Deciding DFA-Primality Is NP-Hard

Daniel Alexander Spenner
Technische Universität Dortmund, Germany

Abstract

A DFA $\mathcal{A}$ is *composite* if there exist DFAs $\mathcal{A}_1, \ldots, \mathcal{A}_t$ with $\mathcal{L}(\mathcal{A}) = \bigcap_{i=1}^{t} \mathcal{L}(\mathcal{A}_i)$ such that each $\mathcal{A}_i$ has strictly less states than the minimal DFA deciding $\mathcal{L}(\mathcal{A})$. Otherwise, it is *prime*.

Prime-DFA is the problem of deciding primality for a given DFA. It was defined by Kupferman and Mosheiff in 2015 and it was shown to be NL-hard and in ExpSpace.

This paper proves the NP-hardness of Prime-DFA, thereby making the first progress in closing this doubly-exponential gap. It proves the NP-hardness by a reduction from the propositional logic satisfiability problem. The correctness of the reduction relies on an involved characterization of primality for a class of DFAs which contains those that can occur in the reduction.

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

Keywords and phrases Deterministic finite automaton (DFA), Regular languages, Finite languages, Decomposition, Primality, NP-Hardness

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.192

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: https://doi.org/10.48550/arXiv.2605.07031 [11]

1 Introduction

We consider the primality of deterministic finite automata (DFA). Intuitively, the basic question is: Given a DFA $\mathcal{A}$, are there a number of DFAs strictly smaller than $\mathcal{A}$ itself that combined decide the same language as $\mathcal{A}$? More formally, a DFA $\mathcal{A}$ is *composite* if there exist DFAs $\mathcal{A}_1, \ldots, \mathcal{A}_t$ with $\mathcal{L}(\mathcal{A}) = \bigcap_{i=1}^{t} \mathcal{L}(\mathcal{A}_i)$ such that the size of every $\mathcal{A}_i$ is strictly smaller than the index of $\mathcal{A}$. Otherwise, $\mathcal{A}$ is *prime*. Here, the size of $\mathcal{A}$ is the number of its states, while the index of $\mathcal{A}$ is the size of the minimal DFA deciding $\mathcal{L}(\mathcal{A})$. Prime-DFA denotes the problem of deciding primality for a given DFA.

The notion of compositionality of finite automata was introduced in [8], although a similar but more restricted notion was already studied in [4]. With [8, Theorems 2.4 and 2.5], Prime-DFA is NL-hard and in ExpSpace. Up to now, progress in closing this surprisingly large doubly-exponential gap has proven elusive. In this paper, we improve the lower complexity bound of Prime-DFA.

Compositionality in general is an important notion in both practical and theoretical computer science [3, 12]. Regular languages and DFAs are key concepts of theoretical computer science and the question whether a DFA can be decomposed in this fashion seems natural and worth studying. Furthermore, the general idea of automaton decomposition can be motivated by LTL model checking, while a question related to automaton decomposition arises in the field of automaton identification. Both will be briefly discussed below.

Contributions. We prove the NP-hardness of Prime-DFA by reducing the propositional logic satisfiability problem to Prime-DFA.

Given a formula Φ, the reduction yields a DFA $\mathcal{A}_\Phi$ so that every assignment for Φ can be associated with a word of a specific form which is accepted by $\mathcal{A}_\Phi$ if and only if the underlying assignment satisfies Φ. The proof that this is actually a reduction to Prime-DFA relies

53rd International Colloquium on Automata, Languages, and Programming (ICALP 2026).
Editors: Sayan Bhattacharya, Danupon Nanongkai, Michael Benedikt, and Gabriele Puppis;
Article No. 192; pp. 192:1–192:16


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

on an involved characterization of the compositionality of a class of DFAs which contains all DFAs of the form $\mathcal{A}_\Phi$. Namely, we characterize the compositionality of "almost-acyclic" linear safety DFAs (essentially, DFAs whose states are linearly ordered, whose only rejecting state is a rejecting sink, who possess an accepting sink, and whose only cycles are its sinks). This characterization then shows that $\mathcal{A}_\Phi$ accepts a word of this specific form if and only if it is prime. This proves the correctness of the reduction.

Thus, the NP-hardness proof entails the construction of $\mathcal{A}_\Phi$, the characterization of the compositionality of the mentioned DFA class, and the use of this characterization to link the satisfiability of Φ and the compositionality of $\mathcal{A}_\Phi$. Additionally, we further exploit the characterization of the compositionality of "almost-acyclic" linear safety DFAs to prove that the restriction of Prime-DFA to such DFAs is NP-complete, meaning that the established lower bound is tight for this specific class of DFAs.

We attempt to give comprehensive proof sketches for these results in Section 3. The full version of this paper contains an additional section in its appendix [11, Appendix A]. There, we present the full argument with complete formal proofs.

Related Work. As already mentioned, the notion of compositionality was introduced in [8], where the mentioned complexity bounds for Prime-DFA were established. This initial work was followed up by [6, 7, 10]. These works focused on restricted classes of DFAs, namely, unary DFAs, permutation DFAs, and acyclic DFAs and thereby finite languages, respectively. However, the initial complexity bounds for Prime-DFA remained unaltered.

The general idea of decomposing finite automata can be motivated by LTL model checking, where the validity of a specification, given as an LTL-formula, is checked for a system. The automata-based approach entails translating the specification into a finite automaton [13]. Since the LTL model checking problem is PSpace-complete in the size of the LTL-formula [1, Theorem 5.48], it is desirable to decompose the formula into a conjunction of subformulas. This can also be understood as decomposing the finite automaton corresponding to the formula.

A question related to the decomposition of automata arises in the field of automaton identification. The basic task here is, given a set of labeled words, to construct an automaton conforming to this set [5]. An interesting approach is to construct multiple automata instead of one, which can lead to smaller and more intuitive solutions [9].

However, the results obtained in this paper use decompositions with DFAs that are only slightly smaller than the given DFA (most often by just one state). For practical applications, decompositions into much smaller components might be more relevant.

2 Preliminaries

A *deterministic finite automaton* (DFA) is a 5-tuple $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ where Q is a finite set of states, Σ is a finite non-empty alphabet, $q_I \in Q$ is an initial state, $\delta : Q \times \Sigma \to Q$ is a transition function, and $F \subseteq Q$ is a set of accepting states. As usual, we extend δ to words: $\delta : Q \times \Sigma^* \to Q$ with $\delta(q, \varepsilon) = q$ and $\delta(q, \sigma_1 \dots \sigma_n) = \delta(\delta(q, \sigma_1 \dots \sigma_{n-1}), \sigma_n)$.

The *run* of $\mathcal{A}$ on a word $w = \sigma_1 \dots \sigma_n$ starting in state q is the sequence $q_0, \sigma_1, q_1, \dots, \sigma_n, q_n$ with $q_0 = q$ and $q_i = \delta(q_{i-1}, \sigma_i)$ for each $i \in \{1, \dots, n\}$. The *initial run* of $\mathcal{A}$ on w is the run of $\mathcal{A}$ on w starting in q_I. The run of $\mathcal{A}$ on w starting in q is *accepting* if $q_n \in F$. Otherwise, it is *rejecting*. The DFA $\mathcal{A}$ *accepts* w if the initial run of $\mathcal{A}$ on w is accepting. Otherwise, it *rejects* w. The *language* $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$ is the set of words accepted by $\mathcal{A}$. We say that $\mathcal{A}$ *decides* $\mathcal{L}(\mathcal{A})$. A language is *regular* if there exists a DFA deciding it. Since we only consider regular languages, we use the terms language and regular language interchangeably.

The *size* $|\mathcal{A}|$ of $\mathcal{A}$ is the number of states in Q. A DFA $\mathcal{A}$ is *minimal* if $\mathcal{L}(\mathcal{A}) \neq \mathcal{L}(\mathcal{B})$ holds for every DFA $\mathcal{B}$ with $|\mathcal{B}| < |\mathcal{A}|$. It is well known that, for every regular language L, there exists a canonical minimal DFA deciding L. The *index* $\operatorname{ind}(L)$ of L is the size of this canonical minimal DFA. The index of $\mathcal{A}$ is the index of the language decided by $\mathcal{A}$, thus $\operatorname{ind}(\mathcal{A}) = \operatorname{ind}(\mathcal{L}(\mathcal{A}))$. Note that $\mathcal{A}$ is minimal if and only if $|\mathcal{A}| = \operatorname{ind}(\mathcal{A})$.

We borrow a few terms from graph theory. Let $q_0, \sigma_1, q_1, \ldots, \sigma_n, q_n$ be the run of $\mathcal{A}$ on $w = \sigma_1 \ldots \sigma_n$ starting in q_0. Then $q_0, \ldots, q_n$ is a *path* in $\mathcal{A}$ from q_0 to q_n. Thus, for two states q, q', there exists a path from q to q' in $\mathcal{A}$ if and only if there exists a $w \in \Sigma^*$ with $\delta(q, w) = q'$. State q' is *reachable from* q if there exists a path from q to q'. Otherwise, q' is *unreachable from* q. We say that q' is *reachable* if it is reachable from q_I. Otherwise, it is *unreachable*. A *cycle* in $\mathcal{A}$ is a path $q_0, \ldots, q_n$ in $\mathcal{A}$ where $q_0 = q_n$ and $n \in \mathbb{N}_{\geq 1}$. We call a state q a *sink* if $\delta(q, \sigma) = q$ for all σ, that is, if all out-q-transitions are self-loops.

We introduce the notions of compositionality and primality of DFAs and languages, following the definitions in [8]:

▶ **Definition 2.1.** For $k \in \mathbb{N}_{\geq 1}$, a DFA $\mathcal{A}$ is *k-decomposable* if there exist DFAs $\mathcal{A}_1, \ldots, \mathcal{A}_t$ with $\mathcal{L}(\mathcal{A}) = \bigcap_{i=1}^{t} \mathcal{L}(\mathcal{A}_i)$ and $|\mathcal{A}_i| \leq k$ for each $i \in \{1, \ldots, t\}$, where $t \in \mathbb{N}_{\geq 1}$. We call such DFAs $\mathcal{A}_1, \ldots, \mathcal{A}_t$ a *k-decomposition* of $\mathcal{A}$. We call $\mathcal{A}$ *composite* if $\mathcal{A}$ is k-decomposable for a $k < \operatorname{ind}(\mathcal{A})$, that is, if it is $(\operatorname{ind}(\mathcal{A}) - 1)$-decomposable. Otherwise, we call $\mathcal{A}$ *prime*. ⌟

When analyzing the compositionality of a given DFA $\mathcal{A}$, it is sufficient to consider minimal DFAs $\mathcal{B}$ strictly smaller than the minimal DFA of $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$. Thus, we define $\alpha(\mathcal{A}) = \{\mathcal{B} \mid \mathcal{B} \text{ is a minimal DFA with } \operatorname{ind}(\mathcal{B}) < \operatorname{ind}(\mathcal{A}) \text{ and } \mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})\}$. Obviously, the DFA $\mathcal{A}$ is composite if and only if $\mathcal{L}(\mathcal{A}) = \bigcap_{\mathcal{B} \in \alpha(\mathcal{A})} \mathcal{L}(\mathcal{B})$. We call a word $w \in (\bigcap_{\mathcal{B} \in \alpha(\mathcal{A})} \mathcal{L}(\mathcal{B})) \setminus \mathcal{L}(\mathcal{A})$ a *primality witness* of $\mathcal{A}$. Clearly, a DFA $\mathcal{A}$ is composite if and only if it has no primality witness.

We say that a regular language is prime (composite) if its minimal automaton is prime (composite).

We denote the problem of deciding primality for a given DFA by Prime-DFA. Prime-DFA is NL-hard and in ExpSpace [8, Theorems 2.4 and 2.5].

3 NP-Hardness of Prime-DFA

We improve the lower complexity bound for Prime-DFA by proving:

▶ **Theorem 3.1.** *The problem Prime-DFA is NP-hard.* ⌟

As mentioned, this is the first improvement of a complexity bound for Prime-DFA since [8] introduced Prime-DFA and proved that it is NL-hard and in ExpSpace.

Proof Idea

We prove the NP-hardness of Prime-DFA (Theorem 3.1) by a reduction from CNFSAT. Given a CNF-formula Φ, the reduction yields a DFA $\mathcal{A}_\Phi$ ([11, Definition A.1] (full version), Figure 1) with the goal that Φ is satisfiable if and only if $\mathcal{A}_\Phi$ is prime (Lemma 3.2).

In the following, we call DFAs of the form $\mathcal{A}_\Phi$ CNF-DFAs. The correctness of the reduction relies on a (somewhat technical) characterization of the compositionality of CNF-DFAs. Simplified, for a given CNF-DFA $\mathcal{A}_\Phi$, only the words on which $\mathcal{A}_\Phi$ visits each of its non-sinks before entering its rejecting sink are relevant for its compositionality. We call these the max-visiting words of $\mathcal{A}_\Phi$. We say that $\mathcal{A}_\Phi$ has the max-pumping-property (mp-prop)

if every max-visiting word of $\mathcal{A}_\Phi$ has a factorization xyz so that $\mathcal{A}_\Phi$ rejects xy^lz for every $l \in \mathbb{N}$ (Definition 3.7). It turns out that a CNF-DFA is composite if and only if it has the mp-prop (Lemma 3.5 and Theorem 3.9).

The CNF-DFAs are designed so that the structure of the max-visiting words and their possible factorizations are strongly restricted (Lemma 3.11). This has two major benefits. Firstly, we will have a lot of control over the behavior of $\mathcal{A}_\Phi$ on each individual max-visiting word. And secondly, we will have a whole set of max-visiting words to reason about, which could potentially break the mp-prop.

More precisely, the max-visiting words have the form udc^μ with $u \in \{0,1\}^r$, where r is the number of Φ's variables, c and d are dummy letters, and μ is a value dependent on $|\Phi|$ (Lemma 3.11 (i)). The prefix u is interpreted as a variable assignment for Φ. Due to the structure of $\mathcal{A}_\Phi$, the only factorization that might witness the mp-prop for such a max-visiting word udc^μ is $x = \varepsilon, y = u, z = dc^\mu$ (Lemma 3.11 (ii)). Thus, exactly the pumpings of the form u^ldc^μ are decisive for the mp-prop and thereby for the compositionality of $\mathcal{A}_\Phi$.

Intuitively, $\mathcal{A}_\Phi$ uses the repetitions of the assignment u in u^l to check whether u satisfies each clause (Lemma 3.12). Indeed, let s be the number of clauses of Φ, then the crucial pumping is $u^{s+2}dc^\mu$: To check the s clauses, s repetitions of u are needed; two further repetitions of u are needed for technical reasons. If u does not satisfy Φ, then $\mathcal{A}_\Phi$ detects an unsatisfied clause while reading u^{s+2} and immediately enters its rejecting sink. Otherwise, that is, if u satisfies Φ, then $\mathcal{A}_\Phi$ detects no unsatisfied clause while reading u^{s+2} and enters the accepting sink when beginning the final u-repetition.

To summarize, if Φ is unsatisfiable, then $\mathcal{A}_\Phi$ has the mp-prop and is therefore composite. Otherwise, that is, if Φ is satisfiable, then $\mathcal{A}_\Phi$ does not have the mp-prop and is therefore prime, which is witnessed by the pumping $u^{s+2}dc^\mu$ of the max-visiting word udc^μ, where $u \in \{0,1\}^r$ encodes a satisfying assignment for Φ.

We outline the reduction in Section 3.1 and prove its correctness in Section 3.2. To do so, we characterize the compositionality of the relevant DFAs in Section 3.2.1 and exploit this characterization to prove the correctness in Section 3.2.2. To conclude, we prove, in Section 3.3, that the restriction of Prime-DFA to the DFAs relevant for the reduction is NP-complete, meaning that the established lower bound is tight for this specific class of DFAs. The formal proofs of the results of Section 3 can be found in the full version [11, Appendix A].

3.1 Construction of $\mathcal{A}_\Phi$

We denote the propositional logic satisfiability problem for formulas in conjunctive normal form (CNF) by CNFSAT. It is well known that CNFSAT is NP-complete [2]. From here on, we call propositional logic formulas in CNF simply CNF-formulas.

As mentioned, we prove Theorem 3.1 by a reduction from CNFSAT. For a given CNF-formula Φ, the reduction yields a CNF-DFA $\mathcal{A}_\Phi$ so that Φ is satisfiable if and only if $\mathcal{A}_\Phi$ is prime.

Before we can describe the reduction, we have to fix some notation. For the remainder of Section 3, let $X = \{x_1, \dots, x_r\}$ be a set of $r \in \mathbb{N}_{\geq 1}$ numbered variables, and let Φ be a CNF-formula over X. W.l.o.g. we assume that Φ has at least one clause and that every variable appears at most once in every clause.

To simplify the construction of $\mathcal{A}_\Phi$, we fix a special notation for Φ. Namely, we write $\Phi = (e_1^1 \vee \cdots \vee e_r^1) \wedge \cdots \wedge (e_1^s \vee \cdots \vee e_r^s)$, where $s \in \mathbb{N}_{\geq 1}$ is the number of clauses of Φ, and $e_i^k \in \{\neg x_i, x_i, \bot\}$ is the i-th *element* in the k-th clause where $k \in \{1, \dots, s\}, i \in \{1, \dots, r\}$. In other words, in our notation, every clause has exactly r elements and the i-th element of every clause is either a literal of the i-th variable x_i or $\bot$. Here, $\bot$ is a special symbol that always evaluates to *False*. It is easy to see that Φ can be written in this way.

Towards the definition of $\mathcal{A}_\Phi$, we introduce some additional notation. Let $\Sigma = \{0, 1, c, d\}$. We call a string $u \in \{0,1\}^r$ an *assignment string*. As usual, an assignment over X is a function $\gamma : X \to \{0,1\}$. An assignment γ over X induces an assignment string $u_\gamma \in \{0,1\}^r$ in the following way: $u_\gamma = \gamma(x_1) \dots \gamma(x_r)$. Conversely, an assignment string $u = \sigma_1 \dots \sigma_r \in \{0,1\}^r$ induces an assignment γ_u over X in the following way: $\gamma_u(x_i) = \sigma_i$ for all $i \in \{1, \dots, r\}$.

Now we can consider the reduction, that is, the construction of the CNF-DFA $\mathcal{A}_\Phi = (Q_\Phi, \Sigma, p_0, \delta_\Phi, F_\Phi)$ out of the given CNF-formula Φ. The CNF-DFA $\mathcal{A}_\Phi$ is outlined in Figure 1. It is formally defined in the full version [11, Definition A.1]. The key property of $\mathcal{A}_\Phi$ that we want to prove is formalized as:

▶ **Lemma 3.2.** *The CNF-formula Φ is satisfiable if and only if the CNF-DFA $\mathcal{A}_\Phi$ is prime.* ⌟

As mentioned above, the compositionality of $\mathcal{A}_\Phi$ depends solely on the max-visiting words of $\mathcal{A}_\Phi$. We will discuss this in Section 3.2.1. For now, we will focus on the design of $\mathcal{A}_\Phi$ and give an intuition for it.

The CNF-DFA $\mathcal{A}_\Phi$ consists of two devices, the *assignment device* and the *formula device*. The assignment device consists of the states $p_0, \dots, p_r$ and p_c^0. To fully traverse the assignment device, that is, to start in p_0 and reach p_c^0, a string ud with $u \in \{0,1\}^r$ has to be read. Therefore, the assignment device enforces that every max-visiting word is prefixed by an assignment string $u \in \{0,1\}^r$ (followed by a single d). Subsequently, the assignment string u is interpreted as the assignment γ_u for Φ which it induces.

The formula device consists of s *clause rows*. For $k \in \{1, \dots, s\}$, the k-th clause row encodes the k-th clause of Φ, that is, $(e_1^k \vee \dots \vee e_r^k)$. Thus, the formula device in its entirety encodes all clauses and thereby Φ. The k-th clause row consists of the states $p_1^k, \hat{p}_1^k, \dots, p_r^k, \hat{p}_r^k$ and p_c^k. Among these, $\hat{p}_r^k$ is the positive target state of the k-th clause row, which is also the designated entry point for the subsequent clause row. And further, p_r^k is the negative target state of the k-th clause row, which prevents $\mathcal{A}_\Phi$ from entering the subsequent clause row.

We point out that p_r is part of the assignment device enforcing the prefix ud, but also functions similarly to a positive target state for letters 0 and 1. In particular, it is the designated entry point for the first clause row. Therefore, we denote the state p_r also by $\hat{p}_r^0$.

The formula device is designed to enforce two assertions.

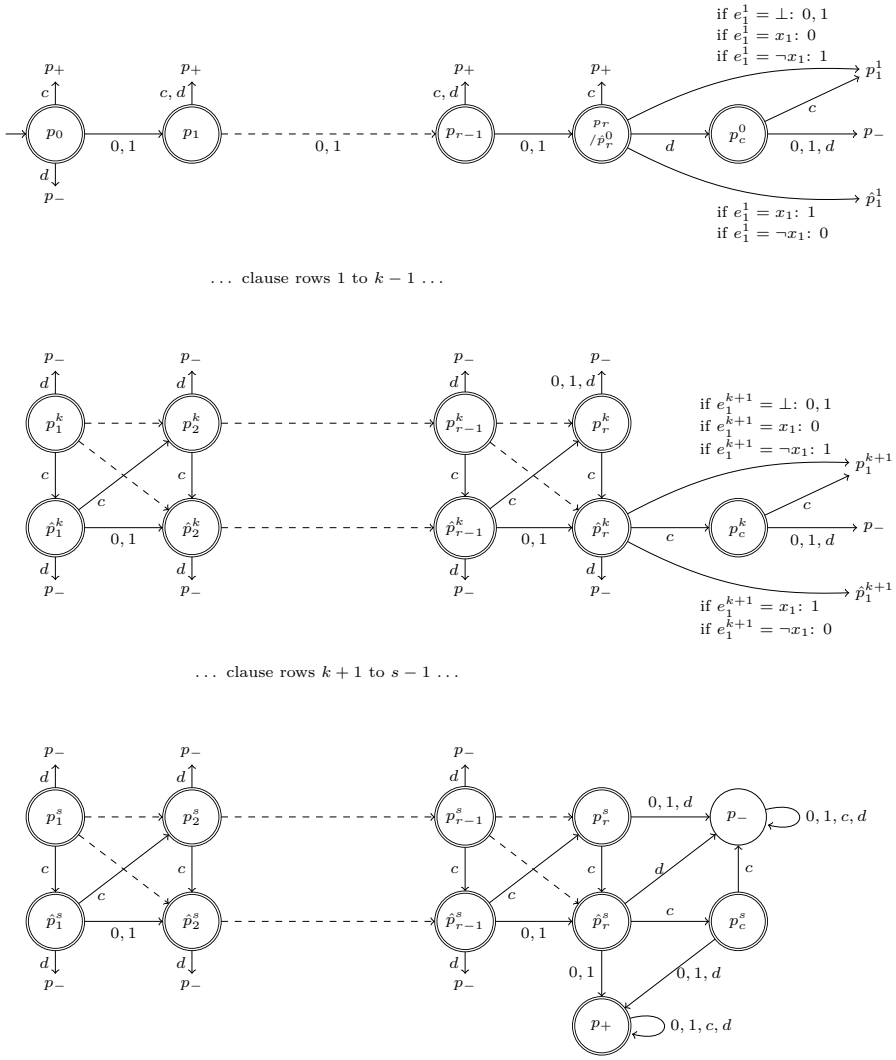
Firstly, every max-visiting word of $\mathcal{A}_\Phi$ is of the form udc^μ, where $u \in \{0,1\}^r$ is an assignment string and μ depends on $|\Phi|$. To be precise, this first assertion is enforced by the two devices together: The assignment device enforces the prefix ud and the formula device the suffix c^μ. The first assertion is formalized further below in Lemma 3.11 (i).

Secondly, if γ_u satisfies the k-th clause, then u induces a run from the positive target state $\hat{p}_r^{k-1}$ of the $(k-1)$-th clause to the positive target state $\hat{p}_r^k$ of the k-th clause. Else, u induces a run from the positive target state $\hat{p}_r^{k-1}$ of the $(k-1)$-th clause to the negative target state p_r^k of the k-th clause. The second assertion is formalized in:

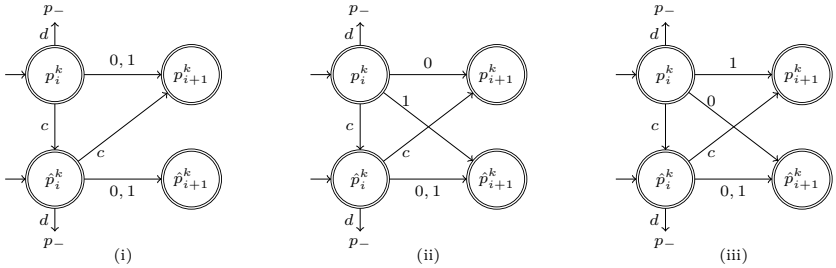
▶ **Lemma 3.3.** *Consider an assignment string $u \in \{0,1\}^r$. Let $k \in \{1, \dots, s\}$. If γ_u satisfies the k-th clause of Φ, then $\delta_\Phi(\hat{p}_r^{k-1}, u) = \hat{p}_r^k$ holds. Else, $\delta_\Phi(\hat{p}_r^{k-1}, u) = p_r^k$ holds.* ⌟

Note that reading 0 or 1 in any negative target state p_r^k leads to p_-, while reading 0 or 1 in the final positive target state $\hat{p}_r^s$ leads to p_+. Together with the second assertion enforced by the formula device, this implies that, for every $l \geq s + 2$, a word of the form $u^l dc^\mu$ is accepted by $\mathcal{A}_\Phi$ if and only if γ_u satisfies Φ. We expand on this argument in our discussion of Lemma 3.12 in Section 3.2.2. Also in Section 3.2.2, we will see that $\mathcal{A}_\Phi$ is prime if and only if $\mathcal{A}_\Phi$ accepts some word of the form $u^l dc^\mu$. Thus, $\mathcal{A}_\Phi$ is prime if and only if Φ is satisfiable.

We have outlined the CNF-DFA $\mathcal{A}_\Phi$ yielded by the reduction and given an intuition for its design. Next, we prove the correctness of the reduction.



(a) CNF-DFA $\mathcal{A}_\Phi$ for CNF-formula $\Phi = (e_1^1 \vee \cdots \vee e_r^1) \wedge \cdots \wedge (e_1^s \vee \cdots \vee e_r^s)$ over set $X = \{x_1, \ldots, x_r\}$ of $r \in \mathbb{N}_{\geq 1}$ numbered variables. Depicted are the assignment device, the k-th clause row with $k \in \{1, \ldots, s-1\}$, and the s-th and last clause row. In the k-th clause row, the out-transitions of the states $p_1^k, \ldots, p_{r-1}^k$ for the letters $0, 1$ are omitted. For such a state p_i^k, these 0/1-out-transitions depend on the element e_{i+1}^k and can lead to p_{i+1}^k or $\hat{p}_{i+1}^k$, as suggested by the dashed lines. The three possible configurations (for $e_{i+1}^k \in \{\bot, \neg x_i, x_i\}$) are depicted in Figure 1b. The same omissions are made in the s-th clause row.



(b) Out-transitions of the states $p_i^k, \hat{p}_i^k$ of the CNF-DFA $\mathcal{A}_\Phi$ depicted in Figure 1a, where $k \in \{1, \ldots, s\}, i \in \{1, \ldots, r-1\}$. The out-transitions of p_i^k depend on e_{i+1}^k. (i): Out-transitions if $e_{i+1}^k = \bot$. (ii): Out-transitions if $e_{i+1}^k = x_i$. (iii): Out-transitions if $e_{i+1}^k = \neg x_i$.

Figure 1 CNF-DFA $\mathcal{A}_\Phi$.

3.2 Correctness of the Reduction

We complete the proof of Theorem 3.1 by proving the correctness of the reduction. That is, we prove Lemma 3.2, that Φ is satisfiable if and only if $\mathcal{A}_\Phi$ is prime. We accomplish this in two steps. In Section 3.2.1, we characterize the compositionality of CNF-DFAs. Then in Section 3.2.2, we exploit this characterization to prove Lemma 3.2 and thereby Theorem 3.1.

3.2.1 Compositionality of Minimal Linear Safety ADFA$^+$s

We first define minimal linear safety ADFA$^+$s and then characterize their compositionality.

Definitions: ADFA$^+$s, Linear DFAs, Safety DFAs

Following convention, a DFA $\mathcal{A}$ is *acyclic* (ADFA) if there are no cycles in $\mathcal{A}$ besides those of the rejecting sinks. Building on that, a DFA $\mathcal{A}$ is an *ADFA-plus* (ADFA$^+$) if $\mathcal{A}$ possesses both a rejecting and an accepting sink and there are no cycles in $\mathcal{A}$ besides those of the sinks. Thus, an ADFA$^+$ is "almost" an ADFA but additionally has an accepting sink.

We call a DFA $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ *linear* if, for every $q, q' \in Q$ with $q \neq q'$, exactly one of the following holds: (i) q' is reachable from q; (ii) q is reachable from q'; (iii) q and q' are unreachable from each other, and q or q' or both are sinks. Obviously, every minimal linear DFA has at least one and at most two sinks.

For a minimal ADFA$^+$ $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$, we denote by $\mathrm{len}(\mathcal{A})$ the length of the longest word $w \in \Sigma^*$ such that words $v, v' \in \Sigma^*$ with $\delta(q_I, wv) \in F$ and $\delta(q_I, wv') \notin F$ exist. Thus, $\mathrm{len}(\mathcal{A})$ is the length of the longest word on which $\mathcal{A}$ does not enter a sink. Because $\mathcal{A}$ is minimal and has two sinks, such a word always exists. The following holds:

▶ **Lemma 3.4.** *Consider a minimal ADFA$^+$ $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$. Then the following assertions hold:*

(i) $|\mathcal{A}| \geq \mathit{len}(\mathcal{A}) + 3$.

(ii) *$\mathcal{A}$ is linear if and only if* $|\mathcal{A}| = \mathit{len}(\mathcal{A}) + 3$. ⌟

Finally, we introduce a type of DFA already inspected in [8]. A regular language $L \subseteq \Sigma^*$ is a *safety language* if, for every $w \in \Sigma^*$, it holds that $w \notin L$ implies $wy \notin L$ for every $y \in \Sigma^*$. A DFA $\mathcal{A}$ is a *safety DFA* if $\mathcal{L}(\mathcal{A})$ is a safety language. Clearly, every non-trivial minimal safety DFA has exactly one rejecting state, and this state is a sink.

The general form of minimal linear safety ADFA$^+$s is outlined in the full version [11, Figure 3]. The following observation can be easily verified by inspecting [11, Definition A.1] and Figure 1.

▶ **Lemma 3.5.** *The CNF-DFA $\mathcal{A}_\Phi$ is a minimal linear safety ADFA$^+$.* ⌟

Clearly, $\mathcal{A}_\Phi$ is a safety DFA since the rejecting sink p_- is the only rejecting state of $\mathcal{A}_\Phi$. Further, to see that $\mathcal{A}_\Phi$ is additionally a linear ADFA$^+$, recall the following observation from Section 3.1: To fully traverse $\mathcal{A}_\Phi$, a word of the form udc^μ for $u \in \{0,1\}^r$ and a suitable μ has to be read. On such a word, the states of the assignment device are traversed in the order $p_0, \ldots, p_r, p_c^0$, and the states of the formula device are traversed in the order $p_1^1, \hat{p}_1^1, \ldots, p_r^1, \hat{p}_r^1, p_c^1, \ldots, p_1^s, \hat{p}_1^s, \ldots, p_r^s, \hat{p}_r^s, p_c^s$. Intuitively, this order describes the "forward direction" in $\mathcal{A}_\Phi$, and there are no "backward transitions" in $\mathcal{A}_\Phi$. Thus, $\mathcal{A}_\Phi$ is acyclic (disregarding p_+) and its states are linearly ordered, meaning that $\mathcal{A}_\Phi$ is a linear ADFA$^+$. Lastly, regarding the minimality, note that two states of $\mathcal{A}_\Phi$ are differentiated by the suffixes of words udc^μ needed to fully traverse the remainder of $\mathcal{A}_\Phi$ and enter the rejecting sink

when starting in the respective states. For example, two states from the formula device are differentiated by the number of c's needed to reach the rejecting sink. Thus, $\mathcal{A}_\Phi$ is minimal. In total, $\mathcal{A}_\Phi$ is a minimal linear safety ADFA$^+$.

Thanks to Lemma 3.5, it is sufficient to characterize the compositionality of minimal linear safety ADFA$^+$s in order to characterize the compositionality of CNF-DFAs.

Characterization of the Compositionality of Minimal Linear Safety ADFA$^+$s

Now we characterize the compositionality of minimal linear safety ADFA$^+$s. With Example 3.10 at the end of this section, we illustrate the main ideas of the characterization by considering, by way of example, two minimal linear safety ADFA$^+$s, one composite and one prime.

We point out that when we consider the compositionality of a restricted class of DFAs, the (smaller) DFAs used in the decompositions are not restricted. In particular, the DFAs we use in the characterization of the compositionality of minimal linear safety ADFA$^+$s need not be minimal linear safety ADFA$^+$s themselves.

The compositionality of a DFA $\mathcal{A}$ boils down to the question whether, for each word $w \notin \mathcal{L}(\mathcal{A})$, there exists some DFA $\mathcal{B}$ with $w \notin \mathcal{L}(\mathcal{B})$ and $\mathcal{B} \in \alpha(\mathcal{A})$, the latter meaning $|\mathcal{B}| < \text{ind}(\mathcal{A})$ and $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$. Indeed, if the answer is yes, the collection of these DFAs yields a decomposition of $\mathcal{A}$ (thanks to $|\mathcal{B}| < \text{ind}(\mathcal{A})$, the collection is finite). If, on the other hand, for some $w \notin \mathcal{L}(\mathcal{A})$, there is no such $\mathcal{B}$, then w is a primality witness of $\mathcal{A}$.

For a minimal linear safety ADFA$^+$ $\mathcal{A}$, the situation is simplified by the fact that there is only one rejecting state, namely, the rejecting sink. For words w on which $\mathcal{A}$ enters the rejecting sink without visiting every non-sink, building an appropriate DFA rejecting w is relatively simple. Essentially, we can just remove one of the skipped-over states q of $\mathcal{A}$ so that the constructed DFA accepts all words on which $\mathcal{A}$ visits q and acts as $\mathcal{A}$ on all other words, including w. Details are given in the full version [11, Definition A.3, Figure 4a and Lemma A.4]. Thus, in the following, we only need to consider the words on which $\mathcal{A}$ enters its rejecting sink after visiting every non-sink. Formally, we define the set of these words as:

▶ **Definition 3.6.** Consider a minimal linear safety ADFA$^+$ $\mathcal{A}$ over an alphabet Σ. Let $\lambda = \text{len}(\mathcal{A})$. Then we define the set $\mathcal{K}(\mathcal{A})$ of the *max-visiting words of* $\mathcal{A}$ as follows:

$$\mathcal{K}(\mathcal{A}) = \{v\sigma \mid v \in \Sigma^\lambda \wedge \sigma \in \Sigma \wedge v \in \mathcal{L}(\mathcal{A}) \wedge v\sigma \notin \mathcal{L}(\mathcal{A})\}.$$ ⌟

On a max-visiting word $w = v\sigma \in \mathcal{K}(\mathcal{A})$, the ADFA$^+$ $\mathcal{A}$ does not skip over any non-sink. Thus, the idea of removing a skipped-over state to construct an appropriate DFA rejecting w is not applicable. Indeed, the situation for the max-visiting words is more involved. It turns out that every relevant $\mathcal{B} \in \alpha(\mathcal{A})$ is a safety DFA with an accepting sink. Thus, every such $\mathcal{B}$ possesses one accepting sink, one rejecting sink, and at most λ accepting non-sinks. Here, we omit the details and refer to the full version [11, Lemmas A.10 and A.11]. Since $\mathcal{B}$ has at most λ non-sinks and the prefix v of w has length λ, the initial run of $\mathcal{B}$ on v has to visit a non-sink twice. Therefore, for some x, y, z with $w = xyz$ and $|y|, |z| > 0$, the DFA $\mathcal{B}$ reaches the same state after reading x, xy and every further xy^l. This observation gives rise to the definition of the max-pumping-property:

▶ **Definition 3.7.** Consider a minimal linear safety ADFA$^+$ $\mathcal{A}$ over an alphabet Σ. We say that $\mathcal{A}$ has the *max-pumping-property* (*mp-prop*) if, for every max-visiting word $w \in \mathcal{K}(\mathcal{A})$, there are $x, y, z \in \Sigma^*$ with $w = xyz$ and $|y|, |z| > 0$ so that $xy^l z \notin \mathcal{L}(\mathcal{A})$ holds for every $l \in \mathbb{N}$. ⌟

Before we state the desired characterization of the compositionality of minimal linear safety ADFA$^+$s in terms of the mp-prop, we introduce an alternative formulation of the mp-prop, for which we need some additional notation. Consider a word $w = \sigma_1 \dots \sigma_n \in \Sigma^n$. For $i, j \in \{1, \dots, n+1\}, i < j, l \in \mathbb{N}$, we define $w[i, j; l] = \sigma_1 \dots \sigma_{i-1}(\sigma_i \dots \sigma_{j-1})^l \sigma_j \dots \sigma_n$. We refer to $w[i, j; l]$ as a *pumping of* w. The following lemma provides an alternative formulation of the mp-prop by translating it into the $w[i, j; l]$-notation.

▶ **Lemma 3.8.** *Consider a minimal linear safety ADFA$^+$ $\mathcal{A}$. Let $\lambda = len(\mathcal{A})$. Then $\mathcal{A}$ has the mp-prop if and only if, for every max-visiting word $w \in \mathcal{K}(\mathcal{A})$, there are $i, j \in \{1, \dots, \lambda + 1\}$ with $i < j$ so that $w[i, j; l] \notin \mathcal{L}(\mathcal{A})$ holds for every $l \in \mathbb{N}$.* ⌟

For a minimal linear safety ADFA$^+$ $\mathcal{A}$, a max-visiting word $w \in \mathcal{K}(\mathcal{A})$ and indices i, j, we say that the *mp-prop-condition holds for* w, i, j if $w[i, j; l] \notin \mathcal{L}(\mathcal{A})$ holds for every $l \in \mathbb{N}$. Note that $w[i, j; l] \notin \mathcal{L}(\mathcal{A})$ implies $w[i, j; l]w' \notin \mathcal{L}(\mathcal{A})$ for all words w' because $\mathcal{A}$ is a safety DFA.

With the mp-prop defined, we can characterize the compositionality of minimal linear safety ADFA$^+$s as follows:

▶ **Theorem 3.9.** *A minimal linear safety ADFA$^+$ is prime if and only if it does not have the mp-prop.* ⌟

We now set out to give an intuition for this result.

First, towards a proof by contraposition, assume that the minimal linear safety ADFA$^+$ $\mathcal{A}$ has the mp-prop and consider a max-visiting word $w = \sigma_1 \dots \sigma_{\lambda+1} \in \mathcal{K}(\mathcal{A})$. Then there are some indices i, j with $w[i, j; l] \notin \mathcal{L}(\mathcal{A})$ for every l. Thus, when trying to build a DFA $\mathcal{B} \in \alpha(\mathcal{A})$ rejecting w, we know that $\mathcal{B}$ is allowed to reject every $w[i, j; l]$ as well. We can use this to construct $\mathcal{B}$ out of $\mathcal{A}$ by essentially merging the states that $\mathcal{A}$ reaches after reading $\sigma_1 \dots \sigma_{i-1}$ and $\sigma_1 \dots \sigma_{j-1}$. Since we can build a suitable DFA for every max-visiting word, $\mathcal{A}$ is composite. Details for this construction are given in the full version [11, Definition A.6, Figure 4b and Lemma A.7].

Now assume that $\mathcal{A}$ does not have the mp-prop. Then there is a word $w \in \mathcal{K}(\mathcal{A})$ that breaks the mp-prop, meaning that, for every indices i, j, there is an l with $w[i, j; l] \in \mathcal{L}(\mathcal{A})$. We argue that this w is a primality witness of $\mathcal{A}$.

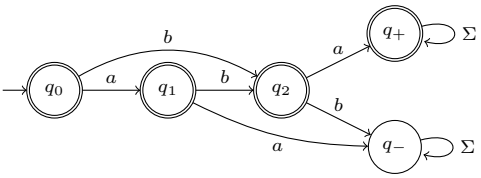
Towards a contradiction, assume that there is a $\mathcal{B} \in \alpha(\mathcal{A})$ with $w \notin \mathcal{L}(\mathcal{B})$. We pointed out above that there have to be indices i, j so that, for every l, the DFA $\mathcal{B}$ cannot differentiate between w and $w[i, j; l]$, meaning that $\mathcal{B}$ rejects every such pumping $w[i, j; l]$. Yet, because w breaks the mp-prop, there is some l so that $w[i, j; l] \in \mathcal{L}(\mathcal{A})$, yielding a contradiction.

Since there is no $\mathcal{B} \in \alpha(\mathcal{A})$ with $w \notin \mathcal{L}(\mathcal{B})$, the selected max-visiting word w is a primality witness of $\mathcal{A}$, meaning that $\mathcal{A}$ is prime.

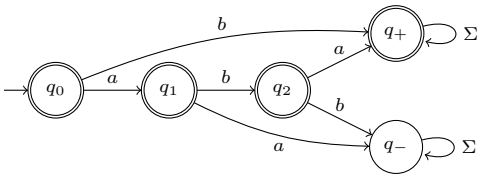
In conclusion, if $\mathcal{A}$ has the mp-prop, we can build, for every $w \in \mathcal{K}(\mathcal{A})$, a suitable DFA rejecting w by merging appropriate states of $\mathcal{A}$. Otherwise, that is, if $\mathcal{A}$ does not have the mp-prop, then every word $w \in \mathcal{K}(\mathcal{A})$ breaking the mp-prop is a primality witness of $\mathcal{A}$ because every DFA in $\alpha(\mathcal{A})$ is necessarily confused about some pumpings of w and can therefore not reject w itself. In total, we argued that a minimal linear safety ADFA$^+$ is prime if and only if it does not have the mp-prop, which is formalized above in Theorem 3.9.

We conclude this discussion of the compositionality of minimal linear safety ADFA$^+$s by using an example to further illustrate the main ideas.

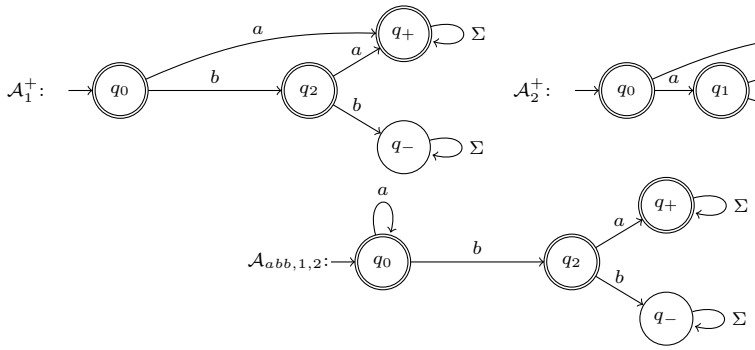
▶ **Example 3.10.** In this example, we consider two simple minimal linear safety ADFA$^+$s and argue that one of them has the mp-prop, while the other has not. Further, we outline how this leads to the one being composite and the other being prime.



(a) Minimal linear safety ADFA$^+$ $\mathcal{A}$ from Example 3.10. It has the mp-prop. Thus, it is composite.



(b) Minimal linear safety ADFA$^+$ $\mathcal{A}'$ from Example 3.10. It does not have the mp-prop. Thus, it is prime.



(c) Decomposition of the composite minimal linear safety ADFA$^+$ $\mathcal{A}$ from Figure 2a. The three DFAs are constructed according to [11, Definitions A.3 and A.6].

Figure 2 Minimal linear safety ADFA$^+$s $\mathcal{A}$ and $\mathcal{A}'$ from Example 3.10. Decomposition of $\mathcal{A}$.

Consider the two DFAs $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ and $\mathcal{A}' = (Q', \Sigma, q_0, \delta', F')$ depicted in Figures 2a and 2b. It is easy to verify that both DFAs are minimal linear safety ADFA$^+$s and that $\text{len}(\mathcal{A}) = \text{len}(\mathcal{A}') = 2$ and $\mathcal{K}(\mathcal{A}) = \mathcal{K}(\mathcal{A}') = \{abb\}$ hold. Thus, for both ADFA$^+$s, their possession of the mp-prop, and with it their compositionality, depends solely on the max-visiting word $w = abb$. Note that $\mathcal{A}$ and $\mathcal{A}'$ differ in only one transition: We have $\delta(q_0, b) = q_2$ but $\delta'(q_0, b) = q_+$. Still, we will see that, with this difference, $\mathcal{A}$ has the mp-prop and is composite, while $\mathcal{A}'$ does not have the mp-prop and is prime.

First, we consider $\mathcal{A}$ and argue that the mp-prop-condition holds for $w = abb, i = 1, j = 2$. For $l = 0$, we have $w[1, 2; l] = a^l bb = a^0 bb = bb \notin \mathcal{L}(\mathcal{A})$. Here, the transition $\delta(q_0, b) = q_2$ (in contrast to $\delta'(q_0, b) = q_+$) is crucial. For $l = 1$, we trivially have $w[1, 2; l] = a^l bb = a^1 bb = w \notin \mathcal{L}(\mathcal{A})$. And lastly, for $l \geq 2$, we have $w[1, 2; l] = a^l bb \notin \mathcal{L}(\mathcal{A})$ due to $\delta(q_0, aa) = q_-$. Thus, the mp-prop-condition holds for $w, i = 1, j = 2$. Since $\mathcal{K}(\mathcal{A}) = \{w\}$, this immediately implies that $\mathcal{A}$ has the mp-prop.

With Theorem 3.9, $\mathcal{A}$ having the mp-prop implies that it is composite. In the full version, we explicitly state the decomposition of ADFA$^+$s with the mp-prop [11, Lemma A.8] and outline all necessary DFAs [11, Definitions A.3 and A.6 and Figure 4]. Here, we discuss the decomposition of the specific ADFA$^+$ $\mathcal{A}$ to motivate how its compositionality is connected to the mp-prop.

The decomposition of $\mathcal{A}$ consists of the three simple DFAs $\mathcal{A}_1^+, \mathcal{A}_2^+$ and $\mathcal{A}_{w,1,2}$ depicted in Figure 2c. In this decomposition, $\mathcal{A}_{w,1,2}$ is the only DFA exploiting the mp-prop. It is used to reject the max-visiting word w (and its extensions). Basically, it omits the state q_1 and instead adds a self-loop $q_0 \xrightarrow{a} q_0$. For this to be allowed, it is crucial that $w[1, 2; l] = a^l bb \notin \mathcal{L}(\mathcal{A})$ holds for every $l \in \mathbb{N}$, that is, that the mp-prop-condition holds for $w, i = 1, j = 2$. Further, $\mathcal{A}_1^+$ and $\mathcal{A}_2^+$ reject the words rejected by $\mathcal{A}$ on which $\mathcal{A}$ does not enter q_1 or q_2, respectively. They are constructed out of $\mathcal{A}$ by omitting the respective state q_i and redirecting all in-q_i-transition into q_+.

Indeed, in this instance, we have $\mathcal{L}(\mathcal{A}_{w,1,2}) \subseteq \mathcal{L}(\mathcal{A}_1^+)$, meaning that $\mathcal{A}_1^+$ can be omitted in the decomposition of $\mathcal{A}$. This is due to the simplicity of $\mathcal{A}$: For example, simply by adding the letter c and the transitions $\delta(q_0, c) = q_2$ and $\delta(q_1, c) = \delta(q_2, c) = q_+$ to $\mathcal{A}$, we get a still composite DFA in whose decomposition all three DFAs of [11, Lemma A.8] are necessary.

Next, we consider $\mathcal{A}'$ and argue that $w = abb$ breaks the mp-prop. This follows from the simple observation that $w[i, j; 0] \in \mathcal{L}(\mathcal{A}')$ holds for all $i, j \in \{1, 2, 3\}, i < j$. For $i = 1, j = 2$, we have $w[1, 2; 0] = a^0bb = bb \in \mathcal{L}(\mathcal{A}')$. Here, the transition $\delta'(q_0, b) = q_+$ (in contrast to $\delta(q_0, b) = q_2$) is crucial. For $i = 1, j = 3$, we have $w[1, 3; 0] = (ab)^0b = b \in \mathcal{L}(\mathcal{A}')$. And lastly, for $i = 2, j = 3$, we have $w[2, 3; 0] = ab^0b = ab \in \mathcal{L}(\mathcal{A}')$. In total, there is no pair $i, j \in \{1, 2, 3\}, i < j$ so that the mp-prop-condition holds for w, i, j, meaning that w breaks the mp-prop.

With Theorem 3.9, $\mathcal{A}'$ not having the mp-prop implies that it is prime. In the full version [11, Appendix A.2.1], we present the complete proof that, for an ADFA$^+$ without the mp-prop, a max-visiting word breaking the mp-prop is a primality witness. Here, we discuss why w is a primality witness of the specific ADFA$^+$ $\mathcal{A}'$.

For $\mathcal{A}'$ to be composite, there would have to be a $\mathcal{B} \in \alpha(\mathcal{A}')$ with $w \notin \mathcal{L}(\mathcal{B})$. This $\mathcal{B}$ would have to have at least one non-sink less than $\mathcal{A}'$ (see full version [11, Lemmas A.10 and A.11]) and could therefore visit only two distinct states while reading the prefix ab of w. Thus, $\mathcal{B}$ would "confuse" w with bb, b or ab. But, as we argued above, these three words are all accepted by $\mathcal{A}'$, meaning that every DFA in $\alpha(\mathcal{A}')$ has to accept them as well. Thus, no suitable $\mathcal{B} \in \alpha(\mathcal{A}')$ with $w \notin \mathcal{L}(\mathcal{B})$ exists, meaning that w is a primality witness of $\mathcal{A}'$. ⌟

3.2.2 From the Characterization to NP-Hardness

For Section 3.2.2, let $\lambda = \text{len}(\mathcal{A}_\Phi)$ and $\mu = (\lambda + 1) - (r + 1)$.

Together, Lemma 3.5 and Theorem 3.9 imply that $\mathcal{A}_\Phi$ is prime if and only if it does not have the mp-prop. Following Definition 3.7, this means that exactly the max-visiting words of $\mathcal{A}_\Phi$ are relevant for the compositionality of $\mathcal{A}_\Phi$. How these look like and where they can potentially be pumped is then answered by:

▶ **Lemma 3.11.** *The following assertions hold:*

(i) $\mathcal{K}(\mathcal{A}_\Phi) = \{udc^\mu \mid u \in \{0, 1\}^r\}$.

(ii) *For every $w \in \mathcal{K}(\mathcal{A}_\Phi)$ and every $i, j \in \{1, \ldots, \lambda + 1\}, i < j$, we have: If the mp-prop-condition holds for w, i, j, then $i = 1$ and $j = r + 1$.* ⌟

Lemma 3.11 (i) formalizes the observation about the form of the max-visiting words of $\mathcal{A}_\Phi$ which we made in Section 3.1 and which we already revisited when discussing Lemma 3.5. In particular, note that, in every state of the formula device (except $\hat{p}_r^s$), reading 0 or 1 results in "skipping over" at least one state, while reading d results in entering the rejecting sink. Thus, the max-vising words of $\mathcal{A}_\Phi$ indeed have to be suffixed by c^μ.

The argumentation behind (ii) is somewhat technical. For all $w = udc^\mu \in \mathcal{K}(\mathcal{A}_\Phi)$ and all suitable i, j with $i \neq 1$ or $j \neq r + 1$, we can show $\delta_\Phi(p_0, w[i, j; 0]) \neq p_-$, meaning that it suffices to consider the pumpings with $l = 0$ to prove (ii). For the simple case $i > r + 1$, we have $w[i, j; 0] = udc^x$ for some $x < \mu$, which is akin to removing some c-suffix of w. Clearly, this means that the initial run of $\mathcal{A}_\Phi$ on $w[i, j; 0]$ is identical to the one on w until it ends somewhere in the formula device, without reaching the rejecting sink. For the other cases, that is, for $1 < i \leq r + 1$ as well as for $i = 1$ and $j \neq r + 1$, the key observation is that, in $w[i, j; 0]$, some part of the prefix ud is removed so that a c or a d is read "out of place". This means that a c is read in one of the states $p_0, \ldots, p_r$, or a d is read in one of the states $p_1, \ldots, p_{r-1}$, which results in entering p_+.

With Lemma 3.11 (i), the form of the max-vising words of $\mathcal{A}_\Phi$ is highly restricted: As mentioned, they consist of an assignment string followed by dc^μ. Thus, every assignment string $u \in \{0,1\}^r$ corresponds to exactly one max-visiting word in $\mathcal{K}(\mathcal{A}_\Phi)$, namely, udc^μ, and vice versa. Then (ii) greatly restricts where pumping positions for the max-visiting words can lie that could witness that $\mathcal{A}_\Phi$ has the mp-prop. For every $w \in \mathcal{K}(\mathcal{A}_\Phi)$, only $i = 1, j = r + 1$ are possible candidates. This means that $\mathcal{A}_\Phi$ does not have the mp-prop, and is therefore prime, if and only if $\delta_\Phi(p_0, w[1, r + 1; l]) \neq p_-$ for some $w \in \mathcal{K}(\mathcal{A}_\Phi)$ and $l \in \mathbb{N}$. And because the words in $\mathcal{K}(\mathcal{A}_\Phi)$ have the form udc^μ for $u \in \{0,1\}^r$, the relevant pumpings are of the form $u^l dc^\mu$, meaning that we pump, and thereby repeat, exactly the assignment strings.

We have established that only the pumpings of the form $u^l dc^\mu$ are relevant for the compositionality of $\mathcal{A}_\Phi$. With the following lemma, we finally cement the connection between the compositionality of $\mathcal{A}_\Phi$ and the satisfiability of Φ, which was already perceivable in Lemma 3.3.

▶ **Lemma 3.12.** *Let $u \in \{0,1\}^r$ be an assignment string and let $w = udc^\mu \in \mathcal{K}(\mathcal{A}_\Phi)$ be the corresponding max-visiting word. Then the mp-prop-condition does not hold for $w, i = 1, j = r + 1$ if and only if the assignment γ_u induced by u satisfies Φ.* ⌟

As Lemma 3.12 connects the compositionality of $\mathcal{A}_\Phi$ and the satisfiability of Φ, it is central for the correctness of our reduction. Also, the main ideas behind the design of $\mathcal{A}_\Phi$ become apparent in its proof. Therefore, we briefly argue that it holds. Let $u \in \{0,1\}^r$ be an assignment string and let $w = udc^\mu$ be the corresponding max-visiting word.

First, towards a proof by contraposition, we assume that the assignment γ_u induced by u does not satisfy Φ. We argue that the mp-prop-condition holds for $w, i = 1, j = r + 1$. Because γ_u does not satisfy Φ, there is a $k \in \{1, \ldots, s\}$ so that γ_u does not satisfy the k-th clause of Φ. Let k be the minimum value for which this holds.

The key insight here is that, firstly, for all $1 < l < k + 1$, the pumping $w[1, r + 1; l]$ is rejected because a d is read "out of place", namely, in the state $\hat{p}_r^{l-1}$; and that, secondly, for all $l \geq k + 1$, the pumping $w[1, r + 1; l]$ is rejected because, after traversing the k-th clause row on u, the state p_r^k is reached, from which every continuation with a letter $\sigma \in \{0, 1, d\}$ leads to p_-.

To be more precise, for all $1 < l < k + 1$, we have $\delta_\Phi(p_0, u^l) = \hat{p}_r^{l-1}$ and therefore $\delta_\Phi(p_0, w[1, r + 1; l]) = p_-$, with Lemma 3.3. This holds because γ_u satisfies the first $k - 1$ clauses, meaning that, if traversing any of these clause rows on u, then $\mathcal{A}_\Phi$ ends up in a $\hat{p}$-state, from which p_- is entered on reading the subsequent d. Further, for all $l \geq k + 1$, we have $\delta_\Phi(p_0, u^{k+1}) = p_r^k$ and therefore $\delta_\Phi(p_0, w[1, r + 1; l]) = p_-$, again with Lemma 3.3. This holds because γ_u does not satisfy the k-th clause, meaning that, if traversing the k-th clause row on u, then $\mathcal{A}_\Phi$ ends up in the state p_r^k, from which p_- is entered on reading any of the possible subsequent letters 0, 1 or d. Lastly, $\delta_\Phi(p_0, w[1, r + 1; 0]) = p_-$ and $\delta_\Phi(p_0, w[1, r + 1; 1]) = p_-$ hold obviously. In total, we have $\delta_\Phi(p_0, w[1, r + 1; l]) = p_-$ for all $l \in \mathbb{N}$. So the mp-prop-condition holds for $w, i = 1, j = r + 1$.

Second, we assume that the assignment γ_u induced by u satisfies Φ. We argue that the mp-prop-condition does not hold for $w, i = 1, j = r + 1$.

The key insight here is that, for a satisfying assignment γ and a sufficiently large l, $\mathcal{A}_\Phi$ reaches $\hat{p}_r^s$ on reading u_γ^l. The reason is that, because γ satisfies every clause, $\mathcal{A}_\Phi$ enters the $\hat{p}$-states in every clause row and therefore always advances to the next clause row, finally reaching $\hat{p}_r^s$. From there, the transitions are such that p_+ is entered.

More precisely, with Lemma 3.3, we have $\delta_\Phi(p_0, u^{s+1}) = \hat{p}_r^s$. The argument is similar to above, only that now the assignment satisfies all clauses, not just the clauses up to some $k < s$. With $\delta_\Phi(\hat{p}_r^s, \sigma) = p_+$ for $\sigma \in \{0, 1\}$, we then immediately have $\delta_\Phi(p_0, w[1, r + 1; s + 2]) =$

$p_+ \neq p_-$. Thus, $l = s + 2$, and indeed every $l \geq s + 2$, witnesses that there is an $l \in \mathbb{N}$ with $\delta_\Phi(p_0, w[1, r + 1; l]) = \delta_\Phi(p_0, u^l dc^\mu) \neq p_-$. So the mp-prop-condition does not hold for $w, i = 1, j = r + 1$.

In total, we have argued that, for every $w = udc^\mu \in \mathcal{K}(\mathcal{A}_\Phi)$, the mp-prop-condition does not hold for $w, i = 1, j = r + 1$ if and only if the assignment γ_u induced by u satisfies Φ. Thus, Lemma 3.12 holds.

With Theorem 3.9 and Lemmas 3.5, 3.11, and 3.12, we have everything we need to prove the correctness of our reduction. That is, we have everything we need to prove Lemma 3.2, that Φ is satisfiable if and only if $\mathcal{A}_\Phi$ is prime.

Proof of Lemma 3.2. With Lemma 3.5, the CNF-DFA $\mathcal{A}_\Phi$ is a minimal linear safety ADFA$^+$. Together with Theorem 3.9, this implies that $\mathcal{A}_\Phi$ is prime if and only if it does not have the mp-prop. Thus, we only have to prove that Φ is satisfiable if and only if $\mathcal{A}_\Phi$ does not have the mp-prop.

First, towards a proof by contraposition, let Φ be unsatisfiable. Let $w \in \mathcal{K}(\mathcal{A}_\Phi)$. With Lemma 3.11 (i), we have $w = udc^\mu$, where $u \in \{0, 1\}^r$ is an assignment string. Because Φ is unsatisfiable, the assignment γ_u induced by u does not satisfy Φ. With with Lemma 3.12, we therefore have that the mp-prop-condition holds for $w, i = 1, j = r + 1$. Thus, the mp-prop-condition holds for every max-visiting word, meaning that $\mathcal{A}_\Phi$ has the mp-prop.

Now let Φ be satisfiable. Let γ be a satisfying assignment. We consider the assignment string u_γ induced by γ and the corresponding word $w = u_\gamma dc^\mu$. With Lemma 3.11 (i), we have $w \in \mathcal{K}(\mathcal{A}_\Phi)$. And with Lemma 3.11 (ii), the mp-prop-condition does not hold for w, i, j where $i \neq 1$ or $j \neq r + 1$. Further, because γ satisfies Φ, we have, with Lemma 3.12, that the mp-prop-condition does not hold for $w, i = 1, j = r + 1$ either. Thus, w witnesses that $\mathcal{A}_\Phi$ does not have the mp-prop.

We have proven that Φ is satisfiable if and only if $\mathcal{A}_\Phi$ does not have the mp-prop. As outlined, this implies that Φ is satisfiable if and only if $\mathcal{A}_\Phi$ is prime. The proof is complete. ◀

With Lemma 3.2, we have proven that our reduction is correct. Because the construction of $\mathcal{A}_\Phi$ is clearly possible in polytime, we then immediately get Theorem 3.1, the NP-hardness of Prime-DFA.

3.3 NP-Completeness of Prime-SADFA$^+$

In Sections 3.1 and 3.2, we improved the lower complexity bound of the general problem Prime-DFA. Now we conclude Section 3 by arguing that this improved lower bound is tight for the restriction of Prime-DFA to minimal linear safety ADFA$^+$s.

We denote the restriction of Prime-DFA to DFAs whose respective minimal DFA is a minimal linear safety ADFA$^+$ by Prime-SADFA$^+$.

The restriction of Prime-DFA to ADFAs, so to truly acyclic DFAs, was studied in [10]. Motivated by this, we study Prime-SADFA$^+$. For a comparison of the compositionality of ADFAs and ADFA$^+$s, we refer to Section 4.1. Here, we prove:

▶ **Theorem 3.13.** *The problem Prime-SADFA$^+$ is NP-complete.* ⌟

We proved Theorem 3.1, the NP-hardness of Prime-DFA, by a reduction from CNFSAT. At the heart of this reduction was the construction of the CNF-DFA $\mathcal{A}_\Phi$ out of the given CNF-formula Φ. With Lemma 3.5, every CNF-DFA is also a minimal linear safety ADFA$^+$. Thus, we immediately get:

▶ **Lemma 3.14.** *The problem Prime-SADFA$^+$ is NP-hard.* ⌟

To arrive at Theorem 3.13, we additionally prove:

▶ **Lemma 3.15.** *The problem* Prime-SADFA$^+$ *is in NP.* ⌟

To prove that Prime-SADFA$^+$ is in NP, we describe a guess-and-verify NP-algorithm for Prime-SADFA$^+$ which exploits Theorem 3.9. Given a minimal linear safety ADFA$^+$ $\mathcal{A}$, the algorithm guesses a max-visiting word $w \in \mathcal{K}(\mathcal{A})$ and then verifies that w breaks the mp-prop, meaning that w witnesses that $\mathcal{A}$ does not have the mp-prop. To do so, it simulates the initial run of $\mathcal{A}$ on $w[i,j;l]$ for every pair $i,j \in \{1,\dots,\mathrm{len}(\mathcal{A})+1\}, i<j$ and every relevant l. The key insight here is that, for every such pair i,j, there is a polynomially bounded value $l'_{i,j}$ such that only the values $l \leq l'_{i,j}$ are relevant. Thus, the algorithm has to simulate only polynomially many runs. If the algorithm finds that w indeed breaks the mp-prop, it accepts. Else, it rejects.

The polynomial runtime of the algorithm follows immediately from the key insight just mentioned. The runtime is dominated by the simulation of the runs. There are only polynomially many pairs i,j. And for each pair, there are only polynomially many relevant pumpings $w[i,j;l]$, each having polynomial length. Thus, the algorithm has polynomial runtime.

Regarding the correctness, if $\mathcal{A}$ is prime, then there is a word breaking the mp-prop. The algorithm guesses this word and subsequently accepts. Else, that is, if $\mathcal{A}$ is composite, then no word breaking the mp-prop exists. Thus, the algorithm is forced to reject no matter which word it guesses. In total, the algorithm works correctly for Prime-SADFA$^+$.

To summarize, we have described an NP-algorithm for Prime-SADFA$^+$. Thus, we have proven Lemma 3.15. Together with the already established Lemma 3.14, we immediately get Theorem 3.13, the NP-completeness of Prime-SADFA$^+$.

4 Discussion

We studied the primality of DFAs and thereby of regular languages. We proved the NP-hardness of Prime-DFA. This is the first improvement of a complexity bound for Prime-DFA since [8] introduced Prime-DFA and proved that it is NL-hard and in ExpSpace.

We proved the NP-hardness of Prime-DFA by a reduction from CNFSAT. For a given CNF-formula Φ, the reduction yields a so-called CNF-DFA $\mathcal{A}_\Phi$ so that Φ is satisfiable if and only if $\mathcal{A}_\Phi$ is prime. To link the satisfiability of Φ and the compositionality of $\mathcal{A}_\Phi$, the construction of $\mathcal{A}_\Phi$ ensures that (1) Φ is encoded in $\mathcal{A}_\Phi$, and (2) the variable assignments for Φ are encoded in the words relevant for the compositionality of $\mathcal{A}_\Phi$.

Central to the NP-hardness proof was the characterization of the compositionality of CNF-DFAs. However, only certain structural aspects of CNF-DFAs were relevant for this characterization. Thus, we actually characterized the compositionality of the more general minimal linear safety ADFA$^+$s. Exploiting this characterization, we then proved the NP-completeness of Prime-SADFA$^+$, the restriction of Prime-DFA to DFAs whose respective minimal DFA is a minimal linear safety ADFA$^+$.

To wrap up this paper, we briefly discuss two points.

Firstly, we compare the complexity of Prime-SADFA$^+$ and Prime-DFA$_{\mathrm{fin}}$, the restriction of Prime-DFA to DFAs deciding finite languages. And secondly, we highlight that the NP-hardness of Prime-DFA carries over to S-Prime-DFA, which is a variant of the problem based on a slightly different notion of compositionality.

4.1 Prime-SADFA$^+$ and Prime-DFA$_{\text{fin}}$

We denote the restriction of PRIME-DFA to DFAs deciding a finite language by PRIME-DFA$_{\text{fin}}$. Note that a DFA decides a finite language if and only if its minimal DFA is an ADFA. Thus, the problems PRIME-DFA$_{\text{fin}}$ and PRIME-SADFA$^+$ are very similar: PRIME-DFA$_{\text{fin}}$ considers ADFAs, so truly acyclic DFAs, while PRIME-SADFA$^+$ considers ADFA$^+$s, so "almost" acyclic DFAs possessing an accepting sink.

In this paper, we have proven that PRIME-SADFA$^+$ is NP-complete. With [10, Theorem 4.1], PRIME-DFA$_{\text{fin}}$ is NL-complete. Therefore, the simple addition of an accepting sink leads to a jump in the complexity of the primality problem (assuming NL $\subset$ NP). We give an intuition for this somewhat surprising fact.

The reduction from CNFSAT crucially hinges on the fact that the CNF-DFA $\mathcal{A}_\Phi$ can potentially enter the accepting sink after reading the prefix u^{s+2} of $w[1, r+1; s+2]$, where $w = udc^\mu \in \mathcal{K}(\mathcal{A}_\Phi)$ is a max-visiting word. In other words, it can enter the accepting sink after "pumping up" w so that the assignment string u is checked against every clause row. Once in the accepting sink, the suffix dc^μ of dummy letters can be read and the pumping $w[1, r+1; s+2]$ is accepted.

No construction of this kind is possible for ADFAs. Every ADFA has a threshold value so that it rejects every word longer than that threshold value. This is obvious, because for words longer than the threshold value, it runs out of non-sinks and necessarily enters the rejecting sink. Therefore, "pumping up" and thereby lengthening the ADFA-equivalent of a max-visiting word necessarily leads to an overlong word that is rejected.

Thus, with the existence of an accepting sink, no threshold value exists so that words longer than that threshold value are rejected. This simple additional complication is enough to lead to the jump in complexity, from NL-completeness for PRIME-DFA$_{\text{fin}}$ to NP-completeness for PRIME-SADFA$^+$.

4.2 NP-Hardness of S-Prime-DFA

With Definition 2.1, we followed [8] and defined compositionality using the index of the given DFA. However, without stating so explicitly, [6, 7] employed a slightly different definition, using the size instead of the index. In [10], this difference was made explicit by introducing the notion of S-compositionality and the respective decision problem. We follow [10] and define:

▶ **Definition 4.1.** A DFA $\mathcal{A}$ is *S-composite* if it is $(|\mathcal{A}| - 1)$-decomposable. Otherwise, it is *S-prime*. ⌟

We denote the problem of deciding S-primality for a given DFA by S-PRIME-DFA.

Many results known for compositionality carry over trivially to S-compositionality. In particular, PRIME-DFA is in EXPSPACE [8, Theorem 2.4]. The proof of this result has to be adapted only slightly to prove that S-PRIME-DFA is in EXPSPACE as well [10, Theorem 6.3].

However, the NL-hardness of PRIME-DFA, established in [8, Theorem 2.5], does not carry over trivially to S-PRIME-DFA. The NL-hardness of S-PRIME-DFA is established in [10, Theorem 6.3] by using a reduction of a different problem than the one used for PRIME-DFA.

Here, we highlight that the improved lower complexity bound of PRIME-DFA does carry over trivially to S-PRIME-DFA. We established the NP-hardness of PRIME-DFA by a reduction from CNFSAT. The CNF-DFAs yielded by this reduction are minimal. Thus, with the same reduction, we get:

▶ **Theorem 4.2.** *The problem S-PRIME-DFA is NP-hard.* ⌟

Indeed, it is well known that DFA minimization can be done in polytime. Thus, by making the transition from NL-hardness to NP-hardness, the distinction between compositionality and S-compositionality loses its edge because we can always minimize the given DFA. This simple argument is an alternative way to arrive at Theorem 4.2.

References

1 Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008. URL: https://mitpress.mit.edu/9780262026499/principles-of-model-checking/.

2 Stephen A. Cook. The complexity of theorem-proving procedures. In Michael A. Harrison, Ranan B. Banerji, and Jeffrey D. Ullman, editors, *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158. ACM, 1971. doi:10.1145/800157.805047.

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 Peter Gazi and Branislav Rovan. Assisted problem solving and decompositions of finite automata. In Viliam Geffert, Juhani Karhumäki, Alberto Bertoni, Bart Preneel, Pavol Návrat, and Mária Bieliková, editors, *SOFSEM 2008: Theory and Practice of Computer Science, 34th Conference on Current Trends in Theory and Practice of Computer Science, Nový Smokovec, Slovakia, January 19-25, 2008, Proceedings*, volume 4910 of *Lecture Notes in Computer Science*, pages 292–303. Springer, 2008. doi:10.1007/978-3-540-77566-9_25.

5 E. Mark Gold. Complexity of automaton identification from given data. *Inf. Control.*, 37(3):302–320, 1978. doi:10.1016/S0019-9958(78)90562-4.

6 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.

7 Ismaël Jecker, Nicolas Mazzocchi, and Petra Wolf. Decomposing permutation automata. In Serge Haddad and Daniele Varacca, editors, *32nd International Conference on Concurrency Theory, CONCUR 2021, August 24-27, 2021, Virtual Conference*, volume 203 of *LIPIcs*, pages 18:1–18:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CONCUR.2021.18.

8 Orna Kupferman and Jonathan Mosheiff. Prime languages. *Inf. Comput.*, 240:90–107, 2015. doi:10.1016/J.IC.2014.09.010.

9 Niklas Lauffer, Beyazit Yalcinkaya, Marcell Vazquez-Chanlatte, Ameesh Shah, and Sanjit A. Seshia. Learning deterministic finite automata decompositions from examples and demonstrations. In Alberto Griggio and Neha Rungta, editors, *22nd Formal Methods in Computer-Aided Design, FMCAD 2022, Trento, Italy, October 17-21, 2022*, pages 325–330. TU Wien Academic Press, 2022. doi:10.34727/2022/ISBN.978-3-85448-053-2_39.

10 Daniel Alexander Spenner. Decomposing finite languages. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science, MFCS 2023, August 28 to September 1, 2023, Bordeaux, France*, volume 272 of *LIPIcs*, pages 83:1–83:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.MFCS.2023.83.

11 Daniel Alexander Spenner. Deciding DFA-primality is NP-hard, 2026. arXiv:2605.07031.

12 Stavros Tripakis. Compositionality in the science of system design. *Proc. IEEE*, 104(5):960–972, 2016. doi:10.1109/JPROC.2015.2510366.

13 Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification. In *Proceedings of the Symposium on Logic in Computer Science (LICS '86), Cambridge, Massachusetts, USA, June 16-18, 1986*, pages 332–344. IEEE Computer Society, 1986. URL: https://hdl.handle.net/2268/116609.