The Tractability Frontier for NFA Minimization[☆]

Henrik Björklund[a], Wim Martens[b,1]

[a] *Umeå University*
Sweden
[b] *Technical University of Dortmund*
Germany

Abstract

We prove that minimizing finite automata is NP-hard for almost all classes of automata that extend the class of deterministic finite automata. More specifically, we show that minimization is NP-hard for all finite automata classes that subsume the class of δNFAs which accept strings of length at most three. Here, δNFAs are the finite automata that are unambiguous, allow at most one state q with a non-deterministic transition for at most one alphabet symbol a, and are allowed to visit state q at most once in a run. As a corollary, we also obtain that the same result holds for all finite automata classes that subsume that class of finite automata that are unambiguous, have at most two initial states, and accept strings of length at most two.

Key words: Finite automata, optimization, state minimization, complexity.

1. Introduction

The regular languages are a cornerstone of computer science and are a very useful tool in both theory and practice. When using regular languages, the developer is often faced with a trade-off between the descriptive complexity and the complexity of optimization. Concretely, it has been known for a long time that there are regular languages for which non-deterministic finite automata (NFAs) can provide an exponentially more succinct description than deterministic finite automata (DFAs) [27]. On the other hand, many decision problems that are solvable in polynomial time for DFAs, i.e., equivalence, inclusion, and universality, are computationally hard for NFAs.

The choice of a representation mechanism can therefore be crucial. If the set of regular languages used in an application is relatively constant, membership tests are the main language operations, and economy of space is an issue, NFAs are probably the right choice. If, on the other hand, the languages change frequently, and inclusion or equivalence tests are frequent, DFAs may be more attractive.

[☆] The present paper is the full version of reference [6], which appeared in the International Colloquium on Automata, Languages and Programming 2008.

Email addresses: henrikb@cs.umu.se (Henrik Björklund), wim.martens@udo.edu (Wim Martens)

[1] Supported by the North-Rhine Westphalian Academy of Sciences, Humanities, and Arts, and by the Stiftung Mercator Essen.

Since both NFAs and DFAs have their disadvantages, a lot of effort has been spent on trying to find intermediate models, i.e., finite automata that have some *limited* form of non-determinism. The *unambiguous* finite automata (UFAs)[2] form such an intermediate model with rather desirable properties. While in general still being exponentially more succinct than an equivalent DFA for the same language, static analysis questions such as inclusion and equivalence can be solved in PTIME for UFAs [32]. However, UFAs do not allow for tractable state minimization [25]. Therefore, the question whether there are good intermediate models between DFAs and NFAs needs to be revisited for state minimization. Our work is motivated by the question whether there exist useful extensions of the class of DFAs for which minimization is tractable, which was first posed by Jiang and Ravikumar [25]. Our answer is that the existence of such a class is quite unlikely, since we prove that minimization is NP-hard already for a very conservative extension of the class of DFAs.

Every undergraduate computer science curriculum teaches its students how to minimize a DFA in polynomial time. In contrast, minimizing unrestricted NFAs is PSPACE-complete [33]. The minimization problem for automata with varying degrees of non-determinism was studied in a seminal paper by Jiang and Ravikumar in 1993 [25]. Among other results, they thoroughly investigated the minimization problem for UFAs. In the following discussion, we will often use the terminology: *Minimization has complexity $\mathcal{C}$ for automata class $\mathcal{A}$, even if the input is given as an automaton from class $\mathcal{B}$.* Formally, this means that the following problem has complexity $\mathcal{C}$:

Given an NFA B from $\mathcal{B}$ and an integer k in binary, does there exist an NFA A from $\mathcal{A}$ with at most k states such that $L(A) = L(B)$?

If the automata class $\mathcal{B}$ is not mentioned, we assume that it equals $\mathcal{A}$. In this terminology, Jiang and Ravikumar showed the following:

- Minimization is NP-complete for UFAs, even if the input is given as a DFA.
- Minimization is PSPACE-complete for NFAs, even if the input is given as a DFA.

Minimization problems have even been studied for automata with unary alphabets; see, e.g., [24, 15].

Recently, Malcher [26] improved on the results of Jiang and Ravikumar in the sense that he showed that finite automata with quite a small amount of non-determinism are hard to minimize. More precisely, he showed the following:

(a) Minimization is NP-complete for automata that can non-deterministically choose between a fixed number of initial states, but are otherwise deterministic.

(b) Minimization is NP-complete for non-deterministic automata with a constant number of computations for each string.[3]

[2] An automaton is unambiguous if it has at most one accepting run for each word.

[3] Actually, he showed this for automata with constant *branching*, which is slightly different from the number of computations; see Section 2.1.

Whereas Malcher made significant progress in showing that minimization is hard for non-deterministic automata, he was not yet able to settle the entire problem. Therefore, he poses the question of whether there are relaxations of the deterministic automata model *at all* for which minimization is tractable as an important open problem. In this context, he mentions the class of automata with at most two computations for each string and the two classes (a) and (b) above with the added restriction of unambiguousness as important remaining cases. We also note that Malcher used different proof techniques for the results (a) and (b) above.

Our Contributions. We improve on Malcher's results in two respects. We settle his open questions and we provide a uniform NP-hardness proof for all classes of automata mentioned above. In brief, we define a class δNFA of automata that are unambiguous, have at most two computations per string, and have at most one state q with two outgoing a-transitions, for at most one symbol a. Then, we show that minimization is NP-hard for all classes of finite automata that include the δNFAs that accept strings of length three, even if the input is a DFA. We show that these hardness results can also be adapted to the setting of unambiguous automata that can non-deterministically choose between two start states, but are deterministic everywhere else. This solves the open cases mentioned by Malcher [26].

On the other hand we show that there *are* (non-trivial) relaxations of the deterministic automaton model that allow tractable minimization. We show that, if we add to the definition of δNFAs that each word should have at most one *rejecting* computation (i.e., δNFAs that are co-unambiguous), minimization becomes tractable again. However, the minimal automata in this class are the DFAs, so it is not likely to be very useful in practice. Therefore, if P $\neq$ NP, the tractability frontier of the NFA minimization problem lies between δNFAs and co-unambiguous δNFAs, which are two classes that are extremely closely together.

Apart from the complexity results of minimizing δNFAs, we also show that δNFAs are quadratically more succinct that DFAs and that there does not exist a unique minimal δNFA for a regular language.

Further related work. A recent overview of finite automata minimization can be found in [4], while transition complexity of NFAs is surveyed in [30, 18]. Known results about the trade-off between amount of non-determinism and description complexity are surveyed in [12]. In a recent paper, Okhotin presented new results on the description complexity of UFAs over unary alphabets [28].

A detailed analysis of the complexity of Hopcroft's minimization algorithm for DFAs has recently been performed by Berstel et al. [3]. An interesting variant of the minimization problem which has been studied recently is the *hyperminimization* problem for DFAs. Here, it is allowed to construct a small DFA which is not equivalent with the input DFA on a finite number of strings [2]. It is possible to hyperminimize a DFA in time $O(n \log n)$ [8, 19].

Since the minimization problem for NFAs is hard, other flavors of minimization have been studied. A very relevant flavor in practice is bisimulation minimization [1, 29].

The problems of producing small NFAs from regular expressions has been considered in [23, 31]. This problem is challenging, since it is known that approximating minimal NFAs is a hard problem [13, 14, 16].

The descriptive complexity of regular expressions has also been studied in the literature. Here, problems of interest are the complexity of translating automata to expressions and the complexity of performing various operations on regular expressions [11, 17]. The descriptive complexity of regular expressions is also practically relevant in the context of XML schema languages. The complexity and succinctness of applying interleaving and counting operators on expressions, sometimes combined with determinism restrictions, has been investigated [5, 10, 9, 21].

2. Preliminaries

Throughout the paper, Σ denotes a finite alphabet. A *(non-deterministic) finite automaton (NFA) over* Σ is a tuple $A = (\text{States}(A), \text{Alpha}(A), \text{Rules}(A), \text{Init}(A), \text{Final}(A))$, where $\text{States}(A)$ is its finite set of states, $\text{Alpha}(A) = \Sigma$, $\text{Init}(A) \subseteq \text{States}(A)$ is its set of initial states, $\text{Final}(A) \subseteq \text{States}(A)$ is its set of final states, and $\text{Rules}(A)$ is a set of transition rules of the form $q_1 \xrightarrow{a} q_2$, where $q_1, q_2 \in \text{States}(A)$ and $a \in \Sigma$. The *size* of an automaton is $|\text{States}(A)|$, i.e., its number of states. For simplicity, we do not consider NFAs to have ε-transitions. Notice that each NFA A with ε-transitions can be transformed (in polynomial time) into an equivalent NFA B without ε-transitions such that B does not have more states than A. As our results are hardness results, our results therefore also apply for NFAs with ε-transitions.

A finite automaton is *deterministic* (a DFA) if $\text{Init}(A)$ is a singleton and, for each $q_1 \in \text{States}(A)$ and $a \in \text{Alpha}(A)$, there is at most one $q_2 \in \text{States}(A)$ such that $q_1 \xrightarrow{a} q_2 \in \text{Rules}(A)$. We can assume without loss of generality in this paper that each state in a finite automaton is reachable from an initial state. Indeed, if this is not the case, an automaton can be turned into a smaller one with this property in polynomial time.

A *run* or *computation* r of A on a word $w = a_1 \cdots a_n \in \Sigma^*$ is a string $q_0 q_1 \cdots q_m \in \text{States}(A)^*$ with $m \leq n$, such that $q_0 \in \text{Init}(A)$, for each $i = 0, \ldots, m-1$, $q_i \xrightarrow{a_{i+1}} q_{i+1} \in \text{Rules}(A)$, and either $m = n$ or there is no $q \in \text{States}(A)$ with $q_m \xrightarrow{a_{m+1}} q \in \text{Rules}(A)$. The run is *accepting* if $n = m$ and $q_n \in \text{Final}(A)$. Otherwise, the run is *rejecting*. Notice that runs always have maximal length. That is, if r is a run of A on w, there does not exist a run r' of A on w such that r is a strict prefix of r'. The *language of* A, denoted $L(A)$, is the set of words w such that there exists an accepting run of A on w. A finite automaton A is *unambiguous* if, for each string w, there exists at most one accepting run of A on w.

Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two classes of NFAs. We say that $\mathcal{N}_1 \subseteq \mathcal{N}_2$ if each automaton in $\mathcal{N}_1$ also belongs to $\mathcal{N}_2$. For example, DFA $\subseteq$ NFA.

2.1. Notions of Non-Determinism

We recall some standard measures of non-determinism in a finite automaton. For a state q and an alphabet symbol a, the *degree of non-determinism* of a pair (q, a), denoted by $\text{degree}(q, a)$, is the number k of different states $q_1, \ldots, q_k$ such that, for all $1 \leq i \leq k$, $q \xrightarrow{a} q_i \in \text{Rules}(A)$. We say that A has *degree of non-determinism* k, denoted by $\text{degree}(A) = k$, if $\text{degree}(q, a) \leq k$ for every $(q, a) \in \text{States}(A) \times \text{Alpha}(A)$, and there is at least one pair (q, a) such that $\text{degree}(q, a) = k$.

The *branching* of an automaton is intuitively defined as the maximum product of the degrees of non-determinism over states in a possible run. Formally, the branching of A on

a word $w = a_1 \cdots a_n$ is $\text{branch}_A(w) = \max\left\{ \prod_{i=1}^{n} \text{degree}(q_{i-1}, a_i) \mid q_0 \cdots q_n \text{ is a run of } A \text{ on } a_1 \cdots a_n \right\}$. The *branching of* A, denoted $\text{branch}(A)$, is $|\text{Init}(A)| \times \max\{\text{branch}_A(w) \mid w \in L(A)\}$ if this quantity is defined, and otherwise ∞.

Hromkovic et al. [22] define three measures of non-determinism for a finite automaton A: *advice*(A), *computations*(A), and *ambig*(A). These measures are defined as follows: $\text{advice}(A)$ is the maximum number of non-deterministic choices during any computation of A (i.e., the number of advice bits that would be needed in advance to make a computation deterministic), $\text{computations}(A)$ is the maximum number of different computations of A on any word,[4] and $\text{ambig}(A)$ is the maximum number of different *accepting* computations of A on any word. For the formal definitions of these concepts, we refer to [22].

2.2. A Notion of Very Little Non-Determinism

Next we define the notion of a δNFA. The intuition is that such an automaton should allow only a very small amount of non-determinism. Since we are interested in non-trivial extensions of the class of DFAs, each DFA is also a δNFA. In particular, this means that δNFAs do not have any restrictions on the alphabet they use.

Definition 1. *A* δNFA *is an NFA* A *with the following properties*

- A *has a single initial state;*
- A *is unambiguous;*
- *branch*$(A) \leq 2$*; and*
- *there is at most one pair* (q, a) *such that degree*$(q, a) = 2$*.*

For δNFAs, we have that $\text{degree}(A) \leq 2$, $\text{advice}(A) \leq 1$, $\text{computations}(A) \leq 2$, and $\text{ambig}(A) = 1$. Notice that any one of $\text{degree}(A) = 1$, $\text{advice}(A) = 0$, or $\text{computations}(A) = 1$ implies that A is deterministic. Also, $\text{ambig}(A) = 1$ is the minimum value possible for any automaton that accepts at least one string.

2.3. The Minimization Problem

We define the minimization problem in two flavors. Let $\mathcal{A}$ and $\mathcal{B}$ be two classes of finite automata. The *minimization problem for* $\mathcal{A}$ is then defined as:

Given an NFA A from class $\mathcal{A}$ and an integer k in binary.
Does there exist an NFA B from class $\mathcal{A}$ with at most k states such that $L(A) = L(B)$?

The *minimization problem for* $\mathcal{B}$*, even if the input is given as* $\mathcal{A}$ is defined as:

Given an NFA A from class $\mathcal{A}$ and an integer k in binary.
Does there exist an NFA B from class $\mathcal{B}$ with at most k states such that $L(A) = L(B)$?

[4] Hromkovic et al. wrote $\text{leaf}(A)$ instead of $\text{computations}(A)$.

For brevity and following Jiang and Ravikumar [25], we sometimes also refer to the minimization problem for $\mathcal{B}$, even if the input is given as $\mathcal{A}$, as the $\mathcal{A} \to \mathcal{B}$ *minimization problem*.

Let $\mathcal{N}_1$, $\mathcal{N}_2$, and $\mathcal{N}_3$ be classes of finite automata. Suppose that the $\mathcal{N}_1 \to \mathcal{N}_2$ minimization problem is hard for a complexity class $\mathcal{C}$, and let $\mathcal{N}_3$ be a class of automata such that $\mathcal{N}_1 \subseteq \mathcal{N}_3$. Then the $\mathcal{N}_3 \to \mathcal{N}_2$ minimization problem is also trivially hard for $\mathcal{C}$. However, assuming that $\mathcal{N}_1 \to \mathcal{N}_2$ is hard for $\mathcal{C}$ and that $\mathcal{N}_2 \subseteq \mathcal{N}_3$, there is, as far as we know, no general argument that also makes the $\mathcal{N}_1 \to \mathcal{N}_3$ minimization problem hard for $\mathcal{C}$, as finding a small $\mathcal{N}_3$ automaton might be easier than finding a small $\mathcal{N}_2$ automaton in general.[5] Therefore, we will prove directly that minimization is NP-hard for *all* classes of automata between δNFAs and NFAs.

2.4. The Tractability Frontier

In this section, we discuss what we know about the tractability frontier for the NFA minimization problem, if PTIME $\neq$ NP. We also discuss the proximity between the class of δNFAs and the DFAs.

We first note that there are in fact *two* incomparable notions of determinism for finite automata: determinism and *reverse* determinism. (An automaton A is reverse deterministic if its automaton with the inverted transitions is deterministic, i.e., if the NFA obtained by replacing every rule $q_1 \xrightarrow{a} q_2$ by $q_2 \xrightarrow{a} q_1$ is deterministic.) Both deterministic and reverse deterministic finite automata can be efficiently minimized by the same algorithm, modulo a simple pre- and post-processing step for reverse deterministic automata. In other words, both the DFA $\to$ DFA minimization problem and the symmetric problem for reverse deterministic finite automata are solvable in polynomial time. We view these two classes as the two possible "optima" in the spectrum of determinism, as they arise very naturally from the fact that one can either read strings from left to right or from right to left. From now on, we only consider the proximity of δNFAs to (left-to-right deterministic) DFAs.

We will prove in Section 3 that, for every class $\mathcal{A}$ of finite automata that includes the δNFAs, the minimization problem is NP-hard, even if the input is given as a DFA and even if the number k in the input is given in unary encoding. (That is, we will show that minimization is strongly NP-hard.) What does this result tell us about the tractability frontier of minimization?

Of course, the class of δNFAs is not the smallest class of automata that include the DFAs and allow non-determinism. One could, for instance, take the class of DFAs and add a finite set N of NFAs. For each such class, minimization would be in PTIME. Indeed, for each of the constantly many possible NFAs in N as input, the minimization algorithm can decide the minimization question in constant time, after having read the input. For each of the DFAs, the algorithm can test, in a first phase, whether the input DFA accepts one of the constantly many languages represented by an NFA. If so, it can, in the second phase, solve the minimization question analogously as for this NFA, and if not, it can, in the second phase, solve the minimization question using standard methods for DFA minimization.

[5] This is also why, e.g., Malcher explicitly proves NP-hardness for minimizing various classes of automata that are included in one another (Lemmas 3 and 11 in [26]).

Classes obtained by adding constantly many NFAs are, of course, not very interesting. Let's consider a class that adds infinitely many NFAs. Define the class cu-δNFA to be the class of δNFAs with the additional condition that they are also *co-unambiguous*. That is, for each word w, there can be at most one *rejecting* computation of A on w. Recall that we defined runs to have a maximal possible length. This means that, if r is a rejecting run of A on w, then the strict prefixes of r are not runs of A on w. In particular, this means that there are non-trivial co-unambiguous δNFAs. Combined with the conditions on δNFAs this implies that for each w, there can still be two runs, but if so, one must be accepting and the other one must be rejecting. This notion of non-determinism lies strictly between DFAs and δNFAs (that is, DFA $\subsetneq$ cu-δNFA $\subsetneq$ δNFA).

Consider the minimization problem for cu-δNFA and let A be an arbitrary cu-δNFA. We will argue that the minimal cu-δNFA for $L(A)$ is a DFA. Suppose that A is not a DFA. Let q and a be the unique state and label such that $\text{degree}(q, a) = 2$. Let q_1 and q_2 be the two states such that $q \xrightarrow{a} q_1$ and $q \xrightarrow{a} q_2$ are in $\text{Rules}(A)$. Let w be an arbitrary string that leads A to state q and let w' be an arbitrary string over alphabet $\text{Alpha}(A)$. Then there are two different runs of A on the string waw', one which reaches q_1 after reading the prefix wa and one which reaches q_2 after reading the prefix wa. By definition of cu-δNFA, one of these runs has to be rejecting and one has to be accepting. Since w' was arbitrary, this implies that A must accept every string of the form waw'. This means that we can make A strictly smaller by merging the two states q_1, q_2 into one state q_3, removing all outgoing transitions from q_3, making q_3 a final state, and adding loop transitions from q_3 to itself for each alphabet symbol. Moreover, by this operation, A becomes deterministic. Hence, every automaton A in cu-δNFA that is not a DFA can be rewritten as a smaller DFA. This means that, in the class cu-δNFA, the minimal automata are DFAs. In particular, this also puts the minimization problem for cu-δNFA into PTIME.

From the above it is clear that δNFAs are certainly not the closest possible to determinism that one can get. Rather, it is the closest class to DFAs we were able to find that takes advantage of the succinctness of nondeterminism in a nontrivial way.

Our NP-hardness result for the minimization of δNFAs therefore puts the tractability frontier precisely between δNFAs and the above mentioned class cu-δNFA; two classes that are very close to one another.

3. Minimizing Non-Deterministic Automata is Hard

The main result of this section is the following.

Theorem 2. *Let $\mathcal{N}$ be a class of finite automata such that $\delta NFA \subseteq \mathcal{N}$. Then the minimization problem for $\mathcal{N}$ is strongly NP-hard, even if the input is given as a DFA.*

Corollary 3. *Let $\mathcal{N}$ be a class of finite automata such that $\delta NFA \subseteq \mathcal{N}$. Then the minimization problem for $\mathcal{N}$ is strongly NP-hard.*

We start by formally defining the decision problems that are of interest to us, and then sketch an intuitive overview of our proof. Given an undirected graph $G = (V, E)$ such that V is its set of vertices and $E \subseteq V \times V$ is its set of edges, we say that a set of vertices $VC \subseteq V$ is a *vertex cover* of G if, for every edge $(v_1, v_2) \in E$, VC contains v_1, v_2, or both.

If B and C are finite collections of finite sets, we say that B is a *set basis* for C if, for each $c \in C$, there is a subcollection B_c of B whose union is c. We say that B is a *normal set basis* for C if, for each $c \in C$, there is a *pairwise disjoint* subcollection B_c of B whose union is c. We say that B is a *separable normal set basis* for C if B is a normal set basis for C and B can be written as a disjoint union $B_1 \uplus B_2$ such that, for each $c \in C$, the subcollection B_c of B contains at most one element from B_1 and at most one from B_2.

The following decision problems are considered in this paper. *Vertex Cover* asks, given a pair (G, k) where G is a graph and k is an integer given in binary, whether there exists a vertex cover of G of size at most k. It is well-known that Vertex Cover is strongly NP-complete, that is, Vertex Cover remains NP-hard even if k is given as a unary number [7]. *Set Basis*, *Normal Set Basis*, and *Separable Normal Set Basis* ask, given a pair (C, s) where C is a finite collection of finite sets and s is an integer, whether there exists a set basis, resp., normal set basis, resp., separable normal set basis for C containing at most s sets.

The proof of Theorem 2 proceeds in several steps. First, we provide a slightly modified version of a known reduction from Vertex Cover to Normal Set Basis (Lemma 4 in [25]), showing that the latter problem is NP-hard. Second, we proceed to show that the set $\mathbf{I}$ of instances of Normal Set Basis obtained through this reduction has a number of interesting properties (Lemma 5). In particular, we show that if such an instance has a set basis of a certain size s, then it also has a *normal* set basis of size s. Third, we show that the the Normal Set Basis problem, for instances in $\mathbf{I}$ reduces to minimization for δNFAs (Lemma 6).

The statement of Theorem 2 says that given a DFA, finding the minimal equivalent automaton in class $\mathcal{N}$ is NP-hard, for any class of finite automata that contains the δNFAs. As argued in Section 2.3, using a DFA instead of a δNFA as input of the problem strengthens the statement. Also, showing that DFA $\to \delta$NFA is NP-hard doesn't immediately imply that DFA $\to \mathcal{N}$ is hard for every $\mathcal{N}$ that contains all δNFAs. To show that this is actually the case, we prove that for the languages obtained in our reduction, the minimal NFAs are precisely one state smaller than the minimal δNFAs (Lemma 6). For these languages, the minimization problem for δNFAs and for NFAs is essentially the same problem.

We revisit a slightly modified reduction which is due to Jiang and Ravikumar [25], as our further results rely on a construction in their proof. We also add the observation of *strong* NP-completeness.

Lemma 4 (Jiang and Ravikumar [25]). *Normal Set Basis is strongly NP-complete.*

Proof. Obviously, Normal Set Basis is in NP. Indeed, given an input (C, s) for Normal Set Basis, if s is at least the number of elements in the union of all sets from C, then the algorithm can return true. Otherwise, the NP algorithm simply guesses a collection B containing at most s sets, guesses the subcollections B_c for each $c \in C$, and verifies whether the sets B_c satisfy the necessary conditions.

In order to prove strong NP-hardness, we give a reduction from Vertex Cover to Normal Set Basis. It is known that Vertex Cover is strongly NP-hard [7]. Given an input (G, k) of Vertex Cover, where $G = (V, E)$ is a graph and k is an integer, we construct in LOGSPACE an input (C, s) of Normal Set Basis, where C is a finite collection of finite sets and s is an integer. In particular, (C, s) is constructed such that G has a vertex cover of size at most k if and only if C has a normal set basis containing at most s sets.

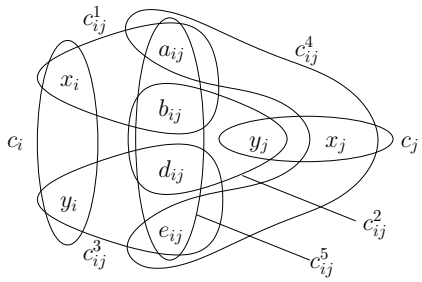

Figure 1: The constructed sets $c_i, c_j, c_{ij}^1, \ldots, c_{ij}^5$ in the proof of Lemma 4.

For a technical reason which will become clear in later proofs, we assume without loss of generality that $k < |E| - 3$. Notice that, under this restriction, Vertex Cover is still NP-complete under LOGSPACE reductions. This can be seen by combining two observations. First of all, we can assume that $|V| \leq |E|$. This is so because Vertex Cover is still NP-complete under LOGSPACE reductions if the input graph is connected and contains at least one cycle. In fact, the proof from Garey and Johnson ([7], Theorem 3.3) showing that Vertex Cover is NP-hard by reduction from 3-SAT produces graphs with this property. Second, if $k \geq |V| - 3$, then Vertex Cover can be solved in LOGSPACE by testing all possibilities of the at most 3 vertices which are not in the vertex cover, and verifying that there does not exist an edge between 2 of these 3 vertices. Combined, these two observations show that we can assume that $k < |E| - 3$.

Formally, let $V = \{v_1, \ldots, v_n\}$. For each $i = 1, \ldots, n$, define c_i to be the set $\{x_i, y_i\}$ which intuitively corresponds to the node v_i. Let (v_i, v_j) be in E with $i < j$. To each such edge we associate five sets as follows:

$$\begin{array}{ll} c_{ij}^1 := \{x_i, a_{ij}, b_{ij}\}, & c_{ij}^4 := \{x_j, a_{ij}, e_{ij}\}, \text{ and} \\ c_{ij}^2 := \{y_j, b_{ij}, d_{ij}\}, & c_{ij}^5 := \{a_{ij}, b_{ij}, d_{ij}, e_{ij}\}. \\ c_{ij}^3 := \{y_i, d_{ij}, e_{ij}\}, & \end{array}$$

Figure 1 contains a graphical representation of the constructed sets $c_i, c_j, c_{ij}^1, \ldots, c_{ij}^5$ for some $(v_i, v_j) \in E$. Then, define

$$C := \{c_i \mid 1 \leq i \leq n\} \cup \{c_{ij}^t \mid (v_i, v_j) \in E, i < j, \text{ and } 1 \leq t \leq 5\}$$

and $s := n + 4|E| + k$. Notice that the collection C contains $n + 5|E|$ sets. Obviously, C and s can be constructed from G and k in LOGSPACE. For strong NP-hardness, observe that, if k is given as a unary number, then a unary representation of s can also be constructed in LOGSPACE.

We show that the reduction is correct, that is, that G has a vertex cover of size at most k if and only if C has a (separable) normal set basis containing at most s sets.

($\Rightarrow$): Let G have a vertex cover VC of size k. We need to show that C has a normal set basis B containing at most $s = n + 4|E| + k$ sets.

To this end, we define a collection B of sets as follows. For every $v_i \in V$,

- if $v_i \in VC$, we include both $\{x_i\}$ and $\{y_i\}$ in B;

- otherwise, we include $c_i = \{x_i, y_i\}$ in B.

The number of sets included in B so far is $2k + (n-k) = k+n$. Let $e = (v_i, v_j)$ (where $i < j$) be an arbitrary edge in G. Since VC is a vertex cover, either v_i or v_j (or both) is in VC. When v_i is in VC, we additionally include the sets

$$r^1_{ij} := \{a_{ij}, b_{ij}\}, \qquad r^2_{ij} := \{d_{ij}, e_{ij}\},$$
$$r^3_{ij} := \{y_j, b_{ij}, d_{ij}\}, \text{ and} \quad r^4_{ij} := \{x_j, a_{ij}, e_{ij}\}$$

in B. When v_i is not in VC, we additionally include the sets

$$r^5_{ij} := \{a_{ij}, e_{ij}\}, \qquad r^6_{ij} := \{b_{ij}, d_{ij}\},$$
$$r^7_{ij} := \{x_i, a_{ij}, b_{ij}\}, \text{ and} \quad r^8_{ij} := \{y_i, d_{ij}, e_{ij}\}$$

in B. This completes the definition of B. Notice that, when $v_i \in VC$, c^1_{ij}, c^3_{ij}, and c^5_{ij} can be expressed as a disjoint union of members of B as

$$c^1_{ij} = \{x_i\} \uplus r^1_{ij}, \qquad c^3_{ij} = \{y_i\} \uplus r^2_{ij}, \qquad c^5_{ij} = r^1_{ij} \uplus r^2_{ij}$$

and that $c^2_{ij} = r^3_{ij}$ and $c^4_{ij} = r^4_{ij}$ are members of B. Analogously, when $v_i \notin VC$, c^2_{ij}, c^4_{ij}, and c^5_{ij} can be expressed as a disjoint union of members of B as

$$c^2_{ij} = \{y_j\} \uplus r^6_{ij}, \qquad c^4_{ij} = \{x_j\} \uplus r^5_{ij}, \qquad c^5_{ij} = r^5_{ij} \uplus r^6_{ij}$$

and $c^1_{ij} = r^7_{ij}$ and $c^3_{ij} = r^8_{ij}$ are members of B. Since the total number of sets included in B for each edge is four, B contains $(k+n) + 4|E| = s$ sets. From the above argument it is also obvious that B is a normal set basis for C.

Notice that B is in fact a separable normal set basis for C. Indeed, we can partition B into the sets

$$\begin{aligned} B_1 = \{\{x_i\}, \{x_j, y_j\} \mid v_i \in VC, v_j \notin VC\} \\ \cup \{r^2_{ij}, r^3_{ij} \mid (v_i, v_j) \in E, i < j, v_i \in VC\} \\ \cup \{r^6_{ij}, r^7_{ij} \mid (v_i, v_j) \in E, i < j, v_i \notin VC\} \end{aligned}$$

and

$$\begin{aligned} B_2 = \{\{y_i\} \mid v_i \in VC\} \\ \cup \{r^1_{ij}, r^4_{ij} \mid (v_i, v_j) \in E, i < j, v_i \in VC\} \\ \cup \{r^5_{ij}, r^8_{ij} \mid (v_i, v_j) \in E, i < j, v_i \notin VC\}, \end{aligned}$$

which satisfy the necessary condition.

($\Leftarrow$): Suppose that C has a normal set basis B containing at most $s = n + 4|E| + k$ sets. We can assume without loss of generality that no proper subcollection of B is a normal set basis. We show that G has a vertex cover VC of size at most k. Define $VC = \{v_i \mid \text{both } \{x_i\} \text{ and } \{y_i\} \text{ are in } B\}$. Let k' be the number of elements in VC. The number of sets in B consisting of only x_i and/or y_i is at least $n + k'$. This can be seen from the fact that B must have the subset c_i for all i such that $v_i \notin VC$. Thus, there are $n - k'$ such sets in addition to $2k'$ singleton sets corresponding to i's such that $v_i \in VC$.

Let $E' \subseteq E$ be the set of edges covered by VC, that is, $E' = \{(v_i, v_j) \mid v_i \text{ or } v_j \text{ is in } VC\}$. The following observation can easily be shown (by checking all possibilities):

Observation: For any $e \in E'$ at least four sets of B (excluding sets $c_i, c_j, \{x_i\}$, $\{y_i\}, \{x_j\}$, or $\{x_j\}$) are necessary to be a normal set basis for the five sets c_{ij}^t, $t = 1, \ldots, 5$. Further, at least five sets (excluding sets $c_i, c_j, \{x_i\}, \{y_i\}, \{x_j\}$, or $\{x_j\}$) are required to be a normal set basis for them if $e \notin E'$. Notice that if $e \notin E'$, then either $\{x_i\} \notin B$ or $\{y_i\} \notin B$ and, furthermore, either $\{x_j\} \notin B$ or $\{y_j\} \notin B$.

Now the total number of sets needed to cover C is at least $n+k'+4|E'|+5(|E|-|E'|)$, which we know is at most $s = n + 4|E| + k$. Hence, we obtain that $n + k' + 5|E| - |E'| \leq n + 4|E| + k$, which implies that $k' + |E| - |E'| \leq k$. We conclude the proof by showing that there is a vertex cover VC' of size $|E| - |E'| + k'$. Add one of the end vertices of each edge $e \in E - E'$ to VC. This vertex cover is of size $|E| - |E'| + k' \leq k$. □

The next lemma now follows from the proof of Lemma 4. It defines a set of inputs $\mathbf{I}$ for which Normal Set Basis remains strongly NP-complete and further shows that for any $(C, s) \in \mathbf{I}$, the collection C has a set basis of size s if and only if C also has a separable normal set basis of size s. Of course, the latter property does not hold for the set of all possible inputs for the normal set basis problem.

Lemma 5. *There exists a set of inputs $\boldsymbol{I}$ for Normal Set Basis, such that*

(1) Normal Set Basis is strongly NP-complete for inputs in $\boldsymbol{I}$;

(2) for each (C, s) in $\boldsymbol{I}$, C contains every set at most once and $s < |C| - 3$;

(3) for each $(C, s) \in \boldsymbol{I}$, the following are equivalent:

- *(a) C has a set basis containing at most s sets.*
- *(b) C has a separable normal set basis containing at most s sets.*

(4) for each (C, s) in $\boldsymbol{I}$, each solution B for (C, s) writes at least two sets of C as a union of at least two sets in B.

Proof. The set $\mathbf{I}$ is obtained by applying the reduction in Lemma 4 to inputs (G, k) of Vertex Cover such that $k < |E| - 3$. This immediately shows (1) and (2). We continue by proving the other parts of the claim.
(3) The direction from (b) to (a) is trivial. For the other direction, notice that we showed in the proof of Lemma 4 that, if C has a *normal* set basis containing at most s sets, then G has a vertex cover of size at most k. We also showed that, if G has a vertex cover of size at most k, then C has a *separable* normal set basis containing at most s sets. Together, this implies that, if C has a normal set basis containing at most s sets, it also has a separable normal set basis containing at most s sets.

Hence, we only need to prove that, if C has a set basis of at most s sets, then C also has a normal set basis containing at most s sets. To this end, let (C, s) be an instance in $\mathbf{I}$, i.e., there is an $n \in \mathbb{N}$ and $E \subseteq \{(i, j) \mid 1 \leq i < j \leq n\}$ such that $C = \{c_i \mid 1 \leq i \leq n\} \cup \{c_{ij}^r \mid (i, j) \in E \wedge 1 \leq r \leq 5\}$, and suppose C has a set basis $B = \{b_1, \ldots, b_s\}$ of size s. We construct a *normal* set basis for C of size s.

To this end, we will show a sequence of assumptions that we can make about B without loss of generality. Put together, these assumptions will imply that B is a normal

set basis for C. Therefore, they thus show that if there is a set basis of size s, there is also a normal set basis of size s. Throughout the proof, it will be helpful for the reader to keep an eye on Figure 1.

Suppose that there is an i such that B contains both $\{x_i\}$ and $\{x_i, y_i\}$. Then we can replace $\{x_i, y_i\}$ with $\{y_i\}$ and still have a set basis for C of size s, since c_i is the only set in C of which $\{x_i, y_i\}$ is a subset. This gives us our first assumption.

Assumption 1. For every $i \in \{1, \ldots, n\}$, set basis B does not contain both $\{x_i, y_i\}$ and $\{x_i\}$ or, symmetrically, B does not contain both $\{x_i, y_i\}$ and $\{y_i\}$.

Suppose there are $1 \leq i < j \leq n$ and $1 \leq r \leq 4$ such that c^r_{ij} cannot be formed as a disjoint union of sets from B. We now show how to change B such that it can form c^r_{ij} as a disjoint union. We will give the argument for $r = 1$, i.e., $c^r_{ij} = c^1_{ij} = \{x_i, a_{ij}, b_{ij}\}$. The three other cases are completely analogous. Since there are no disjoint sets from B whose union is c^1_{ij}, there must be two different sets b^1 and b^2 in B that are subsets of c^1_{ij} and contain precisely two elements each. At least one of these subsets must contain x_i. Assume w.l.o.g. that this set is b^1. No subset of size two of c^1_{ij} that contains x_i is a subset of any set of C other than c^1_{ij}. This means that we can replace b^1 with $b^1 \setminus b^2$ in B and still have a set basis of C of size at most s. This gives us our second assumption.

Assumption 2. For any i, j and any $r \in \{1, \ldots, 4\}$, the set c^r_{ij} can be formed as a union of disjoint sets from B.

If B satisfies Assumptions 1 and 2, but is not a normal set basis, then there are $1 \leq i < j \leq n$ such that c^5_{ij} cannot be formed as a disjoint union of sets from B. In particular, this means that $c^5_{ij} \notin B$. Let B^5_{ij} be a subset of B such that the union of the sets in B^5_{ij} is c^5_{ij}. We can assume that B^5_{ij} is inclusion free, i.e., there are no two sets in B^5_{ij} such that one is a subset of the other, since if there are $b^1, b^2 \in B^5_{ij}$ with $b^1 \subseteq b^2$, we can replace b^2 with $b^2 \setminus b^1$.

Assumption 3. All collections B^5_{ij} are inclusion free.

If B^5_{ij} has four members, then we can replace B^5_{ij} with the collection containing the four singletons $\{a_{ij}\}, \{b_{ij}\}, \{d_{ij}\}$, and $\{e_{ij}\}$ without increasing the size of B and we would be able to write c^5_{ij} as a disjoint union. Therefore, the only case in which we still cannot write c^5_{ij} as a disjoint union is the case where the collection B^5_{ij} has at most three members.

Assumption 4. All collections B^5_{ij} have at most three members.

Suppose there is a set b in B^5_{ij} such that b is not a subset of any set in C other than c^5_{ij}. Then we can replace b with $b \setminus (\bigcup_{b' \in B^5_{ij} \setminus \{b\}} b')$ in B and still have a set basis of size at most s.

Assumption 5. Each member of B^5_{ij} is a subset of some set from C other that c^5_{ij}. In particular, since no set in C, other than c^5_{ij} itself, has an intersection with c^5_{ij} of size larger than two, this means that each member of B^5_{ij} has at most two elements.

If we take three different subsets of c^5_{ij} with at most two elements, that are also subsets of other sets from C than c^5_{ij}, then at least two of them are disjoint; see Figure 1. Let these two disjoint sets be b^1 and b^2. If b^1 and b^2 both contain two elements, we can replace B^5_{ij} with $\{b^1, b^2\}$ and we would be able to write c^5_{ij} as a disjoint union. Therefore, the only case in which we still cannot write c^5_{ij} as a disjoint union is the case summarized in Assumption 6.

Assumption 6. There are at most two sets in B^5_{ij} with two elements. Furthermore, these two sets have a non-empty intersection. This means that B^5_{ij} must have exactly three members, two with two elements and one singleton. Without loss of generality, we may assume that $B^5_{ij} = \{\{a_{ij}, b_{ij}\}, \{b_{ij}, d_{ij}\}, \{e_{ij}\}\}$. (All other cases are symmetrical.) We may also assume that neither $\{a_{ij}\}$ nor $\{b_{ij}\}$ belong to B. (If $\{a_{ij}\}$ belongs to B then we can replace $\{a_{ij}, b_{ij}\}$ by $\{b_{ij}\}$ in B^5_{ij}, and if $\{b_{ij}\}$ belongs to B we can replace $\{b_{ij}, d_{ij}\}$ by $\{d_{ij}\}$ in B^5_{ij}.)

In order to form c^1_{ij} either $\{x_i\}$, $\{x_i, a_{ij}\}$, $\{x_i, b_{ij}\}$, or $\{x_i, a_{ij}, b_{ij}\}$ must be a member of B. Since B^5_{ij} satisfies Assumption 6, we can replace this member of B with $\{x_i\}$, since none of the other sets is a subset of any other set in C than c^1_{ij}. But if $\{x_i\} \in B$ we can, analogously to Assumption 1, also assume that $\{y_i\} \in B$. To form c^3_{ij}, B must, apart from $\{y_i\}$ and $\{e_{ij}\}$, contain some subset of c^3_{ij} that contains d_{ij}. Since we have both $\{y_i\}$ and $\{e_{ij}\}$ in B, we may replace this subset with $\{d_{ij}\}$, since all other subsets of c^3_{ij} are no subsets of other elements of C. Once we have $\{d_{ij}\}$ in B, we can replace $\{b_{ij}, d_{ij}\}$ by $\{d_{ij}\}$ in B^5_{ij} and we can write c^5_{ij} as a disjoint union.

Assumption 7. There are disjoint members of B whose union is c^5_{ij}.

Together, Assumptions 1, 2, and 7 imply that B is in fact a normal set basis for C. Since each assumption was made without loss of generality, we have shown that from any set basis for C of size s we can form a *normal* set basis for C of size at most s.

(4) We simply observe that a normal set basis writing at most one set of C as a union of at least two sets must contain at least $|C|$ sets, and hence cannot be a solution for (C, s), since $s < |C|$. □

The proof of the following lemma is partly inspired by the proof of Theorem 3.1 of [25], but we significantly strengthen it for our purposes.

Lemma 6. *There exists a set of regular languages $\mathcal{L}$ such that*

(1) the minimization problem for δNFAs accepting a language from $\mathcal{L}$ is strongly NP-complete, even if the input is given as a DFA and

(2) for each $L \in \mathcal{L}$, the size of the minimal NFA for L is equal to the size of the minimal δNFA for L, minus 1.

Proof. The NP upper bound is immediate, as equivalence testing for unambiguous finite automata is in PTIME [32]. An NP algorithm can therefore guess a δNFA of sufficiently small size and test in PTIME whether it is equivalent to the given DFA.

For the lower bound, we reduce from Separable Normal Set Basis. To this end, let (C, s) be an input of Separable Normal Set Basis. Hence, C is a collection of n sets and s is an integer. According to Lemma 5, we can assume without loss of generality that $(C, s) \in \mathbf{I}$, that is, C has a separable normal set basis containing s sets if and only if C has a normal set basis of size s. Moreover, we can assume that $s < n - 3$ and that s is given in unary.

We construct in LOGSPACE a DFA A and an integer ℓ such that the following are equivalent:

- C has a separable normal set basis of size at most s.
- There exists a δNFA N_δ for $L(A)$ of size at most ℓ.
- There exists an NFA N for $L(A)$ of size at most $\ell - 1$.

The DFA A accepts the language $\{acb \mid c \in C \text{ and } b \in c\}$, which is a finite language of strings of length three.

Formally, let $C = \{c_1, \ldots, c_n\}$ and $c_i = \{b_{i,1}, \ldots, b_{i,n_i}\}$ for every $i = 1, \ldots, n$. Then, A is defined over $\text{Alpha}(A) = \{a\} \cup \bigcup_{1 \leq i \leq n} \{c_i, b_{i,1}, \ldots, b_{i,n_i}\}$. The state set of A is $\text{States}(A) = \{q_0, q_0', q_1, \ldots, q_n, q_f\}$, and the initial and final state sets of A are $\{q_0\}$ and $\{q_f\}$, respectively. The transitions $\text{Rules}(A)$ are formally defined as follows:

- $q_0 \xrightarrow{a} q_0' \in \text{Rules}(A)$;
- for every $i = 1, \ldots, n$, $q_0' \xrightarrow{c_i} q_i \in \text{Rules}(A)$; and
- for every $i = 1, \ldots, n$ and $j = 1, \ldots, n_i$, $q_i \xrightarrow{b_{i,j}} q_f \in \text{Rules}(A)$.

Finally, define $\ell := s + 4$. Notice that $|A| = n + 3$.

Obviously, A and ℓ can be constructed from C and s using logarithmic space, even if s and ℓ are in unary. Observe that due to Lemma 5, C contains every set at most once, and hence does not contain two different sets c_i and c_j with different name (i.e., $i \neq j$) but with the same elements. Therefore, A is a minimal DFA for $L(A)$.

Recall that, in this paper, the size of an NFA is defined to be its number of states. In this terminology, we now show that,

(a) if C has a separable normal set basis containing at most s sets, then there exists a δNFA N_δ for $L(A)$ of size at most ℓ and an NFA N for $L(A)$ of size at most $\ell - 1$;

(b) if there exists a δNFA N_δ for $L(A)$ of size at most ℓ then C has a separable normal set basis containing at most s sets; and

(c) if there exists an NFA N for $L(A)$ of size at most $\ell - 1$ then C has a separable normal set basis containing at most s sets.

(a) Assume that C has a separable normal set basis containing s sets. We construct a δNFA N_δ for $L(A)$ of size at most $\ell = s + 4$.

Let $B = \{r_1, \ldots, r_s\}$ be the separable normal set basis for C containing at most s sets. Without loss of generality, we can assume that B does not contain duplicate sets. Also, let B_1 and B_2 be disjoint subcollections of B such that each element of C is either

an element of B_1, an element of B_2, or a disjoint union of an element of B_1 and an element of B_2.

To describe N_δ, we first fix the representation of each set c in C as a disjoint union of at most one set in B_1 and at most one set in B_2. Say that each basic member of B in this representation *belongs to* c.

We define the state set of N_δ as $\text{States}(N_\delta) = \{q_0, q_1, q_2, q_f\} \cup \{r_i \in B_1\} \cup \{r_i \in B_2\}$. The transition rules of N_δ are defined as follows. First, $\text{Rules}(N_\delta)$ contains the non-deterministic transitions $q_0 \xrightarrow{a} q_1$ and $q_0 \xrightarrow{a} q_2$. Furthermore, for every $i = 1, \ldots, n$, $j = 1, \ldots, s$, and $m = 1, 2$, $\text{Rules}(N_\delta)$ contains the rule

- $q_m \xrightarrow{c_i} r_j$, if $r_j \in B_m$ and r_j belongs to c_i; and
- $r_j \xrightarrow{b} q_f$, if $b \in r_j$.

Notice that the size of N_δ is $|B| + 4 = s + 4 = \ell$. By construction, we have that $L(N_\delta) = L(A)$.

We argue that N_δ is a δNFA. First, we argue that the only non-determinism in N_δ is in the transitions $q_0 \xrightarrow{a} q_1$ and $q_0 \xrightarrow{a} q_2$. By construction, all transitions going to q_f are deterministic. It is also easy to see that all outgoing transitions from q_1 are deterministic because, if we assume that N_δ contains transitions of the form $q_1 \xrightarrow{c_i} r_j$ and $q_1 \xrightarrow{c_i} r_{j'}$ with $j \neq j'$, this would mean that both r_j and $r_{j'}$ belong to c_i, which contradicts the separable normal basis property for B_1. The argument for q_2 is analogous.

Next, we show that N_δ is unambiguous. Towards a contradiction, assume that there is an $i = 1, \ldots, n$ and a $b \in c_i$ such that the string ac_ib has two accepting runs. Since the only non-deterministic transitions of A are from q_0 to q_1 and q_2, the only way in which this can happen is that one run visits state q_1 and the other run visits state q_2. Let r_{j_1} (respectively, r_{j_2}) be the state such that $q_1 \xrightarrow{c_i} r_{j_1}$ (respectively, $q_2 \xrightarrow{c_i} r_{j_2}$) are transitions in N_δ. By construction of N_δ and since B does not contain duplicate sets, we have that $j_1 \neq j_2$. But this means that both r_{j_1} and r_{j_2} belong to c_i, and their intersection contains the element b, which contradicts the disjointness condition of the normal set basis B.

Finally, the NFA N for $L(A)$ is obtained by merging the two states q_1 and q_2 from N_δ. Since the size of N_δ is ℓ, the size of N is $\ell - 1$.

(b) Assume that $L(A)$ can be accepted by a δNFA N_δ of size at most ℓ. We can assume that N_δ is minimal. We need to show that there exists a separable normal set basis for C containing at most $s = \ell - 4$ sets.

Recall that we assumed that $s < n - 3$ in our reduction. Hence, we have that $\ell = s + 4 < n + 1 < |A|$. As we observed that A is a minimal DFA for $L(A)$, it must be the case that N_δ has at least one non-deterministic transition.

Notice that, as N_δ is minimal and accepts only strings of length three, we can assume that N_δ has a unique final state. Furthermore, since N is a δNFA, it also has a unique initial state. Let q_0 and q_f be the initial and final state of N_δ, respectively. We can partition N_δ's states into four sets $\mathcal{Q}_0$, $\mathcal{Q}_1$, $\mathcal{Q}_2$, and $\mathcal{Q}_3$ such that, for each $0 \leq i \leq 3$, $\mathcal{Q}_i$ is precisely the set of states that N_δ can be in after having read a string of length i. We already know that $\mathcal{Q}_0 = \{q_0\}$ and $\mathcal{Q}_3 = \{q_f\}$. For each state $q \in \mathcal{Q}_2$, define a set $B_q = \{b \mid q \xrightarrow{b} q_f \in \text{Rules}(N_\delta)\}$.

Next, we show that the collection $B = \{B_q \mid q \in \mathcal{Q}_2\}$ is a normal set basis for C. By definition of $L(A)$, we have that every $c \in C$ is the union of $B_c := \{B_q \mid \exists p \in \mathcal{Q}_1 : p \xrightarrow{c} q \in \text{Rules}(M_i)\}$. It remains to show that B_c is also a disjoint subcollection of B. When B_c contains only one set, there is nothing to prove. Towards a contradiction, assume that B_c contains two different sets B_{q_1} and B_{q_2} such that $b \in B_{q_1} \cap B_{q_2}$. By definition of B, this would mean that the string acb has two accepting runs: $q_0 q q_1 q_f$ and $q_0 q' q_2 q_f$ with $q_1 \neq q_2$. But as N_δ is unambiguous, this is impossible. Hence, B_c is a disjoint subcollection of B.

Finally, we want to prove that B contains at most $\ell - 4$ sets. Since the number of sets in B equals the number of states in $\mathcal{Q}_2$, we know that B contains at most $\ell - 4$ sets if and only if $\mathcal{Q}_1$ contains at least two states. Towards a contradiction, assume that $\mathcal{Q}_1$ is a singleton q. By Lemma 5(4), we know that there are at least two sets c_1, c_2 in C such that B_{c_1} and B_{c_2} contain two sets. By definition of B, this would mean that there are at least two alphabet symbols c_1 and c_2 that have non-deterministic outgoing transitions from q, which contradicts that N_δ is a δNFA. Hence, B is a normal set basis for C of size at most $\ell - 4$. As $(C, s) \in \mathbf{I}$, we also have that C has a separable normal set basis of size at most $s = \ell - 4$ by Lemma 5(3).

(c) Let N be an NFA for $L(A)$ of size at most $\ell - 1$. We can assume that N is minimal. We will first show that there exists a set basis for C containing at most $s = \ell - 4$ sets.

Recall that we assumed that $s < n - 3$ in our reduction. Hence, we have that $\ell = s + 4 < n + 1 < |A|$. As we observed that A is a minimal DFA for $L(A)$, it must be the case that N has at least one non-deterministic transition.

Notice that, as N is minimal and accepts only strings of length three, we can assume that N has a unique final state. We can partition N's states into four sets $\mathcal{Q}_0$, $\mathcal{Q}_1$, $\mathcal{Q}_2$, and $\mathcal{Q}_3$ such that, for each $0 \leq i \leq 3$, $\mathcal{Q}_i$ is precisely the set of states that N can be in after having read a string of length i. We already know that $\mathcal{Q}_3 = \{q_f\}$. For each state $q \in \mathcal{Q}_2$, define a set $B_q = \{b \mid q \xrightarrow{b} q_f \in \text{Rules}(N)\}$. Next, we show that the collection $B = \{B_q \mid q \in \mathcal{Q}_2\}$ is a set basis for C. By definition of $L(A)$, we have that every $c \in C$ is the union of $B_c := \{B_q \mid \exists p \in \mathcal{Q}_1 : p \xrightarrow{c} q \in \text{Rules}(N)\}$. As $\mathcal{Q}_0$, $\mathcal{Q}_1$, and $\mathcal{Q}_3$ contain at least one state, we also have that B contains at most $\ell - 1 - 3 = s$ sets.

From Lemma 5(3), it now follows that C also has a separable normal set basis containing at most s sets. □

We are now ready to finish the proof of Theorem 2.

Proof of Theorem 2. *Let $\mathcal{N}$ be a class of finite automata such that δNFA $\subseteq \mathcal{N}$. Then the minimization problem for $\mathcal{N}$ is strongly NP-hard, even if the input is given as a DFA.*

Proof. In this Section we provided a reduction from Vertex Cover to DFA $\to \delta$NFA minimization, and showed that, for the regular languages we consider, the minimal NFA is 1 state smaller than the minimal δNFA.

Let $\mathcal{N}$ be a class of finite automata such that $\delta\text{NFA} \subseteq \mathcal{N} \subseteq \text{NFA}$. As was shown in the proof of Lemma 6, any decision algorithm for DFA $\to \mathcal{N}$ minimization can approximate the DFA $\to \delta$NFA minimization problem within a term 1 (as the minimal NFA is only one state smaller than the minimal δNFA).

As can be seen from the other proofs in Section 3, this approximation algorithm for DFA $\to$ δNFA minimization can easily be adapted to an approximation algorithm for Vertex Cover within a term 1. As we know that it is strongly NP-hard to approximate Vertex Cover within a constant term, we can conclude that DFA $\to \mathcal{N}$ minimization is also strongly NP-hard. □

Until now, our results focused on classes of finite automata that can accept all regular languages. Our proof shows that this is not even necessary, as the NP-hard instances we construct only accept strings of length tree. Therefore, we also have the following Corollary.

Corollary 7. *Let δNFA_3 be the class of δNFAs that accept only strings of length three. Let $\mathcal{N}$ be a class of finite automata such that $\delta NFA_3 \subseteq \mathcal{N}$. Then the minimization problem for $\mathcal{N}$ is strongly NP-hard, even if the input is given as a DFA.*

4. Automata with Multiple Initial States

As we mentioned in the Introduction, the minimization problem for finite automata that can non-deterministically choose between multiple initial states, but are otherwise deterministic, has also been studied [20, 26]. We now define a variant of the δNFAs which are geared towards automata with multiple initial states, but in which the only non-determinism is in the choice of initial state (MDFAs in [26]).

Definition 8. *A δMDFA is an NFA A with the following properties*

- *A has at most two initial states;*
- *A is unambiguous; and*
- *for each pair (q, a), degree$(q, a) = 1$.*

From our main proof, we can also infer the following result, strengthening the results from [26] in a uniform manner:

Theorem 9. *Let $\delta MDFA_2$ be the class of δMDFA that accept only strings of length two. Let $\mathcal{N}$ be a class of finite automata such that $\delta MDFA_2 \subseteq \mathcal{N}$. Then the minimization problem for $\mathcal{N}$ is strongly NP-hard, even if the input is given as a DFA.*

Proof. We simply have to reconsider the proof of Lemma 6, change $L(A)$ so that it accepts the *cb* suffixes of its current language, consider the δNFA we construct without its start state, and make the two successors of the start state initial states (and analogously for the NFA considered in the proof). The rest of the proof carries through analogously. □

Together with Theorem 2 this answers all the open questions mentioned by Malcher [26].

5. Succinctness and Uniqueness

As mentioned in the Introduction, when a developer selects a description mechanism for regular languages, she faces a trade-off between succinctness and complexity of minimization. The following proposition shows that in the case of δNFAs, the succinctness bought at the price of NP-completeness is limited.

Proposition 10. *For every δNFA of size n, there is an equivalent DFA of size $O(n^2)$.*

Proof. For every finite automaton that can choose non-deterministically between two different start states but is otherwise deterministic (a 2-MDFA), there is an equivalent DFA with at most quadratically many states [20].

Let A be a δNFA, and let (q, a) be the only pair in $\text{States}(A) \times \text{Alpha}(A)$ such that $\text{degree}(q, a) = 2$. Let q_1 and q_2 be the two states reachable from q when reading an a. If we remove all states from A that are reachable neither from q_1 nor from q_2, and make q_1 and q_2 initial states, we obtain a 2-MDFA A'. We can compute the smallest DFA A'' equivalent to A' in quadratic time. Now, all we need to do is to add A'' to A and replace the two rules going from q to q_1 and q_2, respectively, by a single rule that reads an a and goes to the initial state of A''. The size of the new, deterministic, automaton is $|\text{States}(A)| + |\text{States}(A'')| = O(|\text{States}(A)|^2)$. □

On the other hand, if we were to remove the $\text{branch}(A) \leq 2$ condition in the definition of δNFAs, then there would be an exponential gain in succinctness. This is witnessed by the standard family of languages $(a + b)^* a(a + b)^n$ for $n \geq 0$ that shows that NFAs are exponentially more succinct than DFAs in general. The canonical NFA for this language is unambiguous and has only one pair (q, a) for which $\text{degree}(q, a) = 2$.

Proposition 11. *The minimal δNFA for a regular language is not unique.*

Proof. Consider the language L defined by the regular expression

$$r = (a + b)aaa + b(a + b)(a + b)b.$$

Figure 2 depicts two δNFAs, A and B, that both accept L. We argue that eight is the minimal number of states for any δNFA that accepts L. First, it is clear that any such automaton has to remember how many letters it has read so far. Second, the automaton has to have at least two states that can be reached after reading one letter, one that accepts the string bbb and one that does not. Third, there must also be at least two states reachable after reading two letters, one that accepts the string bb and one that doesn't. Fourth, there must be at least two states that can be reached after reading three letters, one that accepts the string b and one that does not. Together with the fact that there has to be one initial state and at least one final state, this shows that any δNFA for L needs at least eight states. □

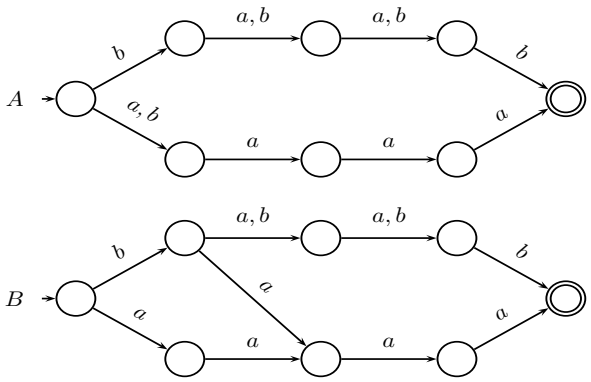

Figure 2: Two minimal δNFAs, that both accept the language L from the proof of Proposition 11.

6. Concluding Remarks

In this paper, we have investigated the following question: *Can we loosen the determinism constraint on finite automata, while still admitting PTIME minimization?* In other words, do there exist significant extensions of the class of DFAs that still allow for polynomial time minimization? We have shown that no such significant extensions exist, under the assumption that PTIME $\neq$ NP. Formally, we proved that minimization is NP-hard for all finite automata classes that contain the δNFAs that accept strings of length three. Here, the class of δNFAs is an extension of the class of DFAs that allows for very little non-determinism.

A natural and important next direction is to look for classes that may not contain the class of all DFAs. While there may be no general relaxation of the determinism constraint that admits PTIME minimization, one could, e.g., additionally restrict the alphabet size of automata. It remains open whether results in the spirit of Theorems 2 and 9 would also hold if δNFAs are required to have fixed-size alphabets.

References

[1] P. Abdulla, J. Deneux, L. Kaati, and M. Nilsson. Minimization of non-deterministic automata with large alphabets. In *International Conference on Implementation and Application of Automata (CIAA)*, pages 31–42, 2005.

[2] A. Badr, V. Geffert, and I. Shipman. Hyper-minimizing minimized deterministic finite state automata. *Informatique Théorique et Applications*, 43(1):69–94, 2009.

[3] J. Berstel, L. Boasson, and O. Carton. Continuant polynomials and worst-case behavior of hopcroft's minimization algorithm. *Theoretical Computer Science*, 410(30–32):2811–2822, 2009.

[4] J. Berstel, L. Boasson, O. Carton, and I. Fagnot. Minimization of automata. Technical Report 1010.5318, CoRR abs, 2010.

[5] G. J. Bex, W. Gelade, W. Martens, and F. Neven. Simplifying XML Schema: effortless handling of nondeterministic regular expressions. In *International Symposium on Management of Data (SIGMOD)*, pages 731–744, 2009.

[6] H. Björklund and W. Martens. The tractability frontier for NFA minimization. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 27–38, 2008.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability — A Guide to the theory of NP-completeness*. W. H. Freeman, 1979.

[8] P. Gawrychowski and A. Jez. Hyper-minimisation made efficient. In *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 356–368, 2009.

[9] W. Gelade. Succinctness of regular expressions with interleaving, intersection and counting. *Theoretical Computer Science*, 411(31–33):2987–2998, 2010.

[10] W. Gelade, M. Gyssens, and W. Martens. Regular expressions with counting: Weak versus strong determinism. In *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 369–381, 2009.

[11] W. Gelade and F. Neven. Succinctness of the complement and intersection of regular expressions. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 325–336, 2008.

[12] J. Goldstine, M. Kappes, C Kintala, H. Leung, A. Malcher, and D. Wotschke. Descriptional complexity of machines with limited resources. *Journal of Universal Computer Science*, 8(2):193–234, 2002.

[13] G. Gramlich and G. Schnitger. Minimizing NFAs and regular expressions. *Journal of Computer and System Sciences*, 73(6):908–923, 2007.

[14] H. Gruber and M. Holzer. Finding lower bounds for nondeterministic state complexity is hard. In *International Conference on Developments in Language Theory (DLT)*, pages 363–374, 2006.

[15] H. Gruber and M. Holzer. Computational complexity of NFA minimization for finite and unary languages. In *International Conference on Language and Automata Theory and Applications (LATA)*, pages 261–272, 2007.

[16] H. Gruber and M. Holzer. Inapproximability of nondeterministic state and transition complexity assuming P $\neq$ NP. In *International Conference on Developments in Language Theory (DLT)*, pages 205–216, 2007.

[17] H. Gruber and M. Holzer. Tight bounds on the descriptional complexity of regular expressions. In *International Conference on Developments in Language Theory (DLT)*, pages 276–287, 2009.

[18] M. Holzer and M. Kutrib. Nondeterministic finite automata — recent results on the descriptional and computational complexity. In *International Conference on Implementation and Application of Automata (CIAA)*, pages 1–16, 2008.

[19] M. Holzer and A. Maletti. An $n \log n$ algorithm for hyper-minimizing states in a (minimized) deterministic automaton. *Theoretical Computer Science*, 411(38–39):3404–3413, 2010.

[20] M. Holzer, K. Salomaa, and S. Yu. On the state complexity of k-entry deterministic finite automata. *Journal of Automata, Languages, and Combinatorics*, 6(4):453–466, 2001.

[21] D. Hovland. Regular expressions with numerical constraints and automata with counters. In *International Colloquium on Theoretical Aspects of Computing (ICTAC)*, pages 231–245, 2009.

[22] J. Hromkovic, J. Karhumäki, H. Klauck, G. Schnitger, and S. Seibert. Measures of nondeterminism in finite automata. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 199–210, 2000.

[23] J. Hromkovic and G. Schnitger. Comparing the size of NFAs with and without epsilon-transitions. *Theoretical Computer Science*, 380(2):100–114, 2007.

[24] T. Jiang, E. McDowell, and B. Ravikumar. The structure and complexity of minimal NFAs over unary alphabet. *International Journal of Foundations of Computer Science*, 2:163–182, 1991.

[25] T. Jiang and B. Ravikumar. Minimal NFA problems are hard. *Siam Journal on Computing*, 22(6):1117–1141, 1993.

[26] A. Malcher. Minimizing finite automata is computationally hard. *Theoretical Computer Science*, 327(3):375–390, 2004.

[27] A. Meyer and M.J. Fischer. Economy of descriptions by automata, grammars, and formal systems. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 188–191. IEEE, 1971.

[28] A. Okhotin. Unambiguous finite automata over a unary alphabet. In *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 556–567, 2010.

[29] R Paige and R. Tarjan. Three parition refinement algorithms. *Siam Journal on Computing*, 16:973–989, 1987.

[30] K. Salomaa. Descriptional complexity of nondeterministic finite automata. In *International Conference on Developments in Language Theory (DLT)*, pages 31–35, 2007.

[31] G. Schnitger. Regular expressions and NFAs without epsilon-transitions. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 432–443, 2006.

[32] R. E. Stearns and H. B. Hunt III. On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *Siam Journal on Computing*, 14(3):598–611, 1985.

[33] L. Stockmeyer and A. Meyer. Word problems requiring exponential time: Preliminary report. In *Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–9, 1973.