Efficiently Computing the Minimum Rank of a Matrix in a Monoid of Zero-One Matrices

Stefan Kiefer
Department of Computer Science, University of Oxford, UK

Andrew Ryzhikov
Department of Computer Science, University of Oxford, UK

Abstract

A zero-one matrix is a matrix with entries from $\{0,1\}$. We study monoids containing only such matrices. A finite set of zero-one matrices generating such a monoid can be seen as the matrix representation of an unambiguous finite automaton, an important generalisation of deterministic finite automata which shares many of their good properties.

Let $\mathcal{A}$ be a finite set of $n \times n$ zero-one matrices generating a monoid of zero-one matrices, and m be the cardinality of $\mathcal{A}$. We study the computational complexity of computing the minimum rank of a matrix in the monoid generated by $\mathcal{A}$. By using linear-algebraic techniques, we show that this problem is in NC and can be solved in $\mathcal{O}(mn^4)$ time. We also provide a combinatorial algorithm finding a matrix of minimum rank in $\mathcal{O}(n^{2+\omega} + mn^4)$ time, where $2 \leq \omega \leq 2.4$ is the matrix multiplication exponent. As a byproduct, we show a very weak version of a generalisation of the Černý conjecture: there always exists a straight line program of size $\mathcal{O}(n^2)$ describing a product resulting in a matrix of minimum rank.

For the special case corresponding to complete DFAs (that is, for the case where all matrices have exactly one 1 in each row), the minimum rank is the size of the smallest image of the set of states under the action of a word. Our combinatorial algorithm finds a matrix of minimum rank in time $\mathcal{O}(n^3 + mn^2)$ in this case.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory; Computing methodologies → Symbolic and algebraic manipulation

Keywords and phrases matrix monoids, minimum rank, unambiguous automata

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.61

Funding *Andrew Ryzhikov*: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 852769, ARiAT).

Acknowledgements We thank the anonymous reviewers for their helpful comments that improved the presentation of the paper.

1 Introduction

Matrix monoids are a rich and versatile object used in formal verification, program analysis, dynamical systems and weighted automata. However, many of their properties are in general undecidable. One such example is the well-studied matrix mortality problem. Given a finite set $\mathcal{A}$ of $n \times n$ matrices, it asks if the monoid generated by $\mathcal{A}$ (that is, the set of all products of matrices from $\mathcal{A}$) contains the zero matrix. This problem is undecidable already for 3×3 integer matrices [40], and was studied for several decidable special cases, see e.g. [16, 6, 47].

Even if $\mathcal{A}$ is a set of zero-one matrices (that is, matrices with entries in $\{0,1\}$), matrix mortality is PSPACE-complete [47]. We thus restrict our attention to the case where the whole monoid generated by $\mathcal{A}$ consists of zero-one matrices; in this case, matrix mortality becomes decidable in polynomial time [34]. We call such monoids *zero-one matrix monoids*. Intuitively, when multiplying any two matrices from such a monoid, we never get $1+1$ as a subexpression. Zero-one matrix monoids have a rich structure while still admitting good algorithmic properties. They correspond precisely to unambiguous finite automata, and find applications in formal verification [3], variable-length codes [9] and symbolic dynamics [38]. They are also an interesting special case of finite monoids of rational matrices (studied in, e.g., [39, 28, 1, 13]), monoids of nonnegative matrices (studied in, e.g., [41, 10, 51, 22]), and, in the case where they do not contain the zero matrix, of matrix monoids with constant spectral radius [42].

In this paper, we consider a problem that can be seen as a natural generalisation of matrix mortality: given a finite set $\mathcal{A}$ generating a zero-one monoid, find the minimum real rank of a matrix in this monoid. By the real rank of a matrix we mean the dimension of the subspace generated by its columns over the reals. Clearly, this rank is zero if and only if the monoid contains the zero matrix. The minimum real rank of a matrix in a zero-one matrix monoid is a much more tractable problem than deciding other similar properties: for example, checking if a zero-one matrix monoid contains a matrix of a given real rank was shown to be NP-hard[1] [24].

The goal of our paper is threefold. Firstly, we present efficient algorithms for analysing monoids of zero-one matrices and unambiguous finite automata. Secondly, to obtain these algorithms, we provide new structural properties of such monoids and automata that might be interesting on their own. Thirdly, we strengthen the connections between the areas of synchronising automata, weighted automata and matrix semigroups by transferring methods and tools between them. We also highlight open problems in the intersection of these areas.

2 Existing results and our contributions

Throughout the paper, we always assume that matrix monoids are defined by sets of generators, and all matrices are square zero-one unless stated otherwise.

Complete DFAs

An $n \times n$ zero-one matrix with exactly one 1 in every row can be equivalently seen as a transformation of a set Q of size n. A set of such matrices generates a zero-one matrix monoid, and can be seen as a complete deterministic finite (semi-)automaton[2] (complete DFA) $\mathcal{A} = (Q, \Sigma, \delta)$. Here, Σ is a finite alphabet whose letters correspond to the generating matrices, and $\delta : Q \times \Sigma \to Q$ is the transition function defined in such a way that for each $a \in \Sigma$, $\delta(\_, a)$ is the transformation of Q induced in the natural way by the matrix corresponding to a. Thus, words over Σ correspond to products of the generating matrices.

The *rank* of a word w in $\mathcal{A}$ is the size of the image of Q under the transformation corresponding to w. Equivalently, it is the real rank of the matrix corresponding to w. The *rank* of a complete DFA is the minimum among the ranks of all its words. This concept was studied from the perspectives of automata theory [43, 30] and the theory of transformation semigroups [48, 31]. It is the subject of the rank conjecture (called the Černý-Pin conjecture in [43]), which states that every complete DFA of rank r admits a word of rank r having length at most $(n-r)^2$. The Černý conjecture, one of the oldest open problems in combinatorial automata theory [50], is a special case with $r = 1$. We refer to surveys [49, 4, 30, 50] for the vast literature on the Černý conjecture. Underlying digraphs of complete DFAs of a given rank were studied in [12, 4] in the context of the road colouring problem.

[1] In fact, it is PSPACE-complete, which follows directly from [8, Theorem 3]: add a fresh state and define all yet undefined transitions to lead to this state.

[2] In this paper, all automata are semi-automata, meaning that they do not have any initial or accepting states, and do not recognise any languages. Following the usual conventions (as in, e.g., [9]), we omit "semi-", in particular because it would make abbreviations like DFA less recognisable.

The rank of an n-state complete DFA over an alphabet of size m can be found in $\mathcal{O}(m^4n^4)$ time [43, Theorem 1]. In contrast, for any fixed $r \geq 2$, the problem of checking if a complete DFA admits a word of rank r is NP-hard [24]. Checking if an n-state complete DFA over an alphabet of size m has rank one is NL-complete [26, 50], and can be done in $\mathcal{O}(mn^2)$ time [20, 49]. For each complete DFAs of rank r, there exists a word of rank r of length at most $\frac{(n-r)^3}{6} + \mathcal{O}((n-r)^2)$ [37], and if $r = 1$, finding a word of rank one can be done in $\mathcal{O}(n^3 + mn^2)$ time and $\mathcal{O}(n^2)$ space [20].

Unambiguous finite automata

Generalising the case of complete DFAs, a set $\mathcal{A}$ of $n \times n$ zero-one matrices generating a zero-one matrix monoid can be equivalently seen as an unambiguous nondeterministic finite (semi-)automaton (UFA). Let $Q = \{q_1, \ldots, q_n\}$ be its set of states. To each matrix in $\mathcal{A}$ we again associate a letter in the alphabet Σ, and the transition relation $\Delta \subseteq Q \times \Sigma \times Q$ is defined so that $(q_i, a, q_j) \in \Delta$ if and only if the entry (i, j) in the matrix corresponding to a is equal to one. Just as in the complete DFA case, words over Σ naturally correspond to products of matrices from $\mathcal{A}$.

The obtained NFA then has the property that is sometimes called *diamond-free*: for every two states p, q and every word w, there is at most one path from p to q labelled by w. A simple reachability argument shows that the length of a shortest word labelling two such paths, if it exists, is at most quadratic in the dimension of the matrices. Hence, deciding whether an NFA is a UFA (and thus whether a set of zero-one matrices generates a zero-one monoid) is in $\mathsf{coNL} = \mathsf{NL}$. It is actually NL-complete as described in the next subsection.

A UFA is called complete if it does not admit a word whose matrix is the zero matrix. For an n-state UFA the length of such a word if it exists is at most n^5 [34]. The best known lower bound is quadratic in n, and is achieved by a series of DFAs [44]. For UFAs, the quadratic upper bound was conjectured to be tight [43, Conjecture 2]. Checking if a UFA is complete can be done in NC^2 [34].

The *real rank* of a UFA is the minimum among the real ranks of the matrices corresponding to words. It was shown in [14] that for an n-state UFA of real rank $r \geq 1$ there always exists a word of minimum rank of length $\mathcal{O}(rn^3)$. For n-state strongly connected Eulerian UFAs of rank one, a subclass with remarkably nice properties, there always exists a word of length at most $(n-1)^2$ of rank one [15, Corollary 4]. All mentioned constructions also provide polynomial time algorithms that construct words with the required properties (in particular, with a length within the stated bounds).

Applications to variable-length codes

A *variable-length code* (or simply a *code*) is a set X of finite words over an alphabet Σ such that every finite word over Σ has at most one factorisation over X. In other words, a code is a basis of a free submonoid of Σ^*.

The definitions of both UFAs and codes rely, intuitively, on the uniqueness of certain representations. In fact, UFAs and codes are tightly related. Let us illustrate this relationship. If the cardinality of a code X is finite, one can construct its flower automaton, which is a UFA with a chosen state s such that, for each word from X, there is a separate cycle containing s and labelled by this word, see Figure 1 (left) for an example. More generally, codes that are regular languages correspond precisely to strongly connected UFAs in a similar way, see [9, Chapter 4] for the details.

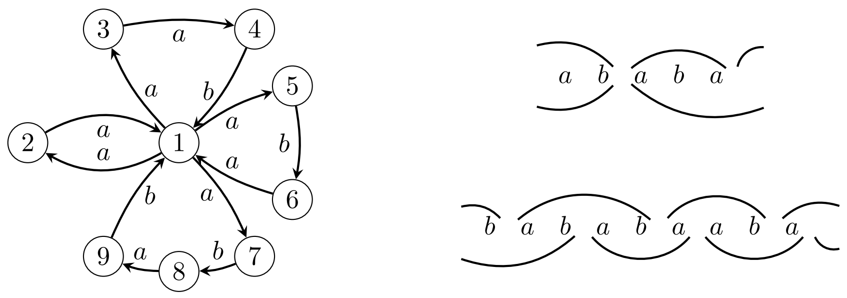


Figure 1 The flower automaton of the code $X = \{aa, aab, aba, abab\}$ (left), two adjacent interpretations of *ababa* over X (top right), and two disjoint interpretations of *bababaaba* over X (bottom right). Note that this code is not complete, but still illustrates all the discussed properties.

A useful corollary of the construction of the flower automaton is the fact that deciding if a set of zero-one matrices generates a zero-one monoid is NL-hard. Indeed, a finite set of words is a code if and only if its flower automaton is unambiguous [9]. Deciding if a finite set of words is a code is NL-complete [45], and the flower automaton can be constructed in AC^0.

A code X over Σ is called *complete* if every word over Σ is a factor of a concatenation of codewords, that is, for every word $w \in \Sigma^*$ there exist $u, v \in \Sigma^*$ with $uwv \in X^*$. A code that is a regular language is complete if and only if the corresponding UFA is complete [9]. For complete codes that are regular languages, the real rank of the corresponding UFA is equal to a natural and important parameter called the degree of a code [9, Proposition 9.6.1].

Let us explain the intuition behind the notion of degree, see [9, Chapter 9.6] for the formal definitions. For each word w we can consider all possible factorisations over X of all its extensions $uwv \in X^*$ with $u, v \in \Sigma^*$, called *interpretations* of w. Two such interpretations either match in at least one position (as in Figure 1 (top right) between the second and the third letter), or do not match in any position (as in Figure 1 (bottom right)), in which case they are called *disjoint*. The degree of a word is the number of pairwise disjoint interpretations of this word. The degree of a code X is the minimum nonzero degree of all words $w \in \Sigma^*$.

A particularly important case is when a complete code has degree one. Then there exists a word $w \in X^*$ (called a synchronising word) such that for any concatenation of codewords $uwwv \in X^*$ with $u, v \in \Sigma^*$ we have $uw, wv \in X^*$. Intuitively, this means that the two halves uw and wv can be decoded separately and independently.

Computational complexity classes

In this paper, we characterise the computational complexity of problems by showing that they belong to the classes $\mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC} \subseteq \mathsf{P}$, see [2, 23] for their formal definitions. NL is the class of problems solvable in nondeterministic logarithmic time. NC^k is the class of problems solvable by $\mathcal{O}((\log n)^k)$-depth polynomial-size bounded fan-in Boolean circuits, and NC is the union of these classes for all $k \geq 1$. The class NC represents problems that have efficient parallel algorithms, and is a subclass of problems solvable in polylogarithmic space [2]. Intuitively, NC is the class of problems that can be solved using local computations, as opposed to P-complete problems, which are inherently sequential and thus require storing the entire structure in the memory unless $\mathsf{NC} = \mathsf{P}$. Showing that a problem is in NC also allows to obtain a polynomial space algorithm for the case of exponential-size input computed by a PSPACE-transducer (as done, e.g., in [3]), which is not necessarily true for arbitrary problems solvable in polynomial time.

NC^2 is an especially important class in computational algebra. To quote [23, page 468], "NC^2 is the habitat of most natural problems in linear algebra". Indeed, matrix multiplication, computing the determinant, inverse and rank of a matrix belong to NC^2 [11, 18, 7, 19].

Our contributions

The known results about reachability properties of zero-one matrix monoids (including the special case of complete DFAs), such as [14, 20, 46, 34], mostly construct a product of minimum rank iteratively, with each iteration decreasing the number of different rows or the rank of a matrix. Such an approach is inherently sequential, since the matrix in the new iteration has to depend on the previous one, which thus has to be constructed explicitly. In particular, this requires matrix multiplication at every step, which heavily increases the time complexity. In this paper, we take a different direction by strongly relying on linear algebra. While linear-algebraic arguments are used widely in the synchronising automata literature, they mostly serve to decrease the number of iterations in the iterative scheme described above. Our approach is to instead relate the rank of a zero-one matrix monoid to efficiently computable linear-algebraic properties, without explicitly constructing a matrix of minimum rank.

Our first main result is that computing the rank of a zero-one matrix monoid provided in the input by a generating set of m matrices of dimension n (or, equivalently, by a UFA with n states and m letters) is in NC^2 (Theorem 18) and can be done in time $\mathcal{O}(mn^4)$ (Theorem 22). Previously, it was not known that this problem is in NC, not even for complete DFAs or finite complete codes. Moreover, the naive implementation of the polynomial time algorithm from the literature works in time $\mathcal{O}(n^{4+\omega} + mn^4)$ [14].

These results rely on a new concept of weight of the matrices in a complete zero-one monoid. This theory of matrix weight, which we develop in Section 4, is our main technical contribution. Matrix weight is a natural generalisation of an existing notion of weight of columns of matrices in complete DFAs, which was used, e.g., in connection with the road colouring problem [21, 29, 25]. We show that all matrices in a zero-one matrix monoid have the same weight, and that this weight is tightly related to both the rank of the monoid and to the maximal weight of the columns and rows of its matrices (Section 4.3). This connection allows us to reduce the computation of the monoid rank to the computation of maximal column and row weight. Then we show that we can instead compute the weight of "maximal pseudo-columns" and "maximal pseudo-rows", as they have the same weight as maximal columns and rows, respectively (Section 4.4). Finally, we transfer linear-algebraic techniques from the literature on weighted automata to compute those weights, and thus the rank of the monoid, efficiently (Section 5 and Section 6.2).

We complement the linear-algebraic algorithms with a combinatorial algorithm, our second main contribution. While the latter has higher time complexity of $\mathcal{O}(n^{2+\omega} + mn^4)$ in the general case (Theorem 23), it also constructs a matrix of minimum rank in addition to computing the rank of the monoid. For complete DFAs, our combinatorial algorithm runs in time $\mathcal{O}(n^3 + mn^2)$ (Theorem 24), thus outmatching the linear-algebraic counterpart and improving upon the $\mathcal{O}(m^4n^4)$ algorithm known before [43]. The two key technical ingredients of our combinatorial algorithm are explained in the beginnings of Section 6.3 and Section 6.4. Our results on the time complexity of computing the rank are summarised in the table below.

class	previous best	linear-algebraic algorithm	combinatorial algorithm
UFA	$\mathcal{O}(n^{4+\omega} + mn^4)$ [14]	$\mathcal{O}(mn^4)$ (Theorem 22)	$\mathcal{O}(n^{2+\omega} + mn^4)$ (Theorem 23)
complete DFA	$\mathcal{O}(m^4n^4)$ [43]	$\mathcal{O}(mn^3)$ (see Section 6.2)	$\mathcal{O}(n^3 + mn^2)$ (Theorem 24)

3 Main definitions

Let Q be a finite set, which we view as a set of states. For $S \subseteq Q$ we write $[S]$ for the column vector $x \in \{0,1\}^Q$ such that $x(q) = 1$ if and only if $q \in S$. We may write $[q]$ for $[\{q\}]$. For a column vector $x \in \{0,1\}^Q$ we write x^T for the transpose, a row vector. For two column vectors $x_1, x_2 \in \mathbb{R}^Q$ we write $x_1 \geq x_2$ if the inequality holds component-wise. We view the elements of $\mathbb{R}^{Q\times Q}$ (and similar sets) as matrices. Vector and matrix addition and multiplication are defined in the usual way (over $\mathbb{R}$). We denote by $\langle X \rangle$ the span of a set X of vectors, i.e., the set of all linear combinations of X with real coefficients. The *real rank* of a matrix $A \in \mathbb{R}^{Q\times Q}$ is, as usual, the dimension of the column space of A over the field of the reals (which equals the dimension of the row space); i.e., $\mathsf{rank}_{\mathbb{R}}(A) = \dim \langle A[q] \mid q \in Q \rangle = \dim \langle [q]^T A \mid q \in Q \rangle$.

Let $\mathcal{A} = \{A_1, \ldots, A_m\}$ be a set of matrices from $\{0,1\}^{Q\times Q}$, and $\Sigma = \{a_1, \ldots, a_m\}$ be a finite alphabet. We associate the letters with the matrices by setting $M(a_i) = A_i$ for $1 \leq i \leq m$. Throughout this paper, when speaking about computational complexity, we assume that the input is the function $M\colon \Sigma \to \{0,1\}^{Q\times Q}$ from letters to zero-one matrices. We can extend $M\colon \Sigma \to \{0,1\}^{Q\times Q}$ naturally (and often implicitly) to $M\colon \Sigma^* \to \mathbb{Z}_{\geq 0}^{Q\times Q}$ by defining $M(a_1 \cdots a_k) = M(a_1) \cdots M(a_k)$. Thus, M is a monoid homomorphism from Σ^* to the matrix monoid $M(\Sigma^*)$ generated by $\mathcal{A} = M(\Sigma)$. Note that $M(\varepsilon) = I$, where ε denotes the empty word and I the $Q \times Q$ identity matrix. In this paper, we consider only monoid morphisms $M\colon \Sigma^* \to \mathbb{Z}_{\geq 0}^{Q\times Q}$ that are *unambiguous*, i.e., $M\colon \Sigma^* \to \{0,1\}^{Q\times Q}$. If M is unambiguous, $\mathcal{A} = M(\Sigma)$ generates a finite matrix monoid $M(\Sigma^*) \subseteq \{0,1\}^{Q\times Q}$.

Viewing the matrices as transition matrices of an automaton, we obtain a *nondeterministic finite (semi-)automaton (NFA)* (Q, Σ, Δ) with transition relation $\Delta = \{(p, a, q) \in Q \times \Sigma \times Q \mid [p]^T M(a)[q] = 1\}$. Recall that in this paper automata do not have dedicated initial or accepting states, see footnote 2 on page 2. We can extend Δ from letters to words in the usual way so that we have $\Delta = \{(p, w, q) \in Q \times \Sigma^* \times Q \mid [p]^T M(w)[q] \geq 1\}$. An NFA (Q, Σ, Δ) is *unambiguous* (or *diamond-free*) if for every two states p, q and for every two words w_1, w_2 there exists at most one $t \in Q$ with $(p, w_1, t) \in \Delta$ and $(t, w_2, q) \in \Delta$; see Figure 2 for an illustration of the forbidden configuration. We denote unambiguous NFAs as UFAs. Recall from the previous section that deciding if an NFA is unambiguous is NL-complete. In the following, we often identify $M\colon \Sigma^* \to \{0,1\}^{Q\times Q}$ with the corresponding UFA (Q, Σ, Δ). In particular, a monoid homomorphism is unambiguous if and only if the corresponding NFA is unambiguous.

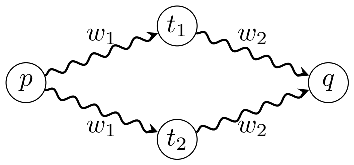


Figure 2 The configuration that is forbidden in a UFA.

When M (or, equivalently, Δ) is clear from the context, we may write $p \cdot w = \{q \in Q \mid (p, w, q) \in \Delta\}$. Then $[p \cdot w]^T = [p]^T M(w)$. Similarly, we may write $w \cdot q = \{p \in Q \mid (p, w, q) \in \Delta\}$, so that $[w \cdot q] = M(w)[q]$. We call M *strongly connected* if for all $p, q \in Q$ there is $w \in \Sigma^*$ with $p \cdot w \ni q$. We call M *complete* if $0 \notin M(\Sigma^*)$, where 0 is the zero matrix. The *real rank* of M (and of $M(\Sigma^*)$) is $\mathsf{rank}_{\mathbb{R}}(M) := \min\{\mathsf{rank}_{\mathbb{R}}(M(w)) \mid w \in \Sigma^*\}$. Note that M is complete if and only if $\mathsf{rank}_{\mathbb{R}}(M) \neq 0$.

Suppose that $|p \cdot a| = 1$ holds for every $p \in Q$ and $a \in \Sigma$, or, equivalently, that every matrix in $\mathcal{A}$ has exactly one 1 in each row. Then $|p \cdot w| = 1$ holds for every $p \in Q$ and $w \in \Sigma^*$. We call such UFAs *complete deterministic finite (semi-)automata (complete DFAs)* and we may write δ instead of Δ to highlight that it is a transition function $\delta\colon Q \times \Sigma \to Q$ instead of a transition relation. A complete DFA (Q, Σ, δ) is complete in the sense defined above (i.e., $0 \notin M(\Sigma^*)$), and for any $w \in \Sigma^*$ we have that $\mathsf{rank}_{\mathbb{R}}(M(w))$ is the number of nonzero columns in $M(w)$.

4 Main concepts and the linear algebra toolbox

In this section, we introduce the main tools that we will use for both linear-algebraic and combinatorial algorithms in later sections. Until Section 4.5, we fix an unambiguous, complete, and strongly connected monoid morphism M. In Section 4.5 we will show that the case where M is not strongly connected can be easily reduced to the strongly connected case.

4.1 Columns, rows and the structure of minimum rank matrices

The concept of maximum columns and rows plays a crucial role in dealing with reachability problems in unambiguous monoid morphisms. Abusing language slightly in the following, by *column* we refer to column vectors of the form $[w \cdot q] = M(w)[q] \in \{0,1\}^Q$ where $w \in \Sigma^*$ and $q \in Q$. Similarly, a *row* is of the form $[q \cdot w]^T = [q]^T M(w)$. See Figure 3 for an example. In the case of complete DFAs, all rows are of the form $[q]^T$. This fact makes complete DFAs significantly simpler to deal with than general complete UFAs.

$$M(a) = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad M(b) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

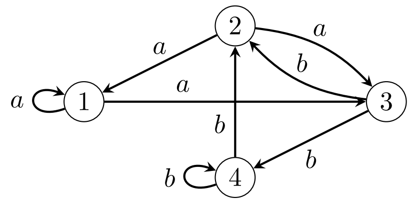


Figure 3 $[a \cdot 3] = M(a)[3] = [\{1,2\}]$ is a column; $[2 \cdot a]^T = [2]^T M(a) = [\{1,3\}]^T$ is a row.

A column $[C]$ is called *maximal* if there is no column $[C']$ such that $[C'] \neq [C]$ and $[C'] \geq [C]$ (that is, $C \subset C'$). Maximal rows are defined in the same way. Recall that the inequalities are taken component-wise.

Let $A \in \{0,1\}^{m \times n}$ be a zero-one matrix. One can view $\mathsf{rank}_{\mathbb{R}}(A)$ as the least number r such that there are matrices $C \in \mathbb{R}^{m \times r}$ and $R \in \mathbb{R}^{r \times n}$ with $A = CR$. Define the *unambiguous rank* $\mathsf{rank}_{\mathsf{un}}(A)$ as the least number r such that there are matrices $C \in \{0,1\}^{m \times r}$ and $R \in \{0,1\}^{r \times n}$ such that $A = CR$. Analogously to $\mathsf{rank}_{\mathbb{R}}(M)$, define also $\mathsf{rank}_{\mathsf{un}}(M) := \min\{\mathsf{rank}_{\mathsf{un}}(M(w)) \mid w \in \Sigma^*\}$. Clearly, $\mathsf{rank}_{\mathbb{R}}(A) \leq \mathsf{rank}_{\mathsf{un}}(A)$, and the inequality can be strict, but in Corollary 2 below we show that $\mathsf{rank}_{\mathbb{R}}(M) = \mathsf{rank}_{\mathsf{un}}(M)$. The reason we are interested in the unambiguous rank is that Theorem 1 below implies that there is always a matrix with a very simple structure such that its unambiguous rank is equal to its real rank and both ranks are minimum.

In the following let us write $r := \mathsf{rank}_{\mathsf{un}}(M)$ when M is understood. A word $u \in \Sigma^*$ is *of minimum unambiguous rank* if $\mathsf{rank}_{\mathsf{un}}(M(u)) = r$. If $u \in \Sigma^*$ is of minimum unambiguous rank then so is vuw for all $v, w \in \Sigma^*$.

Words of unambiguous rank one, known as synchronising words, play an especially important role due to their applications in the theory of codes, as explained in Section 2. It is easy to see that a word w has unambiguous rank one if and only if there exist $C, R \subseteq Q$ such that w maps a state p to a state q if and only if $p \in C$ and $q \in R$. For complete DFAs, we moreover have that $C = Q$ and R has cardinality one.

▶ **Theorem 1** (Césari [17]). *Let $u \in \Sigma^*$ be of minimum unambiguous rank. There are pairwise disjoint sets $C_1, \ldots, C_r \subseteq Q$ and pairwise disjoint sets $R_1, \ldots, R_r \subseteq Q$ such that*

$$M(u) = \sum_{i=1}^{r} [C_i][R_i]^T.$$

Moreover, each $[C_i]$ and $[R_i]^T$ is, respectively, a maximal column and a maximal row.

This theorem will play a central role. A proof can be found in [5, Proposition 4]. In the case of a complete DFA, R_1 is a singleton and Theorem 1 is fairly obvious.

In Theorem 1, since the C_i are pairwise disjoint and the R_i are pairwise disjoint, each $[C_i][R_i]^T$ forms, intuitively, a "combinatorial rectangle", and no such rectangle shares a row or a column with any other rectangle. The column vectors $[C_i]$ are exactly the nonzero columns of $M(u)$ and linearly independent, and the row vectors $[R_i]^T$ are exactly the nonzero rows of $M(u)$ and linearly independent. Thus, r is the number of distinct nonzero columns and also the number of distinct nonzero rows in $M(u)$. It follows that $r = \mathsf{rank}_{\mathsf{un}}(M(u)) = \mathsf{rank}_{\mathbb{R}}(M(u))$. Thus we have:

▶ **Corollary 2.** *We have $r = \mathsf{rank}_{\mathsf{un}}(M) = \mathsf{rank}_{\mathbb{R}}(M)$.*

We can thus define $\mathsf{rank}(M)$ as $\mathsf{rank}_{\mathsf{un}}(M) = \mathsf{rank}_{\mathbb{R}}(M)$. For words $w \in \Sigma^*$ that are not of minimum unambiguous rank, we may have $\mathsf{rank}_{\mathbb{R}}(M(w)) < \mathsf{rank}_{\mathsf{un}}(M(w))$, but the rank of such matrices will rarely play a role in the following. In what follows, we call words of minimum unambiguous rank simply words of minimum rank. Since we never refer to the real rank of words below, this will not lead to any confusion.

4.2 The weight of columns and rows

The results in this subsection, about the column and row vectors that appear in the matrices $M(w)$, are mostly due to [17]; see also [5, Section 3]. Since a notion of column and row weight will be crucial for us in the later development, we phrase and prove the results around these concepts, but we do not view the lemmas of this subsection as novel.

Define $\overline{A} = \frac{1}{|\Sigma|} \sum_{a \in \Sigma} M(a) \in [0,1]^{Q \times Q}$. Since M is strongly connected, $\overline{A}$ is irreducible. Since M is unambiguous, the spectral radius of $\overline{A}$ is at most 1, and since M is complete, it is at least 1. Thus, the spectral radius of $\overline{A}$ equals 1. Since $\overline{A}$ is irreducible, it follows from basic Perron-Frobenius theory that $\overline{A}$ has an eigenvalue 1 and every right eigenvector with eigenvalue 1 is a multiple of a strictly positive vector, say $\beta \in \mathbb{R}^Q_{>0}$. Since $\overline{A}$ has only rational entries, we can assume $\beta \in \mathbb{Q}^Q_{>0}$. Similarly for left eigenvectors. Therefore, there are $\alpha, \beta \in \mathbb{Q}^Q_{>0}$ with $\alpha^T \overline{A} = \alpha^T$ and $\overline{A}\beta = \beta$. Without loss of generality, we assume that $\alpha^T \beta = 1$.

In the complete DFA case, since $M(a)[Q] = [Q]$ for all $a \in \Sigma$, we have $\overline{A}[Q] = [Q]$ and so it is natural to take $\beta = [Q]$. In that case, $\alpha^T[Q] = \alpha^T \beta = 1$ means that $\alpha^T = \alpha^T \overline{A}$ is the (unique) stationary distribution of the Markov chain whose transition probabilities are given by the row-stochastic matrix $\overline{A}$; intuitively, in this Markov chain a letter $a \in \Sigma$ is picked uniformly at random in every step.

Define the *weight* of a column y and of a row x^T by $\alpha^T y \in \mathbb{R}$ and $x^T \beta \in \mathbb{R}$, respectively. Denote the maximum column weight and the maximum row weight by mcw and mrw, respectively, i.e.,

$$\mathsf{mcw} := \max\{\alpha^T y \mid y \text{ is a column}\} \quad \text{and} \quad \mathsf{mrw} := \max\{x^T \beta \mid x^T \text{ is a row}\}.$$

A column y is called *of maximum weight* if $\alpha^T y = \mathsf{mcw}$, and analogously for rows. In the complete DFA case, every row is of the form $[q]^T$ for some $q \in Q$, hence every row is of maximum weight.

▶ **Lemma 3.** *A column (respectively, row) is maximal if and only if it is of maximum weight.*

An important property that we will need later is that the set of maximal columns is closed under left multiplication by matrices from the monoid, as stated in the following lemma. Note that this is no longer true without the completeness assumption, and is the key reason why the case of complete matrix monoids is easier to deal with.

▶ **Lemma 4.** *Let $v \in \Sigma^*$ and $q \in Q$ be such that $[v \cdot q]$ is a maximal column. Then $[uv \cdot q]$ is a maximal column for all $u \in \Sigma^*$.*

The following lemma will be useful later to construct minimum rank matrices from maximal columns and rows.

▶ **Lemma 5.** *Let $w \in \Sigma^*$ be such that all non-zero columns and rows in $M(w)$ are maximal. Then ww is of minimum rank.*

4.3 Weight preservation property and minimum rank

Every word w of minimum rank in a complete DFA induces a partition of the state set into subsets of states mapped by w to the same state (that is, into columns). It was observed by Friedman in [21] that all sets of such a partition have the same weight. This observation has many applications to variations of the road colouring problem [21, 29, 25, 30]. Moreover, it was proved in [25, Theorem 6], again in connection with road colouring, that for every w the weights of all columns in $M(w)$ sum up to 1 (assuming $\beta = [Q]$ as suggested previously). This can be seen as a weight preservation property: the total weight of columns in the matrix of a word is preserved under multiplication by any matrix from the monoid. As a result we get that $1 = r \cdot \mathsf{mcw}$, and hence $r = \frac{1}{\mathsf{mcw}}$. The proof of the weight preservation property for complete DFAs is quite simple and relies on the fact that for a state q and a word w the set $q \cdot w$ is always a singleton. For complete UFAs this is no longer true; in particular, $q \cdot w$ can be the empty set, thus permanently "losing" some weight collected in q. Hence, a more refined property is required. The following result provides such a property. It also turns out that its proof requires more sophisticated techniques than in the complete DFA case.

▶ **Theorem 6.** *For all $w \in \Sigma^*$ we have $1 = \alpha^T M(w)\beta = r \cdot \mathsf{mcw} \cdot \mathsf{mrw}$.*

Similarly to the complete DFA case, this result allows us to reduce computing r to computing mcw and mrw, which we will use later in our algorithms. Recall that we have defined α^T and β so that $\alpha^T\beta = 1$.

Towards a proof of Theorem 6 we first prove the following lemma.

▶ **Lemma 7.** *Let $u \in \Sigma^*$ be of minimum rank. Then $\alpha^T M(u)\beta = r \cdot \mathsf{mcw} \cdot \mathsf{mrw}$.*

Proof. Let $M(u) = \sum_{i=1}^r [C_i][R_i]^T$ be as in Theorem 1. Each $[C_i]$ and each $[R_i]$ is of maximum weight. Thus,

$$\alpha^T M(u)\beta = \sum_{i=1}^r \alpha^T [C_i][R_i]^T\beta = \sum_{i=1}^r \mathsf{mcw} \cdot \mathsf{mrw} = r \cdot \mathsf{mcw} \cdot \mathsf{mrw}\,. \qquad ◀$$

We also need the following proposition.

▶ **Proposition 8.** *Let $x \in \mathbb{R}^Q$ and $c \in \mathbb{R}$ be such that $x^T M(u)\beta = c$ holds for all $u \in \Sigma^*$ of minimum rank. Then $x^T M(w)\beta = c$ holds for all $w \in \Sigma^*$.*

Proof. Let $w \in \Sigma^*$. Let $u \in \Sigma^*$ be of minimum rank. Recall that every word that contains u as a factor is of minimum rank. For every $k \geq 0$, partition Σ^k into sets $W_0(k)$ and $W_1(k)$ so that $W_0(k) = \Sigma^k \cap (\Sigma^* u \Sigma^*)$ and $W_1(k) = \Sigma^k \setminus (\Sigma^* u \Sigma^*)$; i.e., $W_0(k), W_1(k)$ are the sets of length-k words that do or do not contain u as a factor, respectively. For all $v \in W_0(k)$ both v and wv are of minimum rank. Thus, we have $x^T M(wv)\beta = c$ for all $v \in W_0(k)$. It follows that

$$\sum_{v \in W_0(k)} \frac{x^T M(wv)\beta}{|W_0(k)|} \;=\; c \quad \text{for all } k \geq 0. \tag{1}$$

Let $d > 0$ be such that $|x^T A\beta| \leq d$ for all $A \in \{0,1\}^{Q \times Q}$. Then we have

$$\sum_{v \in W_1(k)} \frac{|x^T M(wv)\beta|}{|W_1(k)|} \;\leq\; d \quad \text{for all } k \geq 0. \tag{2}$$

Let $m \geq 0$. Define $p_1(m) := \frac{|W_1(m|u|)|}{|\Sigma|^{m|u|}}$. We can view $p_1(m)$ as the probability of picking a word in $W_1(m|u|)$ when a word of length $m|u|$ is picked uniformly at random. We have $p_1(m) \leq \left(1 - \frac{1}{|\Sigma|^{|u|}}\right)^m$, as in order to avoid u as a factor, it has to be avoided in each of the m consecutive blocks of length $|u|$. Thus, $\lim_{m\to\infty} p_1(m) = 0$. We have

$$\begin{aligned} x^T M(w)\beta \;&=\; x^T M(w)\overline{A}\beta \;=\; x^T M(w)\overline{A}^{m|u|}\beta \;=\; \frac{1}{|\Sigma|^{m|u|}} \sum_{v \in \Sigma^{m|u|}} x^T M(wv)\beta \\ &=\; \frac{|W_0(m|u|)|}{|\Sigma|^{m|u|}} \sum_{v \in W_0(m|u|)} \frac{x^T M(wv)\beta}{|W_0(m|u|)|} + \frac{|W_1(m|u|)|}{|\Sigma|^{m|u|}} \sum_{v \in W_1(m|u|)} \frac{x^T M(wv)\beta}{|W_1(m|u|)|} \\ &=\; (1 - p_1(m)) \cdot c + p_1(m) \cdot \sum_{v \in W_1(m|u|)} \frac{x^T M(wv)\beta}{|W_1(m|u|)|} \qquad \text{(by Equation (1)).} \end{aligned}$$

With Equation (2) it follows that $|x^T M(w)\beta - c| \;\leq\; p_1(m)(|c| + d)$. Since this holds for all $m \geq 0$ and $\lim_{m\to\infty} p_1(m) = 0$, we conclude that $x^T M(w)\beta = c$. ◀

Now we prove Theorem 6.

Proof of Theorem 6. It follows from Lemma 7 and Proposition 8 that

$$\alpha^T M(w)\beta \;=\; r \cdot \mathsf{mcw} \cdot \mathsf{mrw} \qquad \text{for all } w \in \Sigma^*.$$

With $w = \varepsilon$ we obtain $1 = \alpha^T\beta = \alpha^T M(\varepsilon)\beta = r \cdot \mathsf{mcw} \cdot \mathsf{mrw}$, as required. ◀

4.4 Maximal pseudo-columns

In this subsection, we define maximal pseudo-columns, which are vectors that can be seen as a relaxation of the notion of maximal columns. We show that the weight of a maximal pseudo-column is equal to the weight of a maximal column, and a maximal pseudo-column is a solution of a system of linear equations, and thus can be computed efficiently. By invoking Theorem 6, this will allow us to efficiently compute r.

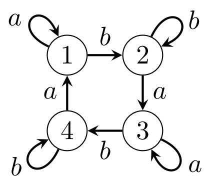
Denote by $\mathsf{MCol} \subseteq \{0,1\}^Q$ the set of maximal columns. By Theorem 1 (bearing in mind also Corollary 2), the vector space spanned by all maximum columns, $\langle \mathsf{MCol} \rangle$, is at least r-dimensional:

▶ **Proposition 9.** *We have $r \le \dim \langle \mathsf{MCol} \rangle$.*

One might hypothesise that $r = \dim \langle \mathsf{MCol} \rangle$ or even that all minimum-rank matrices have the same r nonzero (hence, maximum) columns. The following example shows that neither is the case in general, not even for complete DFAs.

▶ **Example 10.** Consider the complete DFA with $\Sigma = \{a, b\}$ and

$$M(a) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad M(b) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



By symmetry, we have $\alpha^T = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$. Since no word maps states 1 and 3 to the same state, we have $r = 2$; i.e., a and b are both minimum rank. Further, MCol consists exactly of the four nonzero columns in $M(a)$ and $M(b)$. Their span $\langle \mathsf{MCol} \rangle$ is 3-dimensional, as $\begin{pmatrix} 1 & -1 & 1 & -1 \end{pmatrix}$ is orthogonal to each maximum column. Thus, $r = 2 < 3 = \dim \langle \mathsf{MCol} \rangle < 4 = |\mathsf{MCol}|$.

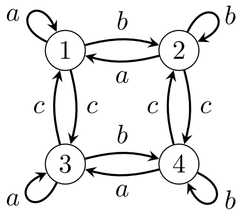
Define the vector space $U := \langle \alpha^T M(w) - \alpha^T \mid w \in \Sigma^* \rangle$. Intuitively, it is the set of all differences of weight distributions over the states before and after a word is applied. Notice that for all $w_1, w_2 \in \Sigma^*$ we have $\alpha^T M(w_1) - \alpha^T M(w_2) \in U$. Later (see the proof of Lemma 19 below) we show that U is closed under post-multiplication with $M(a)$ for all $a \in \Sigma$. Such "forward spaces" play an important role in weighted automata; see, e.g., [32]. Denote the orthogonal complement of U by $U^\perp$; i.e., $U^\perp = \{y \in \mathbb{R}^Q \mid \forall w \in \Sigma^* : \ \alpha^T M(w) y = \alpha^T y\}$. Intuitively, it is the set of vectors whose weight does not change under pre-multiplication with $M(w)$ for any w (where by the weight of a vector y we understand $\alpha^T y$). Clearly, $\dim U + \dim U^\perp = |Q|$. The following proposition follows immediately from Lemma 4.

▶ **Proposition 11.** *We have $\mathsf{MCol} \subseteq U^\perp$.*

It follows that $\langle \mathsf{MCol} \rangle$ is a subspace of $U^\perp$. With Proposition 9, we have $r \le \dim \langle \mathsf{MCol} \rangle \le \dim U^\perp$. One might hypothesise that $\langle \mathsf{MCol} \rangle = U^\perp$. The following example shows that this is not the case in general, not even for complete DFAs.

▶ **Example 12.** Consider the DFA with $\Sigma = \{a, b, c\}$ and

$$M(a) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, M(b) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, M(c) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$



We have $\mathsf{Mer}(1) = \mathsf{Mer}(2) = \{1,2\}$ and $\mathsf{Mer}(3) = \mathsf{Mer}(4) = \{3,4\}$. Thus, $\mathsf{MCol} = \{\begin{pmatrix}1 & 1 & 0 & 0\end{pmatrix}^T, \begin{pmatrix}0 & 0 & 1 & 1\end{pmatrix}^T\}$. Hence, $\dim\langle\mathsf{MCol}\rangle = 2$.

On the other hand, by symmetry we have $\alpha^T = \begin{pmatrix}\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4}\end{pmatrix}$. For any $w \in \Sigma^*$,

$$\alpha^T M(w) \; = \; \begin{cases} \begin{pmatrix}\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4}\end{pmatrix} & \text{if } w \in \{c\}^* \\ \begin{pmatrix}\frac{1}{2} & 0 & \frac{1}{2} & 0\end{pmatrix} & \text{if the last non-}c\text{ letter in } w \text{ is } a \\ \begin{pmatrix}0 & \frac{1}{2} & 0 & \frac{1}{2}\end{pmatrix} & \text{if the last non-}c\text{ letter in } w \text{ is } b\,. \end{cases}$$

It follows that $U = \langle\begin{pmatrix}1 & -1 & 1 & -1\end{pmatrix}\rangle$. Thus, $\dim U^\perp = 4 - 1 = 3 > 2 = \dim\langle\mathsf{MCol}\rangle$, and $\langle\mathsf{MCol}\rangle$ is a strict subspace of $U^\perp$. For example, the vector $\begin{pmatrix}1 & 0 & 0 & 1\end{pmatrix}^T$ is in $U^\perp$ but not in $\langle\mathsf{MCol}\rangle$.

Although the dimension of $U^\perp$ does not generally equal r, the vector space $U^\perp$ turns out useful for computing r. Recall that, by Theorem 6, we can obtain r by computing mcw (and, symmetrically, mrw). For $q \in Q$ define

$$\mathsf{Mer}(q) \;:=\; \{q' \in Q \mid \exists\, S \supseteq \{q, q'\} \text{ such that } [S] \text{ is a column}\}\,.$$

Intuitively, $\mathsf{Mer}(q)$ consists of the states that can "appear" in a column together with q, or, equivalently, the states that are "mergeable" with q (that is, can be mapped to the same state by a word). Note that $q \in \mathsf{Mer}(q)$. We will need the following lemma which is easy to prove.

▶ **Lemma 13.** *Let $v \in \Sigma^*$ and $q \in Q$ be such that $[v \cdot q]$ is a maximal column. Then $v \cdot q' = \emptyset$ holds for all $q' \in \mathsf{Mer}(q) \setminus \{q\}$.*

We call a vector $y \in U^\perp$ a *maximal pseudo-column* if there is $q \in Q$ with $y(q) = 1$ and $y(q') = 0$ for all $q' \notin \mathsf{Mer}(q)$. This notion, which is closely related to the "pseudo-cuts" from [35], can be seen as a relaxation of the notion of a maximal column: clearly, every maximal column is a maximal pseudo-column, but the converse is not true, since a maximal pseudo-column is not necessarily a vector over $\{0,1\}$, let alone a *column* in the strict sense, i.e., of the form $[w \cdot p]$. The following lemma however shows that the weight of a maximal pseudo-column is equal to the weight of a maximal column. We will later show that computing the former can be done in NC^2.

▶ **Lemma 14.** *Let y be a maximal pseudo-column. Then $\alpha^T y = \mathsf{mcw}$.*

Proof. Let $q \in Q$ be such that $y(q) = 1$ and $y(q') = 0$ for all $q' \notin \mathsf{Mer}(q)$. Let $w \in \Sigma^*$ be such that $[w \cdot q]$ is a maximal column. We have

$$\begin{aligned} \alpha^T y &= \alpha^T M(w) y && (y \in U^\perp) \\ &= \sum_{q' \in Q} y(q') \alpha^T [w \cdot q'] \\ &= \sum_{q' \in \mathsf{Mer}(q)} y(q') \alpha^T [w \cdot q'] && (y(q') = 0 \text{ for } q' \notin \mathsf{Mer}(q)) \\ &= \alpha^T [w \cdot q] + \sum_{q' \in \mathsf{Mer}(q) \setminus \{q\}} y(q') \alpha^T [w \cdot q'] && (y(q) = 1) \\ &= \alpha^T [w \cdot q] && (\text{Lemma 13}) \\ &= \mathsf{mcw} && (\text{by the choice of } w, q). \end{aligned}$$

◀

▶ **Example 15.** We continue Example 10. We have $U = \langle \begin{pmatrix} 1 & -1 & 1 & -1 \end{pmatrix} \rangle$ and $\mathsf{Mer}(2) = \{1, 2, 3\}$. Let $y = \begin{pmatrix} 4/3 & 1 & -1/3 & 0 \end{pmatrix}^T$. Then $y \in U^\perp$. Since $y(2) = 1$ and $y(4) = 0$, vector y is a maximal pseudo-column. Thus, by Lemma 14, $\mathsf{mcw} = \alpha^T y = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} y = \frac{1}{2}$.

▶ **Theorem 16.** *Let Γ be a basis of U, and let $q \in Q$. Then the following linear system for $y \in \mathbb{R}^Q$ has a solution, and all its solutions are maximal pseudo-columns:*

$$\begin{aligned} \gamma^T y &= 0 \quad \text{for all } \gamma^T \in \Gamma \\ y(q) &= 1 \\ y(q') &= 0 \quad \text{for all } q' \notin \mathsf{Mer}(q)\,. \end{aligned}$$

Proof. By Proposition 11, any maximal column solves the linear system. Let $y \in \mathbb{R}^Q$ be a solution of the linear system. The equations on the first line guarantee that $y \in U^\perp$. Then, the equations on the second and third line guarantee that y is a maximal pseudo-column. ◀

4.5 Dealing with the non-strongly connected case

The following lemma shows that in order to compute the minimum rank we can focus on the strongly connected case.

▶ **Proposition 17.** *Let $M : \Sigma \to \{0,1\}^{Q \times Q}$ be an unambiguous matrix monoid morphism. Suppose that $Q_1 \cup Q_2 = Q$ is a partition of Q such that for all $w \in \Sigma^*$ it holds that $[Q_2]^T M(w)[Q_1] = 0$; i.e., for all $w \in \Sigma^*$ matrix $M(w)$ has the block form $M(w) = \begin{pmatrix} M_1(w) & M_{12}(w) \\ 0 & M_2(w) \end{pmatrix}$, where $M_1(w) \in \{0,1\}^{Q_1 \times Q_1}$ and $M_{12}(w) \in \{0,1\}^{Q_1 \times Q_2}$ and $M_2(w) \in \{0,1\}^{Q_2 \times Q_2}$. We have $\mathsf{rank}_{\mathbb{R}}(M) = \mathsf{rank}_{\mathbb{R}}(M_1) + \mathsf{rank}_{\mathbb{R}}(M_2)$ and $\mathsf{rank}_{\mathsf{un}}(M) = \mathsf{rank}_{\mathsf{un}}(M_1) + \mathsf{rank}_{\mathsf{un}}(M_2)$.*

By a straightforward induction it follows from Proposition 17 that the minimum rank of an unambiguous matrix monoid is the sum of the minimum ranks of its strongly connected components (where "incomplete" components count as having rank 0).

5 Computing the rank in NC^2

In this section, we prove our first main result, which is as follows.

▶ **Theorem 18.** *The problem of computing the (real) rank of an unambiguous matrix monoid is in NC^2.*

In order to use Theorem 16, we need the following lemma. We use the notation defined in the previous section. Recall that we defined $U := \langle \alpha^T M(w) - \alpha^T \mid w \in \Sigma^* \rangle$.

▶ **Lemma 19.** *If M is strongly connected, one can compute a basis of U in NC^2.*

For each $a \in \Sigma$ define $M'(a) := M(a) - I \in \{-1,0,1\}^{Q \times Q}$ and extend M' to $M' : \Sigma^* \to \mathbb{Z}^{Q \times Q}$ by defining $M'(a_1 \cdots a_k) = M'(a_1) \cdots M'(a_k)$. Define $U' := \langle \alpha^T M'(w) \mid w \in \Sigma^+ \rangle$. Note that here w ranges over Σ^+, i.e., nonempty words, only. By definition, U' is closed under right multiplication by $M'(a)$ for all $a \in \Sigma$. We first show the following lemma.

▶ **Lemma 20.** *We have $U = U'$.*

Proof. For the inclusion $U \subseteq U'$, we prove by induction on $i \geq 0$ that for all length-i words $w \in \Sigma^i$ we have $\alpha^T(M(w) - I) \in U'$. Concerning the induction base, $i = 0$, we have $\alpha^T(M(\varepsilon) - I) = 0 \in U'$. Concerning the induction step, let $i \geq 0$, and let $w \in \Sigma^i$ and $a \in \Sigma$. We have

$$\begin{aligned}\alpha^T(M(wa) - I) &= \alpha^T(M(w) - I)M(a) + \alpha^T(M(a) - I)\\ &= \alpha^T(M(w) - I)(M(a) - I) + \alpha^T(M(w) - I) + \alpha^T(M(a) - I)\\ &= \alpha^T(M(w) - I)M'(a) + \alpha^T(M(w) - I) + \alpha^T M'(a)\,.\end{aligned}$$

It holds that $\alpha^T M'(a) \in U'$, and, by the induction hypothesis, $\alpha^T(M(w) - I) \in U'$. It follows that $\alpha^T(M(wa) - I) \in U'$.

For the converse, $U' \subseteq U$, we proceed similarly by induction. Concerning the induction base, $i = 1$, for all $a \in \Sigma$ we have $\alpha^T M'(a) = \alpha^T(M(a) - I) \in U$. Concerning the induction step, let $i \geq 1$, and let $w \in \Sigma^i$ and $a \in \Sigma$. By the induction hypothesis there are $n \leq |Q|$ and $w_1, \ldots, w_n \in \Sigma^*$ and $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$ such that $\alpha^T M'(w) = \sum_{i=1}^n \lambda_i \alpha^T(M(w_i) - I)$. Thus, we have

$$\begin{aligned}\alpha^T M'(wa) &= \alpha^T M'(w)(M(a) - I) = \sum_{i=1}^n \lambda_i \alpha^T(M(w_i) - I)(M(a) - I)\\ &= \sum_{i=1}^n \lambda_i \alpha^T\big((M(w_i a) - I) - (M(a) - I) - (M(w_i) - I)\big) \in U\,,\end{aligned}$$

as required. ◀

Proof of Lemma 19. For each $a \in \Sigma$ define $U'_a := \langle \alpha^T M'(a)M'(w) \mid w \in \Sigma^* \rangle$. Using the technique from [33, Section 4.2] (see [32, Proposition 5.2] for a clearer explanation), for each $a \in \Sigma$ one can compute[3] a basis of U'_a in NC^2. The union of these bases, say $\Gamma = \{\gamma_1^T, \ldots, \gamma_n^T\}$ for some $n \leq |\Sigma||Q|$, spans U', which equals U by Lemma 20. To shrink Γ to a basis of U, for each $i \in \{1, \ldots, n\}$ include γ_i^T in the basis if and only if $\dim \langle \gamma_1^T, \ldots, \gamma_{i-1}^T \rangle < \dim \langle \gamma_1^T, \ldots, \gamma_i^T \rangle$. The latter (rank) computation can be done in NC^2 [27]. ◀

Now we can prove Theorem 18.

Proof of Theorem 18. Let $M : \Sigma \to \{0,1\}^{Q \times Q}$ be an unambiguous monoid morphism. Its strongly connected components can be computed in $\mathsf{NL} \subseteq \mathsf{NC}^2$. It follows from the proof of [34, Proposition 3] that one can check each component for completeness in NC^2, since a zero-one monoid contains the zero matrix if and only if the joint spectral radius of the set of its generators is strictly less than one [34]. Therefore, using Proposition 17, we can assume in the rest of the proof that M is complete and strongly connected.

We use the fact that one can compute a solution of a possibly singular linear system of equations in NC^2 [11, Section 5]. First, compute in NC^2 vectors $\alpha, \beta \in \mathbb{Q}_{>0}^Q$ with $\alpha^T \overline{A} = \alpha^T$ and $\overline{A}\beta = \beta$ and $\alpha^T \beta = 1$. Using Lemma 19 compute in NC^2 a basis of U. Choose an arbitrary $q \in Q$ and compute $\mathsf{Mer}(q)$ in $\mathsf{NL} \subseteq \mathsf{NC}^2$ with a reachability analysis. Then, solve the linear system from Theorem 16 to compute in NC^2 a maximal pseudo-column $y \in \mathbb{Q}^Q$. Hence, using Lemma 14, we can compute $\mathsf{mcw} = \alpha^T y$ in NC^2. Symmetrically, we can compute mrw in NC^2. Finally, by Theorem 6, we can compute $r = \frac{1}{\mathsf{mcw} \cdot \mathsf{mrw}}$ in NC^2. ◀

[3] In [33, 32] only membership in NC is claimed, but the bottleneck computations are matrix powering and rank computation, which can in fact be done in DET $\subseteq \mathsf{NC}^2$; see [18]. The computations in [32, Proposition 5.2] are on polynomially larger matrices, but this does not impact the membership in NC^2, as $\log^2(\mathit{poly}(n)) = O(\log^2(n))$.

▶ **Example 21.** We continue Examples 10 and 15. Since M is a complete DFA, it is natural to take $\beta = [Q]$. Note that $\alpha^T \beta = 1$. Since every row is of the form $[q]^T$ for some q, we have $\mathsf{mrw} = 1$. Recall from Example 15 that $\mathsf{mcw} = \frac{1}{2}$. With Theorem 6 we conclude that $r = \frac{1}{\mathsf{mcw} \cdot \mathsf{mrw}} = 2$, as observed in Example 10.

6 Time and space complexity

In this section, we study the time and space complexity of computing the rank of a zero-one matrix monoid and finding a matrix of minimum rank in it. We provide two approaches. The first one relies on the linear-algebraic tools developed in Section 4. It turns out to be faster than the second, combinatorial, approach, but is limited to only computing the rank. This is due to the fact that we never explicitly construct a maximal column in this approach, which is required in order to find a matrix of minimum rank by Theorem 1. Moreover, it is not known if one can find a maximal column in NC. In contrast, the combinatorial approach explicitly constructs a matrix of minimum rank step by step. In a way, this is exactly the reason why it is slower than the linear-algebraic approach, since this requires performing a large number of matrix multiplications. In the complete DFAs case, where direct matrix multiplication can be avoided due to the special shape of transformation matrices, it becomes much more efficient, and outmatches its linear-algebraic counterpart by a factor of the alphabet size.

The three main results of this section are as follows.

▶ **Theorem 22.** *The (real) rank of an n-state UFA over an alphabet of size m can be computed in $\mathcal{O}(mn^4)$ time and $\mathcal{O}(n^2)$ space.*

▶ **Theorem 23.** *A matrix of minimum (real) rank in an n-state UFA over an alphabet of size m can be found in $\mathcal{O}(n^{2+\omega} + mn^4)$ time and $\mathcal{O}(n^3)$ space.*

▶ **Theorem 24.** *A matrix of minimum (real) rank in an n-state complete DFA over an alphabet of size m can be found in $\mathcal{O}(n^3 + mn^2)$ time and $\mathcal{O}(n^2)$ space.*

Until the end of the section, fix a strongly connected complete UFA $\mathcal{A} = (Q, \Sigma, \Delta)$. Denote $n = |Q|$, $m = |\Sigma|$. Section 4.5 shows that strong connectivity can be assumed without loss of generality.

6.1 Square automaton and square digraph

We will need the construction of the square automaton of an NFA. The *square automaton* $\mathcal{A}^{(2)} = (Q^{(2)}, \Sigma, \Delta^{(2)})$ of $\mathcal{A}$ is defined as follows. Let $Q^{(2)} = \{(p,q) \mid p, q \in Q\}$, and for $p, q \in Q$ and $a \in \Sigma$, the transitions are defined component-wise, that is,

$$\Delta^{(2)} = \{((p,q), a, (p',q')) \in Q^{(2)} \times \Sigma \times Q^{(2)} \mid (p,a,p'), (q,a,q') \in \Delta\}.$$

Note that the square automaton of a complete DFA is also a complete DFA.

We call states of the form (q,q) in $\mathcal{A}^{(2)}$ *singletons*. Observe that the restriction of $\mathcal{A}^{(2)}$ to singletons is equal to $\mathcal{A}$. We denote by $G^{(2)} = (V^{(2)}, E^{(2)})$ the underlying digraph of $\mathcal{A}^{(2)}$ obtained by forgetting the labels of the transitions. Note that $|E^{(2)}| = \mathcal{O}(mn^4)$, and there exists an infinite series of complete UFAs over a two-letter alphabet with $|E^{(2)}| = \Theta(n^4)$ [36, Appendix A]. If $\mathcal{A}$ is a complete DFA, then $|E^{(2)}| = mn^2$.

6.2 Minimum rank in $\mathcal{O}(mn^4)$ time

We now perform the steps of the linear-algebraic algorithm described in Section 5, but implement them efficiently in terms of time and space complexity.

▶ **Lemma 25.** *For a state p, the set* $\mathsf{Mer}(p)$ *can be computed in $\mathcal{O}(mn^4)$ time and $\mathcal{O}(n^2)$ space.*

Proof. Perform a multi-source backwards digraph search starting from all singletons in $G^{(2)}$ and label all states $q \in Q$ such that (p, q) or (q, p) is visited during this search. This search can be performed in time linear in the number of edges of $|E^{(2)}|$, and $|E^{(2)}| = \mathcal{O}(mn^4)$. Observe that only the vertices of this digraph have to be stored in the memory explicitly, since the edges can be computed on the fly from the input without increasing the time complexity, hence the space complexity of the algorithm is $\mathcal{O}(n^2)$. ◀

▶ **Lemma 26.** *A maximal pseudo-column can be found in $\mathcal{O}(mn^4)$ time and $\mathcal{O}(n^2)$ space.*

Proof. To use Theorem 16 we first need to set up the linear system of equations described there. The average matrix $\overline{A}$ can be computed in time $\mathcal{O}(mn^2)$. The weight vectors $\alpha, \beta \in \mathbb{Q}^Q$ can then be computed in time $\mathcal{O}(n^3)$ by solving a system of linear equations. Then, using Lemma 25, we compute in $\mathcal{O}(mn^4)$ time the set $\mathsf{Mer}(p)$ for some state p. We also need to compute a basis of the vector space U defined in Section 4.4. As in the proof of Lemma 19, we compute a basis of $U = U' = \langle \alpha^T M'(w) \mid w \in \Sigma^+ \rangle$, which is the smallest vector space that contains $\alpha^T M(a)$ for all $a \in \Sigma$ and is closed under post-multiplication with $M(a)$ for all $a \in \Sigma$. This can be done in $\mathcal{O}(mn^3)$ time and $\mathcal{O}(n^2)$ space using a worklist algorithm and keeping a basis in echelon form using Gaussian elimination, as described, e.g., in [32, Section 2]. Finally, we solve the system of linear equations from Theorem 16, which can be done in $\mathcal{O}(n^3)$ time. Each step requires at most $\mathcal{O}(n^2)$ space. ◀

By Lemma 14, we thus get that mcw and mrw can be computed in $\mathcal{O}(mn^4)$ time and $\mathcal{O}(n^2)$ space, which together with Theorem 6 proves Theorem 22. We remark that for complete DFAs the proof of Lemma 26 gives $\mathcal{O}(mn^3)$ time for computing the rank. The combinatorial algorithm provided below will improve it to $\mathcal{O}(n^3 + mn^2)$ (see Section 6.5), while additionally finding a matrix of minimum rank.

6.3 Efficiently constructing a maximal column

We now consider a more general problem of finding a matrix of minimum rank in a zero-one matrix monoid. As mentioned above, by Theorem 1 we have to explicitly construct a maximal column for that, which turns out to be a more difficult task in terms of the time complexity. We will see in the next subsection that we only need one maximal column (together with a word constructing it) to get a matrix of minimum rank. The goal of this subsection is thus to show how to efficiently compute a short representation of a word constructing a maximal column, since the word itself may be longer than the time complexity we are aiming for. We do so by reusing repeating subwords in such a word, and describing their occurrences with a straight line program.

In [20, Section 5], an algorithm for constructing a word of rank one in complete DFAs in $\mathcal{O}(m^3 + mn^2)$ time and $\mathcal{O}(n^2)$ space was suggested. Our approach for finding a maximal column and a word constructing it follows a similar direction, with a few key differences. Firstly, we observe that it is enough to only compute words merging states with one chosen state p, instead of finding all pairs of mergeable states. This both simplifies the algorithm (in [20] an additional step is required to decrease the space complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$, which we get for free) and allows to present our results in terms of straight line programs, giving a better insight into the regularities present in the constructed word of minimum rank.

We define set straight line programs (set-SLPs), which are just SLPs with multiple initial symbols, and thus encode a set of words instead of one word. Formally, a *set-SLP* is a tuple $(\mathcal{V}, \Sigma, R, \mathcal{S})$, where $\mathcal{V}$ and Σ are disjoint finite sets of *nonterminal* and *terminal* symbols respectively, $R\colon \mathcal{V} \to (\mathcal{V} \cup \Sigma)^*$ is a function defining a derivation rule for each nonterminal symbol, and $\mathcal{S} \subseteq \mathcal{V}$ is a set of *initial* symbols. For $v \in \mathcal{V}$, we write $R(v)$ as $v \to w$ with $w \in (\mathcal{V} \cup \Sigma)^*$, and we call v and w the left- and right-hand sides of this derivation rule respectively. The *length* of a set-SLP is the total length of the right-hand sides of all the derivation rules. The semantics of a set-SLP is defined as follows. Given an initial symbol $s \in \mathcal{S}$, we recursively replace each symbol in $R(s)$ with the right-hand side of its derivation rule until we obtain a word over Σ, which is called *the word encoded by s*. We require that each initial symbol produces a unique word over Σ as a result of such derivation. Namely, we require that there exists a total linear order $\leq$ on the set $\mathcal{V}$ such that for all v with $v \to w$, w does not contain $v' \in \mathcal{V}$ with $v' \leq v$. The (multi-)set of words encoded by all initial symbols is called *the set of words encoded by* a set-SLP.

▶ **Example 27.** Consider a set-SLP $(\{w_1, w_3, w_5, u_1, u_2, u_3\}, \{a, b\}, R, \{w_1, w_3, w_5\})$ with

$$w_1 \to u_1, w_3 \to u_3 u_2, w_5 \to u_2, u_1 \to aab, u_2 \to aab, u_3 \to aaba.$$

This set-SLP encodes the (multi-)set $\{aab, aabaaab, aab\}$, and illustrates the reason why we are using set-SLPs: they allow to construct sets of words out of smaller "pieces" without having to explicitly repeat these "pieces" multiple times (in our example, we are reusing u_2). Note that the set-SLPs that we construct below only encode sets of words whose total length is polynomial in the size of the set-SLPs.

▶ **Lemma 28.** *Given a state p, a set-SLP of length $\mathcal{O}(n^2)$ defining a set $\{w_q \mid q \in \mathsf{Mer}(p)\}$, where w_q is a word with $p \in p \cdot w_q$ and $p \in q \cdot w_q$, can be computed in $\mathcal{O}(mn^4)$ time and $\mathcal{O}(n^2)$ space.*

Proof sketch. Call vertices (p, q) with $q \in \mathsf{Mer}(p)$ *merging*. The idea is to construct, by a digraph search of $G^{(2)}$, a directed tree T rooted in (p, p) and containing a path from each merging vertex to the root, and then use the joint subpaths of these paths in the tree to obtain a short set-SLP describing these paths. See Figure 5 for an example. ◀

▶ **Lemma 29.** *For a given state p of $\mathcal{A}$, an SLP of length $\mathcal{O}(n^2)$ encoding a word w such that $[w \cdot p]$ is a maximal column can be computed in $\mathcal{O}(n^{2+\omega} + mn^4)$ time and $\mathcal{O}(n^3)$ space.*

Proof sketch. Compute the matrices of the words encoded by the set-SLP from Lemma 28. We rely on the property, already used in some form in [14], that if for all $q \in \mathsf{Mer}(p)$, $q \neq p$, the vector $[w \cdot q]$ is zero, then $[w \cdot p]$ is a maximal column. To construct a word with this property, we iteratively concatenate the words w_q depending on nonzero columns in the matrix in each iteration. The number of iterations is bounded by n. ◀

6.4 Finding a matrix of minimum rank

We now use the results of the previous section to construct a matrix of minimum rank. The key idea is as follows: since the set of maximal columns is stable under left multiplication by matrices from the monoid, we can iteratively make each column of the matrix maximal or

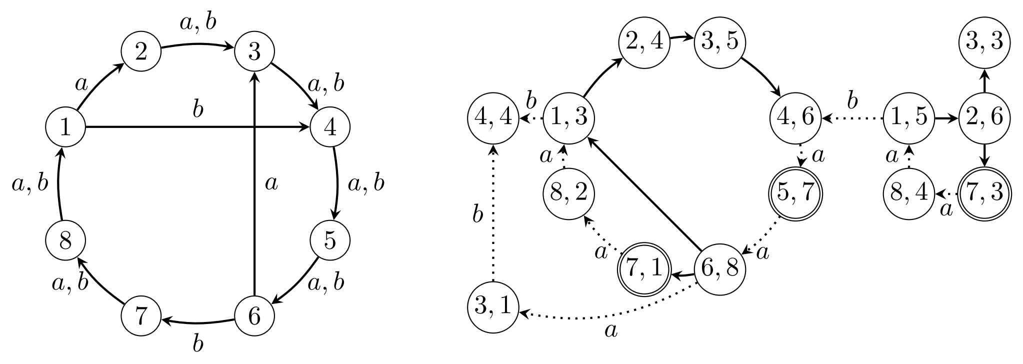


Figure 5 An example of $\mathcal{A}$ (left), and a part of the underlying digraph $G^{(2)}$ of its square automaton (right). Merging vertices are doubly circled. The edges of T for $p = 7$ are represented by dotted edges, and these edges are labelled with one of the letters labelling the corresponding transition in $\mathcal{A}^{(2)}$. Furthermore, we have $\rho_1 = (7,1) \to (8,2) \to (1,3) \to (4,4)$, $\rho_2 = (5,7) \to (6,8) \to (3,1) \to (4,4)$, $\rho_3 = (7,3) \to (8,4) \to (1,5) \to (4,6) \to (5,7)$. A set-SLP encoding the labels of these path is presented in Example 27, with u_i labelling the path ρ_i, $i \in \{1,2,3\}$.

zero by, intuitively, applying the same word (together with a short "reachability" word) to a state in each column. This simple observation significantly decreases the time complexity of our algorithm compared to the naive implementation of the algorithm from [14] constructing a word of minimum rank. Indeed, in the approach of [14], the word is constructed letter by letter, and requires to know the result of applying the last letter at every step. Since the constructed word has length $\mathcal{O}(rn^3) = \mathcal{O}(n^4)$, where n is the number of states and r is the rank, this results in $\mathcal{O}(n^{4+\omega})$ time complexity. By efficiently constructing only one maximal column (as described in the previous section) and reusing it for the whole set of states (as described in this section), we decrease the time complexity to $\mathcal{O}(n^{2+\omega})$.

▶ **Proposition 30.** *An SLP of length $\mathcal{O}(n^2)$ encoding a word w of minimum rank can be computed in $\mathcal{O}(n^{2+\omega} + mn^4)$ time and $\mathcal{O}(n^3)$ space.*

Proof sketch. Compute the matrix of the word w encoded by the SLP from Lemma 29. Iteratively, for each $q \in Q$, concatenate w with a word of length at most n mapping p to a state corresponding to a nonzero element of the row in the current iteration. Denote by w_n the resulting word, which has the property that all nonzero columns of $M(w_n)$ are maximal. Symmetrically compute w'_n for rows. Then by Lemma 5 the word $w_n w'_n w_n w'_n$ matrix has minimum rank. ◀

Given an SLP of length $\mathcal{O}(n^2)$ encoding a word w, we can compute the matrix of w by computing the matrices of words occurring in the derivation of w from bottom to top in time $\mathcal{O}(n^{2+\omega})$. Thus we prove Theorem 23. We also get the following result, which can be seen as a proof of a very weak version of the Černý conjecture generalised from rank one words in complete DFAs to minimum rank words in complete UFAs.

▶ **Theorem 31.** *For every n-state complete UFA, there exists an SLP of length $\mathcal{O}(n^2)$ encoding a word of minimum rank.*

We remark that the length of the word encoded by the constructed SLP asymptotically matches the best known upper bound for words of minimum rank: $\mathcal{O}(n^4)$ for complete UFAs [14] and $\mathcal{O}(n^3)$ for complete DFAs [37]. In particular, one can efficiently compute words of minimum rank within these bounds.

6.5 Complete DFAs

For complete DFAs, we follow the same algorithms as in the proof of Theorem 23, but exploit the fact that elementary matrix operations can be performed more efficiently. Namely, if $\mathcal{A}$ is a complete DFA, then each word defines a transformation on Q. By storing matrices of words as transformations, we get that matrix multiplication can be performed in $\mathcal{O}(n)$ time, and each matrix requires $\mathcal{O}(n)$ space. Moreover, we have $|E^{(2)}| = mn^2$. By taking these improvements into account, we get the proof of Theorem 24.

7 Conclusions and open problems

We list a few open questions that follow directly from our work.

- In [20], it was asked if a word of rank one for a complete DFA can be found in NC. Similarly, can a matrix of minimum rank for a complete DFA be computed in NC?
- Given an unambiguous morphism $M\colon \Sigma \to \{0,1\}^{Q\times Q}$ and a vector $\alpha \in \mathbb{Q}^Q_{\geq 0}$, can a basis of $\langle \alpha^T M(w) \mid w \in \Sigma^* \rangle$ be computed faster than in $\mathcal{O}(|Q|^3)$ time? This would improve algorithms for several fundamental problems for weighted automata [32]. Similarly, for complete DFAs, computing a basis of U from Section 4.4 in subcubic time (see the proof of Lemma 26) would allow to compute the minimum rank of a complete DFA faster than in cubic time.
- The bottleneck in the time complexity in Theorem 22 is the very first step, computing $\mathsf{Mer}(q)$ via digraph search in the square digraph of $\mathcal{A}$. The number of edges of this digraph can be quadratic in the number of its vertices [36, Appendix A], hence of order $|Q|^4$. Can $\mathsf{Mer}(q)$ be computed faster than in time $\mathcal{O}(|Q|^4)$? Very little seems to be known about general properties of square automata of DFAs or UFAs.
- One of the bottlenecks of the combinatorial algorithm is performing matrix multiplication. Can it be done faster for matrices from a zero-one matrix monoid? If not, are there any subclasses of UFAs (other than DFAs) where it can be done more efficiently?

References

1 Jorge Almeida and Benjamin Steinberg. Matrix mortality and the Černý-Pin conjecture. In Volker Diekert and Dirk Nowotka, editors, *Developments in Language Theory*, pages 67–80, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. doi:10.1007/978-3-642-02737-6_5.

2 Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach.* Cambridge University Press, 2009.

3 Christel Baier, Stefan Kiefer, Joachim Klein, David Müller, and James Worrell. Markov chains and unambiguous automata. *Journal of Computer and System Sciences*, 136:113–134, 2023. doi:10.1016/J.JCSS.2023.03.005.

4 M.-P. Béal and D. Perrin. Synchronised automata. In Valérie Berthé and Michel Rigo, editors, *Combinatorics, Words and Symbolic Dynamics*, Encyclopedia of Mathematics and its Applications, pages 213–240. Cambridge University Press, 2016. doi:10.1017/CBO9781139924733.008.

5 Marie-Pierre Béal, Eugen Czeizler, Jarkko Kari, and Dominique Perrin. Unambiguous automata. *Math. Comput. Sci.*, 1(4):625–638, 2008. doi:10.1007/S11786-007-0027-1.

6 Paul C. Bell, Igor Potapov, and Pavel Semukhin. On the mortality problem: From multiplicative matrix equations to linear recurrence sequences and beyond. *Information and Computation*, 281:104736, 2021. doi:10.1016/J.IC.2021.104736.

7 Stuart J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18(3):147–150, 1984. doi:10.1016/0020-0190(84)90018-8.

8 Mikhail V. Berlinkov. On two algorithmic problems about synchronizing automata (short paper). In Arseny M. Shur and Mikhail V. Volkov, editors, *Developments in Language Theory – 18th International Conference, DLT 2014, Ekaterinburg, Russia, August 26-29, 2014. Proceedings*, volume 8633 of *Lecture Notes in Computer Science*, pages 61–67. Springer, 2014. doi:10.1007/978-3-319-09698-8_6.

9 Jean Berstel, Dominique Perrin, and Christophe Reutenauer. *Codes and automata*, volume 129. Cambridge University Press, 2010.

10 Vincent D. Blondel, Raphaël M. Jungers, and Alex Olshevsky. On primitivity of sets of matrices. *Automatica*, 61:80–88, 2015. doi:10.1016/J.AUTOMATICA.2015.07.026.

11 Allan Borodin, Joachim von zur Gathen, and John E. Hopcroft. Fast parallel matrix and GCD computations. *Information and Control*, 52(3):241–256, 1982. doi:10.1016/S0019-9958(82)90766-5.

12 Greg Budzban and Philip Feinsilver. The generalized road coloring problem and periodic digraphs. *Applicable Algebra in Engineering, Communication and Computing*, 22:21–35, 2011. doi:10.1007/S00200-010-0135-Z.

13 Georgina Bumpus, Christoph Haase, Stefan Kiefer, Paul-Ioan Stoienescu, and Jonathan Tanner. On the size of finite rational matrix semigroups. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 115:1–115:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.ICALP.2020.115.

14 Arturo Carpi. On synchronizing unambiguous automata. *Theoretical Computer Science*, 60:285–296, 1988. doi:10.1016/0304-3975(88)90114-4.

15 Arturo Carpi and Flavio D'Alessandro. Strongly transitive automata and the Černý conjecture. *Acta Informatica*, 46(8):591–607, 2009. doi:10.1007/S00236-009-0106-7.

16 Julien Cassaigne, Vesa Halava, Tero Harju, and François Nicolas. Tighter undecidability bounds for matrix mortality, zero-in-the-corner problems, and more. *CoRR*, abs/1404.0644, 2014. arXiv:1404.0644.

17 Yves Césari. Sur l'application du théorème de Suschkewitsch à l'étude des codes rationnels complets. In Jacques Loeckx, editor, *Automata, Languages and Programming, 2nd Colloquium, University of Saarbrücken, Germany, July 29 - August 2, 1974, Proceedings*, volume 14 of *Lecture Notes in Computer Science*, pages 342–350. Springer, 1974. doi:10.1007/3-540-06841-4_73.

18 Stephen A. Cook. A taxonomy of problems with fast parallel algorithms. *Inf. Control.*, 64(1-3):2–21, 1985. doi:10.1016/S0019-9958(85)80041-3.

19 Laszlo Csanky. Fast parallel matrix inversion algorithms. *SIAM Journal on Computing*, 5(4):618–623, 1976. doi:10.1137/0205040.

20 David Eppstein. Reset sequences for monotonic automata. *SIAM Journal on Computing*, 19(3):500–510, 1990. doi:10.1137/0219033.

21 Joel Friedman. On the road coloring problem. *Proceedings of the American Mathematical Society*, 110(4):1133–1135, 1990.

22 Balázs Gerencsér, Vladimir V. Gusev, and Raphaël M. Jungers. Primitive sets of nonnegative matrices and synchronizing automata. *SIAM Journal on Matrix Analysis and Applications*, 39(1):83–98, 2018. doi:10.1137/16M1094099.

23 Oded Goldreich. *Computational Complexity: A Conceptual Perspective.* Cambridge University Press, 2008. doi:10.1017/CBO9780511804106.

24 Pavel Goralčík and Václav Koubek. Rank problems for composite transformations. *International Journal of Algebra and Computation*, 05(03):309–316, 1995. doi:10.1142/S0218196795000185.

25 Vladimir V. Gusev and Elena V. Pribavkina. On synchronizing colorings and the eigenvectors of digraphs. In Piotr Faliszewski, Anca Muscholl, and Rolf Niedermeier, editors, *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, August 22-26, 2016 - Kraków, Poland*, volume 58 of *LIPIcs*, pages 48:1–48:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICS.MFCS.2016.48.

26 Markus Holzer and Sebastian Jakobi. On the computational complexity of problems related to distinguishability sets. *Information and Computation*, 259(2):225–236, 2018. doi:10.1016/J.IC.2017.09.003.

27 Oscar H. Ibarra, Shlomo Moran, and Louis E. Rosier. A note on the parallel complexity of computing the rank of order n matrices. *Information Processing Letters*, 11(4/5):162, 1980. doi:10.1016/0020-0190(80)90042-3.

28 Gérard Jacob. Un algorithme calculant le cardinal, fini ou infini, des demi-groupes de matrices. *Theoretical Computer Science*, 5(2):183–204, 1977. doi:10.1016/0304-3975(77)90006-8.

29 Jarkko Kari. A counter example to a conjecture concerning synchronizing words in finite automata. *Bulletin of the EATCS*, 73:146, 2001.

30 Jarkko Kari, Andrew Ryzhikov, and Anton Varonka. Words of minimum rank in deterministic finite automata. In Piotrek Hofman and Michal Skrzypczak, editors, *Developments in Language Theory – 23rd International Conference, DLT 2019, Warsaw, Poland, August 5-9, 2019, Proceedings*, volume 11647 of *Lecture Notes in Computer Science*, pages 74–87. Springer, 2019. doi:10.1007/978-3-030-24886-4_5.

31 Nasim Karimi. Reaching the minimum ideal in a finite semigroup. *Semigroup Forum*, 94(2):390–425, 2017.

32 Stefan Kiefer. Notes on equivalence and minimization of weighted automata. https://arxiv.org/abs/2009.01217, 2020. arXiv:arXiv:2009.01217.

33 Stefan Kiefer, Ines Marusic, and James Worrell. Minimisation of multiplicity tree automata. *Logical Methods in Computer Science*, 13(1), 2017. doi:10.23638/LMCS-13(1:16)2017.

34 Stefan Kiefer and Corto N. Mascle. On nonnegative integer matrices and short killing words. *SIAM Journal on Discrete Mathematics*, 35(2):1252–1267, 2021. doi:10.1137/19M1250893.

35 Stefan Kiefer and Cas Widdershoven. Efficient analysis of unambiguous automata using matrix semigroup techniques. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPIcs*, pages 82:1–82:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.MFCS.2019.82.

36 Stefan Kiefer and Cas Widdershoven. Efficient analysis of unambiguous automata using matrix semigroup techniques. *CoRR*, abs/1906.10093, 2019. arXiv:1906.10093.

37 A.A. Klyachko, I.K. Rystsov, and M.A. Spivak. An extremal combinatorial problem associated with the bound on the length of a synchronizing word in an automaton. *Cybernetics*, 23(2):165–171, 1987.

38 Douglas A. Lind and Brian Marcus. *An introduction to symbolic dynamics and coding.* Cambridge university press, 2021.

39 Arnaldo Mandel and Imre Simon. On finite semigroups of matrices. *Theoretical Computer Science*, 5(2):101–111, 1977. doi:10.1016/0304-3975(77)90001-9.

40 Mike Paterson. Unsolvability in 3×3 matrices. *Studies in Applied Mathematics*, 49:105–107, 1970.

41 Vladimir Yu. Protasov. Analytic methods for reachability problems. *Journal of Computer and System Sciences*, 120:1–13, 2021. doi:10.1016/J.JCSS.2021.02.007.

42 Vladimir Yu. Protasov and Andrey S. Voynov. Matrix semigroups with constant spectral radius. *Linear Algebra and its Applications*, 513:376–408, 2017.

43 Igor Rystsov. Rank of a finite automaton. *Cybernetics and Systems Analysis*, 28(3):323–328, 1992.

44 Igor K. Rystsov. Reset words for commutative and solvable automata. *Theoretical Computer Science*, 172(1-2):273–279, 1997. doi:10.1016/S0304-3975(96)00136-3.

45 Wojciech Rytter. The space complexity of the unique decipherability problem. *Information Processing Letters*, 23(1):1–3, 1986. doi:10.1016/0020-0190(86)90121-3.

46 Andrew Ryzhikov. Mortality and synchronization of unambiguous finite automata. In Robert Mercas and Daniel Reidenbach, editors, *Combinatorics on Words – 12th International Conference, WORDS 2019, Loughborough, UK, September 9-13, 2019, Proceedings*, volume 11682 of *Lecture Notes in Computer Science*, pages 299–311. Springer, 2019. doi:10.1007/978-3-030-28796-2_24.

47 Andrew Ryzhikov. On shortest products for nonnegative matrix mortality. In Laura Kovács and Ana Sokolova, editors, *Reachability Problems – 18th International Conference, RP 2024, Vienna, Austria, September 25-27, 2024, Proceedings*, volume 15050 of *Lecture Notes in Computer Science*, pages 104–119. Springer, 2024. doi:10.1007/978-3-031-72621-7_8.

48 Sujin Shin and Jisang Yoo. A note on the rank of semigroups. *Semigroup Forum*, 81(2):335–343, 2010.

49 Mikhail V. Volkov. Synchronizing automata and the Černý conjecture. In Carlos Martín-Vide, Friedrich Otto, and Henning Fernau, editors, *Language and Automata Theory and Applications, Second International Conference, LATA 2008, Tarragona, Spain, March 13-19, 2008. Revised Papers*, volume 5196 of *Lecture Notes in Computer Science*, pages 11–27. Springer, 2008.

50 Mikhail V. Volkov. Synchronization of finite automata. *Russian Mathematical Surveys*, 77(5):819–891, 2022. doi:10.4213/rm10005e.

51 Yaokun Wu and Yinfeng Zhu. Primitivity and Hurwitz primitivity of nonnegative matrix tuples: A unified approach. *SIAM Journal on Matrix Analysis and Applications*, 44(1):196–211, 2023. doi:10.1137/22M1471535.