Commutative Regular Languages with Product-Form Minimal Automata

Stefan Hoffmann(✉)

Informatikwissenschaften, FB IV, Universität Trier, Universitätsring 15, 54296 Trier, Germany
hoffmanns@informatik.uni-trier.de

Abstract. We introduce a subclass of the commutative regular languages that is characterized by the property that the state set of the minimal deterministic automaton can be written as a certain Cartesian product. This class behaves much better with respect to the state complexity of the shuffle, for which we find the bound $2nm$ if the input languages have state complexities n and m, and the upward and downward closure and interior operations, for which we find the bound n. In general, only the bounds $(2nm)^{|\Sigma|}$ and $n^{|\Sigma|}$ are known for these operations in the commutative case. We prove different characterizations of this class and present results to construct languages from this class. Lastly, in a slightly more general setting of partial commutativity, we introduce other, related, language classes and investigate the relations between them.

Keywords: Finite automaton · State complexity · Shuffle · Upward closure · Downward closure · Commutative language · Product-form minimal automaton · Partial commutation

1 Introduction

The state complexity, as used here, of a regular language L is the minimal number of states needed in a complete deterministic automaton recognizing L. The state complexity of an operation on regular languages is the greatest state complexity of the result of this operation as a function of the (maximal) state complexities of its arguments.

Investigating the state complexity of the result of a regularity-preserving operation on regular languages, see [7] for a survey, was first initiated by Maslov in [20] and systematically started by Yu, Zhuang and Salomaa in [27].

A language is called commutative, if for each word in the language, every permutation of this word is also in the language. The class of commutative automata, which recognize commutative regular languages, was introduced in [2].

The shuffle and iterated shuffle have been introduced and studied to understand the semantics of parallel programs. This was undertaken, as it appears to be, independently by Campbell and Habermann [3], by Mazurkiewicz [22] and by Shaw [25]. They introduced *flow expressions*, which allow for sequential operators (catenation and iterated catenation) as well as for parallel operators (shuffle and iterated shuffle) to specify sequential and parallel execution traces.

The shuffle operation as a binary operation, but not the iterated shuffle, is regularity-preserving on all regular languages. The state complexity of the shuffle operation in the general cases was investigated in [1] for complete deterministic automata and in [4] for incomplete deterministic automata. The bound $2^{nm-1} + 2^{(m-1)(n-1)}(2^{m-1} - 1)(2^{n-1} - 1)$ was obtained in the former case, which is not known to be tight, and the tight bound $2^{nm} - 1$ in the latter case.

A word is a (scattered) subsequence of another word, if it can be obtained from the latter word by deleting letters. This gives a partial order, and the upward and downward closure and interior operations refer to this partial order. The upward closures are also known as shuffle ideals. The state complexity of these operations was investigated in [11–13, 19, 23]

The state complexity of the projection operation was investigated in [17, 18, 26]. In [26], the tight upper bound $3 \cdot 2^{n-2} - 1$ was shown, and in [18] the refined, and tight, bound $2^{n-1} + 2^{n-m} - 1$ was shown, where m is related to the number of unobservable transitions for the projection operator. Both results were established for incomplete deterministic automata.

In [14–17] the state complexity of these operations was investigated for commutative regular languages. The results are summarized in Table 1.

Table 1. Overview of results for commutative regular languages. The state complexities of the input languages are n and m. Also, $f(n,m) = 2^{nm-1} + 2^{(m-1)(n-1)}(2^{m-1} - 1)(2^{n-1} - 1)$ is the general bound for shuffle from [1] in case of complete automata.

Operation	Upper bound	Lower bound	References
$\pi_\Gamma(U)$, $\Gamma \subseteq \Sigma$	n	n	[14, 17]
$U \sqcup\!\sqcup V$	$\min\{(2nm)^{\lvert\Sigma\rvert}, f(n,m)\}$	$\Omega(nm)$	[1, 14, 15]
$\uparrow U$	$n^{\lvert\Sigma\rvert}$	$\Omega\left(\left(\frac{n}{\lvert\Sigma\rvert}\right)^{\lvert\Sigma\rvert}\right)$	[13, 14, 16]
$\downarrow U$	$n^{\lvert\Sigma\rvert}$	n	[14, 16]
$\circlearrowleft U$	$n^{\lvert\Sigma\rvert}$	$\Omega\left(\left(\frac{n}{\lvert\Sigma\rvert}\right)^{\lvert\Sigma\rvert}\right)$	[14, 16]
$\circlearrowright U$	$n^{\lvert\Sigma\rvert}$	n	[14, 16]
$U \cup V$, $U \cap V$	nm	Tight for each Σ	[14, 15]

Table 2. State complexity results on the subclass of commutative languages with product-form minimal automaton for input languages with state complexities n and m.

Operation	Upper bound	Lower bound	Reference
$\pi_\Gamma(U)$, $\Gamma \subseteq \Sigma$	n	n	Theorem 12
$U \sqcup\!\sqcup V$	$2nm$	$\Omega(nm)$	Theorem 12
$\uparrow U$, $\downarrow U$, $\circlearrowleft U$, $\circlearrowright U$	n	n	Theorem 12
$U \cap V$, $U \cup V$	nm	Tight for each Σ	Theorem 12

In [8] the minimal commutative automaton was introduced, which can be associated with every commutative regular language. This automaton played a crucial role in [14,15] to derive the bounds mentioned in Table 1. Here, we will investigate the subclass of those language for which the minimal commutative automaton is in fact the smallest automaton recognizing a given commutative language. For this language class, we will derive the following state complexity bounds summarized in Table 2. Additionally, we will prove other characterizations and properties of the subclass considered and relate it with other subclasses, in a more general setting, in the final chapter.

2 Preliminaries

In this section and Sect. 3, we assume that $k \geqslant 0$ denotes our alphabet size and $\Sigma = \{a_1, \ldots, a_k\}$ is our *alphabet*. We will also write a, b, c for a_1, a_2, a_3 in case of $|\Sigma| \leqslant 3$. The set Σ^* denotes the set of all finite sequences over Σ, i.e., of all *words*. The finite sequence of length zero, or the *empty word*, is denoted by ε. For a given word we denote by $|w|$ its *length*, and for $a \in \Sigma$ by $|w|_a$ the *number of occurrences of the symbol* a in w. For $a \in \Sigma$, we set $a^* = \{a\}^*$. A *language* is a subset of Σ^*. For $u \in \Sigma^*$, the *left quotient* is $u^{-1}L = \{v \in \Sigma^* \mid uv \in L\}$ and the *right quotient* is $Lu^{-1} = \{v \in \Sigma^* \mid vu \in L\}$.

The *shuffle operation*, denoted by $\sqcup\!\sqcup$, is defined by

$$u \sqcup\!\sqcup v = \{w \in \Sigma^* \mid w = x_1 y_1 x_2 y_2 \cdots x_n y_n \textit{ for some words}$$
$$x_1, \ldots, x_n, y_1, \ldots, y_n \in \Sigma^* \textit{ such that } u = x_1 x_2 \cdots x_n \textit{ and } v = y_1 y_2 \cdots y_n\},$$

for $u, v \in \Sigma^*$ and $L_1 \sqcup\!\sqcup L_2 := \bigcup_{x \in L_1, y \in L_2} (x \sqcup\!\sqcup y)$ for $L_1, L_2 \subseteq \Sigma^*$. If $L_1, \ldots, L_n \subseteq \Sigma^*$, we set $\bigsqcup\!\bigsqcup_{i=1}^n L_i = L_1 \sqcup\!\sqcup \ldots \sqcup\!\sqcup L_n$.

Let $\Gamma \subseteq \Sigma$. The *projection homomorphism* $\pi_\Gamma : \Sigma^* \to \Gamma^*$ is given by $\pi_\Gamma(x) = x$ for $x \in \Gamma$ and $\pi_\Gamma(x) = \varepsilon$ for $x \notin \Gamma$ and extended to Σ^* by $\pi_\Gamma(\varepsilon) = \varepsilon$ and $\pi_\Gamma(wx) = \pi_\Gamma(w)\pi_\Gamma(x)$ for $w \in \Sigma^*$ and $x \in \Sigma$. As a shorthand, we set, with respect to a given naming $\Sigma = \{a_1, \ldots, a_k\}$, $\pi_j = \pi_{\{a_j\}}$. Then $\pi_j(w) = a_j^{|w|_{a_j}}$.

A language $L \subseteq \Sigma^*$ is *commutative*, if, for $u, v \in \Sigma^*$ such that $|v|_x = |u|_x$ for every $x \in \Sigma$, we have $u \in L$ if and only if $v \in L$, i.e., L is closed under permutation of letters in words from L.

A quintuple $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ is a *finite deterministic and complete automaton* (DFA), where Σ is the *input alphabet*, Q the *finite set of states*, $q_0 \in Q$ the *start state*, $F \subseteq Q$ the set of *final states* and $\delta : Q \times \Sigma \to Q$ is the *totally defined state transition function*. Here, we do not consider incomplete automata. The transition function $\delta : Q \times \Sigma \to Q$ extends to a transition function on words $\delta^* : Q \times \Sigma^* \to Q$ by setting $\delta^*(q, \varepsilon) := q$ and $\delta^*(q, wa) := \delta(\delta^*(q, w), a)$ for $q \in Q$, $a \in \Sigma$ and $w \in \Sigma^*$. In the remainder, we drop the distinction between both functions and also denote this extension by δ. The language *recognized* by an automaton $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ is $L(\mathcal{A}) = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$. A language $L \subseteq \Sigma^*$ is called *regular* if $L = L(\mathcal{A})$ for some finite automaton $\mathcal{A}$.

The *Nerode right-congruence* with respect to $L \subseteq \Sigma^*$ is defined, for $u, v \in \Sigma^*$, by $u \equiv_L v$ if and only if $\forall x \in \Sigma^* : ux \in L \Leftrightarrow vx \in L$. The equivalence class of $w \in \Sigma^*$ is denoted by $[w]_{\equiv_L} = \{x \in \Sigma^* \mid x \equiv_L w\}$. A language is regular if and only if the above right-congruence has finite index, and it can be used to define the *minimal deterministic automaton* $\mathcal{A}_L = (\Sigma, Q_L, \delta_L, [\varepsilon]_{\equiv_L}, F_L)$ with $Q_L = \{[u]_{\equiv_L} \mid u \in \Sigma^*\}$, $\delta_L([w]_{\equiv_L}, a) = [wa]_{\equiv_L}$ and $F_L = \{[u]_{\equiv_L} \mid u \in L\}$. Let $L \subseteq \Sigma^*$ be regular with minimal automaton $\mathcal{A}_L = (\Sigma, Q_L, \delta_L, [\varepsilon]_{\equiv_L}, F_L)$. The number $|Q_L|$ is called the *state complexity* of L and denoted by $\mathrm{sc}(L)$. The *state complexity of a regularity-preserving operation* on a class of regular languages is the greatest state complexity of the result of this operation as a function of the (maximal) state complexities for argument languages from the class.

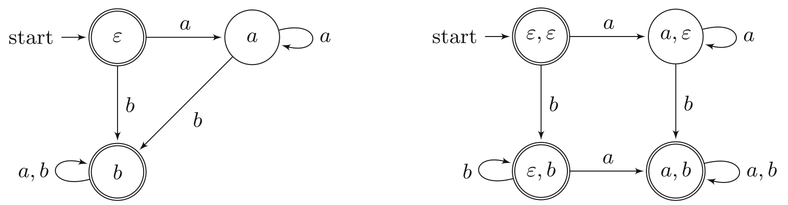

Fig. 1. The minimal deterministic automaton (left) and the minimal commutative automaton (right) of the language $\{w \in \Sigma^* \mid |w|_a = 0 \text{ or } |w|_b > 0\}$.

Given two automata $\mathcal{A} = (\Sigma, S, \delta, s_0, F)$ and $\mathcal{B} = (\Sigma, T, \mu, t_0, E)$, an *automaton homomorphism* $h : S \to T$ is a map between the state sets such that for each $a \in \Sigma$ and state $s \in S$ we have $h(\delta(s, a)) = \mu(h(s), a)$, $h(s_0) = t_0$ and $h^{-1}(E) = F$. If $h : S \to T$ is surjective, then $L(\mathcal{B}) = L(\mathcal{A})$. A bijective homomorphism between automata $\mathcal{A}$ and $\mathcal{B}$ is called an *isomorphism*, and the two automata are said to be isomorphic.

The *minimal commutative automaton* was introduced in [8] to investigate the learnability of commutative languages. In [14,15] this construction was used to define the index and period vector and in the derivation of the state complexity bounds mentioned in Table 1.

Definition 1 (minimal commutative aut.). *Let $L \subseteq \Sigma^*$ be regular. The* minimal commutative automaton *for L is $\mathcal{C}_L = (\Sigma, S_1 \times \ldots \times S_k, \delta, s_0, F)$ with*

$$S_j = \{[a_j^m]_{\equiv_L} : m \geqslant 0\}, \quad F = \{([\pi_1(w)]_{\equiv_L}, \ldots, [\pi_k(w)]_{\equiv_L}) : w \in L\}$$

and $\delta((s_1, \ldots, s_j, \ldots, s_k), a_j) = (s_1, \ldots, \delta_j(s_j, a_j), \ldots, s_k)$ with one-letter transitions $\delta_j([a_j^m]_{\equiv_L}, a_j) = [a_j^{m+1}]_{\equiv_L}$ for $j = 1, \ldots, k$ and $s_0 = ([\varepsilon]_{\equiv_L}, \ldots, [\varepsilon]_{\equiv_L})$.

In [8], the next result was shown.

Theorem 2 (Gómez and Alvarez [8]). *Let $L \subseteq \Sigma^*$ be a commutative regular language. Then, $L = L(\mathcal{C}_L)$.*

In general the minimal commutative automaton is not equal to the minimal deterministic and complete automaton for a regular commutative language L, see Example 1.

Example 1. For $L = \{w \in \Sigma^* \mid |w|_a = 0 \text{ or } |w|_b > 0\}$ with $\Sigma = \{a, b\}$ the minimal deterministic and complete automaton and the minimal commutative automaton are not the same, see Fig. 1. This language is from [8]. In fact, the difference can get quite large, as shown by $L_p = \{w \in \Sigma^* \mid \sum_{j=1}^{k} j \cdot |w|_{a_j} \equiv 0 \pmod{p}\}$ for a prime $p > k$. Here, $\mathrm{sc}(L_p) = p$, but $\mathcal{C}_{L_p}$ has p^k states.

The next definition from [14,15] generalizes the notion of a cyclic and non-cyclic part for unary automata [24], and the notion of periodic language [6,14,15].

Definition 3 (index and period vector). *The* index vector $(i_1, \ldots, i_k)$ *and* period vector $(p_1, \ldots, p_k)$ *for a commutative regular language $L \subseteq \Sigma^*$ with minimal commutative automaton $\mathcal{C}_L = (\Sigma, S_1 \times \ldots \times S_k, \delta, s_0, F)$ are the unique minimal numbers such that $\delta(s_0, a_j^{i_j}) = \delta(s_0, a_j^{i_j + p_j})$ for all $j \in \{1, \ldots, k\}$.*

Note that, in Definition 3, we have, for all $j \in \{1, \ldots, k\}$, $|S_j| = i_j + p_j$. Also note that for unary languages, i.e., if $|\Sigma| = 1$, $\mathcal{C}_L$ equals $\mathcal{A}_L$ and $i_1 + p_1$ equals the number of states of the minimal automaton.

Example 2. Let $L = (aa)^* ⧢ (bb)^* \cup (aaaa)^* ⧢ b^*$. Then $(i_1, i_2) = (0, 0)$, $(p_1, p_2) = (4, 2)$, $\pi_1(L) = (aa)^*$ and $\pi_2(L) = b^*$.

Let $u, v \in \Sigma^*$. Then, u is a *subsequence*[1] of v, denoted by $u \preccurlyeq v$, if and only if $v \in u ⧢ \Sigma^*$. The thereby given order is called the *subsequence order*. Let $L \subseteq \Sigma^*$. Then, we define (1) the *upward closure* $\uparrow L = L ⧢ \Sigma^* = \{u \in \Sigma^* : \exists v \in L : v \preccurlyeq u\}$; (2) the *downward closure* $\downarrow L = \{u \in \Sigma^* : u ⧢ \Sigma^* \cap L \neq \emptyset\} = \{u \in \Sigma^* : \exists v \in L : u \preccurlyeq v\}$; (3) the *upward interior*, denoted by $\circlearrowleft L$, as the largest upward-closed set in L, i.e. the largest subset $U \subseteq L$ such that $\uparrow U = U$ and (4) the *downward interior*, denoted by $\circlearrowright L$, as the largest downward-closed set in L, i.e., the largest subset $U \subseteq L$ such that $\downarrow U = U$. We have $\circlearrowright L = \Sigma^* \setminus \uparrow(\Sigma^* \setminus L)$ and $\circlearrowleft L = \Sigma^* \setminus \downarrow(\Sigma^* \setminus L)$.

The following two results, which will be needed later, are from [14,15].

Theorem 4. *Let $U, V \subseteq \Sigma^*$ be commutative regular languages with index and period vectors $(i_1, \ldots, i_k), (j_1, \ldots, j_k)$ and $(p_1, \ldots, p_k), (q_1, \ldots, q_k)$. Then, the index vector of $U ⧢ V$ is at most*

$$(i_1 + j_1 + \mathrm{lcm}(p_1, q_1) - 1, \ldots, i_k + j_k + \mathrm{lcm}(p_k, q_k) - 1)$$

and the period vector is at most $(\mathrm{lcm}(p_1, q_1), \ldots, \mathrm{lcm}(p_k, q_k))$. *So,* $\mathrm{sc}(U ⧢ V) \leqslant \prod_{l=1}^{k}(i_l + j_l + 2 \cdot \mathrm{lcm}(p_l, q_l) - 1)$.

Theorem 5. *Let $\Sigma = \{a_1, \ldots, a_k\}$. Suppose $L \subseteq \Sigma^*$ is commutative and regular with index vector $(i_1, \ldots, i_k)$ and period vector $(p_1, \ldots, p_k)$. Then,* $\max\{\mathrm{sc}(\uparrow L), \mathrm{sc}(\downarrow L), \mathrm{sc}(\circlearrowleft L), \mathrm{sc}(\circlearrowright L)\} \leqslant \prod_{j=1}^{k}(i_j + p_j)$.

[1] Also called a *scattered subword* in the literature [11,19].

3 Product-Form Minimal Automata

As shown in Example 1, the minimal automaton, in general, does not equal the minimal commutative automaton. Here, we introduce the class of commutative regular languages for which both are isomorphic. The corresponding commutative languages are called *languages with a minimal automaton of product-form*, as the minimal commutative automaton is built with the Cartesian product.

Definition 6 (languages with product-form minimal automaton). *A commutative and regular language $L \subseteq \Sigma^*$ is said to have a* minimal automaton of product-form*, if $\mathcal{C}_L$ is isomorphic to $\mathcal{A}_L$.*

If $|\Sigma| = 1$, we see easily that $\mathcal{C}_L$ is the minimal deterministic and complete automaton.

Proposition 7. *If $|\Sigma| = 1$, then each commutative and regular $L \subseteq \Sigma^*$ has a minimal automaton of product-form. More generally, if $L \subseteq \{a\}^*$, then $L \sqcup\!\sqcup (\Sigma \backslash \{a\})^*$ has a minimal automaton of product-form.*

Apart from the unary languages, we give another example of a language with minimal automaton of product-form next.

Example 3. Let $L = (aa)^* \sqcup\!\sqcup (bb)^* \cup (aaa)^* \sqcup\!\sqcup b(bb)^*$ over $\Sigma = \{a, b\}$. See Fig. 2 for the minimal commutative automaton. Here, the minimal commutative automaton equals the minimal automaton.

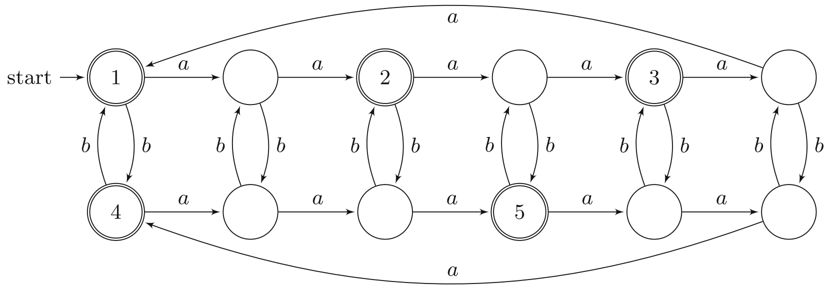

Fig. 2. $\mathcal{C}_L$ for $L = (aa)^* \sqcup\!\sqcup (bb)^* \cup (aaa)^* \sqcup\!\sqcup b(bb)^*$. Here $\mathcal{C}_L$ is isomorphic to $\mathcal{A}_L$.

However, the next proposition gives a strong necessary criterion for a commutative language to have a minimal automaton of product-form.

Proposition 8. *If $L \subseteq \Sigma^*$ is commutative and regular with a minimal automaton of product-form, then $|\{x \in \Sigma \mid \pi_{\{x\}}(L)$ is finite $\}| \leqslant 1$. So, $\pi_\Gamma(L)$ is infinite for $|\Gamma| \geqslant 2$, in particular no finite language over an at least binary alphabet is in this class.*

For example, $L = \{\varepsilon\}$ over Σ does not have a minimal automaton of product-form if $|\Sigma| > 1$. Recall that the minimal automaton, as defined here, is always complete. Note that the converse of Proposition 8 is not true, as shown by aa^* over $\Sigma = \{a, b\}$.

In the following statement, we give alternative characterizations for commutative languages with minimal automata of product-form.

Theorem 9. *Let $L \subseteq \Sigma^*$ be a commutative regular language with index vector $(i_1, \ldots, i_k)$ and period vector $(p_1, \ldots, p_k)$. The following are equivalent:*

1. *the minimal automaton has product-form;*
2. $\mathrm{sc}(L) = \prod_{j=1}^{k}(i_j + p_j)$*;*
3. *$u \equiv_L v$ implies $\forall a \in \Sigma : a^{|u|_a} \equiv_L a^{|v|_a}$;*
4. *$u \equiv_L v$ if and only if $\forall a \in \Sigma : a^{|u|_a} \equiv_L a^{|v|_a}$.*

Next, we give a way to construct commutative regular languages with minimal automata of product-form.

Lemma 10. *Let $\Sigma = \{a_1, \ldots, a_k\}$ and, for $j \in \{1, \ldots, k\}$, $L_j \subseteq \{a_j\}^*$ be regular and infinite with index i_j and period p_j. Then,* $\mathrm{sc}\left(\bigsqcup_{j=1}^{k} L_j\right) = \prod_{j=1}^{k} \mathrm{sc}(L_j) = \prod_{j=1}^{k}(i_j + p_j)$ *and $\bigsqcup_{j=1}^{k} L_j$ has index vector $(i_1, \ldots, i_k)$ and period vector $(p_1, \ldots, p_k)$. With Theorem 9, $\bigsqcup_{j=1}^{k} L_j$ has a product-form minimal automaton.*

In the next theorem and the following remark, we investigate closure properties of the class in question.

Theorem 11. *The class of commutative regular languages with minimal automata of product-form is closed under left and right quotients and complementation. It is not closed under union, intersection and projection.*

Remark 1. We have $a \sqcup\!\sqcup\, b^* \cap a^* \sqcup\!\sqcup\, b = a \sqcup\!\sqcup\, b$, showing, using Proposition 7 and 8, that this class is not closed under intersection and by DeMorgan's laws, as we have closure under complementation, we also cannot have closure under union. Also, $L = aa^* \sqcup\!\sqcup\, bb^* \sqcup\!\sqcup\, cc^* \cup bb^* \sqcup\!\sqcup\, a^* \cup b^*$ has a minimal automaton of product-form, but $\pi_{\{a,b\}}(L) = bb^* \sqcup\!\sqcup\, a^* \cup b^*$ is the language from Example 1. So, this class is also not closed under projection.

Theorem 12. *Let $U, V \subseteq \Sigma^*$ be commutative regular languages with product-form minimal automata with* $\mathrm{sc}(U) = n$ *and* $\mathrm{sc}(V) = m$*.*

1. *We have* $\mathrm{sc}(U \sqcup\!\sqcup\, V) \leqslant 2nm$ *if $|\Sigma| > 1$ and* $\mathrm{sc}(U \sqcup\!\sqcup\, V) \leqslant nm$ *if $|\Sigma| = 1$. Furthermore, for any Σ, there exist U, V as above such that* $nm \leqslant \mathrm{sc}(U \sqcup\!\sqcup\, V)$*.*
2. *In the worst case, n states are sufficient and necessary for a DFA to recognize $\uparrow U$. Similarly for the downward closure and interior operations.*
3. *In the worst case, n states are sufficient and necessary for a DFA to recognize the projection of U.*

4. *In the worst case, nm states are sufficient and necessary for a DFA to recognize $U \cap V$ or $U \cup V$.*

Remark 2. I do not know if the bound $2nm$ stated in Theorem 1 for the shuffle operation is tight, but the next example shows that if we have a binary alphabet, we can find commutative languages with state complexities n and m and product-form minimal automata whose shuffle needs an automaton with strictly more than nm states. A similar construction works for more than two letters. Let $p, q > 11$ be two coprime numbers. Set $U = a \sqcup\!\sqcup b^{p-1}(b^p)^* \cup a^{p-1}(a^p)^* \sqcup\!\sqcup bb^{p-1}(b^p)^*$ and $V = b^{q-1}(b^q)^* \cup a^{q-1}(a^q)^* \sqcup\!\sqcup bbb^{q-1}(b^q)^*$. Then, using that shuffle distributes over union and a number-theoretical result from [27, Lemma 5.1], we find

$$U \sqcup\!\sqcup V = a \sqcup\!\sqcup W \cup a^{p-1}(a^p)^* \sqcup\!\sqcup bW \cup \\ a^q(a^q)^* \sqcup\!\sqcup bbW \cup a^{q-1+p-1}(a^p)^*(a^q)^* \sqcup\!\sqcup bbbW,$$

where $a^{q-1+p-1}(a^p)^*(a^q)^* = F \cup a^{pq-1}a^*$ for some finite set $F \subseteq \{\varepsilon, a, \ldots, a^{pq-3}\}$ and $W = E \cup b^{pq-1}b^*$ for some $E \subseteq \{\varepsilon, b, \ldots, b^{pq-3}\}$. Note that by [27, Lemma 5.1] we have $a^{pq-2} \sqcup\!\sqcup bbbW \cap U \sqcup\!\sqcup V = \emptyset$. All languages involved have a product-form minimal automaton. The minimal automaton for U has $(2+p)\cdot(1+p)$ states, the minimal automaton for V has $(1+q)\cdot(q+2)$ states and that for $U \sqcup\!\sqcup V$ has $2pq \cdot (pq+3)$ states. As $(p-11)(q-11) > 0$ we can deduce $(1+p)(2+p)(1+q)(2+q) < 2(pq)^2 < 2pq(pq+3)$.

4 Partial Commutativity and Other Subclasses

A *partial commutation* on Σ is a symmetric and irreflexive relation $I \subseteq \Sigma \times \Sigma$, often called the *independence relation.* Of interest is the *congruence* $\sim_I$ generated on Σ^* by the relation $\{(ab, ba) \mid (a, b) \in I\}$. A language $L \subseteq \Sigma^*$ is *closed under I-commutation* if $u \in L$ and $u \sim_I v$ implies $v \in L$. If $I = \{(a, b) \in \Sigma \times \Sigma \mid a \neq b\}$, then the languages closed under I-commutation are precisely the commutative languages.

Languages closed under some partial commutation relation have been extensively studied, see [10], also for further references, and in particular with relation to (Mazurkiewicz) trace theory [5,10,21], a formalism to describe the execution histories of concurrent programs.

Here, we will focus on the case that $(\Sigma \times \Sigma)\backslash I$ is transitive, i.e., if $u \not\sim_I v$ and $v \not\sim_I w$ implies $u \not\sim_I w$. In this case, $(\Sigma \times \Sigma)\backslash I$ is an equivalence relation and we will write $\Sigma_1, \ldots, \Sigma_k$ for the different equivalence classes.

The reason to focus on this particular generalization is, as we will see later, that the definition of the minimal commutative automaton transfers to this more general setting without much difficulty.

To ease the notation, if we have a partial commutation relation as above with a corresponding partition $\Sigma = \Sigma_1 \cup \ldots \Sigma_k$ of the alphabet, we also write $\mathcal{L}_{\Sigma_1,\ldots,\Sigma_k}$ for the *class of languages closed under this partial commutation.* Then, as is easily seen, we have $L \in \mathcal{L}_{\Sigma_1,\ldots,\Sigma_k}$ if and only if, for $x \in \Sigma_i$, $y \in \Sigma_j$

$(i \neq j)$ and each $u, v \in \Sigma^*$ we have $uxyv \in L \Leftrightarrow uyxv \in L$. For example, L is commutative if and only if $L \in \mathcal{L}_{\{a_1\},\ldots,\{a_k\}}$ for $\Sigma = \{a_1, \ldots, a_k\}$.

4.1 The Canonical Automaton

Here, we generalize our notion of commutative minimal automaton, Definition 1, to have uniform recognition devices for languages in $\mathcal{L}_{\Sigma_1,\ldots,\Sigma_k}$.

Definition 13. *Let $\Sigma = \Sigma_1 \cup \ldots \cup \Sigma_k$ be a partition and $L \subseteq \Sigma^*$. Set $\mathcal{C}_{L,\Sigma_1,\ldots,\Sigma_k} = (\Sigma, S_1 \times \ldots \times S_k, \delta, s_0, F)$ with, for $i \in \{1, \ldots, k\}$, $S_i = \{[u]_{\equiv_L} \mid u \in \Sigma_i^*\}$, $F = \{([\pi_{\Sigma_1}(u)]_{\equiv_L}, \ldots, [\pi_{\Sigma_k}(u)]_{\equiv_L}) \mid u \in L\}$, $s_0 = ([\varepsilon]_{\equiv_L}, \ldots, [\varepsilon]_{\equiv_L})$ and, for $x \in \Sigma_i$,*

$$\delta(([u_1]_{\equiv_L}, \ldots, [u_i]_{\equiv_L}, \ldots, [u_k]_{\equiv_L}), x) = ([u_1]_{\equiv_L}, \ldots, [u_i x]_{\equiv_L}, \ldots, [u_k]_{\equiv_L})$$

with words $u_j \in \Sigma_j^*$, $j \in \{1, \ldots, k\}$. This is called the canonical automaton *for the given L with respect to $\Sigma = \Sigma_1 \cup \ldots \cup \Sigma_k$.*

Next, we show that the canonical automata recognize precisely the languages in $\mathcal{L}_{\Sigma_1,\ldots,\Sigma_k}$. Note that we have dropped the assumption of regularity of L.

Theorem 14. *Let $L \subseteq \Sigma^*$ and $\Sigma = \Sigma_1 \cup \ldots \cup \Sigma_k$ be a partition. Then,*

1. *$L \subseteq L(\mathcal{C}_{L,\Sigma_1,\ldots,\Sigma_k})$ and $L(\mathcal{C}_{L,\Sigma_1,\ldots,\Sigma_k}) \in \mathcal{L}_{\Sigma_1,\ldots,\Sigma_k}$.*
2. *$L = L(\mathcal{C}_{L,\Sigma_1,\ldots,\Sigma_k}) \Leftrightarrow L \in \mathcal{L}_{\Sigma_1,\ldots,\Sigma_k}$.*
3. *Let $L \in \mathcal{L}_{\Sigma_1,\ldots,\Sigma_k}$. Then L is regular if and only if $\mathcal{C}_{L,\Sigma_1,\ldots,\Sigma_k}$ is finite.*

Also, used in defining a subclass in the next subsection, we will derive a canonical automaton for certain projected languages from $\mathcal{C}_{L,\Sigma_1,\ldots,\Sigma_k}$. Essentially, the next definition and proposition mean that if we only use one "coordinate" of $\mathcal{C}_{L,\Sigma_1,\ldots,\Sigma_k}$, then this recognizes a projection of L.

Definition 15. *Let $i \in \{1, \ldots, k\}$ and $L \in \mathcal{L}_{\Sigma_1,\ldots,\Sigma_k}$. The* canonical projection automaton *(for Σ_i) is $\mathcal{C}_{L,\Sigma_i} = (\Sigma_i, S_i, \delta_i, [\varepsilon]_{\equiv_L}, F_i)$ with $S_i = \{[u]_{\equiv_L} \mid u \in \Sigma_i^*\}$, $\delta_i([u]_{\equiv_L}, x) = [ux]_{\equiv_L}$ for $x \in \Sigma_i$ and $F_i = \{[\pi_{\Sigma_i}(u)]_{\equiv_L} \mid u \in L\}$.*

Proposition 16. *Let $L \in \mathcal{L}_{\Sigma_1,\ldots,\Sigma_k}$. Then, for $i \in \{1, \ldots, k\}$, $\pi_{\Sigma_i}(L) = L(\mathcal{C}_{L,\Sigma_i})$.*

4.2 Subclasses in $\mathcal{L}_{\Sigma_1,\ldots,\Sigma_k}$

Here, we investigate several subclasses of $\mathcal{L}_{\Sigma_1,\ldots,\Sigma_k}$. Recall that, for $L \subseteq \Sigma^*$, the minimal automaton of L is denoted by $\mathcal{A}_L$.

Definition 17. *Let $\Sigma = \Sigma_1 \cup \ldots \cup \Sigma_k$ be a partition. Then, define the following classes of languages.*

$$\mathcal{L}_1 = \{L \mid \mathcal{C}_{L,\Sigma_1,\ldots,\Sigma_k} \textit{ has a single final state and } L = L(\mathcal{C}_{L,\Sigma_1,\ldots,\Sigma_k}).\ \},$$

$$\mathcal{L}_2 = \left\{ L \mid L = \bigsqcup_{i=1}^{k} \pi_{\Sigma_i}(L) \right\},$$

$$\mathcal{L}_3 = \{L \mid L = L(\mathcal{C}_{L,\Sigma_1,\ldots,\Sigma_k}), \forall i \in \{1, \ldots, k\} : \mathcal{A}_{\pi_{\Sigma_i}(L)} \textit{ is isomorphic to } \mathcal{C}_{L,\Sigma_i}\},$$

$$\mathcal{L}_4 = \{L \mid \mathcal{A}_L \textit{ is isomorphic to } \mathcal{C}_{L,\Sigma_1,\ldots,\Sigma_k}\}.$$

First, we show that these are in fact subclasses of $\mathcal{L}_{\Sigma_1,\ldots,\Sigma_k}$.

Proposition 18. *Let $\Sigma = \Sigma_1 \cup \ldots \cup \Sigma_k$ be a partition. For each $i \in \{1,2,3,4\}$ we have $\mathcal{L}_i \subseteq \mathcal{L}_{\Sigma_1,\ldots,\Sigma_k}$.*

Remark 3. Regarding $\mathcal{L}_1$, note that there exist languages $L = L(\mathcal{C}_{L,\Sigma_1,\ldots,\Sigma_k})$ such that the minimal automaton has a single final state, but $\mathcal{C}_{L,\Sigma_1,\ldots,\Sigma_k}$ has more than one final state. For example, $L = \{w \in \{a,b\}^* \mid |w|_a > 0 \text{ or } |w|_b > 0\}$. However, if $\mathcal{C}_{L,\Sigma_1,\ldots,\Sigma_k}$ has a single final state, then the minimal automata also has only a single final state.

Example 4. Let $\Sigma = \Sigma_1 \cup \Sigma_2$ with $\Sigma_1 = \{a\}$ and $\Sigma_2 = \{b\}$. Set $L = (aa(aaa)^* \sqcup\!\sqcup\, bb(bbb)^*) \cup (a(aaa)^* \sqcup\!\sqcup\, b(bbb)^*)$. Then $L \in (\mathcal{L}_3 \cap \mathcal{L}_4) \backslash \mathcal{L}_2$.

Example 5. Set $L = (a(aaa)^* \sqcup\!\sqcup\, b) \cup aa(aaa)^*$. Then $L \in \mathcal{L}_3 \backslash \mathcal{L}_4$.

The languages in $\mathcal{L}_1$ arise in connection with the canonical automaton.

Proposition 19. *Let $L \in \mathcal{L}_{\Sigma_1,\ldots,\Sigma_k}$ and $\mathcal{C}_{L,\Sigma_1,\ldots,\Sigma_k} = (\Sigma, S_1 \times \ldots \times S_k, \delta, s_0, F)$. Then, for all $s \in S_1 \times \ldots \times S_k$, $\{w \in \Sigma^* \mid \delta(s_0, w) = s\} \in \mathcal{L}_1$.*

Next, we give alternative characterization for $\mathcal{L}_2, \mathcal{L}_3$ and $\mathcal{L}_4$.

Theorem 20. *Let $L \in \mathcal{L}_{\Sigma_1,\ldots,\Sigma_k}$. Then,*

1. $L \in \mathcal{L}_2$ if and only if, for each $w \in \Sigma^*$, the following is true:

$$w \in L \Leftrightarrow \forall i \in \{1,\ldots,k\} : \pi_{\Sigma_i}(w) \in \pi_{\Sigma_i}(L);$$

2. $L \in \mathcal{L}_3$ if and only if, for all $i \in \{1,\ldots,k\}$ and $u \in \Sigma_i^*$, we have

$$[u]_{\equiv_L} \cap \Sigma_i^* = [u]_{\equiv_{\pi_{\Sigma_i}(L)}} \cap \Sigma_i^*;$$

3. $L \in \mathcal{L}_4$ if and only if, for each $u, v \in \Sigma^*$,

$$u \equiv_L v \Leftrightarrow \forall i \in \{1,\ldots,k\} : \pi_{\Sigma_i}(u) \equiv_L \pi_{\Sigma_i}(v).$$

Example 6. Let L_1 be the language from Example 3. Set $L_2 = a_1 \sqcup\!\sqcup\, a_2 = \{a_1a_2, a_2a_1\}$. Both of their letters commute for the partition $\{a_1, a_2\} = \{a_1\} \cup \{a_2\}$. Then, $L_1 \in \mathcal{L}_4 \backslash \mathcal{L}_3$ and $L_2 \in \mathcal{L}_1 \backslash \mathcal{L}_4$.

Finally, in Theorem 21, we establish inclusion relations, which are all proper, between $\mathcal{L}_1, \mathcal{L}_2$ and $\mathcal{L}_3$, also see Fig. 3.

Theorem 21. *We have $\mathcal{L}_1 \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_3$.*

Remark 4. Theorem 21 and Example 6 show that $\mathcal{L}_4$ is incomparable to each of the other language classes with respect to inclusion.

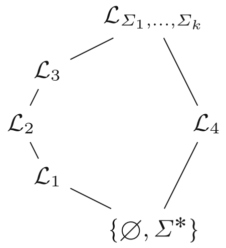
$\mathcal{L}_{\Sigma_1,\ldots,\Sigma_k}$
$\mathcal{L}_3$
$\mathcal{L}_2$
$\mathcal{L}_4$
$\mathcal{L}_1$
$\{\emptyset, \Sigma^*\}$

Fig. 3. Inclusion relations between the language classes.

5 Conclusion

The language class of commutative regular languages with minimal automata of product-form behaves well with respect to the descriptional complexity measure of state complexity for certain operations, see Table 2, and Lemma 10 allows us to construct infinitely many commutative regular languages with product-form minimal automaton. The investigation started could be carried out for other operations and measures of descriptional complexity as well. Likewise, as done in [8,9] for commutative and more general partial commutativity conditions, it might be interesting if the learning algorithms given there could be improved for the language class introduced.

Lastly, if the bound $2nm$ for shuffle is tight is an open problem. Remark 2 shows that the bound nm is not sufficient, however, giving an infinite family of commutative regular languages with minimal automata of product-form attaining the bound $2nm$ for shuffle is an open problem.

Acknowledgement. I thank the anonymous referees of [14] (the extended version of [15]), whose feedback also helped in the present work. I also sincerely thank the referees of the present submission, which helped me alot in identifying unclear or ungrammatical formulations and a missing definition.

References

1. Brzozowski, J., Jirásková, G., Liu, B., Rajasekaran, A., Szykuła, M.: On the state complexity of the shuffle of regular languages. In: Câmpeanu, C., Manea, F., Shallit, J. (eds.) DCFS 2016. LNCS, vol. 9777, pp. 73–86. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41114-9_6
2. Brzozowski, J.A., Simon, I.: Characterizations of locally testable events. Discret. Math. **4**(3), 243–271 (1973)
3. Campbell, R.H., Habermann, A.N.: The specification of process synchronization by path expressions. In: Gelenbe, E., Kaiser, C. (eds.) OS 1974. LNCS, vol. 16, pp. 89–102. Springer, Heidelberg (1974). https://doi.org/10.1007/BFb0029355
4. Câmpeanu, C., Salomaa, K., Yu, S.: Tight lower bound for the state complexity of shuffle of regular languages. J. Autom. Lang. Comb. **7**(3), 303–310 (2002)

5. Diekert, V., Rozenberg, G. (eds.): The Book of Traces. World Scientific, River Edge (1995)
6. Ehrenfeucht, A., Haussler, D., Rozenberg, G.: On regularity of context-free languages. Theor. Comput. Sci. **27**, 311–332 (1983)
7. Gao, Y., Moreira, N., Reis, R., Yu, S.: A survey on operational state complexity. J. Autom. Lang. Comb. **21**(4), 251–310 (2017)
8. Cano Gómez, A., Álvarez, G.I.: Learning commutative regular languages. In: Clark, A., Coste, F., Miclet, L. (eds.) ICGI 2008. LNCS (LNAI), vol. 5278, pp. 71–83. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88009-7_6
9. Cano Gómez, A.: Inferring regular trace languages from positive and negative samples. In: Sempere, J.M., García, P. (eds.) ICGI 2010. LNCS (LNAI), vol. 6339, pp. 11–23. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15488-1_3
10. Gómez, A.C., Guaiana, G., Pin, J.: Regular languages and partial commutations. Inf. Comput. **230**, 76–96 (2013)
11. Gruber, H., Holzer, M., Kutrib, M.: The size of Higman-Haines sets. Theor. Comput. Sci. **387**(2), 167–176 (2007)
12. Gruber, H., Holzer, M., Kutrib, M.: More on the size of Higman-Haines sets: effective constructions. Fundam. Informaticae **91**(1), 105–121 (2009)
13. Héam, P.: On shuffle ideals. RAIRO Theor. Inform. Appl. **36**(4), 359–384 (2002)
14. Hoffmann, S.: Commutative regular languages - properties and state complexity. Inf. Comput. (submitted)
15. Hoffmann, S.: Commutative regular languages – properties and state complexity. In: Ćirić, M., Droste, M., Pin, J.É. (eds.) CAI 2019. LNCS, vol. 11545, pp. 151–163. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21363-3_13
16. Hoffmann, S.: State complexity investigations on commutative languages - the upward and downward closure, commutative aperiodic and commutative group languages. In: Han, Y.-S., Ko, S.-K. (eds.) DCFS 2021, LNCS 13037, pp. 64–75. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-93489-7_6
17. Hoffmann, S.: State complexity of projection on languages recognized by permutation automata and commuting letters. In: Moreira, N., Reis, R. (eds.) DLT 2021. LNCS, vol. 12811, pp. 192–203. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81508-0_16
18. Jirásková, G., Masopust, T.: On a structural property in the state complexity of projected regular languages. Theor. Comput. Sci. **449**, 93–105 (2012)
19. Karandikar, P., Niewerth, M., Schnoebelen, P.: On the state complexity of closures and interiors of regular languages with subwords and superwords. Theor. Comput. Sci. **610**, 91–107 (2016)
20. Maslov, A.N.: Estimates of the number of states of finite automata. Dokl. Akad. Nauk SSSR **194**(6), 1266–1268 (1970)
21. Mazurkiewicz, A.: Concurrent program schemes and their interpretations. Technical report, DAIMI Report Series 6(78) (1977)
22. Mazurkiewicz, A.: Parallel recursive program schemes. In: Bečvář, J. (ed.) MFCS 1975. LNCS, vol. 32, pp. 75–87. Springer, Heidelberg (1975). https://doi.org/10.1007/3-540-07389-2_183
23. Okhotin, A.: On the state complexity of scattered substrings and superstrings. Fundam. Informaticae **99**(3), 325–338 (2010)
24. Pighizzini, G., Shallit, J.: Unary language operations, state complexity and Jacobsthal's function. Int. J. Found. Comput. Sci. **13**(1), 145–159 (2002)
25. Shaw, A.C.: Software descriptions with flow expressions. IEEE Trans. Softw. Eng. **4**, 242–254 (1978)

26. Wong, K.: On the complexity of projections of discrete-event systems. In: Proceedings of WODES 1998, Cagliari, Italy, pp. 201–206 (1998)
27. Yu, S., Zhuang, Q., Salomaa, K.: The state complexities of some basic operations on regular languages. Theor. Comput. Sci. **125**(2), 315–328 (1994)