Abstract. It is known that deterministic finite automata (DFAs) can be algorithmically minimized, i.e., a DFA $M$ can be converted to an equivalent DFA $M'$ which has a minimal number of states. The minimization can be done efficiently [6]. On the other hand, it is known that unambiguous finite automata (UFAs) and nondeterministic finite automata (NFAs) can be algorithmically minimized too, but their minimization problems turn out to be NP-complete and PSPACE-complete, respectively [8]. In this paper, the time complexity of the minimization problem for two restricted types of finite automata is investigated. These automata are nearly deterministic, since they only allow a small amount of nondeterminism to be used. The main result is that the minimization problems for these models are computationally hard, namely NP-complete. Hence, even the slightest extension of the deterministic model towards a nondeterministic one, e.g., allowing at most one nondeterministic move in every accepting computation or allowing two initial states instead of one, results in computationally intractable minimization problems.

1 Introduction

Finite automata are a well-investigated concept in theoretical computer science with a wide range of applications such as lexical analysis, pattern matching, or protocol specification in distributed systems. Due to time and space constraints it is often very useful to provide minimal or at least succinct descriptions of such automata. Deterministic finite automata (DFAs) and their corresponding language class, the set of regular languages, possess many nice properties such as, for example, closure under many language operations and many decidable questions. In addition, most of the decidability questions for DFAs, such as membership, emptiness, or equivalence, are efficiently solvable (cf. Sect. 5.2 in [14]). Furthermore, in [6] a minimization algorithm for DFAs is provided working in time $O(n \log n)$, where $n$ denotes the number of states of the given DFA.

It is known that both nondeterministic finite automata (NFAs) and DFAs accept the set of regular languages, but NFAs can achieve exponentially savings in size when compared to DFAs [13]. Unfortunately, certain decidability questions, which are solvable in polynomial time for DFAs, are computationally hard for NFAs such as equivalence, inclusion, or universality [14][15]. Furthermore,
the minimization of NFAs is proven to be \textsc{PSPACE}-complete in \cite{8}. In the latter paper, it is additionally shown that unambiguous finite automata (UFAs) have an \textsc{NP}-complete minimization problem.

Therefore, we can summarize that determinism permits efficient solutions whereas the use of nondeterminism often makes solutions computationally intractable. Thus, one might ask what amount of nondeterminism is necessary to make things computationally hard, or, in other words, what amount of nondeterminism may be allowed so that efficiency is preserved.

Measures of nondeterminism in finite automata were first considered in \cite{12} and \cite{2} where the relation between the amount of nondeterminism of an NFA and the succinctness of its description is studied. Here, we look at computational complexity aspects of NFAs with a fixed finite amount of nondeterminism. In particular, these NFAs are restricted such that within every accepting computation at most a fixed number of nondeterministic moves is allowed to be chosen. It is easily observed that certain decidability questions then become solvable in polynomial time in contrast to arbitrary NFAs. However, the minimization problem for such NFAs is proven to be \textsc{NP}-complete.

We further investigate a model where the nondeterminism used is not only restricted to a fixed finite number of nondeterministic moves, but is additionally cut down such that only the first move is allowed to be a nondeterministic one. Hence we come to DFAs with multiple initial states (MDFAs) which were introduced in \cite{5} and recently studied in \cite{11} and \cite{3}. The authors of the latter paper examine the minimization problem for MDFAs and prove its \textsc{PSPACE}-completeness. Their proof is a reduction from Finite State Automata Intersection \cite{4} which states that it is \textsc{PSPACE}-complete to answer the question whether there is a string \( x \in \Sigma^* \) accepted by each \( A_i \), where DFAs \( A_1, A_2, \ldots, A_n \) are given. As is remarked in \cite{4}, the problem becomes solvable in polynomial time when the number of DFAs is fixed. We would like to point out that the number of initial states is not part of the instance of the minimization problem for MDFAs discussed in \cite{3}. Thus, one might ask whether minimization of MDFAs with a fixed number of initial states is possible in polynomial time. We will show in Sect. 3 that the minimization problem of such MDFAs is \textsc{NP}-complete even if only two initial states are given. In analogy to NFAs with fixed finite branching, certain decidability questions can be shown to be efficiently solvable.

\section{Preliminaries and Definitions}

Let \( \Sigma^* \) denote the set of all strings over the finite alphabet \( \Sigma \), \( \epsilon \) the empty string, and \( \Sigma^+ = \Sigma^* \setminus \{\epsilon\} \). By \( |w| \) we denote the length of a string \( w \) and by \( |S| \) the cardinality of a set \( S \). We assume that the reader is familiar with the common notions of formal language theory as presented in \cite{7} as well as with the common notions of computational complexity theory that can be found in \cite{4}. Let \( L \) be a regular set; then \( \text{size}(L) \) denotes the number of states of the minimal DFA accepting \( L \). We say that two finite automata are equivalent if both accept the same language. The size of an automaton \( M \), denoted by \( |M| \), is defined to be...