# State Complexity of Projection on Languages Recognized by Permutation Automata and Commuting Letters

Stefan Hoffmann(✉)

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

**Abstract.** The projected language of a general deterministic automaton with $n$ states is recognizable by a deterministic automaton with $2^{n-1}+2^{n-m}-1$ states, where $m$ denotes the number of states incident to unobservable non-loop transitions, and this bound is best possible. Here, we derive the tight bound $2^{n-\lceil \frac{m}{2} \rceil}-1$ for permutation automata. For a state-partition automaton with $n$ states (also called automata with the observer property) the projected language is recognizable with $n$ states. Up to now, these, and finite languages projected onto unary languages, were the only classes of automata known to possess this property. We show that this is also true for commutative automata and we find commutative automata that are not state-partition automata.

**Keywords:** State complexity · Finite automata · Projection · Permutation automata · State-partition automata · Commutative automata

## 1 Introduction

The state complexity of a regularity-preserving operation is the minimal number of states needed in a recognizing automaton for the result of this operation, dependent on the size of the input automaton. The study of the state complexity was initiated in [18] and systematically started in [33]. As the number of states of a recognizing automaton could be interpreted as the memory required to describe the recognized language and is directly related to the runtime of algorithms employing regular languages, obtaining state complexity bounds is a natural question with applications in verification, natural language processing or software engineering [7,15,21,25,30].

Here, in terms of state complexity, we are concerned with deterministic automata only. There were also investigations using nondeterministic automata [8]. However, deterministic automata have better algorithmic properties: (1) equality could be done in almost linear time [10], (2) the minimal automaton is unique up to isomorphism [11] and (3) there is an $O(n \log n)$-time minimization algorithm [9]. Contrary, for nondeterministic automata, equality testing is


N. Moreira and R. Reis (Eds.): DLT 2021, LNCS 12811, pp. 192–203, 2021.
https://doi.org/10.1007/978-3-030-81508-0_16

PSPACE-complete [27], minimal automata are not unique and minimization is a PSPACE-complete problem [8].

The state complexity of the projection operation was investigated in [13,31]. In [31], the tight upper bound $3 \cdot 2^{n-2} - 1$ was shown, and in [13] 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.

The projection operator has applications in engineering, verification, fault diagnosis and supervisory control [5,16,17,32], as it corresponds to the observable behavior, a simplified or a restricted view of a modeled system. However, as, in general, the resulting automaton could be exponentially large, in practical applications only those projections that avoid this blow-up are interesting. Motivated by this, in [14] state-partition automata for a projection were introduced, a class of automata for which the projection is recognizable with $n$ states, if the input automaton has $n$ states.

Permutation automata were introduced in [29] and by McNaugthon [19] in connection with the star-height problem. The languages recognized by permutation automata are called (pure-)group languages [19,23,24]. However, one could argue that, if not viewed as language recognizing devices, but as mere state-transition systems, sometimes also just called semi-automata, permutation automata were around under the disguise of finite permutation groups, i.e., subgroups of the group of all permutation on a finite set, since the beginning of the 19th century, starting with the work of Galois, Lagrange, Jordan and others [3,22]. However, certainly, the viewpoint was different.

Languages recognized by permutation automata are not describable by first-order formulae using only the order relation [20] and commutative regular languages correspond to threshold and modulo counting of letters [24]. The languages recognized by certain permutation automata, for example whose transformation monoids are solvable or supersoluble groups, were described in [4,6,28]. Investigation of the state complexity of common operations on permutation automata was initiated on last years edition of this conference [12].

Here, we investigate the projection operator on permutation automata. We give a better tight bound for permutation automata, also parameterized by the number of unobservable transitions, that, however, also grows exponentially. We give sufficient conditions, related to normal subgroups, to yield a state-partition permutation automaton for a given projection. Then, we investigate projections for commuting letters, this in particular encompasses commutative languages and automata. We show that if we delete commuting letters by a projection operator, then we also just need $n$ states for an $n$-state input automaton for the projected language. In particular this applies to commutative automata. We find commutative automata that are not state-partition automata for a given projection. This is in particular interesting, as in [13], it was noted that up to then, only state-partition automata and automata describing finite language with a unary projected language were known to have the property that we only need $n$ states for the projected languages.

Lastly, we derive that the projection operator preserves every variety of commutative languages. This includes, for example, the commutative aperiodic, the commutative group languages or the commutative piecewise-testable languages.

## 2 General Notions

By $\Sigma$ we denote a finite set of symbols, also called an *alphabet*. By $\Sigma^*$ we denote the set of all *words* over $\Sigma$, i.e., finite sequences with the concatenation operation. The *empty word* is denoted by $\varepsilon$. A *language* $L$ is a subset $L \subseteq \Sigma^*$. Languages using only a single symbol are called *unary languages*.

If $X$ is a set, by $\mathcal{P}(X) = \{Y \mid Y \subseteq X\}$ we denote the *power set* of $X$.

If $x$ is a non-negative real number, by $\lceil x \rceil$ we denote the smallest natural number greater or equal to $x$ and by $\lfloor x \rfloor$ the largest natural number smaller or equal to $x$.

Let $\Gamma \subseteq \Sigma$. The homomorphism $\pi_\Gamma : \Sigma^* \to \Gamma^*$ given by $\pi_\Gamma(x) = x$ for $x \in \Gamma$ and $\pi_\Gamma(x) = \varepsilon$ for $x \in \Sigma \backslash \Gamma$ is called a *projection (for $\Gamma$)*. If $p, q \in Q$, $x \in \Sigma$, then a transition $\delta(p, x) = q$ is said to be *unobservable* with respect to the projection $\pi_\Gamma$ if $x \in \Sigma \backslash \Gamma$, i.e., $\pi_\Gamma(x) = \varepsilon$. Here, only non-loop unobservable transitions are of interest, i.e., those such that $p \neq q$.

A *(partial) deterministic finite automaton (DFA)* is denoted by a quintuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, where $Q$ is a *finite set of states*, $\Sigma$ the *input alphabet*, $\delta : Q \times \Sigma \to Q$ is a *partial transition function*, $q_0$ the *start state* and $F \subseteq Q$ the set of *final states*. The DFA is said to be *complete* if $\delta$ is a total function. In the usual way, the transition function $\delta$ can be extended to a function $\hat{\delta} : Q \times \Sigma^* \to Q$ by setting, for $q \in Q$, $u \in \Sigma^*$ and $a \in \Sigma$, $\hat{\delta}(q, \varepsilon) = q$ and $\hat{\delta}(q, ua) = \delta(\hat{\delta}(q, u), a)$. In the following, we drop the distinction between $\delta$ and $\hat{\delta}$ and denote both functions simply by $\delta$.

For $S \subseteq Q$ and $u \in \Sigma^*$, we set $\delta(S, u) = \{\delta(s, u) \mid s \in S \text{ and } \delta(s, u) \text{ is defined}\}$.

The language *recognized* by $\mathcal{A}$ is $L(\mathcal{A}) = \{u \in \Sigma^* \mid \delta(q_0, u) \in F\}$. A language $L \subseteq \Sigma^*$ is called *regular*, if there exists an automaton $\mathcal{A}$ such that $L = L(\mathcal{A})$.

For $u \in \Sigma^*$, we write $\delta(p, u) = \delta(q, u)$ if both are defined and the results are equal or both are undefined.

We say that $q$ is *reachable* from $p$ (in $\mathcal{A}$) if there exists a word $u \in \Sigma^*$ such that $\delta(p, u) = q$. The DFA $\mathcal{A}$ is called *initially connected*, if every state is reachable from the start state.

The DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ is called *commutative*, if, for each $a, b \in \Sigma$ and $q \in Q$, we have $\delta(q, ab) = \delta(q, ba)$.

Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a complete DFA. For a word $u \in \Sigma^*$, the *transition function (in $\mathcal{A}$) associated to $u$* is the function $\delta_u : Q \to Q$ given by $\delta_u(q) = \delta(q, u)$ for $q \in Q$. The *transformation monoid* is $\mathcal{T}_\mathcal{A} = \{\delta_u \mid u \in \Sigma^*\}$. Note that we defined the transformation monoid only for complete DFAs, as this is the only context where we need this notion here.

To denote transitions in permutation DFAs, we use a *cycle notation* also used in [2,12]. More formally, $(q_1, \ldots, q_k)$ denotes the cyclic permutation mapping $q_i$ to $q_{i+1}$ for $i \in \{1, \ldots, k-1\}$ and $q_k$ to $q_1$. For example, $a = (1, 2)(3, 4, 5)$ means the letter $a$ swaps the states 1 and 2, cyclically permutes the states 3, 4 and 5 in the indicated order and fixes all other states.

A *variety (of formal languages)* $\mathcal{V}$ [6,23,24] associates, to each alphabet $\Sigma$, a class of recognizable languages $\mathcal{V}(\Sigma^*)$ over $\Sigma$ such that (1) $\mathcal{V}(\Sigma^*)$ is a boolean algebra, (2) if $\varphi : \Sigma^* \to \Gamma^*$ is a homomorphism, then $L \in \mathcal{V}(\Gamma^*)$ implies $\varphi^{-1}(L) \in$

$\mathcal{V}(\Sigma^*)$ and (3) if $L \in \mathcal{V}(\Sigma^*)$ and $x \in \Sigma$, then $\{u \in \Sigma^* \mid xu \in L\}$ and $\{u \in \Sigma^* \mid ux \in L\}$ are in $\mathcal{V}(\Sigma^*)$.

## 3 Orbit Sets, Projected Languages and Permutation Automata

First, we introduce the orbit set of a set of states for a subalphabet. An orbit set collects those states that are reachable from a given set of states by only using words from a given subalphabet. This is also called *unobservable reach* in [5].

**Definition 1.** *Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Suppose $\Sigma' \subseteq \Sigma$ and $S \subseteq Q$. The* $\Sigma'$-orbit of $S$ *is the set*

$$\mathrm{Orb}_{\Sigma'}(S) = \{\delta(q, u) \mid \delta(q, u) \text{ is defined, } q \in S \text{ and } u \in \Sigma'^*\}.$$

*Also, for $q \in Q$, we set* $\mathrm{Orb}_{\Sigma'}(q) = \mathrm{Orb}_{\Sigma'}(\{q\})$.

Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a DFA and $\Gamma \subseteq \Sigma$. Set $\Delta = \Sigma \backslash \Gamma$. Next, we define the *projection automaton* of $\mathcal{A}$ for $\Gamma$ as $\mathcal{R}_{\mathcal{A}}^{\Gamma} = (\mathcal{P}(Q), \Gamma, \mu, \mathrm{Orb}_{\Delta}(q_0), E)$ with, for $S \subseteq Q$ and $x \in \Gamma$, the transition function

$$\mu(S, x) = \mathrm{Orb}_{\Delta}(\delta(S, x)) \tag{1}$$

and $E = \{T \subseteq Q \mid T \cap F \neq \varnothing\}$. In general, $\mathcal{R}_{\mathcal{A}}^{\Gamma}$ is not initially connected. However, non-reachable states could be omitted. Actually, by the definition of the start state and transition function, we can restrict the state set to subsets of the form $\mathrm{Orb}_{\Delta}(S)$ for $\varnothing \neq S \subseteq Q$.

**Theorem 2.** *Let $\mathcal{A}$ be a DFA and $\Gamma \subseteq \Sigma$. Then, $\pi_{\Gamma}(L(\mathcal{A})) = L(\mathcal{R}_{\mathcal{A}}^{\Gamma})$.*

We do not introduce $\varepsilon$-NFAs formally here, but only refer to the literature [11]. However, we note in passing that, usually, an automaton for a projected language of a regular language is constructed by replacing the letters to be deleted by $\varepsilon$-transitions and then determinizing the resulting $\varepsilon$-NFA [11,13]. Our construction is a more direct formulation of these steps, where the orbit sets are used in place of the $\varepsilon$-closure computations.

In [13,14], an automaton was called a *state-partition automaton* with respect to a projection $\pi_{\Gamma}$ (or, for short, a state-partition automaton for $\Gamma$), if the states of the resulting automaton from the above procedure, after discarding non-reachable subsets, form a partition of the original state set. Hence, in our terminology, an automaton $\mathcal{A}$ is a state partition automaton if the reachable states of $\mathcal{R}_{\mathcal{A}}^{\Gamma}$ form a partition of the states of $\mathcal{A}$.

A *permutation automaton* (or *permutation DFA*) is a DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ such that every letter permutes the state set, i.e., the function $\delta_x : Q \to Q$ given by $\delta_x(q) = \delta(q, x)$ is a permutation, or bijection, of $Q$ for every $x \in \Sigma$. The languages recognized by permutation automata are called *group languages*. Note that permutation DFAs are complete DFAs.

The *identity transformation (on $Q$)* is the permutation $\mathrm{id} : Q \to Q$ given by $\mathrm{id}(q) = q$ for each $q \in Q$.

Next, we take a closer look at the orbit sets for permutation automata. But first, a general property of permutation automata.

**Lemma 3.** *Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a permutation automaton and $\Sigma' \subseteq \Sigma^*$. Then, for every $u \in \Sigma'^*$ there exists $u' \in \Sigma'^*$ such that $\delta(q, uu') = q$ for each $q \in Q$, i.e., the word $uu'$ represents the identity transformation on $Q$.*

With the previous lemma, we can show that the orbit sets for permutation automata partition the state set. This property is crucial to derive our state complexity bound for projection, as it vastly reduces the possible subsets that are reachable in $\mathcal{R}_{\mathcal{A}}^{\Gamma}$, namely only unions of orbit sets.

**Lemma 4.** *Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a permutation automaton. Suppose $\Sigma' \subseteq \Sigma$. Then, the sets $\mathrm{Orb}_{\Sigma'}(q)$, $q \in Q$, partition $Q$ and for every $S \subseteq Q$, $\mathrm{Orb}_{\Sigma'}(S) = \bigcup_{q \in S} \mathrm{Orb}_{\Sigma'}(q)$.*

## 4 Projection on Permutation Automata

Here, we state a tight upper bound for the number of states of the projection of a language recognized by a permutation automaton.

Our bound is parameterized by the number of states of the input automaton and by the number of non-loop unobservable transitions. More specifically, we consider the number of states that are incident with non-loop unobservable transitions. Hence, we disregard unobservable multi-transitions and do not take the direction into account, i.e., counting multiple transitions resulting from multiple letters between the same states only once and do not take their direction into account. This is the same usage of this parameter as in [14] for the general case.

**Theorem 5.** *Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a permutation DFA and $\Gamma \subseteq \Sigma$. Set $m = |\{\, p, q \in Q \mid p \neq q \text{ and } q \in \delta(p, \Sigma \backslash \Gamma)\}|$. Then, if $m > 0$, the projected language $\pi_\Gamma(L)$ is recognizable by a DFA with at most $2^{|Q| - \lceil \frac{m}{2} \rceil} - 1$ states and if $m = 0$, the projected language is recognizable by a DFA with at most $|Q|$ states.*

*Proof.* Set $\Delta = \Sigma \backslash \Gamma$, $S = \{p, q \in Q \mid p \neq q \text{ and } q \in \delta(p, \Delta)\}$ and $T = \{p \in Q \mid \forall x \in \Delta : \delta(p, x) = p\}$. Then, as $\mathcal{A}$ is a permutation automaton, $Q$ is the disjoint union of $S$ and $T$ and

$$q \in T \Leftrightarrow \mathrm{Orb}_\Delta(q) = \{q\} \text{ and } q \in S \Leftrightarrow |\,\mathrm{Orb}_\Delta(q)| \geqslant 2. \tag{2}$$

Set $\mathcal{B} = \mathcal{R}_{\mathcal{A}}^{\Gamma}$. If $m = 0$, then $Q = T$ and every $a \in \Delta$ induces a self-loop at every state. In this case, it is clear that we can simply leave out all the transitions labeled with letters from $\Delta$ and the resulting permutation automaton recognizes $\pi_\Gamma(L(\mathcal{A}))$. More formally, in the definition of $\mathcal{B}$, in this case, the starting state is $\{q_0\}$ and as $\mathcal{A}$ is deterministic we have $|\delta(R, x)| \leqslant |R|$ for every $R \subseteq Q$. So, as the empty set is never reachable for permutation DFAs, only the singleton sets $\{q\}$ are reachable in $\mathcal{B}$.

Now, suppose $m = |S| > 0$, which implies $m \geqslant 2$.

Claim: Let $m > 0$. Then, in $\mathcal{B}$ at most $2^{|Q|-\lceil \frac{m}{2} \rceil} - 1$ states are reachable from the start state.

*Proof of the Claim:* With the assumption $m > 0$, there exists $q \in Q$ such that $|\operatorname{Orb}_\Delta(q)| > 1$. By Equation (2) and Lemma 4, we have at most $|T| + \left\lfloor \frac{|S|}{2} \right\rfloor$ many $\Delta$-orbits, where the maximum number of $\Delta$-orbits is reached if every $\Delta$-orbit of a state from $S$ has size exactly two if $|S|$ is even or every such orbit has size two, except one that has size three, if $|S|$ is odd. By Lemma 4, the sets $\operatorname{Orb}_\Delta(\{q\})$, $q \in Q$, partition the state set and, for every $R \subseteq Q$, we have $\bigcup_{q \in R} \operatorname{Orb}_a(q) = \operatorname{Orb}_a(R)$. So, by Equation (1), every set reachable is a union of $\Delta$-orbits, i.e., every such set corresponds uniquely to a subset of $\Delta$-orbits for a single state. Finally, note that, as $\mathcal{A}$ is a permutation automaton, and hence complete, we have $\delta(R, x) \neq \varnothing$ for every non-empty $R \subseteq Q$, which also gives that in $\mathcal{B}$ the empty set is not reachable. So, in total, we find that at most

$$2^{|T|+\left\lfloor \frac{|S|}{2} \right\rfloor} - 1 = 2^{|Q|-m+\lfloor \frac{m}{2} \rfloor} - 1 = 2^{|Q|-\lceil \frac{m}{2} \rceil} - 1$$

subsets of states are reachable. [*End, Proof of the Claim*]

So, we have shown the upper bound. □

Next, we show that the bound stated in the previous theorem is actually tight for permutation automata.

**Theorem 6.** *Let $n, m > 0$ be such that $0 < 2m + 1 < n$, $\Sigma = \{a, b, c, d, e, f, g\}$ and $\Gamma = \{b, c, d, e, f, g\}$. Then, there exists a permutation automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ with $2m$ states incident to non-loop unobservable transitions for $\pi_\Gamma$, i.e.,*

$$2m = |\{\ p, q \in Q \mid p \neq q \text{ and } q \in \delta(p, \Sigma \backslash \Gamma)\}|,$$

*such that every DFA for $\pi_\Gamma(L(\mathcal{A}))$ needs at least $2^{n-m} - 1$ states.*

*Proof (sketch).* See Fig. 1 for a permutation automaton giving the lower bound. The automaton has $n$ states, and the letters act the following way:

$$\begin{aligned} a &= (1,2)(3,4)\cdots(2m-1,2m), \\ b &= (2m+1,2m+2), & c &= (2m+1,2m+2,\ldots,n), \\ d &= (1,3)(2,4), & e &= (1,3,\ldots,2m-1)(2,4,\ldots,2m), \\ f &= (1,n), & g &= (1,n)(2,n-1). \end{aligned}$$

With $\Delta = \{a\}$, the $\Delta$-orbits are $\{1,2\}, \{3,4\}, \ldots, \{2m-1, 2m\}, \{2m+1\}, \ldots, \{n\}$. The letters $b, c$ are chosen such that every permutation of the states $\{2m+1, \ldots, n\}$ could be written as a word over them, and the letters $d$ and $e$ such that every permutation on the $\Delta$-orbits $\{1,2\}, \ldots, \{2m-1, 2m\}$ could be written as a word over them. The letters $f$ and $g$ help to map between these $\Delta$-orbits in such a way that every non-empty union of $\Delta$-orbits is reachable, and all these $\Delta$-orbits give distinguishable states. By mapping onto the two element $\Delta$-orbits and back, we can enlarge the sets that are reachable. □

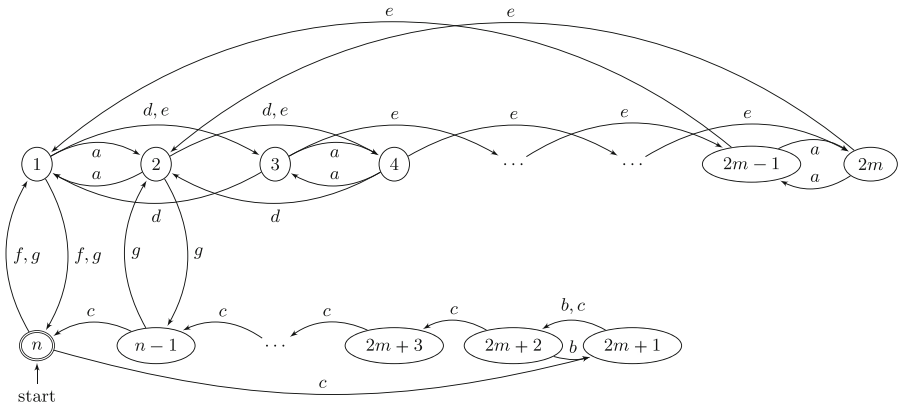


**Fig. 1.** All transitions not shown, for example for the letter $b$ at the state $n$, correspond to self-loops, as permutation automata are complete. Then, the permutation automaton shown reaches the upper bound stated in Theorem 5 for the projection $\pi_\Gamma : \{a, b, c, d, e, f, g\}^* \to \Gamma^*$ with $\Gamma = \{b, c, d, e, f, g\}$.

*Remark 1.* Note that if $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ is initially connected and has the property that from every state $q \in Q$ a final state is reachable, then also $\mathcal{R}_\mathcal{A}^\Gamma$ has this property. As permutation DFAs are complete by definition, this implies that the tight bound stated in Theorem 5 remains the same if we would additionally demand the resulting DFA for the projection to be complete.

We used an alphabet of size seven to match the bound. So, the question arises if we can reach the bound using a smaller alphabet. I do not know the answer yet, but by using a result from [13, Theorem 6] that every projection onto a unary language needs less than $\exp((1 + o(1))\sqrt{n \ln(n)})$ states, we can deduce that we need at least a ternary alphabet to reach the bound stated in Theorem 5. For the bound stated in Theorem 5 is lowest possible, apart from the trivial case $m = 0$, if $m = n$. Then, the bound is $2^{\lfloor n/2 \rfloor} - 1$. However, asymptotically, this grows way faster than $\exp((1 + o(1))\sqrt{n \ln(n)})$, in fact, the ratio of both expressions could be arbitrarily large.

**Proposition 7.** *Each permutation automaton $\mathcal{A}$ such that $\pi_\Gamma(L(\mathcal{A}))$ for a nonempty and proper subalphabet $\Gamma \subseteq \Sigma$ attains the bound stated in Theorem 5 with $m > 0$ must be over an alphabet with at least three letters and $|\Gamma| \geqslant 2$.*

## 5 State-Partition Automata and Normal Subgroups

First, we derive a sufficient condition for a permutation automaton to be a state-partition automaton for a projection. Then, we introduce normal subgroups and show that if the letters generate a normal subgroup, this condition is fulfilled.

**Proposition 8.** *Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a permutation automaton and $\Gamma \subseteq \Sigma$. Set $\Delta = \Sigma \backslash \Gamma$. Then, $\mathcal{A}$ is a state-partition automaton for $\pi_\Gamma$ if the $\Delta$-orbits of the form $\mathrm{Orb}_\Delta(q)$ are permuted, i.e., for each $x \in \Sigma$ and $q \in Q$, we have $\delta(\mathrm{Orb}_\Delta(q), x) = \mathrm{Orb}_\Delta(\delta(q, x))$.*

With Lemma 4, if the orbits for some $\Delta \subseteq \Sigma$ are permuted, then, for each $q \in Q$ and $x \in \Sigma$, $\delta(\mathrm{Orb}_\Delta(q), x) = \mathrm{Orb}_\Delta(q)$ or $\delta(\mathrm{Orb}_\Delta(q), x) \cap \mathrm{Orb}_\Delta(q) = \varnothing$.

*Remark 2.* The following example shows that $\mathcal{A}$ being a state-partition automaton for $\Gamma$ does not imply that the sets $\mathrm{Orb}_\Delta(q)$ are permuted. Let $\mathcal{A} = (\{1,2,3,4,5,6,7,8\}, \{a,b\}, \delta, 1, \{1\})$ with the transitions $a = (1,2,3,4)(5,6)(7,8)$ and $b = (1,5)(2,6)(3,7)(4,8)$. Then, for $\Gamma = \{b\}$ the automaton is a state-partition automaton, as the reachable states in $\mathcal{R}_\mathcal{A}^\Gamma$ are $\{1,2,3,4\}$ and $\{5,6,7,8\}$, but the $a$-orbits are $\{1,2,3,4\}, \{5,6\}$ and $\{7,8\}$.

Recall that $\mathcal{T}_\mathcal{A}$ denotes the transformation semigroup of $\mathcal{A}$. A subgroup of $\mathcal{T}_\mathcal{A}$, if $\mathcal{A}$ is a finite permutation automaton, is a subset containing the identity transformation and closed under function composition. As we are only concerned with finite automata, this also implies closure under inverse functions.

Next, we show that when the symbols deleted by a projection generate a normal subgroup, then the automaton is a state-partition automaton for this projection.

Normal subgroups are ubiquitous [3,26] in abstract group theory as well as in permutation group theory. We give a definition for subgroups of $\mathcal{T}_\mathcal{A}$, when $\mathcal{A}$ is a permutation automaton, using our notation. We refer to more specialized literature for other definitions and more motivation [3,26].

**Definition 9.** *Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a permutation automaton. Then, a subgroup $N$ of $\mathcal{T}_\mathcal{A}$ is called* normal, *if, for each $\delta_u, \delta_v \in \mathcal{T}_\mathcal{A}$ $(u, v \in \Sigma^*)$,*

$$(\exists \delta_w \in N : \delta_u = \delta_{wv}) \Leftrightarrow (\exists \delta_{w'} \in N : \delta_u = \delta_{vw'}).$$

If a set of letters generates a normal subgroup, then the orbits of these letters are permuted by the other letters. As they are invariant under the letters themselves that generate these orbits, every word over $\Sigma$ permutes these orbits. This is the statement of the next lemma.

**Lemma 10.** *Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a permutation automaton and $\Sigma' \subseteq \Sigma$ be such that $N = \{\delta_u : Q \to Q \mid u \in \Sigma'^*\}$ is a normal subgroup of $\mathcal{T}_\mathcal{A}$. Then, for each $x \in \Sigma$ and $q \in Q$, we have $\delta(\mathrm{Orb}_{\Sigma'}(q), x) = \mathrm{Orb}_{\Sigma'}(\delta(q, x))$.*

So, combining Proposition 8 and Lemma 10.

**Theorem 11.** *Let $\Gamma \subseteq \Sigma$, $\Delta = \Sigma \backslash \Gamma$ and $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a permutation automaton. Set $N = \{\delta_u : Q \to Q \mid u \in \Delta^*\}$, the subgroup in $\mathcal{T}_\mathcal{A}$ generated by $\Delta$. If $N$ is normal in $\mathcal{T}_\mathcal{A}$, then $\mathcal{A}$ is a state-partition automaton for $\pi_\Gamma$. Hence, in this case, $\pi_\Gamma(L(\mathcal{A}))$ is recognizable by an automaton with at most $|Q|$ states.*

## 6 Commuting Letters

Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a DFA. We say that two letters $a, b \in \Sigma$ *commute (in* $\mathcal{A}$*)*, if $\delta(q, ab) = \delta(q, ba)$ for each $q \in Q$. Hence, an automaton $\mathcal{A}$ is commutative precisely if all letters commute pairwise.

Here, we investigate commuting letters with respect to the projection operation. Our first lemma states that if we can partition the alphabet of an $n$-state DFA into two subalphabets of letters such that each letter in the first set commutes with each letter in the second set, then for a projection onto one subalphabet, the projected language is recognizable by an $n$-state automaton. By this result, the projected language of every $n$-state commutative automaton is recognizable by an $n$-state automaton. We construct commutative automata that are not state-partition automata. Hence, we have new examples of automata whose projected languages are recognizable by automata with no more states than the original automaton, but which are not state-partition automata. Lastly, by investigating the proofs, we can show, with not much more effort, that varieties of commutative languages are closed under projections.

**Lemma 12.** *Suppose* $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ *is an arbitrary DFA. Let* $\Gamma \subseteq \Sigma$ *be such that, for each* $a \in \Sigma \backslash \Gamma$, $b \in \Gamma$ *and* $q \in Q$, *we have* $\delta(q, ab) = \delta(q, ba)$. *Then,* $\pi_\Gamma(L)$ *is recognizable by a DFA with at most* $|Q|$ *states.*

*Proof.* Intuitively, we take the input automaton and leave out all unobservable transitions and make a state accepting if, in the input automaton, we can go from this state to a final state by a word formed out of the deleted letters.

Let $\mathcal{B} = (Q, \Gamma, \delta_{|\Gamma}, q_0, E)$ be the DFA with $\delta_{|\Gamma}(q, x) = \delta(q, x)$, the same start state $q_0$ and $E = \{p \in Q \mid \exists q \in F \ \exists u \in (\Sigma \backslash \Gamma)^* : \delta(p, u) = q\}$. Then, $L(\mathcal{B}) = \pi_\Gamma(L(\mathcal{A}))$.

If $\delta_{|\Gamma}(q_0, u) \in E$, then there exists $v \in (\Sigma \backslash \Gamma)^*$ such that $\delta(q_0, uv) \in F$. So, $uv \in L(\mathcal{A})$ and $u = \pi_\Gamma(uv)$.

Conversely, suppose $u = \pi_\Gamma(v)$ for some $v \in L(\mathcal{A})$. By assumption, as we can successively push all letters in $\Sigma \backslash \Gamma$ to the end, we have $\delta(q_0, v) = \delta(q_0, \pi_\Gamma(v)\pi_{\Sigma \backslash \Gamma}(v))$. So, $\delta(q_0, \pi_\Gamma(v)\pi_{\Sigma \backslash \Gamma}(v)) \in F$, which yields $\delta_{|\Gamma}(q_0, \pi_\Gamma(v)) \in E$, hence $u \in L(\mathcal{B})$. □

So, with Lemma 12, we get the next result.

**Theorem 13.** *Let* $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ *be a DFA such that* $L(\mathcal{A})$ *is commutative. If* $\Gamma \subseteq \Sigma$, *then* $\pi_\Gamma(L(\mathcal{A}))$ *is recognizable by a DFA with at most* $|Q|$ *states.*

The definition of normality could be seen as a generalization of commutativity. Hence, with Theorem 11, we can deduce the next statement.

**Proposition 14.** *Let* $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ *be a commutative permutation automaton and* $\Gamma \subseteq \Sigma$. *Then,* $\mathcal{A}$ *is a state-partition automaton for* $\pi_\Gamma$.

However, there exist commutative automata that are not state-partition automata, as shown by Example 1.

*Example 1.* Let $\mathcal{A} = (\{q_\varepsilon, q_a, q_b\}, \{a, b\}, \delta, q_\varepsilon, \{q_\varepsilon, q_b\})$ with

$$\delta(q_x, y) = \begin{cases} q_a \text{ if } x = \varepsilon, y = a; \\ q_b \text{ if } x = \varepsilon, y = b; \\ q_b \text{ if } x = a, y = b; \\ q_x \text{ otherwise.} \end{cases}$$

Then, $L(\mathcal{A}) = \{u \in \{a, b\}^* \mid |u|_a = 0 \text{ or } |u|_b > 0\}$. However, we see that $\mathrm{Orb}_{\{b\}}(q_\varepsilon) = \{q_\varepsilon, q_b\}$, $\mathrm{Orb}_{\{b\}}(q_b) = \{q_b\}$ and $\mathrm{Orb}_{\{b\}}(q_a) = \{q_a, q_b\}$. Hence, $\mathcal{A}$ is not a state-partition automaton for the projection $\pi_{\{a\}} : \{a, b\}^* \to \{a\}^*$. Also, it is not a state-partition automaton for the projection onto $\{b\}^*$.

The proofs of Lemma 12 and Theorem 13 also show that the projected language of a commutative permutation automaton is recognizable by a permutation automaton, i.e., a group language. On the other hand, in the general case, Example 2 below gives a permutation automaton whose projected language is not a group language. Also, most properties defined in terms of automata are preserved by projection in the commutative case. For example the property of being aperiodic [6,23,24]. We give a more general statement next, showing that many classes from the literature [6,23,24] are closed under projection when restricted to commutative languages.

**Theorem 15.** *Let $\Sigma$ be an alphabet and $\Gamma \subseteq \Sigma$. Suppose $\mathcal{V}$ is a variety of commutative languages. If $L \in \mathcal{V}(\Sigma^*)$, then $\pi_\Gamma(L) \in \mathcal{V}(\Gamma^*)$. In particular, the variety of commutative languages is closed under projection.*

Hence, for example commutative locally-testable, piecewise-testable, star-free or group languages are preserved under every projection operator, as these classes form varieties [23].

*Remark 3.* In [13] it was stated that languages satisfying the observer property, i.e., that are given by a state-partition automaton for a given projection operator, and the finite languages projected onto unary finite languages were the only known languages for which we can recognize the projected language with at most the number of states as the original language. Note that Theorem 13 provides genuinely new instances for which this holds true, see Example 1.

*Example 2.* Also, for projections, consider the group language given by the permutation automaton $\mathcal{A} = (\{a, b\}, \{0, 1, 2\}, \delta, 0, \{2\})$ with $a = (0, 1)$ and $b = (0, 1, 2)$. Then, $\pi_{\{b\}}(L(\mathcal{A})) = bb^*$, which is not a group language. For example, $b$ is the projection of $ab \in L(\mathcal{A})$, or $bbb$ the projection of $abbab \in L(\mathcal{A})$.

## 7 Conclusion

We have continued the investigation of the state complexity of operations on permutation automata, initiated in [12], and the investigation of the projection operation [13,31]. We improved the general bound to the tight bound $2^{n-\lceil \frac{m}{2} \rceil} - 1$

in this case. Note that the general bound $2^{n-1} + 2^{n-m} - 1$ for the projection is only achieved for automata with precisely $m - 1$ non-loop unobservable transitions [13]. However, if we have more such unobservable transitions, then it was also shown in [13] that we have the better tight bound $2^{n-2}+2^{n-3}+2^{n-m}-1$. For our lower bound stated in Theorem 6 in the case of permutation automata, we have precisely the same number of non-loop unobservable transitions as states incident with them. Lastly, note that the condition in Proposition 8 could be easily checked. Likewise, checking if a subset of letters $\Delta$ generate a normal subgroup could also be checked efficiently using results from [1].

**Acknowledgement.** I sincerely thank the anonymous reviewers for careful reading and detailed feedback that helped me in finding better formulations or fixing typos. Also, Sect. 5 was restructured after this feedback and the cycle notation was pointed out to me by one reviewer.

## References

1. Babai, L., Luks, E.M., Seress, Á.: Permutation groups in NC. In: Aho, A.V. (ed.) STOC 1987, pp. 409–420. ACM (1987)
2. Brzozowski, J.A., Sinnamon, C.: Complexity of proper prefix-convex regular languages. Theor. Comput. Sci. **787**, 2–13 (2019)
3. Cameron, P.J.: Permutation Groups. London Mathematical Society Student Texts. Cambridge University Press, Cambridge (1999)
4. Carton, O., Pin, J., Soler-Escrivà, X.: Languages recognized by finite supersoluble groups. J. Autom. Lang. Comb. **14**(2), 149–161 (2009)
5. Cassandras, C.G., Lafortune, S.: Introduction to Discrete Event Systems, 2nd edn. Springer, Boston (2008). https://doi.org/10.1007/978-0-387-68612-7
6. Eilenberg, S.: Automata, Languages, and Machines, vol. B. Academic Press Inc., Orlando (1976)
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. Holzer, M., Kutrib, M.: Nondeterministic finite automata - recent results on the descriptional and computational complexity. Int. J. Found. Comput. Sci. **20**(4), 563–580 (2009)
9. Hopcroft, J.: An $n$ log $n$ algorithm for minimizing states in a finite automaton. In: Theory of Machines and Computations (Proceedings International Symposium, Technion, Haifa, 1971), pp. 189–196. Academic Press, New York (1971)
10. Hopcroft, J., Karp, R.: A linear algorithm for testing equivalence of finite automata, technical Report 71–114, University of California (1971)
11. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley Publishing Company (1979)
12. Hospodár, M., Mlynárčik, P.: Operations on permutation automata. In: Jonoska, N., Savchuk, D. (eds.) DLT 2020. LNCS, vol. 12086, pp. 122–136. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-48516-0_10
13. Jirásková, G., Masopust, T.: On a structural property in the state complexity of projected regular languages. Theor. Comput. Sci. **449**, 93–105 (2012)
14. Jirásková, G., Masopust, T.: On properties and state complexity of deterministic state-partition automata. In: Baeten, J.C.M., Ball, T., de Boer, F.S. (eds.) TCS 2012. LNCS, vol. 7604, pp. 164–178. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33475-7_12

15. Kohavi, Z., Jha, N.K.: Switching and Finite Automata Theory, 3 edn. Cambridge University Press (2009)
16. Komenda, J., Masopust, T., van Schuppen, J.H.: Supervisory control synthesis of discrete-event systems using a coordination scheme. Autom. **48**(2), 247–254 (2012)
17. Komenda, J., Masopust, T., van Schuppen, J.H.: Coordination control of distributed discrete-event systems. In: Seatzu, C., Silva, M., van Schuppen, J.H. (eds.) Control of Discrete-Event Systems, LNCIS, vol. 433, pp. 147–167. Springer, London (2013). https://doi.org/10.1007/978-1-4471-4276-8_8
18. Maslov, A.N.: Estimates of the number of states of finite automata. Dokl. Akad. Nauk SSSR **194**(6), 1266–1268 (1970)
19. McNaughton, R.: The loop complexity of pure-group events. Inf. Control **11**(1/2), 167–176 (1967)
20. McNaughton, R., Papert, S.A.: Counter-Free Automata (M.I.T. Research Monograph No. 65). The MIT Press (1971)
21. Mihov, S., Schulz, K.U.: Finite-state techniques: automata, transducers and Bimachines. In: Cambridge Tracts in Theoretical Computer Science. Cambridge University Press (2019)
22. Neumann, P.M.: The Mathematical Writings of Évariste Galois. European Mathematical Society, Heritage of European Mathematics (2011)
23. Pin, J.: Varieties Of Formal Languages. Plenum Publishing Co. (1986)
24. Pin, J.-E.: Syntactic Semigroups. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, pp. 679–746. Springer, Heidelberg (1997). https://doi.org/10.1007/978-3-642-59136-5_10
25. Roche, E., Schabes, Y. (eds.): Finite-State Language Processing. MIT Press (1997)
26. Rotman, J.: An Introduction to the Theory of Groups. Springer, New York (1995). https://doi.org/10.1007/978-1-4612-4176-8
27. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time (preliminary report). In: STOC, pp. 1–9. ACM (1973)
28. Thérien, D.: Languages of nilpotent and solvable groups (extended abstract). In: Maurer, H.A. (ed.) ICALP 1979. LNCS, vol. 71, pp. 616–632. Springer, Heidelberg (1979). https://doi.org/10.1007/3-540-09510-1_49
29. Thierrin, G.: Permutation automata. Math. Syst. Theory **2**(1), 83–90 (1968)
30. Wang, J. (ed.): Handbook of Finite State Based Models and Applications. Chapman and Hall/CRC (2012)
31. Wong, K.: On the complexity of projections of discrete-event systems. In: Proceedings of WODES 1998, Cagliari, Italy, pp. 201–206 (1998)
32. Wonham, W.M., Cai, K.: Supervisory Control of Discrete-Event Systems. CCE, Springer, Cham (2019). https://doi.org/10.1007/978-3-319-77452-7
33. Yu, S., Zhuang, Q., Salomaa, K.: The state complexities of some basic operations on regular languages. Theoret. Comput. Sci. **125**(2), 315–328 (1994)