On the descriptional complexity of finite automata with modified acceptance conditions

Markus Holzer* a, Martin Kutrib b

a Institut für Informatik, Technische Universität München, Boltzmannstraße 3, D-85748 Garching bei München, Germany
b Institut für Informatik, Universität Gießen, Arndtstraße 2, D-35392 Gießen, Germany

Received 21 November 2003; received in revised form 13 April 2004; accepted 3 June 2004

Abstract

We consider deterministic and nondeterministic finite automata with acceptance conditions that rely on the whole history of a computation on a given word and not only on the last state of the computation under consideration. Formally, these conditions can be seen as the natural analogies of the Büchi and Muller acceptance for finite automata on infinite words. We study the computational power of these new acceptance mechanisms and prove some results on the descriptional complexity of conversions between automata with these new acceptance criteria and finite automata with ordinary acceptance.

© 2004 Elsevier B.V. All rights reserved.

Keywords: Finite automata; Acceptance conditions; Computational power; Descriptional complexity

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 during the last decade. Tight upper bounds for the state complexity of many operations on regular languages and finite languages are known [3,8,9,19,26,28,29]. As pointed out by Yu [27] there are several

* Corresponding author.
E-mail addresses: holzer@informatik.tu-muenchen.de (M. Holzer), kutrib@informatik.uni-giessen.de (M. Kutrib).

0304-3975/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2004.06.030
good reasons why the number of states of a deterministic finite automata is a natural and objective measure for regular languages. E.g., the number of transitions of a deterministic finite automaton is linear in the number of states of the machine under consideration, but this is not necessarily true for nondeterministic finite automata. Nevertheless, this measure is also the one mostly used for nondeterministic finite automata—see, e.g., [4,8–10]. A state-of-the-art survey for deterministic finite automata was given by Yu [27], and it is worth mentioning that a survey on the descriptional complexity of machines with limited resources was recently written by Goldstine et al. [7].

The study of finite automata on finite and infinite words dates back to the early days of computer science. Nowadays, finite automata on finite words are used in linguistic applications and those on infinite words for the verification of nonterminating programs. While for finite automata on finite words, in principle, only deterministic, nondeterministic, and alternating devices are known, which precisely characterize the regular languages, there is a large variety of models for finite automata on infinite words. To mention only a few of them, there are Büchi [2], Muller [17], Rabin [20], Street [24] automata, etc. For all of these devices, except for Büchi automata, determinism is as powerful as nondeterminism and precisely characterizes the $\omega$-regular languages, an extension of regular languages to infinite words. From the descriptional complexity point of view all these devices are still objects of intense investigations. For example, the conversion problem for different finite automata on infinite words has some interesting open questions. For an overview on state complexity results for finite automata on infinite words we refer to, e.g., [13,18].

In this paper we consider finite automata on finite words. It is natural to consider the impact of determinism, nondeterminism, or alternation on the computational capacity and on the descriptional complexity of the underlying devices. On the other hand, changing the mode of acceptance can have an impact on the computational capacity and on the descriptional complexity, too. Wotschke [25] introduced this idea by considering so-called degree automata. Basically, degree automata are nondeterministic automata that accept an input $w$ if and only if for $w$ the number of reachable accepting states divided by the number of all reachable states is not less than some rational number, the degree. In [11] descriptional complexity issues of degree finite automata are considered. E.g., a class of languages is exhibited for which degree automata are polynomially more concise than equivalent nondeterministic finite automata.

Here we consider modified acceptance conditions, which are inspired by finite automata on infinite words. In particular, we adapt Büchi’s and Muller’s acceptance condition to work on finite automata accepting finite words—we call these acceptance conditions $B$- and $M$-acceptance, respectively. In this way, the acceptance depends on the whole history of the computation. It is quite clear that with these acceptance mechanisms we cannot accept more than the regular languages—in fact, it turns out that some acceptance conditions are not even able to characterize all of the regular languages. On the other hand, the purpose of this paper is to continue the investigations started in [11,25], by considering two different acceptance conditions in order to shed more light on the power of acceptance mechanisms. E.g., for descriptional complexity we show that nondeterministic non-complete or complete devices with $M$-acceptance can be exponentially more concise than equivalent nondeterministic finite automata.
In particular, our first results concern the computational power of $B$- and $M$-acceptance. In particular, we distinguish non-complete and complete automata as well as nondeterministic and deterministic computations. In the world of classical finite automata this distinction is not necessary as far as the general computational power is concerned. All devices accept exactly the regular languages. For the new acceptance conditions this distinctions are necessary, since altogether we obtain four separated language classes for the eight types of automata. It turns out that the $M$-condition yields stronger devices and we show that nondeterministic finite automata $M$-accepting languages characterize the regular languages, regardless whether they are non-complete or complete. For $B$-acceptance one has to address languages modulo the empty word. Under this restriction, nondeterministic automata with $B$-acceptance are also able to accept the regular languages. For $M$-acceptance in the deterministic classes there is no difference between non-complete and complete devices. But determinism is separated from nondeterminism. The situation changes for $B$-acceptance. Deterministic non-complete finite automata with $B$-acceptance are strictly weaker than nondeterministic non-complete automata with $B$-acceptance. They are also strictly weaker than the deterministic devices with $M$-acceptance. Finally, we have the weakest class consisting of nondeterministic and deterministic complete automata with $B$-acceptance.

Further results concern the descriptional complexity of the automata in question. In particular, we consider upper and lower bounds on the number of states when converting from one type of automata to another, if both types have the same computational power. This includes classical nondeterministic finite automata, too. Most of the shown bounds are tight, which means that the bound is sufficient and necessary in the worst case. For the (in some sense) weak automata with $B$-acceptance the bounds are more or less natural. Between nondeterministic devices they are linear, and from nondeterministic to deterministic devices they are of exponential order. For $M$-acceptance we obtain linear bounds between deterministic classes and between nondeterministic classes, respectively. But, surprisingly, there are exponential bounds when converting from nondeterministic non-complete or complete devices with $M$-acceptance to classical nondeterministic finite automata.

The paper is organized as follows: The next section contains the basic definitions. Then in Section 3 we define two new types of acceptance conditions for finite automata and study the computational power of these acceptance criteria. In the penultimate Section 4 we consider the descriptional complexity of finite automata with modified acceptance conditions in more detail. Finally, we summarize our results and highlight some of the remaining open questions in Section 5.

2. Definitions

Let $\Sigma$ be a finite alphabet. Then the cardinality of $\Sigma$ (or any other set) is denoted by $|\Sigma|$ and the set of finite words on $\Sigma$ including the empty word $\lambda$ is denoted by $\Sigma^*$. Here set $\Sigma^*$ is called the free monoid on $\Sigma$. The length of a word $w$ is denoted by $|w|$, where $|\lambda| = 0$. If we consider two languages $L_1$ and $L_2$ to be equal modulo the empty word, we simply write $L_1 \equiv L_2$, and mean $L_1 \setminus \{\lambda\} = L_2 \setminus \{\lambda\}$. Finally, by $\Sigma^\omega$ we denote the set of all infinite words over $\Sigma$. 
A nondeterministic finite automaton (NFA) is a quintuple $A = (Q, \Sigma, \delta, q_0, F)$, where $Q$ is the finite set of states, $\Sigma$ is the finite set of input symbols, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of accepting states, and $\delta : Q \times \Sigma \rightarrow 2^Q \setminus \{\emptyset\}$ is the transition function. Here $2^Q$ denotes the power-set of $Q$. The set of rejecting states is implicitly given by the partitioning, i.e., $Q \setminus F$. A finite automaton is complete, if for all $q \in Q$ and $a \in \Sigma$ we have $|\delta(q, a)| \geq 1$, i.e., the transition function becomes a total function. Finite automata not obeying this condition are called non-complete. Moreover, a finite automaton is deterministic (DFA) if and only if for all $q \in Q$ and $a \in \Sigma$ the set $\delta(q, a)$ is a singleton, i.e., $|\delta(q, a)| = 1$, whenever it is defined. In this case we simply write $\delta(q, a) = p$ instead of $\delta(q, a) = \{p\}$ assuming that transition function is a mapping from $\delta : Q \times \Sigma \rightarrow Q$.

In the sequel, if not stated otherwise, we assume that a finite automaton is always minimal. This means that its number of states is minimal with respect to the accepted language.

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton. A configuration of a finite automaton is a tuple $(q, w)$, where $q \in Q$ and $w$ is a string. A finite automaton $A$ is said to be in configuration $(q, w)$, if $A$ is in state $q$ with remaining input $w$. If $a$ is in $\Sigma$ and $w$ in $\Sigma^*$, then we write $(q, aw) \vdash_A (p, w)$ if $p$ is in $\delta(q, a)$. As usual, the reflexive transitive closure of $\vdash_A$ is denoted by $\vdash^*_A$. The subscript $A$ will be dropped from $\vdash_A$ and $\vdash^*_A$ if the meaning is clear. Then the language accepted by $A$ is defined as

$$L(A) = \{ w \in \Sigma^* \mid (q_0, w) \vdash^*_A (p, \lambda) \text{ for some } p \in F \}.$$

It is well known that deterministic and nondeterministic finite automata on finite words are equally powerful and that they precisely characterize the lowest level of the Chomsky hierarchy, namely the regular languages.

**Theorem 1.** A language $L$ is regular if and only if it is accepted by a deterministic or nondeterministic finite automaton $A$, i.e., $L = L(A)$.

The equivalence of nondeterministic and deterministic finite automata is due to Rabin and Scott [21]. They showed via a subset construction how to convert an $n$-state nondeterministic finite automaton into an equivalent deterministic finite automaton with at most $2^n$ states. Later by Meyer and Fischer [15], and independently by Moore [16] (see also Ershov [5]), it was shown that in general one cannot improve the power-set construction, by giving a sequence of languages $(L_n)_{n \geq 1}$, which are accepted by a nondeterministic finite automata with $n$ states and whose equivalent deterministic finite automata require at least $2^n$ states—see Fig. 1.

![Fig. 1. A nondeterministic finite automaton $A_n$ with $n$ states accepting the language $L_n = L(A_n)$, for which any deterministic finite automaton needs at least $2^n$ states.](image-url)
This can be summarized in the following theorem.

**Theorem 2.** For any integer \( n \geq 1 \) let \( A \) be an \( n \)-state NFA. Then \( 2^n \) states are sufficient and necessary in the worst case for a DFA to accept the language \( L(A) \).

### 3. Finite automata with modified acceptance conditions

In this section we consider finite automata with modified acceptance conditions. Inspired by the variety of acceptance conditions for finite automata on infinite words we introduce two new acceptance conditions for finite automata and study their computational power. Recall that for a Büchi automaton it is required that there is a computation that passes through an accepting state infinitely often. More formally, an infinite word \( w = av \) in \( \Sigma^\omega \) is accepted by a finite automaton \( A = (Q, \Sigma, \delta, q_0, F) \) if and only if there is a computation \( \pi = (q_0, avv') \mapsto_A (q_1, v) \mapsto_A \cdots \) of \( A \) on \( w \) such that \( \text{inf}(\pi) \cap F \neq \emptyset \), where \( \text{inf}(\pi) \) denotes the set of states that appear infinitely often in the computation \( \pi \). Thus, the language of infinite words accepted by \( A \) is defined as

\[
L^\omega(A) = \{ w \in \Sigma^\omega | \text{there is computation } \pi \text{ of } A \text{ on } w \text{ such that } \text{inf}(\pi) \cap F \neq \emptyset \}.
\]

For finite automata on finite words we have to adapt this appropriately, considering the states that are visited during the computation. Thus, we define the language \( B \)-accepted by the automaton \( A \) as

\[
L_B(A) = \{ w \in \Sigma^* | (q_0, w) \mapsto_A \cdots \mapsto_A (q_n, \lambda) \text{ such that } \{q_0, \ldots, q_n\} \cap F \neq \emptyset \},
\]

which is the set of all words \( w \) such that at least one prefix is accepted by \( A \).

For finite automata on infinite words there are many more acceptance conditions, triggered by the fact that deterministic Büchi automata are strictly weaker than nondeterministic ones. The languages accepted by deterministic Büchi automata obey a characterization as a limit set. The limit set of a language \( L \subseteq \Sigma^* \cup \Sigma^\omega \) is defined as

\[
\overrightarrow{L} = \{ w \in \Sigma^\omega | \text{infinitely many prefixes of } w \text{ belong to } L \}.
\]

With this terminology the acceptance power of deterministic Büchi automata can be characterized as follows [12]: A language \( L \subseteq \Sigma^\omega \) is accepted by a deterministic Büchi automaton \( A \), i.e., \( L = L^\omega(A) \), if and only if \( L = \overrightarrow{L}(A) \). Then one can show that the language

\[
L = \{ w \in \{a, b\}^\omega | w \text{ contains a finite number of } b \text{’s} \}
\]

cannot be accepted by any deterministic Büchi automaton, thus separating determinism from nondeterminism.

Nevertheless, it is possible to replace any nondeterministic Büchi automaton by an equivalent deterministic automaton model. To this end, one has to introduce a more powerful acceptance mechanism as, e.g., a Rabin [20], Street [24], or Muller condition. In the forthcoming we focus on the latter condition, which was introduced by Muller [17]. A Muller automaton is a quintuple \( A = (Q, \Sigma, \delta, q_0, F) \), where \( Q, \Sigma, \delta, \) and \( q_0 \) are as in the case of
finite automata and $F \subseteq 2^Q$. Observe, that $F$ is no longer a subset of $Q$, but rather a subset of $2^Q$. Then the language of infinite words accepted by $A$ is

$$L^\omega(A) = \{ w \in \Sigma^\omega | \text{there is a computation } \pi \text{ of } A \text{ on } w \text{ such that } \inf(\pi) \in F \}. $$

For Muller automata the situation is completely different compared to Büchi automata, since determinism and nondeterminism coincides and precisely characterizes the $\omega$-regular languages, which was shown by McNaughton [14].

For ordinary finite automata on finite words we adapt the Muller condition as follows. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton with $F \subseteq 2^Q$. Then the language $M$-accepted by $A$ is defined as

$$L_M(A) = \{ w \in \Sigma^* | (q_0, w) \mathrel{\vdash_A} \ldots \mathrel{\vdash_A}(q_n, \lambda) \text{ such that } \{q_0, \ldots, q_n\} \in F \},$$

which is the set of all words such that the set of all states that have been seen during the computation are a member of $F \subseteq 2^Q$. Clearly, given a finite automaton $A$, in general, the languages $L(A), L_B(A),$ and $L_M(A)$ are different.

It remains to determine the computational power of finite automata that $B$- or $M$-accept languages, which is done in the following two subsections, starting with $B$-acceptance. Observe, that for both acceptance conditions it will be important whether the underlying device is complete or non-complete. Moreover, we distinguish deterministic and nondeterministic computations.

### 3.1. B-Acceptance

First we show that nearly every regular language, except those languages which contain the empty word $\lambda$, are $B$-accepted by a nondeterministic non-complete finite automaton.

**Theorem 3.** A language $L$ is $B$-accepted by a nondeterministic non-complete finite automaton, if $L$ is regular and does not contain the empty word.

**Proof.** Let $A = (Q, \Sigma, \delta, q_0, F)$ be a (nondeterministic) finite automaton accepting $L$, i.e., $L = L(A)$. Observe, that since $\lambda$ is not a member of $L$, the initial state $q_0$ is not contained in $F$. We construct a non-complete nondeterministic finite automaton $C$ that $B$-accepts $L$. Define $C = (Q', \Sigma, \delta', q_0, F')$, where the union $Q' = Q \cup \{q_f\}$ is disjoint, $F' = \{q_f\}$, and the transition function $\delta'$ is specified as follows: (1) For all $q \in Q$ and $a \in \Sigma$, let $\delta'(q, a)$ include all elements of $\delta(q, a)$, and (2) or all $q \in Q$ and $a \in \Sigma$, if $\delta(q, a) \cap F \neq \emptyset$, then $\delta'(q, a)$ contains $q_f$.

Transitions from (1) cause $C$ to simulate $A$ step-by-step. If $A$ accepts the input by taking a transition to a final state, then $C$ takes the transition specified in (2) and moves to the sole final state $q_f$. Then it is obvious to see that $w \in L(A)$ if and only if $w \in L_B(C)$. Since any computation continuing in the final state $q_f$ must be blocked, the constructed automaton is non-complete. Thus, the stated claim follows. $\square$

For the converse relation we find the following situation.
Theorem 4. If the language \( L \) is \( B \)-accepted by a deterministic or nondeterministic finite automaton, regardless whether the automaton is complete or non-complete, then \( L \) is regular.

Proof. Let \( C = (Q, \Sigma, \delta, q_0, F) \) be a finite automaton that \( B \)-accepts language \( L \). In other words, \( L = L_B(C) \). We define an ordinary nondeterministic finite automaton \( A = (Q \times \{1, 2\}, \Sigma, \delta', q_0', F') \), where \( q_0' \) equals \((q_0, 2)\), if \( q_0 \) is in \( F \), and \( q_0' \) is \((q_0, 1)\), otherwise, \( F' = \{(q, 2) \mid q \in Q\} \), and \( \delta' \) is given as follows: (1) For all \( p, q \in Q \), \( a \in \Sigma \), and \( 1 \leq i \leq 2 \), let \( \delta'((q, i), a) \) contain \((p, i)\), if \( p \in \delta(q, a) \), and (2) for all \( q \in Q \), \( p \in F \), and \( a \in \Sigma \), set \( \delta'((q, 1), a) \) contains \((p, 2)\), if \( p \in \delta(q, a) \).

Observe, that each state in \( A \) is a pair, where the second component indicates whether the original automaton has already seen a final state or not. The transitions from (1) cause \( A \) to simulate \( C \), and whenever \( C \) takes a transition to a final state, then \( A \) has either the possibility to continue the simulation not changing the second component of the state or changing the second component from 1 to 2, thus moving to a final state in \( A \). With this observation it is easy to see that \( w \in L_B(C) \) if and only if \( w \in L(A) \). Therefore, \( L_B(C) \) is regular. \( \square \)

The following corollary is an easy consequence of Theorems 3 and 4. Therefore, we omit its proof.

Corollary 5. Let \( L \) be a language not containing the empty word. Then \( L \) is regular if and only if it is \( B \)-accepted by a nondeterministic non-complete finite automaton.

Concerning the computational power of nondeterministic or deterministic complete finite automata \( B \)-accepting languages, we find a nice characterization of these languages. Observe, that the theorem given below nicely parallels the results on deterministic Büchi automata mentioned at the beginning of this section.

Theorem 6. A language \( L \subseteq \Sigma^* \) is \( B \)-accepted by a nondeterministic or deterministic complete finite automaton \( A \) if and only if
\[
L = \{ w \in \Sigma^* \mid \text{at least one prefix of } w \text{ is in } L(A) \}.
\]
In other words, there is a regular language \( X \subseteq \Sigma^* \) such that \( L = X \Sigma^* \).

Proof. Whenever a complete automaton \( A = (Q, \Sigma, \delta, q_0, F) \) has reached a final state in \( F \), the remaining input is no longer relevant for \( B \)-acceptance anymore, i.e., it is accepted regardless of the remaining input, which can be completely read without blocking the computation of the automaton. Thus, the language \( L \) obeys the following characterization \( L = \left( \bigcup_{q \in F} L(A_{q_0, q}) \right) \cdot \Sigma^* \), where \( A_{q,p} \) is the finite automaton \( A_{q,p} = (Q, \Sigma, \delta, q, \{p\}) \).
Since \( F \) is finite, the set \( \bigcup_{q \in F} L(A_{q_0, q}) \) is regular, and proves the implication from left to right. Conversely, let \( X \) be accepted by the deterministic finite automaton \( A = (Q, \Sigma, \delta, q_0, F) \), i.e. \( X = L(A) \). Then define the deterministic complete finite automaton \( C = (Q', \Sigma, \delta', q_0, F') \), where the union \( Q' = Q \cup \{f\} \) is disjoint, \( F' = F \cup \{q_f\} \), and the
transition function obeys (1) \( \delta'(q, a) = \delta(q, a) \) for all \( q \in Q \setminus F \) and \( a \in \Sigma \), (2) for all \( q \in F \) and \( a \in \Sigma \) let \( \delta(q, a) = q_f \), and (3) \( \delta(q_f, a) = q_f \) for all \( a \in \Sigma \). Then the language \( X\Sigma^* \) is \( B \)-accepted by the automaton \( C \). Since determinism is a restriction of nondeterminism the stated claim follows. \( \square \)

It remains to clarify the inclusion relations between the above-studied language families and the family of all languages \( B \)-accepted by deterministic non-complete finite automata. Obviously, by definition the latter family is a superset of the family of languages \( B \)-accepted by deterministic complete finite automata, and by Theorem 4 a subset of the family of regular languages. The following theorem shows that both inclusions are strict.

**Theorem 7.** There exists a regular language which cannot be \( B \)-accepted by any deterministic non-complete finite automaton. Moreover, there is a language \( B \)-accepted by a deterministic non-complete finite automaton but not by any (non)deterministic complete finite automaton.

**Proof.** For the first strict inclusion consider the regular language \( L = \{a, b\}^*a \) over alphabet \( \Sigma = \{a, b\} \). Now assume to the contrary that \( L \) is \( B \)-accepted by a deterministic non-complete finite automaton \( A = (Q, \Sigma, \delta, q_0, F) \). Since \( \lambda \) is not in \( L \), we deduce that the initial state \( q_0 \) is not a final state. Now consider the word \( a \) that belongs to \( L \). Thus, state \( q_1 = \delta(q_0, a) \) is defined and an accepting state. If we continue the computation with letter \( b \), we distinguish two cases: (1) Either \( \delta(q_1, b) \) is not defined, but then the word \( aba \) cannot be accepted anymore, or (2) \( \delta(q_1, b) = q_2 \), which implies that \( ab \) is accepted since we have seen a final state during the computation. Therefore, in both cases we obtain a contradiction. Thus, language \( L \) cannot be \( B \)-accepted by any deterministic non-complete finite automaton.

For the second strict inclusion consider the finite language \( L = \{a\} \) over alphabet \( \Sigma = \{a\} \). The deterministic non-complete finite automaton \( A = (\{q_0, q_1\}, \Sigma, \delta, q_0, \{q_1\}) \) with \( \delta(q_0, a) = q_1 \) \( B \)-accepts \( L \), i.e., \( L = L_B(A) \). Now assume to the contrary that \( L \) is \( B \)-accepted by a (non)deterministic complete finite automaton. Then, by Theorem 6, language \( L \) can be written as \( X\Sigma^* \), for some regular language \( X \). Depending on \( X \), the set \( X\Sigma^* \) is either empty, if \( X = \emptyset \), or infinite, if \( X \neq \emptyset \). This contradicts our assumption, since \( L \) is a finite non-empty language, and therefore it cannot be \( B \)-accepted by any (non)deterministic complete finite automaton. \( \square \)

### 3.2. \( M \)-Acceptance

We continue our consideration with \( M \)-acceptance, showing that \( B \)-acceptance can be simulated, which is similar for Büchi and Muller automata.

**Theorem 8.** Every language which is \( B \)-accepted by a nondeterministic (deterministic, respectively) non-complete finite automaton is also \( M \)-accepted by a nondeterministic (deterministic, respectively) non-complete finite automaton. The statement remains valid if complete finite automata are considered.
Theorem 9. A language is regular if and only if it is $M$-accepted by a nondeterministic complete finite automaton.

Proof. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a nondeterministic non-complete finite automaton. We define the non-complete finite automaton $B = (Q, \Sigma, \delta, q_0, F')$, where $F' = \{ M \in 2^Q \mid M \cap F \neq \emptyset \}$. By an induction on the length of $w$ one can see that $w \in L_B(A)$ if and only if $w \in L_M(B)$. Observe, that $B$ is deterministic, if $A$ is deterministic. □

As expected, the theorem given below shows that $M$-acceptance on a nondeterministic finite state device is as powerful as ordinary acceptance on finite automata, regardless whether the device is complete or non-complete.

Theorem 10. A language is regular if and only if it is $M$-accepted by a nondeterministic complete or non-complete finite automaton.

Proof. For the direction from left to right we use a similar construction as in the proof of Theorem 3, modifying the set of final states accordingly. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a nondeterministic complete finite automaton accepting $L$. Define the nondeterministic complete finite automaton $B = (Q', \Sigma, \delta', q_0, F')$, where the union $Q' = Q \cup\{q_f, q_r\}$ is disjoint,

$$F' = \{ M \in 2^{Q' \cup\{q_f\}} \mid M \cap \{q_f\} \neq \emptyset \} \cup (F \cap \{q_0\})$$

and the transition function $\delta'$ is specified as follows: (1) For all $q \in Q$ and $a \in \Sigma$, let $\delta'(q, a)$ include all elements of $\delta(q, a)$, (2) for all $q \in Q$ and $a \in \Sigma$, if $\delta(q, a) \cap F \neq \emptyset$, then $\delta'(q, a)$ contains $q_f$, and (3) for all $a \in \Sigma, q_r$ is contained in both $\delta(q_f, a)$ and $\delta(q_r, a)$.

Then by induction one can verify that $w \in L(A)$ if and only if $w \in L_M(B)$. This shows that if $L$ is regular, then language $L$ is $M$-accepted by a nondeterministic complete—hence also non-complete—finite automaton.

The direction from right to left is seen as follows. We can restrict ourselves to start with a nondeterministic non-complete finite automaton $B = (Q, \Sigma, \delta, q_0, F)$ that $M$-accepts $L$. Recall, that $F \subseteq 2^\Sigma$. We construct an ordinary nondeterministic finite automaton $A = (Q \times 2^Q, \Sigma, \delta', (q_0, \{q_0\}), F')$, where $F' = \{(q, M) \in Q \times 2^Q \mid M \in F\}$ and $\delta'$ is defined as follows: For all $q, p \in Q, M \in 2^Q$, and $a \in \Sigma$, let $\delta'((q, M), a)$ contain $(p, M \cup\{p\})$, if $p \in \delta(q, a)$. It is easy to see that $w \in L_M(B)$ if and only if $w \in L(A)$. Thus, the stated claim on $M$-acceptance follows. □

For deterministic $M$-acceptance we first show in the next theorem that completeness is not an issue for computational power.

Theorem 10. A language is $M$-accepted by a deterministic complete finite automaton if and only if it is $M$-accepted by a deterministic non-complete finite automaton.

Proof. The implication from left to right is obvious by definition. So let $B = (Q, \Sigma, \delta, q_0, F)$ be a deterministic non-complete finite automaton. Define $A = (Q', \Sigma, \delta', q_0, F)$, where the union $Q' = Q \cup\{q_r\}$ is disjoint, and the transition function $\delta'$ is specified as follows: (1) For all $q \in Q$ and $a \in \Sigma$, let $\delta'(q, a) = \delta(q, a)$, (2) for all $q \in Q$ and $a \in \Sigma$, whenever $\delta(q, a)$ is undefined, then let $\delta'(q, a) = q_r$, and (3) for all $a \in \Sigma$, define $\delta'(q_r, a) = q_r$. 


It is obvious, that \( w \in L_M(B) \) if and only if \( w \in L_M(A) \). Thus, the stated claim on deterministic \( M \)-acceptance follows. \( \square \)

It remains to clarify the relation between the family of languages \( M \)-accepted by deterministic (complete or non-complete) finite automata and the language families considered so far. By definition and Theorems 8 and 9, the family under consideration is sandwiched in between the family of languages which are \( B \)-accepted by deterministic non-complete finite automata and the family of regular languages. The next theorem shows that both inclusions are strict.

**Theorem 11.** There exists a regular language which cannot be \( M \)-accepted by any deterministic complete or non-complete finite automaton. Moreover, there is a language which is \( M \)-accepted by a deterministic complete finite automaton but not \( B \)-accepted by any deterministic non-complete finite automaton.

**Proof.** Consider the infinite unary regular language \( L = \{a^n\}^+ \), for some \( n \geq 2 \). By Theorem 10 it suffices to show that \( L \) cannot be \( M \)-accepted by any deterministic complete finite automaton. Assume to the contrary that \( L \) is \( M \)-accepted by a deterministic complete finite automaton \( A \) with state set \( Q \). Observe, that the transition graph of \( A \) with unary input alphabet consists of a path, which starts from the initial state, followed by a cycle of one or more states. Thus, there is a word \( a^{i \cdot n} \in L \) for some \( i \geq 1 \), such that all states of \( A \) are reached and, therefore, \( Q \in F \) must hold. But then \( a^{i \cdot n}a \) is also \( M \)-accepted by \( A \), which is a contradiction since \( a^{i \cdot n+1} \) is not a member of \( L \). Hence, language \( L \) cannot be \( M \)-accepted by any deterministic (complete or non-complete) finite automaton.

For the second strict inclusion consider the finite language \( L = \{a, a^3\} \) over alphabet \( \Sigma = \{a\} \). It is an easy exercise to show that \( L \) is \( M \)-accepted by a deterministic (complete or non-complete) finite automaton. For the sake of contradiction we assume that \( L \) is \( B \)-accepted by a deterministic non-complete finite automaton. Since \( \lambda \) is not in \( L \), the initial state \( q_0 \) does not belong to the set of final states. Since the computations on both words \( a \) and \( a^2 \) must be accepting and thus are non-blocking, we deduce that also the computation on word \( a^2 \) is non-blocking. Here a computation on a word \( w \) is said to be blocking if and only if \( \delta(q_0, w) \) is undefined, where \( q_0 \) is the initial state. Because the underlying machine is deterministic and \( a \) is prefix of \( a^2 \), the latter word must be in \( L \), too. This contradicts our assumption. \( \square \)

Observe, that every finite language is \( M \)-accepted by a deterministic (complete or non-complete) finite automaton. We leave the proof to the interested reader.

We summarize our results on languages accepted by finite automata with modified acceptance conditions in Fig. 2, where the strict inclusion relations are depicted by arrows. Here \( \mathcal{L}(\text{DBA}_c) \) (\( \mathcal{L}(\text{DBA}_nc) \), respectively) denotes the family of languages \( B \)-accepted by deterministic complete (non-complete, respectively) finite automata. Replacing letter \( D \) by \( N \) refers to the nondeterministic counterpart and changing the letter \( B \) to \( M \) refers to the family of languages \( M \)-accepted. Finally, let \( \mathcal{L}(\text{FIN}) \) refer to the family of finite languages and \( \mathcal{L}(\text{REG}) \) to the family of regular languages.
4. Descriptional complexity of automata with modified acceptance conditions

In this section we study the descriptional complexity of finite automata with modified acceptance conditions (compared to ordinary finite automata). First let us recall what is known for Büchi and Muller automata on infinite words. The conversion of a nondeterministic Büchi automaton to an equivalent deterministic Muller automaton is quite costly and needs at most $2^{O(n \log n)}$ states. This can be shown by a modified power-set construction, which is due to Safra [22]. The idea of the extension of the power-set construction is to branch a new computation path every time the given automaton reaches a final state. Therefore, the states are not just sets anymore, but finite ordered trees. Moreover, the conversion of a nondeterministic Muller into its equivalent deterministic counterpart is even worse. There is a double exponential upper bound of $2^{2^{O(n \log n)}}$, combining the conversion of a Muller automaton into an equivalent Büchi automaton—see, e.g., Safra and Vardi [23]. Whether the above-mentioned upper bounds are optimal is an open problem.

Now let us draw our attention to finite automata accepting finite words. We distinguish $B$- and $M$-acceptance starting with the first.

4.1. $B$-Acceptance

The next theorem shows that the construction given in Theorems 3 and 9 to convert an ordinary nondeterministic finite automaton into a finite state device that $B$- or $M$-accepts the same language is optimal with respect to the number of states. Recall, that an $(n + 1)$-state finite automaton was constructed from a given $n$-state device.

**Theorem 12.** For any integer $n \geq 2$ let $A$ be an $n$-state NFA, such that $\lambda \notin L(A)$. Then $n + 1$ states are sufficient and necessary in the worst case for an NBA$_{nc}$ to $B$-accept the language $L(A)$.

**Proof.** The upper bound $n + 1$ on the number of states needed by an NBA$_{nc}$ to $B$-accept any regular language not containing the empty word follows from Theorem 3. Now consider
the regular language \( L = a\{a^n\}^* \). One can show by simple pumping arguments that every NFA \( A \) accepting \( L \) needs at least \( n + 1 \) states.

First we argue that any NBA\(_{nc}\) that \( B \)-accepts the language \( L \) satisfies that all (reachable) accepting states are dead-ends—here a state \( q \) is called a dead-end if \( \delta(q, a) \) is undefined, for all \( a \in \Sigma \). Now let \( C = (Q, \{a\}, \delta, q_0, F) \) be some automaton that \( B \)-accepts language \( L \). Assume to the contrary that the above property is not satisfied. Then there is a reachable state \( q \in F \) or any integers \( m \in \mathbb{N} \).

**Theorem 13.** Let \( C = (Q, \{a\}, \delta, q_0, F) \). Assume that \( C(q, a) \) either \( \delta(q, a) \) is undefined, for all \( a \in \Sigma \). Consider a computation in the state \( qf \) which is not a member of \( F \). Then by the pigeon hole principle there is a state \( q_{m+1} \in Q \setminus F \) with \( i \in \mathbb{N} \). Since \( C(q, a) \) is always defined, the computation can loop. More formally, we find a computation \( C(q_0, a)\delta^r C(q, a)\delta^r \cdots \delta^r C(q, a)\delta^r C(q_{m+1}, \lambda) = (q_{m+1}, \lambda) \) where \( k \) is an integer and \( p \) is some state in \( Q \). The loop from \( q_0 \) to \( q_{m+1} \) is of length \( 1 \leq k \leq n - 1 \).

**Proof.** Basically, the idea for the construction of a \((2m + k)\)-state NBA\(_{nc}\) is to use two copies of \( C \). The computation starts in the first copy, whose final states are deleted. Each transition to a final state of the first copy is redirected to the matching state of the second copy. In addition, all states of the second copy are made final.

Let \( C = (Q, \Sigma, \delta, q_0, F) \), then \( A = (Q', \Sigma, \delta', q_0', F') \) is formally constructed as follows: Assume \( Q = \{p_1, \ldots, p_{m+k}\} \) and \( F = \{p_{m+1}, \ldots, p_{m+k}\} \). Then \( Q' = \{p_1, \ldots, p_m, \tilde{p}_1, \ldots, \tilde{p}_{m+k}\} \), \( F' = \{\tilde{p}_1, \ldots, \tilde{p}_{m+k}\} \), if \( q_0 \notin F \), then \( \tilde{q}_0 = q_0 \) else \( \tilde{q}_0 = q_0 \), and \( \delta'(s, a) = \{\tilde{r} \mid r \in \delta(s, a) \} \) and \( \delta'(s, a) = \{\tilde{r} \mid r \in \delta(s, a) \cap F \} \). Clearly, automaton \( A \) accepts the language \( L_B(C) \), i.e., \( L(A) = L_B(C) \), with \( 2m + k \) states.

In order to show that \( 2m + k \) states are necessary in the worst case, define languages \( L_{k, m} \) over the alphabet \{a, b, c\} dependent on the integers \( m \geq 2 \) and \( m - 1 \geq k \geq 1 \) as follows.
Fig. 3. A nondeterministic non-complete finite automaton that $B$-accepts language $L_{k,m}$ with $m+k$ states.

Set

$$X = \{ x_1 \ldots x_m \mid x_1, \ldots, x_k \in \{ a, ba \} \text{ and } x_{k+1}, \ldots, x_{m-1} = a \text{ and } x_m = c \}$$

and define

$$L_{k,m} = \{ vw \mid v \in X^* \text{ and } w \text{ is a non-empty prefix of some word in } X \text{ and } vw \text{ includes at least one letter } b \}.$$  

The NBA$_{nc}$ depicted in Fig. 3 $B$-accepts the language $L_{k,m}$ with $k+m$ states.

Now let $A = (Q, \{a, b, c\}, \delta, q_0, F)$ be an NFA accepting $L_{k,m}$. We show that $A$ needs at least $2m+k$ states. To this end, consider the inputs $u_1 = a^{m-1}cb$, $u_2 = ba^{m-1}c$, and $u_3 = (ba)^ka^{m-1}c$, which belong to $L_{k,m}$, respectively.

During an accepting computation on $u_1$, automaton $A$ passes through the states $q_0, q_1, \ldots, q_{m-1}, q_m, q_{m+1}$. The states $q_0, q_1, \ldots, q_{m-1}$ have to be non-accepting since neither the empty word nor one of the prefixes $a, a^2, \ldots, a^{m-1}$ do belong to $L_{k,m}$. If two of the states $q_0, q_1, \ldots, q_{m-1}$ were identical, say $q_i = q_j$, for $0 \leq i < j \leq m-1$, then the input $a^{m-1-j}cb$ would be accepted, though it does not belong to $L_{k,m}$. Thus, the $m$ non-accepting states $q_0, q_1, \ldots, q_{m-1}$ are different.

When accepting $u_2$, automaton $A$ passes through the states $q_0, r_1, \ldots, r_{m+1}$. The states $r_1, \ldots, r_m$ are different. Otherwise, if $r_i = r_j$, for $1 \leq i < j \leq m$, then the input $ba^{m-1-j}c$ would be accepted.

During an accepting computation on $u_3$, automaton $A$ passes through the states $q_0, s_1, \ldots, s_{2k}, s_{2k+1}, \ldots, s_{m-1+k}, s_{m+k}$. Similarly as before it can be seen that all states $s_1, \ldots, s_{m+k}$ are different.

Now we turn to show that at least $2m+k$ of the states $q_0, \ldots, q_{m-1}, r_1, \ldots, r_m, s_1, \ldots, s_{m+k}$ are different. We start with the $m+k$ different states $s_1, \ldots, s_{m+k}$. We compare these states with the $m$ different states $q_0, \ldots, q_{m-1}$. If none of the states $q_i$ appears in the sequence $s_1, \ldots, s_{m+k}$, we are done. If, otherwise some state $q_i$ appears as some $s_j$, then we observe that $j$ must be $2i$ or $2i+1$, and $i$ has to be smaller than $k$, $i < k$. The observation is due to the fact that after entering state $q_i$, automaton $A$ has to read exactly furthermore $m-1-i$ symbols $a$ until a symbol $c$ may appear in the input. In order to reject inputs not of the appropriate form, automaton $A$ has to be in corresponding states $s_j$. Moreover, if $i \geq k$, then the corresponding state is $s_{2k+i-1}$. But this implies an accepting computation on $a^{m-1}c$ through the sequence of states $q_0, \ldots, q_{i-1}, s_{2k+i-1}, \ldots, s_{m+k-1}, s_{m+k}$, a contradiction.
Now we compare a state $q_i$, $i < k$, with the states $s_{2i}$ and $s_{2i+1}$. Clearly, $q_0$ does not appear in the sequence $s_1, \ldots, s_{m+k}$. So let $i > 0$ in addition. Since $s_{2i}$ and $s_{2i+1}$ are different, $q_i$ matches at most one of both. Without loss of generality, say $s_{2i}$. If this happens, consider the sequence of different states $r_1, \ldots, r_m$. For the same reasons as before, $s_{2i}$ can never match $q_i$ and $r_{i+1}$ at the same time, since they are different. If they would not be different, then the input $a^{m-1}c$ would be accepted through the sequence of states $q_0, \ldots, q_{i-1}, r_{i+1}, \ldots, r_m, r_{m+1}$.

Altogether, we conclude $m + k$ different states $s_1, \ldots, s_{m+k}$ and $m$ different states $p_0, \ldots, p_{m-1}$, where $p_i$ is either $q_i$ or $r_{i+1}$, that do not appear in the sequence $s_1, \ldots, s_{m+k}$. Hence, automaton $A$ has at least $2m + k$ states. □

Finally, in the remainder of this subsection we consider the relation between complete finite automata that $B$-accept languages in more detail. From nondeterminism to determinism we show that exponential state saving is possible.

**Theorem 14.** Let $n \geq 3$ be an integer. (1) Let $A$ be an $n$-state NBA$_C$. Then $2^n - 1 + 1$ states are sufficient for a DBA$_C$ to accept the language $L_B(A)$. (2) There exists an $n$-state NBA$_C$ $A$, such that any DBA$_C$ accepting the language $L_B(A)$ needs at least $2^{n-2} + 1$ states.

**Proof.** The upper bound easily follows from the power-set construction and from the observation, that for complete automata whenever a final state is reached the remaining input is not relevant for $B$-acceptance anymore, i.e., the input is accepted regardless of the remaining input. The latter fact implies that the automaton needs only one sole accepting state, which loops for every letter of the alphabet. Thus, one can modify the power-set construction, such that only states are constructed that do not contain a final state, and acceptance is coordinated by an extra accepting final state. Assuming that the NBA$_C$ $A$ has at least one final state $2^n - 1 + 1$ states are sufficient for a DBA$_C$ to accept the language $L_B(A)$.

Next we prove the lower bound. For $k \geq 0$ let $L_k = \{a, b\}^*a(a, b)^kb(a, b)^*$. It is clear that $L_k$ is accepted by the $(k + 3)$ state NBA$_C$, which is depicted in Fig. 4. Intuitively, $A$ has to guess the position of an input symbol $a$, which is followed by $k$ arbitrary input symbols, and then by a $b$.

In order to $B$-accept language $L_k$, a DBA$_C$ $C = (Q, \Sigma, \delta, q_0, F)$ has to verify that the input has a substring $a(a, b)^kb$. Therefore, after reading a symbol $a$, the deterministic finite automaton must be able to remember the next $k$ input symbols. Altogether this needs $2^{k+1}$ states and a sole accepting state.
More formally, we consider inputs of length $k+1$. Let $q_v = \delta(q_0, v)$, for $v \in \{a, b\}^{k+1}$. Observe, that every $q_v$ must be a non-accepting state. Assume $q_v = q_{v'}$ for $v \neq v'$ with $v, v' \in \{a, b\}^{k+1}$. Then without loss of generality $v = uaw$ and $v' = ubw'$, for $u, w, w' \in \Sigma^*$. Observe, that both $w$ and $w'$ are of the same length. But then the word $va^{[l]}b = uawa^{[l]}b$ is accepted since it contains the pattern $a[a, b]^k b$. To be more precisely, we find $\delta(q_v, va^{[l]}b) = \delta(q_0, a^{[l]}b) \in F$. On the other hand, then $v'a^{[l]}b$ is accepted, too, since $\delta(q_{v'}, a^{[l]}b) = \delta(q_v, a^{[l]}b)$, but it does not contain the pattern we are looking for. This is a contradiction. Therefore, $q_v \neq q_{v'}$, if $v \neq v'$ with $v, v' \in \{a, b\}^{k+1}$. Since there are $2^{k+1}$ words of length $k+1$, we must have at least $2^{k+1}$ different non-accepting states. Thus, $B$ has at least $2^{k+1} + 1$ states. The assertion follows by substituting $n = k + 3$. □

The next theorem shows the converse conversion.

**Theorem 15.** *For any integer $n \geq 1$ let $A$ be an $n$-state DBA$_c$. Then $n$ states are sufficient and necessary in the worst case for an NBA$_c$ to accept the language $L_B(A)$.*

**Proof.** The upper bound is immediate since determinism is a restriction of nondeterminism. In order to show the lower bound, we use the language $L_n = \{a^{n-1}\}a^n$ over the alphabet $\Sigma = \{a\}$. Clearly, a deterministic finite automaton needs $n$ states to $B$-accept the language $L_n$. By a simple counting argument one can show that even a nondeterministic finite automaton needs at least $n$ states to $B$-accept language $L_n$. □

We summarize our results on $B$-acceptance in Table 1.

### 4.2. $M$-Acceptance

We continue with the descriptional complexity of finite automata $M$-accepting languages. The first theorem shows that going from an NFA to a non-complete or complete NBA only increases the number of states by one or two in the worst case.

**Theorem 16.** *For any integer $n \geq 2$ let $A$ be an $n$-state NFA. Then $n + 1$ ($n + 2$, respectively) states are sufficient and necessary in the worst case for an NBA$_{nc}$ (NBA$_c$, respectively) to accept the language $L(A)$.*
Theorem 17. Let $L_k$ be a language over the alphabet $\{a, b\}$. Then $L_k$ accepts the language $L$ if and only if (1) $0 \leq i < k$, for all $1 \leq i \leq n$, and (2) for all $1 \leq i \leq n$, we have either $i \in M$ iff $i + n \in M$ or $i' \in M$ iff $(i + n)' \in M$. By the definition of the accepting subsets it is easy to see that $A$ accepts the language $L_k$. The NMA $A$ is depicted in Fig. 5. Observe, that a non-complete finite automaton does not need the sink state $2k + 1$. Therefore, a NMA accepting $L_k$ has only $4k + 1$ states.

In order to show that an NFA needs an exponential number of states, we use the fooling set technique [1,6]. A set of pairs of strings $P = \{(x_i, y_j) \mid 1 \leq i \leq n\}$ is a fooling set for a language $L$, if (1) $x_i y_j \in L$, for $1 \leq i \leq n$, and (2) if $i \neq j$ with $1 \leq i, j \leq n$, then $x_i y_j \notin L$. Whenever a language $L$ has a fooling set $P$, then any NFA accepting $L$ needs at least $|P|$ states. The language $L_k$ has a fooling set

$$\{(u, a^{|u|}u a^{|u|}) \mid u \in \{a, b\}^i \text{ with } 0 \leq i < k\}$$

$$\cup \{(a^{|u|}u a^{|u|}, u) \mid u \in \{a, b\}^i \text{ with } 0 \leq i < k\}$$

$$\cup \{(u, u) \mid u \in \{a, b\}^k\}.$$
Fig. 5. Nondeterministic complete finite automaton that $M$-accepts language $L_k$.

Since there are $\sum_{i=0}^{k-1} 2^i + 2^k + \sum_{i=0}^{k-1} 2^i = 3 \cdot 2^k - 2$ different pairs of strings, any NFA accepting $L_k$ needs at least $3 \cdot 2^k - 2$ states. Thus, the stated claim follows by substituting $n = 4k + 2$ respectively $n = 4k + 1$. □

Finally, we show how complete and non-complete NMA are related to each other with respect to the number of states.

**Theorem 18.** Let $n \geq 2$ be an integer. (1) Let $A$ be an $n$-state $\text{NMA}_{nc}$. Then $n + 1$ states are sufficient and necessary in the worst case for a $\text{NMA}_c$ to accept the language $L_M(A)$. (2) Let $A$ be an $n$-state $\text{NMA}_c$. Then $n$ states are sufficient and necessary in the worst case for a $\text{NMA}_{nc}$ to accept the language $L_M(A)$.

**Proof.** The upper bounds are immediate in both cases. The lower bounds rely on simple counting arguments. In the first case one can use the language $L_n = \{a^{n-1}b\}$ over the alphabet $\{a, b\}$, and for the second case, the unary language $L_n = \{a^{n-1}\}$ may serve as a witness. The tedious details are left to the reader. □

The proof of the following theorem is similar to the proof of Theorem 18, and is therefore omitted.

**Theorem 19.** Let $n \geq 2$ be an integer. (1) Let $A$ be an $n$-state $\text{DMA}_{nc}$. Then $n + 1$ states are sufficient and necessary in the worst case for a $\text{DMA}_c$ to accept the language $L_M(A)$. (2) Let $A$ be an $n$-state $\text{DMA}_c$. Then $n$ states are sufficient and necessary in the worst case for a $\text{DMA}_{nc}$ to accept the language $L_M(A)$. □

We summarize our results on $M$-acceptance in Table 2.

5. Conclusions

We have investigated deterministic and nondeterministic finite automata with modified acceptance conditions, which were inspired by the Büchi and Muller automata for accepting languages on infinite words. We have compared the computational power of $B$- and
Table 2

<table>
<thead>
<tr>
<th>M-acceptance</th>
<th>NFA</th>
<th>NMA&lt;sub&gt;nc&lt;/sub&gt;</th>
<th>NMA&lt;sub&gt;c&lt;/sub&gt;</th>
<th>DMA&lt;sub&gt;nc&lt;/sub&gt;</th>
<th>DMA&lt;sub&gt;c&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>From … to …</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NFA</td>
<td>trivial</td>
<td>$n + 1$</td>
<td>$n + 2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NMA&lt;sub&gt;nc&lt;/sub&gt;</td>
<td>$\leq n \cdot 2^{n-2}$</td>
<td></td>
<td></td>
<td>$n + 1$</td>
<td></td>
</tr>
<tr>
<td>NMA&lt;sub&gt;c&lt;/sub&gt;</td>
<td>$\geq 3 \cdot 2^{\lfloor \frac{n-1}{4} \rfloor - 2}$</td>
<td>trivial</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMA&lt;sub&gt;nc&lt;/sub&gt;</td>
<td></td>
<td>$\leq n \cdot 2^{n-2}$</td>
<td>$n$</td>
<td>trivial</td>
<td></td>
</tr>
<tr>
<td>DMA&lt;sub&gt;c&lt;/sub&gt;</td>
<td></td>
<td>$\geq 3 \cdot 2^{\lfloor \frac{n-2}{4} \rfloor - 2}$</td>
<td></td>
<td></td>
<td>$n + 1$</td>
</tr>
</tbody>
</table>

M-acceptance. There it turned out, that the issue of completeness for finite automata is most relevant for this consideration. For instance, while every regular language can be $B$-accepted modulo the empty word by a nondeterministic non-complete finite automata (in fact these machines characterize the regular languages) the language family induced by nondeterministic complete finite automata is a strict subset of the regular languages. Finally, we have studied descriptional complexity aspects of $B$- and $M$-acceptance compared with ordinary acceptance on finite automata. In some cases, we were able to prove tight bounds for the conversion problems. In particular, for $M$-acceptance an $n \cdot 2^{n-2}$ upper and a $3 \cdot 2^{\lfloor \frac{n-2}{4} \rfloor - 2}$ lower bound for the conversion to an ordinary nondeterministic finite automaton was shown. We state it as an open problem to improve these bounds in either direction. Moreover, it remains to consider state complexity results for the direct conversion of a nondeterministic finite automaton that $B$-accepts a language to an equivalent finite state device $M$-accepting the same language and vice versa.

References