New tools for state complexity

Pascal Caron¹  Edwin Hamel-De Le Court¹  Jean-Gabriel Luque¹  Bruno Patrou¹

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], [8], [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 ([8], [13]). 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 [13]) 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 \( \Sigma \) denote a finite alphabet. A word \( w \) over \( \Sigma \) is a finite sequence of symbols of \( \Sigma \). The length of \( w \), denoted by \(|w|\), is the number of occurrences of symbols of \( \Sigma \) 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 \( \Sigma \) is denoted by \( \Sigma^* \).

The empty word is denoted by \( \epsilon \). A language is a subset of \( \Sigma^* \). 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 \( \Sigma \) 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 \( \delta \) 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 (CDF A) if \#\( I = 1 \) and for all \( q \in Q \), for all \( a \in \Sigma \), \#\( \delta(q, a) = 1 \).

Let \( D = (\Sigma, Q_D, I_D, F_D, \delta_D) \) be a CDF A. When there is no ambiguity, we identify \#\( D \) to \#\( Q_D \). For any word \( w \), we denote by \( \delta^w \) the function \( q \rightarrow \delta(q, w) \). Two states \( q_1, q_2 \) of \( D \) are equivalent if for any word \( w \) of \( \Sigma \), \( \delta(q_1, w) \in F_D \) if and only if \( \delta(q_2, w) \in F_D \). Such an equivalence is denoted by \( q_1 \sim q_2 \).

A CDF A is minimal if there does not exist any equivalent CDF with less states and it is well known that for any DFA, there exists a unique minimal equivalent one [12]. Such a minimal CDF A 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 \#\( \text{Min}(D) \).

In a minimal CDF A, 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 \( X \) and any integer \( k \) given by the context, we write \( X \) for \((X_1, \ldots, X_k)\). The state complexity of a regular language \( L \) denoted by \( sc(L) \) is the number of states of its minimal CDF A. Let \( L_n \) 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 \( L \in L_n \). A language \( L \in L_n \) 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
New tools for state complexity

3

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

Let $L$ and $L'$ be two regular languages defined over an alphabet $\Sigma$. Let $\text{Union}(L, L') = \{w \mid w \in L \lor w \in L'\}$, $\text{Inter}(L, L') = \{w \mid w \in L \land w \in L'\}$, $\text{Xor}(L, L') = \{w \mid (w \in L \land w \not\in L') \lor (w \not\in L \land w \in L')\}$, $\text{Prefin}(L) = \{w = uv \mid u \in L, v \in \Sigma^*\}$, $\text{Comp}(L) = \{w \mid w \not\in L\}$, $\text{Conc}(L, L') = \{w = uv \mid u \in L, v \in L'\}$, $\text{Star}(L) = \{w = u_1 \cdots u_n \mid u_i \in L\}$, $\text{SRoot}(L) = \{w \in \Sigma^* \mid \text{w}w \in L\}$.

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 $m$, acting on DFAs $A_1, \ldots, A_k$ over the same alphabet $\Sigma$ and producing a new DFA such that

- the alphabet of $m(A_1, \ldots, A_k)$ is $\Sigma$,
- the state configuration of $m(A_1, \ldots, A_k)$ 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 $m(A_1, \ldots, A_k)$ 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 $A_1, \ldots, A_k$ (not on the letter itself nor on any other letter or transition function).

More formally,

**Definition 1** A $k$-modifier $m$ is a 4-tuple of mappings $(\Sigma, i, f, d)$ acting on $k$ DFAs $\underline{A}_j = (\Sigma, Q_j, i_j, F_j, \delta_j)$ to build a DFA $\underline{A}$ with $A_j = (\Sigma, Q, i, F, \delta)$, where

$$Q = \Sigma Q_j, \quad i = i(Q, j, F), F = f(Q, j, F) \quad \text{and} \quad \forall a \in \Sigma, \quad \delta(a) = d(Q, j, F(a)).$$
Notice that we do not need to put explicitly the dependency of $d$ on $Q$ because the information is already present in $\delta^\iota$.

For $1$-modifiers, as $Q = (Q_1)$, we denote $\iota(Q_1, i_1, F_1)$ for $\iota(Q, i, F)$, and $d(i_1, F_1, \delta^\iota_1)$ for $d(i, F, \delta^\iota)$.

**Example 1** Consider the modifier $\text{Prefin}$ of Table 1. If $A_1 = (\Sigma, Q_1, i_1, F_1, \delta_1)$ is a complete deterministic automaton then $\text{Prefin}(A_1) = (\Sigma, Q_1, i_1, F_1, \delta)$ where for any state $q \in Q_1$ for any $a \in \Sigma$ we have $\delta^\iota(q) = \delta_1^\iota(q)$ if $q \notin F_1$ and $\delta^\iota(q) = q$ if $q \in F_1$.

For instance consider the automaton $A_1$ with the following graphical representation:

![Graphical representation of $A_1$](image)

The automaton $\text{Prefin}(A_1)$ is given by

![Graphical representation of $\text{Prefin}(A_1)$](image)

**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 CDFAs $A_1, \ldots, A_k$, we have $L(m(A_1, \ldots, A_k)) = \otimes(L(A_1), \ldots, L(A_k))$.

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

For the modifiers $\text{Union}$, $\text{Inter}$ and $\text{Xor}$, a state is an element of the cartesian product of the states of the input. For the $\text{Conc}$ 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 $\text{Star}$ modifier, a state is a subset of states of the input. For the $\text{SRoot}$ modifier, each state is a function from the set of states to the set of states of the input.
New tools for state complexity

Tab. 1: Description of modifiers for some describable operations

<table>
<thead>
<tr>
<th>Modifier</th>
<th>(Q)</th>
<th>(i)</th>
<th>(f)</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Sigma)</td>
<td>(Q)</td>
<td>(i)</td>
<td>(Q_1 \setminus F_1)</td>
<td>(\delta_i)</td>
</tr>
<tr>
<td>(\Psi)</td>
<td>(Q_1)</td>
<td>(i_1)</td>
<td>(F_1)</td>
<td>(q \rightarrow {\delta_i(q)\text{ if }q \not\in F_1\ \delta_i(q)\text{ if }q \in F_1})</td>
</tr>
<tr>
<td>(\text{Union})</td>
<td>(Q_1 \times Q_2)</td>
<td>(i_1, i_2)</td>
<td>(F_1 \times Q_2 \cup Q_1 \times F_2)</td>
<td>(\delta^a)</td>
</tr>
<tr>
<td>(\text{Inter})</td>
<td>(Q_1 \times Q_2)</td>
<td>(i_1, i_2)</td>
<td>(F_1 \times F_2)</td>
<td>(\delta^a)</td>
</tr>
<tr>
<td>(\text{Xor})</td>
<td>(Q_1 \times Q_2)</td>
<td>(i_1, i_2)</td>
<td>(F_1 \times (Q_2 \setminus F_2) \cup (Q_1 \setminus F_1) \times F_2)</td>
<td>(\delta^a)</td>
</tr>
<tr>
<td>(\text{Conc})</td>
<td>(Q_1 \times 2^{Q_2})</td>
<td>(i_1, \emptyset)</td>
<td>({(q_1, E) \mid E \cap F_2 \neq \emptyset})</td>
<td>(q_1, E \rightarrow \Xi Q_1^{i_1}(\delta_i(q_1), \delta_i^a(E)))</td>
</tr>
</tbody>
</table>
| \(\Sigma\text{Star}\) | \(2^{Q_1}\) | \(\emptyset\) | \(\{E \mid E \cap F_1 \neq \emptyset\} \cup \{\emptyset\}\) | \(E \rightarrow \left\{\begin{array}{ll}
\delta_{i_1}(i_1) & \text{ if } E = \emptyset \\
\delta_{i_1}(E)_{i_1,j_1} & \text{ otherwise}
\end{array}\right.\) |
| \(\Xi\text{Root}\) | \(Q_1^{i_1}\) | \(I_d\) | \(\left\{g \mid g^2(i_1) \in F_1\right\}\) | \(g \rightarrow (\delta^a_i \circ g)\) |

where \(E^x = E \cup \{x\}\) if \(E \cap F \neq \emptyset\) and \(E\) otherwise, and \(\Xi Q_1^{i_1}(x, E) = (x, E \cup \{y\})\) if \(x \in F\) and \((x, E)\) otherwise.

**Example 3 (Mirror modifier)**  Let us define the 1-modifier \(\text{Mirror} = (\Sigma, i, f, d)\) as:

- \(\Sigma(Q_1) = 2^{Q_1}\),
- \(i(Q_1, i, F_1) = F_1\),
- \(f(Q_1, i, F_1) = \{E \subset Q_1 \mid i \in E\}\),
- \(d(i, F_1, \delta_i^a)\) is defined as \(E \rightarrow E'\) with \(E' = \bigcup_{q \in E} \{q' \mid \delta_i^a(q') = q\}\),

The mirror operation is describable, indeed, for any DFA \(A_1\), the mirror of \(L(A_1)\) is \(L(\text{Mirror}(A_1))\).

Applying the \(\text{Mirror}\) modifier to the automaton \(A\) of Figure 1 leads to the DFA of Figure 2.

---

**Figure 1:** The automaton \(A\).
Fig. 2: The automaton $\mathcal{M}_{\text{Mirror}}(A)$.

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 $\varphi$ 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 $(Y, Q, i, F, \delta)$ where $\delta^Y = \delta^{\varphi^{-1}}(\delta)$ for any $y \in Y$.

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

$$\otimes(\eta(L_1), \ldots, \eta(L_k)) = \eta(\otimes L).$$

**Proof:** Let $A$ 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 $m = (Q, i, f, b)$ such that $L(mA) = \otimes L$. We have $mA = (X \cup X', Q, i, f(Q, i, Q, i, \delta), b)$ with $\delta = b(i, f)$. Then by Claim 1, the language $\eta(\otimes L)$ is recognized by the CDFA $A_\otimes = (Y, Q, i, \varphi(Q, i, Q, i, \delta), \varphi \delta)$ with $\delta^Y = \delta^{\varphi^{-1}}(\delta)$. Now, let $A_\otimes$ be the $k$-tuple of CDFA $A_\otimes = (Y, Q, i, F, \delta_\otimes)$ with $\delta_\otimes = \delta^{\varphi^{-1}(\delta)}$. Clearly, by Claim 1, $A_\otimes$
recognizes $\eta(L_i)$. Since $\otimes$ is describable, $mA_a = (Y, \sqcup Q, \iota(Q, i, F), \iota(Q, i, F), \delta_0)$ with $\delta_0 = \delta(i, F, \delta^i F) = 
abla(i, F, \delta^i F) = \delta_0^i$ which ends the proof.

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 $X \subseteq Y$ then $\otimes(L_1 \cap X', \ldots, L_k \cap X') = \otimes L \cap X'$.
- For any bijection $\sigma : Y \to Y$ extended as an automorphism of monoids, we have $\otimes(\sigma(L_1), \ldots, \sigma(L_k)) = \sigma(\otimes L)$.

**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 $\otimes(L_1 \cap \{a, b\}, L_2 \cap \{a, b\}) = \emptyset \cdot \emptyset^{-1} = \emptyset$ 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 $\otimes to = (Q, i, f, b)$ such that

- $\sqcup Q = Q$.
- $\iota(Q, i, F) = i$.
- $\iota(Q, i, F) = F$.
- $b(i, F, \delta^i F)(q) = \delta^i_1(q)$ if $q \notin F$ and $b(i, F, \delta^i F)(q) = \begin{cases} 1 & \text{if } 1 \in Q \\ \delta^i_1(q) & \text{otherwise} \end{cases}$ if $q \in F$.

If $A_1$ and $A'_1$ are two deterministic automata recognizing the same language then we have in general $L(\otimes to (A_1)) \neq L(\otimes to (A'_1))$ 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 $\otimes to$ on the first one gives
which recognizes \((aa)^+\) while \(\emptyset\)10 lets the second automaton unchanged.

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

\[
m_1 \circ m_2(A_1, \ldots, A_{k_1+k_2-1}) = m_1(A_1, \ldots, A_{j-1}, m_2(A_j, \ldots, A_{j+k_2-1}), A_{j+k_2}, \ldots, A_{k_1+k_2-1}).
\]

We have

**Proposition 2** The composition \(m_1 \circ m_2\) is a modifier.

**Proof:** Let \(m_1 = (\Sigma^{(1)}, \iota^{(1)}, \delta^{(1)}, \nu^{(1)})\) be a \(k_1\)-ary modifier and \(m_2 = (\Sigma^{(2)}, \iota^{(2)}, \delta^{(2)}, \nu^{(2)})\) be a \(k_2\)-ary modifier. We define the \((k_1 + k_2 - 1)\)-ary modifier \((\Sigma, \iota, \delta, \nu)\) by

\[
\begin{align*}
\hat{\Sigma} &= \Sigma^{(1)} \oplus \Sigma^{(2)}, \\
\hat{\iota} &= \iota^{(1)} \oplus \iota^{(2)}, \\
\hat{\delta} &= \delta^{(1)} \otimes \delta^{(2)}, \\
\hat{\nu} &= \nu^{(1)} \otimes \nu^{(2)}.
\end{align*}
\]

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

\[
\begin{align*}
\otimes &\circ \oplus(L_1, \ldots, L_{k_1+k_2-1}) \\
&= \otimes(L_1, \ldots, L_{j-1}, \oplus(L_j, \ldots, L_{j+k_2-1}), L_{j+k_2}, \ldots, L_{k_1+k_2-1}) \\
&= L(m_1(A_1, \ldots, A_{j-1}, m_2(A_j, \ldots, A_{j+k_2-1}), A_{j+k_2}, \ldots, A_{k_1+k_2-1})) \\
&= L(m_1 \circ m_2(A_1, \ldots, A_{k_1+k_2-1})).
\end{align*}
\]

From Proposition 3, \(m_1 \circ m_2\) is a modifier. Therefore \(\otimes \circ \oplus\) is describable. \(\square\)

## 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 \(\square\) to obtain results for two-way automata, or by Birget \(\square\) to obtain deterministic state complexity.
4.1 Definitions

Definition 3 A k-monster is a k-tuple of automata $M_{k, f} = (M_1, \ldots, M_k)$ where each $M_j = (\Sigma, \llbracket n_j \rrbracket, 0, F_j, \delta_j)$ is defined by

- the common alphabet $\Sigma = \llbracket n_1 \rrbracket \times \llbracket n_2 \rrbracket \times \cdots \times \llbracket n_k \rrbracket$,
- the set of states $\llbracket n_j \rrbracket$,
- the initial state $0$,
- the set of final states $F_j$,
- the transition function $\delta_j$ defined by $\delta_j(q, g_j) = g_j(q)$ for $g_j = (g_{1j}, \ldots, g_{kj}) \in \Sigma$, i.e. $g_j = g$.

Example 5 (k-monster for $k = 1$ and $k = 2$)

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

```
0 1
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a,c</td>
</tr>
<tr>
<td>b,d</td>
</tr>
<tr>
<td>a,b</td>
</tr>
</tbody>
</table>
```

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

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

- The 2-monster $M_{2, \llbracket 1,1 \rrbracket}$ 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 $x \in \{1, \ldots, 4\}$:

```
0 1
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a_{1,1}, a_{3,1}</td>
</tr>
<tr>
<td>a_{2,1}, a_{4,1}</td>
</tr>
<tr>
<td>a_{1,2}, a_{3,2}</td>
</tr>
<tr>
<td>a_{2,2}, a_{4,2}</td>
</tr>
<tr>
<td>a_{1,3}, a_{3,3}</td>
</tr>
<tr>
<td>a_{2,3}, a_{4,3}</td>
</tr>
<tr>
<td>a_{1,4}, a_{3,4}</td>
</tr>
<tr>
<td>a_{2,4}, a_{4,4}</td>
</tr>
</tbody>
</table>
```

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

$a_{1,1} = [01, 01] \quad a_{1,2} = [01, 11] \quad a_{1,3} = [01, 00] \quad a_{1,4} = [01, 10]$

$a_{2,1} = [11, 01] \quad a_{2,2} = [11, 11] \quad a_{2,3} = [11, 00] \quad a_{2,4} = [11, 10]$

$a_{3,1} = [00, 01] \quad a_{3,2} = [00, 11] \quad a_{3,3} = [00, 00] \quad a_{3,4} = [00, 10]$

$a_{4,1} = [10, 01] \quad a_{4,2} = [10, 11] \quad a_{4,3} = [10, 00] \quad a_{4,4} = [10, 10]$. 
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 \(m\) be a modifier and \(\otimes\) be an \(m\)-describable operation. We have

\[
\text{sc}_{\otimes m} = \max \{ \#_{\text{Min}}(mM_{A, E}) \mid E \subseteq [n_1] \times \ldots \times [n_k] \}.
\]

**Proof:** Let \(A\) be a \(k\)-tuple of automata having \(n\) states and having \(E\) as set of final states recognizing a \(k\)-tuple of languages \(L\) over an alphabet \(\Sigma\). Up to a relabelling, we assume that \(A_i = (\Sigma, [n_i], 0, F_i, \delta_i)\) for \(i \in \{1, \ldots, k\}\).

Let \(\delta_A\) be the transition function of \(mA\) and \(\delta_M\) the transition function of \(mM_{A, E}\). By definition of a modifier, the states of \(mA\) and of \(mM_{A, E}\) are the same. For any letter \(a\), and any state \(q\) of \(mA\), we have:

\[
\delta_A^a(q) = \delta((0, \ldots, 0), E, \delta_A^b(q)) = \delta((0, \ldots, 0), E, \delta_M^{a\otimes}(q)) = \delta_M^a(q).
\]

And so, for any word \(w\) over alphabet \(\Sigma\):

\[
\delta_A^w(q) = \delta_M^w(q).
\]

Therefore, all states accessible in \(mA\) are also accessible in \(mM_{A, E}\) and, for any word \(w\) over the alphabet \(\Sigma\), \(\delta_A^w(q) \in f([n_1], \ldots, [n_k], (0, \ldots, 0), E)\) if and only if \(\delta_M^w(q) \in f([n_1], \ldots, [n_k], (0, \ldots, 0), E)\), which implies that all pairs of states separable in \(mA\) are also separable in \(mM_{A, E}\).

Therefore,

\[
\#_{\text{Min}} mA \leq \#_{\text{Min}} mM_{A, E}.
\]

\(\square\)

**Example 6 (Mirror modifier of a 1-monster)** Let us now compute the automaton \(\mathcal{M}_{\text{mirror}}(M_{n_1, [n-1]})\) as in Example 3.

We show that the automaton \(\mathcal{M}_{\text{mirror}}(M_{n_1, [n-1]})\) 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 \((\llbracket n_1 \rrbracket \setminus E)\) 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 \(\mathcal{M}_{\text{mirror}}(M_{n_1, [n-1]})\). We assume there exists \(i \in E \setminus E'\). 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 \(\delta^g(E') \subset [n_1] \setminus \{0\}\).
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 $(\text{Star} \circ \text{Inter})M$ is a witness for this operation.

Consider the 2-modifier $\text{Star} \circ \text{Inter} = (\mathcal{Q}, \iota, \delta, \mathcal{F})$. This modifier satisfies (using Table 1 and Proposition 2)

- $\mathcal{Q}(Q_1, Q_2) = 2^{Q_1 \times Q_2}$,
- $\iota(Q, i) = \emptyset$,
- $\{(Q, i, F) = \{E \in 2^{Q_1 \times Q_2} | E \cap (F_1 \times F_2) \neq \emptyset\} \cup \{\emptyset\}$,
- For $\delta(i, E, \delta^a)$ we have

$$\delta(i, E, \delta^a) (E) = \begin{cases} \delta^a_1 (i_1), \delta^a_2 (i_2)) & \text{if} E = \emptyset \\ (\delta^a_1, \delta^a_2)(E) & \text{otherwise.} \end{cases}$$
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 $1$s 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 $A = (A_1, A_2)$ such that $L_1 = L(A_1)$ and $L_2 = L(A_2)$ we have $(L_1 \cap L_2) \neq \emptyset \Rightarrow q_0 \in E$. Combining these two facts, the accessible states of $A$ are subsets $E \subseteq Q$ satisfying $E \cap F \neq \emptyset \Rightarrow q_0 \in E$. We first remark that the initial state of $\tilde{\mathcal{M}}_{M,F_1,F_2}$ is $(0,0)$ and its set of final states is $F_1 \times F_2$. Furthermore, again from Table I, if $A = (\Sigma, Q, q_0, F, \delta)$, the accessible states of $\tilde{\mathcal{M}}(A)$ are subsets $E \subseteq Q$ satisfying $E \cap F \neq \emptyset \Rightarrow q_0 \in E$. Combining these two facts, the accessible states of $(\tilde{\mathcal{M}} \circ \text{Inter})M_{M,F_1,F_2}$ are subsets $E \subseteq \{n_1\} \times \{n_2\}$ satisfying $E \cap (F_1 \times F_2) \neq \emptyset \Rightarrow (0,0) \in E$.

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

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

**Proof:**

To maximize the number of states of $\mathcal{M}_{M,F_1,F_2}$, we look for the largest number of states such that $F_1 \times F_2 \neq \{(0,0)\}$. This occurs when $#F_1 = 1$ and $#F_2 = 1$, so $#(F_1 \times F_2) = 1$.

**Corollary 3** $\#_{\text{Min}}((\tilde{\mathcal{M}} \circ \text{Inter})M_{M,F_1,F_2}) \leq \frac{1}{2}2^{n_1 n_2}$

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

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

**Lemma 2** All states of $\tilde{\mathcal{M}}$ are accessible.
Proof: Let $T$ be a state of $\hat{M}$. Let us define an order $< \$such that $T < T'$ if and only if
\begin{enumerate}
\item $\#(T) < \#(T')$ or
\item $\#(T) = \#(T')$ and $T_{n_1-1,n_2-1} = 1$ and $T'_{n_1-1,n_2-1} = 0$ or
\item $\#(T) = \#(T')$ and $T_{n_1-1,n_2-1} = T'_{n_1-1,n_2-1}$ and $T_{0,0} = 1$ and $T'_{0,0} = 0$.
\end{enumerate}

Let us prove the assertion by induction on non-empty tableau of $\hat{M}$ for the partial order $<$(the empty tableau is the initial state of $\hat{M}$, and so it is accessible):
The only minimal tableau for non-empty tableaux of $\hat{M}$ 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 $(Id, Id)$. Let us notice that each letter is a pair of functions of $[n_1 \times n_2]$. Let us define an order $\pi$ on tableaux as $\langle \emptyset \rangle < \langle \emptyset, \emptyset \rangle < \langle \emptyset, 1 \rangle < \langle 0, \emptyset \rangle < \langle 0, 1 \rangle$.

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
\begin{enumerate}
\item $T_{0,0} = 1$ (which implies that $T$ is a state of $\hat{M}$),
\item $\pi(T) = (f, g)(T) = T'$ (where $(f, g)(T) = \{(f(i), g(j)) \mid (i, j) \in T\}$), and
\item $T < T'$.
\end{enumerate}

- $T_{n_1-1,n_2-1} = 0$.

- $T_{0,0} = 0$. 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 $((Id, (n_2-1, 0)), (Id, (n_2-1, 0)))$, 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 $\left(\begin{smallmatrix} n_2-1 \\ 0 \end{smallmatrix}, Id\right)$. 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 $(f, g) = \left(\begin{smallmatrix} n_1-1 \\ i \\
n_2-1 \\ j \end{smallmatrix}\right)$, and define $T$ as follows

\[
\begin{cases}
T_{i,j} = 0 \\
T_{n_1-1,n_2-1} = 1 \\
T_{i',j'} = T'_{i',j'} \text{ if } (i', j') \notin (i, j), (n_1-1, n_2-1)
\end{cases}
\]
Let us prove that for any $S_R$, if $(f, g) = ((n_1 - 1, 0), (n_2 - 1, 0))$. 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 $\hat{M}$, 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 $\hat{M}$.

Furthermore, if $T < T'$ implies that $T = T'$.

\[ \square \]

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

Proof: Let $T$ and $T'$ be two different states of $\hat{M}$. 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 $(f, g) \in [n_1]^{\times n_1} \times [n_2]^{\times n_2}$ such that:

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

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 $\hat{M}$.

\[ \square \]

Theorem 1 and the previous lemmas give us:

Theorem 2 The state complexity of the star of intersection is $32^{n_1n_2}$.

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 $sc_{\sqrt{L}}(2) = 2$ and thus we investigate only the case $n > 2$. Maslov [13] showed that the square root preserves regularity and he gave a construction that can be summarized in terms of suffix by $S_R$ (see Table 1). We first remark that this construction gives us an upper bound of $n^2$ for the state complexity of square root. We also notice that in this case $d$ is a morphism in the sense that $d(0, E, \delta') \circ d(0, E, \delta') = d(0, E, \delta' \circ \delta')$.

Therefore, the application $\phi \rightarrow d(0, E, \phi)$ is a morphism of semigroups from $[n]^{\times n}$ to $d(0, E, [n]^{\times n})$. As $[n]^{\times n}$ can be generated by 3 elements, $d(0, E, [n]^{\times n})$ can be too. Therefore, there exists a witness with at most 3 letters. These two properties have been already noticed by Maslov in [13].

Let us consider the automaton $S_R(M_{n,F})$. We notice that all the states in $S_R(M_{n,F})$ 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 $S_R(M_{n,F})$.

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:
We deduce that the two states are not separable. Notice that the number of transformations \( g_{a,b} \) is exactly \( 2^k \).

**Corollary 4**

\[
\text{sc}(n) \leq n^n - \binom{n}{2}
\]

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

**Lemma 5** Let \( F = \{n - 1\} \), and \( P = \{(g, g') \mid g \neq g'\} \) 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 \( \mathcal{R} \text{oot}(M_{a,b}) \).

**Proof:** Three cases have to be considered:

- **Suppose that \( g(0) = g'(0) \).**
  Then there exists \( x \in \mathbb{N} \setminus \{0\} \) such that \( g(x) \neq g'(x) \). We set \( h(g(0)) = x \). Hence, \( h(\mathcal{P}(g(0))) = h(g(x)) \) and \( h(g'(0)) = h(g'(x)). \) 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 \( \#(\text{Im}(g) \cup \text{Im}(g')) > 2 \).**
  Without loss of generality, one assumes that there exists \( x \in \text{Im}(g) \) such that \( x \notin \text{Im}(g, g')(0) \). 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 \( \#(\text{Im}(g) \cup \text{Im}(g')) = 2 \).**
  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 \( \#(\text{Im}(g) \cup \text{Im}(g')) = 2 \), 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 \( \mathbb{N} \) 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, \( h(\mathcal{P}(g(0))) = h(g(n - 1)) = h(g'(0)) = n - 1 \). Hence, \( h \circ g' \) is final while \( h \circ g \) is not final. If \( m > 0 \) then we have \( h(g(m)) = g'(0) \) (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 8 and Lemma 3.

**Theorem 3**  
\( \text{sc} \gamma(n) = n^n - \binom{n}{2} \).

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 \( \mathbb{N}^{[n]} \) generated by two distinct elements is a proper submonoid of \( \mathbb{N}^{[n]} \).

**Proof:** Suppose that \( \mathbb{N}^{[n]} \) is generated by two elements \( f \) and \( g \). Recall first that we need at least two permutations for generating the symmetric group \( S_n \) (see [2]). We notice also that \( \mathbb{N}^{[n]} \) is an ideal of \( \mathbb{N}^{[n]} \), that is if \( t \in \mathbb{N}^{[n]} \setminus S_n \) and \( t' \in \mathbb{N}^{[n]} \) then \( t \circ t', t' \circ t \in \mathbb{N}^{[n]} \). This shows that \( f, g \in S_n \). But since \( S_n \) is a submonoid, it is stable by composition. It follows that \( S_n = \mathbb{N}^{[n]} \). Since this is absurd we deduce the result. \[ \square \]

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

**Proof:** It is known (see [2]) that \( \mathbb{N}^{[n]} \) is generated by \((0, 1), (0, 1, \ldots, n - 1)\) and \((0, 1)\) and any permutation is generated by \((0, 1)\) and \((0, 1, \ldots, n - 1)\). Thus, let \( \sigma \) be a permutation sending 0 to \( i \) and 1 to \( j \). The result is just a consequence of the equality \( \sigma^{-1} \circ (i, j) \circ \sigma = \sigma^{-1} \circ (0, 1) \). \[ \square \]

**Lemma 8**  
There exists \( i, j \in \mathbb{N} \) such that \( i \neq j \) and \((i, j) \notin \{g_{p,q} \mid p, q \in \mathbb{N}, p \neq q\}\).

**Proof:** Since \( n > 2 \), either \#F > 1 or \( n - \#F > 1 \). Assume without loss of generality \#F > 1. Let \( i, i' \in F \) and \( j \in \mathbb{N} \setminus F \). We have \((i, j) \notin \{g_{p,q} \mid p, q \in \mathbb{N}, p \neq q\}\) 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 \( \text{sc}(L) = n > 2 \) then \( \text{sc}(\sqrt{L}) < \text{sc}(\gamma(n)) \).

**Proof:** Let \( L \) be a language with \( \text{sc}(L) = n > 2 \) and \( A = (\{a, b\}, \mathbb{N}, \{0\}, F, \cdot) \) be a minimal C DFA recognizing \( L \). Since \( \text{sc}(L) = n > 2 \), the set of final states \( F \) is a proper subset of \( \mathbb{N} \), otherwise \( L = \Sigma^* \) and \( \text{sc}(L) = 1 \). Since the application \( \phi : b(0, F, \cdot) \) is a morphism of semigroups, the set of the states of \( \mathbb{N} \) generated by two elements. Now, we just have to prove that any submonoid of \( \mathbb{N}^{[n]} \) generated by two elements has strictly less than \( n^n - \binom{n}{2} \) elements. Suppose that \( \#M(\mathbb{N}) = \text{sc}(\gamma(n)) \). Thus we have \( t \notin \{g_{p,q} \mid p, q \in \mathbb{N}, p \neq q\}\) implies \( t \in M \). Obviously, we have \((0, 1), (0, 1, \ldots, n - 1) \notin \{g_{p,q} \mid p, q \in \mathbb{N}, p \neq q\}\) and so \((0, 1), (0, 1, \ldots, n - 1) \in M \). Furthermore, we have: From Lemma 3 there exists \( i, j \in \mathbb{N} \) such that \( i \neq j \) and \((i, j) \in M \). So by Lemma 3 \( M = \mathbb{N}^{[n]} \). But, by Lemma 3 as \( M \) has only two generators,
it is a proper submonoid of \([n]^{(2)}\) which contradicts the previous sentence. So there exists a transformation \(t \notin \{g_{p,q} \mid p, q \in [n] \land p \neq q\}\) such that \(t \notin M\) and thus \(#_{\text{Min}}(\exists \text{Root}(A)) < n^n - (\begin{pmatrix} n \\ 2 \end{pmatrix}) = \sqrt[sc]{\sqrt{n}}\).

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

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


