Computational Complexity of NFA Minimization for Finite and Unary Languages

Hermann Gruber[1] and Markus Holzer[2]

[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 the general case of this problem is **PSPACE**-complete [13], it becomes theoretically easier when restricted to the aforementioned language families. It is easy to see that in both cases, an upper bound is $\mathbf{\Sigma_2^P}$, the second level of the Polynomial Hierarchy. Concerning a respective lower bound, we show that the minimization problem for NFAs accepting finite languages is hard for the complexity class **DP**, which includes both **NP** and **coNP**, and is a subset of $\mathbf{\Sigma_2^P}$. 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 $\mathrm{O}(\sqrt{n})$, where n is the number of states of the given deterministic finite state machine. This generalizes a result obtained recently for cyclic unary languages [6]. We also show that one cannot approximate the unary NFA minimization problem with $\mathrm{o}(n)$, if the input is an NFA, which is an optimal bound, unless $\mathbf{P} = \mathbf{NP}$.

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 [19]. 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 [13]. This theoretical problem is quite relevant for applications where finite automata are involved, such as computational biology or natural language processing [4, 20], because it measures the amount of space needed to store the devices under consideration in memory. Common to most 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 $\mathbf{P} = \mathbf{PSPACE}$, if the input is given by an NFA with n states [7]. That is, no polynomial-time algorithm can always determine an approximate solution of size $o(n)$ times the optimum size. If the input is a DFA the problem remains inapproximable within a factor of at least $n^{1/5-\varepsilon}$, for all $\varepsilon > 0$, unless $\mathbf{P} = \mathbf{NP}$ [9].

This immediately prompts the question whether the complexity of the minimization problem drops if restricted types of regular languages such as finite or unary regular languages are considered. Recently in [15] it was shown that for restricted types of NFAs, in particular for models where the nondeterministic moves are limited, the minimization problem remains intractable, i.e., is at least **NP**-hard. What concerns the complexity of minimization for the aforementioned restrictions to finite and unary regular languages?

For finite languages, NFA minimization can be done 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 **coNP**-complete [21] the minimization problem belongs to $\mathbf{\Sigma_2^P}$, regardless of whether a deterministic or nondeterministic finite state device is given. The best lower bound, to our knowledge, is **NP**-hardness, which follows from [1]. The problem of minimizing a given unary NFA is **coNP**-hard [21] and similarly as in the case of finite languages contained in $\mathbf{\Sigma_2^P}$, and the number of states of a minimal NFA equivalent to a given unary cyclic DFA cannot be computed in polynomial time, unless $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{\mathrm{O}(\log n)})$, as shown in [12]. Note that in the latter case the corresponding decision version belongs to **NP**. Inapproximability results for the problem in question have been found recently, if the input is a unary NFA: The problem cannot be approximated within $\sqrt{n}/(\ln n)$ [6], and if we require in addition the explicit construction of an equivalent NFA, the inapproximability ratio can be raised to $n^{1-\varepsilon}$, for every $\varepsilon > 0$, unless $\mathbf{P} = \mathbf{NP}$ [7]. 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 $\mathrm{O}(\log n)$. In this paper we contribute to the known results as follows:

1. For unary languages we improve some of the aforementioned (in)approximability results, which only hold for the *cyclic* case, to unary languages in general. In particular, we prove that for a given an n-state NFA accepting a unary language L, it is impossible to approximate the nondeterministic state complexity of L within $o(n)$, unless $\mathbf{P} = \mathbf{NP}$. Observe that this bound is tight. In contrast, it is shown that the NFA minimization problem can be *constructively* approximated within $\mathrm{O}(\sqrt{n})$, where n is the number of states of the given DFA. Here by "constructively approximated" we mean 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 [6, 13] on the complexity of converting a DFA to an approximately optimal NFA in the case of unary languages.

2. In the case of finite languages we improve the **NP**-hardness of the NFA minimization problem to **DP**-hardness, even if the input is a DFA accepting a finite language. The complexity class **DP** includes both **NP** and **coNP**, and is a subset of $\mathbf{\Sigma_2^P}$. This nicely contrasts with a recent result [9] on the **NP**-completeness of NFA minimization for finite languages given by truth tables. Hence, the NFA minimization problem for finite languages with DFAs as input is more complicated than that with truth tables as input, unless **NP** = **coNP**. Whether this lower bound can be substantially raised to, e.g., $\mathbf{\Sigma_2^P}$-hardness, is left open.

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.

2 Definitions

We assume the reader to be familiar with the basic notions in formal language and automata theory as contained in [11]. In particular, let Σ be an alphabet and Σ^* the set of all words over the alphabet Σ containing the empty word λ. 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, Σ is a finite set of input symbols, $\delta : Q \times \Sigma \to 2^Q$ is the transition function, $q_0 \in Q$ is the initial state, and $F \subseteq S$ is the set of accepting states. The transition function δ is extended to a function from $\delta : Q \times \Sigma^* \to 2^Q$ 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 $\mathrm{sc}(L)$ ($\mathrm{nsc}(L)$, 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* NFA*s*. This problem is defined as follows:

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

In order to classify this problem we assume the reader to be familiar with some basic notions of complexity theory, as contained in [18]. In particular we consider the following well-known sequence of containments: **P** $\subseteq$ **NP** $\subseteq$ **PSPACE**.

Here **P** (**NP**, respectively) is the set of problems accepted by deterministic (nondeterministic, respectively) polynomial time bounded Turing machines, and **PSPACE** is the class of languages accepted by deterministic or nondeterministic Turing machines within polynomial space. Moreover, we introduce the class **DP**, called difference polytime, which is the class of languages that can be written as the difference of two **NP** languages. Obviously, difference polytime equals $\{ A \cap B \mid A \in \mathbf{NP} \text{ and } B \in \mathbf{coNP} \}$. The class is located at the second level of the Boolean hierarchy over **NP**, and thus a superset of **NP** and **coNP**, but $\mathbf{DP} \subseteq \mathbf{\Sigma_2^P}$. Completeness and hardness for complexity classes are always meant with respect to deterministic many-one polynomial time reducibilities.

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 [3], the *size* of a unary DFA is the pair (λ, μ), 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 [3]. 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. A unary NFA of this form is said to be in *Chrobak normal form*. Naturally, the size notation (λ, μ) 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. 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 $\lambda \geq 1$. In that case we say that the language is λ-cyclic. A cyclic language has *minimal period* λ if it is λ-cyclic, but not cyclic for any proper divisor of λ.

3 Inapproximability of Unary NFA Minimization for a given NFA

In this section we give a tight bound on the approximability of the NFA minimization problem for the case where the input is a unary NFA.

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

Proof. Our proof is an adaptation of the classical proof of the fact that the problem of determining whether a unary NFA accepts the universal language $\{a\}^*$ is **coNP**-hard [21]. 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 that will become later obvious, on this point we differ from the classical reduction. Now the core idea of the original construction is to find a suitable unary representation 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 one. Define the function $\mu : \mathbb{N} \to \mathbb{N}^n$ by $\mu(x) = (x \bmod p_1, x \bmod p_2, \ldots, x \bmod p_n)$. If $\mu(x)$ is a n-dimensional vector with 0-1 entries, we call x a *representation*. According to the Chinese Remainder Theorem [10], every assignment in $\{0,1\}^n$ has a unique representation modulo $\prod_{i=1}^{n} p_i$, but not every number in this module represents an assignment in general.

We will define a language L_F which is equal to $\{a\}^*$ if and only if F is unsatisfiable. First, let $\overline{\mathrm{R}_i} = \{\, a^k \mid k \mod p_i \notin \{0,1\} \,\}$. Then we have

$$\overline{\mathrm{R}} = \{\, a^k \mid k \text{ does not represent an assignment} \,\} = \bigcup_{i=1}^{n} \overline{\mathrm{R}_i},$$

and an NFA accepting this language can be constructed in time $\mathrm{O}(n \cdot p_n)$ from the list of primes. Next, observe for a clause C with variables, say, x_1, x_2, x_3, there is a unique assignment a_1, a_2, a_3 to these variables such that the clause is not satisfied. Thus the language of all representations x such that $\mu(x)$ does not satisfy C is given by

$$L_C = \bigcap_{i=1}^{3} \{\, a^k \mid k \mod p_i = a_i \,\}.$$

Also, an NFA accepting 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 L_F as $\bigcup_{i=1}^{m} L_{C_i} \cup \overline{\mathrm{R}}$. It can be readily seen that L_F is a cyclic language, and that $L_F = \{a\}^*$ if and only if F is unsatisfiable.

Given the list of primes and the formula F, we can construct an NFA accepting L_F in time polynomial in $p_n \cdot m$, whose states are arranged in a union of cycles. Overmore, if p_n is used to represent the special variable x_n, then the constructed NFA needs only one cycle whose length is a multiple of p_n, namely for the language $\overline{\mathrm{R}_n} \cup L_{C_m}$, which has period p_n. So we can assume that the size of this automaton is $N = p_n + \mathrm{O}\big(m \cdot p_{n-1}^3\big)$. 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 $p_{n-1} \leq 2n \ln n$ for n large enough [10], thus these primes can be found in time polynomial in n. Now comes the second point where we deviate from the classical reduction: We want to achieve that the size of p_n predominates in the size of the constructed NFA, so we set p_n to be the first prime greater than $m(p_{n-1})^3$. Bertrand's Postulate [10] 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 $L_F = \{a\}^*$, and $\mathrm{nsc}(L_F) = 1$. For the other case, the classical construction was analyzed in [6, Lemma 3]. There it was shown that the minimal period of L_F is at least $\frac{1}{2} \prod_{i=1}^{n} p_i$, provided L_F is

not universal—the proof was given in the setup where the involved primes to represent the truth assignments are the first n primes, but the proof is valid for any set of n distinct primes. As L_F is cyclic, it is not hard to prove that its nondeterministic state complexity is bounded below by the largest prime power dividing its minimal period, see [12, Corollary 2.1]. Thus $\mathrm{nsc}(L_F) \geq p_n = \Omega(N)$ in this case, where N is the number of states of the given NFA.

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

The reader should note that the assumption $\mathbf{P} \neq \mathbf{NP}$ probably cannot be replaced by a weaker assumption such as $\mathbf{P} \neq \mathbf{PSPACE}$. As mentioned in the introduction the NFA minimization problem belongs to $\mathbf{\Sigma_2^P}$, and the assumption $\mathbf{P} \neq \mathbf{NP}$ is logically equivalent to $\mathbf{P} \neq \mathbf{\Sigma_2^P}$.

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 a 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 [6], of which our algorithm is an extension. For the proof of the next theorem, we collect first some known facts about unary finite automata. The following is a simple consequence of the characterization of minimal unary DFAs given in [16, Lemma 1]:

Corollary 2. *Assume A is a minimal unary* DFA *of size (λ, μ). Then both λ and μ are minimal parameters among all* DFA *accepting L, i.e., there is no equivalent* DFA *of size (λ', μ') with $\lambda' < \lambda$ or $\mu' < \mu$.* □

Moreover, we recall some of the main results relating nondeterministic state complexity and unary NFAs in Chrobak normal form from [3].

Theorem 3. *For every n-state unary* NFA*, there is an equivalent* NFA *in Chrobak normal form of size (n, μ) and an equivalent* DFA *of size (λ, μ) with $\lambda = 2^{\mathrm{O}(\sqrt{n \log n})}$ and $\mu = \mathrm{O}(n^2)$.*

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

Theorem 4. *There is a polynomial-time algorithm which, given a* DFA *of size (λ, μ) accepting a unary language L, constructs an equivalent* NFA *which has $\mathrm{O}(\sqrt{\mu} + \log \lambda)$ times the size of the minimum state* NFA*—observe that this ratio guarantees a size in $\mathrm{O}(\mathrm{nsc}(L)^2)$.*

Proof. Without loss of generality, we assume that the given DFA is minimal. Our 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 λ. This can be easily done by "chopping the tail" of the DFA. In [6], it is shown that the problem under consideration is approximable within $\mathrm{O}(\log \lambda)$ in the special case where the input is a cyclic DFA of period λ. In this way, we can construct an NFA N' in Chrobak normal form accepting L' of size at most $\ell = \mathrm{nsc}(L') \cdot \mathrm{O}(\log \lambda)$ in time polynomial in the size of the input. By prepending a tail of length μ before the original start state of N', we obtain an 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 [6] and gives the performance ratio $\mathrm{O}(\log \lambda)$. Thus, the claimed performance ratio is correct in this case. For the case $\mu > 0$, we proceed as follows: First, we claim that each NFA in Chrobak normal form accepting L has at least μ states which are not part of any cycle, which we will refer to as the tail length of the automaton. As the NFA N constructed by the above algorithm is in Chrobak normal form, is of size (ℓ, μ), and the parameter μ is minimal among all automata in Chrobak normal form, Theorem 3 implies that $\mu = \mathrm{O}\big(\mathrm{nsc}(L)^2\big) = \mathrm{nsc}(L) \cdot \mathrm{O}\big(\sqrt{\mu}\big)$. As $\ell = \mathrm{nsc}(L') \cdot \mathrm{O}(\log \lambda)$, Then the last step in establishing $\ell + \mu = \mathrm{nsc}(L) \cdot \mathrm{O}\big(\sqrt{\mu} + \log \lambda\big)$ is to prove the claim $\mathrm{nsc}(L') = \mathrm{O}(\mathrm{nsc}(L))$. The proofs of both claims make use of Corollary 2 and Theorem 3. The technical details of the analysis are omitted due to lack of space. □

For the special case of unary cyclic languages, quite a few 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 **NP**, but not in **P** unless $\mathbf{NP} \subseteq \mathbf{DTIME}\left(n^{\mathrm{O}(\log n)}\right)$ [12]. In contrast, to our knowledge, the complexity analysis of the non-cyclic case, stated as an open problem in [12], remains open and the best known upper bound for it is $\mathbf{\Sigma_2^P}$, as detailed in Section 3. Things look quite tough here: Even for the seemingly simple task of converting a unary NFA into Chrobak normal form, a quasi-polynomial time algorithm has been given only recently [14].

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 **NP**-hardness, which follows from [1], and the equivalence problem for NFAs accepting finite languages is **coNP**-complete [21], thus giving again an immediate upper bound of $\mathbf{\Sigma_2^P}$. The next theorem lifts the above mentioned lower bound to **DP**-hardness.

Theorem 5. *The problem of determining for a given* DFA *accepting a finite language and an integer k, whether there exists an equivalent* NFA *having at most k states is* **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 task will be to find such suitable sets. We obtain the **NP**-hard set by chaining some known reductions:

Lemma 6. *There is a* polynomial time *recognizable set M of pairs $\langle A, k\rangle$ such that*

1. *A is a* DFA *accepting a finite language 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* $\mathrm{nsc}(L(A))$ *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 6(3) is **NP**-hard.

Proof. The problem of determining whether the edge set of a given bipartite graph $G = (U, V, E)$ can be covered with at most j bipartite graphs is **NP**-complete [17, Theorem 8.1]. The minimum number of bicliques among these coverings is the bipartite dimension of G and denoted by $d(G)$. The problem remains **NP**-hard even for a polynomial time recognizable subset of instances whose bipartite dimension is guaranteed to be at most $j + 1$ and in $\mathrm{O}(\log |E|)$, see [5, Theorem A.3]. We combine this with a reduction given in [1]: Given such 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 $\mathrm{nsc}(L) = d(G)+2$ 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. □

Finding a **coNP**-hard set with a similar property takes more effort, compared to above. To our knowledge, **coNP**-hardness of the problem given below has not been established yet.

Lemma 7. *The problem of determining for a given* DFA *accepting a finite language 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 DFAs—recall the construction described in [8, 13]. We combine this with an adaption of a folk reduction that shows **coNP**-hardness for the equivalence problem of two given nondeterministic finite automata accepting finite languages.

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 $t = t_1 t_2 \ldots t_n$ satisfying the ith clause. Then $\bigcup_i L(A_i) = \{0,1\}^n$ if and only if F is a tautology, the latter being a **coNP**-complete problem.

Without loss of generality, we assume that every DFA 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 $P(i,j) = \{\, w \in \{0,1\}^{\leq n} \mid \delta_i(q_{i0}, w) = q_{ij} \,\}$. 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

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

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

The role of 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, 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 this union equals L.

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 if and only if F is a tautology. If F is a tautology, then the k-state NFA sketched in Figure 1 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

$$S_1 = \{\, (a_i w_{ij}, b_{ij}) \mid 1 \leq i \leq n \text{ and } 1 \leq j \leq m \,\}, \quad \text{and}$$
$$S_2 = \{\, (x,y) \mid xy = c_1 c_2 \ldots c_n d \,\},$$

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 $a_i w_{ij} b_{i\ell} \in L$.

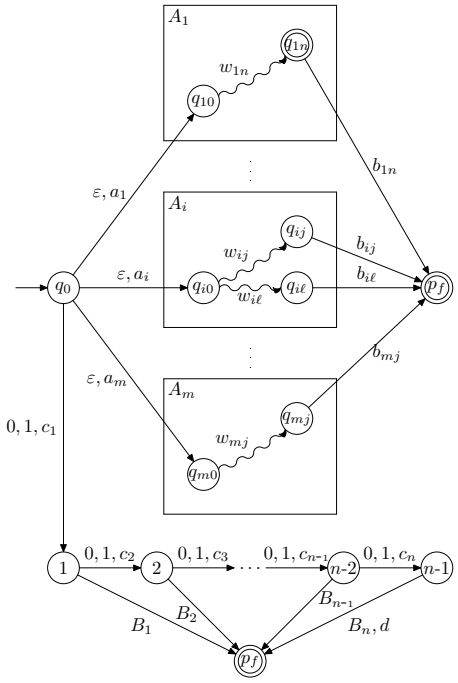

Fig. 1. Sketch of construction for a minimal NFA accepting L_F in case F is a tautology. The drawn two final states labeled p_f are actually a single state.

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 $a_i w_{ij} b_{i\ell} \notin L$. Thus, all elements in S_1 have the fooling set property.

We turn to the elements in S_2: Let $w = c_1 c_2 \dots c_n d$. 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 (ε, w) and (w, ε) 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.

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') is in S''. Then $x a_1 b_{1,1} \notin L$, if $|x| \geq 1$, and $a_1 b_{1,1} y \notin L$, if $|y| \geq 1$. Any word in L beginning with c ends either with f, or b_{ij}, for some i and j. Hence, neither cy nor cdy is in L. So (x', y') must be in S' and of the form $(a_i w_{ij}, b_{ij})$. Then both $a_i w_{ij} y$ and $x b_{ij}$ are in L. We can

deduce that $w_{ij}y \in L(A_i)$, since the word $a_i w_{ij} y$ 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 [11] 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. □

We designed the above reduction in a way such that the involved set of instances of the minimization problem, apart from being **coNP**-hard, has a non-interference property similar to the one given in Lemma 6.

Lemma 8. *There is a* polynomial time *recognizable set N of pairs $\langle B, \ell \rangle$ such that*

1. *B is a* DFA *accepting a finite language and ℓ an integer, and*
2. *the nondeterministic state complexity of $L(B)$ is at least ℓ,*
3. *but the problem of deciding, for given $\langle B, \ell \rangle \in N$, whether* $\mathrm{nsc}(L(B))$ *is at most ℓ, is* **coNP***-hard.* □

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

Proof (of Theorem 5). 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 **DP**-hard: If $\mathrm{nsc}(L) \geq k + \ell + 1$, then by the properties of the sets M and N established in Lemma 6 and Lemma 8, $\mathrm{nsc}(L(A)) > k$, or $\mathrm{nsc}(L(B)) > \ell$. Thus $\mathrm{nsc}(L) \leq k + \ell$, if and only if both $\mathrm{nsc}(L(A)) \leq k$ and $\mathrm{nsc}(L(B)) \leq \ell$. As the latter problems are **NP**-hard and **coNP**-hard, respectively, the proof is completed. □

Acknowledgement. We would like to thank the anonymous referees for their careful reading and many useful suggestions.

References

1. J. Amilhastre, Ph. Janssen, and M.-C. Vilarem. FA minimisation heuristics for a class of finite languages. In O. Boldt and H. Jürgensen, editors, *Proceedings of the 4th International Workshop on Implementing Automata*, number 2214 in LNCS, pages 1–12, Potsdam, Germany, July 2001. Springer.
2. J.-C. Birget. Intersection and union of regular languages and state complexity. *Information Processing Letters*, 43:185–190, 1992.

3. M. Chrobak. Finite automata and unary languages. *Theoretical Computer Science*, 47(2):149–158, 1986. Errata in 302(1–3):497–498, 2003.
4. F. Coulon. *Minimisation d'automates non-déterministes, recherche d'expressions dans un texte et comparaison de génomes*. Ph.D. thesis, Université Rouen, 2004.
5. L. J. Gottlieb, J. E. Savage, and A. Yerukhimovich. Efficient data storage in large nanoarrays. *Theory of Computing Systems*, 38(4):503–536, 2005.
6. G. Gramlich. Probabilistic and nondeterministic unary automata. In B. Rovan and P. Vojtás, editors, *Proceedings of the* 28*th Conference on Mathematical Foundations of Computer Science*, number 2747 in LNCS, pages 460–469, Bratislava, Slovakia, August 2003. Springer.
7. G. Gramlich and G. Schnitger. Minimizing NFA's and regular expressions. In V. Diekert and B. Durand, editors, *Proceedings of the* 22*nd Annual Symposium on Theoretical Aspects of Computer Science*, number 3404 in LNCS, pages 399–411, Stuttgart, Germany, February 2005. Springer.
8. H. Gruber and M. Holzer. Finding lower bounds for nondeterministic state complexity is hard (extended abstract). In O. H. Ibarra and Z. Dang, editors, *Proceedings of the* 10*th International Conference on Developments in Language Theory*, number 4036 in LNCS, pages 363–374, Santa Barbara, USA, June 2006. Springer.
9. H. Gruber and M. Holzer. Inapproximability of nondeterministic state and transition complexity assuming P $\neq$ NP. In T. Harju, J. Karhumäki, and A. Lepistö, editors, *Proceedings of the* 11*th International Conference on Developments in Language Theory*, number 4588 in LNCS, pages 205–216, Turku, Finland, July 2007, Springer.
10. G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford Science Publications, 1979.
11. J. E. Hopcroft and J. D. Ullman. *Formal Languages and Their Relation to Automata*. Addison-Wesley, 1968.
12. T. Jiang, E. McDowell, and B. Ravikumar. The structure and complexity of minimal NFAs over a unary alphabet. *International Journal of Foundations of Computer Science*, 2(2):163–182, June 1991.
13. T. Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM Journal on Computing*, 22(6):1117–1141, December 1993.
14. B. Litow. A special case of a unary regular language containment. *Theory of Computing Systems*, 39(5):743–751, September 2006.
15. A. Malcher. Minimizing finite automata is computationally hard. *Theoretical Computer Science*, 327(3):375–390, November 2004.
16. C. Nicaud. Average state complexity of operations on unary automata. In M. Kutylowski, L. Pacholski, and T. Wierzbicki, editors, *Proceedings of the* 24*th International Symposium on Mathematical Foundations of Computer Science*, number 1672 in LNCS, pages 231–240, Szklarska Poreba, Poland, September 1999. Springer.
17. J. Orlin. Contentment in graph theory: Covering graphs with cliques. *Indagationes Mathematicæ*, 80:406–424, 1977.
18. C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
19. M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:114–125, 1959.
20. K. N. Sgarbas, N. D. Fakotakis, and G. K. Kokkinakis. Incremental construction of compact acyclic NFAs. In *Proceedings of the* 39*th Annual Meeting on Association for Computer Linguistic and* 10*th Conference of the European Chapter*, pages 474–481, Toulouse, France, July 2001. Morgan Kaufmann.
21. L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. In *Proceedings of the 5th Symposium on Theory of Computing*, pages 1–9, 1973.