Computational Complexity of NFA Minimization for Finite and Unary Languages

Hermann Gruber and Markus Holzer

1 Institut für Informatik, Ludwig-Maximilians-Universität München, Oettingenstraße 67, D-80538 München, Germany
email: gruberh@tcs.ifi.lmu.de

2 Institut für Informatik, Technische Universität München, Boltzmannstraße 3, D-85748 Garching bei München, Germany
email: holzer@in.tum.de

Abstract. We investigate the computational complexity of the nondeterministic finite automaton (NFA) minimization problem for finite and unary regular languages, if the input is specified by a deterministic finite state machine. While in general the NFA minimization problem is PSPACE-complete [15], it becomes easier when considering the aforementioned language families. It is easy to see that in both cases, the upper bound is $\Sigma^P_2$, the second level of the Polynomial Hierarchy.

Concerning the lower bound, we show that minimization problem for NFAs accepting finite languages is hard for the complexity class $\text{DP}$, which includes both $\text{NP}$ and $\text{coNP}$, and is a subset of $\Sigma^P_2$. Moreover, we show that the corresponding problem for unary regular languages in general, i.e., not limited to the cyclic case, can be approximated in polynomial time within a performance ratio of $O(\sqrt{n})$, where $n$ is the number of states of the given deterministic finite state machine. This improves a recent result on the approximation of NFAs accepting cyclic unary languages [8]. We also show that one cannot approximate the unary NFA minimization problem with $o(n)$, if the input is a NFA, which is an optimal bound, unless $P = \text{NP}$.

1 Introduction

Finite automata are one of the oldest and most intensely investigated computational models. It is well known that deterministic and nondeterministic finite automata are computationally equivalent, and that nondeterministic finite automata can offer exponential state savings compared to deterministic ones [24]. On the other hand, minimizing deterministic finite automata (DFAs) can be carried out efficiently, whereas the state minimization problem for nondeterministic finite state automata (NFAs) is PSPACE-complete, even if the regular language is specified as a DFA [15]. This theoretical problem is quite relevant for applications where finite automata are involved such as, e.g., circuit design, natural language processing, computational biology, parallel processing, image compression, to mention a few [3, 6, 13, 26], because it measures the amount of space needed to store the devices under consideration in memory. Common to most of these applications is that they have to deal with huge masses of data. The situation is even worse, because recently it was shown that the NFA minimization problem cannot even be approximated within $o(n)$, unless $P = \text{PSPACE}$, if the input is given by a NFA with $n$ states [9]. If the input is a DFA the problem remains inapproximable within a factor of at least $n^{2/3 - \epsilon}$, for all $\epsilon > 0$, unless $P = \text{NP}$ [10].

This immediately prompts the question whether the complexity of the minimization problem drops, if restricted types of regular languages such as, e.g., finite or unary regular languages are considered. From the descriptional complexity point of view in all the
aforementioned cases, the blow-up between NFAs and DFAs remains exponential \([4, 5, 25, 14]\), but is slightly better compared to the general case. Recently in \([17]\) it was shown that for a large variety of restricted types of NFAs, in particular to models where the nondeterministic moves are limited, the minimization problem remains intractable, i.e., is at least \(\text{NP}\)-hard. What concerns the complexity of minimization of the aforementioned restrictions of finite and unary regular languages?

For finite languages NFAs minimization can be solved by the following algorithm: A nondeterministic Turing machine with an NFA equivalence oracle for finite languages can guess an NFA with at most \(k\) states and ask the oracle whether the guessed automaton is equivalent to the input automaton and accept if and only if the oracle answer is yes. Since NFA equivalence for finite languages is \(\text{coNP}\)-complete \([27]\) the minimization problem belongs to \(\Sigma^P_2\), regardless whether a deterministic or nondeterministic finite state device is given. The best lower bound, to our knowledge, is \(\text{NP}\)-hardness, which follows from \([27]\). The problem of minimizing a given unary NFA is \(\text{coNP}\)-hard \([27]\) and similarly as in the case of finite languages contained in \(\Sigma^P_2\), and the number of states of a minimal NFA, equivalent to a given unary DFA, cannot be computed in polynomial time, unless \(\text{NP} \subseteq \text{DTIME}(n^{O(\log n)})\), as shown in \([14]\). Note that in the latter case the corresponding decision version belongs to \(\text{NP}\). Nonapproximability results for the problem in question have been found recently, if the input is a unary NFA—these results even hold for unary cyclic languages: The problem cannot be approximated in \(\frac{n}{\log n}\) \([8]\), and if we require in addition the explicit construction of an equivalent NFA, the nonapproximability ratio can be raised to \(n^{1-\epsilon}\), for every \(\epsilon > 0\), unless \(\text{P} = \text{NP}\) \([9]\). On the other hand, if a unary cyclic DFA with \(n\) states is given, the nondeterministic state complexity of the considered language can be approximated within a factor of \(O(\log n)\). In this paper we contribute to these results as follows:

1. In case of finite languages we improve the NFA minimization \(\text{NP}\) lower bound to \(\text{DP}\)-hardness, if the input is a DFA accepting a finite language. The complexity class \(\text{DP}\) includes both \(\text{NP}\) and \(\text{coNP}\), and is a subset of \(\Sigma^P_2\). This nicely contrasts a recent result \([10]\) on the \(\text{NP}\)-completeness of NFA minimization for finite languages given by truth tables. Hence, the NFA minimization for finite languages, when given a DFA, is provably more complicated compared to truth table input, unless \(\text{NP} = \text{coNP}\). Whether this lower bound can be substantially improved further to, e.g., \(\Sigma^P_2\)-hardness, has to be left open.

2. For unary languages we improve some of the aforementioned (non)approximability results, which only hold for the cyclic case, to unary languages in general. In particular, we prove that for a given a \(n\)-state NFA accepting an unary language \(L\), it is impossible to approximate the nondeterministic state complexity of \(L\) within \(o(n)\), unless \(\text{P} = \text{NP}\). Observe that this bound is tight. In contrast, it is shown that the NFA minimization can be constructively approximated within \(O(\sqrt{n})\), where \(n\) is the number of states of the given DFA. Here constructively approximated means that we can build the NFA, instead of only approximately determining the number of states needed. Note that in the latter result we solve an open problem stated in \([15]\) on the complexity of converting a DFA to an approximately optimal NFA in the case of unary languages.

The paper is organized as follows: In the next section we introduce the basic notions on finite automata and complexity theory. Sections 3 and 4 are devoted to results on the approximation complexity of the unary NFA minimization problem. The former section deals with NFAs as input to the problem under consideration, while the latter treats the case where the input is given as a DFA. Then in Section 5 the minimization problem for
NFAs accepting finite languages is investigated. Here the DP lower bound, based on a modified construction first presented in [27], is shown.

2 Definitions

We assume the reader to be familiar with the basic notations in formal language and automata theory as contained in [12]. In particular, let $\Sigma$ be an alphabet and $\Sigma^*$ the set of all words over the alphabet $\Sigma$ containing the empty word $\lambda$. The length of a word $w$ is denoted by $|w|$, where $|\lambda| = 0$.

A nondeterministic finite automaton (NFA) is a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$, where $Q$ is a finite set of states, $\Sigma$ is a finite set of input symbols, $\delta : Q \times \Sigma \rightarrow 2^Q$ is the transition function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of accepting states. The transition function $\delta$ is extended to a function from $\delta : Q \times \Sigma^* \rightarrow 2^Q$ in the natural way, i.e., $\delta(q, \lambda) = \{q\}$ and $\delta(q, aw) = \bigcup_{q' \in \delta(q, a)} \delta(q', w)$, for $q \in Q$, $a \in \Sigma$, and $w \in \Sigma^*$. A nondeterministic finite automaton $A = (Q, \Sigma, \delta, q_0, F)$ is deterministic (DFA), if $|\delta(q, a)| = 1$, for every $q \in Q$ and $a \in \Sigma$. In this case we simply write $\delta(q, a) = p$ instead of $\delta(q, a) = \{p\}$. The language accepted by $A$ is $L(A) = \{w \in \Sigma^* \mid \delta(q_0, w) \cap F \neq \emptyset\}$. Two automata are equivalent if they accept the same language. For a regular language $L$, the deterministic (nondeterministic, respectively) state complexity of $L$, denoted by $sc(L)$ ($\text{nsc}(L)$, respectively) is the minimal number of states needed by a DFA (NFA, respectively) accepting $L$.

In this paper we are interested in the minimization problem for NFAs. This problem is defined as follows:

- For a given finite automaton $A$ and an integer $k$, decide whether there exists a NFA $B$ with at most $k$ states such that $L(A) = L(B)$.

In order to classify this problem we assume the the reader to be familiar with some basic notions of complexity theory, as contained in [21]. In particular we consider the following well-known sequence of containments: $\text{P} \subseteq \text{NP} \subseteq \text{PSPACE}$. Here $\text{P}$ ($\text{NP}$, respectively) is the set of problems accepted by deterministic (nondeterministic, respectively) polynomial time bounded Turing machines, and $\text{PSPACE}$ is the class of languages accepted by deterministic or nondeterministic Turing machines within polynomial space. Moreover, we introduce the class $\text{DP}$, called difference polytime, which is the class of languages that can be written as the difference of two $\text{NP}$ languages [22]. Obviously, difference polytime $\text{DP} = \{A \cap B \mid A \in \text{NP} \text{ and } B \in \text{coNP}\}$. The class $\text{DP}$ equals $\text{BH}_2$, the second level of the Boolean hierarchy over $\text{NP}$, and thus is a superset of $\text{NP}$ and a subset of $\text{NP}^{\text{NP}}$. By definition, the latter class is $\Sigma_2^p$, the second level of the Polynomial Hierarchy. Completeness and hardness are always meant with respect to deterministic many-one polynomial time reducibilities.

As already mentioned in the introduction, it is well known that the minimization problem for NFAs is $\text{PSPACE}$-complete. Furthermore, the problem remains $\text{PSPACE}$-complete even if the given automaton is guaranteed to be deterministic [15]. However, the computational complexity aspects may vary if the instance to minimize is a restricted type of a finite automaton accepting such as a finite automaton accepting a finite language or a unary language.

A language $L \subseteq \Sigma^*$ is finite if $L = L \cap \Sigma^k$ for some integer $k$, where $\Sigma^k$ refers to all words over the alphabet $\Sigma$ of length at most $k$. For a DFA or NFA to accept a finite language, it is necessary and sufficient, that the directed graph induced by the automaton is acyclic. Thus, a DFA (NFA, respectively) accepting a finite language is said to be an acyclic DFA (NFA, respectively). Moreover, a language $L \subseteq \Sigma^*$ is unary if the alphabet
is a singleton, i.e., $|\Sigma| = 1$. Without loss of generality we may assume that $\Sigma = \{a\}$ in this case. We say that a DFA (NFA, respectively) accepting a unary language is a unary DFA (NFA, respectively). It is not difficult to see that a unary DFA consists of a path, which starts from the initial state, followed by a cycle of one or more states. Following the convention in [4, 5, 18], the size of a unary DFA is the pair $(\lambda, \mu)$, where $\lambda \geq 1$ and $\mu \geq 0$ denote the number of states in the cycle and in the path, respectively. For unary NFAs a normal form which generalizes that for DFAs was established in [4, 5]. There a unary NFA consists of a path, which starts from the initial state, and several cycles, where the last state of the path branches nondeterministically into one state of each cycle. An unary NFA of this form is said to be in Chrobak normal form. Naturally, the size notation $(\lambda, \mu)$ for DFAs carries over to NFAs, where $\lambda \geq 1$ now refers to the number of states belonging to the cycles, and $\mu \geq 0$ is defined as above.

Due to the special shape of unary DFAs, it is easy to see that unary regular languages correspond to ultimately periodic sets of integers. A unary regular language is said to be cyclic if and only if it can be accepted by a DFA of size $(\lambda, 0)$, for some $\lambda \geq 1$. To emphasize the periodicity of this language, we say that it is $\lambda$-cyclic.

3 Approximability of Unary NFA Minimization for a given NFA

As mentioned in the introduction, NFA minimization is PSPACE-hard and [27] and cannot be approximated well, if the input is specified as a NFA. Some nonapproximability results for the corresponding problem for unary input alphabets have been found recently: The decision version cannot be approximated in $\frac{n}{\ln n}$ [8], and if we require in addition the explicit construction of an equivalent NFA the nonapproximability ratio can be raised to $n^{1-\varepsilon}$ for every $\varepsilon > 0$, unless $P = NP$ [9]. But this is not yet the end of the line. We show that, in fact, the decision version cannot be approximated within $o(n)$.

Theorem 1. Given a $n$-state NFA accepting an unary language $L$, it is impossible to approximate $\text{nsc}(L)$ within $o(n)$, unless $P = NP$.

Proof. Our proof is an adaption of the classical proof of the fact that the problem of determining whether an unary NFA accepts the universal language $\{a\}^*$ is coNP-hard [27]. For convenience and ease of notation, we outline the modified construction completely, not just the modifications.

This proof is by a reduction from the coNP-complete unsatisfiability problem for 3SAT-formulae: Given $F$ as the conjunction of clauses $C_1, C_2, \ldots, C_m$ in the variables $x_1, x_2, \ldots, x_n$, where each clause is the disjunction of at most 3 literals, it is coNP-complete to determine whether $F$ is unsatisfiable. This problem remains coNP-hard if we require that no clause has more than one occurrence of each variable, and that the last clause is of the form $C_m = (x_n)$, where $x_n$ is a variable occurring only in $C_m$. For reasons later obvious, we deviate in this point from the classical reduction. Now the core idea of the original construction is to find a suitable unary encoding of truth assignments in $\{0,1\}^n$ for the variables $x_1, x_2, \ldots, x_n$. Let $p_1, p_2, \ldots, p_n$ be $n$ distinct primes (to be fixed later), among which $p_n$ is the largest and $p_{n-1}$ is the second largest prime. Define the function $\mu : \mathbb{N} \to \mathbb{N}^n$ by

$$
\mu(x) = \left( \begin{array}{c}
x \mod p_1 \\
x \mod p_2 \\
\vdots \\
x \mod p_n
\end{array} \right).
$$

4
We call $x$ a code, if $\mu(x)$ is a $n$-dimensional vector with 0-1 entries. According to the Chinese Remainder Theorem [11], every assignment in $\{0, 1\}^n$ has a unique code modulo $\prod_{i=1}^n p_i$, but not every number in this module represents a code in general.

We will define a language $LF$ which is equal to $\{a\}^+$ iff $F$ is unsatisfiable. First, let $\text{CODE}_n = \{ a^k \mid k \mod p_i \notin \{0, 1\} \}$. Then we have

$$\text{CODE} = \{ a^k \mid k \text{ is not a code} \} = \bigcup_{i=1}^n \text{CODE}_i,$$

and a NFA accepting this language can be constructed in time polynomial in $n \cdot p_n$ from $F$ and the list of primes. Next, observe for a clause $F$, a unique assignment can be found in $\{0, 1\}^n$ such that the clause is not satisfied. Thus the language of all codes $x$ such that $\mu(x)$ does not satisfy $F$ is given by

$$\text{L}_C = \{ a^k \mid k \mod p_1 = a_1 \} \cap \{ a^k \mid k \mod p_2 = a_2 \} \cap \{ a^k \mid k \mod p_3 = a_3 \}.$$ 

And a NFA accepting $\text{L}_C$ of size $p_1 \cdot p_2 \cdot p_3$ can be constructed in time polynomial in $p_n$. Finally, we define the language $LF$ as $\bigcup_{m=1}^n \text{L}_C \cup \text{CODE}$. It can be seen from the construction that $LF$ is a cyclic language, and that $LF = \{a\}^*$ iff $F$ is unsatisfiable.

Given the list of primes and the formula $F$, we can construct a NFA accepting $LF$ in time polynomial in $p_n \cdot m$. Moreover, if $p_n$ is used to encode the special variable $x_n$, then we need only one cycle whose length is a multiple of $p_n$, namely $a$ for the language $\text{CODE}_n \cup L_{C_n}$, which has period $p_n$. So we can assume that the size of this automaton is $N = p_n + O(m \cdot p_{n-1}^3)$.

Now we fix the primes $p_1, p_2, \ldots, p_{n-1}$ to be the first $n - 1$ primes. By the prime number theorem holds $p_n \leq 2n \ln n$ for $n$ large enough [11], thus these primes can be enumerated in time polynomial in $n$. In order to achieve that the size of $p_n$ predominates in the size of the constructed NFA we set $p_n$ to be the prime larger than $m(p_{n-1})^3$. Betrand’s Postulate [11] asserts that $p_n \leq 2m(p_{n-1})^3$, and thus $p_n$ can also be found in time polynomial in $m \cdot n$. We conclude that for the size of the constructed NFA holds $N = \Theta(p_n)$.

Clearly, if $F$ is unsatisfiable, then the minimal period of $LF$ equals 1, and $\text{nsc}(LF) = 1$. For the other case, the classical construction was analyzed in [8, Lemma 3]. There it was shown that the minimal period of $LF$ is at least $\frac{1}{2} \prod_{i=1}^n p_i$, provided $LF$ is not universal—the proof was given in the setup where the involved primes to encode the truth assignments are the first $n$ primes, but the proof remains valid in general for any list of primes. As $LF$ is cyclic, a special lower bound technique from [14, Corollary 2.1] can be applied, implying that the nondeterministic state complexity is bounded below by the largest prime factor of its minimal period. As noted before, $p_n = \Theta(N)$, and we have $\text{nsc}(LF) = \Theta(N)$ in this case.

Now assume there is a polynomial time algorithm approximating the size of a minimal equivalent unary NFA within $o(N)$, where $N$ is the number of states of the given NFA. Then this algorithm could be applied to decide whether $\text{nsc}(LF) = o(p_n)$, thus solving a coNP-hard problem in polynomial time, which implies $P = \text{NP}$. □

The reader should note that the assumption $P \neq \text{NP}$ probably cannot be replaced by a weaker assumption such as $P \neq \text{PSPACE}$. As mentioned in the introduction the NFA minimization problem belongs to $\Sigma_2^P$. And note that the assumption $P \neq \text{NP}$ is logically equivalent to $P \neq \Sigma_2^P$.
4 Approximability of Unary NFA Minimization for a given DFA

In contrast to the result in the previous section we describe an approximation algorithm, which, for a given DFA accepting an unary language, constructs in polynomial time an equivalent NFA whose size is at most quadratic in the size of the equivalent minimal NFA. A similar result was known for the special case where the given DFA is cyclic [8], of which our algorithm is an extension. For the proof of the next theorem, we collect some known facts about unary finite automata. The following characterization is due to [19, Lemma 1]—we present a version given in [23]:

Theorem 2. An unary DFA of size $(\lambda, \mu)$ accepting the language $L$ is minimal if and only if the following two conditions are met:

1. For any maximal proper divisor $d$ of $\lambda$, there exists an integer $h$ with $0 < h < \lambda$ such that $a^{\mu + h} \in L$ if and only if $a^{\mu + h + d} \not\in L$, and
2. $a^{\mu - 1} \in L$ if and only if for all $k > 0$ holds $a^{\mu + k \cdot \lambda - 1} \not\in L$.

As a corollary, we obtain:

Corollary 3. Assume $A$ is a minimal unary DFA of size $(\lambda, \mu)$. Then both $\lambda$ and $\mu$ are minimal parameters among all DFA accepting $L$, i.e., there is no equivalent DFA of size $(\lambda', \mu')$ with $\lambda' < \lambda$ or $\mu' < \mu$. □

Moreover, we recall some of the main results relating nondeterministic state complexity and unary NFA in Chrobak normal form from [4, 5].

Theorem 4. For every $n$-state unary NFA, there is an equivalent NFA in Chrobak normal form of size $(n, \mu)$ and an equivalent DFA of size $(\lambda, \mu)$ with $\lambda = 2^{O(\sqrt{n \log n})}$ and $\mu = O(n^2)$.

Now we are ready to prove the main result of this section:

Theorem 5. Given an DFA of size $(\lambda, \mu)$ accepting an unary language $L$, the size of a minimal equivalent NFA can be constructively approximated within $O(\sqrt{\mu} + \log \lambda)$ in polynomial time—observe that $O(\sqrt{\mu} + \log \lambda)$ is in $O(\text{nsc}(L))$.

Proof. We begin with a description of the algorithm: Without loss of generality, we assume that the given DFA is a minimal DFA. The algorithm first constructs a minimal DFA of size $(\lambda, 0)$ accepting the residue language $L' = a^{-\mu}L$, which is cyclic and of minimal period $\lambda$. This can be easily done by “chopping the tail” of the DFA. In [8], the NFA minimization problem for unary cyclic languages is reduced to the weighted set cover problem. While the decision version of the latter problem is NP-complete, it admits an approximation algorithm with an acceptable performance ratio. The analysis in [8] shows that in this way, we can construct a NFA $N'$ in Chrobak normal form accepting $L'$ of size at most $\ell = \text{nsc}(L') \cdot O(\log \lambda)$ in time polynomial in the size of the input. By prepending a tail of length $\mu$ before the original start state of $N'$, we obtain a NFA $N$ accepting the language $L$. Clearly this algorithm runs in polynomial time and the constructed NFA $N$ accepts $L$. It remains to argue that the algorithm achieves the desired performance ratio. In the case $\mu = 0$, the described algorithm coincides with the one given in [8] and gives the performance ratio $O(\log \lambda)$. Thus, the claimed performance ratio is correct.

For the case $\mu > 0$, we will prove first that each NFA in Chrobak normal form accepting $L$ has at least $\mu$ states which are not part of any cycle, which we will refer to as the
tail length of the automaton. For the sake of contradiction, assume $C$ is a NFA in Chrobak normal form accepting $L$ whose tail length is $\mu' < \mu$. A closer look at the determinization procedure given in the proof of [4, 5, Theorem 4.4] shows that an equivalent DFA of size $(\lambda', \mu')$ can be constructed, for some $\lambda'$. But this DFA still accepts $L$, contradicting Corollary 3. As the NFA $N$ constructed by the above algorithm is in Chrobak normal form, is of size $(\ell, \mu)$, and the parameter $\mu$ is minimal among all automata in Chrobak normal form, Theorem 4 implies that

$$\mu = O(\text{nsc}(L)^2) = \text{nsc}(L) \cdot O(\sqrt{\mu}).$$

As $\ell = \text{nsc}(L') \cdot O(\log \lambda)$, the last step in establishing

$$\ell + \mu = \text{nsc}(L) \cdot O(\sqrt{\mu} + \log \lambda)$$

is to show that $\text{nsc}(L') = O(\text{nsc}(L))$: Assume $\text{nsc}(L) = n$. Using again Theorem 4, we deduce that there is a NFA in Chrobak normal form of size $(n, m)$, with $m \geq \mu$. A NFA accepting $a^{-m}L$ in Chrobak normal form of size at most $n + 1$ is obtained from this automaton by replacing the states in the tail with a single start state and connecting it to the cycles appropriately.

We claim that $\text{nsc}(a^{-m}L) = \Theta(\text{nsc}(L'))$. To see this, note $a^{-m}L$ can be written as $a^{x}(a^{-m}L)$ for some $x \geq 0$. As $L' = a^{-\mu}L$ is an unary cyclic language with period $\lambda$, it also holds $a^{-m}L = a^{-x+k\lambda}(a^{-\mu}L)$, for all $k \leq 0$. For $k$ large enough, $-x + k\lambda > 0$, and we have obtained another quotient equation: $a^{-m}L = a^{-k\lambda+x}(a^{-m}L)$. As these languages are mutual quotients, and building quotients increases the nondeterministic state complexity by at most one, we have $\text{nsc}(a^{-m}L) = \Theta(\text{nsc}(L'))$. Hence we have $\text{nsc}(L') = O(\text{nsc}(L))$ as desired. Finally, using Corollary 3 with Theorem 4, we observe that $O(\sqrt{\mu} + \log \lambda) = O(\text{nsc}(L))$. □

For the special case of unary cyclic languages, quite some facts are known about the computational complexity of the unary NFA minimization problem when the input is specified as a DFA. For instance, the problem for this special case is in $\text{NP}$, but not in $\text{P}$ unless $\text{NP} \subseteq \text{DTIME}(n^{O(\log n)})$ [14], and the problem can be approximated within $O(\log \lambda)$, where $(\lambda, 0)$ is the size of the given DFA. In contrast, to our knowledge, the complexity analysis of non-cyclic case, which was an open question in [14], remains open and the best known upper bound for it is $\Sigma^P_2$, as we detailed in Section 3. Things look quite complicated here: Even for the seemingly simple task of the converting an unary NFA into Chrobak normal form, a quasi-polynomial time algorithm has been given only recently [16].

5 Computational Complexity of Minimal NFA Problems for Finite Languages

For finite languages, the situation looks similar to the case of non-cyclic unary languages, at least from the viewpoint of complexity analysis: To our knowledge, the best known lower bound in this case is $\text{NP}$-hardness, which follows from [1], and the equivalence problem for acyclic NFAs is $\text{coNP}$-complete [27], thus giving again an immediate upper bound of $\Sigma^P_2$. The next theorem lifts the above mentioned lower bound to $\text{DP}$-hardness.

**Theorem 6.** The problem of determining for a given acyclic DFA and an integer $k$, whether there exists an equivalent NFA having at most $k$ states is $\text{DP}$-hard.
The proof of this theorem is established in several steps. In the following we briefly summarize the basic line of attack. Recall that every set in DP is the intersection of a NP set with a coNP set. However, also note that the intersection of a NP-hard set and a coNP-hard set is not necessarily DP-hard; the intersection can even be empty in general. But if these two languages satisfy certain additional properties, which prevent too much “interference,” we can prove the sets DP-hard. To establish DP-hardness, we will have to find two sets of instances of the NFA minimization problem under consideration, one of which is hard for NP and the other for coNP, and these sets will have to satisfy some special “non-interference” property.

The next tasks will be to find such suitable sets. We obtain the NP-hard set by careful inspection of a chain of known reductions:

Lemma 7. There is a polynomial time recognizable set \( M \) of pairs \( \langle A, k \rangle \) such that
1. \( A \) is an acyclic DFA and \( k \) an integer, and
2. the nondeterministic state complexity of \( L(A) \) is at least \( k \),
3. but the problem of deciding, for given \( \langle A, k \rangle \in M \), whether \( \text{nsc}(L(A)) \) is at most \( k \), is NP-hard.

Note that, while the membership problem for \( M \) is in P, the question associated with \( M \) as stated in Lemma 9(3) is NP-hard.

Proof. A close inspection of the proof of [20, Theorem 8.1] shows that the concatenation of the first two reductions given there yields a polynomial time construction transforming a graph \( \Gamma = (V', E') \) with \( n \) vertices and \( m \) edges into a bipartite graph \( G \) such that the bipartite dimension \( d(G) \) of \( G \) equals \( \chi(\Gamma) + 2 \cdot \binom{n}{2} - m \), where \( d(G) \) is the least number of complete bipartite subgraphs that are needed to cover all edges in \( G \), and \( \chi(\Gamma) \) denotes the chromatic number of the graph.

The problem of determining whether a non-bipartite graph \( \Gamma \) has chromatic number at most 3 is NP-complete [7]. The set of all non-bipartite graphs is polynomial time recognizable, and every graph in this set has chromatic number at least three, by definition. Thus, if we set \( j = 3+2 \cdot \binom{n}{2} - m \), we obtain a set of graph-integer pairs \( \langle G, j \rangle \) with analogous properties to those stated in the Lemma. Finally, we combine this with a reduction given in [1]: Given a bipartite graph \( G = (U, V, E) \), set \( \Sigma = U \cup V \) and define the language \( L \subseteq \Sigma^2 \) by \( L = \{ uv \mid (u, v) \in E \} \). Then \( \text{nsc}(L) = d(G) + 2 \) and a DFA \( A \) accepting \( L \) can be constructed from \( G \) in polynomial time. Hence, the pairs \( \langle A, k \rangle \) with \( k = j + 2 \) have all of the postulated properties. \( \square \)

Finding a coNP-hard set with a similar property takes more effort, compared to above. As, to our knowledge, coNP-hardness of the problem under consideration had not been established yet.

Lemma 8. The problem of determining for a given acyclic DFA and an integer \( k \), whether there exists an equivalent NFA having at most \( k \) states is coNP-hard.

Proof. The reduction establishing the hardness result relies on a definition of a special language \( L \) commonly specified by multiple deterministic finite automata—recall the construction given in [15]. We combine this with an adaption of a folk reduction showing that the bounded universality problem for acyclic nondeterministic finite automata is coNP-hard.

Given a Boolean formula \( F \) in disjunctive normal form involving variables \( x_1, x_2, \ldots, x_n \) and having \( m \) clauses, we can construct in polynomial time trim DFAs \( A_1, A_2, \ldots, A_m \)
such that \( A_i \) accepts the set of assignments \( t = t_1t_2 \ldots t_n \) satisfying the \( i \)th clause. Then \( \bigcup_i L(A_i) = \{0,1\}^n \) iff \( F \) is a tautology, the latter being a coNP-complete problem.

Without loss of generality, we assume that that \( A_i \) has state set \( Q_i = \{q_{i0}, q_{i1}, \ldots, q_{in}\} \), and for each \( j \), there is a word \( w_{ij} \) of length \( j \) such that \( A_i \) is in state \( q_{ij} \) after reading \( w_{ij} \). We also assume that \( Q_i \cap Q_j = \emptyset \) for \( i \neq j \). The language \( P(i,j) \) is defined as the set of words which could be accepted by \( A_i \) if \( q_{ij} \) was redefined as the only accepting state, that is \( P(i,j) = \{ w \in \{0,1\}^* \mid \delta_i(q_{i0}, w) = q_{ij}\} \). We introduce a new symbol \( a_i \) for each automaton \( A_i \), and a new symbol \( b_{ij} \) for each state \( q_{ij} \) in \( \bigcup_{i=1}^m Q_i \). In addition, we have new symbols \( c_1, c_2, \ldots c_n \), and \( d \). Define the language \( P(i) \) as a marked version of the language accepted by \( A_i \): \( P(i) = \bigcup_{j=0}^n [a_i \cdot P(i,j) \cdot b_{ij}] \). Let \( B_j = \{ b_{ij} \mid 1 \leq i \leq m \} \). Then the auxiliary language \( R \) is defined as the set

\[
R = \{(0,1,c_1) \cdot \{0,1,c_2\} \cdots \{0,1,c_n\} \cdot \{d\} \} \bigcup_{i=1}^n \{(0,1,c_1) \cdot \{0,1,c_2\} \cdots \{0,1,c_i \} \cdot B_i \}.
\]

Lastly, let \( L = \bigcup_{i=1}^m [P(i) \cup a_i L(A_i)] \cup R \cup \{0,1\}^n \).

The role of the set \( R \) is to assert that all strings of the form \( x_{b_{ij}} \) with \( x \in \{0,1\}^j \) are in \( L \), and the marker symbols \( c_j \) ensure any NFA accepting \( L \) needs \( n-1 \) states in addition to those needed to accept \( \bigcup_{i=1}^m [P(i) \cup a_i L(A_i)] \cup \{0,1\}^n \).

Given \( A_1, A_2, \ldots A_m \), it is easy to construct in polynomial time a partial DFA with \( n + 1 \) states accepting \( \{0,1\}^n \), a partial DFA with \( n + 2 \) states accepting \( R \) depicted in Figure 1, and a partial DFA with \( 2 + m(n+1) \) states accepting \( \bigcup_{i=1}^m [P(i) \cup a_i L(M_i)] \). By the well-known product construction, a DFA accepting the union of these three languages can be obtained in polynomial time, and the union of these languages equals \( L \).

![Fig. 1. A partial DFA accepting language R.](image)

We will show that the size of the minimal NFA accepting \( L \) has in any case at least \( k = 2 + m(n + 1) + n \) states, and that this lower bound is exact iff \( F \) is a tautology. If \( F \) is a tautology, then the \( k \)-state NFA sketched in Figure 2 accepts \( L \).

To give a lower bound on the size of any NFA accepting \( L \), we use the (extended) fooling set technique [2]: Define the set of pairs \( S = S_1 \cup S_2 \) with

\[
S_1 = \{(a_i w_{ij}, b_{ij}) \mid 1 \leq i \leq n \text{ and } 1 \leq j \leq m\}
\]

and

\[
S_2 = \{(x,y) \mid xy = c_1 c_2 \ldots c_n d\},
\]

where \( w_{ij} \) is any word in \( P(i,j) \), for each \( 1 \leq i \leq n \) and \( 1 \leq j \leq m \). We claim that \( S \) is an extended fooling set for \( L \).
It is readily observed that \( xy \in L \) for all pairs \((x, y) \in S\). Next, we note that the word \( a_i w_{ij} b_{i\ell} \) is in \( L \) if and only if \( j = \ell \). Of course, if \( j = \ell \) then \( a_i w_{ij} b_{i\ell} \in L \). Assume now \( i \neq \ell \). Since the word begins with \( a_i \) and ends with \( b_{i\ell} \), it is not in \( L \), or it is in \( P(i) \cup a_i \cdot L(A_i) \).

It is clear that \( w_{ij} \in P(i, j) \). Any word in \( P(i) \) ending with \( b_{i\ell} \) is in \( a_i \cdot P(i, \ell) \cdot b_{i\ell} \), so \( w_{ij} \in P(i, j) \cap P(i, \ell) \). But automaton \( A_i \) is deterministic, so \( P(i, j) \cap P(i, \ell) = \emptyset \) if \( j \neq \ell \), and thus \( a_i w_{ij} b_{i\ell} \not\in L \). Thus, all elements in \( S_1 \) have the fooling set property.

We turn to the elements in \( S_2 \): Let \( w = c_1 c_2 \ldots c_n d \). Obviously, \( w \) is in \( L \), but \( ww \) is not. And none of the words \( a_i w_{ij} w \) or \( w b_{ij} \) are in \( L \), so we can add the pairs \((\varepsilon, w)\) and \((w, \varepsilon)\) to \( S_1 \) to form a larger fooling set. Next, note that no proper subword of \( w \) is in \( L \), so \( S_2 \) for itself is also a fooling set for \( L \). To see that all the remaining pairs in \( S_2 \) can be added to \( S_1 \), observe that \( a_i w_{ij} y \) cannot be in \( L \) if \( y \) ends with the letter \( d \).

![Fig. 2. Sketch of construction for a minimal NFA accepting \( L_F \) in case \( F \) is a tautology. The drawn two final states labeled \( p_f \) are actually a single state.](image-url)
Now assume \( F \) is not a tautology, and let \( t \) represent a truth value assignment such that \( F \) evaluates to 0. Write \( t = xy \) with \( 0 < |x| < n \). We claim that \( S \cup \{(x, y)\} \) (the union is disjoint) is also a fooling set for \( L \). For sake of contradiction, assume this is not the case. Then there is \((x', y') \in S \) such that \( xy' \) and \( x'y \) are both in \( L \). We first rule out the case that \( (x', y') \in S'' \). Then \( x_0b_{1,1} \notin L \), if \( |x| \geq 1 \), and \( a_1b_{1,1}y \notin L \), if \( |y| \geq 1 \). Any word in \( L \) beginning with \( c \) ends either with \( f \), or \( b_{i,j} \), for some \( i,j \). Hence, neither \( cy \) nor \( cdy \) is in \( L \). So \( (x', y') \) must be in \( S' \) and of the form \( (a_iw_{ij}, b_{ij}) \). Then both \( a_iw_{ij}y \) and \( x_{b_{ij}} \) are in \( L \). We can deduce that \( w_{ij}y \in L(A_i) \), since the word \( a_iw_{ij}y \) begins with \( a_i \), and \( x \in P(i,j) \), since the word \( xb_{ij} \) ends with \( b_{ij} \). Since \( A_i \) is deterministic and \( w_{ij} \) is also in \( P(i,j) \), we have \( w_{ij} \equiv_{L(A_i)} x \), where \( \equiv_{L(A_i)} \) is the well-known Myhill-Nerode equivalence relation [12] for \( L(A_i) \). But \( w_{ij}y \in L(A_i) \) implies, by definition of the equivalence relation, that \( xy \in L(A_i) \). Thus \( t = xy \) is a satisfying assignment for \( F \), contradicting our original assumption. \( \square \)

We designed the above reduction in a way such that the involved set of instances of the minimization problem, apart from being \( \text{coNP} \)-hard, has a non-interference property similar to the one given in Lemma 7.

**Lemma 9.** There is a polynomial time recognizable set \( N \) of pairs \( \langle B, \ell \rangle \) such that

1. \( B \) is an acyclic DFA and \( \ell \) an integer, and
2. the nondeterministic state complexity of \( L(B) \) is at least \( \ell \),
3. but the problem of deciding, for given \( \langle B, \ell \rangle \in N \), whether \( \text{nsc}(L(B)) \) is at most \( \ell \), is \( \text{coNP} \)-hard.

Now we are ready to complete the proof of the main theorem of this section.

**Proof (of Theorem 6).** Without loss of generality, we assume that for each \( \langle A, k \rangle \in M \) and \( \langle B, \ell \rangle \in N \), the input alphabets of \( A \) and \( B \) have an empty intersection. Let \$ be a new symbol present in neither of the two alphabets. Observe that then the nondeterministic state complexity of the marked concatenation \( L = L(A)\$L(B) \) is precisely the sum of the nondeterministic state complexities of \( L(A) \) and \( L(B) \), and a DFA accepting this language can be constructed in time polynomial in the size of \( A \) and \( B \).

Now the problem of determining whether \( L \) admits an NFA with at most \( k + \ell \) states is \( \text{DP} \)-hard: If \( \text{nsc}(L) \geq k + \ell + 1 \), then by the properties of the sets \( M \) and \( N \) established in Lemma 7 and Lemma 9, \( \text{nsc}(L(A)) > k \) or \( \text{nsc}(L(B)) > \ell \). Thus \( \text{nsc}(L) \leq k + \ell \), if and only if both \( \text{nsc}(L(A)) \leq k \) and \( \text{nsc}(L(B)) \leq \ell \). As the latter problems are \( \text{NP} \)-hard and \( \text{coNP} \)-hard, respectively, the proof is completed. \( \square \)

Our \( \text{DP} \)-hardness result shows again, that the computational complexity of NFA minimization problems heavily depends on the given input format, because \( \text{DP} \)-hardness drops to \( \text{NP} \)-completeness if the finite language is homogenous and given by a truth table [10].

**References**