformed into a deterministic finite automaton B with 2^n states such that $T(\mathcal{A}) = T(\mathcal{B})$, i. e., both automata accept the same language; moreover, already in the sixties, it was independently shown by Lupanov, Moore, Meyer and Fischer, that there are nondeterministic finite automata $\mathcal{A}$ with n states such that any deterministic finite automaton B with $T(\mathcal{A}) = T(\mathcal{B})$ has 2^n states. For any two regular languages L_1 and L_2 with $\mathtt{sc}(L_1) = m$ and $\mathtt{sc}(L_2) = n$, we have $\mathtt{sc}(L_1 \cup L_2) \leq m \cdot n$; and furthermore, there are regular languages L_1 and L_2 with $\mathtt{sc}(L_1) = m$ and $\mathtt{sc}(L_2) = n$ such that $\mathtt{sc}(L_1 \cup L_2) = m \cdot n$.

However, there is another measure concerning finite automaton: the number of accepting states, i. e., we are not interested in all states of the automaton and count the accepting states only. This measure already occurred in some situations. For instance (see e. g. [10]), if we consider the number states which are necessary to accept the concatenation $T(\mathcal{A}_1) \cdot T(\mathcal{A}_2)$ for two deterministic finite automata $\mathcal{A}_1$ and $\mathcal{A}_2$, then it is known that $\mathtt{sc}(T(\mathcal{A}_1) \cdot T(\mathcal{A}_2))$ depends on the number of accepting states of $\mathcal{A}_1$. An analogous situation holds if we consider the cut-operation instead of concatenation (see [2]).

The state complexity was also considered for Büchi automata which accept languages of infinite words. In the paper [1], the authors discuss the minimization of Büchi automata. The time complexity of the method they propose also depends on the number of accepting states.

Büchi automata are often used in model checking where the model is based on linear temporal logic. In the papers [3] and [4], the authors study algorithms for such model checking; they show that the complexities of their algorithms depend on the number of accepting states of the Büchi automaton, too.

In this paper we start a systematic investigation of the number of accepting states of finite automata. We shall see that, with respect to minimization this measure has properties analogous to those known from "classical" state complexity, however, with respect to the behaviour under operations, the measure number of accepting states essentially differs from that of state complexity.

The paper is organized as follows: In Section 2, we give the definitions of the considered complexity measures of finite automata and regular languages. Furthermore, we present some basic properties of the measures. In Section 3, we discuss the behaviour of the number of accepting states with respect to the operations complement, union, product, Kleene closure, and set difference. In Section 4, we present some concluding remarks.

2. Definitions and Basic Results

We assume that the reader is familiar with the basic concepts of the theory of automata and formal languages (see e. g. [9]). Here we give some notation.

By $\mathrm{card}(M)$ we denote the cardinality of the set M. The set of all positive integers is denoted by $\mathbb{N}$.

The empty word and the length of a word w are designated by λ and $|w|$, respectively.

For a language L, by $\mathrm{alph}(L)$ we denote the minimal alphabet X (with respect to inclusion) such that $L \subseteq X^*$, and define the complement $C(L)$ by $C(L) = \mathrm{alph}(L)^* \setminus L$.

In this paper we consider only regular languages; therefore we omit the adjective "regular".

A nondeterministic finite automaton (shortly written as NFA) is a quintuple $\mathcal{A} = (X, Z, z_0, F, \delta)$ where X is a finite non-empty set of input symbols, Z is a finite non-empty set of states, $z_0 \in Z$ is the initial state, $F \subseteq Z$ is the set of accepting states, and $\delta : Z \times X \to 2^Z$ is the transition function of $\mathcal{A}$. The language $T(\mathcal{A})$ accepted by $\mathcal{A}$ is defined as

$$T(\mathcal{A}) = \{w \mid \delta(z_0, w) \cap F \neq \emptyset\},$$

where the transition function δ is recursively extended to a mapping $\delta : Z \times X^* \to Z$ as usual.

If δ is a total mapping and $\delta(z, x)$ is a singleton for any $z \in Z$ and any $x \in X$, then $\mathcal{A}$ is called a deterministic finite automaton (DFA, for short). In case of determinism, we write simply $\delta(z, x) = z'$ instead of $\delta(z, x) = \{z'\}$.

Let $\mathcal{A} = (X, Z, z_0, F, \delta)$ be a DFA. Then we define

$$\texttt{sc}(\mathcal{A}) = \text{card}(Z) \text{ and } \texttt{asc}(\mathcal{A}) = \text{card}(F).$$

For a regular language L, we define

$$\texttt{sc}(L) = \min\{\texttt{sc}(\mathcal{A}) \mid \mathcal{A} \text{ is a DFA and } T(\mathcal{A}) = L\}$$

and

$$\texttt{asc}(L) = \min\{\texttt{asc}(\mathcal{A}) \mid \mathcal{A} \text{ is a DFA and } T(\mathcal{A}) = L\}.$$

We call the functions sc and asc the *state complexity* and *accepting state complexity*, respectively.

A deterministic finite automata $\mathcal{A}$ is called *minimal* if $\texttt{sc}(\mathcal{A}) = \texttt{sc}(T(\mathcal{A}))$, i. e., a minimal DFA has the minimal number of states which is necessary to accept the language $T(\mathcal{A})$. There exists a well-developed theory of minimization of finite automata (see e. g. [8]). We assume that the reader is familiar with basic concepts of minimization as equivalent states, isomorphism of automata, etc.

A deterministic finite automata $\mathcal{A}$ is called *a-minimal* if $\texttt{asc}(\mathcal{A}) = \texttt{asc}(T(\mathcal{A}))$. We first show that the minimization of automata with respect to the number of states is also useful for the minimization with respect to the number of accepting states.

Theorem 1 *i) If $\mathcal{A}$ is a minimal automaton, then we have $\texttt{asc}(T(\mathcal{A})) = \texttt{asc}(\mathcal{A})$, i. e., any minimal DFA is also a-minimal.*

ii) There is an algorithm which, for a given regular language L, determines $\texttt{asc}(L)$.

Proof. i) Let $\mathcal{A} = (X, Z, z_0, F, \delta)$ be a minimal automaton. Assume that $\texttt{asc}(T(\mathcal{A})) < \text{card}(F)$. Then there is an a-minimal DFA $\mathcal{B} = (X, Z', z_0', F', \delta')$ such that $T(\mathcal{A}) = T(\mathcal{B})$ and $\text{card}(F') = \texttt{asc}((T(\mathcal{A}))$. We now apply a minimization algorithm to $\mathcal{B}$ and obtain a DFA $\mathcal{B}' = (X, Z'', z_0'', F'', \delta'')$. Obviously, $\text{card}(F') = \text{card}(F'')$ since $\mathcal{B}$ is a-minimal. Moreover, $\mathcal{B}'$ is also minimal by construction and $T(\mathcal{B}') = T(\mathcal{A})$. Thus $\mathcal{A}$ and $\mathcal{B}'$ are isomorphic which implies that $\text{card}(F) = \text{card}(F'')$. But this equality contradicts $\text{card}(F) > \texttt{asc}(T(\mathcal{A})) = \text{card}(F'')$.

ii) Let $\mathcal{A}$ be a DFA which accepts L (note that $\mathcal{A}$ can be constructed if L is given by a NFA, or a regular grammar, or a regular expression etc.). Now we apply a minimization algorithm to $\mathcal{A}$. The cardinality of the set of accepting states of the resulting automaton gives then $\texttt{asc}(L)$ by i). $\square$

We note that the converse of Theorem 1, i) does not hold. Obviously, the deterministic finite automaton $\mathcal{A} = (\{a\}, \{z_0, z_1, z_2, z_3\}, z_0, \{z_1\}, \delta)$ with $\delta(z_i, a) = z_{i+1}$ for $i \in \{0, 1, 2\}$ and $\delta(z_3, a) = z_2$ is a-minimal, since its number of accepting states is 1. On the other hand, $\mathcal{A}$ is not minimal, because the states z_2 and z_3 are equivalent.

By Theorem 1 we determine the complexity $\texttt{asc}$ of a family of languages which are needed later.

Example 1 For any alphabet X and any integers $k, r_1, r_2, \ldots, r_k, r, s$ with $k \geq 0$, $0 \leq r_1 < r_2 < \cdots < r_k < r$, $s \geq 0$, and $r - r_k \neq s$, let

$$L_X(r_1, r_2, \ldots, r_k; r, s) = \{w \mid w \in X^*, |w| \in \{r_1, r_2, \ldots, r_k\} \cup \{r + is \mid i \in \{0\} \cup \mathbb{N}\}\}.$$

Then we have

$$\texttt{asc}(L_X(r_1, r_2, \ldots, r_k; r, s)) = k + 1.$$

Let $r - r_k > s$. We consider the DFA

$$\mathcal{A} = (X, \{z_0, z_1, \ldots, z_r\}, z_0, \{z_{r_1}, z_{r_2}, \ldots, z_{r_k}, z_r\}, \delta)$$

with

$$\delta(z_i, a) = z_{i+1} \text{ for } 0 \leq i \leq r - 1,\ \delta(z_r, a) = z_{r-s+1}.$$

It is easy to see that $T(\mathcal{A}) = L_X(r_1, r_2, \ldots, r_k; r, s))$. We prove that $\mathcal{A}$ is minimal by showing that all states z_u and z_v with $u \neq v$ are pairwise not equivalent. Obviously, without loss of generality we can assume that $u > v$. We distinguish the following cases.

$u-v$ is not a multiple of s. Let w be a word of length $r-v$. Then $\delta(z_v, w) = z_r \in F$, $\delta(z_u, w) = z_q$ with $q \geq r_s + 1$ and $q \neq r$, i.e., $\delta(z_u, w) \notin F$. Therefore z_u and z_v are not equivalent.

$u - v$ is a multiple of s. If $v \leq r_k$, we choose a word w' of length $r - s - y$ and get $\delta(z_v, w') = z_{r-s} \notin F$ and $\delta(z_u, w') = z_r \in F$, which proves that z_u and z_v are not equivalent. If $v > r_k$, we choose a word w'' of length $r - u$. Then $\delta(z_u, w'') = z_r \in F$ and $\delta(z_v, w'') = z_{q'}$ with $r_k < q' < r$, i.e., $\delta(z_v, w'') \notin F$, which proves that z_u and z_v are not equivalent.

By Theorem 1, $\texttt{asc}(T(\mathcal{A})) = \texttt{asc}(L_X(r_1, r_2, \ldots, r_k; r, s)) = \text{card}(F) = k + 1$.

If $r - r_k = u < s$, then we consider the DFA

$$\mathcal{A} = (X, \{z_0, z_1, \ldots, z_{r+s-u-1}\}, z_0, \{z_{r_1}, z_{r_2}, \ldots, z_{r_k}, z_r\}, \delta)$$

with

$$\delta(z_i, a) = z_{i+1} \text{ for } 0 \leq i \leq r - +s - u - 2,\ \delta(z_{r+s-u-1}, a) = z_{r-s+u},$$

prove its minimality and $\texttt{asc}(L_X(r_1, r_2, \ldots, r_k; r, s)) = \text{card}(F) = k + 1$ as above.

Obviously, $L_X(r_1, r_2, \ldots, r_k; r, 0) = \{w \mid w \in X^*, |w| \in \{r_1, r_2, \ldots, r_k, r\}\}$, for which we also use the notation $L_X(r_1, r_2, \ldots, r_k, r)$.

From Example 1, we immediately obtain the following statement which shows that `asc` is a useful complexity measure for DFAs.

Theorem 2 *i) For any alphabet X and any natural number $n \geq 0$, there is a finite language L_n over X such that $\mathtt{asc}(L_n) = n$.*

ii) For any alphabet X and any natural number $n \geq 1$, there is an infinite language L_n over X such that $\mathtt{asc}(L_n) = n$.

Obviously, $\mathtt{asc}(L) = 0$ if and only if $L = \emptyset$. Therefore, we have the restriction $n \geq 1$ in Theorem 2, ii).

We now consider nondeterministic finite automata. For an NFA $\mathcal{A} = (X, Z, z_0, F, \delta)$, we define

$$\mathtt{nsc}(\mathcal{A}) = \mathrm{card}(Z) \text{ and } \mathtt{nasc}(\mathcal{A}) = \mathrm{card}(F),$$

and, for a regular language L, we set

$$\mathtt{nsc}(L) = \min\{\mathtt{nsc}(\mathcal{A}) \mid \mathcal{A} \text{ is an NFA and } T(\mathcal{A}) = L\}$$

and

$$\mathtt{nasc}(L) = \min\{\mathtt{nasc}(\mathcal{A}) \mid \mathcal{A} \text{ is an NFA and } T(\mathcal{A}) = L\}.$$

However, the measure `nasc` is not very interesting, since we have the following result.

Theorem 3 *i) For any regular language L, $\mathtt{nasc}(L) \leq 2$.*

ii) For any non-empty regular language L with $\lambda \notin L$, $\mathtt{nasc}(L) = 1$.

Proof. i) Let $\mathcal{A} = (X, Z, z_0, F, \delta)$ be a deterministic finite automata such that $T(\mathcal{A}) = L$. Then we construct the NFA $\mathcal{B} = (X, Z \cup \{q\}, z_0, F', \delta')$ with

$$\begin{aligned}
\delta'(z, a) &= \{\delta(z, a)\} && \text{for } \delta(z, a) \notin F \text{ or } \delta(z, a) = z_0, \\
\delta'(z, a) &= \{\delta(z, a), q\} && \text{for } \delta(z, a) \in F \setminus \{z_0\}, \\
F' &= \begin{cases} \{q\} & \text{for } z_0 \notin F \\ \{z_0, q\} & \text{for } z_0 \in F \end{cases}.
\end{aligned}$$

For any $w \in X^*$, $z_0 \in \delta'(z_0, w)$ if and only if $z_0 = \delta(z_0, w)$. Moreover, for $w = w'a$ with $a \in X$, $q \in \delta'(z_0, w)$ if and only if there is a $z' \in Z$ such that $z' \in \delta'(z_0, w')$ and $\delta(z', a) \in F$ if and only if $z' = \delta(z_0, w')$ and $\delta(z', a) \in F$ if and only if $\delta(z_0, w'a) = \delta(z_0, w) \in F$. From these facts, it follows that $\delta(z_0, w) \cap F' \neq \emptyset$ if and only if $\delta(z_0, w) \in F$. This implies $T(\mathcal{B}) = T(\mathcal{A})$. Obviously, we have $\mathtt{nasc}(\mathcal{B}) \leq 2$. Therefore $\mathtt{nasc}(L) \leq 2$.

ii) If $\lambda \notin L$, then $z_0 \notin F$. The construction in i) shows $\mathtt{nasc}(\mathcal{B}) \leq 1$ and hence $\mathtt{nasc}(L) \leq 1$. Because $\mathtt{nasc}(L) = 0$ holds if and only if $L = \emptyset$, we get $\mathtt{nasc}(L) = 1$. □

First, we note that there also exist regular languages L with $\lambda \in L$ and $\mathtt{nasc}(L) = 1$. As an example we give the language $\{a^{2n} \mid n \geq 0\}$ which is accepted by the DFA $\mathcal{A} = (\{a\}, \{z_0, z_1\}, z_0, \{z_0\}, \delta)$ with $\delta(z_0, a) = z_1$ and $\delta(z_1, a) = z_0$ with one accepting state.

It is left as an *open problem* to characterize those regular languages which satisfy $\mathtt{nasc}(L) = 1$.

We remark that Theorem 3 differs essentially from the fact that, for any n, there is a regular language L such that $\mathtt{nsc}(L) = n$.

By Theorem 2 (and its proof) and Example 1, it is easy to see that, for any natural number n, there is a regular language L_n such that $\mathtt{nasc}(L_n) = 1$ (or $\mathtt{nasc}(L_n) = 2$) and $\mathtt{asc}(L_n) = n$.

3. Behaviour under Operations

We first define a set of integers which characterizes the behaviour of complexities under operations.

Definition 1 *For $c \in \{\mathtt{sc}, \mathtt{asc}\}$, a k-ary regularity preserving operation $\circ$ on languages, and natural numbers $n_1, n_2, \ldots, n_k$, we define $g_\circ^c(n_1, n_2, \ldots, n_k)$ as the set of all integers r such that there are k regular languages L_i, $1 \leq i \leq k$, such that $c(L_i) = n_i$ for $1 \leq i \leq k$ and $c(\circ(L_1, L_2, \ldots, L_k)) = r$.*

We recall some results for the state complexity (see [5]).

For complement, it is obvious that

$$g_C^{\mathtt{sc}}(m) = \{m\} \text{ for } m \geq 1.$$

For union (denoted by $\cup$), in [6], M. Hricko, G. Jirásková, and A. Szabari showed that

$$g_\cup^{\mathtt{sc}}(m, n) = \{1, 2, \ldots, mn\} \text{ for } m \geq 2 \text{ and } n \geq 2.$$

For Kleene closure (denoted by $*$), by G. Jirásková in [7], it was proved that

$$g_*^{\mathtt{sc}}(m) = \begin{cases} \{1, 2\} & \text{for } m = 1 \\ \{1, 2, \ldots, 2^{m-1} + 2^{m-2}\} & \text{for } m \geq 2 \end{cases}.$$

We now discuss $g_\circ^{\mathtt{asc}}$.

Theorem 4 *We have*

$$g_C^{\mathtt{asc}}(m) = \begin{cases} \{1\} & \text{for } m = 0 \\ \{0\} \cup \mathbb{N} & \text{for } m = 1 \\ \mathbb{N} & \text{for } m \geq 2 \end{cases}.$$

Proof. Let $L \subseteq X^*$ be a language with $\mathtt{asc}(L) = 0$. Then $L = \emptyset$ and $C(L) = X^*$. Hence $\mathtt{asc}(C(L)) = 1$.

Let $m \geq 1$. We consider the language $L = L_X(0, 1, \ldots, m-2; m+r-1, 1)$. Then we have $\mathtt{asc}(L) = m$ by Example 1. Moreover, $C(L) = L_X(m-1, m, \ldots, m+r-2)$ which satisfies $\mathtt{asc}(C(L)) = r$ by Example 1. Thus $\mathbb{N} \subseteq g_C^{\mathtt{asc}}(m)$.

Let $m \geq 2$. If $\mathtt{asc}(L) = m$ for some language L, then $L \neq X^*$ for all X. Hence $C(L) \neq \emptyset$ and $\mathtt{asc}(C(L)) \geq 1$. Therefore $0 \notin g_C(m)$ and $g_C^{\mathtt{asc}}(m) = \mathbb{N}$.

Let $m = 1$. For X^*, we get $\mathtt{asc}(X^*) = 1$ and $\mathtt{asc}(C(X^*)) = \mathtt{asc}(\emptyset) = 0$. Thus $g_C^{\mathtt{asc}}(1) = \{0\} \cup \mathbb{N}$. □

Lemma 5 *For any three positive integers m, n, r with $r \geq m \geq n \geq 1$ and any alphabet X, there are languages L_m and L_n such that* $\mathrm{alph}(L_m) = \mathrm{alph}(L_n) = X$, $\mathtt{asc}(L_m) = m$, $\mathtt{asc}(L_n) = n$ *and* $\mathtt{asc}(L_m \cup L_n) = \mathtt{asc}(L_m \cdot L_n) = r$.

Proof. Let x be a number with $x \geq m + 1 + 2(r-m)$. We consider

$$L_m = L_X(0, x+1, x+2, \ldots, x+m-2; x+m+1+2(r-m), 1)$$

and

$$L_n = L_X(0, x+1, x+2, \ldots, x+n-2; x+m+1, 2).$$

By Example 1, $\mathtt{asc}(L_m) = m$ and $\mathtt{asc}(L_n) = n$. Clearly,

$$\begin{aligned} L_m \cup L_n = L_X(0, x+1, x+2, \ldots, x+m-2, \\ x+m+1, x+m+3, x+m+1+2(r-m)-2; \\ x+m+1+2(r-m), 1). \end{aligned}$$

By Example 1, $\mathtt{asc}(L_m \cup L_n) = r$.

Since $\lambda \in L_m$ and $\lambda \in L_n$, $L_m \cup L_n \subseteq L_m \cdot L_n$. Moreover, for two non-empty words $w_1 \in L_m$ and $w_2 \in L_n$, we have $|w_1 \cdot w_2| \geq x + x \geq x + m + 1 + 2(r-m)$ which proves that $w_1 \cdot w_2 \in L_m \cup L_n$. Therefore $L_m \cdot L_n = L_m \cup L_n$ and, consequently, $\mathtt{asc}(L_m \cdot L_n) = r$. □

Lemma 6 *For any three positive integers m, n, r with $m > r \geq 1$ and $n \geq 1$ and any alphabet X, there are languages L_m and L_n such that* $\mathrm{alph}(L_m) = \mathrm{alph}(L_n) = X$, $\mathtt{asc}(L_m) = m$, $\mathtt{asc}(L_n) = n$ *and* $\mathtt{asc}(L_m \cup L_n) = \mathtt{asc}(L_m \cdot L_n) = r$.

Proof. Let $m \geq 2$ and $n \geq 1$. We consider

$$L_m = L_X(0, 1, 2, \ldots, r-2, r+1, r+2 \ldots m; m+2, 1)$$

and

$$L_n = L_X(0, m+1, m+2, \ldots m+n-1) \text{ for } n \geq 2$$

and

$$L_n = L_X(; 0, m+1) \text{ for } n = 1.$$

By Example 1, $\mathtt{asc}(L_m) = m$ and $\mathtt{asc}(L_n) = n$. Furthermore,

$$L_m \cup L_n = L_X(0, 1, 2, \ldots, r-2; r+1, 1),$$

which gives $\mathtt{asc}(L_m \cup L_n) = r$ by Example 1.

Moreover, since $\lambda \in L_m$ and $\lambda \in L_n$, we have $L_m \cup L_n \subseteq L_m \cdot L_n$. Furthermore, for two non-empty wordw $w_1 \in L_m$ and $w_2 \in L_n$, we have $|w_1 \cdot w_2| \geq |w_1| + m + 1 \geq m + 2 \geq r + 1$ which proves that $w_1 \cdot w_2 \in L_m \cup L_n$. Therefore $L_m \cdot L_n = L_m \cup L_n$. Thus, $\mathtt{asc}(L_m \cdot L_n) = r$. □

Since union is a commutative operation, we have the following statement.

Lemma 7 *For any non-negative integers m and n, $g_\cup^{\mathtt{asc}}(m,n) = g_\cup^{\mathtt{asc}}(n,m)$.*

Theorem 8 *We have*

$$g_\cup^{\mathtt{asc}}(m,n) = \begin{cases} \{m\} & \text{for } n = 0 \\ \{n\} & \text{for } m = 0 \\ \mathbb{N} & \text{for } m \geq 1,\ n \geq 1 \end{cases}.$$

Proof. If $\mathtt{asc}(L_m) = 0$, then $L_m = \emptyset$ and therefore $L_m \cup L_n = L_n$ which implies $\mathtt{asc}(L_m \cup L_n) = \mathtt{asc}(L_n) = n$. Thus $g_\cup^{\mathtt{asc}}(0,n) = \{n\}$.

Analogously, $g_\cup^{\mathtt{asc}}(m,0) = \{m\}$ can be shown.

If $m \geq 1$ and $n \geq 1$, then L_m and L_n are not empty, and consequently $L_m \cup L_n \neq \emptyset$ which proves $\mathtt{asc}(L_m \cup L_n) \geq 1$.

If $m \geq n$, then any positive number can be obtained as $\mathtt{asc}(L_m \cup L_n)$ by Lemmas 5 and 6. Thus $g_\cup^{\mathtt{asc}}(m,n) = \mathbb{N}$.

If $n \geq m$, then from the preceding considerations we have $g_\cup^{\mathtt{asc}}(n,m) = \mathbb{N}$. By Lemma 7, $g_\cup^{\mathtt{asc}}(m,n) = \mathbb{N}$. □

Theorem 9 *We have*

$$g_\cdot^{\mathtt{asc}}(m,n) = \begin{cases} \{0\} & \text{for } n = 0 \\ \{0\} & \text{for } m = 0 \\ \mathbb{N} & \text{for } m \geq 1,\ n \geq 1 \end{cases}.$$

Proof. If $\mathtt{asc}(L_m) = 0$, then $L_m = \emptyset$ and therefore $L_m \cdot L_n = \emptyset$ which implies $\mathtt{asc}(L_m \cdot L_n) = 0$. Thus $g_\cdot^{\mathtt{asc}}(0,n) = \{0\}$.

Analogously, $g_\cdot^{\mathtt{asc}}(m,0) = \{0\}$ can be shown.

If $m \geq 1$ and $n \geq 1$, then L_m and L_n are not empty, and consequently $L_m \cdot L_n \neq \emptyset$ which proves $\mathtt{asc}(L_m \cup L_n) \geq 1$.

Let $m \geq n$. Then any positive number can be obtained as $\mathtt{asc}(L_m \cdot L_n)$ by Lemmas 5 and 6. Thus $g_\cdot^{\mathtt{asc}}(m,n) = \mathbb{N}$.

Let $n \geq m$, then from the preceding considerations we have $g_\cdot^{\mathtt{asc}}(n,m) = \mathbb{N}$. Since for all languages used in the proof of lemmas 5 and 6, $L_m \cdot L_n = L_n \cdot L_m$ holds, we get $g_\cdot^{\mathtt{asc}}(m,n) = \mathbb{N}$. □

For the set difference (denoted by \), the following result holds.

Theorem 10 *We have*

$$g_{\setminus}^{\mathtt{asc}}(0,n) = \{0\},$$
$$g_{\setminus}^{\mathtt{asc}}(m,0) = \{m\},$$
$$\{r \mid r \in \mathbb{N} \cup \{0\},\ r \geq m-n\} \subseteq g_{\setminus}^{\mathtt{asc}}(m,n) \text{ for } m \geq 1,\ n \geq 1.$$

Proof. The first two statements immediately follow from $\emptyset \setminus K = \emptyset$ and $K \setminus \emptyset = K$ for all languages K and $\mathtt{asc}(L) = 0$ if and only if $L = \emptyset$.

For $r > m \geq 1$ and $n \geq 1$, we consider the languages

$$L_m = \{1, 2, \ldots, m-1; m+n+1, 1)$$

and

$$L_n = \{m+1, m+2, \ldots m+n-1, r+n+1\}$$

with $\mathtt{asc}(L_m) = m$ and $\mathtt{asc}(L_n) = n$. Then

$$L_m \setminus L_n = \{1, 2, \ldots, m-1, m+n+1, m+n+2, \ldots, r+n; r+n+2, 1)$$

satisfies $\mathtt{asc}(L_m \setminus L_n) = (m-1) + (r-m) + 1 = r$.

For $m \geq r \geq m-n$, and $n \geq 1$, we consider the languages

$$L_m = \{1, 2, \ldots, m-1, m)$$

and

$$L_n = \{r+1, r+2, \ldots m, m+1, m+2, \ldots, r+n\}$$

with $\mathtt{asc}(L_m) = m$ and $\mathtt{asc}(L_n) = n$. Then $L_m \setminus L_n = \{1, 2, \ldots, r)$, which satisfies $\mathtt{asc}(L_m \setminus L_n) = r$. □

Obviously, we have $g_{\setminus}^{\mathtt{asc}}(m,n) = \mathbb{N} \cup \{0\}$ for $n \geq m \geq 1$; however, the exact determination of $g_{\setminus}^{\mathtt{asc}}(m,n)$ for $m \geq n \geq 1$ remains an open problem.

Finally we discuss the Kleene closure denoted by $*$ and the positive Kleene closure denoted by $+$.

Theorem 11 *We have*

$$g_{*}^{\mathtt{asc}}(m) = \begin{cases} \{1\} & \text{for } m = 0 \\ \mathbb{N} & \text{for } m \geq 1 \end{cases}.$$

Proof. First we note that $L^* \neq \emptyset$ for all languages L. Hence $0 \notin g_*(m)$ for all $m \geq 0$

If $m = 0$, then $L_0 = \emptyset$ and $L_0^* = \{\lambda\}$ and $\mathtt{asc}(L_0^*) = 1$. Therefore $g_*^{\mathtt{asc}}(0) = \{1\}$.

If $m \geq 3$ and $r \geq 2$, we consider $L_m = L_X(2, 2r-1, 2r, \ldots 2r+m-4; 2r+m-2, 1)$ with $\mathtt{asc}(L_m) = m$ by Example 1. Then $L_m^* = L_X(0, 2, 4, \ldots, 2r-4; 2r-2, 1)$ (since $X^2 \subseteq L_m$ implies that all words of even length are in L_m^*, and all words of odd length $\leq 2r-1$ are in L_m^* because X^2 and X^{2r-1} are subsets of L_m) and $\mathtt{asc}(L_m^*) = r$ by Example 1.

If $m \geq 3$ and $r = 1$, we consider $L_m = L_X(1, 3, 4, 5, m; m+2, 1)$ with $\mathtt{asc}(L_m) = m$. Then $L_m^* = X^*$ and $\mathtt{asc}(L_m^*) = 1$.

Thus $g_*^{\mathtt{asc}}(m) = \mathbb{N}$ for $m \geq 3$.

If $m = 2$ and $r \geq 3$, we consider $L_2 = L_X(2; 2r-2, 1)$ with $\mathtt{asc}(L_m) = 2$ by Example 1. Then $L_2^* = L_X(0, 2, 4, \dots, 2r-4; 2r-2, 1)$ and $\mathtt{asc}(L_2^*) = r$ by Example 1.

If $m = 2$ and $r = 2$, we consider $L_2 = L_X(0; 2, 1)$ with $\mathtt{asc}(L_m) = 2$ by Example 1. Then $L_2^* = L_2$ and $\mathtt{asc}(L_2^*) = 2$.

If $m = 2$ and $r = 1$, we consider $L_2 = L_X(1; 3, 1)$ with $\mathtt{asc}(L_m) = 2$ by Example 1. Then $L_2^* = X^*$ and $\mathtt{asc}(L_m^*) = 1$.

Thus $g_*^{\mathtt{asc}}(2) = \mathbb{N}$.

If $m = 1$ and $r \geq 2$, we consider $L_1 = L_X(; 2, 2r-3)$ with $\mathtt{asc}(L_1) = 1$ by Example 1. Then $L_1^* = L_X(0, 2, 4, \dots, 2r-4; 2r-2, 1)$ and $\mathtt{asc}(L_1^*) = r$ by Example 1.

If $m = 1$ and $r = 1$, we consider $L_1 = L_X(2)$ with $\mathtt{asc}(L_m) = 1$ by Example 1. Then $L_1^* = L_X(; 0, 2)$ and $\mathtt{asc}(L_1^*) = 1$ by Example 1.

Thus $g_*^{\mathtt{asc}}(1) = \mathbb{N}$. □

By slight modifications of the parameters of the languages used in the proof of Theorem 11, we obtain the following statement.

Theorem 12 *We have*

$$g_+^{\mathtt{asc}}(m) = \begin{cases} \{0\} & \text{for } m = 0 \\ \mathbb{N} & \text{for } m \geq 1 \end{cases}.$$

4. Conclusion

In this paper we started a systematic study of the complexity measure $\mathtt{asc}$ (and $\mathtt{nasc}$) given by the number of accepting states of (non-)deterministic finite automata. We have seen that the measure $\mathtt{nasc}$ is not interesting, because any regular language satisfies $\mathtt{nasc}(L) \leq 2$. But, for the deterministic version, the situation changes completely, and for any natural number $n \geq 0$, there is a regular language L_n such that $\mathtt{asc}(L_n) = n$. Moreover, for deterministic finite automata, we can use the classical minimization theory in order to determine a-minimal automata.

With respect to the behaviour under automata, for the accepting state complexity $\mathtt{asc}$, we presented results, which strongly differ from the results known for the state complexity $\mathtt{sc}$. For complement, union, product, and (positive) Kleene-closure, we have shown that, for $m \geq 1$, $n \geq 1$, and $r \geq 1$, there are languages L_m and L_n with accepting state complexities m and n, respectively, such that $\mathtt{asc}(\alpha(L_m)) = r$ in case of a unary operation $\alpha \in \{C, *\}$ and $\mathtt{asc}(L_m \circ L_n) = r$ in case of a binary operation $\circ \in \{\cup, \cdot\}$.

We mention that such a result does not hold for the intersection. The following statement is well-known: If L_m is accepted by a DFA $\mathcal{A}_m = (X, Z_m, z_{0m}, F_m, \delta_m)$ with $\mathrm{card}(F_m) = m \geq 1$ and L_n is accepted by a DFA $\mathcal{A}_n = (X, Z_n, z_{0n}, F_n, \delta_n)$ with $\mathrm{card}(F_n) = n \geq 1$, then $L_m \cap L_n$ is accepted by the deterministic finite automaton $\mathcal{A} = (X, Z_m \times Z_n, (z_{0m}, z_{0n}), F_m \times F_n, \delta)$ with $\delta((z, z'), a) = (\delta_m(z, a), \delta(z', a))$. Hence, $g_\cap^{\mathtt{asc}}(m, n) \leq m \cdot n$ for $m \geq 1$ and $n \geq 1$. We were able to determine languages L_m and L_n such that $\mathtt{asc}(L_m \cap L_n) = m \cdot n$; and obviously, there are languages L'_m

and L'_n such that $\mathtt{asc}(L'_m \cap L'_n) < m \cdot n$; but we have not completely determined $g_{\cap}^{\mathtt{asc}}(m,n)$. This remains as an open problem as well as the exact determination of $g_{\setminus}^{\mathtt{asc}}(m,n)$ for $m > n \geq 1$ as well as the study of $g_{\circ}^{\mathtt{asc}}$ for further operations $\circ$ (as reversal, proportional removals, etc.).

We mention that, in the case of the measure `sc`, with respect to the behaviour under operations, the situations between finite and arbitrary languages as well as between languages over arbitrary alphabets and over single letter alphabets are different (see [5] and [10]).

Our results show that, for the measure `asc`, there is no difference between arbitrary alphabets and single letter alphabets for the operations complement, union, product, and (positive) Kleene-closure.

Moreover, if we start with finite languages L_m and L_n over a single letter alphabet and $\mathtt{asc}(L_m) = m$ and $\mathtt{asc}(L_n) = n$, then $\mathtt{asc}(L_m \cup L_n) \leq m + n$. Hence we expect that we get different results if we restrict to finite languages, but an investigation of this problem remains to do.

References

[1] J.-M. Champarnaud, F. Coulon, Büchi automata reduction by means of left and right trace inclusion preorder. Manuscript, 2004.

[2] F. Drewes, M. Holzer, S. Jakobi, B. van der Merwe, Tight bounds for cut-operations on deterministic finite automata. In: J. Durand-Lose, B. Nagy (Eds.), *Machines, Computations, Universality*, LNCS 9288, Springer-Verlag, Heidelberg, 2015, 45–60.

[3] St. Edelkamp, Sh. Jabbar, Large scale directed model checking LTL. In: A. Valmer (ed.), *Model Checking Software - SPIN*, LNCS 3925, Springer-Verlag, Berlin, 2006, 1–18.

[4] S. Evangelista, L. M. Kristensen, Hybrid on-the-fly LTL model checking with the sweep-line method. In: S. Haddad, L. Pomello (Eds.), *Applications and Theory of Petri Nets*, LNCS 7347, Springer-Verlag, Berlin, 2012, 248–267.

[5] Y. Gao, N. Moreira, R. Reis, Sh. Yu, A review on state complexity of individual operations. Techn. report series DCC-2011-08, Version 1.1 September 2012, University of Porto, Faculty of Sciences, Department of Computer Science, 2012.

[6] M. Hricko, G. Jirásková, A. Szabari, Union and intersection of regular languages and descriptional complexity. In: C. Mereghetti, B. Palano, G. Pighizzini, D. Wotschke (Eds.), *Proc. 7th Intern. Workshop of Descriptional Complexity of Formal Systems*, University of Milano, 2005, 170–181.

[7] G. Jirásková, On the state complexity of complements, stars, and reversals of regular languages. In: M. Ito, M. Toyama (Eds.), *Developments in Language Theory*, LNCS 5257, Springer-Verlag, Berlin, 2008, 431–442.

[8] D. Kozen, *Automata and Computability.* Springer-Verlag, New York, Berlin, 1997.

[9] G. Rozenberg, A. Salomaa (Eds.), *Handbook of Formal Languages.* Vol. I –III, Springer-Verlag, Berlin, 1997.

[10] Sh. Yu, State complexity of regular languages. *Journal of Automata, Languages and Combinatorics* **6** (2001) 221–234.

(Received: ????? ??, 2016; revised: ?????? ??, ????)