Decomposition of Automata Recognizing Ideals

Mathias Berry
Université Marie et Louis Pasteur, CNRS, institut FEMTO-ST, F-25000 Besançon, France

Pierre-Cyrille Héam
Université Marie et Louis Pasteur, CNRS, institut FEMTO-ST, F-25000 Besançon, France

Ismaël Jecker
Université Marie et Louis Pasteur, CNRS, institut FEMTO-ST, F-25000 Besançon, France

Abstract

Minimizing the size of finite automata is a fundamental problem in theoretical computer science. Beyond standard minimization, further reductions can be achieved by decomposing an automaton into smaller components whose languages combine via intersection or union to recover the original language. However, in general, no polynomial-time algorithm is known for computing such decompositions.

In this paper, we focus on automata that recognize ideals, that is, languages at level 1/2 in the Straubing–Thérien hierarchy. Equivalently, these languages are expressible as a finite union of languages of the form $\Sigma^* a_1 \Sigma^* \dots \Sigma^* a_n \Sigma^*$ where Σ is an alphabet and a_i are letters of Σ. We show that the two problems of deciding whether an automata recognizing an ideal can be decomposed into an intersection or a union of smaller automata are decidable in NL. Moreover, we provide a polynomial-time algorithm that computes a decomposition into an intersection, if one exists, while ensuring that the resulting components also recognize ideal languages.

2012 ACM Subject Classification Theory of computation → Regular languages

Keywords and phrases Finite state automata, decomposition, Shuffle ideals

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2026.16

Related Version *Extended Version*: `https://arxiv.org/abs/2604.25619` [4]

Funding *Ismaël Jecker*: This research was partially funded by the Agence Nationale de la Recherche (ANR) grant ANR-25-CE48-2447 FAVOR.

1 Introduction

Finite automata are fundamental tools in computer science, with applications ranging from text processing to system modelling and machine learning, thanks to their strong algorithmic properties. In contexts such as model verification, automata can grow prohibitively large, limiting applicability to industrial-scale problems. Significant research has focused on automata minimization (see for instance [11, 15, 17, 5]), yet an alternative approach consists in decomposing complex systems into simpler components [7].

In this context, the notion of deterministic automaton decomposition was introduced by O. Kupferman and J. Mosheiff [22]. In this paper, we push this line of research further and distinguish two types of decompositions. An intersection [resp. union] decomposition of an automaton $\mathcal{A}$ is a finite collection of smaller automata $\mathcal{A}_1, \dots, \mathcal{A}_n$ such that the intersection [resp. union] of their recognized languages coincides with that $\mathcal{A}$. When no such decomposition exists, $\mathcal{A}$ is called prime. Such decompositions reduce the complexity of models or specifications by splitting them into simpler components. For instance, verifying that a system satisfies a specification that has an intersection decomposition reduces to verifying whether the system satisfies each component individually. This approach can improve computational efficiency, for instance through distributed computation [29, 3, 6].

We focus on decomposition problems for the class of *ideal languages*, that occupy level 1/2 in the Straubing–Thérien hierarchy [8, 30, 32]. Formally, this the class of languages expressible as a finite union of languages of the form $\Sigma^* a_1 \Sigma^* \dots \Sigma^* a_n \Sigma^*$ where Σ is a finite alphabet and a_i are letters. Ideal languages are also known as *shuffle ideals* [9], or simply *ideals*, and they arise naturally in the study of word combinatorics [10] and the formal verification of systems with communication channels [1]. They also correspond to the languages definable by the existential fragment of FO($<$) [2], and generate, via Boolean combinations, the class of *piecewise testable* languages [26, 21, 20, 23]. Moreover, ideal languages are upward closed under the subword relation, a property that has been studied in relation to state complexity [19, 24] and combinatorial properties of words [18]. This property underlies the relevance of the class of ideals.

Contributions. We study both intersection and union decompositions for automata recognizing ideals, focusing on the complexity of deciding primality and on the effective construction of decompositions. Note that these two problems are interreducible in general, since an automaton is prime for intersection if and only if its complement is prime for union. However, ideal languages are not closed by complement, thus these two problems are different in that case. Our results first address intersection decomposition.

▶ **Theorem 1.** *Given a minimal automaton $\mathcal{A}$ recognizing an ideal, deciding whether $\mathcal{A}$ is prime for intersection is in NL.*

▶ **Theorem 2.** *Let $\mathcal{A} = (S, \Sigma, \iota, F, \delta)$ be an automaton recognizing an ideal. Determining whether $\mathcal{A}$ has an intersection decomposition can be done in time $\mathcal{O}(|\mathcal{A}||\Sigma|)$, and if so, such a decomposition can be constructed in time $\mathcal{O}(|\mathcal{A}||\Sigma|^2)$.*

We then turn to union decomposition, and prove a symmetric complexity result.

▶ **Theorem 3.** *Given a minimal automaton $\mathcal{A}$ recognizing an ideal, deciding whether $\mathcal{A}$ is prime for union is in NL.*

Related work. The general problem of intersection decomposition was introduced in the seminal work of Orna Kupferman and Jonathan Mosheiff [22], which defines prime automata, i.e., automata that are not decomposable, and shows that deciding whether an automaton is prime for intersection is in EXPTIME. To date, no better algorithm is known, and the problem is only known to be NP-hard [28]. Subsequent work has focused on specific subclasses of automata. For automata over a single-letter alphabet, intersection primality is decidable in NL [13], and for *permutation* automata, where each input letter induces a bijection over the state set, intersection primality is decidable in NP [14].

Permutation and *aperiodic* automata play complementary roles in the algebraic study of regular languages: by the Krohn–Rhodes theorem, every regular language can be decomposed into a cascade of permutation and aperiodic components. With the permutation case understood, the next step is to tackle aperiodic automata. Prior work in this direction has addressed acyclic automata, providing a polynomial time criterion for their intersection decomposability [27]. Moreover, for *almost-acyclic* automata, that might contain sink states with self loops, the problem is known to be NP-complete [28]. In this paper, we focus on automata recognizing ideals, another subclass of aperiodic automata.

Structure of the paper. The paper is organized as follows:

- Section 2 recalls standard notions and notations for finite automata and ideals.
- Section 3 studies intersection decomposition of *non-linear* automata recognizing ideals, i.e., automata whose state sets cannot be totally ordered compatibly with the transition function. We prove that every automaton of this form is decomposable into at most $|\Sigma|$ smaller automata, each recognizing an ideal, and that such a decomposition can be computed in polynomial time (Theorem 6).
- Section 4 focuses on intersection decomposition of *linear* automata recognizing ideals, and provides a primality criterion. Moreover, we show that every decomposable automaton of this form is decomposable into two smaller automata, each recognizing a ideal, and that such a decomposition can be computed in polynomial time (Theorem 15).
- Section 5 addresses union decompositions, and provides a primality criterion (Theorem 21).
- Finally, Section 6 combines the results of the two previous sections to establish that intersection primality of automata recognizing ideals is decidable in NL (Theorem 1) and computable in polynomial time (Theorem 2).

All results are proven; proofs omitted from the main text are deferred to the extended version [4].

2 Preliminaries

Mathematics. For all $n, m \in \mathbb{N}$, we denote by $[\![n, m]\!]$ the set $\{k \in \mathbb{N} \mid n \leqslant k \text{ and } k \leqslant m\}$. For every set E and for every subset F of E, the subset $E \setminus F$ is just denoted $\overline{F}$ when there is no ambiguity on E.

Ordered relations. For every set S, a binary relation $\leqslant$ on S is a *partial order* if $\leqslant$ is reflexive, transitive and antisymmetric. A partial order $\leqslant$ is said to be *total* if for all $q, r \in S$ either $q \leqslant r$ or $r \leqslant q$. For an order $\leqslant$ on S, an element x of some subset $X \subseteq S$ is a *minimal* [resp. *maximal*] element of X if for all $y \in X$, $y \leqslant x$ implies $x = y$ [resp. $x \leqslant y$ implies $x = y$]. If x is a minimal [resp. maximal] element of X such that every $y \in X$ satisfies $x \leqslant y$ [resp. $y \leqslant x$], then x is called the *least element* [resp. *greatest element*] of X. Note that if a subset X has a least [resp. greatest] element, then it is unique and denoted $\min X$ [resp. $\max X$]. Note too that any finite subset X admits at least one minimal and one maximal element.

Words and languages. An *alphabet* is a finite set whose elements are called *letters*. A *finite word* over an alphabet Σ is a finite sequence of letters of Σ. The empty sequence is called the *empty word* and is denoted ε. A word $(w(i))_{i \in [\![1,n]\!]}$ is classically written $w_1 \dots w_n$. Its length n is denoted $|w|$. The empty word is the unique word of length 0. For every alphabet Σ, the set of all finite words over Σ is denoted Σ^*. Subsets of Σ^* are called *languages* over Σ.

Shuffle product. Let Σ be an alphabet and let $u, v \in \Sigma^*$ be words of respective length m and n. The *shuffle* of u and v, denoted by $u \sqcup\!\sqcup v$, is the set of words $w \in \Sigma^*$ of length $m + n$ such that there exist two strictly increasing sequences $(i(k))_{k \in [\![1,m]\!]}$ and $(j(k))_{k \in [\![1,n]\!]}$ whose images partition $[\![1, m + n]\!]$ and such that $u_k = w_{i(k)}$ for all $k \in [\![1, m]\!]$ and $v_\ell = w_{j(\ell)}$ for all $\ell \in [\![1, n]\!]$. This notion extends to languages in the usual way: for two languages $L_1, L_2 \subseteq \Sigma^*$, the *shuffle* of L_1 and L_2 is the language $L_1 \sqcup\!\sqcup L_2 = \bigcup_{u \in L_1, v \in L_2} u \sqcup\!\sqcup v$. For all $u, v \in \Sigma^*$, if $v \in \{u\} \sqcup\!\sqcup \Sigma^*$ we say that u is a *sub-word* of v and that v is an *upper-word* of u. Every language $L \subseteq \Sigma^*$ satisfies $L \subseteq L \sqcup\!\sqcup \Sigma^*$. A language is an *ideal* if $L \sqcup\!\sqcup \Sigma^* = L$, or, equivalently, if for all $u \in L$ every upper-word of u also belongs to L.

Finite automata. A *finite state automaton* is a 5-tuple $\mathcal{A} = (S, \Sigma, \iota, F, \delta)$ where S is a finite set of *states*, Σ is an alphabet, $\iota \in S$ is the *initial state*, $F \subseteq S$ is the set of *accepting (or final) states* and $\delta \subseteq S \times \Sigma \times S$ is the set of *transitions*. If for all $q \in S$ and for all $a \in \Sigma$ there exists at least [resp. at most] one $q' \in S$ such that $(q, a, q') \in \delta$, the automaton is said to be *complete* [resp. *deterministic*]. By abuse of notation, for a complete deterministic automaton, the unique state $q' \in S$ such that $(q, a, q') \in \delta$ is denoted $\delta(q, a)$. For deterministic complete automata, the notation δ is extended to words in the following way: for all $q \in S$ we set $\delta(q, \varepsilon) = q$, and for all $u \in \Sigma^*$ and $a \in \Sigma$ we define inductively $\delta(q, ua) = \delta(\delta(q, u), a)$. Let $w \in \Sigma^*$ and $q_1, q'_n \in S$. A run σ of $\mathcal{A}$ over w from q_1 to q'_n is a sequence of transitions $(q_i, w_i, q'_i)_{i \in [\![1,n]\!]}$ satisfying $w_1 \dots w_n = w$ and such that for all $i \in [\![1, n-1]\!]$, the state q'_i is equal to q_{i+1}. The *size* of a deterministic automaton $\mathcal{A}$, denoted $|\mathcal{A}|$, is its number of states. For every automaton $\mathcal{A} = (S, \Sigma, \iota, F, \delta)$, the *language recognized* by $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, is the set of words $w \in \Sigma^*$ such that $\delta(\iota, w) \in F$. Let $\mathcal{A} = (S, \Sigma, \iota, F, \delta)$ and $q \in S$. The *residual* of $\mathcal{A}$ in q, denoted $R(\mathcal{A}, q)$, is the language recognized by the automaton $(S, \Sigma, q, F, \delta)$ obtained by replacing the initial state of $\mathcal{A}$ by q. A language is *regular* if there exists an automaton that recognizes it. A deterministic complete automaton is called minimal if there is no smaller automaton that recognizes the same language or, equivalently, if it is the minimal automaton of its language. In a minimal automaton $\mathcal{A}$, if $R(\mathcal{A}, q) = R(\mathcal{A}, q')$, then $q = q'$. We say that a state q of a deterministic complete automaton is a *sink state* if for all $a \in \Sigma$ one has $\delta(q, a) = q$. For any regular language L, there exists a unique minimal deterministic complete automaton (up to state names) recognizing L called the minimal automaton of L.

Accessibility for automata. Let $\mathcal{A} = (S, \Sigma, \iota, F, \delta)$ be a deterministic complete automaton and let $q, r \in S$. We say that r is *accessible* from q when there exists $w \in \Sigma^*$ such that $r = \delta(q, w)$. In that case we denote it by $q \preccurlyeq_{\mathcal{A}} r$. When $q \preccurlyeq_{\mathcal{A}} r$ and $q \neq r$, we write $q \prec_{\mathcal{A}} r$. When either $q \preccurlyeq_{\mathcal{A}} r$ or $r \preccurlyeq_{\mathcal{A}} q$ we say that q and r are *comparable*. When $\mathcal{A}$ is clear from the context we simply write $\preccurlyeq$ and $\prec$. A deterministic complete automaton is called *trim* if every state is accessible from the initial state and from every state there exists an accessible final state. In this paper all automata are assumed to be deterministic complete and trim unless otherwise stated. Let $\mathcal{A}$ be a deterministic complete automaton, and let q be a state of $\mathcal{A}$. The set of *ancestors* of q, denoted $\mathrm{Anc}_{\mathcal{A}}(q)$, is $\{s \in S \mid s \preccurlyeq_{\mathcal{A}} q\}$. The set of *descendants* of q, denoted $\mathrm{Desc}_{\mathcal{A}}(q)$, is $\{s \in S \mid q \preccurlyeq_{\mathcal{A}} s\}$. The *family* of q, denoted $\mathrm{Fam}_{\mathcal{A}}(q)$, is $\mathrm{Anc}(q) \cup \mathrm{Desc}(q)$. When $\mathcal{A}$ is clear from the context we simply write $\mathrm{Anc}(q)$, $\mathrm{Desc}(q)$ and $\mathrm{Fam}(q)$. An automaton $\mathcal{A}$ is called *linear* if $\preccurlyeq_{\mathcal{A}}$ is a total order on its set of states, otherwise $\mathcal{A}$ is *non-linear*. For instance, the automaton on Fig. 1 is non-linear since q_1 and q_2 are not comparable. We denote $\mathcal{I}$ the set of automata recognizing ideal.

Decomposition Problems. For a minimal automaton $\mathcal{A}$, an *intersection [resp. union] decomposition* of $\mathcal{A}$ is a finite sequence of automata $(\mathcal{A}_i)_{i \in [\![1,n]\!]}$ such that $|\mathcal{A}_i| < |\mathcal{A}|$ for all $i \in [\![1, n]\!]$, and

$$\mathcal{L}(\mathcal{A}) = \bigcap_{i \in [\![1,n]\!]} \mathcal{L}(\mathcal{A}_i) \quad [\text{resp.} \quad \mathcal{L}(\mathcal{A}) = \bigcup_{i \in [\![1,n]\!]} \mathcal{L}(\mathcal{A}_i)] \ .$$

When $\mathcal{A}$ admits an intersection [resp. union] decomposition, we say that $\mathcal{A}$ is *composite for intersection* [resp. *for union*] and otherwise it is *prime for intersection* [resp. *for union*].

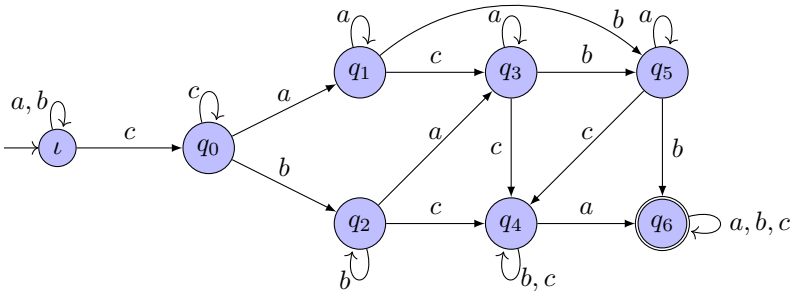

Figure 1 Minimal automaton of $\{cabb, cacca, cbca\} \sqcup\!\sqcup \{a, b, c\}^*$.

2.1 Ideals

In this section, we recall several useful results on ideals. Most of them follow either from Higman's Lemma [10] (there is no infinite language of incomparable words for the subword relation), or from the fact that ideals form a positive variety of languages [25]. See also [9, 31]. An example of a minimal automaton accepting an ideal is shown in Fig. 1. These classical results can be summarized as follows.

▶ **Proposition 4.** *The following properties hold.*

1. *All ideals are regular.*
2. *If L is an ideal, then there exists a finite language K such that $L = K \sqcup\!\sqcup \Sigma^*$. Moreover, there exists a unique finite language with this property of minimal cardinality, which we denote by $L_{\min}$. Finally, for any $u \in L_{\min}$, every strict subword v of u satisfies $v \notin L$.*
3. *If $\mathcal{A}$ is a deterministic automaton recognizing an ideal, then for all $q \in S$, $R(\mathcal{A}, q)$ is an ideal.*
4. *If $\mathcal{A}$ is a minimal automaton recognizing an ideal, then $\mathcal{A}$ is trim and $\preccurlyeq$ is a partial order.*
5. *If $\mathcal{A}$ is a minimal automaton recognizing an ideal, then it has a unique final state, and this final state is a sink state.*

We now provide more technical statements that directly follow from these known properties.

▶ **Proposition 5.** *Let $\mathcal{A} = (S, \Sigma, \iota, F, \delta)$ be an automaton recognizing an ideal. The following properties hold.*

1. *Let q, r be two states satisfying $q \preccurlyeq r$. One has $R(\mathcal{A}, q) \subseteq R(\mathcal{A}, r)$.*
2. *Let $q \in S$ and $u, v \in \Sigma^*$. One has $R(\mathcal{A}, \delta(q, u)) \subseteq R(\mathcal{A}, \delta(q, vu))$.*
3. *Let $q \in S$ and $u, v \in \Sigma^*$. Then either $\delta(q, u) \preccurlyeq \delta(q, vu)$, or $\delta(q, u)$ and $\delta(q, vu)$ are incomparable.*
4. *Let $q \in S$ be a sink state. If $\mathcal{L}(\mathcal{A}) \neq \emptyset$ then $q \in F$.*

We will denote by $\mathcal{I}$ the class of minimal automata recognizing ideals.

3 Intersection-decomposition of non-linear automata

In this section, decomposition and prime automata are understood with respect to intersection. We show that every non-linear automaton recognizing an ideal is decomposable.

▶ **Theorem 6.** *Let $\mathcal{A}$ be a non-linear minimal automaton over an alphabet Σ that recognizes an ideal. Then $\mathcal{A}$ has a decomposition into at most $|\Sigma|$ automata, each recognizing an ideal.*

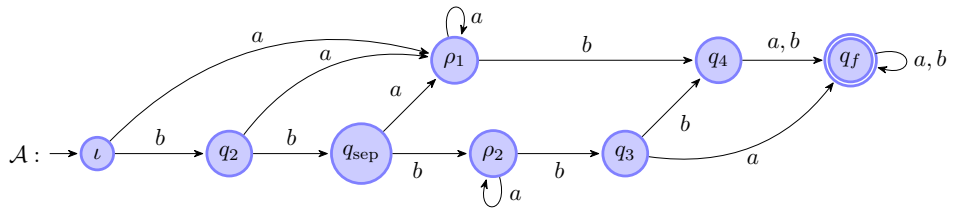

Figure 2 A non-linear automaton recognizing an ideal.

The proof follows from the results established in the following sections. In Section 3.1, we define a set of states, the *separator set* S_{sep}, and establish its main properties. In Section 3.2, we associate with each state $\rho \in S_{\text{sep}}$ a *family automaton* $\mathcal{A}(\rho)$, and prove that these automata form a decomposition of $\mathcal{A}$. More precisely, for each $\rho \in S_{\text{sep}}$ the automaton $\mathcal{A}(\rho)$ is strictly smaller than $\mathcal{A}$ (Proposition 11) and recognizes an ideal (Proposition 13). Moreover, $\mathcal{L}(\mathcal{A}) = \bigcap_{\rho \in S_{\text{sep}}} \mathcal{L}(\mathcal{A}(\rho))$ (Proposition 14). The bound on the size of the decomposition follows from the fact that $|S_{\text{sep}}| \leqslant |\Sigma|$ (Proposition 10.3).

3.1 Ranks and Separation States

We introduce in this section the key notion of *separator state* q_{sep} of a non-linear automaton $\mathcal{A}$. Intuitively, this state is the first point in $\mathcal{A}$ at which a branching towards two incomparable states occurs. We then define the notion of *rank* on the states of $\mathcal{A}$. This notion provides an alternative characterization of the separator state (Lemma 9), and also allows us to introduce an important ingredient of our construction: the *separator set* S_{sep}, which consists of the states immediately following the separator state.

Separator state. We characterize the state in non-linear automata just before the first two states that are not comparable. To introduce this notion formally, we first establish a structural property on non-linear automata recognizing ideals.

▶ **Proposition 7.** *Let $\mathcal{A} = (S, \Sigma, \iota, \{q_f\}, \delta)$ be a minimal automaton accepting an ideal. Then $\mathcal{A}$ is non-linear if and only if the subset of states*

$$\{q \in S \setminus F \mid \text{Desc}(q) \setminus \{q\} \text{ has no least element for } \preccurlyeq\}$$

is non-empty and has a least element for $\preccurlyeq$.

Now for every non-linear automaton $\mathcal{A} = (S, \Sigma, \iota, F, \delta)$ that recognizes an ideal, the *separator state* of $\mathcal{A}$, denoted q_{sep}, is defined by

$$q_{\text{sep}} = \min_{\preccurlyeq} \{q \in S \mid \text{Desc}(q) \setminus \{q\} \text{ has no least element for } \preccurlyeq\}.$$

In other words, q_{sep} is the least element (for $\preccurlyeq$) whose strict successors do not admit a least element. This notion is illustrated in Fig. 2: the set $\{q_{\text{sep}} \in S \mid \text{Desc}(q_{\text{sep}}) \setminus \{q_{\text{sep}}\}\}$ has two distinct minimal elements ρ_1 and ρ_2.

Rank. The notion of rank is a classical one in directed acyclic graphs and corresponds intuitively to a depth notion in the graph. This notion represents how far a state can be from the initial state. More formally, let $\mathcal{A} = (S, \Sigma, \iota, \{q_f\}, \delta)$ be in $\mathcal{I}$ and let $q \in S$. The *rank* of q, denote $\text{rk}(q)$, is the size of a longest path in $\mathcal{A}$ from ι to q that doesn't visit twice any state We denote by S_{sep} the set of states of the lowest rank held by at least two states. It has the property that there exists a transition from q_{sep} to each state of S_{sep}.

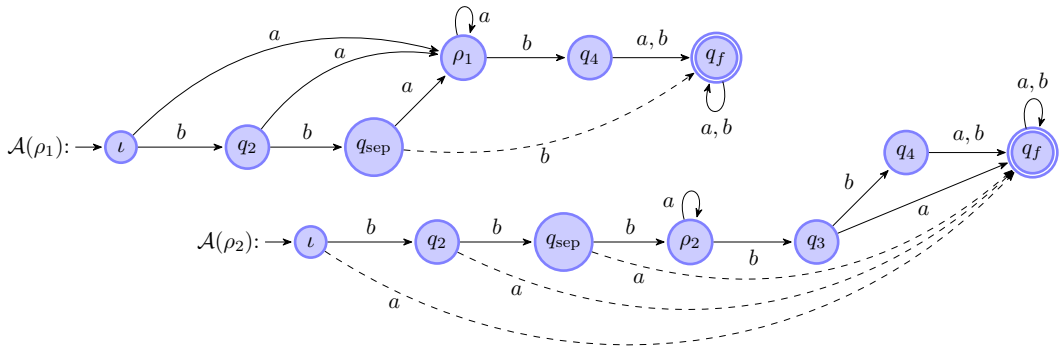

Figure 3 Naive decomposition of the automaton $\mathcal{A}$ depicted in Fig. 2 into automata that don't recognize ideals. The transitions modified compared to the initial automaton are dashed.

For instance, on illustrating Fig. 1, we have $\mathrm{rk}(\iota) = 0$, $\mathrm{rk}(q_2) = 1$, $\mathrm{rk}(q_{\text{sep}}) = 2$, $\mathrm{rk}(\rho_1) = \mathrm{rk}(\rho_2) = 3$, $\mathrm{rk}(q_3) = 4$, $\mathrm{rk}(q_4) = 5$ and $\mathrm{rk}(q_f) = 6$.

The main properties linking ranks and states are summed up in Proposition 8.

▶ **Proposition 8.** *Let $\mathcal{A} = (S, \Sigma, \iota, \{q_f\}, \delta)$ be in $\mathcal{I}$. The following statements holds.*

1. *For all $q, s \in S$ verifying $q \prec s$, one has $\mathrm{rk}(q) < \mathrm{rk}(s)$.*
2. *For all $q, s \in S$. If $\mathrm{rk}(q) = \mathrm{rk}(s)$ and $q \neq s$ then q and s are incomparable for $\preccurlyeq$.*
3. *Let $q \in S$ and $n \in \mathbb{N}$ such that $n \leqslant \mathrm{rk}(q)$. There exists $r \in \mathrm{Anc}(q)$ of rank n.*

These statements allow us to define another characterization of q_{sep}.

▶ **Lemma 9.** *Let $\mathcal{A} = (S, \Sigma, \iota, \{q_f\}, \delta)$ be a non-linear automaton in $\mathcal{I}$ and for all $m \in \mathbb{N}$ let $a_m = |\{s \in S \mid \mathrm{rk}(s) = m\}|$. Then q_{sep} is the only state of rank $\max\{n \in \mathbb{N} \mid \forall m \leqslant n, a_m = 1\}$.*

Separator set. The rank characterization of the separator state makes it natural to consider the set of states of rank $\mathrm{rk}(q_{\text{sep}})+1$. The decomposition we construct contains one automaton for each state in this set. We also introduce the set of letters leading from q_{sep} to the separator set. More formally, let $\mathcal{A} = (S, \Sigma, \iota, \{q_f\}, \delta)$ be a non-linear automaton in $\mathcal{I}$. The *separator set*, denoted S_{sep}, is the subset of S defined by $S_{\text{sep}} = \{q \in S \mid \mathrm{rk}(q) = \mathrm{rk}(q_{\text{sep}}) + 1\}$.

Finally we can define the technical properties of q_{sep} and S_{sep}.

▶ **Proposition 10.** *Let $\mathcal{A} = (S, \Sigma, \iota, \{q_f\}, \delta)$ be a non-linear minimal automaton recognizing an ideal.*

1. *If $q \in S$ satisfies $q \preccurlyeq q_{\text{sep}}$, then $\mathrm{Fam}(q) = S$.*
2. *If $\rho \in S_{\text{sep}}$, then $\preccurlyeq$ induces a total order on $\mathrm{Anc}(\rho) \setminus \{\rho\}$.*
3. *Let $\rho \in S_{\text{sep}}$. There exists $a \in \Sigma$ such that $\delta(q_{\text{sep}}, a) = \rho$.*
4. *$2 \leqslant |S_{\text{sep}}| \leqslant |\Sigma|$.*

3.2 Decomposition of non-linear Automata

Let us first describe intuitively the proposed construction to decompose non-linear automata. For every state ρ of S_{sep} we define a *family automaton* $\mathcal{A}(\rho)$ whose set of states is $\mathrm{Fam}(\rho)$. The main idea is to redirect into $\mathrm{Fam}(\rho)$ all the transitions of $\mathcal{A}$ that leave this set, while preserving the property of recognizing an ideal. A naive approach, illustrated on Fig. 3, would be to redirect such transitions to the unique final sink state. Although this yields a

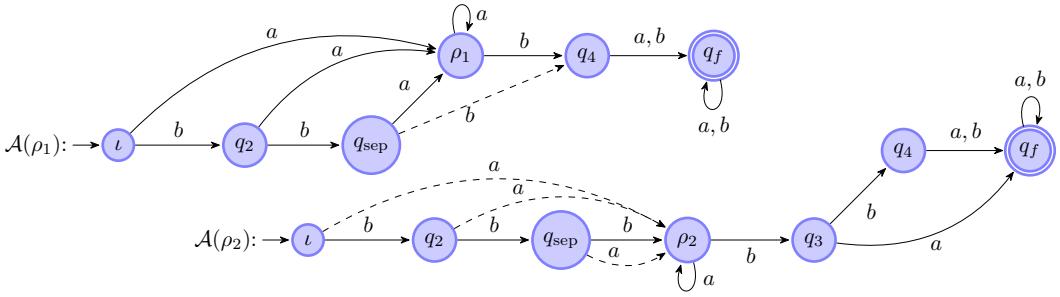

Figure 4 Decomposition of the automaton $\mathcal{A}$ depicted in Fig. 2 into its family automata $\mathcal{A}(\rho_1)$ and $\mathcal{A}(\rho_2)$. The transitions modified compared to the initial automaton are dashed.

decomposition of $\mathcal{A}$, it does not, in general, preserve automata recognizing ideals. Ensuring that the decomposition remains within this class therefore requires a more careful construction. The two main results underlying Theorem 6 are Proposition 13, which shows that each family automaton $\mathcal{A}(\rho)$ recognizes an ideal, and Proposition 14, which ensures that the family automata $(\mathcal{A}(\rho))_{\rho \in S_{\text{sep}}}$ provide a valid decomposition of $\mathcal{A}$.

Family automaton. Let $\mathcal{A} = (S, \Sigma, \iota, \{q_f\}, \delta)$ be a non-linear minimal automaton that recognizes an ideal. For all $\rho \in S_{\text{sep}}$, we define the *family automaton* $\mathcal{A}(\rho) = (\text{Fam}(\rho), \Sigma, \iota, F, \delta_\rho)$, where the set of transitions δ_ρ is defined as follows. For all $q \in \text{Fam}(\rho)$ and $a \in \Sigma$, we let

$$\delta_\rho(q, a) = \delta(s^\rho_{(q,a)}, a), \text{ where } s^\rho_{(q,a)} = \min\{s \in \text{Fam}(\rho) \mid q \preccurlyeq s \wedge \delta(s, a) \in \text{Fam}(\rho)\}.$$

The existence of the least element is provided by Lemma 12 which is deferred to appendix. This construction is illustrated in Fig. 2 and 4: the automaton depicted in Fig. 2 is decomposed into the two automata $\mathcal{A}_{\rho_1}$ and $\mathcal{A}_{\rho_2}$ depicted in Fig. 4. Note that the set of states of each automaton $\mathcal{A}(\rho_i)$ is a strict subset of the sets of $\mathcal{A}$, providing the size constraint for the decomposition. All the transitions starting from a removed state are deleted. Furthermore, δ_{ρ_i} and δ coincide on transitions whose source and target both belong to $\text{Fam}(\rho_i)$. The two sets of transitions critically differ for transitions starting from a state of $\text{Fam}(\rho)$ and leaving this set: in this case, the transition set δ_{ρ_i} redirects the transition to the least state greater than or equal to ρ_i pointed by the same letter in δ. We show that the family automata are smaller than $\mathcal{A}$.

▶ **Proposition 11.** *Let $\mathcal{A}$ be a non-linear automaton that recognizes an ideal. Then, for every $\rho \in S_{\text{sep}}$, one has $|\mathcal{A}(\rho)| < |\mathcal{A}|$.*

Proof. Proposition 10.4 states that $|S_{\text{sep}}| \geqslant 2$, hence there exists $q \in S_{\text{sep}} \setminus \{\rho\}$. Then, by definition of S_{sep}, $\text{rk}(q) = \text{rk}(\rho)$, and Proposition 8.2 implies $q \notin \text{Fam}(\rho)$. Consequently, the construction of $\mathcal{A}(\rho)$ excludes at least one state of $|\mathcal{A}|$, namely q, hence $|\mathcal{A}(\rho)| < |\mathcal{A}|$. ◀

▶ **Lemma 12.** *Let $\mathcal{A} = (S, \Sigma, \iota, \{q_f\}, \delta)$ be in $\mathcal{I}$. Let $\rho \in S_{\text{sep}}$, $q \in \text{Fam}(\rho)$ and $a \in \Sigma$. The set $\{s \in \text{Fam}(\rho) \mid q \preccurlyeq s \wedge \delta(s, a) \in \text{Fam}(\rho)\}$ has a least element.*

Proposition 13 applied to the initial state yields that $\mathcal{A}(\rho)$ accepts an ideal. The proof is rather technical, relying both on the partially ordered structure of the automata and on the subword closure of ideals.

▶ **Proposition 13.** *Let $\mathcal{A} = (S, \Sigma, \iota, \{q_f\}, \delta)$ be a non-linear minimal automaton that recognizes an ideal and $\rho \in S_{\text{sep}}$. For all $q \in \text{Fam}(\rho)$, the language $R(\mathcal{A}(\rho), q)$ is an ideal.*

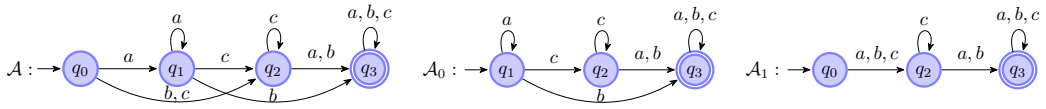

Figure 5 An automaton $\mathcal{A}$, along with two smaller automata satisfying $\mathcal{L}(\mathcal{A}_0) \cap \mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A})$.

Finally, to prove Theorem 6, it remains to show that the language of $\mathcal{A}$ is equal to the intersection of the languages of its family automata.

▶ **Proposition 14.** *Let $\mathcal{A} = (S, \Sigma, \iota, \{q_f\}, \delta)$ be a non-linear minimal automaton that recognizes an ideal. One has*

$$\mathcal{L}(\mathcal{A}) = \bigcap_{\rho \in S_{\text{sep}}} \mathcal{L}(\mathcal{A}(\rho)) \ .$$

The proof is deferred to the extended version [4]; the key idea is that,by construction, $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}(\rho))$, one directly has that every word accepted by $\mathcal{A}$ is also accepted by all the $\mathcal{A}(\rho)$. The reverse inclusion can be showed by proving that for any $w \notin \mathcal{L}(\mathcal{A})$, there exists a $\mathcal{A}(\rho)$ that doesn't accept it.

4 Intersection-decomposition of linear automata

The goal of this section is to prove Proposition 15, which characterizes intersection-composite linear automata recognizing ideals. Therefore, during the whole section, we use terms such as "composite" and "prime" to specifically refer to the *intersection* decomposition.

We introduce notations adapted to linear automata. For any linear automaton $\mathcal{A}$, we denote it set of states as $\{q_0, q_1, \ldots, q_n\}$, where the state enumeration agrees with the transition relation: $q_i \preccurlyeq q_j$ if and only if $i \leqslant j$. Remark that $\iota = q_0$ as otherwise q_0 is not reachable. Moreover, if $\mathcal{A}$ is a minimal automaton recognizing an ideal, it set of final states is $\{q_n\}$ as Proposition 4.1 implies that such automata have one final sink state, and q_n is the only sink state of $\mathcal{A}$.

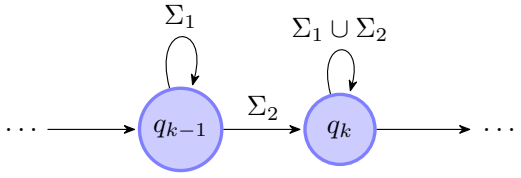

Figure 6 Illustration of a damping pattern between q_{k-1} and q_k.

Damping pattern.. We characterize composite linear automata through the notion of *damping patterns*. A damping pattern of a linear automaton is a pair of consecutive states such that each letter that either keeps the first state unchanged or moves from the first state to the second also induces a loop on the second state. Formally, let $\mathcal{A} = (\{q_0, q_1, \ldots, q_n\}, \Sigma, q_0, F, \delta)$ be a linear automaton. For all $i, j \in [\![1, n]\!]$, we denote by $\Sigma_{i,j}(\mathcal{A})$, or simply $\Sigma_{i,j}$ when the automaton is clear, the set $\{a \in \Sigma \mid \delta(q_i, a) = q_j\}$ of letters labelling the transitions from q_i to q_j. For all $k \in [\![1, n-1]\!]$, $\mathcal{A}$ has a damping pattern between q_{k-1} and q_k if

$$\Sigma_{k-1,k-1} \cup \Sigma_{k-1,k} \subseteq \Sigma_{k,k}.$$

For instance, this pattern is illustrated in Fig. 6. And the automaton $\mathcal{A}$ in Fig. 5 has a damping pattern between q_0 and q_1. Indeed in that case $\Sigma_{0,0} \cup \Sigma_{0,1} = \{a\} = \Sigma_{1,1}$. The figure shows that $\mathcal{A}$ is composite, which is consistent with the presence of a damping pattern: the following proposition states that damping patterns characterize composite linear automata.

▶ **Theorem 15.** *Let $\mathcal{A}$ be a linear minimal automaton that recognizes an ideal. Then $\mathcal{A}$ is prime if and only if it has no damping pattern. Moreover, if $\mathcal{A}$ is composite it is decomposable into two linear automata recognizing ideals.*

The proof of Theorem 15 in the case where $\mathcal{A}$ linear is done in the next two subsections.

In Section 4.1, we show how to decompose automata that have a damping pattern between two states q_{k-1} and q_k. To this end, we define the reduced automata $\mathcal{A}_{k-1}$ and $\mathcal{A}_k$, obtained by removing the state q_{k-1} (resp. q_k) from $\mathcal{A}$, and redirecting the transitions leading towards the removed state to its successor q_k (resp. q_{k+1}). We prove that both automata recognize ideals (Proposition 16), and that they satisfy $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_{k-1}) \cap \mathcal{L}(\mathcal{A}_k)$ (Proposition 17). Since both automata have strictly fewer states than $\mathcal{A}$, this establishes that $\mathcal{A}$ is composite.

In Section 4.2 we show that the automata without damping patterns are prime. To that end, we rely on the notion of *primality witness* of an automaton $\mathcal{A}$, which is a word $w \in \Sigma^*$ that does not belong to $\mathcal{L}(\mathcal{A})$ but is in the language recognized by every automaton that is strictly smaller than $\mathcal{A}$ and recognizes a superset of $\mathcal{L}(\mathcal{A})$. Formally, we construct a set of words $S_{\mathrm{wit}}(\mathcal{A}) \subseteq \Sigma^*$ that is non-empty as long as $\mathcal{A}$ does not have a damping pattern (Lemma 18). Then, we prove that $S_{\mathrm{wit}}(\mathcal{A}) \cap \mathcal{L}(\mathcal{A}) = \emptyset$ (Proposition 19) yet $S_{\mathrm{wit}} \subseteq \mathcal{L}(\mathcal{A}')$ for every automaton $\mathcal{A}'$ satisfying $|\mathcal{A}'| < |\mathcal{A}|$ and $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$ (Proposition 20). Therefore the intersection of every sequence of languages recognized by automata strictly smaller than $\mathcal{A}$ either does not contain $\mathcal{L}(\mathcal{A})$ or contains S_{wit}, which proves that $\mathcal{A}$ is prime.

4.1 Decomposition of automata with a damping pattern

In this section, we show that every automaton with a damping pattern can be decomposed into two automata recognizing ideals. We begin by defining, for every $k \in [\![1, n-1]\!]$, an automaton $\mathcal{A}_k$, called the reduced automaton, obtained by removing the state q_k from $\mathcal{A}$ and redirecting the transitions heading to q_k toward q_{k+1}. We then show that, if $\mathcal{A}$ recognizes an ideal, so does each reduced automaton. Finally, we prove that every pair of reduced automata corresponding to a damping pattern yields a decomposition of $\mathcal{A}$.

Reduced automaton.. Let $\mathcal{A} = (\{q_0, q_1, \dots, q_n\}, \Sigma, q_0, \{q_n\}, \delta)$ be a linear automaton. For every $k \in [\![0, n-1]\!]$, the *reduced automaton* $\mathcal{A}_k$ is defined as follows. Let

$$\delta_k^- = \{(q, a, q_k) \in \delta \mid q \in S\} \text{ and } \delta_k^+ = \{(q, a, q_{k+1}) \mid (q, a, q_k) \in \delta\}.$$

Then we set $\mathcal{A}_k = (\{q_0, q_1, \dots, q_n\} \setminus \{q_k\}, \Sigma, \iota_k, \{q_n\}, \delta_k)$, where

$$\delta_k = (\delta \setminus \delta_k^-) \cup \delta_k^+ \quad \text{and} \quad \iota_k = \begin{cases} q_1 & \text{if } k = 0; \\ q_0 & \text{if } k > 0. \end{cases}$$

Fig. 5 illustrates this construction by showing an automaton $\mathcal{A}$ along with the automata $\mathcal{A}_0$ and $\mathcal{A}_1$ obtained by this process. Remark that this construction guarantees that each reduced automaton is linear. Our first result states that if $\mathcal{A}$ recognizes an ideal, so do its reduced automata.

▶ **Proposition 16.** *Let $\mathcal{A}$ be a linear automaton that recognizes an ideal. For all $k \in [\![0, n-1]\!]$, the automaton $\mathcal{A}_k$ also recognizes an ideal.*

To conclude, we show that the reduced automata can be used to form a decomposition of $\mathcal{A}$.

▶ **Proposition 17.** *Let $\mathcal{A}$ be a linear automaton that recognizes an ideal. If $\mathcal{A}$ has a damping pattern between two states q_{k-1} and q_k, then $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_{k-1}) \cap \mathcal{L}(\mathcal{A}_k)$.*

4.2 Primality witnesses of automata without damping patterns

For every linear automaton $\mathcal{A} = (S, \Sigma, \iota, F, \delta)$ we define a set of words $S_{\text{wit}}(\mathcal{A})$, we show that this set is not empty if $\mathcal{A}$ does not have a damping pattern (Lemma 18), and that all the words in this set are primality witnesses of $\mathcal{A}$ (Lemma 20).

Witness set. Let $\mathcal{A}$ be a linear automaton without damping pattern. The witnesses of $\mathcal{A}$ are defined by concatenating smaller words as follows:

$$S_{\text{wit}}(\mathcal{A}) = \{w(1)\dots w(n-1) \in \Sigma^* \mid \delta(q_{i-1}, w(i)) = q_i \neq \delta(q_i, w(i)) \text{ for every } i \in [\![1, n-1]\!]\}.$$

We denote the elements of $S_{\text{wit}}(\mathcal{A})$ by their decomposition in words $w(1)\dots w(n-1)$, and implicitly assume that this decomposition satisfies the above condition.

▶ **Lemma 18.** *Every linear automaton $\mathcal{A}$ without damping pattern satisfies $S_{\text{wit}}(\mathcal{A}) \neq \emptyset$.*

Proof. Let $\mathcal{A} = (\{q_0, q_1, \dots, q_n\}, \Sigma, q_0, F, \delta)$ be a linear automaton without damping pattern. Then for every $i \in [\![1, n-1]\!]$ we have $(\Sigma_{i-1,i-1} \cup \Sigma_{i-1,i}) \setminus \Sigma_{i,i} \neq \emptyset$, and we can set

$$w(i) = \begin{cases} a \in \Sigma_{i-1,i} \setminus \Sigma_{i,i} & \text{if } \Sigma_{i-1,i} \setminus \Sigma_{i,i} \neq \emptyset; \\ a_1 a_2 \in (\Sigma_{i-1,i-1} \setminus \Sigma_{i,i}) \cdot \Sigma_{i-1,i} & \text{if } \Sigma_{i-1,i} \setminus \Sigma_{i,i} = \emptyset \text{ and } \Sigma_{i-1,i-1} \setminus \Sigma_{i,i} \neq \emptyset. \end{cases}$$

Both cases imply that $\delta(q_{i-1}, w(i)) = q_i \neq \delta(q_i, w(i))$ as required by the definition of $S_{\text{wit}}(\mathcal{A})$, hence the word $w(1)w(2)\dots w(n-1)$ is in $S_{\text{wit}}(\mathcal{A})$. ◀

To prove that each word $w_{\text{wit}} = w(1)\dots w(n-1) \in S_{\text{wit}}(\mathcal{A})$ is a primality witness, we rely on the following technical lemma stating that, while $w_{\text{wit}} \notin \mathcal{L}(\mathcal{A})$, duplicating any $w(i)$ yields a word in $\mathcal{L}(\mathcal{A})$.

▶ **Proposition 19.** *Let $\mathcal{A}$ be a linear minimal automaton that recognizes an ideal. Every $w_{\text{wit}} = w(1)\dots w(n-1) \in S_{\text{wit}}(\mathcal{A})$ satisfies $w_{\text{wit}} \notin \mathcal{L}(\mathcal{A})$ and*

$$w(1)w(2)\dots w(i-1)w(i)w(i)w(i+1)\dots w(n-1) \in \mathcal{L}(\mathcal{A}) \text{ for all } i \in [\![1, n-1]\!].$$

▶ **Proposition 20.** *Let $\mathcal{A}$ be a linear minimal automaton that recognizes a non-empty ideal. For every automaton $\mathcal{A}'$ smaller than $\mathcal{A}$, $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$ implies $S_{\text{wit}}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$.*

5 Union decomposition

Throughout this section, the notions of decomposition, composite and prime refer to union decomposition. The main result is Theorem 21, which characterizes prime minimal automata recognizing ideals. For composite automata, we show how to construct a decomposition into $|L_{\min}|$ prime automata recognizing ideals. The characterization relies on the notion of *accelerating patterns*.

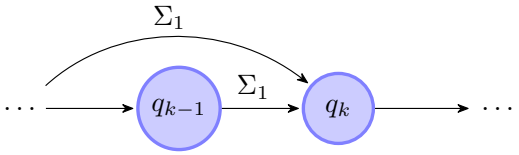

Figure 7 Illustration of an accelerating pattern between q_{k-1} and q_k.

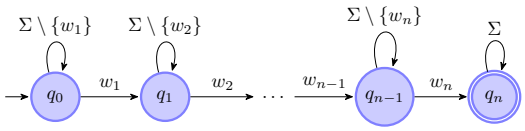

Figure 8 Example of $\mathcal{A}_w$ when $w = w_1 \dots w_n$.

Accelerating pattern. An *accelerating pattern* in a linear automaton recognizing an ideal is a state such that each letter labelling an incoming transition from its greatest predecessor also labels an incoming transition from a strictly smaller state. Formally, let $\mathcal{A} = (\{q_0, q_1, \dots, q_n\}, \Sigma, q_0, \{q_n\}, \delta)$ be a linear automaton recognizing an ideal such that $q_{k-1} \prec q_k$ for all $k \in [\![1, n]\!]$. For all $k \in [\![1, n]\!]$, $\mathcal{A}$ has an accelerating pattern if there exists k such that

$$\Sigma_{k-1,k} \quad \subseteq \quad \bigcup_{j=0}^{k-2} \Sigma_{j,k}.$$

For instance,this pattern is illustrated in Fig. 7. And the automaton $\mathcal{A}$ in Fig. 5 has an accelerating pattern in q_2. This automaton can be decomposed into automata accepting $w \sqcup\!\sqcup\ \Sigma^*$ for $w \in \{ab, ba, bb, ca, cb\}$. This illustrates the connection between accelerating patterns and decomposability, which we now formalize.

▶ **Theorem 21.** *Let $\mathcal{A}$ be a minimal automaton that recognizes an ideal. We denote $\mathcal{L}(\mathcal{A}) = L$. If we denote $m = \max\{|w| \mid w \in L_{\min}\}$ then the following are equivalent:*

1. *$\mathcal{A}$ is prime.*
2. *The size of $\mathcal{A}$ is equal to $m + 1$.*
3. *$\mathcal{A}$ is linear and has no accelerating pattern.*

Before proving the above theorem we need to define the automata used in the decomposition. They are the ones that recognize languages of the form $\{w\} \sqcup\!\sqcup\ \Sigma^*$ where w is a word.

Principal Automata. The *principal automaton* generated by $w \in \Sigma^n$, written $\mathcal{A}_w$, is the minimal automaton that recognizes $\{w\} \sqcup\!\sqcup\ \Sigma^*$. Its set of states is $S_w = \{q_i \mid i \in [\![0, n]\!]\}$, its alphabet is Σ, its initial state is q_0, its final state is $\{q_n\}$ and its set of transitions δ_w is

$$\{(q_{i-1}, w_i, q_i) \mid i \in [\![1, n-1]\!]\} \cup \bigcup_{i \in [\![0, n-1]\!]} \{(q_i, a, q_i) \mid a \in \Sigma \setminus \{w_{i+1}\}\} \cup \{(q_n, a, q_n) \mid a \in \Sigma\}.$$

This definition is illustrated in Fig. 8.

Proof. We show that (1.) and (2.) of Theorem 21 are equivalent. The equivalence between (2.) and (3.) is proved in the extended version [4].

Let $w \in L_{\min}$ such that $|w| = m$. We first claim that every automaton of size smaller than or equal to m recognizing w also recognizes at least one word that is not in L. Let $\mathcal{A}' = (S, \Sigma, \iota, F, \delta)$ be an automaton such that $|\mathcal{A}'| \leqslant m$ and $w \in \mathcal{L}(\mathcal{A})$. Let $(q_i)_{i \in [\![0,m]\!]} \in S^{m+1}$ be

the states visited by the accepting path on w. Then $q_0 = \iota$, $q_m \in F$ and $\delta(q_{i-1}, w_i) = q_i$ for all $i \in [\![1, m]\!]$. By a cardinality argument, there exist two $i, j \in [\![0, m]\!]$ such that $i < j$ and $q_i = q_j$. Let $w' = w_1 \dots w_{i-1} w_i w_{j+1} w_{j+2} \dots w_m$. By construction, $\delta(\iota, w') = q_m$, proving that $w' \in \mathcal{L}(\mathcal{A}')$. But w' is a subword of w. By Proposition 4.2, $w' \notin L$, proving the claim. Hence $|\mathcal{A}| \geqslant m + 1$ always hold.

Let us now show that (2.) implies (1.). Assume now that $|\mathcal{A}| = m + 1$. Towards building a contradiction, assume that $\mathcal{A}$ admits a decomposition $(\mathcal{A}_i)_{i \in [\![1,k]\!]}$. Since $w \in \mathcal{L}(\mathcal{A})$, there exists $i \in [\![1, k]\!]$ such that $w \in \mathcal{L}(\mathcal{A}_i)$. Using the claim, either $|\mathcal{A}_i| \geqslant m + 1$ or $\mathcal{A}_i$ recognizes a word that is not in $\mathcal{L}(\mathcal{A})$, and both cases contradict the fact that $\mathcal{A}_i$ is a component of a union-decomposition of $\mathcal{A}$. Therefore $\mathcal{A}$ is prime, proving that (2.) implies (1.).

To finish we show that (1.) implies (2.). Assume that the size of $\mathcal{A}$ is not $m + 1$. Since $w \in \mathcal{L}(\mathcal{A})$, using the claim, $|\mathcal{A}| \geqslant m + 1$. Therefore $|\mathcal{A}| > m + 1$. But, one has

$$\mathcal{L}(\mathcal{A}) \sqcup\!\sqcup \Sigma^* = \bigcup_{u \in L_{\min}} \{u\} \sqcup\!\sqcup \Sigma^* \ .$$

By construction of principal automata, $|\mathcal{A}_u| = |u|+1$. Therefore$(\mathcal{A}_u)_{u \in L_{\min}}$ is a decomposition of $\mathcal{A}$ in $|L_{\min}|$ linear automata that recognize ideals, proving that $\mathcal{A}$ is not prime, which proves that that (1.) implies (2.). ◀

Note that the proof of equivalence of (1.) and (2.) is constructive: whenever $\mathcal{A}$ is composite we can first compute $L_{\min}$ using the algorithm in [9], and then the present proof yields a decomposition into $|L_{\min}|$ prime linear automata that recognize ideals.

6 Complexity results

In this section, we show that the problems of deciding whether a minimal automaton recognizing an ideal is decomposable with respect to intersection and union are both in NL. We also give bounds on the size of the decomposition into prime automata obtained by iterating the algorithms.

▶ **Theorem 1.** *Given a minimal automaton $\mathcal{A}$ recognizing an ideal, deciding whether $\mathcal{A}$ is prime for intersection is in NL.*

Proof. Since the Immerman-Szelepcsényi theorem [12] states that NL = co-NL, it suffices to design an NL algorithm deciding whether $\mathcal{A}$ is composite. To this end, we use five procedures: **InterComposite**, **NoPath**, **NonLinear**, **NoNext** and **NoDamping**, depicted in Fig. 9.

Our main procedure, **InterComposite**, is based on the characterization of intersection-prime automata given by Theorems 6 and 15: $\mathcal{A}$ is composite if and only if it is non-linear or admits a damping pattern. Accordingly, the algorithm first invokes **NonLinear**, which determines whether $\mathcal{A}$ is non-linear. If this is not the case, it nondeterministically guesses two states, invokes **NoNext** to test whether they are consecutive, and then **Damping** to check for the existence of a damping pattern. We now describe each subroutine in detail and show that they all operate in logarithmic space.

- **Path** and **NoPath** respectively decide whether there is a path or no path between two given states. It is known that both problems are in NL [12], therefore we do not redefine the corresponding routines here.
- **NonLinear** takes as input an automaton and accepts if it is non-linear. It guesses two states and returns **True** if and only if they are incomparable, using **NoPath** as a subroutine to perform the reachability checks. Since only two state indices need to be stored in addition to the workspace of the NL **NoPath** procedure, this algorithm is in NL.

InterComposite($\mathcal{A}$)
if NonLinear($\mathcal{A}$) **then**
  return(**True**)
else
  guess $q, r \in S$
  return(**Next**$(q, r) \wedge$ **Damping**(q, r))
end

NoDamping(q_i, q_j)
for $a \in \Sigma$ **do**
  if $\delta(q_i, a) \in \{q_i, q_j\} \wedge \delta(q_j, a) \neq q_j$ **then**
    return(**True**)
  end
end
return(**False**)

NonLinear($\mathcal{A}$)
guess $q, r \in S$
return(**NoPath**$(q, r) \wedge$ **NoPath**(r, q))

NoNext(q_i, q_j)
if NoPath$(q_i, q_j) \vee i = j$ **then**
  return(**True**)
else
  guess $q \in S \setminus \{q_i, q_j\}$
  if Path(q_i, q) $\wedge$ **Path**(q, q_j) **then**
    return(**True**)
  end
  return(**False**)
end

Figure 9 An NL algorithm deciding whether a minimal automaton $\mathcal{A}$ recognizing an ideal is composite for intersection, together with auxiliary NL procedures performing local structural checks: detecting a damping pattern between two states, determining whether an automaton is linear, and checking whether two states are consecutive.

- **NoNext** decides whether, in a linear automaton, q_j is not the direct successor of q_i for the total order on states. The first **if** statement relies on **NoPath** to check whether $q_i \not\preccurlyeq q_j$ or $q_i = q_j$, both of which witness that q_j is not the direct successor of q_i. Otherwise, since $q_i \prec q_j$, it remains to check whether there exists q such that $q_i \prec q \prec q_j$, which is checked using the **Path** subroutine. Since only three state indices need to be stored in addition to the workspace of the NL **Path** and **NoPath** procedures, this algorithm is in NL. And since Nl = co-Nl, there exists **Next** an NL algorithm.
- **NoDamping** decides, given i and j, whether there is no damping pattern between q_i and q_j. The **if** statement checks whether there exist a letter $a \in \Sigma_{i,i} \cup \Sigma_{i,j}$ which is not in $\Sigma_{j,j}$. Since a damping pattern corresponds to the inclusion $\Sigma_{i,i} \cup \Sigma_{i,j} \subseteq \Sigma_{j,j}$, the existence of such a letter witnesses its absence. Only two state indices and one letter name need to be stored, which can be done in logarithmic space. Moreover, since NL = co-NL, there also exists an NL algorithm for the complement **Damping** of **NoDamping**. ◀

We show that these steps can be performed in linear time, relying on a topological sorting (e.g., Kahn's algorithm [16]). Such a sorting can be used, first, to determine whether the automaton is linear. If it is not, it can then be used to compute the separator set. Otherwise, we simply traverse the states in topological order while searching for a damping pattern.

▶ **Theorem 2.** *Let $\mathcal{A} = (S, \Sigma, \iota, F, \delta)$ be an automaton recognizing an ideal. Determining whether $\mathcal{A}$ has an intersection decomposition can be done in time $\mathcal{O}(|\mathcal{A}||\Sigma|)$, and if so, such a decomposition can be constructed in time $\mathcal{O}(|\mathcal{A}||\Sigma|^2)$.*

We now study the size of the decomposition obtained by iteratively applying our constructions until reaching intersection-prime components. We show that repeated application of our procedure on a non-linear automaton yields at most exponentially many linear automata, and that this bound is tight. Similarly, applying our procedure to a linear automaton produces at most exponentially many prime components, and this bound is also tight. In both cases, we construct families of automata witnessing the tightness of the exponential bound.

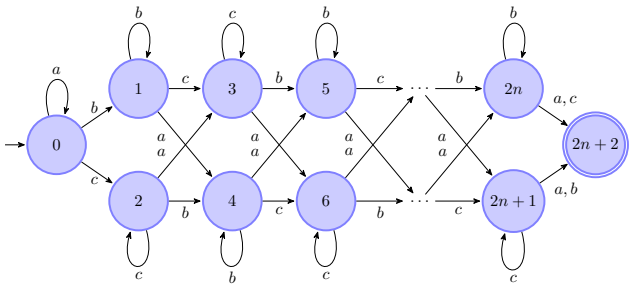

Figure 10 Example of the construction of $\mathcal{A}_n$ non-linear that is decomposed by the algorithm in $2^{|\mathcal{A}|/2-1}$ linear automata.

▶ **Proposition 22.** *Let $\mathcal{A} = (S, \Sigma, \iota, \{q_f\}, \delta)$ be an non-linear minimal automaton that recognizes an ideal. Iterating the decomposition described in Theorem 6 yields at most $2^{|\mathrm{Desc}_{\mathcal{A}}(q_{\mathrm{sep}})|}$ linear automata.*

We conjecture that this bound can be lowered to $2^{\frac{|\mathrm{Desc}_{\mathcal{A}}(q_{\mathrm{sep}})|}{2}}$, and we know it cannot be further improved. Indeed, for all $n \in \mathbb{N}$, let $\mathcal{A}_n$ be the automaton in Fig. 10. This automaton has $2n+2$ states, and is decomposed in 2^n linear automata by iterating the construction of Theorem 6, one for each path from state 0 to state $2n+2$.

▶ **Proposition 23.** *Let $\mathcal{A}$ be a linear automaton that recognizes an ideal. Iterating the decomposition described in Theorem 15 yields at most $2^{|\mathcal{A}|}$ automata.*

The bound of Proposition 23 is tight. Let $\mathcal{A} = (S, \Sigma, \iota, \{q_f\}, \delta)$ and $\mathcal{A}' = (S', \Sigma, \iota', \{q'_f\}, \delta')$ be two minimal automata that recognize ideals. We define $\mathcal{A} \cdot \mathcal{A}' = (S \sqcup S' \setminus F, \Sigma, \iota, F', \delta'')$, where $\sqcup$ is the disjoint union (we rename states if necessary) and where

$$\delta'' = \{(q, a, q') \in \delta \mid q' \neq q_f\} \cup \{(q, a, \iota') \mid (q, a, q_f) \in \delta\} \cup \delta' \ .$$

We write, for an automaton $\mathcal{A}$ that recognizes an ideal and $n \in \mathbb{N} \setminus \{0\}$, $\mathcal{A}^1 = \mathcal{A}$ and inductively $\mathcal{A}^n = \mathcal{A}^{n-1} \cdot \mathcal{A}$. Let $\mathcal{A}, \mathcal{A}_0$ and $\mathcal{A}_1$ be the automata of Fig. 5. The size of $\mathcal{A}^n$ is then $3n+1$. Using iteratively Proposition 15, $\mathcal{A}^n$ is decomposed in $\{\mathcal{A}_{i(1)} \dots \mathcal{A}_{i(n)} \mid (i(j))_{j \in [\![1,n]\!]} \in \{0,1\}^n\}$, which contains $2^n = \Omega(2^{|\mathcal{A}_i|})$ prime automata. Using the two previous propositions, one can conclude the following:

▶ **Corollary 24.** *Let $\mathcal{A}$ be a minimal automaton recognizing an ideal. Using algorithms of the previous sections, one can compute an intersection-decomposition of $\mathcal{A}$ into at most $2^{2|\mathcal{A}|}$ intersection-prime automata that recognize ideals.*

We now investigate the complexity of deciding the primality for the union decomposition.

▶ **Theorem 3.** *Given a minimal automaton $\mathcal{A}$ recognizing an ideal, deciding whether $\mathcal{A}$ is prime for union is in NL.*

Proof. Since the Immerman-Szelepcsényi theorem [12] states that NL = co-NL, it suffices to design an NL algorithm deciding whether $\mathcal{A}$ is composite. To this end, we use five procedures: **UnionComposite**, **NoPath**, **NonLinear** and **Next**, depicted in Fig. 9 and Fig. 11. We refer the reader to the proof of Theorem 1 for the detailed description of the subroutines and the fact that they are in NL.

The main procedure **UnionComposite** relies of the characterization of union-composite automata stated in Theorem 21: $\mathcal{A}$ is composite if and only if it is non-linear or has an accelerating pattern. Accordingly, **UnionComposite** first calls **NonLinear** to check whether

```
UnionComposite(A)
if NonLinear(A) then
  | return(True)
else
  | guess q, s ∈ S ; // We guess a state that has the accelerating pattern
  | if Next(q, s) ;                    // Verify that that q = q_{i-1} and s = q_i
  |  then
  |  | for a ∈ Σ do
  |  |  | if δ(q, a) = s ;                                  // For all a ∈ Σ_{i-1,i}
  |  |  |  then
  |  |  |  | guess r ∈ S ;          // we try to guess a q_j such that a ∈ Σ_{j,i}
  |  |  |  | if ¬(Path(r, q) ∧ r ≠ q ∧ δ(r, a) = s) then
  |  |  |  |  | return(False)
  |  |  |  | end
  |  |  | end
  |  | end
  |  | return(True) ;        // We accept if for all a ∈ Σ_{i-1,i} we found a q_j
  | else
  |  | return(False)
  | end
end
```

Figure 11 An NL algorithm deciding whether a minimal automaton $\mathcal{A}$ recognizing an ideal is composite for union. The procedure guesses candidate states witnessing the accelerating pattern and verifies locally the required transition constraints.

$\mathcal{A}$ is non-linear. If it is the case, it accepts. Otherwise, it tries to guess a state $s = q_i$ that may witness an accelerating pattern, together with its predecessor $q = q_{i-1}$. Next, it sequentially enumerates all the letters a in Σ, and for each $a \in \Sigma_{i-1,i}$ it nondeterministically guesses a state $q_j \prec q_{i-1}$ such that $a \in \Sigma_{j,i}$. If this succeeds for all letters, then the algorithm has found an accelerating pattern and accepts. Otherwise, the algorithm returns **False**. At each step, a fixed number of state indexes and letter names must be stored, which can be achieved using a logarithmic space. ◀

7 Conclusion

We studied decomposition problems for automata recognizing ideal languages, a subclass of aperiodic languages corresponding to level 1/2 of the Straubing-Thérien hierarchy. We showed that both intersection and union primality can be decided in NL, which contrasts with the general setting, where no algorithms below EXPTIME are known. We now discuss how our results relate to wider open questions.

Are there efficient decomposition algorithms for aperiodic languages? A natural direction for future work is to extend our techniques beyond ideals, and to climb further in the Straubing–Thérien hierarchy. Higher levels are obtained by iterating Boolean and polynomial closure operations. The techniques developed in this work for both intersection and union decompositions may provide a starting point for studying Boolean closures of ideals.

Which classes of regular languages are preserved by decomposition? One notable aspect of our results is that decompositions preserve the class of ideal languages: whenever an automaton recognizing an ideal is decomposable (either as an intersection or as a union), it can be decomposed into automata that also recognize ideals. This mirrors what is known for permutation automata [14]. In contrast, for automata recognizing finite languages, existing decomposition techniques do not preserve the class [27]. This contrast raises several natural questions: can one prove that some finite languages necessarily require infinite languages in their decompositions? More generally, can we characterize which subclasses of languages are closed under decomposition, and which are not?

How to decompose infinite word languages? In the context of model checking, it would be particularly worthwhile to explore how the various decomposition results extend to infinite words. Investigating the languages defined by LTL formulas, or equivalently by $FO(<)$, presents a natural and relevant challenge, especially with the aim of simplifying specifications. A promising starting point would be the first existential fragment of $FO(<)$, which, over finite words, corresponds to the ideals studied in this paper.

References

1 Parosh Aziz Abdulla, Aurore Collomb-Annichini, Ahmed Bouajjani, and Bengt Jonsson. Using forward reachability analysis for verification of lossy channel systems. *Formal Methods in System Design*, 25(1):39–65, 2004. doi:10.1023/B:FORM.0000033962.51898.1A.

2 Corentin Barloy, Michaël Cadilhac, Charles Paperman, and Thomas Zeume. The regular languages of first-order logic with one alternation. In *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, 2022. doi:10.1145/3531130.3533371.

3 Jiri Barnat, Vincent Bloemen, Alexandre Duret-Lutz, Alfons Laarman, Laure Petrucci, Jaco van de Pol, and Etienne Renault. Parallel model checking algorithms for linear-time temporal logic. In *Handbook of Parallel Constraint Reasoning*, pages 457–507. Springer, 2018. doi:10.1007/978-3-319-63516-3_12.

4 Mathias Berry, Pierre-Cyrille Héam, and Ismaël Jecker. Decomposition of automata recognizing ideals, 2026. doi:10.48550/arXiv.2604.25619.

5 Walter S Brainerd. The minimalization of tree automata. *Information and Control*, 13(5):484–491, 1968. doi:10.1016/S0019-9958(68)90917-0.

6 Andreas Engelbredt Dalsgaard, Alfons Laarman, Kim G. Larsen, Mads Chr. Olesen, and Jaco van de Pol. Multi-core reachability for timed automata. In *FORMATS 2012, London, UK, September 18-20, 2012. Proceedings*, Lecture Notes in Computer Science, pages 91–106. Springer, 2012. doi:10.1007/978-3-642-33365-1_8.

7 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.

8 Leonard H. Haines. On free monoids partially ordered by embedding. *Journal of Combinatorial Theory*, 6:94–98, 1969. doi:10.1016/S0021-9800(69)80111-0.

9 Pierre-Cyrille Héam. On shuffle ideals. *RAIRO-Theoretical Informatics and Applications*, 36(4):359–384, 2002. doi:10.1051/ITA:2003002.

10 Graham Higman. Ordering by divisibility in abstract algebras. *Proc. Lond. Math. Soc.*, 3(2):326–336, 1952. doi:10.1112/PLMS/S3-2.1.326.

11 John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979. doi:10.1145/568438.568455.

12 Neil Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on computing*, 17(5):935–938, 1988. doi:10.1109/SCT.1988.5270.

13 Ismaël Jecker, Orna Kupferman, and Nicolas Mazzocchi. Unary prime languages. In *MFCS 2020, Prague, Czech Republic, August 24-28, 2020*, LIPIcs, pages 51:1–51:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.MFCS.2020.51.

14 Ismaël Jecker, Nicolas Mazzocchi, and Petra Wolf. Decomposing permutation automata. *CONCUR 2021*, 2021. doi:10.4230/LIPIcs.CONCUR.2021.18.

15 Tao Jiang and Bala Ravikumar. Minimal NFA problems are hard. *SIAM J. Comput.*, 22(6):1117–1141, 1993. doi:10.1137/0222067.

16 Arthur B. Kahn. Topological sorting of large networks. *Commun. ACM*, 5(11):558–562, 1962. doi:10.1145/368996.369025.

17 Tsunehiko Kameda and Peter Weiner. On the state minimization of nondeterministic finite automata. *IEEE Trans. Computers*, 19(7):617–627, 1970. doi:10.1109/T-C.1970.222994.

18 Prateek Karandikar, Manfred Kufleitner, and Philipe Schnoebelen. On the index of Simon's congruence for piecewise testability. *Information Processing Letters*, 115(4):515–519, 2015. doi:10.1016/J.IPL.2014.11.008.

19 Prateek Karandikar and Philippe Schnoebelen. On the state complexity of closures and interiors of regular languages with subwords. In *International Workshop on Descriptional Complexity of Formal Systems*, pages 234–245. Springer, 2014. doi:10.1007/978-3-319-09704-6_21.

20 Prateek Karandikar and Philippe Schnoebelen. The height of piecewise-testable languages with applications in logical complexity. In *CSL 2016, Marseille, France, August 29 - September 1, 2016*, LIPIcs, pages 37:1–37:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CSL.2016.37.

21 Ondrej Klíma. Piecewise testable languages via combinatorics on words. *Discret. Math.*, 311(20):2124–2127, 2011. doi:10.1016/J.DISC.2011.06.013.

22 Orna Kupferman and Jonathan Mosheiff. Prime languages. *Information and Computation*, 240:90–107, 2015. doi:10.1016/J.IC.2014.09.010.

23 Tomás Masopust and Markus Krötzsch. Partially ordered automata and piecewise testability. *Log. Methods Comput. Sci.*, 17(2), 2021. doi:10.23638/LMCS-17(2:14)2021.

24 Alexander Okhotin. On the state complexity of scattered substrings and superstrings. *Fundamenta Informaticae*, 99(3):325–338, 2010. doi:10.3233/FI-2010-252.

25 Jean-Eric Pin and Pascal Weil. Polynomial closure and unambiguous product. In *ICALP95, Szeged, Hungary, July 10-14, 1995, Proceedings*, Lecture Notes in Computer Science, pages 348–359. Springer, 1995. doi:10.1007/3-540-60084-1_87.

26 Imre Simon. Piecewise testable events. In *Automata Theory and Formal Languages, 2nd GI Conference, Kaiserslautern, May 20-23, 1975*, Lecture Notes in Computer Science, pages 214–222. Springer, 1975. doi:10.1007/3-540-07407-4_23.

27 Daniel Alexander Spenner. Decomposing finite languages. In *MFCS 2023, Bordeaux, France, August 28 - September 1, 2023*, volume 272 of *LIPIcs*, pages 83:1–83:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.MFCS.2023.83.

28 Daniel Alexander Spenner. Deciding dfa-primality is np-hard. *CoRR*, abs/2605.07031, 2026. doi:10.48550/arXiv.2605.07031.

29 Ulrich Stern and David L. Dill. Parallelizing the Murϕ verifier. In *CAV '97, Haifa, Israel, June 22-25, 1997, Proceedings*, Lecture Notes in Computer Science, pages 256–278. Springer, 1997. doi:10.1007/3-540-63166-6_26.

30 Howard Straubing. Finite semigroup varieties of the form V*D. *Journal of Pure and Applied Algebra*, 36:53–94, 1985. doi:10.1016/S0304-3975(96)00297-6.

31 Howard Straubing and Denis Thérien. Partially ordered finite monoids and a theorem of I. Simon. *Journal of Algebra*, 119(2):393–399, 1988. doi:10.1016/0021-8693(88)90067-1.

32 Denis Thérien. Classification of finite monoids: the language approach. *Theoretical Computer Science*, 14(2):195–208, 1981. doi:10.1016/0304-3975(81)90057-8.