The Tractability Frontier for NFA Minimization[⋆]

Henrik Björklund and Wim Martens

TU Dortmund
henrik.bjoerklund@udo.edu,
wim.martens@udo.edu

Abstract. We essentially show that minimizing finite automata is NP-hard as soon as one deviates from the class of deterministic finite automata. More specifically, we show that minimization is NP-hard for all finite automata classes that subsume the class that is unambiguous, allows at most one state q with a non-deterministic transition for at most one alphabet symbol a, and is allowed to visit state q at most once in a run. Furthermore, this result holds even for automata that only accept finite languages.

1 Introduction

The regular languages are immensely important, not only in theoretical computer science, but also in practical applications. When using regular languages in practice, 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 automata (NFAs) can provide an exponentially more succinct description than deterministic finite automata (DFAs) [13]. 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.

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. A rather successful intermediate model is the class of *unambiguous* finite automata (UFAs).[1] 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 on UFAs [17]. However, UFAs do not allow for tractable state minimization [11]. Therefore, the question whether there are good intermediate models between

⋆ This work was supported by the DFG Grant SCHW678/3-1.

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

DFAs and NFAs needs to be revisited for state minimization. The main result of this paper is that, probably, no such good models exist. Even the tiniest bit of non-determinism makes the minimization problem NP-hard.

Minimizing unrestricted NFAs is PSPACE-complete [18] and every undergraduate computer science curriculum teaches its students how to minimize a DFA in polynomial time. The minimization problem for automata with varying degrees of non-determinism was studied in a seminal paper by Jiang and Ravikumar in 1993 [11]. Among other results, they thoroughly investigated the minimization problem for UFAs. They showed the following.

– Given a UFA, finding the minimal equivalent UFA is NP-complete.
– Given a DFA, finding the minimal equivalent UFA is NP-complete.
– Given a DFA, finding the minimal equivalent NFA is PSPACE-complete.

Minimization problems have even been studied for automata with unary alphabets; see, e.g., [10,5].

Recently, Malcher [12] 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.[2]

Whereas Malcher made significant progress in showing that minimization is hard for non-deterministic automata, he was not yet able to solve the entire question. Therefore, he states the question whether there are relaxations of the deterministic automata model *at all* for which minimization is tractable as an important open problem. In this respect, 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. In the present paper, we settle these open questions and 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 δNFAs, and 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. On the other hand, there *are* relaxations of the deterministic automaton model that allow tractable minimization. We show that, if we add to the definition of

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

δNFAs that each word should have at most one *rejecting* computation, minimization becomes tractable again. However, the minimal automata in this class are the DFAs, so this class is not likely to be very useful in practice.

Other related work. An overview of state and transition complexity of NFAs can be found in [15]. Known results about the trade-off between amount of non-determinism and descriptional complexity are surveyed in [2]. The problems of producing small NFAs from regular expressions has been considered in [9,16]. There has also been some work on approximating minimal NFAs [3,4,6], basically showing that approximation is very hard. The work of Hromkovic et al. [8] about measuring non-determinism in finite automata will be relevant to us in Section 2. Since the minimization problem for NFAs is hard, the bisimulation minimization problem has also been considered; see, e.g., [14,1].

Due to space restrictions, some proofs and parts of proofs have been omitted, and will appear in the full version of the paper.

2 Preliminaries

By Σ we always denote 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, $\text{Alpha}(A) = \Sigma$, $\text{init}(A) \in \text{States}(A)$ is its initial state, $\text{Final}(A) \subseteq \text{States}(A)$ 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. A finite automaton is *deterministic* if, 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 $q_1 \xrightarrow{a} q_2 \in \text{Rules}(A)$. By DFA we denote the class of deterministic finite automata.

A *run*, or *computation*, r of A on a word $w = a_1 \cdots a_n \in \Sigma^*$ is a string $q_1 \cdots q_n \in \text{States}(A)^*$ such that $\text{init}(A) \xrightarrow{a_1} q_1 \in \text{Rules}(A)$ and, for each $i = 1, \ldots, n-1$, $q_i \xrightarrow{a_{i+1}} q_{i+1} \in \text{Rules}(A)$. The run is *accepting* if $q_n \in \text{Final}(A)$. The *language of* 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 $\text{branch}_A(w) = \max\{\Pi_{i=1}^{n} \text{degree}(q_{i-1}, a_i) \mid q_1 \cdots q_n$ is a run of A on $a_1 \cdots a_n$, and $\text{init}(A) = q_0\}$. The *branching of* A, denoted branch(A), is $\max\{\text{branch}_A(w) \mid w \in L(A)\}$ if this quantity is defined, and otherwise ∞.

Hromkovic et al. [8] define three measures non-determinism for a finite automaton A: *advice*(A), *computations*(A), and *ambig*(A). These measures are defined as follows: advice(A) is the maximum number of non-deterministic choices during any computation of A, computations(A) is the maximum number of different computations of A on any word,[3] and ambig(A) is the maximum number of different *accepting* computation of A on any word. For the formal definitions of these concepts, we refer to [8].

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.

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

- A is unambiguous;
- $\text{branch}(A) \leq 2$; and
- there is at most one pair (q, a) such that $\text{degree}(q, a) = 2$.

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 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. For two classes of finite automata $\mathcal{N}_1$ and $\mathcal{N}_2$ the $\mathcal{N}_1 \to \mathcal{N}_2$ *minimization problem* is the following problem. Given a finite automaton A in $\mathcal{N}_1$ and an integer k, does there exist a finite automaton B in $\mathcal{N}_2$ of size at most k such that $L(A) = L(B)$? For a class $\mathcal{N}$ of finite automata, the *minimization problem for* $\mathcal{N}$ is then simply the $\mathcal{N} \to \mathcal{N}$ minimization problem.

Suppose that the $\mathcal{N}_1 \to \mathcal{N}_2$ minimization problem is hard for a complexity class 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 C. However, assuming that $\mathcal{N}_1 \to \mathcal{N}_2$ is hard for 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 C, as finding a small $\mathcal{N}_3$ automaton might be easier than finding a small $\mathcal{N}_2$ automaton in general.[4] Therefore, we will prove directly that minimization is NP-hard for *all* classes of automata between δNFAs and NFAs.

[3] Hromkovic et al. wrote leaf(A) instead of computations(A).

[4] 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 [12]).

2.4 Are δNFAs the Closest Possible to Determinism?

Before we give more intuition about this question, we first note that there are in fact *two* incomparable notions of determinism for finite automata: determinism and *reverse* determinism.[5] Both classes can be efficiently minimized by the same algorithm, modulo a simple pre- and post-processing step for reverse deterministic automata. 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 believe that one can always think of more and more exotic notions of non-determinism that come closer and closer to DFAs.[6] We provide an example here. Define the class $\mathcal{C}$ to be the class of δNFAs with the additional condition that, for each word w, there can be at most one *rejecting* computation of A on w. (Thus, for each w, there can be at most two runs — one accepting and one rejecting.) This notion of non-determinism lies strictly between DFAs and δNFAs (DFA $\subseteq C \subseteq \delta$NFA).

Consider the minimization problem for $\mathcal{C}$ and let A be an arbitrary automaton in $\mathcal{C}$. We will argue that the minimal $\mathcal{C}$-automaton for $L(A)$ is a DFA. Suppose that A is not a DFA. Let q and a be the unique state and label so 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 Rules(A). Let w be an arbitrary string that leads A to state q. By definition of $\mathcal{C}$, A must accept every string of the form waw', where w' is an arbitrary word in Alpha$(A)^*$. (If waw' would be rejected, then there would be two rejecting runs, one over q_1 and one over q_2.) Therefore, we can make A strictly smaller by merging the two states q_1, q_2 to q_3, making q_3 a final state, and adding loop transitions to q_3 for each alphabet symbol. Moreover, by this operation, A becomes deterministic. Hence, every automaton A in $\mathcal{C}$ that is not a DFA can be rewritten as a smaller DFA. This means that, in the class $\mathcal{C}$, the minimal automata are the DFAs. In particular, this also puts the minimization problem for $\mathcal{C}$ 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 $\mathcal{C}$; two classes that are extremely close to one another.

3 Minimizing Non-deterministic Automata is Hard

The main result of this section is the following.

[5] The latter is the class for which the inverted transitions are deterministic.

[6] One could, for instance, take the class of DFAs and add a single NFA.

Theorem 2. *Let $\mathcal{N}$ be a class of finite automata. If δNFA $\subseteq \mathcal{N}$ then DFA $\rightarrow \mathcal{N}$ minimization is NP-hard.*

Corollary 3. *For each class $\mathcal{N}$ of finite automata such that δNFA $\subseteq \mathcal{N}$, the minimization problem for $\mathcal{N}$ is 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, whether there exists a vertex cover of G of size at most k. *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 [11]), 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 $\rightarrow$ δNFA is NP-hard doesn't immediately imply that DFA $\rightarrow \mathcal{N}$ 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 [11], as our further results rely on a construction in their proof.

Lemma 4 (Jiang and Ravikumar [11]). *Normal Set Basis is NP-complete.*

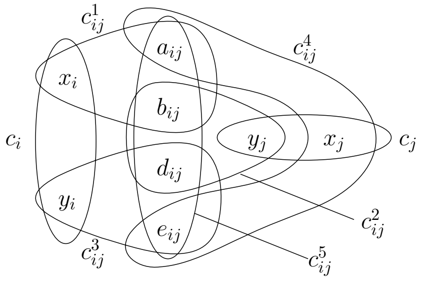

Fig. 1. The constructed sets $c_i, c_j, c_{ij}^1, \ldots, c_{ij}^5$ in the proof of Lemma 4

Proof (Sketch). Obviously, Normal Set Basis is in NP. Indeed, given an input (C, s) for Normal Set Basis, 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.

We give a reduction from Vertex Cover to Normal Set Basis but omit the correctness proof. 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.

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. Indeed, if $k \geq |E| - 3$, 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.

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:

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

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 polynomial time.

It can be shown 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. □

The next lemma now follows from the proof of Lemma 4. It shows that C has a set basis containing s sets if and only if C has a separable normal set basis containing s sets for any input (C, s) in $\mathbf{I}$. 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 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 s sets.
(b) C has a separable normal set basis containing 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 $k \leq |E| - 3$. This immediately shows (1) and (2). We continue by proving the other cases.
(3) The direction from (b) to (a) is trivial. For the other direction the full proof of Lemma 4 actually shows that if G has a vertex cover of size k, then C has a *separable* normal set basis containing s sets. Conversely, if C has a *normal* set basis containing at most s sets, then G has a vertex cover of size k. This would imply that C also has a separable normal set basis containing s sets.

Hence, we still 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. 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.

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, since c_i is the only set in C of which $\{x_i, y_i\}$ is a subset. Therefore, we can assume without loss of generality that B either contains $\{x_i, y_i\}$ or $\{x_i\}$ and $\{y_i\}$, but never both $\{x_i, y_i\}$ and $\{x_i\}$ (or, symmetrically, $\{x_i, y_i\}$ and $\{y_i\}$).

Suppose there are $1 \leq i < j \leq n$ and $1 \leq r \leq 4$ such that c_{ij}^r cannot be formed as a disjoint union of sets from B. Without loss of generality, we may assume that $r = 1$, i.e., $c_{ij}^r = c_{ij}^1 = \{x_i, a_{ij}, b_{ij}\}$, since all other cases follow by symmetry. Since there are no disjoint sets from B whose union is c_{ij}^1, there must be two different sets b^1 and b^2 in B that are subsets of c_{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_{ij}^1 that contains x_i is a subset of any set of C other than c_{ij}^1. This means that we can replace b^1 with $b^1 - b^2$ in B and still have a set basis of size at most s. Thus we can assume that for any i, j and any $r \in \{1, \ldots, 4\}$, the set c_{ij}^r can be formed as a union of disjoint sets from B.

The only remaining case is if there are $1 \leq i < j \leq n$ such that c_{ij}^5 cannot be formed as a disjoint union of sets from B. Let B_{ij} be a subset of B such that the union of the sets in B_{ij} is c_{ij}^5. We can assume that B_{ij} is inclusion free, i.e., there are no two sets in B_{ij} such that one is a subset of the other. If B_{ij} has four members, then we can replace B_{ij} with the four singletons and still have a set basis of size s, so we can assume that B_{ij} has at most three members. Suppose there is a set b in B_{ij} such that b is not a subset of any set in C other than c^5_{ij}. Then we can replace b with $b - (\bigcup_{b' \in B_{ij} - \{b\}} b')$ in B and still have a set basis of size at most s. Thus we can assume that each member of B_{ij} is a subset of some set from C other that c^5_{ij}. In particular, this means that each member of B_{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_{ij} with $\{b^1, b^2\}$. Thus we can assume that there are at most two sets in B_{ij} with two elements. Furthermore, these two sets must overlap. This means that we can assume that B_{ij} has exactly three members, two with two elements and one singleton.

Without loss of generality, we may assume that $B_{ij} = \{\{a_{ij}, b_{ij}\}, \{b_{ij}, d_{ij}\}, \{e_{ij}\}\}$. 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_{ij}, and if $\{b_{ij}\}$ belongs to B we can replace $\{b_{ij}, d_{ij}\}$ by $\{d_{ij}\}$in B_{ij}.) This means that 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. If it is not $\{x_i\}$, we can replace it 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 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 $\{d_{ij}\}$. Once we have $\{d_{ij}\}$ in B, we can replace $\{b_{ij}, d_{ij}\}$ by $\{b_{ij}\}$ in B and still have a set basis for C of size at most s, one that can form c^5_{ij} as a union of disjoint members.

In summary, 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) One simply has to 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). □

The proof of the following lemma is partly inspired by the proof of Theorem 3.1 of [11], but we significantly strengthen it for our purposes.

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

(1) DFA $\rightarrow$ δNFA minimization is NP-complete for DFAs accepting $\mathcal{L}$ 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 (Sketch). The NP upper bound is immediate, as equivalence testing for unambiguous finite automata is in PTIME [17]. We can guess a δNFA of sufficiently small size and test in PTIME whether it is equivalent to the given δNFA.

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 $s < n - 3$.

We construct in LOGSPACE a δNFA 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 δNFA 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 $c_i = \{b_{i,1}, \ldots, b_{i,n_i}\}$. 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 Rules(A) are formally defined as follows:

– $q_0 \xrightarrow{a} q_0'$;
– for every $i = 1, \ldots, n$, $q_0' \xrightarrow{c_i} q_i$; and
– for every $i = 1, \ldots, n$ and $j = 1, \ldots, n_i$, $q_i \xrightarrow{b_{i,j}} q_f$.

Finally, define $\ell := s + 4$.

Obviously, A and ℓ can be constructed from C and s using logarithmic space. Observe that due to Lemma 5, C contains every set at most once, and hence does not contain $c_i = c_j$ with $i \neq j$. Hence, A is a minimal DFA for $L(A)$.

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 $\ell = s + 4$.

Let $B = \{r_1, \ldots, r_s\}$ be the separable normal set basis for C containing s 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, Rules(N_δ) 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 $m = 1, 2$, Rules(N_δ) 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 $r_j \in B_m$ and $b \in r_j$.

Notice that the size of N_δ is $|B| + 4 = s + 4 = \ell$. By construction, we have that $L(N_\delta) = L(A)$. It can be shown that N_δ is a δNFA.
(b) This part of the proof is omitted.
(c) It can be shown that an NFA for $L(A)$ of size at most $\ell - 1$ gives rise to a set basis of size s. From Lemma 5, it now follows that C also has a separable normal set basis containing at most s sets. □

Theorem 2 now follows from the proof of Lemma 6.

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$_{\mathit{finite}}$ be the class of δNFAs that accept only finite languages. Let $\mathcal{N}$ be class of finite automata. If δNFA$_{\mathit{finite}} \subseteq \mathcal{N}$ then the DFA $\rightarrow \mathcal{N}$ minimization problem is NP-hard.*

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

On the other hand, if we were to remove the 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 degree$(q, a) = 2$.

Proposition 9. *The minimal δNFA for a regular language is not unique.*

5 Automata with Multiple Initial States

Throughout the paper, to simplify definitions, we have assumed that finite automata have a unique start state. 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 [7,12].

Proposition 10. *Minimization is NP-hard for unambiguous finite automata that have at most two initial states but are otherwise deterministic.*

Together with Theorem 2 this solves all the open cases mentioned by Malcher [12].

References

1. Abdulla, P., Deneux, J., Kaati, L., Nilsson, M.: Minimization of non-deterministic automata with large alphabets. In: CIAA, pp. 31–42 (2006)
2. Goldstine, J., Kappes, M., Kintala, C., Leung, H., Malcher, A., Wotschke, D.: Descriptional complexity of machines with limited resources. J. Univ. Comp. Science 8(2), 193–234 (2002)
3. Gramlich, G., Schnitger, G.: Minimizing NFAs and regular expressions. JCSS 73(6), 908–923 (2007)
4. Gruber, H., Holzer, M.: Finding lower bounds for nondeterministic state complexity is hard. In: H. Ibarra, O., Dang, Z. (eds.) DLT 2006. LNCS, vol. 4036, pp. 363–374. Springer, Heidelberg (2006)
5. Gruber, H., Holzer, M.: Computational complexity of NFA minimization for finite and unary languages. In: LATA, pp. 261–272 (2007)
6. Gruber, H., Holzer, M.: Inapproximability of nondeterministic state and transition complexity assuming $P \neq NP$. In: Harju, T., Karhumäki, J., Lepistö, A. (eds.) DLT 2007. LNCS, vol. 4588, pp. 205–216. Springer, Heidelberg (2007)
7. Holzer, M., Salomaa, K., Yu, S.: On the state complexity of k-entry deterministic finite automata. J. Automata, Languages, and Combinatorics 6(4), 453–466 (2001)
8. Hromkovic, J., Karhumäki, J., Klauck, H., Schnitger, G., Seibert, S.: Measures of nondeterminism in finite automata. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 199–210. Springer, Heidelberg (2000)
9. Hromkovic, J., Schnitger, G.: Comparing the size of NFAs with and without epsilon-transitions. TCS 380(2), 100–114 (2007)
10. Jiang, T., McDowell, E., Ravikumar, B.: The structure and complexity of minimal NFAs over unary alphabet. Int. J. Found. Comp. Science 2, 163–182 (1991)
11. Jiang, T., Ravikumar, B.: Minimal NFA problems are hard. Siam J. Comp. 22(6), 1117–1141 (1993)
12. Malcher, A.: Minimizing finite automata is computationally hard. TCS 327(3), 375–390 (2004)
13. Meyer, A., Fischer, M.J.: Economy of descriptions by automata, grammars, and formal systems. In: FOCS, pp. 188–191. IEEE, Los Alamitos (1971)
14. Paige, R., Tarjan, R.: Three parition refinement algorithms. Siam J. Comp. 16, 973–989 (1987)
15. Salomaa, K.: Descriptional complexity of nondeterministic finite automata. In: Harju, T., Karhumäki, J., Lepistö, A. (eds.) DLT 2007. LNCS, vol. 4588, pp. 31–35. Springer, Heidelberg (2007)
16. Schnitger, G.: Regular expressions and NFAs without epsilon-transitions. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 432–443. Springer, Heidelberg (2006)
17. Stearns, R.E., Hunt III., H.B.: On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. Siam J. Comp. 14(3), 598–611 (1985)
18. Stockmeyer, L., Meyer, A.: Word problems requiring exponential time: Preliminary report. In: STOC, pp. 1–9. ACM, New York (1973)