Decomposing Finite Languages

Daniel Alexander Spenner
Technische Universität Dortmund, Germany

Abstract

The paper completely characterizes the primality of acyclic DFAs, where a DFA $\mathcal{A}$ is *prime* if there do not 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 recognizing the same language as $\mathcal{A}$. A regular language is prime if its minimal DFA is prime. Thus, this result also characterizes the primality of finite languages.

Further, the NL-completeness of the corresponding decision problem Prime-DFA$_{\text{fin}}$ is proven. The paper also characterizes the primality of acyclic DFAs under two different notions of compositionality, union and union-intersection compositionality.

Additionally, the paper introduces the notion of *S-primality*, where a DFA $\mathcal{A}$ is S-prime if there do not 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 $\mathcal{A}$ itself. It is proven that the problem of deciding S-primality for a given DFA is NL-hard. To do so, the NL-completeness of 2Minimal-DFA, the basic problem of deciding minimality for a DFA with at most two letters, is proven.

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, Minimality

Digital Object Identifier 10.4230/LIPIcs.MFCS.2023.83

Related Version The full version includes the appendix, which contains proofs for all results.
Full Version: https://arxiv.org/abs/2307.06802

Acknowledgements I want to thank Thomas Schwentick for his advice and encouragement.

1 Introduction

Under intersection compositionality a deterministic finite automaton (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 each $\mathcal{A}_i$ is smaller than the index of $\mathcal{A}$. Otherwise, $\mathcal{A}$ is *prime* [10]. The index of $\mathcal{A}$ is the size of the minimal DFA recognizing the same language as $\mathcal{A}$. Prime-DFA denotes the problem of deciding primality for a given DFA. Prime-DFA$_{\text{fin}}$ denotes the restriction of Prime-DFA to DFAs recognizing a finite language.

Compositionality in general is a key concept in both practical and theoretical computer science [3, 16]. The intersection decomposition of finite automata can be motivated by LTL model checking as well as automaton identification. Both will be briefly discussed below.

The notion of intersection compositionality of finite automata was introduced in [10], while a limitation of this notion was already studied in [5]. Surprisingly, [10] found even the complexity of the basic problem Prime-DFA to be open. They proved that Prime-DFA is in ExpSpace and is NL-hard. So far, this doubly exponential gap has not been closed.

Given the difficulties in tackling the general problem, it has proven fruitful to characterize the intersection compositionality of fragments of the regular languages [10, 8, 9]. Our study joins this line of research by completely characterizing the intersection compositionality of acyclic DFAs (ADFA) and thereby of finite languages. Further, we prove the NL-completeness of Prime-DFA$_{\text{fin}}$ and characterize the compositionality of finite languages under two different notions of compositionality suggested in [10], union and union-intersection compositionality.

Additionally, we present a proof of the NL-completeness of the basic problem 2Minimal-DFA, the problem of deciding minimality for a DFA with at most two letters. For arbitrary alphabets, the NL-hardness is a folklore result that seemingly has not been explicitly published but follows from the constructions in [2], while the NL-hardness of 2Minimal-DFA appears to be new [4]. We use this result to establish complexity boundaries for S-Prime-DFA, a modification of Prime-DFA using the size of the given DFA, not its index.

Related Work. The notion of intersection compositionality was introduced in [10], where the aforementioned complexity boundaries were established. They already considered language fragments, analyzing safety DFAs and permutation DFAs. This line of research was followed up in [8, 9], which focused on unary DFAs and permutation DFAs, respectively.

The intersection decomposition of 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 [17]. Since the LTL model checking problem is PSpace-complete in the size of the LTL formula [1], 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.

Another application of intersection decomposition arises in the field of automaton identification. The basic task here is, given a set of labeled words, to construct a finite automaton conforming to this set [6]. An interesting approach is to construct multiple automata instead of one, which can lead to smaller and more intuitive solutions [11].

An alternative notion of compositionality uses concatenation. Here, a language L is composite if there exist two non-trivial languages L_1, L_2 with $L = L_1 L_2$. The concatenation primality problem for regular languages is PSpace-complete [12]. The restriction to finite languages is known to be NP-hard [15], while the conjectured NP-completeness of this restriction remains open [14, 13, 18].

Contributions. In Section 3 we completely characterize the intersection compositionality of ADFAs and thereby of finite languages. We expand on this by proving the NL-completeness of Prime-DFA$_{\text{fin}}$ in Section 4, thus showing that finite languages are significantly easier to handle under intersection compositionality than under concatenation compositionality. We characterize the union and union-intersection compositionality of finite languages in Section 5, where we also prove the existence of languages that are union-intersection composite but both union prime and intersection prime.

In Section 6 we introduce the problem S-Prime-DFA, which is analogous to Prime-DFA but uses the size for the definition of compositionality, not the index. We prove that S-Prime-DFA is in ExpSpace and is NL-hard. We also prove these boundaries for 2Prime-DFA and 2S-Prime-DFA, the restrictions of the respective problems to DFAs with at most two letters. To establish these boundaries we prove the NL-completeness of 2Minimal-DFA.

Detailed proofs of these results are provided in the appendix.

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 $\delta(q, \sigma_1 \dots \sigma_n) = \delta(\delta(q, \sigma_1 \dots \sigma_{n-1}), \sigma_n)$. For $q \in Q$, the DFA $\mathcal{A}^q$ is constructed out of $\mathcal{A}$ by setting q as the initial state, thus $\mathcal{A}^q = (Q, \Sigma, q, \delta, F)$.

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 $\mathcal{A}$ *recognizes* $\mathcal{L}(\mathcal{A})$. A language is *regular* if there exists a DFA recognizing 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. The 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 recognizing 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 recognized by $\mathcal{A}$, thus $\operatorname{ind}(\mathcal{A}) = \operatorname{ind}(\mathcal{L}(\mathcal{A}))$. Note that $\mathcal{A}$ is minimal iff $|\mathcal{A}| = \operatorname{ind}(\mathcal{A})$.

We borrow a few terms from graph theory. Let $q_0, \sigma_1, q_1, \dots, \sigma_n, q_n$ be the run of $\mathcal{A}$ on $w = \sigma_1 \dots \sigma_n$ starting in q_0. Then $q_0, \dots, q_n$ is a *path* in $\mathcal{A}$ from q_0 to q_n. The *length* of this path is n. Thus, for two states q, q' there exists a path from q to q' in $\mathcal{A}$ of length n iff there exists a $w \in \Sigma^n$ with $\delta(q, w) = q'$. The state q' is *reachable from* q if there exists a path from q to q'. Otherwise, q' is *unreachable from* q. Obviously, if q' is reachable from q then there exists a path from q to q' of a length strictly smaller than $|\mathcal{A}|$. 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 $q_0, \dots, q_n$ in $\mathcal{A}$ where $q_0 = q_n$ and $n \in \mathbb{N}_{\geq 1}$. The DFA $\mathcal{A}$ is *acyclic* (ADFA) if every cycle in $\mathcal{A}$ begins in a rejecting sink. Clearly, a DFA recognizes a finite language iff its minimal DFA is acyclic.

We call a DFA $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ *linear* if for every $q, q' \in Q$ with $q \neq q'$ either q' is reachable from q or q is reachable from q', but not both. Thus, in a linear DFA reachability induces a linear order over the states. Obviously, every linear DFA has exactly one sink. Furthermore, a minimal ADFA $\mathcal{A}$ is linear iff $|\mathcal{A}| = n + 2$, where n is the length of the longest word in $\mathcal{L}(\mathcal{A})$.

Consider a word $w = \sigma_1 \dots \sigma_n \in \Sigma^n$. A word wv with $v \in \Sigma^+$ is an *extension* of w. A word $\sigma_1 \dots \sigma_i \sigma_{i+l} \dots \sigma_n$ with $i \in \{0, \dots, n-2\}, l \in \{2, \dots, n-i\}$ is a *compression* of w. An ADFA $\mathcal{A}$ has the *compression-extension-property* (CEP) if for every $w \in \mathcal{L}(\mathcal{A})$ with $|w| = n$, where n is the length of the longest word in $\mathcal{L}(\mathcal{A})$, there exists a compression w' of w such that every extension of w' is rejected by $\mathcal{A}$.

We introduce a type of DFA already inspected in [10]. A regular language $L \subseteq \Sigma^*$ is a *safety language* if $w \notin L$ 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. A regular language $L \subseteq \Sigma^*$ is a *co-safety language* if the complement language $\overline{L}$ of L is a safety language. A DFA $\mathcal{A}$ is a *co-safety DFA* if $\mathcal{L}(\mathcal{A})$ is a co-safety language. Clearly, every non-trivial minimal safety DFA has exactly one rejecting state, and this state is a sink. Conversely, every non-trivial minimal co-safety DFA has exactly one accepting state, and this state is a sink.

We introduce the notions of intersection compositionality and primality of DFAs and languages, following the definitions in [10]:

▶ **Definition 2.1.** For $k \in \mathbb{N}_{\geq 1}$, a DFA $\mathcal{A}$ is *k-decomposable* if there exist DFAs $\mathcal{A}_1, \dots, \mathcal{A}_t$ with $\mathcal{L}(\mathcal{A}) = \bigcap_{i=1}^{t} \mathcal{L}(\mathcal{A}_i)$ and $|\mathcal{A}_i| \leq k$ for each $i \in \{1, \dots, t\}$, where $t \in \mathbb{N}_{\geq 1}$. We call such DFAs $\mathcal{A}_1, \dots, \mathcal{A}_t$ 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*. ⌟

We use compositionality or $\cap$-compositionality when referring to intersection compositionality.

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 iff $\mathcal{L}(\mathcal{A}) = \bigcap_{\mathcal{B} \in \alpha(\mathcal{A})} \mathcal{L}(\mathcal{B})$. 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, the DFA $\mathcal{A}$ is composite iff $\mathcal{A}$ has no primality witness.

We extend the notions of k-decompositions, compositionality, primality and primality witnesses to regular languages by identifying a regular language with its minimal DFA.

We denote the problem of deciding primality for a given DFA with Prime-DFA. We denote the restriction of Prime-DFA to DFAs recognizing finite languages with Prime-DFA$_{\text{fin}}$. Prime-DFA is in ExpSpace and is NL-hard [10].

We denote the connectivity problem in directed graphs, which is NL-complete [7], with STCON. We denote the restriction of STCON to graphs with a maximum outdegree of two with 2STCON. Clearly, 2STCON is NL-complete as well. We denote the problem of deciding minimality for a given DFA with Minimal-DFA. For $k \in \mathbb{N}_{\geq 2}$, the problem kMinimal-DFA is the restriction of Minimal-DFA to DFAs with at most k letters. As mentioned in Section 1, the NL-completeness of kMinimal-DFA for $k \in \mathbb{N}_{\geq 3}$ is folklore, while the NL-hardness of 2Minimal-DFA appears to be open.

3 Compositionality of Finite Languages

We characterize the compositionality of ADFAs and thereby of finite languages by proving:

▶ **Theorem 3.1.** *Consider a minimal ADFA $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ recognizing a non-empty language. Then $\mathcal{A}$ is prime iff $\mathcal{A}$ is linear and:*

(i) *$\sigma^n \in \mathcal{L}(\mathcal{A})$ for some $\sigma \in \Sigma$, where $n \in \mathbb{N}$ is the length of the longest word in $\mathcal{L}(\mathcal{A})$, or*

(ii) *$\mathcal{A}$ is a safety DFA and $\mathcal{A}$ does not have the CEP.* ⌟

To prove Theorem 3.1 we will consider five cases in turn.

First, if the ADFA $\mathcal{A}$ is not linear we essentially have a surplus of states, allowing us to construct one DFA rejecting overlong words and one specific DFA for each of the remaining words also rejected by $\mathcal{A}$. This approach fails with linear ADFAs. Nevertheless, we will come back to the idea of excluding words longer than a threshold value and tailoring a DFA for each word shorter than the threshold value which has to be rejected as well.

Second, if $\mathcal{A}$ is linear and $\sigma^n \in \mathcal{L}(\mathcal{A})$ holds the DFAs in $\alpha(\mathcal{A})$ do not possess enough states to differentiate the words $\sigma^0, \ldots, \sigma^n$ but have to accept σ^n, which implies cyclic behavior on the words in $\{\sigma\}^*$ from which primality follows.

Third, if there is no $\sigma \in \Sigma$ with $\sigma^n \in \mathcal{L}(\mathcal{A})$ and $\mathcal{A}$ is not a safety DFA we can return to the idea of excluding words longer than a threshold value. For each of the words left to reject, it is possible to construct a DFA similar to $\mathcal{A}$ but without the rejecting sink, which circles back to the rejecting non-sink.

Fourth, if there is no $\sigma \in \Sigma$ with $\sigma^n \in \mathcal{L}(\mathcal{A})$ and $\mathcal{A}$ has the CEP we can utilize DFAs similar to $\mathcal{A}$ possessing a rejecting sink, since the CEP allows us to skip over one state.

Fifth and finally, if $\mathcal{A}$ is linear and $\mathcal{A}$ is a safety DFA and does not have the CEP both of the above approaches fail. There is no state to circle back to, and for the word breaching the CEP skipping over states is not possible either, which implies primality.

Formalizing these five cases, we get:

▷ Claim 3.2. Consider a minimal ADFA $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ recognizing a non-empty language. Let $n \in \mathbb{N}$ be the length of the longest word in $\mathcal{L}(\mathcal{A})$. The following assertions hold:

(a) $\mathcal{A}$ is composite if $\mathcal{A}$ is not linear.

(b) $\mathcal{A}$ is prime if $\mathcal{A}$ is linear and $\sigma^n \in \mathcal{L}(\mathcal{A})$ holds for some $\sigma \in \Sigma$.

(c) $\mathcal{A}$ is composite if there is no $\sigma \in \Sigma$ with $\sigma^n \in \mathcal{L}(\mathcal{A})$ and $\mathcal{A}$ is not a safety DFA.

(d) $\mathcal{A}$ is composite if there is no $\sigma \in \Sigma$ with $\sigma^n \in \mathcal{L}(\mathcal{A})$ and $\mathcal{A}$ has the CEP.

(e) $\mathcal{A}$ is prime if $\mathcal{A}$ is linear and $\mathcal{A}$ is a safety DFA and $\mathcal{A}$ does not have the CEP. ⌟

Formalizing the intuition given above for (a) and (b) is not too complex. Assertions (c)–(e) prove to be much harder. Thus, we commence by discussing (c) in Section 3.1 and (d) and (e) in Section 3.2. Henceforth, we consider a minimal ADFA $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ recognizing the non-empty language L with $\sigma^n \notin L$ for each $\sigma \in \Sigma$, where $n \in \mathbb{N}$ is the length of the longest word in L. W.l.o.g. we assume $Q = \{q_0, \ldots, q_{n+1}\}$ with q_j being reachable from q_i for all $i < j$, which implies $q_I = q_0$ and $q_n \in F$ with q_{n+1} being the rejecting sink. Finally, we define $\Sigma_{i,j} = \{\sigma \in \Sigma \mid \delta(q_i, \sigma) = q_j\}$.

3.1 Linear non-safety ADFAs

We consider Claim 3.2 (c). Therefore, we assume that $\mathcal{A}$ is not a safety DFA, which implies $\{q_n\} \subseteq F \subset Q \setminus \{q_{n+1}\}$. Let $d \in \{0, \ldots, n-1\}$ with $q_d \notin F$.

We show the compositionality of $\mathcal{A}$ by specifying an $(n+1)$-decomposition of $\mathcal{A}$. First, we construct DFAs rejecting words not in L that are not extensions of words $u \in L, |u| = n$. Afterwards, we turn to such extensions, whose handling poses the main difficulty. Here, we first construct DFAs rejecting such extensions that are longer than a certain threshold value. For the remaining extensions we employ the idea of circling back to q_d.

We begin by considering words not in L which are not extensions of words $u \in L, |u| = n$. We introduce three DFA types handling these words.

First, let $\mathcal{A}_0$ be the DFA constructed out of $\mathcal{A}$ by removing q_n, redirecting every transition $q \to q_n$ to q_0, and including q_0 into the acceptance set. Clearly, $\mathcal{A}_0 \in \alpha(\mathcal{A})$ and $\mathcal{A}_0$ rejects every $w \notin L$ on which $\mathcal{A}$ enters the rejecting sink prematurely, that is, without entering q_n.

Second, let $\hat{\mathcal{A}}_d$ be the DFA constructed out of $\mathcal{A}$ by removing q_{n+1}, redirecting every transition $q_i \to q_{n+1}$ with $i < n$ to q_n and every transition $q_n \to q_{n+1}$ to q_d. Clearly, $\hat{\mathcal{A}}_d \in \alpha(\mathcal{A})$ and $\hat{\mathcal{A}}_d$ rejects every $w \notin L$ on which $\mathcal{A}$ does not enter the rejecting sink.

Third, we construct DFAs rejecting extensions of words $w \in L, |w| < n$ with $\delta(q_0, w) = q_n$. Let $I = \{0, \ldots, n\}$. For each $m \in \{1, \ldots, n-1\}$ let $I_m = \{(i_0, \ldots, i_m) \in I^{m+1} \mid 0 = i_0 < \cdots < i_m = n\}$. For each $\underline{i} \in I_m$ define $\mathcal{A}_{\underline{i}}$ as in Figures 1a and 1b. It is easy to confirm that each $\mathcal{A}_{\underline{i}}$ is in $\alpha(\mathcal{A})$ and rejects extensions of words on which $\mathcal{A}$ visits the states $q_{i_0}, \ldots, q_{i_m}$.

Lemma 3.3 formalizes the results concerning $\mathcal{A}_0$, $\hat{\mathcal{A}}_d$ and $\mathcal{A}_{\underline{i}}$:

▶ **Lemma 3.3.** *The following assertions hold:*

(i) $\mathcal{A}_0, \hat{\mathcal{A}}_d, \mathcal{A}_{\underline{i}} \in \alpha(\mathcal{A})$*, where* $\underline{i} \in \bigcup_{m=1}^{n-1} I_m$*.*

(ii) *Consider a word* $w \notin L$*, where* w *is not an extension of a word* $u \in L, |u| = n$*. Then* $w \notin \mathcal{L}(\mathcal{A}_0) \cap \mathcal{L}(\hat{\mathcal{A}}_d) \cap \bigcap_{m=1}^{n-1} \bigcap_{\underline{i} \in I_m} \mathcal{L}(\mathcal{A}_{\underline{i}})$ *holds.* ⌟

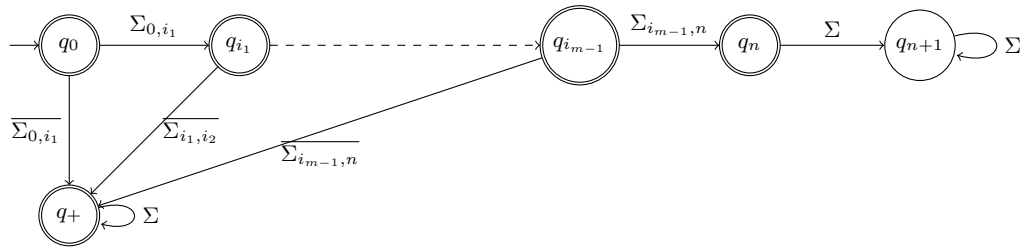
Next, we turn to the extensions of words $u \in L, |u| = n$. We begin by constructing DFAs that taken together reject every word strictly longer than $n + (n-2)$. Then we turn to the remaining extensions one by one, of which only a finite number are left to reject.

Let $\sigma \in \Sigma$. Since $\sigma^n \notin L$, there exists a value $i \in \{1, \ldots, n\}$ with $\sigma \notin \Sigma_{i-1,i}$. Define $\mathcal{A}_{\sigma,i}$ as in Figure 1c. First, note that $\mathcal{A}_{\sigma,i} \in \alpha(\mathcal{A})$ because a word rejected by $\mathcal{A}_{\sigma,i}$ is strictly longer n or is of length n with letter σ at position i. Next, consider a word $w = \sigma_1 \ldots \sigma_m \in \Sigma^m$ such that $\sigma_j = \sigma$ for a $j \in \{1, \ldots, m\}$ with $j \geq i$ and $m \geq j + (n-i)$. After reading the prefix $\sigma_1 \ldots \sigma_{j-1}$ the DFA $\mathcal{A}_{\sigma,i}$ is at least in state q_{i-1}. Thus, after reading $\sigma_1 \ldots \sigma_j$ it is at least in state q_i and will reject after reading $n-i$ more letters. Since $m \geq j + (n-i)$, we have $w \notin \mathcal{L}(\mathcal{A}_{\sigma,i})$. Lemma 3.4 formalizes this result:

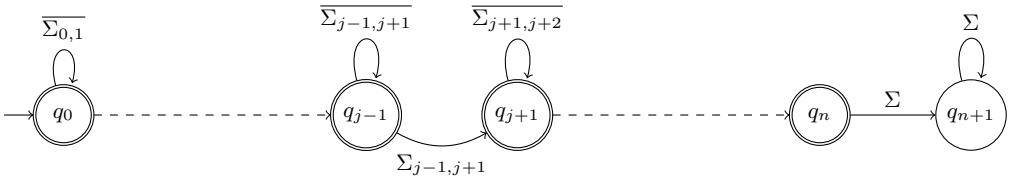
▶ **Lemma 3.4.** *Let* $\sigma \in \Sigma$ *and* $i \in \{1, \ldots, n\}$ *with* $\sigma \notin \Sigma_{i-1,i}$*. The following assertions hold:*

(i) $\mathcal{A}_{\sigma,i} \in \alpha(\mathcal{A})$*.*

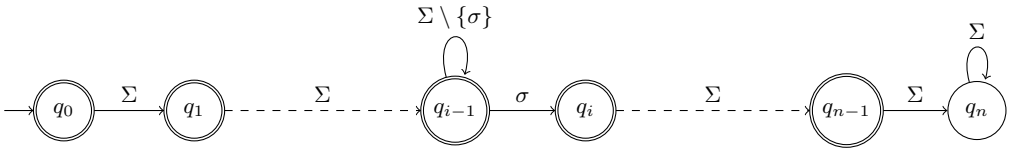
(ii) *Let* $m \in \mathbb{N}$*. Let* $w \in \sigma_1 \ldots \sigma_m \in \Sigma^m$ *such that* $\sigma_j = \sigma$ *for a* $j \in \{1, \ldots, m\}$ *with* $j \geq i$ *and* $m \geq j + (n-i)$*. Then* w *is rejected by* $\mathcal{A}_{\sigma,i}$*.* ⌟



(a) $\mathcal{A}_{\underline{i}}$ if $m < n-1$.



(b) $\mathcal{A}_{\underline{i}}$ if $m = n-1$, where $\underline{i} = (0, \dots, j-1, j+1, \dots, n)$.



(c) $\mathcal{A}_{\sigma,i}$.

Figure 1 DFA $\mathcal{A}_{\underline{i}}$ for $\underline{i} \in I_m$ with $m \in \{1, \dots, n-1\}$ and DFA $\mathcal{A}_{\sigma,i}$ for $\sigma \in \Sigma, i \in \{1, \dots, n\}$.

Now consider a word $w = \sigma_1 \dots \sigma_m \in \Sigma^m$ with $m \geq n + (n-1)$ and $\sigma_1 \dots \sigma_n \in L$. Note that Lemma 3.4 implies $w \notin \mathcal{L}(\mathcal{A}_{\sigma_n,i})$ where $i \in \{1, \dots, n\}$ with $\sigma_n \notin \Sigma_{i-1,i}$. With this limitation of length, we only need DFAs to reject the extensions of words $u \in L, |u| = n$ with a maximum length of $n + (n-2)$. Consider such an extension $w = \sigma_1 \dots \sigma_m \in \Sigma^m$. That is, $n+1 \leq m \leq n + (n-2)$ and $\sigma_1 \dots \sigma_n \in L$. This implies $\sigma_i \in \Sigma_{i-1,i}$ for each $i \in \{1, \dots, n\}$ but provides no information about the σ_i with $i \in \{n+1, \dots, m\}$. Therefore, we construct DFAs rejecting every such extension not confirming to a certain structure. This structure will be key to the further DFA constructions.

For a word $w \in \Sigma^*$, let $\mathcal{A}^!_w$ be the DFA rejecting exactly the words containing w as a subsequence. Clearly, the following holds:

▶ **Lemma 3.5.** *Let $w \notin L, |w| = n$. Then $\mathcal{A}^!_w \in \alpha(\mathcal{A})$ holds.* ⌟

With the DFAs $\mathcal{A}^!_w$ for every $w \notin L, |w| = n$ in hand, we only have to consider extensions of words $u \in L, |u| = n$ with a maximum length of $n + (n-2)$ for which every subsequence of length n is in L.

Let $w = \sigma_1 \dots \sigma_m$ be an extension satisfying these conditions. We construct a DFA $\tilde{\mathcal{A}}_w \in \alpha(\mathcal{A})$ rejecting w. We utilize the rejecting state q_d and define $\tilde{\mathcal{A}}_w = (\tilde{Q}_w, \Sigma, q_0, \tilde{\delta}_w, \tilde{F}_w)$ with $\tilde{Q} = \{q_0, \dots, q_n\}$, $\tilde{F}_w = \tilde{Q}_w \setminus \{q_d\}$ and $\tilde{\delta}_w(q_0, w) = q_d$. Further, we have $\tilde{\delta}_w(q_0, v) = q_d$ for a $v \in \Sigma^*$ only if $\delta(q_0, v) \in \{q_d, q_{n+1}\}$, ensuring $\tilde{\mathcal{A}}_w \in \alpha(\mathcal{A})$. In order to utilize q_d in this manner, the DFA $\tilde{\mathcal{A}}_w$ simulates the behavior of $\mathcal{A}$ for the states $q_0, \dots, q_{d-1}$. The task then is to select the transitions of states $q_d, \dots, q_n$.

If $|\sigma_{d+1} \dots \sigma_m|_{\sigma_m} \leq n - d$ the DFA $\tilde{\mathcal{A}}_w$ can simply advance for occurrences of σ_m and the first $n - d - |\sigma_{d+1} \dots \sigma_{m-1}|_{\sigma_m}$ occurrences of letters unequal to σ_m. Thus, we only have to consider the case $|\sigma_{d+1} \dots \sigma_m|_{\sigma_m} > n - d$.

If $\sigma_{n+1} \neq \sigma_m$ the DFA $\tilde{\mathcal{A}}_w$ can advance for each letter in Σ, ensuring $\tilde{\delta}_w(q_d, \sigma_{d+1} \dots \sigma_n) = q_n$. Further, we can define $\tilde{\delta}_w(q_n, \sigma_{n+1}) = q_{n-[(m-1)-(n+2)+1]}$ and $\tilde{\delta}_w(q_n, \sigma_m) = q_d$. Note that $|\sigma_{n+2} \dots \sigma_{m-1}| = (m-1) - (n+2) + 1$. Since every subsequence of w of length n is in L, we have $\tilde{\delta}_w(q_{n-[(m-1)-(n+2)+1]}, \sigma_{n+2} \dots \sigma_{m-1}) = q_n$.

The case $\sigma_{n+1} = \sigma_m$ is more complex and needs a further case distinction, but the idea used above of circling back after reading an appropriate prefix can be employed again.

Lemma 3.6 summarizes these ideas:

▶ **Lemma 3.6.** *Let $w \in \Sigma^*$ with $|w| > n$ such that $w \in \mathcal{L}(\mathcal{A}_v^!)$ for each $v \notin L, |v| = n$ and $w \in \bigcap_{\sigma \in \Sigma} \mathcal{L}(\mathcal{A}_{\sigma,i_\sigma})$, where for each $\sigma \in \Sigma$ it is $i_\sigma = \max(\{i \in \{1, \dots, n\} \mid \sigma \notin \Sigma_{i-1,i}\})$. Then there exists a DFA $\tilde{\mathcal{A}}_w \in \alpha(\mathcal{A})$ rejecting w.* ⌟

Lemmas 3.3–3.6 imply Claim 3.2 (c). To be more precise, we have $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_0) \cap \mathcal{L}(\hat{\mathcal{A}}_d) \cap \bigcap_{m=1}^{n-1} \bigcap_{\underline{i} \in I_m} \mathcal{L}(\mathcal{A}_{\underline{i}}) \cap \bigcap_{\sigma \in \Sigma} \mathcal{L}(\mathcal{A}_{\sigma,i_\sigma}) \cap \bigcap_{w \in X^!} \mathcal{L}(\mathcal{A}_w^!) \cap \bigcap_{w \in \tilde{X}} \mathcal{L}(\tilde{\mathcal{A}}_w)$, where $X^! = \{w \in \Sigma^n \mid w \notin L\}$ and $\tilde{X}$ is the set of all extensions w of words $u \in L, |u| = n$ with $|w| \leq n + (n-2)$ for which every subsequence of length n is in L. This proves the compositionality of $\mathcal{A}$ and thereby Claim 3.2 (c).

3.2 Linear safety ADFAs

Next, we consider Claim 3.2 (d) and (e). For (d) we argue that $\mathcal{A}$ is composite if it has the CEP, even if $\mathcal{A}$ is a safety DFA, which makes circling back impossible. For (e) we argue that $\mathcal{A}$ is prime if it is a safety DFA and it does not have the CEP.

First, we consider (d). We assume that $\mathcal{A}$ has the CEP and argue that this implies compositionality. Note that we can reuse the DFAs $\mathcal{A}_0$ and $\mathcal{A}_{\underline{i}}$, while $\hat{\mathcal{A}}_d$ is not needed. This again leaves the task of rejecting the extensions of words $w \in L, |w| = n$. But, since for every such word $w = \sigma_1 \dots \sigma_n$ there now exist $i \in \{0, \dots, n-2\}, l \in \{2, \dots, n-i\}$ such that $\delta(q_0, \sigma_1 \dots \sigma_i \sigma_{i+l} \dots \sigma_n) \in \{q_n, q_{n+1}\}$, we can construct a DFA $\mathcal{A}_{i,l} \in \alpha(\mathcal{A})$ rejecting every extension of w.

The DFA $\mathcal{A}_{i,l}$ possesses states $q_0, \dots, q_{i+l-2}, q_{i+l}, \dots, q_{n+1}$. It simulates the behavior of $\mathcal{A}$ for states $q_0, \dots, q_{i-1}$, redirecting transitions $q_j \to q_{i+l-1}$ to q_i. From q_i it directly advances to q_{i+l} if a letter in $\bigcup_{j=i+l}^{n+1} \Sigma_{i,j}$ is read, otherwise it advances to q_{i+1}. The states $q_i, \dots, q_{i+l-2}$ form a loop. For states $q_{i+l}, \dots, q_n$, every transition leads to the direct successor state. The state q_{n+1} is a rejecting sink.

It is shown in the appendix that every extension of w is rejected by $\mathcal{A}_{i,l}$, where i is the largest possible value belonging to w, and that $\mathcal{A}_{i,l} \in \alpha(\mathcal{A})$. Thus, $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_0) \cap \bigcap_{m=1}^{n-1} \bigcap_{\underline{i} \in I_m} \mathcal{L}(\mathcal{A}_{\underline{i}}) \cap \bigcap_{i=0}^{n-2} \bigcap_{l=2}^{n-i} \mathcal{A}_{i,l}$ holds, proving the compositionality of $\mathcal{A}$ and thus (d).

Next, we consider (e) and assume that $\mathcal{A}$ is a safety DFA and does not have the CEP. Thus, there is a $w = \sigma_1 \dots \sigma_n$ such that $\delta(q_0, \sigma_1 \dots \sigma_i \sigma_{i+l} \dots \sigma_n) \notin \{q_n, q_{n+1}\}$ holds for every $i \in \{0, \dots, n-2\}, l \in \{2, \dots, n-i\}$. This implies the existence of a letter $\sigma \in \Sigma_{n-1,n}$ with $\sigma \notin \Sigma_{j,n+1}$ for every $j \in \{0, \dots, n-1\}$. We show in the appendix that $w\sigma$ is a primality witness of $\mathcal{A}$, thus proving the primality of $\mathcal{A}$ and thereby (e).

This completes our discussion of Claim 3.2 (a)-(e). Since they imply Theorem 3.1, we have characterized the compositionality of ADFAs and thereby of finite languages.

4 Complexity of Prime-DFA$_{\text{fin}}$

After characterizing the compositionality of ADFAs and thereby of finite languages in Section 3, we now analyze the complexity of Prime-DFA$_{\text{fin}}$. We argue:

▶ **Theorem 4.1.** *The problem Prime-DFA$_{fin}$ is NL-complete. The NL-completeness holds true even when restricting Prime-DFA$_{fin}$ to DFAs with at most two letters.* ⌟

Algorithm 1 NL-algorithm for PRIME-DFA$_{\text{fin}}$.

Require: DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ with $Q = \{q_0, \dots, q_m\}$ recognizing a finite language L.
Ensure: The DFA $\mathcal{A}$ is prime.

1: Accept if $L = \emptyset$.
2: $c \leftarrow 0$
3: **for all** $i \in \{0, \dots, m\}$ **do**
4: **if** q_i is unreachable **then**
5: $c \leftarrow c + 1$
6: **else**
7: $j \leftarrow 0$, $b \leftarrow \mathit{true}$
8: **while** $j \leq i - 1$ **and** b **do**
9: **if** q_j is reachable and $\mathcal{L}(\mathcal{A}^{q_i}) = \mathcal{L}(\mathcal{A}^{q_j})$ **then**
10: $c \leftarrow c + 1$
11: $b \leftarrow \mathit{false}$
12: **end if**
13: $j \leftarrow j + 1$
14: **end while**
15: **end if**
16: **end for**
17: $n \leftarrow (m + 1) - c - 2$
18: Choose nondeterministically a word $w \in \Sigma^n$. Reject if $w \notin L$.
19: Choose nondeterministically a letter $\sigma \in \Sigma$. Accept if $\sigma^n \in L$.
20: **for all** $i \in \{0, \dots, m\}$ where q_i is not unreachable **do**
21: Reject if $q_i \notin F$ and $\mathcal{L}(\mathcal{A}^{q_i}) \neq \emptyset$.
22: **end for**
23: **for all** $x \in \{1, \dots, n\}$ **do**
24: Choose nondeterministically a word $w = \sigma_1 \dots \sigma_n \in \Sigma^n$. Reject if $w \notin L$.
25: **for all** $i \in \{0, \dots, n-2\}, l \in \{2, \dots, n-i\}$ with $i + l = x$ **do**
26: Choose nondeterministically a word $w' = \sigma'_1 \dots \sigma'_n \in \Sigma^n$ with $\sigma'_{i+l} = \sigma_x$ and a word $v \in \Sigma^+$. Reject if $w' \notin L$ or if $\sigma'_1 \dots \sigma'_i \sigma'_{i+l} \dots \sigma'_n v \notin L$.
27: **end for**
28: **end for**
29: Accept.

We begin by arguing that PRIME-DFA$_{\text{fin}}$ is in NL, providing an NL-algorithm for PRIME-DFA$_{\text{fin}}$ with Algorithm 1. The algorithm accepts in line 1 if the given DFA $\mathcal{A}$ recognizes the empty language. Then lines 2-18 ensure that the minimal DFA belonging to $\mathcal{A}$ is linear. Lines 19-22 ensure that $\mathcal{A}$ is accepted if a letter $\sigma \in \Sigma$ with $\sigma^n \in L$ exists or else that $\mathcal{A}$ is rejected if it is not a safety DFA. Finally, in lines 23-29 the CEP is checked for $\mathcal{A}$.

The NL-hardness of PRIME-DFA$_{\text{fin}}$ can be proven by L-reducing STCONDAG to PRIME-DFA$_{\text{fin}}$, where STCONDAG is the restriction of STCON to acyclic graphs. The L-reduction is similar to the L-reduction of STCON to the emptiness problem for DFAs.

5 Finite Languages under Different Notions of Compositionality

So far, we have only considered $\cap$-compositionality. Now we will define two further notions of compositionality and characterize the compositionality of finite languages for these notions.

▶ **Definition 5.1.** For $k \in \mathbb{N}_{\geq 1}$, a DFA $\mathcal{A}$ is *k-$\cup$-decomposable* (*k-DNF-decomposable*) if there exist DFAs $\mathcal{A}_1, \ldots, \mathcal{A}_t$ ($\mathcal{A}_{1,1}, \ldots, \mathcal{A}_{1,t_1}, \ldots, \mathcal{A}_{s,1}, \ldots, \mathcal{A}_{s,t_s}$) with $\mathcal{L}(\mathcal{A}) = \bigcup_{i=1}^{t} \mathcal{L}(\mathcal{A}_i)$ ($\mathcal{L}(\mathcal{A}) = \bigcup_{i=1}^{s} \bigcap_{j=1}^{t_i} \mathcal{L}(\mathcal{A}_{i,j})$) and $|\mathcal{A}_i| < k$ for every i ($|\mathcal{A}_{i,j}| < k$ for every pair i, j). The further concepts introduced in Definition 2.1 are defined analogously. ⌟

In [10], it is correctly remarked that many results for $\cap$-compositionality can be trivially transferred to $\cup$-compositionality. For example, the complexity boundaries for Prime-DFA established in [10] also hold for $\cup$-compositionality. This does not hold true for results concerning language fragments that are not closed under complement. In particular, the complement language of a finite language is not finite, but co-finite. Thus, characterizing the $\cup$-compositionality of finite languages is equivalent to characterizing $\cap$-compositionality of co-finite languages.

Also in [10], the notion of compositionality allowing both union and intersection is suggested. Note that DNF-compositionality enforces a structure similar to a disjunctive normal from, but is as strong as unrestricted union-intersection compositionality. It is correctly remarked in [10] that union-intersection compositionality - and thus, DNF-compositionality - is strictly stronger than $\cap$-compositionality. Obviously, it is also strictly stronger than $\cup$-compositionality. It is less obvious whether languages exist that are DNF-composite, but are neither $\cap$- nor $\cup$-composite. We will see that there are finite languages witnessing this.

The following result characterizes the $\cup$- and DNF-compositionality of finite languages:

▶ **Theorem 5.2.** *Consider a minimal ADFA $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ recognizing a non-empty language. Let $n \in \mathbb{N}$ be the length of the longest word in $\mathcal{L}(\mathcal{A})$. The following assertions hold:*

(i) *$\mathcal{A}$ is $\cup$-prime iff $\mathcal{A}$ is linear.*

(ii) *$\mathcal{A}$ is DNF-prime iff $\mathcal{A}$ is linear and there exists a $\sigma \in \Sigma$ with $\sigma^n \in \mathcal{L}(\mathcal{A})$.* ⌟

These conditions are similar to the conditions in Theorem 3.1, but much simpler. Let $\mathcal{A}$ and n be as required. It is easy to show $\cup$- and DNF-compositionality if $\mathcal{A}$ is not linear.

The proof of $\cup$-primality if $\mathcal{A}$ is linear relies on the observation that every minimal DFA $\mathcal{B}$ with $\mathcal{L}(\mathcal{B}) \subseteq \mathcal{L}(\mathcal{A})$ and $\mathrm{ind}(\mathcal{B}) < \mathrm{ind}(\mathcal{A})$ has to have a rejecting sink. From this follows that no such DFA $\mathcal{B}$ can accept a word $w \in \mathcal{L}(\mathcal{A}), |w| = n$. Thus, $\mathcal{A}$ is $\cup$-prime.

If $\mathcal{A}$ is linear and there exists no $\sigma \in \Sigma$ with $\sigma^n \in \mathcal{L}(\mathcal{A})$ the DNF-compositionality of $\mathcal{A}$ follows from [10, Example 3.2]. On the other hand, if $\mathcal{A}$ is linear and there exists a $\sigma \in \Sigma$ with $\sigma^n \in \mathcal{L}(\mathcal{A})$ DNF-primality can be shown by adapting the proof of Claim 3.2 (b).

As mentioned, Theorems 3.1 and 5.2 immediately imply:

▶ **Theorem 5.3.** *There exists a finite language that is DNF-composite but $\cap$- and $\cup$-prime.* ⌟

To summarize, Theorems 3.1 and 5.2 characterize the $\cap$-, $\cup$- and DNF-compositionality of ADFAs and thus of finite languages. Obviously, this characterizes the $\cap$-, $\cup$- and DNF-compositionality of co-finite languages as well. The results further imply the existence of languages that are DNF-composite but $\cap$- and $\cup$-prime.

6 2Minimal-DFA and S-Prime-DFA

We defined compositionality using the index of the given DFA. Thus, the compositionality of a DFA $\mathcal{A}$ is a characteristic of $\mathcal{L}(\mathcal{A})$. Slightly changing the definition, using the size instead of the index, turns compositionality of $\mathcal{A}$ into a characteristic of $\mathcal{A}$ itself. It is interesting to analyze the effects of this change, which results in the notion of S-compositionality.

Many results known for compositionality hold for S-compositionality as well. The characterization of finite languages in Section 3 and other results concerning language fragments [10, 8, 9] are valid with only minor technical modifications. In fact, [8, 9]

already implicitly used S-compositionality instead of compositionality without discussing the differences. The upper complexity boundary of Prime-DFA holds for S-Prime-DFA as well. But the known lower boundary, the NL-hardness of Prime-DFA, cannot simply be adapted for S-Prime-DFA. The lower boundary for S-Prime-DFA is connected to Minimal-DFA, since non-minimal DFAs are trivially S-composite. Note that Prime-DFA is connected to the emptiness problem for DFAs in a similar manner [10].

We begin by discussing Minimal-DFA, proving the NL-hardness of 2Minimal-DFA. Then we formally introduce S-compositionality and prove the NL-hardness of the restriction 2S-Prime-DFA and thereby of S-Prime-DFA as well. We also prove the NL-hardness of the restriction 2Prime-DFA, so far only known for the unrestricted problem Prime-DFA.

6.1 NL-hardness of 2Minimal-DFA

As mentioned, the NL-hardness and thus NL-completeness of kMinimal-DFA for $k \in \mathbb{N}_{\geq 3}$ is folklore, while the NL-hardness of 2Minimal-DFA appears to be open. We prove:

▶ **Theorem 6.1.** *The problem 2Minimal-DFA is NL-hard and thus NL-complete.* ⌟

The NL-hardness of 3Minimal-DFA can be proven by L-reducing 2STCON to 3Minimal-DFA. This known reduction uses an additional letter and cannot be used to prove the NL-hardness of 2Minimal-DFA. We give an L-reduction of 2STCON not using an additional letter, proving the NL-hardness and thus the NL-completeness of 2Minimal-DFA.

Let (G, s, t) be an input for 2STCON. That is, $G = (V, E)$ is a graph with a maximum outdegree of two and $s, t \in V$ are nodes of G. We construct a DFA $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ with $\Sigma = \{0, 1\}$, which is minimal iff there exists a path in G from s to t. If $s = t$ such a path exists trivially and we can construct the minimal DFA for the empty language. Thus, we only have to consider the case $s \neq t$. W.l.o.g. we assume $V = \{0, \ldots, n-1\}$ and $s = 0, t = n-1$.

Let $\mathcal{A}' = (Q', \Sigma, 0, \delta', F')$ be the DFA constructed out of G in the usual manner, that is, by turning nodes into states, edges into transitions, setting the state 0 as the initial state and $n-1$ as the only accepting state. For $\mathcal{A}$, we introduce the new states $p_0, \ldots, p_{n-1}$, called p-states, the new states $q_0, \ldots, q_{n-1}$ and q'_0, called q-states, and for each $i \in Q'$ the states i'_0, i'_1, i_0, i_1. We call the states i, i'_0, i'_1, i_0, i_1 for $i \in Q'$ v-states. We say that states $p_i, q_i, i, i'_0, i'_1, i_0, i_1$ for an $i \in Q'$ are located on the same layer. Figure 2 specifies the DFA $\mathcal{A}$ constructed for the L-reduction. We now discuss the key ideas of this construction.

First, note that the idea of the p- and q-states is similar to the known L-reduction of 2STCON to 3Minimal-DFA. The p-states are used to access every state in Q, thus avoiding unreachable states. The q-states are used to allow the return to 0 from every state.

Second, we cannot use an additional letter to switch from p_i to i to q_i. Thus, letter 1 is used to leave the p-states and to exit q_0 to state 0. Letter 0 is used to advance to the next layer in both the p- and q-states. To allow switching from the v-states to the q-states, we introduce for each $i \in Q'$ a component consisting of i and the two branches i'_0, i_0 and i'_1, i_1. The states i'_0, i'_1 are waiting states used to prove the non-equivalence of q- and v-states. The states i_0, i_1 implement on the one hand the original transitions in $\mathcal{A}'$, that is, $\delta(i_j, j) = \delta'(i, j)$, and on the other hand the transitions into the q-states, that is, $\delta(i_j, 1-j) = q_i$.

Third, an extra q-state q'_0 is introduced, which is only directly accessible from q_0. Without q'_0 the situation $\delta(1_1, 1) = 0 = \delta(q_0, 1)$ and $\delta(1_1, 0) = q_1 = \delta(q_0, 0)$ would be possible, immediately implying the non-minimality of $\mathcal{A}$. The introduction of q'_0 solves this problem.

Note that there is a path from 0 to $n-1$ in $\mathcal{A}$ iff there is such a path in G. Using this it follows that $\mathcal{A}$ is minimal iff there exists a path from 0 to $n-1$ in G. Since $\mathcal{A}$ can obviously be constructed in logarithmic space, the given construction is indeed an L-reduction of 2STCON to 2Minimal-DFA. Consequently, 2Minimal-DFA is NL-hard.

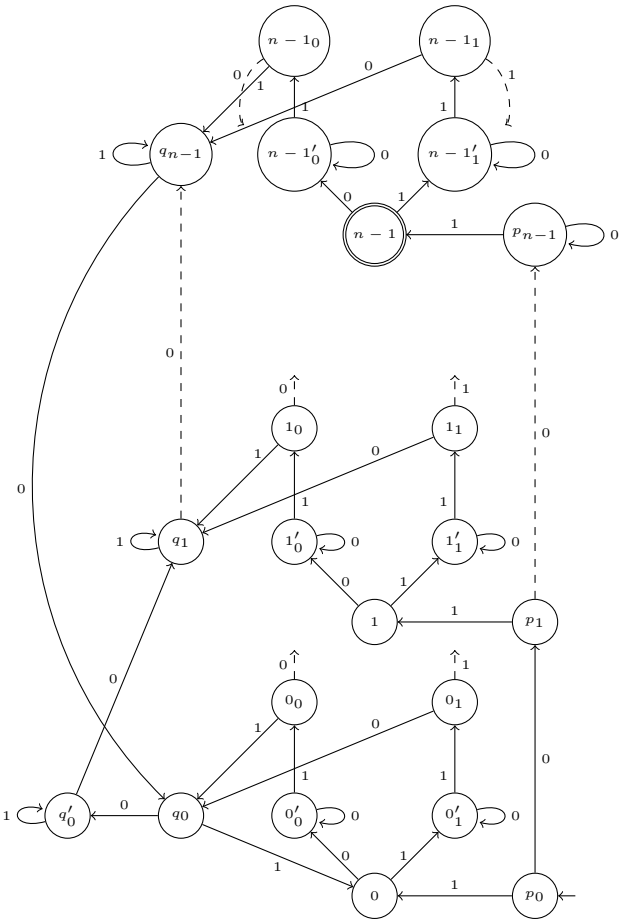


Figure 2 DFA $\mathcal{A}$ constructed for the L-reduction of 2STCON to 2Minimal-DFA. The j-transitions exiting states of the form i_j are only indicated.

6.2 Complexity of S-Prime-DFA

We end our discussion by using the construction presented in Section 6.1 to establish complexity boundaries for S-Prime-DFA. First, we define the notion of S-compositionality.

▶ **Definition 6.2.** A DFA $\mathcal{A}$ is *S-composite* if there is a $k \in \mathbb{N}_{\geq 1}, k < |\mathcal{A}|$ such that $\mathcal{A}$ is k-decomposable. Otherwise, $\mathcal{A}$ is *S-prime*. ⌟

We denote the problem of deciding S-primality for a given DFA with S-Prime-DFA and the restriction of S-Prime-DFA to DFAs with at most $k \in \mathbb{N}_{\geq 2}$ letters with kS-Prime-DFA.

Note that the proof used in [10] to show that Prime-DFA is in ExpSpace is applicable for S-Prime-DFA with only slight modifications. Next, note that the L-reduction of the emptiness problem for DFAs to Prime-DFA used in [10] to prove the NL-hardness of Prime-DFA relies on the fact that every DFA recognizing the empty language is prime. Thus, it is not easily adaptable for S-Prime-DFA. Instead, the NL-hardness of 2S-Prime-DFA is shown by using a reduction from 2STCON, which adapts the construction outlined in Section 6.1. We get:

▶ **Theorem 6.3.** *The problems S-Prime-DFA and kS-Prime-DFA for $k \in \mathbb{N}_{\geq 2}$ are in ExpSpace and they are NL-hard.* ⌟

Further, we denote with kPrime-DFA the restriction of Prime-DFA to DFAs with at most $k \in \mathbb{N}_{\geq 2}$ letters and remark that the results presented in [10] can be expanded to:

▶ **Theorem 6.4.** *The problems Prime-DFA and kPrime-DFA for $k \in \mathbb{N}_{\geq 2}$ are in ExpSpace and they are NL-hard.* ⌟

This ends our discussion of the complexity of S-Prime-DFA and its restrictions, in which we have applied the construction outlined in Section 6.1 to prove NL-hardness.

7 Discussion

We studied the intersection compositionality, also denoted with $\cap$-compositionality, of regular languages. We added to the existing line of research focusing on fragments of the regular languages by analyzing the $\cap$-compositionality of ADFAs and thereby of finite languages. This research was in part motivated by existing results concerning the concatenation compositionality of finite languages.

We completely characterized the $\cap$-compositionality of ADFAs and thus finite languages. Using this characterization we proved the NL-completeness of Prime-DFA$_{\text{fin}}$. Thus, finite languages are significantly easier to handle under $\cap$-compositionality than under concatenation compositionality, where the respective primality problem for finite languages is NP-hard [15].

With notions of compositionality using union and both union and intersection already suggested in [10], we formally introduced the notions of $\cup$- and DNF-compositionality. We characterized the $\cup$- and DNF-compositionality of finite languages, which proved to be far simpler than the characterization of $\cap$-compositionality. These results also imply the characterization of the $\cap$-, $\cup$- and DNF-compositionality of co-finite languages.

This suggests that the key feature of finite languages regarding compositionality is not the finiteness of the languages per se, but rather the existence of only finitely many meaningfully different runs of the respective DFAs, a feature finite languages have in common not only with co-finite languages, but also with languages whose minimal DFAs allow for cycles in both accepting and rejecting sinks. A logical next step would therefore be the characterization of the compositionality of these DFAs.

We also note that in our proofs we employed $\cap$-compositionality results concerning a different language fragment, namely co-safety DFAs, studied in [10]. This suggests the possibility of employing the results concerning finite languages in future analyses and stresses the usefulness of working with language fragments. We provided one application of the results concerning finite languages by using them to prove the existence of a language that is DNF-composite but $\cap$- and $\cup$-prime.

Furthermore, we presented a proof of the NL-hardness and thereby NL-completeness of the basic problem 2Minimal-DFA. While the NL-hardness of kMinimal-DFA for $k \in \mathbb{N}_{\geq 3}$ is folklore, this result appears to be new.

We utilized this result to establish the known complexity boundaries of Prime-DFA for the here newly introduced problem S-Prime-DFA. We extended these results to the restrictions kPrime-DFA and kS-Prime-DFA for $k \in \mathbb{N}_{\geq 2}$.

While it is interesting that a slight variation in the definition of $\cap$-compositionality, which does not touch the validity of most results, requires a whole new approach to establish the known lower complexity boundary, the big task of closing the doubly exponential complexity gap for Prime-DFA still remains. And now, this gap exists for S-Prime-DFA as well.

Therefore, with the analysis of language fragments, further notions of compositionality, and the complexity gaps for Prime-DFA and S-Prime-DFA, there is still need for further research.

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 Sang Cho and Dung T. Huynh. The parallel complexity of finite-state automata problems. *Inf. Comput.*, 97(1):1–22, 1992. doi:10.1016/0890-5401(92)90002-W.

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 Henning Fernau and Markus Holzer. Personal communication.

5 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.

6 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.

7 Neil Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999. doi:10.1007/978-1-4612-0539-5.

8 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.

9 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.

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

11 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 1–6. IEEE, 2022. doi:10.34727/2022/isbn.978-3-85448-053-2_39.

12 Wim Martens, Matthias Niewerth, and Thomas Schwentick. Schema design for XML repositories: complexity and tractability. In Jan Paredaens and Dirk Van Gucht, editors, *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*, pages 239–250. ACM, 2010. doi:10.1145/1807085.1807117.

13 Alexandru Mateescu, Arto Salomaa, and Sheng Yu. Factorizations of languages and commutativity conditions. *Acta Cybern.*, 15(3):339–351, 2002. URL: https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3583.

14 Arto Salomaa and Sheng Yu. On the decomposition of finite languages. In Grzegorz Rozenberg and Wolfgang Thomas, editors, *Developments in Language Theory, Foundations, Applications, and Perspectives, Aachen, Germany, 6-9 July 1999*, pages 22–31. World Scientific, 1999. doi:10.1142/9789812792464_0003.

15 Philip Sieder. A lower bound for primality of finite languages. *CoRR*, abs/1902.06253, 2019. arXiv:1902.06253.

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

17 Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification (preliminary report). 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.

18 Wojciech Wieczorek. An algorithm for the decomposition of finite languages. *Log. J. IGPL*, 18(3):355–366, 2010. doi:10.1093/jigpal/jzp032.