Monoids of Upper Triangular Matrices over the Boolean Semiring

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

Petra Wolf
LaBRI, CNRS, Université de Bordeaux, Bordeaux INP, France

Abstract

Given a finite set $\mathcal{A}$ of square matrices and a square matrix B, all of the same dimension, the membership problem asks if B belongs to the monoid $\mathcal{M}(\mathcal{A})$ generated by $\mathcal{A}$. The rank one problem asks if there is a matrix of rank one in $\mathcal{M}(\mathcal{A})$. We study the membership and the rank one problems in the case where all matrices are upper triangular matrices over the Boolean semiring. We characterize the computational complexity of these problems, and identify their PSPACE-complete and NP-complete special cases.

We then consider, for a set $\mathcal{A}$ of matrices from the same class, the problem of finding in $\mathcal{M}(\mathcal{A})$ a matrix of minimum rank with no zero rows. We show that the minimum rank of such matrix can be computed in linear time.We also characterize the space complexity of this problem depending on the size of $\mathcal{A}$, and apply all these results to the ergodicity problem asking if $\mathcal{M}(\mathcal{A})$ contains a matrix with a column consisting of all ones. Finally, we show that our results give better upper bounds for the case where each row of every matrix in $\mathcal{A}$ contains at most one non-zero entry than for the general case.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases matrix monoids, membership, rank, ergodicity, partially ordered automata

Digital Object Identifier 10.4230/LIPIcs.MFCS.2024.81

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). *Petra Wolf*: Supported by the French ANR, project ANR-22-CE48-0001 (TEMPOGRAL).

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

1 Introduction

Membership in matrix monoids. Given a finite set $\mathcal{A}$ of $n \times n$ matrices and an $n \times n$ matrix B, *the membership problem* asks if B belongs to the monoid $\mathcal{M}(\mathcal{A})$ generated by $\mathcal{A}$, which is the set of all products of matrices from $\mathcal{A}$. For matrices over rational numbers, membership is solvable in polynomial time if $\mathcal{A}$ consists of one matrix [22]. This result was later extended to the case where $\mathcal{A}$ is a set of commuting matrices [1], and then to a special class of non-commutative matrices [28]. Membership is also decidable for 2×2 non-singular integer matrices [34]. On the other hand, membership is already undecidable for 3×3 integer matrices with determinant equal to 0 [32].

For sets of matrices over the Boolean semiring (the set $\{0, 1\}$ with addition and multiplication defined the same way as for integer numbers, except that $1 + 1 = 1$), many natural problems, including membership, are trivially in PSPACE, and are thus more tractable. However, membership in this case is actually PSPACE-complete [25]. In this paper, we further restrict the setting and study upper triangular matrices over the Boolean semiring. Everywhere below in this paper, we assume all matrices to be square and over the Boolean

semiring, unless specified otherwise. Upper triangular matrices may seem to be a simple and restrictive class, but sets of such matrices and the corresponding automata have been studied extensively, and many problems for them are known to be PSPACE-complete, see e.g. [33, 26, 31, 37, 30]. Nevertheless, we are not aware of any results on the four problems we consider in this paper (membership, rank one, minimum rank of a total word and ergodicity) specifically for the case of upper triangular matrices, except for the results of [43] on the ergodicity problem, which we explain below.

Matrices and sets of matrices over the Boolean semiring play an important role in automata theory [4], discrete-time Markov chains [38], graph theory [2], variable-length codes [12], and symbolic dynamics [27]. They are often used in the theory of nonnegative matrices, since in many problems one is not interested in the actual values of the entries, but only in which values are zero and which are strictly positive [3].

Boolean square matrices can also be viewed as binary relations on a finite set Q. Matrices with at most one non-zero entry in each row thus correspond to partial transformations, and if they have exactly one non-zero entry in every row, to complete transformations. We call such matrices transformation and complete transformation matrices respectively. For monoids of complete transformation matrices, membership is PSPACE-complete [25], and its complexity was studied for several subsclasses [7, 6, 9, 8]. Connections of membership with the automata intersection problem are explored in [13].

The rank one problem. Instead of asking if the monoid $\mathcal{M}(\mathcal{A})$ contains a given matrix, one can ask if it contains a matrix from a certain class. Given a set $\mathcal{A}$ of matrices, *the rank one problem* asks if there is a matrix of rank one in $\mathcal{M}(\mathcal{A})$.

The rank one problem is known to be solvable in polynomial time for monoids of complete transformation matrices (see e.g. [41]), strongly connected monoids of transformation matrices [11] and, more generally, for strongly connected monoids of unambiguous relations and their subclasses [36, 14, 24, 4] (a monoid of $n \times n$ matrices is called *strongly connected* if for each pair (i, j), $1 \leq i, j \leq n$, it contains a matrix whose entry (i, j) is equal to one). The latter case plays an important role in the theory of variable-length codes because of its connections with synchronizing codes [12]. On the other hand, the rank one problem is PSPACE-complete already for monoids of transformation matrices which are not strongly connected [10]. The complexity of deciding if a monoid of complete transformation matrices contains a matrix of given rank is studied in [18].

The rank one problem is related to the Černý conjecture and its generalizations. Indeed, the case of monoids of complete transformations corresponds exactly to the case of complete DFAs, and hence in this case the rank one problem is nothing else than the problem of checking if a complete DFA is synchronizing. We refer to the surveys [41, 42, 23, 5] for the vast literature on synchronizing DFAs.

Careful synchronization and ergodicity. A similar problem is, given a set $\mathcal{A}$ of matrices, to find in $\mathcal{M}(\mathcal{A})$ a minimum rank matrix without zero rows. We call matrices with no zero rows *total*, since they correspond to total relations. For sets of transformation matrices, monoids of matrices containing a total matrix of rank one correspond to carefully synchronizing DFAs [29]. Deciding if $\mathcal{M}(\mathcal{A})$ contains a total matrix of rank one is PSPACE-complete even if $\mathcal{A}$ consists of two matrices [29], and the length of a shortest product of matrices from $\mathcal{A}$ resulting in a total matrix of rank one can be exponential [29]. If $\mathcal{A}$ is a set of upper triangular transformation matrices, deciding if $\mathcal{M}(\mathcal{A})$ contains a total matrix of rank one and finding a product resulting in such a matrix is solvable in polynomial time [43]. We discuss the contributions of [43] and their connection to the results of this paper in more details in the beginning of Section 4.

A closely related notion is that of ergodic matrices (also known as column-primitive [16] or D3-directing [20, 21]). A matrix is called *ergodic* [44] if it has a column consisting of all ones. Note that a transformation matrix is ergodic if and only if it is a total matrix of rank one, that is, this case also coincides with carefully synchronizing DFAs. *The ergodicity problem* asks if a given matrix monoid contains an ergodic matrix. The ergodicity problem is PSPACE-complete [29], but is solvable in polynomial time for monoids of matrices with no zero rows [35, 44]. An application of ergodicity to deciding if a discrete-time multi-agent system can be driven to consensus is presented in [16]. We emphasize that the existing polynomial-time solvability results on ergodicity [44, 16, 15] (except for [43]) consider monoids of matrices with no zero rows, while our results for this problem do not have this requirement.

2 Main definitions and our contributions

Boolean rank. *The Boolean semiring* is the set $\{0,1\}$ with addition and multiplication defined in the usual way except that $1+1=1$. All matrices in this paper are over the Boolean semiring. The *Boolean rank* [17] (which we call just *rank* for brevity in this paper) of an $n\times n$ matrix A is the smallest number r such that $A=CR$, where C is an $n\times r$ matrix, and R is an $r\times n$ matrix, with usual matrix multiplication over the Boolean semiring.

For example, the following equality shows that the rank of A is at most two (moreover, it cannot be one since there are two different non-zero rows in A):

$$A=\begin{pmatrix}1&1&1\\1&1&0\\0&1&1\end{pmatrix}=\begin{pmatrix}1&1\\1&0\\0&1\end{pmatrix}\begin{pmatrix}1&1&0\\0&1&1\end{pmatrix}=CR.$$

Matrices and automata. Let $\mathcal{A}=\{A_1,\dots,A_m\}$ be a set of $n\times n$ matrices over the Boolean semiring. The *monoid generated by* $\mathcal{A}$ is the set of all possible products (with respect to the usual matrix multiplication) of matrices from $\mathcal{A}$, including the empty product corresponding to the identity matrix. By assigning to each matrix $A_i\in\mathcal{A}$ a letter a_i from a finite alphabet Σ of size m, the set $\mathcal{A}$ can be equivalently seen as a nondeterministic finite automaton (NFA) with state set $Q=\{1,2,\dots,n\}$, alphabet Σ and transition relation $\Delta: Q\times\Sigma\to 2^Q$ defined for each letter a_i by the action of the corresponding matrix A_i. That is, we have $j\in\Delta(i,a_k)$ if and only if the entry (i,j) in A_k is one. If the transition relation is clear from the context, we denote the set $\Delta(q,a)$ as $q\cdot a$. Equivalently, given an NFA (Q,Σ,Δ), we can consider for each $a\in\Sigma$ the matrix such that its entry (p,q), $p,q\in Q$, is non-zero if and only if $q\in\Delta(p,a)$. This allows to consider an NFA as a set of matrices corresponding to its letters. The transition relation can be homomorphically extended to the set Σ^* of words over Σ. Words over Σ thus naturally correspond to products of matrices. In particular, by the matrix of a word we mean the result of this product, and by the rank of a word we mean the rank of its matrix. In this paper we make use of this correspondence between NFAs and matrix monoids and switch between the languages of matrices and automata whenever convenient. We remark that the described construction is nothing more than a homomorphism from the free monoid Σ^* to the monoid of matrices over the Boolean semiring.

In NFA terms, the matrix M of a word w has rank at most r if and only if there exist r subsets $Q_1,\dots,Q_r$ of Q such that the image of every state of Q under w is a union of some of these subsets. If r is the smallest such number, the rank of M is equal to r. In particular, the rank of a word is one if and only if there is a subset $Q_1\subseteq Q$ such that the image of every state under w is either Q_1 or the empty set, and not all images are empty.

For a set $S \subseteq Q$ of states and a word $w \in \Sigma^*$, we denote $S \cdot w = \{p \in Q \mid p \in q \cdot w \text{ for some } q \in S\}$. We say that the action of a word w is *defined* for a state q if $q \cdot w$ is non-empty. A word is called *total* if its action is defined for every state. Equivalently, it means that the matrix of a word does not have zero rows.

An NFA is a *deterministic finite automaton* (DFA) if for every $p \in Q$ and $a \in \Sigma$ we have that $|\Delta(p, a)| \leq 1$. In this case we use δ instead of Δ to emphasize that the transition relation is a (partial) transition function. If moreover $\delta(p, a)$ is defined for every $p \in Q$ and $a \in \Sigma$, the DFA is called *complete*. For DFAs, the rank of a word w is simply the size $|Q \cdot w|$.

Ergodic words. A word w is called *ergodic* for an NFA $\mathcal{A} = (Q, \Sigma, \Delta)$ if there is a state $q \in Q$ such that for every state $p \in Q$ we have $q \in p \cdot w$. Equivalently, a word is ergodic if its matrix contains a column of all ones. Clearly, every ergodic word is total. We call an NFA *ergodic* if it admits an ergodic word.

Partially ordered automata. An NFA $\mathcal{A} = (Q, \Sigma, \Delta)$ is called *partially ordered* (poNFA) if there exists an order $\prec$ on its set of states Q which is preserved by all its transitions. That is, if $p \in \Delta(q, a)$ for $p, q \in Q$ and $a \in \Sigma$, then $q \prec p$ or $q = p$. Such order can be found in time $\mathcal{O}(mn)$ by topological sorting, and in deterministic logarithmic space [40], hence we always assume that a poNFA is provided together with the order $\prec$. Equivalently, an NFA is a poNFA if the underlying digraph of $\mathcal{A}$ does not have any cycles other than self-loops, or, alternatively, if the matrices of all letters of $\mathcal{A}$ are upper triangular. A poNFA $\mathcal{A}$ is called *self-loop deterministic* if for every letter $a \in \Sigma$ such that $q \in q \cdot a$ for a state $q \in Q$ we have $q \cdot a = \{q\}$. Following [26], we denote self-loop deterministic partially ordered NFAs as rpoNFAs. We denote partially ordered DFAs as poDFAs.

Decision problems. An *NFA acceptor* $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ is an NFA (Q, Σ, Δ) with distinguished sets $I \subseteq Q$ and $F \subseteq Q$ of, respectively, initial and final states. The *language* $L(\mathcal{A})$ of an NFA acceptor is the set of words $w \in \Sigma^*$ such that there exist states $i \in I$, $f \in F$ with $f \in i \cdot w$. Given an NFA acceptor, *the universality problem* asks if it accepts all words over its alphabet. Universality of poNFAs is PSPACE-complete even over a binary alphabet [26], and remains PSPACE-complete for rpoNFAs over polynomial size alphabet [26, 30].

Given an NFA $\mathcal{A}$ and a matrix B, *the membership problem* asks if there exists a word such that its matrix in $\mathcal{A}$ is equal to B. Given an NFA $\mathcal{A}$, the *rank one problem* asks if there is a word whose matrix has rank one in $\mathcal{A}$.

We assume that the reader is familiar with basic notions of automata theory and computational complexity, see e.g. [39].

Our contributions. In the first half of this paper, Section 3, we analyze the computational complexity of the membership and rank one problems for poNFAs (equivalently, for monoids of upper triangular matrices). We show that they are PSPACE-complete for rpoNFAs and ternary poNFAs (Proposition 6). The main results of Section 3 are that for rpoNFAs over a fixed alphabet the membership problem is in NP (Theorem 1), and remains NP-complete even for complete poDFAs over a binary alphabet (Theorem 8). We show containment in NP by proving that for each word there is a short word with the same matrix (Proposition 2). We complement this result with a lower bound on the length of words with a given matrix (Lemma 7).

In the second half of the paper, Section 4, we study the problem of finding the minimum rank of a total word in a poNFA. We show that, depending on the size of the alphabet, this problem lies between L and P-complete, and moreover can be solved in linear time. These

results are summarized in Theorem 11. All these results transfer to the problem of ergodicity of a set of upper triangular matrices (Theorem 15), which asks if a matrix monoid contains a matrix with a column consisting of all ones. We then further extend these results to get improved bounds for ergodicity of poDFAs (Theorem 16).

3 Membership and minimum rank

3.1 Upper bounds for rpoNFAs

The goal of this subsection is to prove the following theorem.

▶ **Theorem 1.** *Membership for rpoNFAs over a fixed alphabet is in* NP.

We start with the following key result which, intuitively, shows that if $\mathcal{A}$ is an rpoNFA, a product of matrices from $\mathcal{A}$ cannot contain too many partial products.

▶ **Proposition 2.** *Let $\mathcal{A} = (Q, \Sigma, \Delta)$ be an rpoNFA with n states and m letters. For every word $w \in \Sigma^*$ there exist at most $n(n+1)^{m-1}$ prefixes of w with pairwise different matrices.*

Proof. We prove the statement by induction on m. Clearly, if $m = 1$, the matrix of every word w of length more than n is equal to the matrix of some word v of length at most n, and thus the matrix of every prefix of w is equal to the matrix of some prefix of v, and there are only at most n prefixes.

Assume now that $m \geq 2$ and the statement is true for rpoNFAs over all alphabets $\Sigma' \subset \Sigma$, $|\Sigma'| \leq m-1$. We can assume that every letter from Σ appears in w, otherwise the statement is proved. Let $Q = Q_1 \cup Q_1'$, where Q_1 is the set of states p such that there exists a letter $a \in \Sigma$ with $p \notin p \cdot a$, and $Q_1' = Q \setminus Q_1$. Let q be a smallest (with respect to $\prec$) state in Q_1, and let $a \in \Sigma$ be a letter such that $q \notin q \cdot a$.

Let w' be the prefix of w before the first appearance of a, that is, w' is the prefix of w such that w' does not contain a, and $w'a$ is also a prefix of w. Clearly, $w' \in (\Sigma \setminus \{a\})^*$. By the induction assumption, there are at most $n(n+1)^{m-2}$ prefixes of w' with pairwise different matrices. After the first application of a, for each state $p \in Q_1 \cdot w'a$ we have that $q \prec p$.

Now we perform the following iterative process. If the suffix of w after the first appearance of a does not contain all letters from Σ, we stop. Otherwise, we consider the smallest state q' in $Q_1 \cdot w'a$ and a letter a' such that $q' \notin q' \cdot a'$, and repeat the argument above with $Q_1 \cdot w'a$ taken as Q_1, q' taken as q, a' taken as a, and w' taken as w. This argument will be repeated at most n times in total, since after n times Q_1 must be empty, and thus the remaining suffix does not change the matrix of the word. Hence we get that there are at most $n \cdot (n(n+1)^{m-2} + 1) \leq n(n+1)^{m-1}$ prefixes of w with pairwise different matrices. ◀

▶ **Lemma 3.** *Let $\mathcal{A}$ be an rpoNFA and w be a word. Let P be the set of matrices of all prefixes of w, and $|P| = k$. Then there exist words $w_1, \ldots, w_k$ such that $w = w_1 w_2 \ldots w_k$, every matrix in P is equal to the matrix of $w_1 \ldots w_i$ for some i, and for each i, $0 \leq i < k$, for every non-empty prefix v of w_{i+1} the matrices of $w_1 w_2 \ldots w_i v$ and $w_1 w_2 \ldots w_i w_{i+1}$ are equal.*

Proof. Represent w as a concatenation $w_1 w_2 \ldots w_k$ of words w_i such that for every i, $0 \leq i < k$, the matrices of $w_1 w_2 \ldots w_i$ and $w_1 w_2 \ldots w_i w_{i+1}$ are different, but the matrix of $w_1 w_2 \ldots w_i v$ is equal to the matrix of $w_1 w_2 \ldots w_i w_{i+1}$ for every non-empty prefix v of w_{i+1}. We only need to show that for $i \neq j$ the matrices of $w_1 w_2 \ldots w_i$ and $w_1 w_2 \ldots w_j$ cannot be equal. This follows from the fact that in rpoNFAs, if for some non-empty words u_1, u_2 the matrices of u_1 and $u_1 u_2$ are different, there is no word u_3 such that the matrices of u_1 and $u_1 u_2 u_3$ are equal. ◀

The following statement implies that membership is in NP for rpoNFAs over a fixed alphabet, thus proving Theorem 1. It also implies that the rank one problem for rpoNFAs over a fixed alphabet is in NP.

▶ **Proposition 4.** *Let $\mathcal{A}$ be an rpoNFA with n states and m letters. Then for every word there exists a word of length at most $n(n+1)^{m-1}$ with the same matrix.*

Proof. Let $w_1, \ldots, w_k$ be the words from the statement of Lemma 3. By Proposition 2, $k \leq n(n+1)^{m-1}$. For each i, $1 \leq i \leq k$, let a_i be the first letter of w_i. Then the matrix of $a_1 a_2 \ldots a_k$ is equal to the matrix of w. ◀

As another consequence of our results, we get an elementary combinatorial proof of a theorem from [26] which originally required some non-trivial formal languages machinery.

▶ **Corollary 5** (Theorem 23 in [26])**.** *Let $\mathcal{A}$ be an NFA acceptor, and $\mathcal{B}$ be an rpoNFA acceptor, both over the same fixed alphabet. Checking if $L(\mathcal{A}) \subseteq L(\mathcal{B})$ is in* coNP*.*

Proof. If $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$, there exists a word w accepted by $\mathcal{A}$ and not accepted by $\mathcal{B}$. Let $w_1, \ldots, w_k$ be the words from the statement of Lemma 3. By Proposition 2, $k \leq n(n+1)^{m-1}$. For each i let a_i be the first letter of w_i, and let $w_i = a_i w_i'$. Furthermore, let Σ_i be the set of letters occurring in w_i'. Consider the regular expression $e = a_1 \Sigma_1^* a_2 \Sigma_2^* a_3 \ldots \Sigma_{k-1}^* a_k \Sigma_k^*$. By construction, w is accepted by e, and for every word accepted by e its matrix in $\mathcal{B}$ is equal to the matrix of w in $\mathcal{B}$, and thus every such word is not accepted by $\mathcal{B}$. Moreover, e can be straightforwardly transformed into a poNFA acceptor $\mathcal{E}$ with at most $n(n+1)^{m-1}$ states and m letters. It remains to note that the intersection of $\mathcal{A}$ and $\mathcal{E}$ can be decided in polynomial time, and hence $\mathcal{E}$ can be used as a certificate for coNP. ◀

3.2 Lower bounds for rpoNFAs

We now complement the positive results from the previous subsection with lower bounds. We first show that if the alphabet is not fixed, membership for rpoNFAs is as hard as for general NFAs. The proof of that is done by constructing a reduction from the universality problem for rpoNFA acceptors, which is PSPACE-complete [26]. The same proof works for ternary poNFAs.

▶ **Proposition 6.** *The membership and rank one problems are* PSPACE*-complete for rpoNFAs and ternary poNFAs.*

Proof. To prove both statements, we use the following reduction from the universality problem for poNFA acceptors. Let $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ be a poNFA acceptor. Construct a poNFA $\mathcal{A}' = (Q \cup \{q_0, p_0, p_1, q_+, q_-\}, \Sigma \cup \{r\}, \Delta')$. The transition relation Δ' includes all transitions in Δ, with the addition of $\Delta'(q_0, r) = I$, $\Delta'(q, r) = \{q_+\}$ if $q \in F$, and $\Delta'(q, r) = \{q_-\}$ if $q \in Q \setminus F$. We also define $\Delta'(p_0, r) = \{p_1\}$, $\Delta'(p_1, r) = \{q_-\}$. Finally, all letters in Σ induce self-loops for q_+, q_-, q_0, p_0, p_1. See Figure 1 for an illustration.

We claim that $\mathcal{A}$ does not accept a word in Σ^* if and only if there exists a word in $\mathcal{A}'$ which maps q_0 and p_0 to $\{q_-\}$, and all other states to the empty set. Indeed, if there exists a word w not accepted by $\mathcal{A}$, then the word rwr has the required action in $\mathcal{A}'$. Conversely, if w' is a word having the required action, then it must contain exactly two occurrences of r, and its factor between the first and the second occurrence of r maps I to a subset of $Q \setminus F$ and is thus not accepted by $\mathcal{A}$. Observe also that every word of rank one must have this action by construction. Furthermore, the described construction preserves the property that the NFA is an rpoNFA. It remains to use the fact that the universality problem is PSPACE-complete for rpoNFA acceptors and binary poNFA acceptors [26]. ◀

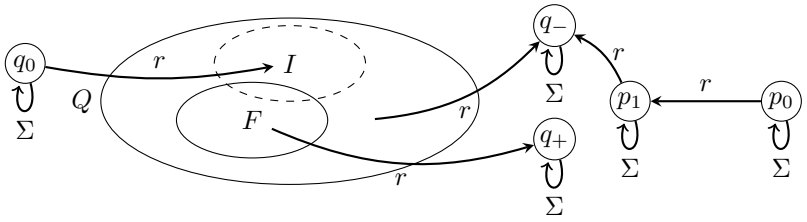

Figure 1 Illustration for the reduction in Proposition 6.

The following lemma provides a lower bound for the value discussed in Proposition 4. We remark that for constant m our lower and upper bounds are asymptotically the same, and if $m = n - 1$ the lower bound is exponential.

▶ **Lemma 7.** *Let n and m be positive natural numbers such that m divides $n-2$. Then there exists an rpoNFA with n states and m letters such that for some states t and f the length of a shortest word w with $f \cdot w = \{f\}$ and $q \cdot w = \{t\}$ for every state $q \neq f$ is $\frac{n-2}{m}(\frac{n-2}{m}+1)^{m-1}$.*

Proof. Let $n = ms+2$. We construct an rpoNFA $\mathcal{A} = (Q, \Sigma, \Delta)$ with the required properties as follows. Let $Q^{(j)} = \{q_1^{(j)}, \ldots, q_s^{(j)}\}$ and $S = \cup_{1 \leq j \leq m} Q^{(j)}$. Take $Q = S \cup \{t, f\}$, where t and f are fresh states, and $\Sigma = \{a_1, a_2, \ldots, a_m\}$. The transition relation Δ is defined for the states in the sets $Q^{(1)}, \ldots, Q^{(m)}$ in such a way that while reading a word w with $S \cdot w = \{t\}$, the states in $Q^{(j)}$ that contain an image of a state from S imitate an m-ary counter counting to zero, with some additional traversal in the process. Formally, for $1 \leq j, k \leq m$, $1 \leq i \leq s$ we set

$$\Delta(q_i^{(k)}, a_j) = \begin{cases} \{f\} & \text{if } j < k; \\ \{q_{i+1}^{(k)}\} & \text{if } j = k \text{ and } i < s; \\ Q^{(k+1)} \cup \ldots \cup Q^{(m)} & \text{if } j = k < m \text{ and } i = s; \\ \{t\} & \text{if } j = k = m \text{ and } i = s; \\ \{q_i^{(k)}\} & \text{if } j > k. \end{cases}$$

See Figure 2 for an illustration.

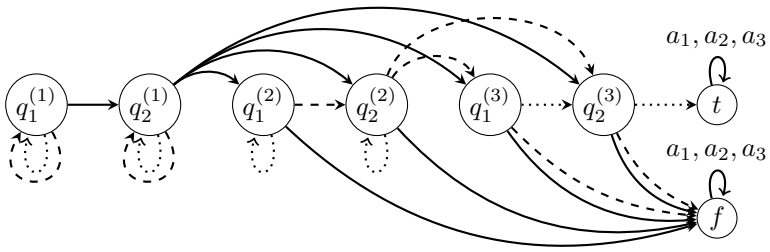

Figure 2 Illustration for the construction in Lemma 7. For unlabeled transitions, solid lines stand for a_1, dashed for a_2, dotted for a_3.

The only way to map a state to $\{t\}$ is to map it first to $\{q_s^{(m)}\}$ and then apply the letter a_m, otherwise the state is mapped to a set containing f. Observe that the shortest word such that $S \cdot w = \{t\}$ is unique. Indeed, at every moment of time there is a unique letter that does not act as the identity and does not map any state of S to f. To estimate the

length of this word, we argue inductively. Mapping $Q^{(m)}$ to $\{t\}$ requires a word of length s. Mapping the set $Q^{(j)}$, $j < m$, to $\{t\}$ requires subsequently mapping each of its states to $\cup_{k>j} Q^{(k)}$, and then mapping this union to $\{t\}$ before we can proceed to $Q^{(j-1)}$. Hence we get the bound of $s(s+1)^{m-1}$. ◀

By applying the idea of the reduction in the proof of Proposition 6 to the automaton constructed in the proof of Lemma 7, one can get a slightly weaker lower bound of $\frac{n-4}{m-1}(\frac{n-4}{m-1}+1)^{m-2}+1$ for the length of a shortest word of rank one in rpoNFAs with n states and m letters, where $m-1$ divides $n-4$. Recall that a word of rank one is allowed to kill some states.

3.3 Partially ordered DFAs

The following theorem characterizes the complexity of membership and rank one for poDFAs.

▶ **Theorem 8.** *The membership problem is* NP*-complete for complete poDFAs, even over a binary alphabet. The rank one problem is* NP*-complete for poDFAs, even over a binary alphabet.*

The proof of NP-hardness is an adaptation of a construction from [37], and is omitted due to space constraints. Containment in NP follows from the following result.

▶ **Proposition 9.** *Let $\mathcal{A}$ be a poDFA with n states. Then for every word there exists a word of length at most $\frac{1}{2}(n+1)n$ with the same matrix.*

Proof. Let $\mathcal{A} = (Q, \Sigma, \delta)$. Introduce a new state f, and define all undefined transitions of $\mathcal{A}$ to end there. Then for every state $q \in Q \setminus \{f\}$ we have $q \prec f$. Let w be a word such that there is no shorter word with the same matrix. For each prefix $w'a$ of w with $w' \in \Sigma^*, a \in \Sigma$, a must map one of the states in the image $Q \cdot w'$ to a larger state, otherwise this occurrence of a can be removed from w without changing its matrix. Hence the maximal length of w is at most $n + (n-1) + (n-2) + \ldots + 1 = \frac{1}{2}(n+1)n$. ◀

A lower bound on the value from Proposition 9 is provided in Lemma 17 below. Its adaptation provides the following lower bound on shortest words of rank one for poDFAs.

▶ **Lemma 10.** *For every natural numbers n and $m \geq 2$ such that m divides $n-1$, there exists a poDFA with $2n+1$ states and $m+1$ letters such that the length of a shortest word of rank one for it is $\frac{m-1}{2m}(n-1)^2 + (n-1)$.*

We remark that the rank one problem is trivial for poNFAs which do not have a word whose matrix is the zero matrix. Indeed, the minimum rank of a word in such poNFA is just the number of states such that each letter induces a self-loop for them. However, this obviously does not affect the complexity of membership, since adding a fresh state having self-loops by all letters guarantees that the matrix of each word is non-zero, while preserving the actions of all letters on the remaining states. For poDFAs, one can even obtain a complete poDFA by adding a sink state and sending all undefined transitions to it.

4 Total words of minimum rank and ergodic words

As shown above, the rank one problem is NP-complete for poDFAs and PSPACE-complete for rpoNFAs. In this section, we show that computing the minimum rank of a total word in a poNFA can be done in polynomial time. We then consider the case of poDFAs, show a tight quadratic bound on the length of shortest total words of minimum rank, and provide a fast algorithm to compute a total word of minimum rank within this length bound.

In [43], the second author proposed two algorithms for finding a total word of rank one in a poDFA if it exists. The first algorithm is greedy: take a letter mapping a currently active state to a larger one without taking undefined transitions. This algorithm does not seem to generalize to poNFAs, and is inherently sequential even for poDFAs. Moreover, its implementation proposed in [43] has time complexity $\mathcal{O}(mn^3)$. The second algorithm proposed in [43], of time complexity $\mathcal{O}(m^2n^2)$, uses a much less obvious approach. In this section we use a somewhat similar but more involved idea to deal with a more general case of poNFAs, and, in particular, show how to compute the minimum rank of a total word in linear time. For poDFAs, we show that a total word of minimum rank can be found in time $\mathcal{O}(mn^2)$, thus further improving on the results of [43]. We further refine the guarantee on the length of a found total word of minimum rank taking into account the size of the alphabet, and show that our upper bound it tight.

4.1 Total and ergodic words in poNFAs

▶ **Theorem 11.** *Let $\mathcal{A} = (Q, \Sigma, \Delta)$ be a poNFA with n states and m letters.*

(a) *The length of a shortest total word of minimum rank in $\mathcal{A}$ is at most $n(n+1)^m$.*

(b) *Finding the minimum rank of a total word in $\mathcal{A}$ can be done in $\mathcal{O}(m \log m + \log n)$ deterministic space.*
If $m = \mathcal{O}(\frac{\log n}{\log \log n})$, this problem is in L, and if $m = \Theta(n)$, it is P-complete.

(c) *The minimum rank of a total word in $\mathcal{A}$ can be computed in $\mathcal{O}(mn + |\Delta|)$ time, that is, in linear time in the size of the input.*

We first describe an algorithm that, given a poNFA $\mathcal{A} = (Q, \Sigma, \Delta)$ with n states and m letters, constructs an auxiliary sequence of sets $C_0 \supset C_1 \supset \ldots \supset C_k$ and a sequence of distinct letters $a_1, a_2, \ldots, a_k \in \Sigma$ for some $k \leq \min\{m, n-1\}$. Note that we use $\supset$ and $\subset$ to denote proper set inclusion. The value of k is determined by the algorithm and depends on the number of steps it makes. The intuition behind this algorithm (and the reason why it is called "stabilization algorithm") are provided below in the proof of its correctness.

Stabilization algorithm

We initialize the set of currently considered states $C_0 = Q$ and the currently explored alphabet $\Sigma_0 = \emptyset$. At step i, $i \geq 1$, find a letter $a_i \in \Sigma \setminus \Sigma_{i-1}$ which is defined for every state in C_{i-1} and does not induce a self-loop for at least one of them. That is, for every state $q \in C_{i-1}$, $q \cdot a_i$ must be non-empty, and for at least one state $q \in C_{i-1}$ we must have $q \notin q \cdot a_i$. If no such letter a_i exists, stop and output $|C_{i-1}|$. Define C_i to be the subset of states in C_{i-1} for which a_i induces a self-loop, that is, $C_i = \{q \in C_{i-1} \mid q \in q \cdot a_i\}$. Take $\Sigma_i = \Sigma_{i-1} \cup \{a_i\}$. Go to the step $i+1$.

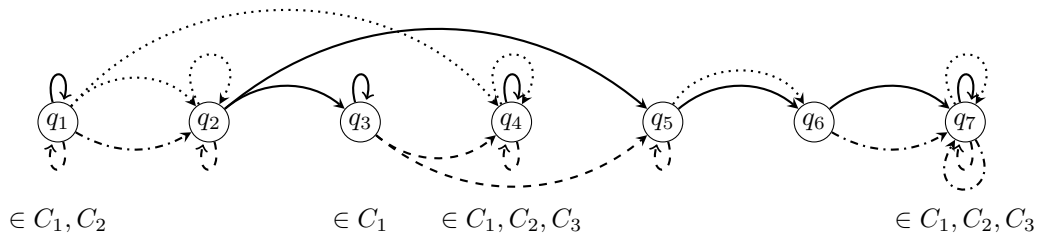

Figure 3 Running example for Section 4.1.

As a running example, we use the rpoNFA in Figure 3. The application of the algorithm gives $C_0 = Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$, $C_1 = \{q_1, q_3, q_4, q_7\}$, $C_2 = \{q_1, q_4, q_7\}$, $C_3 = \{q_4, q_7\}$. The letters chosen by the algorithm are as follows: a_1 is solid, a_2 is dashed, a_3 is dotted. The dashdotted letter is not used by the algorithm.

Correctness

We now analyze the output of the algorithm and show that it provides an upper bound on the minimum rank of a total word in $\mathcal{A}$. For rpoNFAs, this upper bound is tight. For general poNFAs, we then extend the algorithm with a second stage using similar ideas.

Assume that the algorithm makes $k + 1 \leq m + 1$ steps and then stops and returns the answer $|C_k|$. Consider the sequence of words defined as $w_0 = \epsilon$, $w_i = w_{i-1}(a_i w_{i-1})^n$ for $1 \leq i \leq k$. In our example, $w_1 = a_1^7$, $w_2 = a_1^7(a_2 a_1^7)^7$, $w_3 = a_1^7(a_2 a_1^7)^7(a_3 a_1^7(a_2 a_1^7)^7)^7$. Table 1 illustrates the images of states under w_i in our example.

The intuition is that with i increasing, the words w_i map every state to larger and larger states with respect to $\prec$, until a fixed point is reached. This can be informally seen as stabilization of the images of states from C_i under the word w_i.

Table 1 The images of states under w_i, $1 \leq i \leq 3$.

q	q_1	q_2	q_3	q_4	q_5	q_6	q_7
$q \cdot w_1$	q_1	q_3, q_7	q_3	q_4	q_7	q_7	q_7
$q \cdot w_2$	q_1	q_4, q_7	q_4, q_7	q_4	q_7	q_7	q_7
$q \cdot w_3$	q_4, q_7	q_4, q_7	q_4, q_7	q_4	q_7	q_7	q_7

▷ Claim 12. For each i, the word w_i is total.

Proof. We show by induction that the image of each state under w_i contains a state from C_i, starting with $i = 1$. By construction, a_1 is defined for every state $q \in C_0 = Q$. Since $\mathcal{A}$ is partially ordered, we get that after n applications of a_1, the image of every state must contain a state q such that $q \in q \cdot a_1$. Every such state is by definition in C_1.

Assume now that the image of every state under w_{i-1} contains a state from C_{i-1}. We thus only need to show that for each state in C_{i-1} its image under w_i contains a state from C_i. By definition, the image of every state from C_{i-1} under a_i is non-empty. Let $q \in C_{i-1}$ be a state. If $q \in C_i$, there is nothing to prove. Otherwise, $q \notin q \cdot a_i$. Thus we have that every state in $q \cdot a_i w_{i-1}$ is larger (with respect to $\prec$) than q. By the induction assumption, there is another state from C_{i-1} in $q \cdot a_i w_{i-1}$. Since $\mathcal{A}$ is partially ordered, by repeating this argument at most n times, we get a state from C_i in the image of q. ◁

▷ Claim 13. For every i, the rank of w_i is at most $|C_i|$.

Proof. For every state $q \in Q$, call *the i-trace of q* the set $q \cdot w_i$. To prove the claim, we show that for every $i \geq 1$ and every state $q \in Q$, the i-trace of q is a union of i-traces of some states from C_i. We prove that by induction on i. For $i = 1$, this is obvious from the construction of w_1 since $\mathcal{A}$ is partially ordered. Assume now that the statement holds true for $i - 1$. Then we need to show that for every $q \in Q$ and $p \in q \cdot w_{i-1}$ we have that $p \cdot (a_i w_{i-1})^n$ is a union of i-traces of some states from C_i. Observe that since $\mathcal{A}$ is partially ordered, for every $p' \in Q$ we have $p' \cdot (a_i w_{i-1})^n = p' \cdot (a_i w_{i-1})^{h(p')}$, where $h(p') = |\{s \in Q \mid p' \prec s \text{ or } p' = s\}|$. Hence, it is enough to prove that $p \cdot (a_i w_{i-1})^{h(p)}$ is a union of i-traces of some states from C_i.

We do that again by induction, now on the order $\prec$. For the largest state t with respect to $\prec$ this is obvious, since its i-trace for every i is $\{t\}$. Assume now that this is true for all states larger than p with respect to $\prec$. If $p \in C_i$, the statement is proved. Assume now that $p \notin C_i$. Then there exists $1 \leq j \leq i$ such that $p \notin p \cdot a_j$, and hence every state in $p \cdot a_i w_{i-1}$ is larger (with respect to $\prec$) than p. For every state p' such that $p \prec p'$, we have that $h(p') < h(p)$, hence we can use the assumption of the induction on $\prec$, which concludes the proof. ◁

Hence we get that w_k is a total word of rank $|C_k|$. If $\mathcal{A}$ is an rpoNFA, the rank of every total word in $\mathcal{A}$ is at least $|C_k|$. Indeed, by the definition of the algorithm, each letter in Σ either induces a self-loop for every state of C_k, or is undefined for at least one state in C_k. Since $\mathcal{A}$ is an rpoNFA, this means that if the action of a word w is defined for every state of $\mathcal{A}$, then for every state $q \in C_k$ we must have $q \cdot w = \{q\}$. Since $\mathcal{A}$ is partially ordered, every set $q \cdot w$ can only contain q and states that are larger with respect to $\prec$ than q, hence the rank of w is at least $|C_k|$. Observe that the length of w_i is $(n+1)^m - 1$, which proves Theorem 11 (a) for rpoNFAs.

To deal with the general case of poNFAs instead of rpoNFAs, we continue our analysis of the stabilization algorithm, and extend it with a very similarly defined second part, which we call moving algorithm. The moving algorithm is executed after the stabilization algorithm. We start with some definitions. Recall that the result of the stabilization algorithm is the set C_k of states and the alphabet Σ_k.

Let $S \subseteq C_k$ be the set of states $q \in C_k$ such that for every letter $a \in \Sigma_k$ we have that $q \cdot a = \{q\}$. Let $\Sigma' \subseteq \Sigma \setminus \Sigma_k$ be the set of letters not from Σ_k which are defined for every state in S. Only letters from $\Sigma_k \cup \Sigma'$ can occur in a total word, since the first occurrence of any other letter will kill at least one state in S. After the stabilization algorithm provides its output, we run the moving algorithm defined as follows.

Moving algorithm

We initialize the set of currently considered states $C'_0 = C_k$ and the currently explored alphabet $\Sigma'_0 = \emptyset$. At step i, $i \geq 1$, find a letter $a'_i \in \Sigma' \setminus \Sigma'_{i-1}$ such that for at least one state $q \in C'_{i-1}$ we have $q \notin q \cdot a'_i$. If no such letter a'_i exists, stop and output $|C'_{i-1}|$. Define C'_i to be the subset of states in C'_{i-1} for which a'_i induces a self-loop, that is, $C'_i = \{q \in C'_{i-1} \mid q \in q \cdot a'_i\}$. Take $\Sigma'_i = \Sigma'_{i-1} \cup \{a'_i\}$. Go to the step $i+1$.

Correctness of the moving algorithm

Assume that the algorithm makes $k'+1$ steps, outputs $|C'_{k'}|$ and stops. Consider the words w'_i defined as follows. We take $w'_0 = (w_k)^n$. This is done to make sure that the image of every state in Q under w'_0 contains at least one state from S. For $1 \leq i \leq k'$, define $w'_i = w'_{i-1}(a'_i w'_{i-1})^n$.

First, we claim that each w'_i is total. This follows inductively from the fact that for every i, the image of every state under w'_i contains at least one state from S, and a'_i is defined for all states in S.

The fact that the rank of w'_i is at most $|C'_i|$ is proved by exactly the same argument as in the proof of Claim 13, since the construction of the words w'_i coincides with the construction of the words w_i. Observe that in the setting of Claim 13, a_i is defined for every state in C_i, but this fact is never used in the proof of this claim.

It remains to show that the rank of every total word in $\mathcal{A}$ is at least $|C'_{k'}|$. Indeed, by the definition of the moving algorithm, for each letter $a \in \Sigma$ which is defined for every state in S, we have that $q \in q \cdot a$ for every state $q \in C'_{k'}$. As argued above, only letters defined for each state in S can occur in a total word, which concludes the proof.

It is easy to see that the length of $w'_{k'}$ is at most $n(n+1)^m$. This concludes the proof of Theorem 11 (a) for general poNFAs.

To simplify the presentation, we assume that if the poNFA is not an rpoNFA, the moving algorithm is the second part of the stabilization algorithm. To comply with the notation of the stabilization algorithm, we define $C_{k+i} = C'_i$ and $\Sigma_{k+i} = \Sigma_k \cup \Sigma'_i$ for $1 \leq i \leq k'$. Observe that by construction, Σ_k and $\Sigma'_{k'}$ do not intersect, every letter in $\Sigma_{k+k'}$ induces a self-loop for every state in $C_{k+k'}$ and $w_{k+k'}$ is a total word of minimum rank. Thus, everywhere below we denote the value $k' + k$ simply by k.

Space complexity

We show how to implement a variant of the stabilization algorithm in $\mathcal{O}(m \log m + \log n)$ deterministic space. To do so, it is enough to note that the sets C_i do not have to be constructed explicitly, and each C_i can be reconstructed on the fly based only on Σ_i. Indeed, assume that the sequence $a_1, \ldots, a_i$ is already constructed and stored in the memory. Then to test if a given state q belongs to C_i, we need to check that all of $a_1, \ldots, a_i$ induce self-loops for it. Testing if a given letter a can be taken as a_i is then straightforward and only requires going through all states of $\mathcal{A}$ in an arbitrary order.

If $m = \mathcal{O}(\frac{\log n}{\log \log n})$, the bit length of the representation of one letter in Σ is $\mathcal{O}(\log \log n)$. Thus storing a sequence $a_1, \ldots, a_d$, $d \leq m$, of letters in Σ requires $\mathcal{O}(\frac{\log n}{\log \log n} \cdot \log \log n) = \mathcal{O}(\log n)$ bits, which provides a deterministic algorithm running in logarithmic space. The described algorithm can obviously also be implemented in polynomial time, and P-hardness is stated in the following lemma. This concludes the proof of Theorem 11 (b).

▶ **Lemma 14.** *Deciding if a poDFA over an alphabet of linear size in the number of its states admits a total word of rank one is* P*-hard under an* AC^0 *reduction.*

Proof. We reduce from the monotone circuit value problem. A *monotone Boolean circuit* is an acyclic digraph $C = (V, E)$ with a labeling function $L : V \to \{\wedge, \vee, \square\}$. Intuitively, the word "monotone" refers to the fact that negation gates are not allowed. Every vertex labeled by $\{\wedge, \vee\}$ is called *an inner gate* and has indegree two. Each vertex labeled by $\square$ is called *an input gate* and has outdegree one and indegree zero. Additionally, there is a designated vertex of outdegree zero, which is called *the output gate*. A vertex v with an outgoing edge to a vertex u is called the child of u.

Let k be the number of input gates in C. Then the value of C on the input $x_1, \ldots, x_k$, $x_i \in \{0, 1\}$, is defined recursively by computing the value of a gate by applying the corresponding Boolean operation to its children. Given a monotone Boolean circuit C with k input gates and the values $x_1, \ldots, x_k$, $x_i \in \{0, 1\}$, of these input gates, the monotone circuit value problem asks if the value of the output gate of C is 1. This problem is P-complete, even if the circuit in the input is topologically ordered [19].

Given $C = (V, E)$ with a labeling function L, and the input values $x_1, \ldots, x_k$, define a poDFA $\mathcal{A} = (Q, \Sigma, \delta)$ as follows. Define $Q = V \cup \{f\}$, where f is a fresh state.

We now define the letters in Σ and their action on the states to force the following behavior of $\mathcal{A}$. We start with the whole set of states active, and then make a state non-active if the corresponding gate can be evaluated to one at the current step. First, we can map every state corresponding to an input gate with value one to f. This is done by a separate letter for each gate, and such a letter leaves every other state in its place. Then we inductively proceed with $\wedge$- and $\vee$-gates, whose corresponding states can be sent to f if and only if all their children in C have value 1 (respectively, at least on child has value 1). This is done by simple gadgets guaranteeing that all the children of a gate (respectively, at least one child

of a gate) are non-active by leaving certain transitions for them undefined, see Figure 4 for an illustration of these gadgets. If by some sequence of such letter applications the state corresponding to the output gate can be made non-active, and hence the value of C is 1, then another letter mapping all other states to f can be taken. This letter cannot be taken initially, since it is undefined for the state corresponding to the output gate.

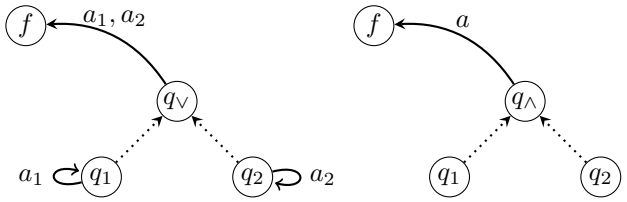

Figure 4 The gadgets for an $\vee$-gate $q_\vee$ with children q_1, q_2 (left) and an $\wedge$-gate $q_\wedge$ with children q_1, q_2 (right). The solid arrows denote transitions of the DFA, and the dotted arrows denote child-parent relations in the circuit.

We now formally define the remaining part of $\mathcal{A}$ to guarantee the described behavior.

- For each input gate v whose value is 1, add a fresh letter mapping v to f and acting as the identity for all states in Q except v.
- For each $\vee$-gate v, add two fresh letters a_1^v, a_2^v acting as follows. Both a_1^v and a_2^v maps v to f. Letter a_1^v induces a self-loop for the first child of v and is undefined for its second child. Symmetrically, a_2^v is undefined for the first child and maps the second child to itself. For all remaining states both a_1^v and a_2^v induce self-loops.
- Similarly, for each $\wedge$-gate v, add a fresh letter a^v to Σ such that a^v maps v to f, is undefined for both children of v and induces self-loops for all other states.
- Finally, add a fresh letter r which is undefined for the state corresponding to the output gate and maps all other states to f.

Applying a total word to $\mathcal{A}$ now resembles sequential evaluation of C: starting from the states corresponding to the input gates, we can make the states corresponding to gates with value one non-active by mapping them to f, and then proceed inductively to their parents. By definition of the action of r, $\mathcal{A}$ has a total word of rank one if and only if the state corresponding to the output gate eventually becomes non-active, which is possible if and only if the value of the output gate in C is one. ◀

Time complexity

To implement the stabilization algorithm with low time complexity, we use a data structure which we simply refer to as a list of states. We implement it as an array of length n that indicates for each state if it appears in the list, and also refers to the previous and next states in the array with respect to $\prec$, if they exist. Hence, testing non-emptiness of such a list can be done in constant time, going through its elements takes time linear in the size of the list (and not in n), and removing an element takes constant time.

We first show how to compute the minimum rank of a total word in time complexity $\mathcal{O}(mn)$. For each letter $a \in \Sigma$ we maintain a list D_a. After step i, this list contains the states in C_i for which a is undefined. We also maintain a list D of letters $a \in \Sigma \setminus \Sigma_i$ for which D_a is currently empty and there is a state $q \in C_i$ which is mapped by a to a different state.

The algorithm makes at most m steps. At step i, we take an arbitrary letter from D as a_i. We then update the set of currently considered states to C_i, which takes $\mathcal{O}(n)$ time, and remove the entry corresponding to each state in $C_{i-1} \setminus C_i$ from all the lists D_a, $a \in \Sigma \setminus \Sigma_i$. Observe that the entry for each state is removed at most once during the execution of the algorithm. Hence, the overall time complexity is $\mathcal{O}(mn + |\Delta|)$, which proves Theorem 11 (c).

Ergodicity

Recall that a word is ergodic if and only if its matrix has a column consisting of all ones. Clearly, in the case of poNFAs, this can only be the last column. We now apply the obtained results to ergodic words for a poNFA $\mathcal{A}$. Let Σ' be the set of letters in Σ which are defined for every state in C_k obtained in the stabilization algorithm. Let $C_k = \{q_1, q_2, \ldots, q_r\}$ such that $q_1 \prec q_2 \prec \ldots \prec q_r$.

Observe that there exists an ergodic word for $\mathcal{A}$ if and only if for every state q_i, $1 \leq i \leq r-1$, there exists a letter $a'_i \in \Sigma'$ such that there is a state $p \in Q$, $q \prec p$, with $p \in q \cdot a'_i$. If such a letter does not exist for some state in $C_k \setminus \{q_r\}$, an ergodic word for $\mathcal{A}$ obviously does not exist. For the opposite direction, observe that for every state in Q its image under $a'_1 w_k$ contains a state from $C_k \setminus \{q_1\}$. Moreover, $q_r \cdot a'_1 w_k = \{q_r\}$, since q_r must be the largest state in Q with respect to $\prec$. By inductively repeating this argument $r-1$ times in total, we get that for every state in Q its image under the concatenation of the words $(a'_i w_k)$ for $1 \leq i \leq r-1$ must contain q_k, and hence this word is ergodic. Since $r \leq n$, its length is at most $(n-1)n(n+1)^m$. We thus obtain the following theorem.

▶ **Theorem 15.** *Let $\mathcal{A}$ be a poNFA with n states and m letters.*
(a) *The length of its shortest ergodic word is at most $(n-1)n(n+1)^m$.*
(b) *Checking if $\mathcal{A}$ is ergodic can be done in $\mathcal{O}(m \log m + \log n)$ deterministic space. If $m = \mathcal{O}(\frac{\log n}{\log \log n})$, this problem is in L, and if $m = \Theta(n)$, it is P-complete.*
(c) *Checking if $\mathcal{A}$ is ergodic can be done in time $\mathcal{O}(mn + |\Delta|)$.*

4.2 Partially ordered DFAs

For poDFAs, a total word has rank one if and only if it is ergodic. Thus the following theorem, which is the main result of this subsection, applies to ergodic words if one takes $r = 1$.

▶ **Theorem 16.** *Let $\mathcal{A}$ be a poDFA with n states and m letters.*
(a) *The length of a shortest total word of minimum rank r in $\mathcal{A}$ is at most $\frac{k-1}{2k}(n-r)^2+(n-r)$, where $k = \min\{n-r, m\}$. If m divides $n-r$, the bound is tight.*
(b) *Finding such a word of at most this length can be done in time $\mathcal{O}(mn^2)$.*

We remark on the values that the bound $\frac{k-1}{2k}(n-r)^2+(n-r)$ takes depending on the size of the alphabet, assuming that $r = 1$. If $m = 2$, the lower bound is $\frac{1}{4}(n-1)(n+3) = \frac{1}{4}n^2+\mathcal{O}(n)$, while for $m = n-1$ it becomes $\frac{1}{2}n(n-1)$. As we will see later, having more than $n-1$ letters in the alphabet cannot increase the bound.

Proof. We start with proving (a). We continue the analysis of the stabilization algorithm for rpoNFAs from Section 4.1. In the proof of its correctness, we constructed a very long word w_k. However, intuitively, during the application of this word to a poDFA, most letters do not change the image of any state at the moment of their application, and each such letter can thus be removed from the word. For $1 \leq i \leq k$, denote $|C_i| = c_i$

Put a token onto each state of $\mathcal{A}$ and move them according to the transitions taken during the application of w_k letter by letter. We are going to track the paths of these tokens. First, we claim that for each state, if a token appears on it and then leaves it, it leaves it every

time along the transition labeled by the same letter. Indeed, by construction of w_k, each state $q \in Q \setminus C_k$ belongs to a set $C_{i-1} \setminus C_i$ for exactly one value of i, and thus we can show by induction that a_i is the letter labeling such a transition from q. Given a state $q \in Q$, we call such a transition *the moving transition of* q. The number of moving transitions labeled by a_i is $|C_{i-1} \setminus C_i| = c_{i-1} - c_i$. The total number of moving transitions labeled by letters in Σ_i is thus $(c_0 - c_1) + (c_1 - c_2) + \ldots + (c_{i-1} - c_i) = c_0 - c_i$.

Now, consider what happens when w_{i-1} is already applied to $\mathcal{A}$, and we start applying $(a_i w_{i-1})^n$. Observe that at this moment the set of states containing a token is precisely the set C_{i-1}, which can be proved by induction. By the application of $(a_i w_{i-1})^n$, each token on a state in $C_{i-1} \setminus C_i$ is moved to a state in C_i. The number of letters in $(a_i w_{i-1})^n$ whose application moves at least one token is thus upper bounded by the number of moved tokens, which is $|C_{i-1} \setminus C_i|$, multiplied by the number of moving transitions labeled by letters in Σ_{i-1}, plus at most one application of a_i per state in $C_{i-1} \setminus C_i$. Hence we get that the number of useful letters in $(a_i w_{i-1})^n$ is at most $(c_{i-1} - c_i)(c_0 - c_{i-1}) + (c_{i-1} - c_i) = (c_{i-1} - c_i)(c_0 - c_{i-1} + 1)$.

By construction, $c_0 = n$ and $c_k = r$. The overall length of the word consisting only of letters in w_k moving at least one token is upper bounded by

$$\begin{aligned}
\sum_{1 \le i \le k} (c_{i-1} - c_i)(c_0 - c_{i-1} + 1) &= (c_0 + 1)(c_0 - c_k) - \Big(\sum_{1 \le i \le k} c_{i-1}^2 - \sum_{1 \le i \le k} c_{i-1} c_i\Big) \\
&= (c_0 + 1)(c_0 - c_k) - \Big(\frac{1}{2}(c_0^2 - c_k^2) + \frac{1}{2} \sum_{1 \le i \le k} (c_{i-1} - c_i)^2\Big) \\
&\le (c_0 + 1)(c_0 - c_k) - \frac{1}{2}(c_0^2 - c_k^2) - \frac{1}{2k}(c_0 - c_k)^2 = \frac{k-1}{2k}(n-r)^2 + (n-r).
\end{aligned}$$

Hence the upper bound is proved. The fact that if m divides $n - r$ the bound is tight follows from the following lemma, concluding the proof of (a).

▶ **Lemma 17.** *For every positive natural numbers n, m and r such that m divides $n - 1$, there exists a poDFA $\mathcal{A}$ with n states and m letters such that the length of its shortest total word of minimum rank r is $\frac{m-1}{2m}(n-r)^2 + (n-r)$.*

Proof. We start with $r = 1$. We construct a poDFA $\mathcal{A} = (Q, \Sigma, \delta)$ as follows. Let $n = ms + 1$ and $Q^{(j)} = \{q_1^{(j)}, \ldots, q_s^{(j)}\}$. Take $Q = \cup_{1 \le j \le m} Q^{(j)} \cup \{q_1^{(m+1)}\}$, and $\Sigma = \{a_1, a_2, \ldots, a_m\}$. For each $1 \le j \le m$, define $\delta(q_i^{(j)}, a_j) = q_{i+1}^{(j)}$ for $1 \le i < s$, $\delta(q_s^{(j)}, a_j) = q_1^{(j+1)}$ and $\delta(q_i^{(j)}, a_\ell) = q_i^{(j)}$ if $j < \ell$ and $1 \le i \le s$. Set each letter in Σ to induce a self-loop for $q_1^{(m+1)}$. Leave all remaining transitions undefined. See Figure 5 for an example.

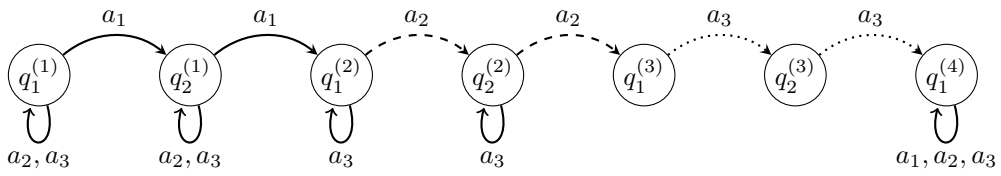

Figure 5 Example of the construction in the proof of Lemma 17 for $m = 3$, $s = 2$ and $r = 1$.

Observe that there exists a unique shortest total word of rank one for $\mathcal{A}$. Indeed, at every moment, only one transition that does not act as the identity can be taken. Namely, if no state in $Q^{(j)}$ is active and some state in $Q^{(j-1)}$ is active, we can take a_j, and as soon as one state in $Q^{(j)}$ becomes active, it has to be sent all the way to $a_1^{(m+1)}$ before any other state

can be sent to another state. Hence, the length of a shortest total word of rank one for $\mathcal{A}$ is $s+(s^2+s)+(2s^2+s)+\ldots+((m-1)s^2+s) = s^2(1+2+\ldots+(m-1))+sm = s^2 \cdot \frac{(m-1)m}{2}+ms$. Since $s = \frac{n-1}{m}$, this value is equal to $\frac{m-1}{2m}(n-1)^2 + (n-1)$.

Finally, observe that adding $r-1$ states such that every letter induces a self-loop for each of them provides a required construction for the case of rank r. ◀

We continue with proving item (b) of Theorem 16. To construct a total word of minimum rank within the required length bound, recall the definition of the moving transitions from the length analysis above. For each state $q \in Q$, find the letter labeling its moving transition. Now we follow the tokens. At step i, all tokens on states in C_i are not moved, so we only need to compute a word that maps all tokens on the states in $C_{i-1} \setminus C_i$ to states in C_i. To do so, we need to decide which letters from $(a_iw_i)^n$ move at least one token. Clearly, if a_i does not move anything, we go to the next step. Otherwise, we apply a_i and recursively perform the argument from step $i-1$ for the moved tokens. The total number of moments of time when we do not move anything and thus go to the next step is upper bounded by $\mathcal{O}(mn^2)$ by the construction of w_k, and the number of moves of tokens is upper bounded by $\frac{m-1}{2m}(n-r)^2 + (n-r) = \mathcal{O}(mn^2)$. This concludes the proof of (b). ◀

References

1 László Babai, Robert Beals, Jin-yi Cai, Gábor Ivanyos, and Eugene M. Luks. Multiplicative equations over commuting matrices. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '96, pages 498–507, USA, 1996. Society for Industrial and Applied Mathematics.

2 Jørgen Bang-Jensen and Gregory Z. Gutin. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, 2008.

3 R. B. Bapat and T. E. S. Raghavan. *Nonnegative Matrices and Applications*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1997.

4 Marie-Pierre Béal, Eugen Czeizler, Jarkko Kari, and Dominique Perrin. Unambiguous automata. *Mathematics in Computer Science*, 1:625–638, 2008. doi:10.1007/s11786-007-0027-1.

5 Marie-Pierre Béal and Dominique 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.

6 M. Beaudry, P. McKenzie, and D. Thérien. The membership problem in aperiodic transformation monoids. *Journal of the ACM*, 39(3):599–616, 1992.

7 Martin Beaudry. Membership testing in commutative transformation semigroups. *Information and Computation*, 79(1):84–93, 1988. doi:10.1016/0890-5401(88)90018-1.

8 Martin Beaudry. Membership testing in transformation monoids. *PhD thesis, McGill University, Montreal, Quebec*, 1988.

9 Martin Beaudry. Membership testing in threshold one transformation monoids. *Information and Computation.*, 113(1):1–25, 1994. doi:10.1006/INCO.1994.1062.

10 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.

11 Mikhail V. Berlinkov, Robert Ferens, Andrew Ryzhikov, and Marek Szykuła. Synchronizing Strongly Connected Partial DFAs. In Markus Bläser and Benjamin Monmege, editors, *38th International Symposium on Theoretical Aspects of Computer Science (STACS 2021)*, volume 187 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:16, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.STACS.2021.12.

12 Jean Berstel, Dominique Perrin, and Christophe Reutenauer. *Codes and automata*, volume 129 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2010.

13 Michael Blondin, Andreas Krebs, and Pierre McKenzie. The complexity of intersecting finite automata having few final states. *computational complexity*, 25(4):775–814, 2016. doi:10.1007/s00037-014-0089-9.

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

15 Pierre-Yves Chevalier, Vladimir V. Gusev, Raphaël M. Jungers, and Julien M. Hendrickx. Sets of stochastic matrices with converging products: Bounds and complexity. *CoRR*, abs/1712.02614, 2017. arXiv:1712.02614.

16 Pierre-Yves Chevalier, Julien M. Hendrickx, and Raphaël M. Jungers. Reachability of consensus and synchronizing automata. In *54th IEEE Conference on Decision and Control, CDC 2015, Osaka, Japan, December 15-18, 2015*, pages 4139–4144. IEEE, 2015. doi:10.1109/CDC.2015.7402864.

17 V. Froidure. Ranks of binary relations. *Semigroup Forum*, 54:381–401, 1997. doi:10.1007/BF02676619.

18 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.

19 Raymond Greenlaw, H. James Hoover, and Walter L. Ruzzo. *Limits to parallel computation: P-completeness theory*. Oxford University Press, 1995.

20 Balázs Imreh and Magnus Steinby. Directable nondeterministic automata. *Acta Cybernetica*, 14(1):105–115, 1999.

21 Masami Ito and Kayoko Shikishima-Tsuji. Shortest directing words of nondeterministic directable automata. *Discrete Mathematics*, 308(21):4900–4905, 2008. doi:10.1016/J.DISC.2007.09.010.

22 R. Kannan and R. J. Lipton. Polynomial-time algorithm for the orbit problem. *Journal of the ACM*, 33(4):808–821, 1986. doi:10.1145/6490.6496.

23 Jarkko Kari and Mikhail V. Volkov. Černý's conjecture and the road colouring problem. In *Handbook of Automata Theory*, pages 525–565. EMS Press, 2021. doi:10.4171/AUTOMATA-1/15.

24 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.

25 Dexter Kozen. Lower bounds for natural proof systems. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 254–266. IEEE Computer Society, 1977. doi:10.1109/SFCS.1977.16.

26 Markus Krötzsch, Tomáš Masopust, and Michaël Thomazo. Complexity of universality and related problems for partially ordered NFAs. *Information and Computation*, 255:177–192, 2017. doi:10.1016/J.IC.2017.06.004.

27 Douglas Lind and Brian Marcus. *An introduction to symbolic dynamics and coding*. Cambridge University Press, 2021.

28 Alexei Lisitsa and Igor Potapov. Membership and reachability problems for row-monomial transformations. In Jiří Fiala, Václav Koubek, and Jan Kratochvíl, editors, *Mathematical Foundations of Computer Science 2004*, pages 623–634, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

29 Pavel Martyugin. Computational complexity of certain problems related to carefully synchronizing words for partial automata and directing words for nondeterministic automata. *Theory Comput. Syst.*, 54(2):293–304, 2014. doi:10.1007/S00224-013-9516-6.

30 Tomáš Masopust and Markus Krötzsch. Partially Ordered Automata and Piecewise Testability. *Logical Methods in Computer Science*, 17(2):14:1–14:36, 2021. doi:10.23638/LMCS-17(2:14)2021.

31 Tomáš Masopust and Michaël Thomazo. On boolean combinations forming piecewise testable languages. *Theor. Comput. Sci.*, 682:165–179, 2017. doi:10.1016/J.TCS.2017.01.017.

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

33 Jean-Eric Pin and Howard Straubing. Monoids of upper triangular boolean matrices. In *Semigroups. Structure and Universal AIgebraic Problems*, volume 39, pages 259–272, 1981.

34 Igor Potapov and Pavel Semukhin. Decidability of the membership problem for 2×2 integer matrices. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 170–186. SIAM, 2017. doi:10.1137/1.9781611974782.12.

35 Vladimir Yu. Protasov. Analytic methods for reachability problems. *J. Comput. Syst. Sci.*, 120:1–13, 2021. doi:10.1016/J.JCSS.2021.02.007.

36 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.

37 Andrew Ryzhikov. Synchronization problems in automata without non-trivial cycles. *Theoretical Computer Science*, 787:77–88, 2019.

38 Eugene Seneta. *Non-negative matrices and Markov chains*. Springer Science & Business Media, 2006.

39 Michael Sipser. *Introduction to the Theory of Computation*. Course Technology, Boston, MA, third edition, 2013.

40 Seinosuka Toda. On the complexity of topological sorting. *Information Processing Letters*, 35(5):229–233, 1990. doi:10.1016/0020-0190(90)90050-8.

41 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.

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

43 Petra Wolf. Synchronization under dynamic constraints. In *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2020)*, 2020.

44 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.