New tools for state complexity

Pascal Caron[1] Edwin Hamel-De Le Court[1] Jean-Gabriel Luque[1]
Bruno Patrou[1]

[1] *LITIS, Université de Rouen-Normandie, France*

A monster is an automaton in which every function from states to states is represented by at least one letter. A modifier is a set of functions allowing one to transform a set of automata into one automaton. We revisit some language transformation algorithms in terms of modifier and monster. These new theoretical concepts allow one to find easily some state complexities. We illustrate this by retrieving the state complexity of the Star of Intersection and the one of the Square root operation.

1 Introduction

The studies around state complexities last for more than twenty years now. Mainly initiated by Yu et al ([22]) and very active ever since, this research area dates back in fact to the beginning of the 1970s. In particular, in [18] Maslov gives values (without proofs) for the state complexity of some operations: square root, cyclic shift and proportional removal. From these foundations, tens and tens of papers have been produced and different sub-domains have appeared depending on whether the used automata are deterministic or not, whether the languages are finite or infinite, belong to some classes (codes, star-free, ...) and so on. We focus here on the deterministic case for any language.

The state complexity of a regular language is the size of its minimal automaton and the state complexity of a regular operation is the maximal one of those languages obtained by applying this operation onto languages of fixed state complexities. So, to compute a state complexity, most of the time the approach is to calculate an upper bound from the characteristics of the considered operation and to provide a witness, that is a specific example reaching the bound which is then the desired state complexity.

This work has been done for numerous unary and binary operations. See, for example, [8], [13], [14], [15], [21] and [10] for a survey of the subject. More recently, the state complexity of combinations of operations has also been studied. In most of the cases the result is not simply the mathematical composition of the individual complexities and studies lead to interesting situations. Examples can be found in [20], [6], [11] or [16].

Beyond the search of state complexities and witnesses, some studies try to improve the given witnesses, especially the size of their alphabet ([5], [4]). Others try to unify the techniques and the approaches used to solve the different encountered problems. In [2], Brzozowski proposes to use some fundamental configurations to produce witnesses in many situations. In [3], the authors show how to compute the state complexities of 16 combinations by only studying three of them.

In this paper, we propose a general method to build witnesses, consisting in maximizing the transition function of automata. Among the resulting automata, called monsters, at least one of them is a witness. We just have to discuss the finality of the states to determine which ones are. We illustrate this technique by recomputing the state complexity of the operation obtained in combining star with intersection. The state complexity of the square root operation is also computed and improved (compared to the bound given by Maslov [18]) as another illustration.

The paper is organized as follows. Section 2 gives definitions and notations about automata and combinatorics. In Section 3, we define *modifiers* and give some properties of these algebraic structures. In Section 4, monsters automata are defined and their use in automata computation is shown. Section 5 is devoted to show how these new tools can be used to compute tight bounds for state complexity. Star of intersection and square root examples are described.

2   Preliminaries

Let Σ denote a finite alphabet. A word w over Σ is a finite sequence of symbols of Σ. The length of w, denoted by $|w|$, is the number of occurrences of symbols of Σ in w. For $a \in \Sigma$, we denote by $|w|_a$ the number of occurrences of a in w. The set of all finite words over Σ is denoted by Σ^*. The empty word is denoted by ε. A language is a subset of Σ^*. The cardinality of a finite set E is denoted by $\#E$, the set of subsets of E is denoted by 2^E and the set of mappings of E into itself is denoted by E^E.

A finite automaton (FA) is a 5-tuple $A = (\Sigma, Q, I, F, \delta)$ where Σ is the input alphabet, Q is a finite set of states, $I \subset Q$ is the set of initial states, $F \subset Q$ is the set of final states and δ is the transition function from $Q \times \Sigma$ to 2^Q extended in a natural way from $2^Q \times \Sigma^*$ to 2^Q.

A word $w \in \Sigma^*$ is recognized by an FA A if $\delta(I, w) \cap F \neq \emptyset$. The language recognized by an FA A is the set $L(A)$ of words recognized by A. Two automata are said to be equivalent if they recognize the same language. A state q is accessible in an FA if there exists a word $w \in \Sigma^*$ such that $q \in \delta(I, w)$.

An FA is complete and deterministic (CDFA) if $\#I = 1$ and for all $q \in Q$, for all $a \in \Sigma$, $\#\delta(q, a) = 1$. Let $D = (\Sigma, Q_D, i_D, F_D, \delta)$ be a CDFA. When there is no ambiguity, we identify $\#D$ to $\#Q_D$. For any word w, we denote by δ^w the function $q \to \delta(q, w)$. Two states q_1, q_2 of D are equivalent if for any word w of Σ^*, $\delta(q_1, w) \in F_D$ if and only if $\delta(q_2, w) \in F_D$. Such an equivalence is denoted by $q_1 \sim q_2$. A CDFA is minimal if there does not exist any equivalent CDFA with less states and it is well known that for any DFA, there exists a unique minimal equivalent one [12]. Such a minimal CDFA can be obtained from D by computing the accessible part of the automaton $D/\sim = (\Sigma, Q_D/\sim, [i_D], F_D/\sim, \delta_\sim)$ where for any $q \in Q_D$, $[q]$ is the $\sim$-class of the state q and satisfies the property $\delta_\sim([q], a) = [\delta(q, a)]$, for any $a \in \Sigma$. The number of its states is denoted by $\#_{Min}(D)$. In a minimal CDFA, any two distinct states are pairwise inequivalent.

For any integer n, let us denote $[\![n]\!]$ for $\{0, \ldots, n - 1\}$. When there is no ambiguity, for any character $\mathtt{X}$ and any integer k given by the context, we write $\underline{\mathtt{X}}$ for $(\mathtt{X}_1, \cdots, \mathtt{X}_k)$. The state complexity of a regular language L denoted by $sc(L)$ is the number of states of its minimal CDFA. Let $\mathcal{L}_n$ be the set of languages of state complexity n. The state complexity of a unary operation $\otimes$ is the function $sc_\otimes$ associating with an integer n, the maximum of the state complexities of $\otimes L$ for $L \in \mathcal{L}_n$. A language $L \in \mathcal{L}_n$ is a witness (for $\otimes$) if $sc(\otimes(L)) = sc_\otimes(n)$. This can be generalized, and the state complexity of a k-ary operation $\otimes$ is the k-ary function which associates with any

k-tuple of integers $\underline{n}$, the integer $\max\{\mathrm{sc}(\otimes\underline{L}) \mid \underline{L} \in \mathcal{L}_{n_1} \times \cdots \times \mathcal{L}_{n_k}\}$. Then, a witness is a tuple $\underline{L} \in (\mathcal{L}_{n_1} \times \cdots \times \mathcal{L}_{n_k})$ such that $\mathrm{sc}\underline{L} = \mathrm{sc}_{\otimes}\underline{n}$.

We also need some background from finite transformation semigroup theory [9]. Let n be an integer. A transformation t is an element of $[\![n]\!]^{[\![n]\!]}$. We denote by it the image of i under t. A transformation of $[\![n]\!]$ can be represented by $t = [i_0, i_1, \ldots i_{n-1}]$ which means that $i_k = kt$ for each $k \in [\![n]\!]$ and $i_k \in [\![n]\!]$. A *permutation* is a bijective transformation on $[\![n]\!]$. The *identity* permutation is denoted by $\mathbb{1}$. A *cycle* of length $\ell \le n$ is a permutation c, denoted by $(i_0, i_1, \ldots, i_{\ell-1})$, on a subset $I = \{i_0, \ldots, i_{\ell-1}\}$ of $[\![n]\!]$ where $i_k c = i_{k+1}$ for $0 \le k < \ell - 1$ and $i_{\ell-1} c = i_0$. A *transposition* $t = (i, j)$ is a permutation on $[\![n]\!]$ where $it = j$ and $jt = i$ and for all elements $k \in [\![n]\!] \setminus \{i, j\}$, $kt = k$. A *contraction* $t = \binom{i}{j}$ is a transformation where $it = j$ and for all elements $k \in [\![n]\!] \setminus \{i\}$, $kt = k$.

Let L and L' be two regular languages defined over an alphabet Σ. Let $\mathrm{Union}(L, L') = \{w \mid w \in L \lor w \in L'\}$, $\mathrm{Inter}(L, L') = \{w \mid w \in L \land w \in L'\}$, $\mathrm{Xor}(L, L') = \{w \mid (w \in L \land w \notin L') \lor (w \notin L \land w \in L')\}$, $\mathrm{Prefin}(L) = \{w = uv \mid u \in L, v \in \Sigma^*\}$, $\mathrm{Comp}(L) = \{w \mid w \notin L\}$, $\mathrm{Conc}(L, L') = \{w = uv \mid u \in L, v \in L'\}$, $\mathrm{Star}(L) = \{w = u_1 \cdots u_n \mid u_i \in L\}$, $\mathrm{SRoot}(L) = \{w \in \Sigma^* \mid ww \in L\}$.

3 Modifier and associated transformations

We first define a mechanism which unifies some automata transformations for regular operations on languages. This mechanism is called a *modifier*. A k-modifier is an algorithm taking k automata as input and outputting an automaton. A lot of regular operations on languages can be described using this mechanism (mirror, complement, Kleene star, . . .). These regular operations are called *describable*. Then, we give some properties for describable operations. We will first see that not all regular operations are describable and that there also exist modifiers which do not correspond to regular operations on languages.

3.1 Definitions

The *state configuration* of a DFA $A = (\Sigma, Q, i, F, \delta)$ is the triplet (Q, i, F). Our purpose is to consider operations on languages that can be encoded on DFA. To this aim, such an operation will be described as a k-ary operator $\mathfrak{m}$, acting on DFAs $A_1, \ldots A_k$ over the same alphabet Σ and producing a new DFA such that

- the alphabet of $\mathfrak{m}(A_1, ..., A_k)$ is Σ,

- the state configuration of $\mathfrak{m}(A_1, ..., A_k)$ depends only on the state configurations of the DFAs $A_1, \ldots, A_k$,

- for any letter $a \in \Sigma$, the transition function of a in $\mathfrak{m}(A_1, \ldots, A_k)$ depends only on the state configurations of the DFAs $A_1, \ldots, A_k$ and on the transition functions of a in each of the DFAs $A_1, ..., A_k$ (not on the letter itself nor on any other letter or transition function).

More formally,

Definition 1 *A k-modifier $\mathfrak{m}$ is a 4-tuple of mappings $(\mathfrak{Q}, \iota, \mathfrak{f}, \mathfrak{d})$ acting on k CDFA $\underline{A}$ with $A_j = (\Sigma, Q_j, i_j, F_j, \delta_j)$ to build a CDFA $\mathfrak{m}\underline{A} = (\Sigma, Q, i, F, \delta)$, where*
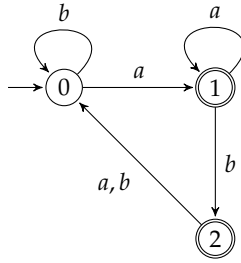
$$Q = \mathfrak{Q}\underline{Q}, \ i = \iota(\underline{Q}, \underline{i}, \underline{F}), F = \mathfrak{f}(\underline{Q}, \underline{i}, \underline{F}) \ and$$
$$\forall a \in \Sigma, \ \delta^a = \mathfrak{d}(\underline{i}, \underline{F}, \underline{\delta^a}).$$

Notice that we do not need to put explicitly the dependency of $\mathfrak{d}$ on $\underline{Q}$ because the information is already present in $\underline{\delta^a}$.
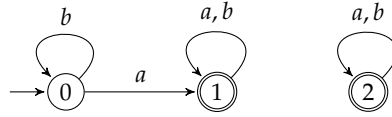
For 1-modifiers, as $\underline{Q} = (Q_1)$, we denote $\iota(Q_1, i_1, F_1)$ for $\iota(\underline{Q}, \underline{i}, \underline{F})$, $\mathfrak{f}(Q_1, i_1, F_1)$ for $\mathfrak{f}(\underline{Q}, \underline{i}, \underline{F})$, and $\mathfrak{d}(i_1, F_1, \delta_1^a)$ for $\mathfrak{d}(\underline{i}, \underline{F}, \underline{\delta^a})$.

Example 1 Consider the modifier $\mathfrak{Prefix}$ of Table 1. If $A_1 = (\Sigma, Q_1, i_1, F_1, \delta_1)$ is a complete deterministic automaton then $\mathfrak{Prefix}(A_1) = (\Sigma, Q_1, i_1, F_1, \delta)$ where for any state $q \in Q_1$ for any $a \in \Sigma$ we have $\delta^a(q) = \delta_1^a(q)$ if $q \notin F_1$ and $\delta^a(q) = q$ if $q \in F_1$.
For instance consider the automaton A_1 with the following graphical representation:

The automaton $\mathfrak{Prefix}(A_1)$ is given by

Definition 2 *We consider an operation $\otimes$ acting on k-tuples of languages defined on the same alphabet. The operation $\otimes$ is said to be* describable *(m-describable) if there exists a k-modifier m such that for any k-tuple of CDFA $\underline{A}$, we have $L(\mathfrak{m}\underline{A}) = \otimes(L(A_1), \ldots, L(A_k))$.*

Example 2 The operation Prefix defined by $\text{Prefix}(L) = L\Sigma^*$ for any $L \subset \Sigma^*$ is the $\mathfrak{Prefix}$-describable operation where $\mathfrak{Prefix}$ is the modifier defined in Table 1.

For the modifiers $\mathfrak{Union}$, $\mathfrak{Inter}$ and $\mathfrak{Xor}$, a state is an element of the cartesian product of the states of the input. For the $\mathfrak{Conc}$ modifier, a state is a pair composed of a state of the first input and a subset of states of the second input. For the $\mathfrak{Star}$ modifier, a state is a subset of states of the input. For the $\mathfrak{SRoot}$ modifier, each state is a function from the set of states to the set of states of the input.

	$\mathfrak{Q}\underline{Q}$	$\iota(Q,\underline{i},\underline{F})$	$\mathfrak{f}(Q,\underline{i},\underline{F})$	$\mathfrak{d}(\underline{i},\underline{F},\delta^a)$
$\mathfrak{Comp}$	Q_1	i_1	$Q_1 \setminus F_1$	δ_1^a
$\mathfrak{Prefin}$	Q_1	i_1	F_1	$q \to \begin{cases} \delta_1^a(q) & \text{if } q \notin F_1 \\ q & \text{if } q \in F_1 \end{cases}$
$\mathfrak{Union}$	$Q_1 \times Q_2$	(i_1,i_2)	$F_1 \times Q_2 \cup Q_1 \times F_2$	$\underline{\delta^a}$
$\mathfrak{Inter}$	$Q_1 \times Q_2$	(i_1,i_2)	$F_1 \times F_2$	$\underline{\delta^a}$
$\mathfrak{Xor}$	$Q_1 \times Q_2$	(i_1,i_2)	$F_1 \times (Q_2 \setminus F_2)$ $\cup(Q_1 \setminus F_1) \times F_2$	$\underline{\delta^a}$
$\mathfrak{Conc}$	$Q_1 \times 2^{Q_2}$	$(i_1,\emptyset)$	$\{(q_1,E) \mid E \cap F_2 \neq \emptyset\}$	$(q_1,E) \to \Xi_{i_1}^{F_1}(\delta_1^a(q_1),\delta_2^a(E))$
$\mathfrak{Star}$	2^{Q_1}	$\emptyset$	$\{E \mid E \cap F_1 \neq \emptyset\} \cup \{\emptyset\}$	$E \to \begin{cases} \overline{\{\delta_1^a(i_1)\}}^{F_1,i_1} & \text{if } E = \emptyset \\ \overline{\delta_1^a(E)}^{F_1,i_1} & \text{otherwise} \end{cases}$
$\mathfrak{SRoot}$	$Q_1^{Q_1}$	Id	$\left\{g \mid g^2(i_1) \in F_1\right\}$	$g \to (\delta_1^a \circ g)$

where $\overline{E}^{F,x} = E \cup \{x\}$ if $E \cap F \neq \emptyset$ and E otherwise, and $\Xi_y^F(x,E) = (x, E \cup \{y\})$ if $x \in F$ and (x,E) otherwise.

Tab. 1: Description of modifiers for some describable operations

Example 3 (Mirror modifier) *Let us define the 1-modifier* $\mathfrak{Mirror} = (\mathfrak{Q}, \iota, \mathfrak{f}, \mathfrak{d})$ *as :*

- $\mathfrak{Q}(Q_1) = 2^{Q_1}$,

- $\iota(Q_1, i_1, F_1) = F_1$,

- $\mathfrak{f}(Q_1, i_1, F_1) = \{E \subset Q_1 \mid i_1 \in E\}$.

- $\mathfrak{d}(i_1, F_1, \delta_1^a)$ *is defined as* $E \to E'$ *with* $E' = \bigcup_{q \in E} \{q' \mid \delta_1^a(q') = q\}$,

The mirror operation is describable, indeed, for any DFA A_1, the mirror of $L(A_1)$ is $L(\mathfrak{Mirror}(A_1))$. Applying the $\mathfrak{Mirror}$ modifier to the automaton A of Figure 1 leads to the DFA of Figure 2.
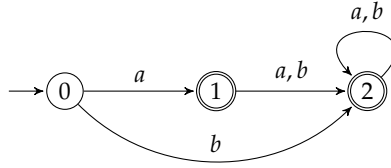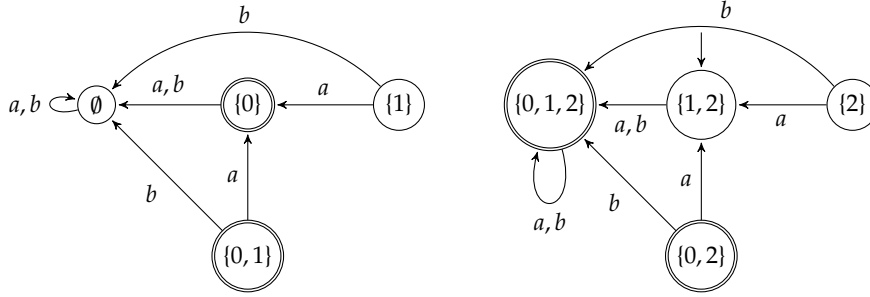

Fig. 1: The automaton A.

Fig. 2: The automaton $\mathfrak{Mirror}(A)$.

For some usual operations on languages (Comp, Union, Inter, Xor, Conc, Star and SRoot), we give one of their modifiers in Table 1. Thus these operations are describable.

3.2 Properties

We want to show that there exists non-describable operations (Example 4). Corollary 1 allows us to show this fact. We also want to prove that there exists a modifier for the composition of describable operations (Section 5). In order to do this, we use the fact that the composition of two modifiers is a modifier (Proposition 2 and Corollary 2).

We thus investigate two kinds of properties:

- commutation with respect to alphabetic renaming and restriction,

- stability by composition.

For the first property, we consider three alphabets X, X' and Y with $X \cap X' = \emptyset$, a bijection φ from X to Y, naturally extended as an isomorphism of monoids from X^* to Y^*, and $\eta : 2^{(X \cup X')^*} \to 2^{Y^*}$ defined by $\eta(L) = \varphi(L \cap X^*)$. We have

Claim 1 *If $A = (X \cup X', Q, i, F, \delta)$ is a DFA recognizing a language L then $\eta(L)$ is the regular language recognized by $(Y, Q, i, F, \delta_\bullet)$ where $\delta_\bullet^y = \delta^{\varphi^{-1}(y)}$ for any $y \in Y$.*

Proposition 1 *Let $\otimes$ be a k-ary describable operation. For any $\underline{L} \in (2^{(X \cup X')^*})^k$ we have*

$$\otimes (\eta(L_1), \cdots, \eta(L_k)) = \eta(\otimes \underline{L}).$$

Proof: Let $\underline{A}$ be a k-tuple of CDFA $A_j = (X \cup X', Q_j, i_j, F_j, \delta_j)$ such that $L(A_j) = L_j$. Since $\otimes$ is describable, there exists a modifier $\mathfrak{m} = (\mathfrak{Q}, \iota, \mathfrak{f}, \mathfrak{d})$ such that $L(\mathfrak{m}\underline{A}) = \otimes \underline{L}$. We have $\mathfrak{m}\underline{A} = (X \cup X', \mathfrak{Q}\underline{Q}, \iota(\underline{Q}, \underline{i}, \underline{F}), \mathfrak{f}(\underline{Q}, \underline{i}, \underline{F}), \delta)$ with $\delta^a = \mathfrak{d}(\underline{i}, \underline{F}, \underline{\delta^a})$. Then by Claim 1, the language $\eta(\otimes \underline{L})$ is recognized by the CDFA $A_\square = (Y, \mathfrak{Q}\underline{Q}, \iota(\underline{Q}, \underline{i}, \underline{F}), \mathfrak{f}(\underline{Q}, \underline{i}, \underline{F}), \delta_\square)$ with $\delta_\square^a = \delta^{\varphi^{-1}(a)} = \mathfrak{d}(\underline{i}, \underline{F}, \underline{\delta^{\varphi^{-1}(a)}})$. Now, let $\underline{A_\diamond}$ be the k-tuple of CDFA $A_{\diamond_j} = (Y, Q_j, i_j, F_j, \delta_{\diamond_j})$ with $\delta_{\diamond_j}^a = \delta_j^{\varphi^{-1}(a)}$. Clearly, by Claim 1, $A_{\diamond_j}$

recognizes $\eta(L_j)$. Since $\otimes$ is describable, $\mathfrak{m}\underline{A_\diamond} = (Y, \mathfrak{Q}\underline{Q}, \iota(\underline{Q}, \underline{i}, \underline{F}), \mathfrak{f}(\underline{Q}, \underline{i}, \underline{F}), \delta_\diamond)$ with $\delta_\diamond^a = \mathfrak{d}(\underline{i}, \underline{F}, \delta_\diamond^a) = \mathfrak{d}(\underline{i}, \underline{F}, \underline{\delta^{\varphi^{-1}(a)}}) = \delta_\square^a$ which ends the proof. $\qquad\qquad\square$

Immediately as special cases of the previous proposition, we obtain:

Corollary 1 *Let $\otimes$ be a k-ary describable operation and Y be an alphabet. Let $\underline{L}$ be a k-tuple of regular languages over Y. Then*

- *If $X \subset Y$ then $\otimes(L_1 \cap X^*, \cdots, L_k \cap X^*) = \otimes\underline{L} \cap X^*$.*

- *For any bijection $\sigma : Y \to Y$ extended as an automorphism of monoids, we have $\otimes(\sigma(L_1), \cdots, \sigma(L_k)) = \sigma(\otimes\underline{L})$.*
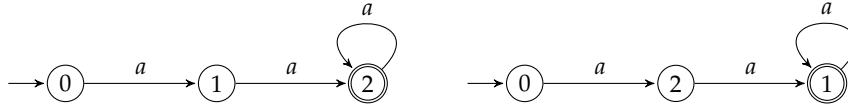
Example 4 *This result allows us to build examples of non-describable operations.*

- *We consider the binary operation defined by $\otimes(L_1, L_2) = L_1 \cdot L_2^{-1} = \{u \mid uv \in L_1 \text{ for some } v \in L_2\}$. This operation is not describable because it violates the first condition of Corollary 1. For instance, let $Y = \{a, b, c\}$, $L_1 = \{abc\}$, and $L_2 = \{c\}$. We have $\otimes(L_1 \cap \{a, b\}^*, L_2 \cap \{a, b\}^*) = \emptyset.\emptyset^{-1} = \emptyset$ while $\otimes(L_1, L_2) \cap \{a, b\}^* = \{ab\}$.*

- *We consider the unary operation defined by $\otimes(L) = L \setminus \{a\}$ if the words a and a^2 belong to L and $\otimes(L) = L$ otherwise. This operation satisfies the first condition of Corollary 1 but it violates the second one. Indeed, if $Y = \{a, b\}$ then $\otimes(\{a, a^2\}) = \{a^2\}$ while $\otimes(\{b, b^2\}) = \{b, b^2\}$. So it is not describable.*
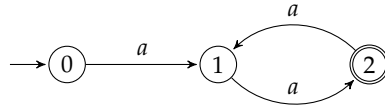
Remark 1 *There exist k-modifiers that can not be associated to operations. For instance, consider the modifier $\mathfrak{Fto}1 = (\mathfrak{Q}, \iota, \mathfrak{f}, \mathfrak{d})$ such that*

- *$\mathfrak{Q}Q = Q$,*

- *$\iota(Q, i, F) = i$.*

- *$\mathfrak{f}(Q, i, F) = F$,*

- *$\mathfrak{d}(i, F, \delta_1^a)(q) = \delta_1^a(q)$ if $q \notin F$ and $\mathfrak{d}(i, F, \delta_1^a)(q) = \begin{cases} 1 & \text{if } 1 \in Q \\ \delta_1^a(q) & \text{otherwise} \end{cases}$ if $q \in F$.*

If A_1 and A_1' are two deterministic automata recognizing the same language then we have in general $L(\mathfrak{Fto}1(A_1)) \neq L(\mathfrak{Fto}1(A_1'))$ because the recognized language depends on the labels of the states of A_1 and A_1'. For instance, the two following automata recognize the same language a^2a^*.

But applying $\mathfrak{Fto}1$ on the first one gives

which recognizes $(aa)^+$ *while* $\mathfrak{F}\mathfrak{to}1$ *lets the second automaton unchanged.*

Modifiers can be seen as functions on automata and as such can be composed as follows:

$$\mathfrak{m}_1 \circ_j \mathfrak{m}_2(A_1, \dots, A_{k_1+k_2-1}) = \mathfrak{m}_1(A_1, \dots, A_{j-1}, \mathfrak{m}_2(A_j, \dots, A_{j+k_2-1}), A_{j+k_2}, \dots, A_{k_1+k_2-1}).$$

We have

Proposition 2 *The composition* $\mathfrak{m}_1 \circ_j \mathfrak{m}_2$ *is a modifier.*

Proof: Let $\mathfrak{m}_1 = (\mathfrak{Q}^{(1)}, \iota^{(1)}, \mathfrak{f}^{(1)}, \mathfrak{d}^{(1)})$ be a k_1-ary modifier and $\mathfrak{m}_2 = (\mathfrak{Q}^{(2)}, \iota^{(2)}, \mathfrak{f}^{(2)}, \mathfrak{d}^{(2)})$ be a k_2-ary modifier. We define the $(k_1 + k_2 - 1)$-ary modifier $(\mathfrak{Q}, \iota, \mathfrak{f}, \mathfrak{d})$ by

- $\mathfrak{Q}\underline{Q} = \mathfrak{Q}^{(1)}\widehat{\underline{Q}}$ with $\widehat{\underline{Q}} = (Q_1, \dots, Q_{j-1}, \mathfrak{Q}^{(2)}(Q_j, \dots, Q_{j+k_2-1}), Q_{j+k_2}, \dots, Q_{k_1+k_2-1})$

- $\iota(\underline{Q}, \underline{i}, \underline{F}) = \iota^{(1)}(\widehat{\underline{Q}}, \widehat{\underline{i}}, \widehat{\underline{F}})$,
  with $\widehat{\underline{i}} = (i_1, \dots, i_{j-1}, \iota^{(2)}((Q_j, \dots, Q_{j+k_2-1}), (i_j, \dots, i_{j+k_2-1}), (F_j, \dots, F_{j+k_2-1})), i_{j+k_2}, \dots, i_{k_1+k_2-1})$ and
  $\widehat{\underline{F}} = (F_1, \dots, F_{j-1}, \mathfrak{f}^{(2)}((Q_j, \dots, Q_{j+k_2-1}), (i_j, \dots, i_{j+k_2-1}), (F_j, \dots, F_{j+k_2-1})), F_{j+k_2}, \dots, F_{k_1+k_2-1})$.

- $\mathfrak{f}(\underline{Q}, \underline{i}, \underline{F}) = \mathfrak{f}^{(1)}(\widehat{\underline{Q}}, \widehat{\underline{i}}, \widehat{\underline{F}})$.

- $\mathfrak{d}(\underline{i}, \underline{F}, \underline{\delta^a}) = \mathfrak{d}^{(1)}(\widehat{\underline{i}}, \widehat{\underline{F}}, \widehat{\underline{\delta}}))$
  with $\widehat{\underline{\delta}} = (\delta_1^a, \dots, \delta_{j-1}^a, \mathfrak{d}^{(2)}((i_j, \dots, i_{j+k_2-1}), (F_j, \dots, F_{j+k_2-1}), (\delta_j^a, \dots, \delta_{j+k_2-1}^a)), \delta_{j+k_2}^a, \dots, \delta_{k_1+k_2-1}^a)$

We check that $(\mathfrak{Q}, \iota, \mathfrak{f}, \mathfrak{d})$ acts on automata as $\mathfrak{m}_1 \circ_j \mathfrak{m}_2$. □

Corollary 2 *Let* $\otimes$ *be a* k_1-*ary* $\mathfrak{m}_1$-*describable operation and* $\oplus$ *be a* k_2-*ary* $\mathfrak{m}_2$-*describable operation. Then the operation* $\otimes \circ_j \oplus$ *is a* $(k_1 + k_2 - 1)$-*ary* $(\mathfrak{m}_1 \circ_j \mathfrak{m}_2)$-*describable operation for any* $j \in \{1, \dots, k_1\}$.

Proof: Let $L_1, \dots, L_{k_1+k_2-1}$ be regular languages recognized respectively by DFAs $A_1, \dots, A_{k_1+k_2-1}$. Since $\oplus$ and $\otimes$ are describable, we have

$$
\begin{aligned}
&\otimes \circ_j \oplus(L_1, \dots, L_{k_1+k_2-1}) \\
=\ &\otimes(L_1, \dots, L_{j-1}, \oplus(L_j, \dots, L_{j+k_2-1}), L_{j+k_2}, \dots, L_{k_1+k_2-1}) \\
=\ &L(\mathfrak{m}_1(A_1, \dots A_{j-1}, \mathfrak{m}_2(A_j, \dots, A_{j+k_2-1}), A_{j+k_2}, \dots, A_{k_1+k_2-1})) \\
=\ &L(\mathfrak{m}_1 \circ_j \mathfrak{m}_2(A_1, \dots, A_{k_1+k_2-1})).
\end{aligned}
$$

From Proposition 2, $\mathfrak{m}_1 \circ_j \mathfrak{m}_2$ is a modifier. Therefore $\otimes \circ_j \oplus$ is describable. □

4 Monsters

One-monster automata of size n are minimal DFAs having n^n letters representing every function from $[\![n]\!]$ to $[\![n]\!]$. There are 2^n different 1-monster automata depending on the set of their final states. The idea of a k-monster is to have a common alphabet for k automata.

The idea of using combinatorial objects to denote letters has already been used by Sakoda and Sipser [19] to obtain results for two-way automata, or by Birget [1] to obtain deterministic state complexity.
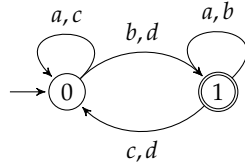
4.1 Definitions

Definition 3 *A k-monster is a k-tuple of automata $\underline{M}_{n,F} = (M_1, \ldots, M_k)$ where each $M_j = (\Sigma, [\![n_j]\!], 0, F_j, \delta_j)$ is defined by*

- *the common alphabet $\Sigma = [\![n_1]\!]^{[\![n_1]\!]} \times [\![n_2]\!]^{[\![n_2]\!]} \times \cdots \times [\![n_k]\!]^{[\![n_k]\!]}$,*

- *the set of states $[\![n_j]\!]$,*

- *the initial state 0,*

- *the set of final states F_j,*

- *the transition function δ_j defined by $\delta_j(q, \underline{g}) = g_j(q)$ for $\underline{g} = (g_1, \ldots, g_k) \in \Sigma$, i.e. $\underline{\delta}^{\underline{g}} = \underline{g}$.*
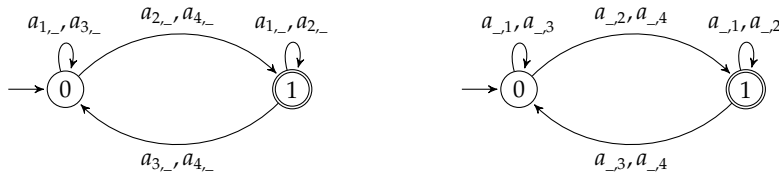
Example 5 (*k*-monster for $k = 1$ and $k = 2$)

- *The 1-monster $M_{2,\{1\}}$ is given by the following automaton*

Each symbol codes a function $\{0, 1\} \to \{0, 1\}$.

$$a = [01], \ b = [11], \ c = [00], \ and \ d = [10].$$

- *The 2-monster $M_{(2,2),(\{1\},\{1\})}$ is given by the following pair of automata on an alphabet with $2^2 \times 2^2 = 16$ symbols where $a_{i,\_}$ (respectively $a_{\_,j}$) denotes the set of transitions $a_{i,x}$ (respectively $a_{x,j}$) for $x \in \{1, \ldots, 4\}$:*

Each symbol codes a pair of functions, denoted by the word of their image.

$$
\begin{array}{llll}
a_{1,1} = [01, 01] & a_{1,2} = [01, 11] & a_{1,3} = [01, 00] & a_{1,4} = [01, 10] \\
a_{2,1} = [11, 01] & a_{2,2} = [11, 11] & a_{2,3} = [11, 00] & a_{2,4} = [11, 10] \\
a_{3,1} = [00, 01] & a_{3,2} = [00, 11] & a_{3,3} = [00, 00] & a_{3,4} = [00, 10] \\
a_{4,1} = [10, 01] & a_{4,2} = [10, 11] & a_{4,3} = [10, 00] & a_{4,4} = [10, 10].
\end{array}
$$

For instance, $a_{1,2} = [01, 11]$ means that the symbol $a_{1,2}$ labels a transition from 0 to 0 and a transition from 1 to 1 in the first automaton and a transition from 0 to 1 and a transition from 1 to 1 in the second automaton.

4.2   Using monsters to compute state complexity

If an operation is describable, it is sufficient to study the behavior of its modifiers over monsters to compute its state complexity.

Theorem 1 *Let $\mathfrak{m}$ be a modifier and $\otimes$ be an $\mathfrak{m}$-describable operation. We have*

$$\mathrm{sc}_{\otimes}\underline{n} = \max\{\#_{Min}(\mathfrak{m}\underline{M}_{\underline{n},\underline{F}}) \mid \underline{F} \subset [\![n_1]\!] \times \ldots \times [\![n_k]\!]\}.$$

Proof: Let $\underline{A}$ be a k-tuple of automata having $\underline{n}$ states and having $\underline{F}$ as set of final states recognizing a k-tuple of languages $\underline{L}$ over an alphabet Σ. Up to a relabelling, we assume that $A_i = (\Sigma, [\![n_i]\!], 0, F_i, \delta_i)$ for $i \in \{1, \ldots, k\}$.
Let δ_A be the transition function of $\mathfrak{m}\underline{A}$, and δ_M the transition function of $\mathfrak{m}\underline{M}_{\underline{n},\underline{F}}$. By definition of a modifier, the states of $\mathfrak{m}\underline{A}$ and of $\mathfrak{m}\underline{M}_{\underline{n},\underline{F}}$ are the same. For any letter a, and any state q of $\mathfrak{m}\underline{A}$, we have:

$$\delta_A^a(q) = \mathfrak{d}((0, \ldots, 0), \underline{F}, \underline{\delta^a})(q) = \mathfrak{d}((0, \ldots, 0), \underline{F}, \underline{\delta_M^a})(q) = \delta_M^{\underline{\delta^a}}(q).$$

And so, for any word w over alphabet Σ:

$$\delta_A^w(q) = \delta_M^{\underline{\delta^w}}(q).$$

Therefore, all states accessible in $\mathfrak{m}\underline{A}$ are also accessible in $\mathfrak{m}\underline{M}_{\underline{n},\underline{F}}$, and, for any word w over the alphabet Σ, $\delta_A^w(q) \in \mathfrak{f}(([\![n_1]\!], \ldots, [\![n_k]\!]), (0, \ldots, 0), \underline{F})$ if and only if $\delta_M^{\underline{\delta^w}}(q) \in \mathfrak{f}(([\![n_1]\!], \ldots, [\![n_k]\!]), (0, \ldots, 0), \underline{F})$, which implies that all pairs of states separable in $\mathfrak{m}\underline{A}$ are also separable in $\mathfrak{m}\underline{M}_{\underline{n},\underline{F}}$. Therefore,

$$\#_{\mathrm{Min}}\mathfrak{m}\underline{A} \leq \#_{\mathrm{Min}}\mathfrak{m}\underline{M}_{\underline{n},\underline{F}}.$$

$\square$

Example 6 (Mirror modifier of a 1-monster) *Let us now compute the automaton $\mathfrak{Mirror}(M_{n_1, \{n_1-1\}})$ as in Example 3.*
We show that the automaton $\mathfrak{Mirror}(M_{n_1, \{n_1-1\}})$ is minimal when $n_1 > 1$. Indeed,

- *Each state is accessible. Let g_E be the symbol that sends each element of a set $E \subset [\![n_1]\!]$ to $n_1 - 1$ and the others ($[\![n_1]\!] \setminus E$) to 0. Then, we have $\delta^{g_E}(n_1 - 1) = g_E^{-1}(n_1 - 1) = E$ (Notice that it also works with $E = \emptyset$).*

- *States are pairwise non-equivalent. Let E and E' be two distinct states of $\mathfrak{Mirror}(M_{n_1, \{n_1-1\}})$. We assume there exists $i \subset E \setminus E'$. Let g be the symbol sending 0 to i and the other states to $j \neq i$. The state $\delta^g(E)$ is final because $\{0\} = g^{-1}(i) \subset \delta^g(E)$ while $\delta^g(E') \subset [\![n_1]\!] \setminus \{0\}$.*
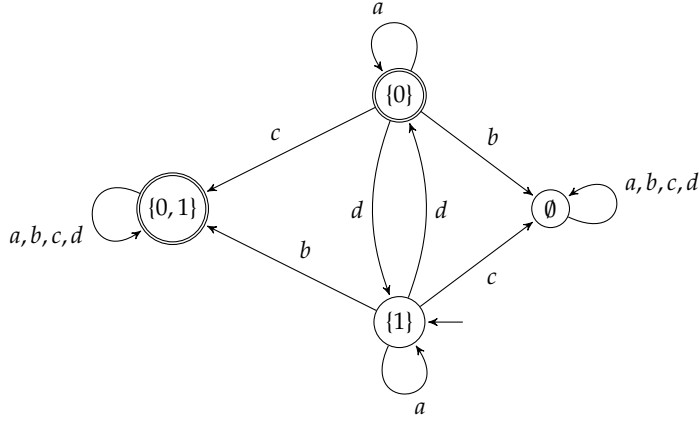
Fig. 3: The automaton $\mathfrak{Mirror}(M_{2,\{1\}})$.

We can describe in an algorithm the way to compute the state complexity of an operation using monsters and modifiers.

1. Describing the transformation with the help of a modifier whose states are represented by combinatorial objects;

2. Applying the modifier to well-chosen k-monsters. We will have to discuss the final states;

3. Minimizing the resulting automaton and estimating its size.

5 Applications

5.1 The Star of intersection example

In this section, we illustrate our method on an operation, the star of intersection, the state complexity of which is already known [20]. After having checked the upper bound, we show that this bound is tight and that the modifier of the monster $(\mathfrak{Star} \circ \mathfrak{Inter})\underline{M}$ is a witness for this operation.

Consider the 2-modifier $\mathfrak{Star} \circ \mathfrak{Inter} = (\mathfrak{Q}, \iota, \mathfrak{f}, \mathfrak{d})$. This modifier satisfies (using Table 1 and Proposition 2)

- $\mathfrak{Q}(Q_1, Q_2) = 2^{Q_1 \times Q_2}$,

- $\iota(\underline{Q}, \underline{i}, \underline{F}) = \emptyset$,

- $\mathfrak{f}(\underline{Q}, \underline{i}, \underline{F}) = \{E \in 2^{Q_1 \times Q_2} \mid E \cap (F_1 \times F_2) \neq \emptyset\} \cup \{\emptyset\}$,

- For $\mathfrak{d}(\underline{i}, \underline{F}, \underline{\delta^a})$ we have

$$\mathfrak{d}(\underline{i}, \underline{F}, \underline{\delta^a})(E) = \begin{cases} \overline{\{(\delta_1^a(i_1), \delta_2^a(i_2))\}}^{F_1 \times F_2, (i_1, i_2)} & \text{if } E = \emptyset \\ \overline{(\delta_1^a, \delta_2^a)(E)}^{F_1 \times F_2, (i_1, i_2)} & \text{otherwise.} \end{cases}$$

For the following of the paper, we will assume that $Q_1 = [\![n_1]\!]$, $Q_2 = [\![n_2]\!]$ for some $n_1, n_2 \in \mathbb{N}$, and $i_1 = i_2 = 0$. We can see elements of $2^{[\![n_1]\!] \times [\![n_2]\!]}$ as boolean matrices of size $n_1 \times n_2$. Such a matrix will be called a tableau. We denote by $T_{x,y}$ the value of the tableau T at row x and column y. The number of 1s' in a tableau T will be denoted by $\#T$.

As a consequence of Corollary 2, for any pair of regular languages (L_1, L_2) over the same alphabet and any pair of complete deterministic automata $\underline{A} = (A_1, A_2)$ such that $L_1 = L(A_1)$ and $L_2 = L(A_2)$ we have $(L_1 \cap L_2)^* = L((\mathfrak{Star} \circ \mathfrak{Inter})\underline{A})$.

Now, let n_1 and n_2 be two positive integers and let (F_1, F_2) be a subset of $[\![n_1]\!] \times [\![n_2]\!]$. An upper bound of the state complexity of the composition of *star* and *inter* operations is obtained by maximizing the number of states of $\widehat{M}_{F_1, F_2}$ where $\widehat{M}_{F_1, F_2}$ is the automaton deduced from $(\mathfrak{Star} \circ \mathfrak{Inter})\underline{M}_{n,(F_1,F_2)}$ by removing tableaux having a 1 in $(x, y) \in F_1 \times F_2$ but no 1 in $(0, 0)$. Indeed, from Table 1 the initial state of $\mathfrak{Inter}\underline{M}_{n,(F_1,F_2)}$ is $(0, 0)$ and its set of final states is $F_1 \times F_2$. Furthermore, again from Table 1, if $A = (\Sigma, Q, q_0, F, \delta)$, the accessible states of $\mathfrak{Star}(A)$ are subsets $E \subset Q$ satisfying $E \cap F \neq \emptyset \Rightarrow q_0 \in E$. Combining these two facts, the accessible states of $(\mathfrak{Star} \circ \mathfrak{Inter})\underline{M}_{n,(F_1,F_2)}$ are subsets $E \subset [\![n_1]\!] \times [\![n_2]\!]$ satisfying $E \cap (F_1 \times F_2) \neq \emptyset \Rightarrow (0, 0) \in E$.

We first remark that the initial state of $\mathfrak{Inter}\underline{M}_{n,(0,0)}$ is the only final state. This implies that $L((\mathfrak{Star} \circ \mathfrak{Inter})\underline{M}_{n,(0,0)}) = L(\mathfrak{Inter}\underline{M}_{n,(0,0)})^* = L(\mathfrak{Inter}\underline{M}_{n,(0,0)})$, which in turn implies that $\#_{\mathrm{Min}}(\widehat{M}_{0,0}) \leq \#_{\mathrm{Min}}(\mathfrak{Inter}\underline{M}_{n,(0,0)}) \leq n_1 n_2$.

Notice also that if $\#(F_1 \times F_2) = 0$, then $\widehat{M}_{F_1, F_2}$ recognizes the empty language, which trivially implies that $\#_{\mathrm{Min}}(M) \leq 1$.

Lemma 1 *The maximal number of states of* $\widehat{M}_{F_1, F_2}$ *with* $F_1 \times F_2 \notin \{\{(0, 0)\}, \emptyset\}$ *is when* $\#(F_1 \times F_2) = 1$.

Proof:
$$
\begin{aligned}
\#\widehat{M}_{F_1, F_2} &= \#2^{[\![n_1]\!] \times [\![n_2]\!]} - \#\{T \in 2^{[\![n_1]\!] \times [\![n_2]\!]} \mid (\exists(x, y) \in F_1 \times F_2 \text{ s.t. } T_{x,y} = 1) \wedge T_{0,0} = 0\} \\
&= 2^{n_1 n_2} - (\#\{T \in 2^{[\![n_1]\!] \times [\![n_2]\!]} \mid T_{0,0} = 0\} \\
&\qquad - \#\{T \in 2^{[\![n_1]\!] \times [\![n_2]\!]} \mid \forall(x, y) \in F_1 \times F_2, T_{x,y} = 0 \wedge T_{0,0} = 0\}) \\
&= \begin{cases} 2^{n_1 n_2} - (2^{n_1 n_2 - 1} - 2^{n_1 n_2 - \#F_1 \#F_2 - 1}) & \text{if } (0, 0) \notin F_1 \times F_2 \\ 2^{n_1 n_2} - (2^{n_1 n_2 - 1} - 2^{n_1 n_2 - \#F_1 \#F_2}) & \text{otherwise} \end{cases}
\end{aligned}
$$

In conclusion, the maximal number of states of $\#\widehat{M}_{F_1, F_2}$ with $F_1 \times F_2 \notin \{\{(0, 0)\}, \emptyset\}$ is reached when $\#F_1 \times \#F_2 = 1$ and is $\frac{3}{4}2^{n_1 n_2}$. $\qquad\square$

Corollary 3 $\#_{\mathrm{Min}}((\mathfrak{Star} \circ \mathfrak{Inter})\underline{M}_{n,(F_1,F_2)}) \leq \frac{3}{4}2^{n_1 n_2}$

Proof: From Lemma 1, we maximize the number of tableaux when $\#F_1 \times \#F_2 = 1$. So the upper bound is $2^{n_1 n_2} - (2^{n_1 n_2 - 1} - 2^{n_1 n_2 - 1 - 1}) = \frac{3}{4}2^{n_1 n_2}$. $\qquad\square$

Now we show that this upper bound is the state complexity of the combination of the star and the intersection operations.

Let F_1, F_2 be $\{n_1 - 1\}, \{n_2 - 1\}$ and let $\widehat{M} = \widehat{M}_{F_1, F_2}$.

Lemma 2 *All states of* $\widehat{M}$ *are accessible.*

Proof: Let T be a state of $\widehat{M}$. Let us define an order $<$ on tableaux as $T < T'$ if and only if
    (1) $\#(T) < \#(T')$ or
    (2) ($\#(T) = \#(T')$ and $T_{n_1-1,n_2-1} = 1$ and $T'_{n_1-1,n_2-1} = 0$) or
    (3) ($\#(T) = \#(T')$ and $T_{n_1-1,n_2-1} = T'_{n_1-1,n_2-1}$ and $T_{0,0} = 1$ and $T'_{0,0} = 0$).

Let us prove the assertion by induction on non-empty tableaux of $\widehat{M}$ for the partial order $<$ (the empty tableau is the initial state of $\widehat{M}$, and so it is accessible):

The only minimal tableau for non-empty tableaux of $\widehat{M}$ and the order $<$ is the tableau with only one 1 at $(0, 0)$. This is accessible from the initial state $\emptyset$ by reading the letter $(\mathrm{Id}, \mathrm{Id})$. Let us notice that each letter is a pair of functions of $[\![n_1]\!]^{[\![n_1]\!]} \times [\![n_2]\!]^{[\![n_2]\!]}$.

Now let us take a tableau T', and find a tableau T such that $T < T'$, and T' is accessible from T. We distinguish the cases below, according to some properties of T'. For each case, we define a tableau T and a letter (f, g). For all cases, except the last one, we easily check that

    (1) $T_{0,0} = 1$ (which implies that T is a state of $\widehat{M}$),
    (2) $\delta^{(f,g)}(T) = (f, g)(T) = T'$ (where $(f, g)(T) = \{(f(i), g(j)) \mid (i, j) \in T\}$), and
    (3) $T < T'$.

- $T'_{n_1-1,n_2-1} = 0$.

    - $T'_{0,0} = 0$. Let (i, j) be the index of a 1 in T'. Define (f, g) as $((0, i), (0, j))$ where $(0, i)$ and $(0, j)$ denote transpositions, and $T = (f, g)(T')$.

    - $T'_{0,0} = 1$.
        * There exists $(i, j) \in \{1, 2, ..., n_1 - 1\} \times \{1, 2, ..., n_2 - 1\}$ such that $T'_{i,j} = 1$. Define (f, g) as $((n_1 - 1, i), (n_2 - 1, j))$, then $T = (f, g)(T')$.
        * For all $(i, j) \in \{1, 2, ..., n_1 - 1\} \times \{1, 2, ..., n_2 - 1\}$, $T'_{i,j} = 0$, $T'_{0,n_2-1} = 1$ and $T'_{n_1-1,0} = 1$. In that case, define (f, g) as $(\mathrm{Id}, (n_2 - 1, 0))$, and T as $(f, g)(T')$.
        * For all $(i, j) \in \{1, 2, ..., n_1 - 1\} \times \{1, 2, ..., n_2 - 1\}$, $T'_{i,j} = 0$, $T'_{0,n_2-1} = 1$ and $T'_{n_1-1,0} = 0$. Define (f, g) as $\left(\binom{n_1-1}{0}, \mathrm{Id}\right)$. Then T is defined as

$$\begin{cases} T_{0,n_2-1} = 0 \\ T_{n_1-1,n_2-1} = 1 \\ T_{i,j} = T'_{i,j} \text{ if } (i, j) \notin (0, n_2 - 1), (n_1 - 1, n_2 - 1) \end{cases}$$

        * For all $(i, j) \in \{1, 2, ..., n_1 - 1\} \times \{1, 2, ..., n_2 - 1\}$, $T'_{i,j} = 0$, $T'_{0,n_2-1} = 0$ and $T'_{n_1-1,0} = 1$. This case is symmetrical to the case above.
        * For all $(i, j) \in \{1, 2, ..., n_1 - 1\} \times \{1, 2, ..., n_2 - 1\}$, $T'_{i,j} = 0$, $T'_{0,n_2-1} = 0$ and $T'_{n_1-1,0} = 0$. Let $(i, j) \neq (0, 0)$ be a 1 in T'. Define $(f, g) = \left(\binom{n_1-1}{i}, \binom{n_2-1}{j}\right)$, and define T as follows

$$\begin{cases} T_{i,j} = 0 \\ T_{n_1-1,n_2-1} = 1 \\ T_{i',j'} = T'_{i',j'} \text{ if } (i', j') \notin (i, j), (n_1 - 1, n_2 - 1) \end{cases}$$

- $T'_{0,0} = 1$ and $T'_{n_1-1,n_2-1} = 1$. Let $(f, g) = ((n_1 - 1, 0), (n_2 - 1, 0))$. Let T'' be the matrix obtained from T' by replacing the 1 in $(0, 0)$ by a 0. Let $T = (f, g)(T'')$. As (f, g) is a bijection over $[\![n_1]\!] \times [\![n_2]\!]$, we have $T_{0,0} = ((f, g)(T''))_{0,0} = T''_{n_1-1,n_2-1} = 1$, which means that T is a state of $\widehat{M}$, and $(f, g)(T) = (f, g)(f, g)(T'') = T''$. As $T''_{n_1-1,n_2-1} = 1$, we have $\delta^{(f,g)}(T) = T'$ in $\widehat{M}$. Furthermore, $\#T < \#T'$ implies that $T < T'$.

$\square$

Lemma 3 *All states of $\widehat{M}$ are separable.*

Proof: Let T and T' be two different states of $\widehat{M}$. There exists $(i, j) \in [\![n_1]\!] \times [\![n_2]\!]$ such that $T_{i,j} \neq T'_{i,j}$. Suppose, for example, that $T_{i,j} = 1$ and $T'_{i,j} = 0$. Let $(f, g) \in [\![n_1]\!]^{[\![n_1]\!]} \times [\![n_2]\!]^{[\![n_2]\!]}$ such that :

$$f(x) = \begin{cases} n_1 - 1 & \text{if } x = i \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad g(x) = \begin{cases} n_2 - 1 & \text{if } x = j \\ 0 & \text{otherwise} \end{cases}$$

We have $\delta^{(f,g)}(T)_{n_1-1,n_2-1} = T_{i,j} = 1$, and $\delta^{(f,g)}(T')_{n_1-1,n_2-1} = T'_{i,j} = 0$. Therefore, T and T' are separable in $\widehat{M}$.

$\square$

Theorem 1 and the previous lemmas give us :

Theorem 2 *The state complexity of the star of intersection is $\frac{3}{4}2^{n_1 n_2}$.*

5.2   The square root example

In this section, we are interested in the square root of a language L defined by $\sqrt{L} = \{x \mid xx \in L\}$. By a straightforward computation, we show easily that $sc_{\sqrt{}}(2) = 2$ and thus we investigate only the case $n > 2$. Maslov [18] showed that the square root preserves regularity and he gave a construction that can be summarized in terms of modifier by $\mathfrak{SRoot}$ (see Table 1). We first remark that this construction gives us an upper bound of n^n for the state complexity of square root. We also notice that in this case $\mathfrak{d}$ is a morphism in the sense that $\mathfrak{d}(0, F, \delta^a) \circ \mathfrak{d}(0, F, \delta^b) = \mathfrak{d}(0, F, \delta^a \circ \delta^b)$. Therefore, the application $\phi \to \mathfrak{d}(0, F, \phi)$ is a morphism of semigroups from $[\![n]\!]^{[\![n]\!]}$ to $\mathfrak{d}(0, F, [\![n]\!]^{[\![n]\!]})$. As $[\![n]\!]^{[\![n]\!]}$ can be generated by 3 elements, $\mathfrak{d}(0, F, [\![n]\!]^{[\![n]\!]})$ can be too. Therefore, there exists a witness with at most 3 letters. These two properties have been already noticed by Maslov in [18].

Let us consider the automaton $\mathfrak{SRoot}(M_{n,F})$. We notice that all the states in $\mathfrak{SRoot}(M_{n,F})$ are accessible. Indeed, the state labeled by the function g is reached from Id by reading the letter g.

For the separability, we consider a state $g_{a,b}$ defined as follows. Let $a \neq b \in [\![n]\!]$ and $g_{a,b}(x) = a$ if $x \in F$ and $g_{a,b}(x) = b$ otherwise.

Lemma 4 *For each pair $a, b \in [\![n]\!]$ such that $a \neq b$, the two states $g_{a,b}$ and $g_{b,a}$ are not separable in $\mathfrak{SRoot}(M_{n,F})$.*

Proof: Let us prove that for any h, the functions $h \circ g_{a,b}$ and $h \circ g_{b,a}$ are both final or both non final. In fact we have only two values of h to investigate: $h(a)$ and $h(b)$. If $h(a), h(b) \in F$ or $h(a), h(b) \notin F$ then the two functions $h \circ g_{a,b}$ and $h \circ g_{b,a}$ are obviously both final or both non final. Without loss of generality, suppose that $h(a) \in F$ (and so $h(b) \notin F$). We have to examine two possibilities:

- Either $0 \in F$, in this case $h(g_{a,b}(0)) = h(a) \in F$. Then $g_{a,b}(h(g_{a,b}(0))) = a$ and $h(g_{a,b}(h(g_{a,b}(0)))) = h(a) \in F$. But $h(g_{b,a}(0)) = h(b) \notin F$. Hence, $g_{b,a}(h(g_{b,a}(0))) = a$, so $h(g_{b,a}(h(g_{b,a}(0)))) \in F$. This implies that the two states are final.

- Or $0 \notin F$, in this case $h(g_{a,b}(0)) = h(b) \notin F$. Then $g_{a,b}(h(g_{a,b}(0))) = b$ and $h(g_{a,b}(h(g_{a,b}(0)))) = h(b) \notin F$. But we also have $h(g_{b,a}(0)) = h(a) \in F$. Hence, $g_{b,a}(h(g_{b,a}(0))) = b$, so $h(g_{b,a}(h(g_{b,a}(0)))) \notin F$. This implies that the two states are not final.

We deduce that the two states are not separable. Notice that the number of transformations $g_{a,b}$ is exactly $2\binom{n}{2}$. $\qquad\square$

Corollary 4

$$\mathrm{sc}_{\sqrt{}}(n) \leq n^n - \binom{n}{2}$$

Notice that the state complexity is lower than the bound given by Maslov [18].

Lemma 5 *Let $F = \{n-1\}$, and $P = \{(g, g') \mid g \neq g' \text{ and } \forall a, b \in [\![n]\!], (g, g') \neq (g_{a,b}, g_{b,a})\}$. For any pair of distinct states $(g, g') \in P$, g and g' are separable in $\mathfrak{S}\mathfrak{Root}(M_{n,F})$.*

Proof: Three cases have to be considered:

- **Suppose that $g(0) = g'(0)$.**
  Then there exists $x \in [\![n]\!] \setminus \{0\}$ such that $g(x) \neq g'(x)$. We set $h(g(0)) = x$. Hence, $h(g(h(g(0))) = h(g(x))$ and $h(g'(h(g'(0))) = h(g'(x))$. But, as $g(x) \neq g'(x)$, it is always possible to choose h such that $h(g(x)) = n-1$ while $h(g'(x)) \neq n-1$. Thus $h \circ g$ is a final state while $h \circ g'$ is not.

- **Suppose that $g(0) \neq g'(0)$ and that $\#(\mathrm{Im}(g) \cup \mathrm{Im}(g')) > 2$.**
  Without loss of generality, one assumes that there exists $x \in \mathrm{Im}(g)$ such that $x \notin \{g(0), g'(0)\}$. So the values $h(g(0))$, $h(g'(0))$ and $h(x)$ can be chosen independently each from the others. We set $h(g(0)) = y$ with $g(y) = x$, $h(g'(0)) = 0$ and $h(x) = n-1$. We check that $h \circ g$ is a final state while $h \circ g'$ is not final.

- **Suppose that $g(0) \neq g'(0)$ and that $\#(\mathrm{Im}(g) \cup \mathrm{Im}(g')) = 2$.**
  If we suppose that for any non final state x, we have $g(x) \neq g(n-1)$ and $g'(x) \neq g'(n-1)$. Since x is not final and $\#(\mathrm{Im}(g) \cup \mathrm{Im}(g')) = 2$, we have $g(x) = g(0)$ and $g'(x) = g'(0)$ (recall that 0 is not final). So, as $g(0) \neq g'(0)$, this implies $g(n-1) \neq g'(n-1)$. In other words, $g = g_{a,b}$ and $g' = g_{b,a}$ for some a, b. By contraposition, if $(g, g') \neq (g_{a,b}, g_{b,a})$ for any a, b then there exists $x \neq n-1$ such that $g(x) = g(n-1)$ or $g'(x) = g'(n-1)$. Let us denote by m the minimal element of $[\![n]\!]$ having this property and without loss of generality assume that $g(m) = g(n-1)$ (in particular, it means that for any $p < m$, $g'(p) \neq g'(n-1)$). We have two cases to consider. If $m = 0$ then we set $h(g(0)) = n-1$ and $h(g'(0)) = 0$. Obviously, $h(g'(h(g'(0)))) = 0$. On the other hand, $h(g(h(g(0))) = h(g(n-1)) = h(g(0)) = n-1$. Hence, $h \circ g$ is final while $h \circ g'$ is not final. If $m > 0$ then we have $g(m) = g'(0)$ (because there are exactly two values in the image of g and g'). Furthermore, $g'(n-1) \neq g'(0)$ and so $g'(n-1) = g(0)$. We set $h(g(0)) = m$ and $h(g'(0)) = n-1$. We have $h(g(h(g(0)))) = h(g(m)) = h(g'(0)) = n-1$.

On the other hand, $h(g'(h(g'(0)))) = h(g'(n-1)) = h(g(0)) = m \neq n-1$. It follows that $h \circ g$ is final while $h \circ g'$ is not final.

$\square$

The following theorem follows directly from Corollary 4 and Lemma 5.

Theorem 3 $\mathrm{sc}_{\sqrt{}}(n) = n^n - \binom{n}{2}$.

In order to show some restriction for alphabets of size < 3, some lemmas will be given. We assume that $n > 2$.

Lemma 6 *Any submonoid of $[\![n]\!]^{[\![n]\!]}$ generated by two distinct elements is a proper submonoid of $[\![n]\!]^{[\![n]\!]}$.*

Proof: Suppose that $[\![n]\!]^{[\![n]\!]}$ is generated by two elements f and g. Recall first that we need at least two permutations for generating the symmetric group $\mathfrak{S}_n$ (see [9]). We notice also that $[\![n]\!]^{[\![n]\!]} \setminus \mathfrak{S}_n$ is an ideal of $[\![n]\!]^{[\![n]\!]}$, that is if $t \in [\![n]\!]^{[\![n]\!]} \setminus \mathfrak{S}_n$ and $t' \in [\![n]\!]^{[\![n]\!]}$ then $t \circ t', t' \circ t \in [\![n]\!]^{[\![n]\!]} \setminus \mathfrak{S}_n$. This shows that $f, g \in \mathfrak{S}_n$. But since $\mathfrak{S}_n$ is a submonoid, it is stable by composition. It follows that $\mathfrak{S}_n = [\![n]\!]^{[\![n]\!]}$. Since this is absurd we deduce the result. $\square$

Lemma 7 *The monoid $[\![n]\!]^{[\![n]\!]}$ is generated by the two permutations $(0,1)$ and $(0,1,\ldots,n-1)$ together with any of the contractions $\binom{i}{j}$.*

Proof: It is known (see [9]) that $[\![n]\!]^{[\![n]\!]}$ is generated by $(0,1), (0,1,\ldots,n-1)$ and $\binom{0}{1}$ and that any permutation is generated by $(0,1)$ and $(0,1,\ldots,n-1)$. Thus, let σ be a permutation sending 0 to i and 1 to j. The result is just a consequence of the equality $\sigma^{-1} \circ \binom{i}{j} \circ \sigma = \binom{0}{1}$. $\square$

Lemma 8 *There exists $i, j \in [\![n]\!]$ such that $i \neq j$ and $\binom{i}{j} \notin \{g_{p,q} \mid p, q \in [\![n]\!], p \neq q\}$.*

Proof: Since $n > 2$, either $\#F > 1$ or $n - \#F > 1$. Assume without loss of generality $\#F > 1$. Let $i, i' \in F$ and $j \in [\![n]\!] \setminus F$. We have $\binom{i}{j} \notin \{g_{p,q} \mid p, q \in [\![n]\!], p \neq q\}$ otherwise we must have $g_{p,q}(i) = p = j$ and $g_{p,q}(i') = p = i'$ which is impossible since $j \neq i'$. $\square$

Proposition 3 *For any regular language L over an alphabet with at most two letters, if $\mathrm{sc}(L) = n > 2$ then $\mathrm{sc}(\sqrt{L}) < \mathrm{sc}_{\sqrt{}}(n)$.*

Proof: Let L be a language with $\mathrm{sc}(L) = n > 2$ and $A = (\{a, b\}, [\![n]\!], \{0\}, F, \cdot)$ be a minimal CDFA recognizing L. Since $\mathrm{sc}(L) = n > 2$, the set of final states F is a proper subset of $[\![n]\!]$, otherwise $L = \Sigma^*$ and $\mathrm{sc}(L) = 1$. Since the application $\phi \to \eth(0, F, \phi)$ is a morphism of semigroups, the set of the states of $\mathfrak{SRoot}(A)$ is a submonoid M of $[\![n]\!]^{[\![n]\!]}$ generated by two elements. Now, we just have to prove that any submonoid of $[\![n]\!]^{[\![n]\!]}$ generated by two elements has strictly less than $n^n - \binom{n}{2}$ elements. Suppose that $\#_{\mathrm{Min}}(\mathfrak{SRoot}(A)) = \mathrm{sc}_{\sqrt{}}(n)$. Thus we have $t \notin \{g_{p,q} \mid p, q \in [\![n]\!], p \neq q\}$ implies $t \in M$. Obviously, we have $(0,1), (0,1,\ldots,n-1) \notin \{g_{p,q} \mid p, q \in [\![n]\!], p \neq q\}$ and so $(0,1), (0,1,\ldots,n-1) \in M$. Furthermore, we have: From Lemma 8 there exists $i, j \in [\![n]\!]$ such that $i \neq j$ and $\binom{i}{j} \in M$. So by Lemma 7, $M = [\![n]\!]^{[\![n]\!]}$. But, by Lemma 6, as M has only two generators,

it is a proper submonoid of $[\![n]\!]^{[\![n]\!]}$ which contradicts the previous sentence. So there exists a transformation $t \notin \{g_{p,q} \mid p, q \in [\![n]\!], p \neq q\}$ such that $t \notin M$ and thus $\#_{\mathrm{Min}}(\mathfrak{SRoot}(A)) < n^n - \binom{n}{2} = \mathrm{sc}_{\sqrt{}}(n)$. $\square$

Let us notice that Krawetz *et al.* [17] found a very similar result not quite for square root, but for the closely related operation Root(L) = $\{w \mid \exists n$ such that $w^n \in L\}$.

6 Conclusion

New tools for computing state complexity are provided. As there is a witness among monster automata, one can focus on them to obtain a tight bound for state complexity. One of our future works is to use these tools on operations where the bound is not tight or not known as cyclic shift or star of xor. As these tools produce very large size alphabet, it remains to study how it is possible to improve this size by obtaining in some cases a constant size alphabet.

The authors learned that Sylvie Davies has independently and in the same time obtained some of the results presented in this paper. In particular, she described our monster approach as the OLPA (One letter Per Action) one. Her work can be found in [7].

References

[1] Jean-Camille Birget. Intersection and union of regular languages and state complexity. *Inf. Process. Lett.*, 43(4):185–190, 1992.

[2] Janusz A. Brzozowski. In search of most complex regular languages. *Intern. J. of Foundations of Comp. Sc.*, 24(6):691–708, 2013.

[3] Pascal Caron, Jean-Gabriel Luque, Ludovic Mignot, and Bruno Patrou. State complexity of catenation combined with a boolean operation: A unified approach. *Int. J. Found. Comput. Sci.*, 27(6):675–704, 2016.

[4] Pascal Caron, Jean-Gabriel Luque, and Bruno Patrou. State complexity of catenation combined with boolean operations. *CoRR*, abs/1707.03174, 2017.

[5] Pascal Caron, Jean-Gabriel Luque, and Bruno Patrou. State complexity of multiple catenations. *Fundam. Inform.*, 160(3):255–279, 2018.

[6] Bo Cui, Yuan Gao, Lila Kari, and Sheng Yu. State complexity of two combined operations: Catenation-union and catenation-intersection. *Int. J. Found. Comput. Sci.*, 22(8):1797–1812, 2011.

[7] Sylvie Davies. A general approach to state complexity of operations: Formalization and limitations. In Mizuho Hoshi and Shinnosuke Seki, editors, *Developments in Language Theory - 22nd International Conference, DLT 2018, Tokyo, Japan, September 10-14, 2018, Proceedings*, volume 11088 of *Lecture Notes in Computer Science*, pages 256–268. Springer, 2018.

[8] Michael Domaratzki. State complexity of proportional removals. *Journal of Automata, Languages and Combinatorics*, 7(4):455–468, 2002.

[9] Olexandr Ganyushkin and Volodymyr Mazorchuk. *Classical finite transformation semigroups: an introduction*. Algebra and Applications. Springer, Dordrecht, 2008.

[10] Yuan Gao, Nelma Moreira, Rogério Reis, and Sheng Yu. A survey on operational state complexity. *Journal of Automata, Languages and Combinatorics*, 21(4):251–310, 2017.

[11] Yuan Gao, Kai Salomaa, and Sheng Yu. The state complexity of two combined operations: Star of catenation and star of reversal. *Fundam. Inform.*, 83(1-2):75–89, 2008.

[12] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading, MA, 1979.

[13] Jozef Jirásek, Galina Jirásková, and Alexander Szabari. State complexity of concatenation and complementation. *Int. J. Found. Comput. Sci.*, 16(3):511–529, 2005.

[14] Galina Jirásková. State complexity of some operations on binary regular languages. *Theor. Comput. Sci.*, 330(2):287–298, 2005.

[15] Galina Jirásková and Alexander Okhotin. State complexity of cyclic shift. *ITA*, 42(2):335–360, 2008.

[16] Galina Jirásková and Alexander Okhotin. On the state complexity of star of union and star of intersection. *Fundam. Inform.*, 109(2):161–178, 2011.

[17] Bryan Krawetz, John Lawrence, and Jeffrey Shallit. State complexity and the monoid of transformations of a finite set. *Int. J. Found. Comput. Sci.*, 16(3):547–563, 2005.

[18] A. N. Maslov. Estimates of the number of states of finite automata. *Soviet Math. Dokl.*, 11:1373–1375, 1970.

[19] William J. Sakoda and Michael Sipser. Nondeterminism and the size of two way finite automata. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 275–286, New York, NY, USA, 1978. ACM.

[20] Arto Salomaa, Kai Salomaa, and Sheng Yu. State complexity of combined operations. *Theor. Comput. Sci.*, 383(2-3):140–152, 2007.

[21] Sheng Yu. State complexity of regular languages. *Journal of Automata, Languages and Combinatorics*, 6(2):221, 2001.

[22] Sheng Yu, Qingyu Zhuang, and Kai Salomaa. The state complexities of some basic operations on regular languages. *Theoret. Comput. Sci.*, 125(2):315–328, 1994.