Decomposing Permutation Automata

Ismaël Jecker
Institute of Science and Technology, Klosterneuburg, Austria

Nicolas Mazzocchi
IMDEA Software Institute, Madrid, Spain

Petra Wolf
Fachbereich IV, Informatikwissenschaften, Universität Trier, Germany

Abstract

A deterministic finite automaton (DFA) $\mathcal{A}$ is composite if its language $L(\mathcal{A})$ can be decomposed into an intersection $\bigcap_{i=1}^{k} L(\mathcal{A}_i)$ of languages of smaller DFAs. Otherwise, $\mathcal{A}$ is prime. This notion of primality was introduced by Kupferman and Mosheiff in 2013, and while they proved that we can decide whether a DFA is composite, the precise complexity of this problem is still open, with a doubly-exponential gap between the upper and lower bounds. In this work, we focus on permutation DFAs, i.e., those for which the transition monoid is a group. We provide an NP algorithm to decide whether a permutation DFA is composite, and show that the difficulty of this problem comes from the number of non-accepting states of the instance: we give a fixed-parameter tractable algorithm with the number of rejecting states as the parameter. Moreover, we investigate the class of commutative permutation DFAs. Their structural properties allow us to decide compositionality in NL, and even in LOGSPACE if the alphabet size is fixed. Despite this low complexity, we show that complex behaviors still arise in this class: we provide a family of composite DFAs each requiring polynomially many factors with respect to its size. We also consider the variant of the problem that asks whether a DFA is *k-factor composite*, that is, decomposable into k smaller DFAs, for some given integer $k \in \mathbb{N}$. We show that, for commutative permutation DFAs, restricting the number of factors makes the decision computationally harder, and yields a problem with tight bounds: it is NP-complete. Finally, we show that in general, this problem is in PSPACE, and it is in LOGSPACE for DFAs with a singleton alphabet.

2012 ACM Subject Classification Theory of computation → Regular languages; Theory of computation → Problems, reductions and completeness

Keywords and phrases Deterministic finite automata (DFA), Permutation automata, Commutative languages, Decomposition, Regular Languages, Primality

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2021.18

Related Version A version of this paper with all proofs is available at `https://arxiv.org/abs/2107.04683`

Funding *Ismaël Jecker*: Marie Skłodowska-Curie Grant Agreement No. 754411.
Nicolas Mazzocchi: BOSCO project PGC2018-102210-B-I00 (MCIU/AEI/FEDER, UE), BLOQUES-CM project S2018/TCS-4339, and MINECO grant RYC-2016-20281.
Petra Wolf: DFG project FE 560/9-1.

1 Introduction

Compositionality is a fundamental notion in numerous fields of computer science [3]. This principle can be summarised as follows: Every system should be designed by composing simple parts such that the meaning of the system can be deduced from the meaning of its parts, and how they are combined. For instance, this is a crucial aspect of modern software engineering: a program split into simple modules will be quicker to compile and easier to maintain. The use of compositionality is also essential in theoretical computer

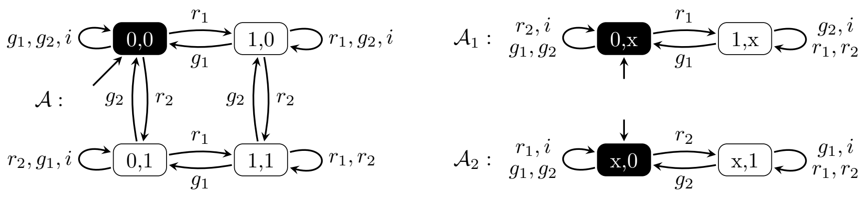

Figure 1 DFAs recognising specifications. Accepting states are drawn in black. The DFAs $\mathcal{A}_1$ and $\mathcal{A}_2$ check that every request of the first, resp. second, client is eventually granted, $\mathcal{A}$ checks both.

science: it is used to avoid the *state explosion* issues that usually happen when combining parallel processes together, and also to overcome the *scalability* issues of problems with a high theoretical complexity. In this work, we study compositionality in the setting of formal languages: we show how to make languages simpler by decomposing them into *intersections* of smaller languages. This is motivated by the *model-checking* problems. For instance, the LTL model-checking problem asks, given a linear temporal logic formula φ and a finite state machine M, whether every execution of M satisfies φ. This problem is decidable, but has a high theoretical complexity (PSPACE) with respect to the size of φ [1]. If φ is too long, it cannot be checked efficiently. This is where compositionality comes into play: if we can decompose the specification language into an intersection of simple languages, that is, decompose φ into a conjunction $\varphi = \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_k$ of small specifications, it is sufficient to check whether all the φ_i are satisfied separately.

Our aim is to develop the theoretical foundations of the compositionality principle for formal languages by investigating how to decompose into simpler parts one of the most basic model of abstract machines: deterministic finite automata (DFAs). We say that a DFA $\mathcal{A}$ is *composite* if its language can be decomposed into the intersection of the languages of smaller DFAs. More precisely, we say that $\mathcal{A}$ is *k-factor composite* if there exist k DFAs $(\mathcal{A}_i)_{1 \leq i \leq k}$ with less states than $\mathcal{A}$ such that $L(\mathcal{A}) = \bigcap_{i=1}^{k} L(\mathcal{A}_i)$. We study the two following problems:

DFA Decomp
Given: DFA $\mathcal{A}$.
Question: Is $\mathcal{A}$ composite?

DFA Bound-Decomp
Given: DFA $\mathcal{A}$ and integer $k \in \mathbb{N}$.
Question: Is $\mathcal{A}$ k-factor composite?

The next example shows that decomposing DFAs can result in substantially smaller machines.

Example. Consider Figure 1. We simulate the interactions between a system and two clients by using finite words on the alphabet $\{r_1, r_2, g_1, g_2, i\}$: At each time step, the system either receives a request from a client (r_1, r_2), grants the open requests of a client (g_1, g_2), or stays idle (i). A basic property usually required is that every request is eventually granted. This specification is recognised by the DFA $\mathcal{A}$, which keeps track in its state of the current open requests, and only accepts if none is open when the input ends. Alternatively, this specification can be decomposed into the intersection of the languages defined by the DFAs $\mathcal{A}_1$ and $\mathcal{A}_2$: each one checks that the requests of the corresponding client are eventually granted. While in this precise example both ways of defining the specification are comparable, the latter scales drastically better than the former when the number of clients increases: Suppose that there are now $n \in \mathbb{N}$ clients. In order to check that all the requests are granted with a single DFA, we need 2^n states to keep track of all possible combinations of open requests, which is impractical when n gets too big. However, decomposing this specification into an intersection yields n DFAs of size two, one for each client. Note that, while in this specific example the decomposition is obvious, in general computing such a conjunctive form can be challenging: currently the best known algorithm needs exponential space.

	Decomp	Bound-Decomp
DFAs	EXPSPACE [9]	**PSPACE**
Permutation DFAs	**NP/FPT**	**PSPACE**
Commutative permutation DFAs	**NL**	**NP-complete**
Unary DFAs	LOGSPACE [7]	**LOGSPACE**

Figure 2 Complexity of studied problems with containing classes, with our contribution in **bold**.

DFAs in hardware. Our considered problems are of great interest in hardware implementations of finite state machines [13] where realizing large DFAs poses a challenge [5]. In [2] the authors describe a state machine language for describing complex finite state hardware controllers, where the compiled state tables can automatically be input into a temporal logic model checker. If the control mechanism of the initial finite state machine can be split up into a conjunction of constraints, considering a decomposition instead could improve this work-flow substantially. Decomposing a complex DFA $\mathcal{A}$ can lead to a smaller representation of the DFA in total, as demonstrated in the previous example in Figure 1, and on top of that the individual smaller DFAs $\mathcal{A}_i$ in the decomposition $L(\mathcal{A}) = \bigcap_{i=1}^{k} L(\mathcal{A}_i)$ can be placed independently on a circuit board, as they do not have to interact with each other and only need to read their common input from a global bus and signal acceptance as a flag to the bus. This allows for a great flexibility in circuit designs, as huge DFAs can be broken down into smaller blocks which fit into niches giving space for inflexible modules such as CPU cores.

Reversible DFAs. We focus our study on *permutation* DFAs, which are DFAs whose transition monoids are groups: each letter induces a one-to-one map from the state set into itself. These DFAs are also called *reversible* DFAs [8, 14]. Reversibility is stronger than determinism: this powerful property allows to deterministically navigate *back and forth* between the steps of a computation. This is particularly relevant in the study of the physics of computation, since irreversibility causes energy dissipation [10]. Remark that in the setting of DFAs, this power results in a loss of expressiveness: contrary to more powerful models (for instance Turing machines), reversible DFAs are less expressive than general DFAs.

Related work. The DFA Decomp problem was first introduced in 2013 by Kupferman and Moscheiff [9]. They proved that it is decidable in EXPSPACE, but left open the exact complexity: the best known lower bound is hardness for NL. They gave more efficient algorithms for restricted domains: a PSPACE algorithm for *permutation* DFAs, and a PTIME algorithm for *normal* permutation DFAs, a class of DFAs that contains all *commutative* permutation DFAs. Recently, the Decomp problem was proved to be decidable in LOGSPACE for DFAs with a singleton alphabet [7]. The trade-off between number and size of factors was studied in [12], where automata showing extreme behavior are presented, i.e., DFAs that can either be decomposed into a large number of small factors, or a small number of large factors.

Contribution. We expand the domain of instances over which the Decomp problem is tractable. We focus on permutation DFAs, and we propose new techniques that improve the known complexities. All proofs omitted due to space restrictions can be found in the full version. Unless specified otherwise, the complexity of our algorithms do not depend on the size of the alphabet of the DFA. Our results, summarised by Figure 2, are presented as follows.

Section 3. We give an NP algorithm for permutation DFAs, and we show that the complexity is directly linked to the number of non-accepting states. This allows us to obtain a fixed-parameter tractable algorithm with respect to the number of non-accepting states (Theorem 1). Moreover, we prove that permutation DFAs with a prime number of states cannot be decomposed (Theorem 2).

Section 4. We consider *commutative* permutation DFAs, where the Decomp problem was already known to be tractable, and we lower the complexity from PTIME to NL, and even LOGSPACE if the size of the alphabet is fixed (Theorem 9). While it is easy to decide whether a commutative permutation DFA is composite, we show that rich and complex behaviours still appear in this class: there exist families of composite DFAs that require polynomially many factors to get a decomposition. More precisely, we construct a family $(\mathcal{A}_n^m)_{m,n\in\mathbb{N}}$ of composite DFAs such that $\mathcal{A}_n^m$ is a DFA of size n^m that is $(n-1)^{m-1}$-factor composite but not $(n-1)^{m-1}-1$-factor composite (Theorem 10). Note that, prior to this result, only families of composite DFAs with sublogarithmic width were known [7].

Section 5. Finally, we study the Bound-Decomp problem. High widths are undesirable for practical purposes: dealing with a huge number of small DFAs might end up being more complex than dealing with a single DFA of moderate size. The Bound-Decomp problem copes with this issue by limiting the number of factors allowed in the decompositions. We show that this flexibility comes at a cost: somewhat surprisingly, this problem is NP-complete for commutative permutation DFAs (Theorem 17), a setting where the Decomp problem is easy. We also show that this problem is in PSPACE for the general setting (Theorem 16), and in LOGSPACE for unary DFAs i.e. with a singleton alphabet (Theorem 18).

2 Definitions

We denote by $\mathbb{N}$ the set of non-negative integers $\{0, 1, 2, \ldots\}$. For a word $w = w_1w_2 \ldots w_n$ with $w_i \in \Sigma$ for $1 \leq i \leq n$, we denote with $w^R = w_n \ldots w_2w_1$ the *reverse* of w. Moreover, for every $\sigma \in \Sigma$, we denote by $\#_\sigma(w)$ the number of times the letter σ appears in w. A natural number $n > 1$ is called *composite* if it is the product of two smaller numbers, otherwise we say that n is *prime*. Two integers $m, n \in \mathbb{N}$ are called *co-prime* if their greatest common divisor is 1. We will use the following well known results [6, 11]:

Bertrand's Postulate. For all $n > 3$ there is a prime number p satisfying $n < p < 2n - 2$.

Bézout's Identity. For every pair of integers $m, n \in \mathbb{N}$, the set $\{\lambda m - \mu n \mid \lambda, \mu \in \mathbb{N}\}$ contains exactly the multiples of the greatest common divisor of m and n.

Deterministic finite automata. A *deterministic finite automaton* (DFA hereafter) is a 5-tuple $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F\rangle$, where Q is a finite set of states, Σ is a finite non-empty alphabet, $\delta\colon Q \times \Sigma \to Q$ is a transition function, $q_I \in Q$ is the initial state, and $F \subseteq Q$ is a set of accepting states. The states in $Q \setminus F$ are called *rejecting* states. We extend δ to words in the expected way, thus $\delta\colon Q \times \Sigma^* \to Q$ is defined recursively by $\delta(q, \varepsilon) = q$ and $\delta(q, w_1w_2 \cdots w_n) = \delta(\delta(q, w_1w_2 \cdots w_{n-1}), w_n)$. The *run* of $\mathcal{A}$ on a word $w = w_1 \ldots w_n$ is the sequence of states $s_0, s_1, \ldots, s_n$ such that $s_0 = q_I$ and for each $1 \leq i \leq n$ it holds that $\delta(s_{i-1}, w_i) = s_i$. Note that $s_n = \delta(q_I, w)$. The DFA $\mathcal{A}$ *accepts* w iff $\delta(q_I, w) \in F$. Otherwise,

$\mathcal{A}$ *rejects* w. The set of words accepted by $\mathcal{A}$ is denoted $L(\mathcal{A})$ and is called the *language of* $\mathcal{A}$. A language accepted by some DFA is called a *regular language*.

We refer to the size of a DFA $\mathcal{A}$, denoted $|\mathcal{A}|$, as the number of states in $\mathcal{A}$. A DFA $\mathcal{A}$ is *minimal* if every DFA $\mathcal{B}$ such that $L(\mathcal{B}) = L(\mathcal{A})$ satisfies $|\mathcal{B}| \geq |\mathcal{A}|$.

Composite DFAs. We call a DFA $\mathcal{A}$ *composite* if there exists a family $(\mathcal{B}_i)_{1\leq i\leq k}$ of DFAs with $|\mathcal{B}_i| < |\mathcal{A}|$ for all $1 \leq i \leq k$ such that $L(\mathcal{A}) = \bigcap_{1\leq i\leq k} L(\mathcal{B}_i)$ and call the family $(\mathcal{B}_i)_{1\leq i\leq k}$ a *decomposition* of $\mathcal{A}$. Note that, all $\mathcal{B}_i$ in the decomposition satisfy $|\mathcal{B}_i| < |\mathcal{A}|$ and $L(\mathcal{A}) \subseteq L(\mathcal{B}_i)$. Such DFAs are called *factors* of $\mathcal{A}$, and $(\mathcal{B}_i)_{1\leq i\leq k}$ is also called a *k-factor decomposition* of $\mathcal{A}$. The *width* of $\mathcal{A}$ is the smallest k for which there is a k-factor decomposition of $\mathcal{A}$, and we say that $\mathcal{A}$ is *k-factor composite* iff $\mathit{width}(\mathcal{A}) \leq k$. We call a DFA $\mathcal{A}$ *prime* if it is not composite. We call a DFA $\mathcal{A}$ *trim* if all of its states are accessible from the initial state. As every non-trim DFA $\mathcal{A}$ is composite, we assume all given DFAs to be trim in the following.

We call a DFA a *permutation DFA* if for each letter $\sigma \in \Sigma$, the function mapping each state q to the state $\delta(q, \sigma)$ is a bijection. For permutation DFAs the transition monoid is a group. Further, we call a DFA $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F\rangle$ a *commutative DFA* if $\delta(q, uv) = \delta(q, vu)$ for every state q and every pair of words $u, v \in \Sigma^*$. In the next sections we discuss the problem of being composite for the classes of permutation DFA, and commutative permutation DFAs.

3 Decompositions of Permutation DFAs

In this section, we study permutation DFAs. Our main contribution is an algorithm for the DECOMP problem that is FPT with respect to the number of rejecting states:

▶ **Theorem 1.** *The* DECOMP *problem for permutation DFAs is in NP. It is in FPT with parameter k, being the number of rejecting states of DFA $\mathcal{A}$, solvable in time $\mathcal{O}(2^k k^2 \cdot |\mathcal{A}|)$.*

We prove Theorem 1 by introducing the notion of *orbit-DFAs*: an orbit-DFA $\mathcal{A}^U$ of a DFA $\mathcal{A}$ is the DFA obtained by fixing a set of states U of $\mathcal{A}$ as the initial state, and letting the transition function of $\mathcal{A}$ act over it (thus the states of $\mathcal{A}^U$ are subsets of the state space of $\mathcal{A}$). We prove three key results:

- A permutation DFA is composite if and only if it can be decomposed into its orbit-DFAs (Corollary 6);
- A permutation DFA $\mathcal{A}$ can be decomposed into its orbit-DFAs if and only if for each of its rejecting states q, there exists an orbit-DFA $\mathcal{A}^U$ smaller than $\mathcal{A}$ that *covers* q, that is, one of the states of $\mathcal{A}^U$ contains q and no accepting states of $\mathcal{A}$ (Lemma 7);
- Given a permutation DFA $\mathcal{A}$ and a rejecting state q, we can determine the existence of an orbit-DFA covering q in non-deterministic time $\mathcal{O}(|\mathcal{A}|^2)$, and in deterministic time $\mathcal{O}(2^k k \cdot |\mathcal{A}|)$, where k is the number of rejecting states of $\mathcal{A}$ (Lemma 8, (apx) Algorithm 1).

These results directly imply Theorem 1. We also apply them to show that the DECOMP problem is trivial for permutation DFAs with a prime number of states.

▶ **Theorem 2.** *Let $\mathcal{A}$ be a permutation DFA with at least one accepting state and one rejecting state. If the number of states of $\mathcal{A}$ is prime, then $\mathcal{A}$ is prime.*

3.1 Proof of Theorem 1

Consider a DFA $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F\rangle$. We extend δ to subsets $U \subseteq Q$ in the expected way:

$$\delta(U, w) = \{q \in Q \mid q = \delta(p, w) \text{ for some } p \in U\} \text{ for every word } w \in \Sigma^*.$$

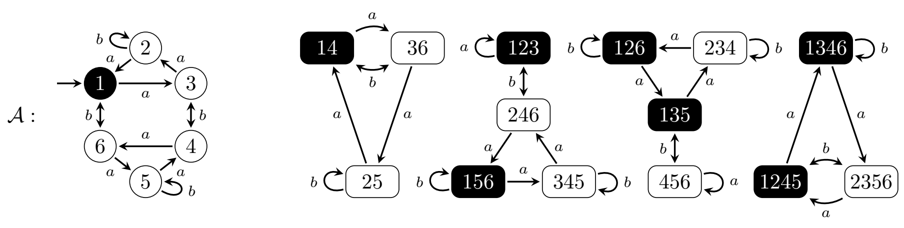

Figure 3 A DFA $\mathcal{A}$ together with some of its orbit-DFAs. Accepting states are depicted in black, an orbit-DFA can be obtained by setting a subset containing a 1 as an initial state. For instance the orbit-DFAs $\mathcal{A}^{\{1,2,3\}}$ and $\mathcal{A}^{\{1,5,6\}}$ form a decomposition of $\mathcal{A}$.

The *orbit* of U is the collection $\mathcal{C}_U = \{\delta(U, w) \subseteq Q \mid w \in \Sigma^*\}$ of subsets of Q that can be reached from U by the action of δ. If the subset $U \subseteq Q$ contains the initial state q_I of $\mathcal{A}$, we define the *orbit-DFA* $\mathcal{A}^U = \langle \Sigma, \mathcal{C}_U, U, \delta, \mathcal{C}' \rangle$, where the state space $\mathcal{C}_U$ is the orbit of U, and the set $\mathcal{C}'$ of accepting states is composed of the sets $U' \in \mathcal{C}_U$ that contain at least one of the accepting states of $\mathcal{A}$: $U' \cap F \neq \varnothing$. Note that $\mathcal{A}^U$ can alternatively be defined as the standard subset construction starting with the set $U \subseteq Q$ as initial state. The definition of the accepting states guarantees that $L(\mathcal{A}) \subseteq L(\mathcal{A}^U)$:

▶ **Proposition 3.** *Every orbit-DFA $\mathcal{A}^U$ of a DFA $\mathcal{A}$ satisfies $L(\mathcal{A}) \subseteq L(\mathcal{A}^U)$.*

Example. Let us detail the orbits of the DFA $\mathcal{A}$ depicted in Figure 3. This DFA contains six states, and generates the following non-trivial orbits on its subsets of states:

- The 15 subsets of size 2 are split into two orbits: one of size 3, and one of size 12;
- The 20 subsets of size 3 are split into three orbits: two of size 4, and one of size 12;
- The 15 subsets of size 4 are split into two orbits, one of size 3, and one of size 12.

Figure 3 illustrates the four orbits smaller than $|\mathcal{A}|$: they induce seven orbit-DFAs, obtained by setting as initial state one of the depicted subsets containing the initial state 1 of $\mathcal{A}$.

In order to prove that a DFA is composite if and only if it can be decomposed into its orbit-DFAs, we prove that every factor $\mathcal{B}$ of a permutation DFA $\mathcal{A}$ can be turned into an orbit-DFA $\mathcal{A}^U$ that is also a factor of $\mathcal{A}$, and satisfies $L(\mathcal{A}^U) \subseteq L(\mathcal{B})$. Our proof is based on a known result stating that factors can be turned into permutation DFAs:

▶ **Lemma 4** ([9, Theorem 7.4]). *Let $\mathcal{A}$ be a permutation DFA. For every factor $\mathcal{B}$ of $\mathcal{A}$, there exists a permutation DFA $\mathcal{C}$ satisfying $|\mathcal{C}| \leq |\mathcal{B}|$ and $L(\mathcal{A}) \subseteq L(\mathcal{C}) \subseteq L(\mathcal{B})$.*

We strengthen this result by showing how to transform factors into orbit-DFAs:

▶ **Lemma 5.** *Let $\mathcal{A}$ be a permutation DFA. For every factor $\mathcal{B}$ of $\mathcal{A}$, there exists an orbit-DFA $\mathcal{A}^U$ of A satisfying $|\mathcal{A}^U| \leq |\mathcal{B}|$ and $L(\mathcal{A}) \subseteq L(\mathcal{A}^U) \subseteq L(\mathcal{B})$.*

Proof. Let $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$ be a permutation DFA, and let $\mathcal{B}$ be a factor of $\mathcal{A}$. By Lemma 4, there exists a *permutation* DFA $\mathcal{B}' = \langle \Sigma, S, s_I, \eta, G \rangle$ satisfying $|\mathcal{B}'| \leq |\mathcal{B}|$ and $L(\mathcal{A}) \subseteq L(\mathcal{B}') \subseteq L(\mathcal{B})$. We build, based on $\mathcal{B}'$, an orbit-DFA $\mathcal{A}^U$ of $\mathcal{A}$ satisfying the statement.

We say that a state $q \in Q$ of $\mathcal{A}$ is *linked* to a state $s \in S$ of $\mathcal{B}'$, denoted $q \sim s$, if there exists a word $u \in \Sigma^*$ satisfying $\delta(q_I, u) = q$ and $\eta(s_I, u) = s$. Let $f : S \to 2^Q$ be the function mapping every state $s \in S$ to the set $f(s) \subseteq Q$ containing all the states $q \in Q$ that are linked to s (i.e. satisfying $q \sim s$). We set $U = f(s_I)$. In particular, the initial state q_I of $\mathcal{A}$ is in U since $\delta(q_I, \varepsilon) = q_I$ and $\eta(s_I, \varepsilon) = s_I$. We show that the orbit-DFA $\mathcal{A}^U$ satisfies the desired conditions: $|\mathcal{A}^U| \leq |\mathcal{B}'|$ and $L(\mathcal{A}) \subseteq L(\mathcal{A}^U) \subseteq L(\mathcal{B}')$.

First, we show that $|\mathcal{A}^U| \leq |\mathcal{B}'|$ by proving that the function f defined earlier maps S surjectively into the orbit of U, which is the state space of $\mathcal{A}^U$. Since both $\mathcal{A}$ and $\mathcal{B}'$ are permutation DFAs, we get that for all $q \in Q$, $s \in S$ and $a \in \Sigma$, then $q \sim s$ if and only if $\delta(q, a) \sim \eta(s, a)$ holds.[1] Therefore, for every word $v \in \Sigma^*$, $f(\eta(s_I, v)) = \delta(f(s_I), v) = \delta(U, v)$. This shows that, as required, the image of the function f is the orbit of U, and f is surjective.

To conclude, we show that $L(\mathcal{A}) \subseteq L(\mathcal{A}^U) \subseteq L(\mathcal{B}')$. Proposition 3 immediately implies that $L(\mathcal{A}) \subseteq L(\mathcal{A}^U)$. Therefore it is enough to show that $L(\mathcal{A}^U) \subseteq L(\mathcal{B}')$. Let $v \in L(\mathcal{A}^U)$. By definition of an orbit-DFA, this means that the set $\delta(U, v)$ contains an accepting state q_F of $\mathcal{A}$. Since, as stated earlier, $f(\eta(s_I, v)) = \delta(U, v)$, this implies (by definition of the function f) that the accepting state q_F of $\mathcal{A}$ is linked to $\eta(s_I, v)$, i.e., there exists a word $v' \in \Sigma^*$ such that $\delta(q_I, v') = q_F$ and $\eta(s_I, v') = \eta(s_I, v)$. Then $\delta(q_I, v') = q_F$ implies that v' is in the language of $\mathcal{A}$. Moreover, since $L(\mathcal{A}) \subseteq L(\mathcal{B}')$ by supposition, v' is also accepted by $\mathcal{B}'$, i.e., $\eta(s_I, v')$ is an accepting state of $\mathcal{B}'$. Therefore, since $\eta(q_I, v') = \eta(q_I, v)$, the word v is also in the language of $\mathcal{B}'$. This shows that $L(\mathcal{A}^U) \subseteq L(\mathcal{B}')$, which concludes the proof. ◀

As an immediate corollary, every decomposition of a permutation DFA can be transformed, factor after factor, into a decomposition into orbit-DFAs.

▶ **Corollary 6.** *A permutation DFA is composite if and only if it can be decomposed into its orbit-DFAs.*

Orbit cover. Given a rejecting state $q \in Q \setminus F$ of $\mathcal{A}$, we say that the orbit-DFA $\mathcal{A}^U$ *covers* q if $|\mathcal{A}^U| < |\mathcal{A}|$, and $\mathcal{A}^U$ contains a rejecting state $U' \subseteq Q$ that contains q. Remember that, by definition, this means that U' contains no accepting state of $\mathcal{A}$, i.e., $U' \cap F = \varnothing$. We show that permutation DFAs that can be decomposed into their orbit-DFAs are characterized by the existence of orbit-DFAs covering each of their rejecting states.

▶ **Lemma 7.** *A permutation DFA $\mathcal{A}$ is decomposable into its orbit-DFAs if and only if every rejecting state of $\mathcal{A}$ is covered by an orbit-DFA $\mathcal{A}'$ of $\mathcal{A}$ satisfying $|\mathcal{A}'| < |\mathcal{A}|$.*

Proof. Let $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$ be a permutation DFA. We prove both implications.

Suppose that $\mathcal{A}$ can be decomposed into its orbit-DFAs $(\mathcal{A}^{U_i})_{1 \leq i \leq k}$, and let $q \in Q \setminus F$ be a rejecting state of $\mathcal{A}$. We show that q is covered by every orbit-DFA $\mathcal{A}^{U_i}$ that rejects a word $w \in \Sigma^*$ satisfying $\delta(q_I, w) = q$. Formally, let $w \in \Sigma^*$ be a word satisfying $\delta(q_I, w) = q$. Then $w \notin L(\mathcal{A}) = \bigcap_{i=1}^n L(\mathcal{A}^{U_i})$, hence there exists $1 \leq i \leq n$ such that $w \notin L(\mathcal{A}^{U_i})$. Let $U' \subseteq Q$ be the state visited by $\mathcal{A}^{U_i}$ after reading w. Then, by applying the definition of an orbit-DFA, we get that $q \in U'$ since $\delta(q_I, w) = q$, and $U' \cap F = \varnothing$ since U' is a rejecting state of $\mathcal{A}^{U_i}$ (as $w \notin L(\mathcal{A}^{U_i})$). Therefore, $\mathcal{A}^{U_i}$ covers q. Moreover, $|\mathcal{A}^{U_i}| < |\mathcal{A}|$ since $\mathcal{A}^{U_i}$ is a factor of $\mathcal{A}$.

Conversely, let us fix an enumeration $q_1, q_2, \ldots, q_m$ of the rejecting states of $\mathcal{A}$, and suppose that for all $1 \leq i \leq m$ there is an orbit-DFA $\mathcal{A}^{U_i}$ of $\mathcal{A}$ that covers q_i and satisfies $|\mathcal{A}^{U_i}| < |\mathcal{A}|$. Let $(U_{i.j})_{1 \leq j \leq n_i}$ be an enumeration of the subsets in the orbit of U_i that contain the initial state q_I of $\mathcal{A}$. We conclude the proof by showing that $S = \{\mathcal{A}^{U_{i.j}} \mid 1 \leq i \leq m, 1 \leq j \leq n_i\}$ is a decomposition of $\mathcal{A}$. Note that we immediately get $|\mathcal{A}^{U_{i.j}}| = |\mathcal{A}^{U_i}| < |\mathcal{A}|$ for all $1 \leq i \leq m$ and $1 \leq j \leq n_i$. Moreover, Proposition 3 implies $L(\mathcal{A}) \subseteq \bigcap_{\mathcal{A}' \in S} L(\mathcal{A}')$. To complete the proof, we show that $\bigcap_{\mathcal{A}' \in S} L(\mathcal{A}') \subseteq L(\mathcal{A})$. Let $w \in \Sigma^*$ be a word rejected by $\mathcal{A}$. To prove the desired inclusion, we show that there is a DFA $\mathcal{A}' \in S$ that rejects w. Since $w \notin L(\mathcal{A})$, the

[1] Remark that for general DFAs we only get that $q \sim s$ implies $\delta(q, a) \sim \eta(s, a)$ from the determinism. It is the *backward determinism* of the permutation DFAs $\mathcal{A}$ and $\mathcal{B}'$ that gives us the reverse implication.

run of $\mathcal{A}$ on w starting from the initial state ends in a rejecting state q_i, for some $1 \leq i \leq m$. By supposition the orbit-DFA $\mathcal{A}^{U_i}$ covers q_i, hence the orbit of U_i contains a set $U' \subseteq Q$ that contains q_i and no accepting state. Note that there is no guarantee that $\mathcal{A}^{U_i}$ rejects w: while the set $\delta(U_i, w)$ contains q_i, it is not necessarily equal to U', and might contain accepting states. However, as $\mathcal{A}$ is a permutation DFA, we can reverse all of the transitions of $\mathcal{A}$ to get a path labeled by the reverse of w that starts from U' (that contains q_i), and ends in one of the sets $U_{i.j}$ (that contains q_I).[2] Therefore, by reversing this path back to normal, we get that $\delta(U_{i.j}, w) = U'$, hence the orbit-DFA $\mathcal{A}^{U_{i.j}} \in S$ rejects w. Therefore, every word rejected by $\mathcal{A}$ is rejected by an orbit-DFA $\mathcal{A}' \in S$, which shows that $\bigcap_{\mathcal{A}' \in S} L(\mathcal{A}') \subseteq L(\mathcal{A})$. ◀

This powerful lemma allows us to easily determine whether a permutation DFA is composite if we know its orbits. For instance, the DFA $\mathcal{A}$ depicted in Figure 3 is composite since the orbit-DFA $\mathcal{A}^{\{1,2,3\}}$ covers its five rejecting states. Following the proof of Lemma 7, we get that $(\mathcal{A}^{\{1,2,3\}}, \mathcal{A}^{\{1,5,6\}})$ is a decomposition of $\mathcal{A}$, and so is $(\mathcal{A}^{\{1,2,6\}}, \mathcal{A}^{\{1,3,5\}})$.

To conclude, we give an algorithm checking if a rejecting state is covered by an orbit-DFA.

▶ **Lemma 8.** *Given a permutation DFA $\mathcal{A}$ and a rejecting state q, we can determine the existence of an orbit-DFA that covers q in nondeterministic time $\mathcal{O}(k \cdot |\mathcal{A}|^2)$, and in deterministic time $\mathcal{O}(2^k k \cdot |\mathcal{A}|^2)$, where k is the number of rejecting states of $\mathcal{A}$.*

Proof. We can decide in NP whether there exists an orbit-DFA $\mathcal{A}^U$ of $\mathcal{A}$ that covers p: we non-deterministically guess among the set of rejecting states of $\mathcal{A}$ a subset U' containing p. Then, we check in polynomial time that the orbit of U' is smaller than $|\mathcal{A}|$. This property can be checked in time $\mathcal{O}(|\mathcal{A}|^2)$. Since $\mathcal{A}$ is trim, in the orbit of U' there is a set U containing the initial state of $\mathcal{A}$. Moreover, since $\mathcal{A}$ is a permutation DFA, U and U' induce the same orbit. Hence, p is covered by the orbit-DFA $\mathcal{A}^U$. Finally, we can make this algorithm deterministic by searching through the 2^k possible subsets U' of the set of rejecting states of $\mathcal{A}$. ◀

3.2 Proof of Theorem 2

Thanks to the notion of orbit DFAs we are able to prove that a permutation DFA which has a prime number of states with at least one accepting and one rejecting, is prime.

Proof. Let $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$ be a trim permutation DFA with a state space Q of prime size that contains at least one accepting state and one rejecting state. We show that the only orbit of $\mathcal{A}$ smaller than $|Q|$ is the trivial orbit $\{Q\}$. This implies that $\mathcal{A}$ cannot be decomposed into its orbit-DFAs, which proves that $\mathcal{A}$ is prime by Lemma 5.

Let us consider a strict subset $U_1 \neq \varnothing$ of the state space Q, together with its orbit $\mathcal{C}_{U_1} = \{U_1, U_2, \ldots, U_m\}$. We prove that $m \geq |Q|$. First, we show that all the U_i have the same size: since U_i is an element of the orbit of U_1, there exists a word $u_i \in \Sigma^*$ satisfying $\delta(U_1, u_i) = U_i$, and, as every word in Σ^* induces via δ a permutation on the state space, $|U_i| = |\delta(U_1, u_i)| = |U_1|$. Second, for every $q \in Q$, we define the *multiplicity* of q in $\mathcal{C}_{U_1}$ as the number $\lambda(q) \in \mathbb{N}$ of distinct elements of $\mathcal{C}_{U_1}$ containing the state q. We show that all the states q have the same multiplicity: since $\mathcal{A}$ is trim, there exists a word $u_q \in \Sigma^*$ satisfying $\delta(q_I, u_q) = q$, hence u_q induces via δ a bijection between the elements of $\mathcal{C}_{U_1}$ containing q_I and those containing q, and $\lambda(q) = \lambda(\delta(q_I, u_q)) = \lambda(q_I)$. By combining these results, we obtain $m \cdot |U_1| = \Sigma_{i=1}^{m} |U_i| = \Sigma_{q \in Q} \lambda(q) = \lambda(q_I) \cdot |Q|$. Therefore, as $|Q|$ is prime by supposition, either m or $|U_1|$ is divisible by $|Q|$. However, $U_1 \subsetneq Q$, hence $|U_1| < |Q|$, which shows that m is divisible by $|Q|$. In particular, we get $m \geq |Q|$, which concludes the proof. ◀

[2] Remark that, if $\mathcal{A}$ is not a permutation DFA, then some states might not have incoming transitions for every letter. Thus, the reversal of w might not be defined.

4 Decompositions of Commutative Permutation DFAs

We now study *commutative* permutation DFAs: a DFA $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$ is commutative if $\delta(q, uv) = \delta(q, vu)$ for every state q and every pair of words $u, v \in \Sigma^*$. Our main contribution is an NL algorithm for the DECOMP problem for commutative permutation DFAs. Moreover, we show that the complexity goes down to LOGSPACE for alphabets of fixed size.

▶ **Theorem 9.** *The* DECOMP *problem for commutative permutation DFAs is in NL, and in LOGSPACE when the size of the alphabet is fixed.*

The proof of Theorem 9 is based on the notion of *covering word*: a word $w \in \Sigma^*$ covers a rejecting state q of a DFA $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$ if $\delta(q, w) \neq q$, and for every $\lambda \in \mathbb{N}$, the state $\delta(q, w^\lambda)$ is rejecting. We prove two related key results:

- A commutative permutation DFA is composite if and only if each of its rejecting states is covered by a word (Lemma 12).
- We can decide in NL (LOGSPACE when the size of the alphabet is fixed) if a given rejecting state of a DFA is covered by a word (Lemma 13, and Algorithm 2 in appendix)

These results immediately imply Theorem 9. We conclude this section by showing an upper bound on the width and constructing a family of DFAs of polynomial width.

▶ **Theorem 10.** *The width of every composite permutation DFA is smaller than its size. Moreover, for all $m, n \in \mathbb{N}$ such that n is prime, there exists a commutative permutation DFA of size n^m and width $(n-1)^{m-1}$.*

We show that the width of a commutative permutation DFA is bounded by its number of rejecting states (Lemma 12). Then, for each $m, n \in \mathbb{N}$ with n prime, we define a DFA $\mathcal{A}_n^m$ of size n^m that can be decomposed into $(n-1)^{m-1}$ factors (Proposition 14), but not into $(n-1)^{m-1} - 1$ (Proposition 15).

4.1 Proof of Theorem 9

The proof is based on the following key property of commutative permutation DFAs: In a permutation DFA $\mathcal{A}$, every input word acts as a permutation on the set of states, generating disjoint cycles, and if $\mathcal{A}$ is commutative these cycles form an orbit.

▶ **Proposition 11.** *Let $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$ be a commutative permutation DFA. For all $u \in \Sigma^*$, the sets $(\{\delta(q, u^\lambda) \mid \lambda \in \mathbb{N}\})_{q \in Q}$ partition Q and form an orbit of $\mathcal{A}$.*

Proof. Let $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$ be a commutative permutation DFA. Given $u \in \Sigma^*$ and $q \in Q$, the sequence of states $\delta(q, u), \delta(q, u^2), \ldots, \delta(q, u^i)$ visited by applying δ on iterations of u eventually repeats i.e. $\delta(q, u^x) = \delta(q, u^y) = p$ for some $x, y \in \mathbb{N}$ and $p \in Q$. Since $\mathcal{A}$ is a permutation DFA, it is both forward and backward deterministic, thus the set of visited states $\{\delta(q, u^\lambda) \mid \lambda \in \mathbb{N}\}$ is a cycle that contain both p and q. The collection $(\{\delta(q, u^\lambda) \mid \lambda \in \mathbb{N}\})_{q \in Q}$ forms an orbit of $\mathcal{A}$ by commutativity. Formally, for all $u, v \in \Sigma^*$ and every $q \in Q$, we have: $\delta(\{\delta(q, u^\lambda) | \lambda \in \mathbb{N}\}, v) = \{\delta(q, u^\lambda v) | \lambda \in \mathbb{N}\} = \{\delta(q, vu^\lambda) | \lambda \in \mathbb{N}\} = \{\delta(\delta(q, v), u^\lambda) | \lambda \in \mathbb{N}\}$. ◀

We proved with Corollary 6 and Lemma 7 that a permutation DFA is composite if and only if each of its rejecting states is covered by an orbit-DFA. We now reinforce this result for *commutative* permutation DFAs. As stated before, we say that a word $u \in \Sigma^*$ *covers* a rejecting state q of a DFA $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$ if u induces from q a non-trivial cycle composed of rejecting states: $\delta(q, u) \neq q$, and $\delta(q, u^\lambda)$ is rejecting for all $\lambda \in \mathbb{N}$. Note that the collection $(\{\delta(q, u^\lambda) \mid \lambda \in \mathbb{N}\})_{q \in Q}$ forms an orbit of $\mathcal{A}$ by Proposition 11. We show that we can determine if $\mathcal{A}$ is composite by looking for words covering its rejecting states.

▶ **Lemma 12.** *For every $k \in \mathbb{N}$, a commutative permutation DFA $\mathcal{A}$ is k-factor composite if and only if there exist k words that, together, cover all the rejecting states of $\mathcal{A}$.*

Proof. Let $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F\rangle$ be a commutative permutation DFA and $k \in \mathbb{N}$. We start by constructing k factors based on k covering words. Suppose that there exist k words $u_1, u_2, \dots, u_k$ such that every rejecting state $q \in Q \setminus F$ is covered by one of the u_i. Note that all the u_i covering at least one state q do not act as the identity on Q (since $\delta(q, u_i) \neq q$), therefore we suppose, without loss of generality, that none of the u_i acts as the identity on Q. For every $1 \leq i \leq k$, let $U_i = \{\delta(q_I, u_i^\lambda) \mid \lambda \in \mathbb{N}\}$. We show that $(\mathcal{A}^{U_i})_{1\leq i \leq k}$ is a decomposition of $\mathcal{A}$. As none of the u_i acts as the identity on Q, Proposition 11 implies that every $\mathcal{A}^{U_i}$ is smaller than $\mathcal{A}$. Moreover, Proposition 3 implies that $L(\mathcal{A}) \subseteq L(\mathcal{A}^{U_i})$, hence $L(\mathcal{A}) \subseteq \bigcap_{j=1}^k L(\mathcal{A}^{U_j})$. To conclude, we show that $\bigcap_{j=1}^k L(\mathcal{A}^{U_j}) \subseteq L(\mathcal{A})$. Let $u \notin L(\mathcal{A})$. By supposition, there exists $1 \leq i \leq k$ such that u_i covers $\delta(q_I, u)$. As a consequence, the set

$$\begin{aligned}\delta(U_i, u) &= \delta(\{\delta(q_I, u_i^\lambda) \mid \lambda \in \mathbb{N}\}, u) = \{\delta(q_I, u_i^\lambda u) \mid \lambda \in \mathbb{N}\} = \{\delta(q_I, u u_i^\lambda) \mid \lambda \in \mathbb{N}\} \\ &= \{\delta(\delta(q_I, u), u_i^\lambda) \mid \lambda \in \mathbb{N}\}\end{aligned}$$

contains no accepting state of $\mathcal{A}$, hence it is a rejecting state of $\mathcal{A}^{U_i}$. As a consequence, we get $u \notin L(\mathcal{A}^{U_i}) \supseteq \bigcap_{j=1}^k L(\mathcal{A}^{U_j})$, which proves that $\bigcap_{j=1}^k L(\mathcal{A}^{U_j}) \subseteq L(\mathcal{A})$.

We now construct k covering words based on k factors. Suppose that $\mathcal{A}$ has a k-factor decomposition $(\mathcal{B}_i)_{1\leq i\leq k}$. Lemma 4 directly implies that this decomposition can be transformed into a decomposition $(\mathcal{C}_i)_{1\leq i\leq k}$ of $\mathcal{A}$, where $\mathcal{C}_i = \langle \Sigma, S_i, s_I^i, \eta_i, G_i\rangle$ are permutation DFAs. For every $1 \leq i \leq k$, we build a word u_i based on $\mathcal{C}_i$, we prove that every rejecting state of $\mathcal{A}$ is covered by one of these u_i. Consider $1 \leq i \leq k$. Since $\mathcal{C}_i$ is a factor of $\mathcal{A}$, in particular $|\mathcal{C}_i| < |\mathcal{A}|$, hence there exist two input words $v_i, w_i \in \Sigma^*$ such that $\mathcal{A}$ reaches different states on v_i and w_i, but $\mathcal{C}_i$ reaches the same state: $\delta(q_I, v_i) \neq \delta(q_I, w_i)$ but $\eta_i(s_I^i, v_i) = \eta_i(s_I^i, w_i)$. Note that both $\mathcal{A}$ and $\mathcal{C}_i$ are permutation DFAs, hence there exists a power $v_i^{\kappa_i}$ of v_i that induces the identity function on both state spaces Q and S_i. We set $u_i = w_i v_i^{\kappa_i - 1}$, which guarantees that:

$$\begin{aligned}&\delta(q_I, u_i) = \delta(\delta(q_I, w_i), v_i^{\kappa_i-1}) \neq \delta(\delta(q_I, v_i), v_i^{\kappa_i-1}) = \delta(q_I, v_i^{\kappa_i}) = q_I; \\ &\eta_i(s_I^i, u_i) = \eta_i(\eta_i(s_I^i, w_i), v_i^{\kappa_i-1}) = \eta_i(\eta_i(s_I^i, v_i), v_i^{\kappa_i-1}) = \eta_i(s_I^i, v_i^{\kappa_i}) = s_I^i.\end{aligned}$$

In other words, u_i moves the initial state q_I of $\mathcal{A}$, but fixes the initial state s_I^i of $\mathcal{C}_i$.

We now prove that each rejecting state of $\mathcal{A}$ is covered by one of the u_i. Let $q \in Q \setminus F$ be a rejecting state of $\mathcal{A}$. Since $\mathcal{A}$ is trim, there exists a word $u_q \in \Sigma^*$ such that $\delta(q_I, u_q) = q$. Then, as $u_q \notin L(\mathcal{A})$ and $(\mathcal{C}_i)_{1\leq i\leq k}$ is a decomposition of $\mathcal{A}$, there exists $1 \leq i \leq k$ such that $u_q \notin L(\mathcal{C}_i)$. We show that the word u_i covers the rejecting state q: we prove that $\delta(q, u_i) \neq q$, and that $\delta(q, u_i^\lambda)$ is rejecting for every $\lambda \in \mathbb{N}$. First, since $\mathcal{A}$ is a commutative permutation DFA and u_i moves q_I, we get that $\delta(q, u_i) = \delta(q_I, u_q u_i) = \delta(q_I, u_i u_q) \neq \delta(q_I, u_q) = q$. Moreover, for all $\lambda \in \mathbb{N}$, Since $u_q \notin L(\mathcal{C}_i)$ by supposition and u_i fixes s_I^i, the DFA $\mathcal{C}_i$ also rejects the word $u_i^\lambda u_q$. Therefore, as $L(\mathcal{A}) \subseteq L(\mathcal{C}_i)$, we finally get that $\delta(q, u_i^\lambda) = \delta(q_I, u_q u_i^\lambda) = \delta(q_I, u_i^\lambda u_q)$ is a rejecting state of $\mathcal{A}$. ◀

By Lemma 12, to conclude the proof of Theorem 9 we show that we can decide in NL (and in LOGSPACE when the size of the alphabet is fixed) whether a given rejecting state of a DFA is covered by a word (since in the DECOMP problem we can afford to pick a covering word for each state). As we consider commutative permutation DFAs, we can represent a covering word by the number of occurrences of each letter, which are all bounded by $|Q|$.

▶ **Lemma 13.** *Let $\mathcal{A}$ be a commutative permutation DFA and p a rejecting state.*

1. *We can determine the existence of a word covering p in space $\mathcal{O}(|\Sigma| \cdot \log |Q|)$;*
2. *We can determine the existence of a word covering p in NL;*

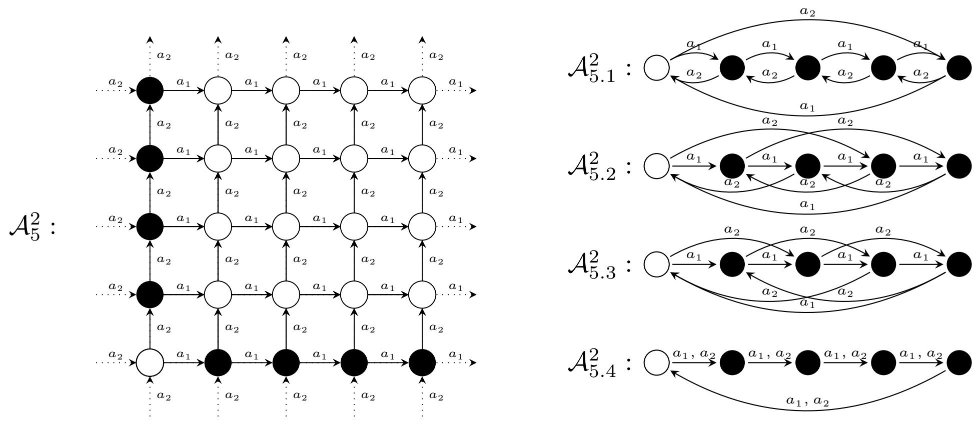

Figure 4 The DFA $\mathcal{A}_5^2$ recognising the language L_5^2, together with its decomposition into four non-trivial orbit-DFAs. Final states are depicted in black.

4.2 Proof of Theorem 10

As a direct consequence of Lemma 12, the width of every commutative permutation DFA $\mathcal{A}$ is bounded by the number of rejecting states of $\mathcal{A}$, hence, it is smaller than $|\mathcal{A}|$. To conclude the proof of Theorem 10, for all $m, n \in \mathbb{N}$ with n prime, we define a DFA $\mathcal{A}_n^m$ of size n^m and width $(n-1)^{m-1}$ on the alphabet $\Sigma = \{a_1, a_2, \ldots, a_m\}$. For all $\ell \in \mathbb{N}$, let $[\ell]$ denote the equivalence class of ℓ modulo n. Let $L_n^m \subseteq \Sigma^*$ be the language composed of the words w such that for at least one letter $a_i \in \Sigma$ the number $\#_{a_i}(w)$ of a_i in w is a multiple of n, and for at least one (other) letter $a_j \in \Sigma$, the number $\#_{a_j}(w)$ of a_j in w is *not* a multiple of n:

$$L_n^m = \{w \in \Sigma^* \mid [\#_{a_i}(w)] = [0] \text{ and } [\#_{a_j}(w)] \neq [0] \text{ for some } 1 \leq i, j \leq m\}.$$

The language L_n^m is recognised by a DFA $\mathcal{A}_n^m$ of size n^m that keeps track of the value modulo n of the number of each a_i already processed. The state space of $\mathcal{A}_n^m$ is the direct product $(\mathbb{Z}/n\mathbb{Z})^m$ of m copies of the cyclic group $\mathbb{Z}/n\mathbb{Z} = ([0], [1], \ldots, [n-1])$; the initial state is $([0], [0], \ldots, [0])$; the final states are the ones containing at least one component equal to $[0]$ and one component distinct from $[0]$; and the transition function increments the i^{th} component when an a_i is read: $\delta(([j_1], [j_2], \ldots, [j_m]), a_i) = ([j_1], [j_2], \ldots, [j_{i-1}], [j_i + 1], [j_{i+1}], \ldots, [j_m])$. Figure 4 illustrates the particular case $n = 5$ and $m = 2$.

To prove that the width of $\mathcal{A}_n^m$ is $(n-1)^{m-1}$, we first show that the $(n-1)^{m-1}$ words $\{a_1 a_2^{\lambda_2} \ldots a_m^{\lambda_m} \mid 1 \leq \lambda_i \leq n-1\}$ cover all the rejecting states, thus by Lemma 12:

▶ **Proposition 14.** *The DFA $\mathcal{A}_n^m$ is $(n-1)^{m-1}$-factor composite.*

Then, we prove that there exist no word that covers two states among the $(n-1)^{m-1}$ rejecting states $\{([1], [k_2], [k_3], \ldots, [k_m]) \mid 1 \leq k_i \leq m-1\}$. Therefore, we need at least $(n-1)^{m-1}$ words to cover all of the states, thus by Lemma 12:

▶ **Proposition 15.** *The DFA $\mathcal{A}_n^m$ is not $((n-1)^{m-1} - 1)$-factor composite.*

5 Bounded Decomposition

We finally study the BOUND-DECOMP problem: Given a DFA $\mathcal{A}$ and an integer $k \in \mathbb{N}$ encoded in unary, can we determine whether $\mathcal{A}$ is decomposable into k factors? For the general setting, we show that the problem is in PSPACE: it can be solved by non-deterministically guessing k factors, and checking that they form a decomposition.

▶ **Theorem 16.** *The* BOUND-DECOMP *problem is in PSPACE.*

For commutative permutation DFAs, we obtain a better algorithm through the use of the results obtained in the previous sections, and we show a matching hardness result.

▶ **Theorem 17.** *The* BOUND-DECOMP *problem for commutative permutation DFAs is NP-complete.*

Both parts of the proof of Theorem 17 are based on Lemma 12: a commutative permutation DFA is k-factor composite if and only if there exist k words covering all of its rejecting states. We prove the two following results:

- Bounded compositionnality is decidable in NP, as it is sufficient to non-deterministically guess a set of k words, and check whether they cover all rejecting states (Lemma 19);
- The NP-hardness is obtained by reducing the HITTING SET problem, a well known NP-complete decision problem. We show that searching for k words that cover the rejecting states of a DFA is as complicated as searching for a hitting set of size k (Lemma 20).

We finally give a LOGSPACE algorithm based on known results for DFAs on unary alphabets [7].

▶ **Theorem 18.** *The* BOUND-DECOMP *problem for unary DFAs is in LOGSPACE.*

Sketch. Recall that a unary DFA $\mathcal{A} = \langle \{a\}, Q, q_I, \delta, F\rangle$ consists of a chain of states leading into one cycle of states. The case where the chain is non-empty is considered in Lemmas 8 and 10 of [7]. We prove that the criteria of these lemmas can be checked in LOGSPACE. If the chain of $\mathcal{A}$ is empty, then $\mathcal{A}$ is actually a commutative permutation DFA. In this case, by Proposition 11 for every word $u = a^i \in \{a\}^*$, the orbit of the set $\{\delta(q_I, u^\lambda) \mid \lambda \in \mathbb{N}\}$ is a partition ρ on Q, and every set in ρ has the same size s_ρ. Both s_ρ and $|\rho|$ divide $|Q|$. For $u = a^i$ where i and $|Q|$ are co-prime, the induced orbit DFA has a single state and thus cannot be a factor of $\mathcal{A}$. Further, if $i_1 < |Q|$ divides $i_2 < |Q|$, then all states covered by a^{i_1} are also covered by a^{i_2}. Hence, w.l.o.g., we only consider words of the form a^i where i is a *maximal divisor* of $|Q|$ in order to generate orbit-DFAs of $\mathcal{A}$ that are candidates for the decomposition. Now, let $p_1^{j_1} \cdot p_2^{j_2} \cdot \ldots \cdot p_m^{j_m} = |Q|$ be the prime factor decomposition of $|Q|$. By Lemma 12 we have that $\mathcal{A}$ is k-factor composite if and only if a selection of k words from the set $\mathcal{W} = \{a^{|Q|/p_i} \mid 1 \leq i \leq m\}$ cover all the rejecting states of $\mathcal{A}$. As $|\mathcal{W}| = m$ is logarithmic in $|Q|$, we can iterate over all sets in $2^{\mathcal{W}}$ of size at most k in LOGSPACE using a binary string indicating the characteristic function. By Lemma 13, checking whether a state $q \in Q$ is covered by the current collection of k words can also be done in LOGSPACE. ◀

5.1 Proof of Theorem 17

By Lemma 12, a commutative permutation DFA $\mathcal{A}$ is k-factor composite if and only if its rejecting states can be covered by k words. As we can suppose that covering words have size linear in $|\mathcal{A}|$ (see proof of Lemma 13), the BOUND-DECOMP problem is decidable in NP: we guess a set of k covering words and check in polynomial time if they cover all rejecting states.

▶ **Lemma 19.** *The* BOUND-DECOMP *problem for commutative permutation DFAs is in NP.*

We show that the problem is NP-hard by a reduction from the HITTING SET problem.

▶ **Lemma 20.** *The* BOUND-DECOMP *problem is NP-hard for commutative permutation DFAs.*

Proof. The proof goes by a reduction from the HITTING SET problem (HIT for short), known to be NP-complete [4]. The HIT problem asks, given a finite set $S = \{1, 2, \ldots, n\} \subseteq \mathbb{N}$, a finite collection of subsets $\mathcal{F} = \{C_1, C_2, \ldots, C_m\} \subseteq 2^S$, and an integer $k \in \mathbb{N}$, whether there

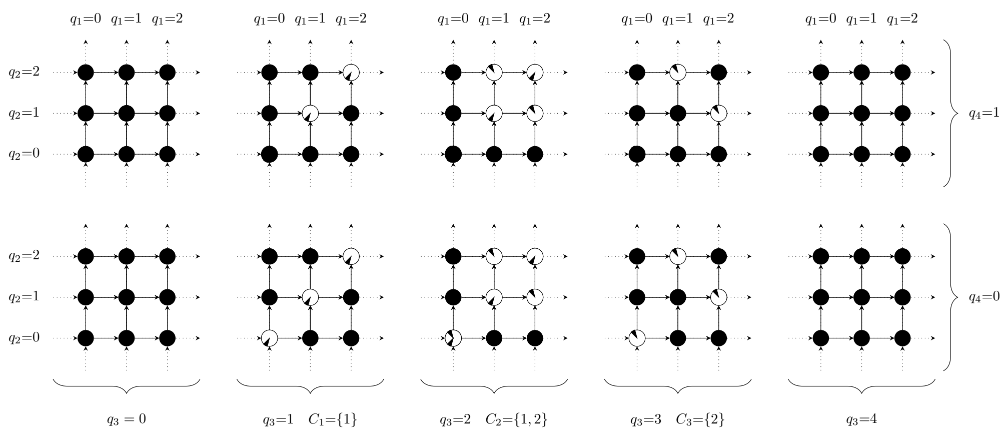

Figure 5 DFA representing the instance of HIT with $S = \{1, 2\}$ and $\mathcal{F} = \{\{1\}, \{1, 2\}, \{2\}\}$ using $\mu = 3$ and $\tau = 5$. Accepting states are filled black while rejecting states are sectored.

is a subset $X \subseteq S$ with $|X| \leq k$ and $X \cap C_i \neq \varnothing$ for all $1 \leq i \leq m$. We describe how to construct a DFA $\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F\rangle$ that is $(k + 1)$-factor composite if and only if the HIT instance $\langle S, \mathcal{F}, k\rangle$ has a solution.

Automaton construction. To be constructed, the automaton $\mathcal{A}$ requires μ, τ defined as the smallest prime numbers that fulfill $n < \mu$ and $m < \tau$ and $2 < \mu < \tau$. By Bertrand's postulate [11], μ and τ have a value polynomial in $m + n$. The state space of $\mathcal{A}$ is defined as $Q = \{0, 1, \dots, \mu - 1\} \times \{0, 1, \dots, \mu - 1\} \times \{0, 1, \dots, \tau - 1\} \times \{0, 1\}$ with $q_I = (0, 0, 0, 0)$ as initial state. Let us define the subset of states $Q_\bot = \{(q_1, q_2, q_3, q_4) \in Q \mid q_4 = 0\}$ to encode instances of HIT and the subset $Q_\top = \{(q_1, q_2, q_3, q_4) \in Q \mid q_4 = 1\}$ which is a copy of $Q_\bot$ with minor changes. The example in Figure 5 gives some intuition on the construction of $\mathcal{A}$. The DFA $\mathcal{A}$ is defined over the alphabet $\Sigma = \{a, b, c, d\}$ with the transition function defined for each state $q = (q_1, q_2, q_3, q_4)$ by $\delta(q, a) = (q_1 + 1 \mod \mu, q_2, q_3, q_4)$, $\delta(q, b) = (q_1, q_2 + 1 \mod \mu, q_3, q_4)$, $\delta(q, c) = (q_1, q_2, q_3 + 1 \mod \tau, q_4)$ and $\delta(q, d) = (q_1, q_2, q_3, q_4 + 1 \mod 2)$. Note that, $\mathcal{A}$ can be seen as a product of four prime finite fields. In particular, for every $q_3 \in \{0, \dots, \tau-1\}$ the subset of states $\{(x, y, q_3, 0) \in Q_\bot \mid 0 \leq x, y \leq \mu-1\}$ can be seen as the direct product of two copies of the field of order μ (a.k.a. $\mathbb{F}_\mu$), thus inheriting the structure of a $\mathbb{F}_\mu$-vector space of origin $(0, 0, q_3, 0)$. We use these τ disjoint vector spaces to represent the collections of $\mathcal{F}$ thanks to the acceptance of states. More precisely, each collection $C_i \in \mathcal{F}$ is encoded through the vector space $\{(x, y, i, 0) \in Q_\bot \mid 1 \leq i \leq m\}$ and each $v \in C_i$ is encoded by the non-acceptance of all states belonging to the line $\{(x, y, i, 0) \in Q_\bot \mid y = vx \mod \mu\}$. In Figure 5, each C_i is presented by an instance of $\mathbb{F}_3 \times \mathbb{F}_3$ and each $v \in \mathcal{C}_i$ is depicted by rejecting states with the same emphasized sector. Since $\tau > m$, there are extra vector spaces for which all states are accepting i.e. $\{(q_1, q_2, q_3, 0) \in Q_\bot \mid q_3 \notin \{1, 2, \dots, m\}\} \subseteq F$. The acceptance of states of $Q_\top$ is defined similarly as for $Q_\bot$ except that the origins of vector spaces are accepting in $Q_\top$ (see Figure 5). Formally, the rejecting states of $\mathcal{A}$ is defined by $\overline{F} = R_\bot \cup R_\top$ where $R_\bot = \{(q_1, q_2, q_3, 0) \in Q_\bot \mid q_2 = vq_1 \mod \mu, 1 \leq q_3 \leq m, v \in C_{q_3}\}$ and $R_\top = \{(q_1, q_2, q_3, 1) \in Q_\top \mid (q_1, q_2, q_3, 0) \in R_\bot, q_1 \neq 0, q_2 \neq 0\}$. All other states are accepting, i.e., we set $F = Q \setminus \overline{F}$. So, the acceptance of the subsets of states $Q_\bot$ and $Q_\top$ only differ by $O \cap Q_\bot \subseteq \overline{F}$ and $O \cap Q_\top \subseteq F$ where $O = \{(0, 0, q_3, q_4) \in Q \mid q_3 \in \{1, \dots, m\}\}$.

The cornerstone which holds the connection between the two problems is the way the rejecting states of O can be covered. In fact, since $Q_\top$ mimics $Q_\bot$ for states in $Q \setminus O$, all rejecting states of $Q \setminus O$ can be covered by the single word $d \in \Sigma$. In addition, most words do not cover any rejecting states of $\mathcal{A}$, as stated by the following claim. Hereafter, we say that a word $w \in \Sigma^*$ is *concise* when it satisfies $\#_\sigma(w) < h_\sigma$ for all $\sigma \in \Sigma$, where $h_\sigma \in \{2, \mu, \tau\}$ is the size of the cycle induced by σ.

$\triangleright$ Claim 21. Let $u \in \Sigma^*$ be a concise word that covers some rejecting state of $\mathcal{A}$:

1. u must belong either in $\{d\}^*$ or in $\{a,b\}^* \setminus (\{a\}^* \cup \{b\}^*)$.
2. u covers some rejecting state of $Q_\top$ iff u covers *all* rejecting states of $Q_\top$ iff $u = d$.
3. u covers $(0,0,i,0) \in O$ iff $u \in \{a,b\}^*$ and $\#_b(u) \equiv v \cdot \#_a(u) \mod \mu$ for some $v \in C_i$.

Proof of Item 1. The statement is a direct consequence of the following:
i. Every concise word u satisfying $\#_c(u) > 0$ covers no rejecting state of $\mathcal{A}$;
ii. Every concise word $u \in \{a\}^* \cup \{b\}^*$ covers no rejecting state of $\mathcal{A}$;
iii. Every concise word u satisfying $\#_a(u) > 0$ and $\#_d(u) > 0$ covers no rejecting state of $\mathcal{A}$;
iv. Every concise word u satisfying $\#_b(u) > 0$ and $\#_d(u) > 0$ covers no rejecting state of $\mathcal{A}$.
In order to prove these four properties, we now fix a state $q = (q_1, q_2, q_3, q_4) \in Q$, and we show that, in each case, iterating a word of the corresponding form starting from q will eventually lead to an accepting state:

(i.) Let u be a concise word satisfying $\#_c(u) > 0$. Since u is concise we have $\#_c(u) < \tau$. Hence, as τ is prime, there exists $\lambda \in \mathbb{N}$ such that $\lambda \cdot \#_c(u) \equiv -q_3 \mod \tau$. Therefore the third component of $\delta(q, u^\lambda)$ is 0, thus it is an accepting state of $\mathcal{A}$.

(ii.) Let $u \in \{a\}^*$ be a concise word (if $u \in \{b\}^*$ instead, the same proof works by swapping the roles of q_1 and q_2). Since u is concise we have $0 < \#_a(u) < \mu$. Hence, as μ is prime there exists $\lambda_1, \lambda_2 \in \mathbb{N}$ satisfying $\lambda_1 \cdot \#_a(u) \equiv -q_1 \mod \mu$ and $\lambda_2 \cdot \#_a(u) \equiv -q_1 + 1 \mod \mu$. Therefore, if $q_2 \neq 0$, we get that $\delta(q, u^{\lambda_1}) = (0, q_2, q_3, q_4)$ is an accepting state of $\mathcal{A}$, and if $q_2 = 0$, we get that $\delta(q, u^{\lambda_2}) = (1, 0, q_3, q_4)$ is an accepting state of $\mathcal{A}$.

(iii.) Let u be a concise word satisfying $\#_a(u) > 0$ and $\#_d(u) > 0$. Since μ is a prime number greater than 2, there exist $\alpha \in \mathbb{N}$ such that $\mu - 2\alpha = 1$, thus $2\alpha \equiv -1 \mod \mu$. Moreover, since u is concise we have $\#_d(u) = 1$ and $\#_a(u) < \mu$. Hence there exists $\beta \in \mathbb{N}$ such that $\beta \cdot \#_a(u) \equiv 1 \mod \mu$. Therefore, if we let $\lambda = 2\alpha\beta q_1 + \mu(1 - p_4)$, we get

$$\#_a(u^\lambda) = 2\alpha \cdot \beta \#_a(u) \cdot q_1 + \mu(1 - p_4) \cdot \#_a(u) \equiv -q_1 \mod \mu;$$
$$\#_d(u^\lambda) = 2\alpha\beta q_1 + \mu \cdot (1 - p_4) \equiv 1 - p_4 \mod 2;$$

As a consequence, the first component of $\delta(q, u^\lambda)$ is 0 and its fourth component is 1, hence it is an accepting state of $\mathcal{A}$.

(iv.) Let u be a concise word satisfying $\#_b(u) > 0$ and $\#_d(u) > 0$. Then we can prove that u does not cover q as in point (3), by swapping the roles of q_1 and q_2. $\triangleleft$

Proof of Item 2. First, remark that d is the only concise word of $\{d\}^*$. By construction of $\mathcal{A}$, we have $(q_1, q_2, q_3, 0) \in F$ if and only if $(q_1, q_2, q_3, 1) \in F$ holds for all $(q_1, q_2, q_3, q_4) \in Q \setminus O$. Thus, for all $(q_1, q_2, q_3, q_4) \in \overline{F} \setminus O$ we have

$$\{\delta((q_1, q_2, q_3, q_4), d^\lambda) \mid \lambda \in \mathbb{N}\} = \{(q_1, q_2, q_3, x) \mid x \in \{0, 1\}\} \subseteq \overline{F}.$$

Hence, if $u = d$ then u covers all rejecting states of of $Q_\top$.

Now suppose that $u \in \Sigma^*$ covers some rejecting state $q = (q_1, q_2, q_3, 1) \in Q_\top$. By Item (1.), either $u \in \{d\}^*$ or $u \in \{a,b\}^* \setminus (\{a\}^* \cup \{b\}^*)$. We show that $u \in \{d\}^*$, by supposing that $\#_a(u) > 0$ and deriving a contradiction. Since μ is prime, there exists $\lambda \in \mathbb{N}$ satisfying $\lambda \cdot \#_a(u) \equiv -q_1 \mod \mu$. Therefore the first component of $\delta(q, u^\lambda)$ is 0 and its fourth component is 1, hence it is accepting, which contradicts the assumption that u covers q. $\triangleleft$

Proof of Item 3. Consider a rejecting state $q = (0, 0, i, 0) \in O$. First, remark that no word in $\{d\}^*$ covers q since $(0, 0, i, 1)$ is accepting. Therefore, by Item (1.), the only concise words that can cover q are the words $u \in \{a, b\}^* \setminus (\{a\}^* \cup \{b\}^*)$. For such a word u, since μ is prime, by Bezout's identity there exists $0 < v < \mu$ satisfying $\#_b(x) \equiv v \cdot \alpha\#_a(x) \mod \mu$, hence

$$\{\delta((0, 0, i, 0), u^\lambda) \mid \lambda \in \mathbb{N}\} = \{(q_1, q_2, i, 0) \in Q \mid q_2 \equiv vq_1 \mod \mu\}.$$

If $v \in C_i$, all the states in this set are rejecting, thus u covers $(0, 0, i, 0)$, but if $v \notin C_i$, all these states except from $(0, 0, i, 0)$ are accepting, thus u does not cover $(0, 0, i, 0)$. ◁

We finally conclude the proof of Lemma 20 by proving that the sets of the initial instance of HIT are hitting if and only if the automaton $\mathcal{A}$ is composite.

If sets are hitting then the automaton is composite. Thanks to Lemma 12, we can show that $\mathcal{A}$ is $(k+1)$-factor composite by finding $(k+1)$ words, namely $w_\top, w_1, w_2, \ldots, w_k$, which all together cover all the rejecting states of $\mathcal{A}$. From the HIT solution $X = \{v_1, v_2, \ldots, v_k\} \subseteq S$, we define $w_j = ab^{v_j}$ for all $1 \leq j \leq k$. We prove now that for all $1 \leq i \leq m$, the rejecting state $(0, 0, i, 0) \in O$ is covered by some w_j. Since $X \cap C_i \neq \varnothing$, there exists $v_j \in X \cap C_i$. Moreover, by definition of w_j, we have $w_j \in \{a, b\}^*$ and $\#_b(w_j) \equiv v_j \cdot \#_a(w_j) \mod \mu$. Therefore, by Claim 21.3, $(0, 0, i, 0)$ is covered by w_j. Finally, we take $w_\top = d$ which covers all rejecting states $\overline{F} \setminus O$ by Claim 21.2.

If the automaton is composite then the sets are hitting. Suppose that $\mathcal{A}$ is $(k+1)$-factor composite. Hence, by Lemma 12, there exists a set W of at most $k+1$ words such that all rejecting states of $\mathcal{A}$ can be covered by some $w \in W$. In addition, we assume that each $w \in W$ is concise: if this is not the case, we can remove the superfluous letter to obtain a concise words that cover the same rejecting states. As a consequence of Claim 21.2, to cover the rejecting states of $Q_\top$, the set W needs the word d, thus W contains at most k words in $\{a, b\}^*$. Moreover, by Claim 21.3, for every $1 \leq i \leq m$, to cover $(0, 0, i, 0) \in O$ the set W needs a word $u_i \in \{a, b\}^*$ satisfying $\#_b(u_i) \equiv v_i \cdot \#_a(u_i) \mod \mu$ for some $v_i \in C_i$. To conclude, we construct $X = \{v_i \mid 1 \leq i \leq m\}$ which is a solution since $|X| \leq k$ due to $W \cap \{d\}^* \neq \varnothing$, and for each $C \in \mathcal{F}$ we have $X \cap C \neq \varnothing$. ◀

6 Discussion

We introduced in this work powerful techniques to treat the DECOMP problem for permutation DFAs. We discuss how they could help solving the related questions that remain open:

- How do the insights obtained by our results translate to the general setting?
- How can we use our techniques to treat other variants of the DECOMP problem?

Solving the general setting. The techniques presented in this paper rely heavily on the group structure of transition monoids of permutation DFAs, thus cannot be used directly in the general setting. They still raise interesting questions: Can we also obtain an FPT algorithm with respect to the number of rejecting states in the general setting? Some known results point that bounding the number of states is not as useful in general as it is for permutation DFAs: while it is known that every permutation DFA with a single rejecting state is prime [9], there exist (non-permutation) DFAs with a single rejecting state that are composite. However, we still have hope to find a way to adapt our techniques: maybe, instead of trying to cover rejecting *states*, we need to cover rejecting *behaviours* of the transition monoid. Another way to improve the complexity in the general setting would be to bound the *width* of DFAs: we defined here a family of DFAs with polynomial width, do there exist families with exponential width? If this is not the case (i.e., every composite DFA has polynomial width), we would immediately obtain a PSPACE algorithm for the general setting.

Variants of the Decomp problem. In this work, we focused on the Bound-Decomp problem, that limits the *number* of factors in the decompositions. Numerous other restrictions can be considered. For instance, the Fragmentation problem bounds the *size* of the factors: Given a DFA $\mathcal{A}$ and $k \in \mathbb{N}$, can we decompose $\mathcal{A}$ into DFAs of size smaller than k? Another interesting restriction is proposed by the Compression problem, that proposes a trade-off between limiting the size and the number of the factors: given a DFA $\mathcal{A}$, can we decompose $\mathcal{A}$ into DFAs $(\mathcal{A}_i)_{1 \leq i \leq k}$ satisfying $\Sigma_{i=1}^{n} |\mathcal{A}_i| < |\mathcal{A}|$? How do these problems compare to the ones we studied? We currently conjecture that the complexity of the Fragmentation problem matches the Decomp problem, while the complexity of the Compression problem matches the Bound-Decomp problem: for commutative permutation DFAs, the complexity seems to spike precisely when we limit the number of factors.

References

1 Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

2 Edmund M. Clarke, David E. Long, and Kenneth L. McMillan. A language for compositional specification and verification of finite state hardware controllers. *Proceedings of the IEEE*, 79(9):1283–1292, 1991. doi:10.1109/5.97298.

3 Willem P. de Roever, Hans Langmaack, and Amir Pnueli, editors. *Compositionality: The Significant Difference, International Symposium, COMPOS'97, Bad Malente, Germany, September 8-12, 1997. Revised Lectures*, volume 1536 of *Lecture Notes in Computer Science*. Springer, 1998. doi:10.1007/3-540-49213-5.

4 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1979.

5 Stephen Gould, Ernest Peltzer, Robert Matthew Barrie, Michael Flanagan, and Darren Williams. Apparatus and method for large hardware finite state machine with embedded equivalence classes, 2007. US Patent 7,180,328.

6 G. H. Hardy. An introduction to the theory of numbers. *Bulletin of the American Mathematical Society*, 35(6):778–818, November 1929. URL: https://projecteuclid.org:443/euclid.bams/1183493592.

7 Ismaël Jecker, Orna Kupferman, and Nicolas Mazzocchi. Unary prime languages. In Javier Esparza and Daniel Král, editors, *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPIcs*, pages 51:1–51:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.MFCS.2020.51.

8 Michal Kunc and Alexander Okhotin. Reversibility of computations in graph-walking automata. In Krishnendu Chatterjee and Jirí Sgall, editors, *Mathematical Foundations of Computer Science 2013 - 38th International Symposium, MFCS 2013, Klosterneuburg, Austria, August 26-30, 2013. Proceedings*, volume 8087 of *Lecture Notes in Computer Science*, pages 595–606. Springer, 2013. doi:10.1007/978-3-642-40313-2_53.

9 Orna Kupferman and Jonathan Mosheiff. Prime languages. *Inf. Comput.*, 240:90–107, 2015. doi:10.1016/j.ic.2014.09.010.

10 Rolf Landauer. Irreversibility and heat generation in the computing process. *IBM J. Res. Dev.*, 5(3):183–191, 1961. doi:10.1147/rd.53.0183.

11 Jaban Meher and M Ram Murty. Ramanujan's proof of Bertrand's postulate. *The American Mathematical Monthly*, 120(7):650–653, 2013. doi:10.4169/amer.math.monthly.120.07.650.

12 Alon Netser. Decomposition of safe languages. Amirim Research Project report from the Hebrew University, 2018.
13 Volnei A. Pedroni. *Finite State Machines in Hardware: Theory and Design (with VHDL and SystemVerilog)*. The MIT Press, 2013.
14 Jean-Eric Pin. On reversible automata. In Imre Simon, editor, *LATIN '92, 1st Latin American Symposium on Theoretical Informatics, São Paulo, Brazil, April 6-10, 1992, Proceedings*, volume 583 of *Lecture Notes in Computer Science*, pages 401–416. Springer, 1992. doi:10.1007/BFb0023844.

A Algorithm: Section 3

Algorithm 1 NP-algorithm for the Decomp problem for permutation DFAs.

```
**Function** isComposite($\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F\rangle$ : permutation DFA)
    **foreach** $p \in Q \setminus F$ **do**
        **guess** $U$ **with** $\{p\} \subseteq U \subseteq Q \setminus F$   /* guess rejecting state $U$ of some orbit-DFA, such that $U$ contains rejecting state $p$ of $\mathcal{A}$ */
        **if not** cover($\mathcal{A}, p, U$) **then return** False
    **return** True

**Function** cover($\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F\rangle$ : permutation DFA, $p \in Q \setminus F$, $U \subseteq Q \setminus F$)
    $\mathcal{C}_U^{\text{old}} = \varnothing$
    $\mathcal{C}_U := \{U\}$
    **while** $\mathcal{C}_U \neq \mathcal{C}_U^{old}$ **and** $|\mathcal{C}_U| < |Q|$ **do**
        $\mathcal{C}_U^{\text{old}} := \mathcal{C}_U$
        $\mathcal{C}_U := \mathcal{C}_U \cup \{\delta(S, \sigma) \mid S \in \mathcal{C}_U, \sigma \in \Sigma\}$
    **if** $|\mathcal{C}_U| \geq |Q|$ **then return** False   /* check that orbit-DFA is factor */
    **foreach** $S \in \mathcal{C}_U$ **do**
        **if** $q_I \in S$ **then return** True   /* check that $U$ is reachable from the inital state of the orbit-DFA */
    **return** False
```

B Algorithm: Section 4

Algorithm 2 Deterministic and non-deterministic version of the algorithm solving the DECOMP problem for commutative permutation DFAs.

```
Function isComposite($\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$ : commutative permutation DFA)
    foreach $p \in Q \setminus F$ do
        cover_found:=False
        foreach $q \in Q \setminus F$ with $q \neq p$ do
            if cover($\mathcal{A}, p, q$) then cover_found:=True   /* covering $p$ with $w_{p,q}$ */
        if not cover_found then return False                  /* no cover found for $p$ */
    return True                                              /* all state $p$ are covered */

Function cover($\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$ : commutative permutation DFA, $p, q \in Q \setminus F$)
    $s := q$
    while $s \neq p$ do                     /* eventually $s = p$ $\mathcal{A}$ is a permuation DFA */
        $s :=$ mimic($p, q, s$)                                  /* thus $s := \delta(s, w_{p,q})$ */
        if $s \in F$ then return False                         /* contradiction of covering */
    return True           /* encountered $p$ again without hitting state in $F$ */

Function mimic($\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$ : commutative permutation DFA,
 $p, q, s \in Q \setminus F$)
    Assumption: $|\Sigma|$ is fixed, let $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_m\}$
    foreach $1 \leq x_1 + \dots + x_{|\Sigma|} \leq |Q|$ do       /* possible since $|\Sigma|$ is fixed */
        if $\delta(p, \sigma_1^{x_1} \sigma_2^{x_2} \dots \sigma_m^{x_m}) = q$ then return $\delta(s, \sigma_1^{x_1} \sigma_2^{x_2} \dots \sigma_m^{x_m})$

Function mimic($\mathcal{A} = \langle \Sigma, Q, q_I, \delta, F \rangle$ : commutative permutation DFA,
 $p, q, s \in Q \setminus F$)
    Assumption: this algorithm is allowed to use non-determinism
    $p' := p, \ell := 0$
    while $p' \neq q$ and $\ell < |Q|$ do
        guess $\sigma \in \Sigma$                /* iteratively contruct $w_{p,q}$ of length $\ell$ */
        $p' := \delta(p', \sigma)$, $s := \delta(s, \sigma)$, $\ell := \ell + 1$
    if $\ell = |Q|$ then return error else return $s$          /* check $q = \delta(p, w_{p,q})$ */
```

C Algorithms: Section 5

Algorithm 3 LOGSPACE-algorithm solving the Bound-Decomp problem for unary DFAs.

```
Function isBoundedComposite($\mathcal{A} = \langle\{a\}, Q, q_I, \delta, F\rangle$: unary DFA, integer $k$)
    if $\mathcal{A}$ is permutation DFA then
        foreach binaryString wordCombination $\in \{0,1\}^{\log|Q|}$ with $\leq k$ ones do
         /* wordCombination represents current set in $2^{\mathcal{W}}$ */
            if testWordCombination($\mathcal{A}$, wordCombination) then return True
                    /* Set of words covering all rejecting states found */
        return False                              /* No covering set found */
    else
        call [7, Algorithm 1]
Function testWordCombination($\mathcal{A} = \langle\{a\}, Q, q_I, \delta, F\rangle$: unary DFA,
 wordCombination: binaryString)
    foreach $q \in Q \setminus F$ do
        if not cover ($\mathcal{A}, q$, wordCombination) then return False    /* Found state
         not covered by current set */
    return True
Function coverBySet($\mathcal{A} = \langle\{a\}, Q, q_I, \delta, F\rangle$: unary DFA, $q \in Q \setminus F$,
 wordCombination: binaryString)
    foreach int $i$ with wordCombination[$i$] ?= 1 do      /* Go through all $\leq k$
     words in the set and test if $q$ is covered */
        compute $p_1 :=$ $i$'th prime divisor of $|Q|$
        if cover($\mathcal{A}, q, \delta(q, a^{|Q|/p_i})$) then return True     /* Function cover from
         Algorithm 2 */
    return False
```

Algorithm 4 NP-algorithm solving the Bound-Decomp problem for commutative permutation DFAs.

```
Function isBoundedComposite(commutative permutation DFA $\mathcal{A}$, integer $k$)
    guess $\mathcal{W} := \{w_i \in \Sigma^{\leq|Q|} \mid i \leq k\}$
    foreach $p \in Q \setminus F$ do
        if not cover($\mathcal{A}, p, \mathcal{W}$) then  return False     /* Some $p$ not covered? */
    return True                                  /* all $p$ are covered */
Function cover(commutative permutation DFA $\mathcal{A}$, state $p$, set of words $\mathcal{W}$)
    foreach $w_i \in \mathcal{W}$ do
        compute $Q_{q,w_i} := \{\delta(q, w_i^\lambda) \mid \lambda \leq |Q|\}$
        if $Q_{q,w_i} \cap F = \varnothing$ then return True
    return False
```