Self-Verifying Finite Automata and Descriptional Complexity

Galina Jirásková

Self-Verifying Finite Automata and Descriptional Complexity*

Galina Jirásková

Mathematical Institute, Slovak Academy of Sciences
Grešákova 6, 040 01 Košice, Slovakia
jiraskov@saske.sk

Abstract. We survey recent results on the descriptional complexity of self-verifying finite automata. In particular, we discuss the cost of simulation of self-verifying finite automata by deterministic finite automata, and the complexity of basic regular operations on languages represented by self-verifying finite automata.

1 Introduction

A self-verifying finite automaton is a nondeterministic automaton whose state set consists of three disjoint groups of states: accepting states, rejecting states, and neutral states. On every input string, at least one computation must end in either an accepting or in a rejecting state. Moreover, there is no input string with both accepting and rejecting computations.

The existence of an accepting computation on an input string proves the membership of the string to the language. This is the same as in a nondeterministic finite automaton (NFA). However, in a self-verifying finite automaton (SVFA), the existence of a rejecting computation definitely proves that the input is not in the language. This is in contrast with NFAs, where the existence of a non-final computation leaves open the possibility that the input may be accepted by a different computation. Thus, even if the transitions are nondeterministic, when a computation of an SVFA ends in an accepting or in a rejecting state, the automaton "can trust" the outcome of that computation, and accept or reject the input. The name "self-verifying" comes from this property. SVFAs were introduced in [4], and were considered mainly in connection with probabilistic Las Vegas computations. However, as pointed in [8], they are also interesting *per se*.

Every SVFA can be converted to an equivalent deterministic finite automaton (DFA) by the standard subset construction [19]. On the other hand, every complete DFA may be viewed as a self-verifying finite automaton with all the final states being accepting, and all the non-final states being rejecting. Hence SVFAs recognize exactly the class of regular languages.

From the descriptional point of view, every n-state NFA can be simulated by a DFA of at most 2^n states [19]. This bound is known to be tight in the

* Research supported by VEGA grant 2/0084/15.

binary case [5, 14, 18]. However, Assent and Seibert [1] proved that in the DFA obtained by applying the subset construction to an SVFA some states must be equivalent. As a consequence, they obtained an upper bound for the conversion of self-verifying automata to deterministic automata in $O(2^n/\sqrt{n})$. Later this result was strengthened in [11], where the tight bound for such a conversion was given by a function $g(n)$ which grows like $3^{n/3}$. The witness languages meeting the bound $g(n)$ were defined over a binary alphabet.

The investigation of self-verifying automata was further deepened by Jirásek et al. [9]. Using the tight bound $g(n)$ from [11], it was shown that a minimal SVFA for a regular language may not be unique. Then the authors introduced an sv-fooling set lower bound technique for the number of states in SVFAs. Using this technique, they obtained tight upper bounds on the complexity of reversal, boolean operations, star, left and right quotients, and asymptotically tight upper bound for concatenation of languages represented by SVFAs.

Here we survey these results. We deal with SVFA-to-DFA conversion in Section 2, and discuss the complexity of basic regular operations on SVFAs in Section 3. To conclude this introduction, let us recall some basic notions and preliminary results. For further details, the reader may refer to [21].

All DFAs in this paper are assumed to be complete, and NFAs have a unique initial state. Sometimes we also consider NNFAs — nondeterministic finite automata with a nondeterministic choice of the initial state [22] — where we admit multiple initial states.

A *self-verifying finite automaton* (SVFA) is a tuple $A = (Q, \Sigma, \delta, s, F^a, F^r)$, where Q, Σ, δ, and s are the same as in an NFA, F^a is the set of accepting states, F^r is the set of rejecting states, and $F^a \cap F^r = \emptyset$; the remaining states in Q are called neutral. It is required that for each input string w in Σ^*, there exists at least one computation ending in an accepting or in a rejecting state, and there are no strings w such that both $\delta(s, w) \cap F^a$ and $\delta(s, w) \cap F^r$ are nonempty.

The language accepted by the SVFA A, denoted as $L^a(A)$, is the set of all input strings having a computation ending in an accepting state, while the language rejected by A, denoted as $L^r(A)$, is the set of all input strings having a computation ending in a rejecting state. It follows directly from the definition that $L^a(A) = (L^r(A))^c$ for each SVFA A. Hence, when we say that an SVFA A *accepts* a language L, we mean that $L = L^a(A)$ and $L^c = L^r(A)$.

The *state complexity* of a regular language L, $\mathrm{sc}(L)$, is defined as the smallest number of states in any DFA for L. The *state complexity of a regular operation* is the maximal state complexity of languages resulting from the operation, considered as a function of the state complexities of the operands. Similarly, the *nondeterministic state complexity* and *self-verifying state complexity* of a regular language L, denoted by $\mathrm{nsc}(L)$ and $\mathrm{svsc}(L)$, is defined as the smallest number of states in any NFA (with a unique initial state) and SVFA, respectively, for L.

Every NNFA $A = (Q, \Sigma, \delta, I, F)$ can be converted to an equivalent DFA $A' = (2^Q, \Sigma, \cdot, I, F')$, where $R \cdot a = \delta(R, a)$ for each R in 2^Q and each a in Σ, and $F' = \{R \in 2^Q \mid R \cap F \neq \emptyset\}$ [19]. The DFA A' is called the *subset automaton* of the NFA A. Let us recall two observations from [8, 11].

Proposition 1 ([8, 11]). *Let a language L be accepted by an n-state SVFA. Then the languages L and L^c are accepted by n-state NFAs.* □

Proof. Let L be accepted by an SVFA $A = (Q, \Sigma, \delta, s, F^a, F^r)$. Then L is accepted by NFA $(Q, \Sigma, \delta, s, F^a)$, while L^c is accepted by NFA $(Q, \Sigma, \delta, s, F^r)$. □

Proposition 2 ([8, 11]). *Let languages L and L^c be accepted by an m-state and n-state NNFAs, respectively. Then* $\mathrm{svsc}(L) \leq m + n + 1$. □

Proof. Let L be accepted by an m-state NNFA $N = (Q, \Sigma, \delta, I, F)$ and L^c be accepted by an n-state NNFA $N' = (Q', \Sigma, \delta', I', F')$. Then we can get an SVFA A for L with $m + n + 1$ states from NFAs N and N' as follows. We add a new initial state s going to $\delta(I, a) \cup \delta'(I', a)$ on each a in Σ. The state s is accepting if $\varepsilon \in L$, and it is rejecting otherwise. All the states in F are accepting in SVFA A, and all the states in F' are rejecting in A. □

2 SVFA-to-DFA Conversion and Minimal SVFAs

The SVFA-to-DFA conversion was first studied by Assent and Seibert [1]. Then Jirásková and Pighizzini [11] obtained a tight upper bound for such a conversion.

Proposition 3 ([1, Theorem 2.1]). *Every n-state SVFA can be converted to an equivalent DFA of at most $O(2^n/\sqrt{n})$ states.*

Proof (Proof Idea). If S and T are two reachable subset of the subset automaton of an SVFA A such that $S \subseteq T$, then S and T are equivalent. This gives an upper bound $\binom{n}{\lfloor n/2 \rfloor} \in O(2^n/\sqrt{n})$. □

Theorem 4 ([11, Theorem 9]). *Every n-state SVFA can be converted to an equivalent DFA of at most $g(n)$ states, where*

$$g(n) = \begin{cases} 1 + 3^{(n-1)/3}, & \text{if } n \bmod 3 = 1 \text{ and } n \geqslant 4, \\ 1 + 4 \cdot 3^{(n-5)/3}, & \text{if } n \bmod 3 = 2 \text{ and } n \geqslant 5, \\ 1 + 2 \cdot 3^{(n-3)/3}, & \text{if } n \bmod 3 = 0 \text{ and } n \geqslant 3, \\ n, & \text{if } n \leqslant 2. \end{cases} \tag{1}$$

Moreover, the bound $g(n)$ is tight, and can be met by a binary n-state SVFA.

Proof (Proof Idea). To an n-state SVFA A, we assign an undirected graph $G(A)$ whose vertex set is Q, and which contains an edge $\{p, q\}$ if and only if two computations starting from p and q cannot give contradictory answers on the same string. Then each reachable subset in the subset automaton of A is represented by a clique in $G(A)$. Moreover, if S and T are two subsets such that $S \cup T$ is a clique in $G(A)$, then S and T are equivalent [11, Lemma 4]. Hence the number of states in the minimal DFA for $L(A)$ is given by the number of maximal cliques in $G(A)$. Next, in $G(A)$ there is exactly one maximal clique containing the initial state of A. This results in at most $1 + f(n-1)$ states, where $f(n)$ denotes the

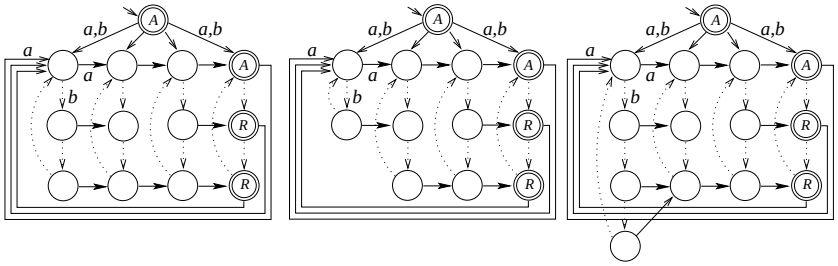

Fig. 1. The witnesses for SVFA-to-DFA conversion; $n = 13, 12$, and 14.

maximum number of possible maximal cliques in a graph with n nodes and, as shown by Moon and Moser [17, Theorem 1], we have $f(n) = 3^{n/3}$ if $n \bmod 3 = 0$, $f(n) = 4 \cdot 3^{\lfloor n/3 \rfloor - 1}$ if $n \bmod 3 = 1$, and $f(n) = 2 \cdot 3^{\lfloor n/3 \rfloor}$ if $n \bmod 3 = 2$. This gives the upper bound. For tightness, let $A = (Q, \{a, b\}, \delta, q_0, F^a, F^r)$, where $n = 1 + 3m$ and $m \geqslant 2$, see Fig. 1 (left) for $m = 13$, be an automaton defined by

$Q = \{q_0\} \cup \{(i, j) \mid 0 \leqslant i \leqslant 2, 1 \leqslant j \leqslant m\}$,
$\delta(q_0, a) = \delta(q_0, b) = \{(0, 1), (0, 2), \ldots, (0, m)\}$,
and for all i, j with $0 \leqslant i \leqslant 2$ and $1 \leqslant j \leqslant m$,
$\delta((i, j), a) = \begin{cases} \{(i, j + 1)\}, \text{ if } j < m, \\ \{(0, 1)\}, \quad \text{otherwise,} \end{cases}$
$\delta((i, j), b) = \{((i + 1) \bmod 3, j)\}$,
$F^a = \{q_0, (0, m)\}$, and $F^r = \{(1, m), (2, m)\}$.

It is shown in [11, Lemma 8] that A is an SVFA whose minimal DFAs requires $g(n)$ states. To get witnesses for $n = 3k$ or $n = 3k + 2$, we modify the SVFA A as shown in Fig. 1 (middle and right). □

Thus if we know that the minimal DFA for a language L has more then $g(n)$ states, then by Theorem 4, every SVFA for L must have at least $n + 1$ states. We use this result to show that a minimal SVFA may not be unique.

Example 5. Consider the two 7-state non-isomorphic SVFAs shown in Fig. 2. Apply the subset construction to both of them. In both cases, the subset automata restricted to the reachable states are the same. These subset automata, and therefore also the two SVFAs, accept the language $(a + b)^* a (a + b)^2$, the minimal DFA for which has 8 states. Since we have $g(6) = 7$, every SVFA for this language has at least 7 states. Hence both SVFAs in Fig. 2 are minimal. □

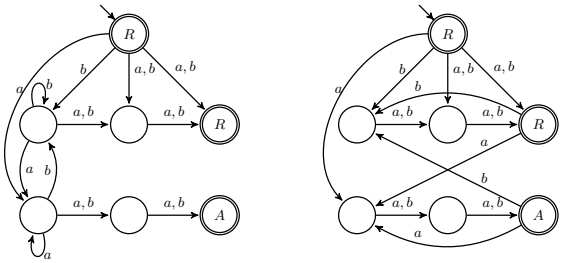

Fig. 2. Two non-isomorphic minimal SVFAs for the language $(a + b)^* a (a + b)^2$.

3 Lower Bound Methods and Operations on SVFAs

To prove that a DFA is minimal, we only need to show that all its states are reachable from the initial state, and that no two distinct states are equivalent. To prove minimality of NFAs, a fooling set lower bound method may be used [2, 6]. A *fooling set* for a language L is a set of pairs of strings $\{(u_1, v_1), \ldots, (u_n, v_n)\}$ satisfying two conditions:

(i) for each i, $u_i v_i \in L$, and

(ii) if $i \neq j$, then $u_i v_j \notin L$ or $u_j v_i \notin L$.

In the case of SVFAs, we change the two conditions. The first condition can be removed since we have either an accepting or rejecting computation on every string. Before modifying the second condition, consider the following example.

Example 6. Let L be accepted by the 3-state NFA shown in Fig. 3. Let A be an SVFA for L. Let us show that A has at least 6 states. Consider the following pairs of strings:

$(u_1, v_1) = (a^2, a^2)$	Acc	$(u_4, v_4) = (a^2ba, a^2)$	Rej
$(u_2, v_2) = (a^2b, a)$	Acc	$(u_5, v_5) = (a, a)$	Rej
$(u_3, v_3) = (a^2b^2, \varepsilon)$	Acc	$(u_6, v_6) = (ab, \varepsilon)$	Rej

The strings u_1v_1, u_2v_2, and u_3v_3 are in L, while u_4v_4, u_5v_5, and u_6v_6 are not in L, so we must have accepting and rejecting computations in A on these strings:

$$s \xrightarrow{u_1} p_1 \xrightarrow{v_1} f_1 \in F^a, \qquad s \xrightarrow{u_4} p_4 \xrightarrow{v_4} f_4 \in F^r,$$
$$s \xrightarrow{u_2} p_2 \xrightarrow{v_2} f_2 \in F^a, \qquad s \xrightarrow{u_5} p_5 \xrightarrow{v_5} f_5 \in F^r,$$
$$s \xrightarrow{u_3} p_3 \xrightarrow{v_3} f_3 \in F^a, \qquad s \xrightarrow{u_6} p_6 \xrightarrow{v_6} f_6 \in F^r.$$

Since $u_1v_2 = a^3$, and a^3 is not in L, we must have $p_1 \neq p_2$ because otherwise $s \xrightarrow{u_1} p_1 = p_2 \xrightarrow{v_2} f_2$ would be an accepting computation on u_1u_2. Similarly, u_1v_3 and u_2v_3 are not in L, so p_1, p_2, and p_3 must be pairwise distinct. On the other hand, the strings u_4v_5, u_4v_6, and u_5v_6 are in L, and therefore the states p_4, p_5, p_6 must be pairwise distinct. Next, let $1 \leq i \leq 3$ and $1 \leq j \leq 3$. If $i \leq j$, then u_jv_i is not in L, and therefore $p_i \neq p_j$ because otherwise $s \xrightarrow{u_j} p_j = p_i \xrightarrow{v_i} f_i$ would be an accepting computation on u_jv_i. Finally, if $i > j$, then u_iv_j is in L, and therefore $p_i \neq p_j$. Thus all the state p_i are pairwise distinct, so the SVFA A has at least 6 states. □

Notice that in the previous example, we were able to interchange the right sides of two pairs (u_i, v_i) and (u_j, v_j) so that at least one of the resulting strings u_iv_j and u_jv_i had a "different finality" than the concatenations u_jv_j and u_iv_i, respectively. We formalize this in the following definition.

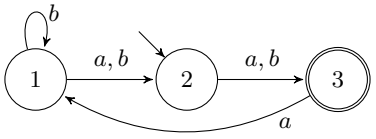

Fig. 3. An NFA for the language L in Example 6.

Definition 7. *A set of pairs of strings* $\mathcal{F} = \{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\}$ *is called an* sv-fooling set *for a language* L *if for all* i, j *with* $i \neq j$ *at least one of the following two conditions holds:*

(i) exactly one of the strings $u_i v_j$ *and* $u_j v_j$ *is in* L*, or*

(ii) exactly one of the strings $u_j v_i$ *and* $u_i v_i$ *is in* L*.*

Lemma 8 (Lower Bound Method for SVFAs). *Let* $\mathcal{F}$ *be an sv-fooling set for a language* L*. Then every SVFA for the language* L *has at least* $|\mathcal{F}|$ *states.*

Proof. Let A be an SVFA for the language L with the initial state s. Then for each $u_i v_i$, there is an accepting or a rejecting computation of SVFA A on $u_i v_i$. Fix such a computation for each $u_i v_i$. Let p_i be the state in this computation that is reached after reading u_i, and let f_i be the final state reached after reading v_i. Let us show that the states $p_1, p_2, \ldots, p_n$ must be pairwise distinct.

Assume for contradiction that there are i and j with $i \neq j$ such that $p_i = p_j$. Then we have

$s \xrightarrow{u_i} p_i = p_j \xrightarrow{v_j} f_j$ and $s \xrightarrow{u_j} p_j \xrightarrow{v_j} f_j$; and

$s \xrightarrow{u_j} p_j = p_i \xrightarrow{v_i} f_i$ and $s \xrightarrow{u_i} p_i \xrightarrow{v_i} f_i$.

It follows that there are computations on $u_i v_j$ and on $u_j v_j$ that end in state f_j. Thus either both this strings are in L, or both of them are in L^c. Moreover, there are computations on $u_j v_i$ and $u_i v_i$ that end in state f_i, so either both these strings are in L, or both of them are in L^c. Hence neither (i) nor (ii) in the definition of an sv-fooling set holds, which is a contradiction. □

Notice that the lemma above may also be applied to a model of SVFAs with multiple initial states [11, Section 5]. Hence if a language L is accepted by an n-state SVFA with multiple initial states, we cannot have an sv-fooling set of size more than n. In such a case, we can use the following observation to prove that an SVFA with a unique initial state needs one more state.

Lemma 9. *Let* $\mathcal{F} = \{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\}$ *be an sv-fooling set for* L*. For each* i*, let there exist a string* w_i *such that* $\{(u_i, v_i)\} \cup \{(\varepsilon, w_i)\}$ *is an sv-fooling set for* L*. Then every SVFA for* L *has at least* $|\mathcal{F}| + 1$ *states.*

Proof. For each pair in $\mathcal{F}$, fix an accepting or a rejecting computation as in Lemma 8. Then the unique initial state, reached after reading ε, must be different from all the states reached after reading the left part of any pair in $\mathcal{F}$. It follows that the SVFA has at least $|\mathcal{F}| + 1$ pairwise distinct states. □

Example 10. Let us continue our previous example. Define $w_1 = w_2 = w_3 = a^2$, $w_4 = w_5 = \varepsilon$, and $w_6 = a$. Notice that we have

$u_i \cdot a^2 \in L$ and $\varepsilon \cdot a^2 \notin L$ for $i = 1, 2, 3$,

$u_i \cdot \varepsilon \in L$ and $\varepsilon \cdot \varepsilon \notin L$ for $i = 4, 5$,

$u_6 \cdot a \notin L$ and $\varepsilon \cdot a \in L$.

By Lemma 9, every SVFA for L has at least 7 states. □

In what follows we use this simple methods to get tight upper bounds on the self-verifying complexity of reversal, boolean operations, star, and left and right quotients. In the case of concatenation, we get an asymptotically tight upper bound.

3.1 Reversal

If a language L is accepted by an n-state DFA A, then the language L^R is accepted by an n-state NNFA A^R obtained from A by swapping the role of the initial and final states of A, and by reversing all the transitions. By applying the subset construction to NNFA A^R, we get a DFA for L^R of at most 2^n states. The bound 2^n is known to be tight [14, 16], and the witness languages can be defined over a binary alphabet [12, 13].

If a language L is represented by an n-state NFA A, then we can construct an NNFA A^R for L^R in the same way as for DFAs. An equivalent NFA may require one more state. The upper bound $n+1$ is known to be tight, with worst-case examples defined over a binary alphabet [7, 10]. Our next result shows that the self-verifying state complexity of the reversal operation is given by the function $2n+1$. Notice that the reverse of our worst-case example is a generalization of our NFA language in Example 6.

Theorem 11 ([9]). *Let $n \geq 3$. Let L be a regular language over an alphabet Σ with $\mathrm{svsc}(L) = n$. Then $\mathrm{svsc}(L^R) \leq 2n+1$, and the bound is tight if $|\Sigma| \geq 2$.*

Proof. Let $A = (Q, \Sigma, \delta, s, F^a, F^r)$ be an SVFA for L. Then L is accepted by the n-state NFA $N = (Q, \Sigma, \delta, s, F^a)$, and L^c is accepted by the n-state NFA $N' = (Q, \Sigma, \delta, s, F^r)$ by Proposition 1. By swapping the role of initial and final states in NFAs N and N', and by reversing all the transitions, we get n-state NNFAs for languages L^R and $(L^c)^R = (L^R)^c$. By Proposition 2, we have $\mathrm{svsc}(L^R) \leq 2n+1$. This proves the upper bound.

For tightness, let L be the language accepted by the DFA A shown in Fig. 4. Construct an NFA A^R for the language L^R as described above. Denote by $[i, j]$ the set of integers $\{k \mid i \leq k \leq j\}$; notice that $[i, j] = \emptyset$ if $i > j$. Consider the following family of $2n$ subsets

$$\mathcal{R} = \big\{[1, i] \mid 1 \leq i \leq n\big\} \cup \big\{[i+1, n] \mid 1 \leq i \leq n\big\}.$$

Notice that each set in $\mathcal{R}$ is reachable in the subset automaton of the NFA A^R from the initial subset $\{n-1\}$. Thus for each subset S in $\mathcal{R}$, there is a string u_S by which the initial state $\{n-1\}$ of the subset automaton of A^R goes to S. Consider the following set of $2n$ pairs of strings:

$$\mathcal{F} = \{(u_{[1,i]}, a^{n-i}) \mid 1 \leq i \leq n\} \cup \{(u_{[j+1,n]}, a^{n-j}) \mid 1 \leq j \leq n\}.$$

Let us show that the set $\mathcal{F}$ is an sv-fooling set for the language L^R.

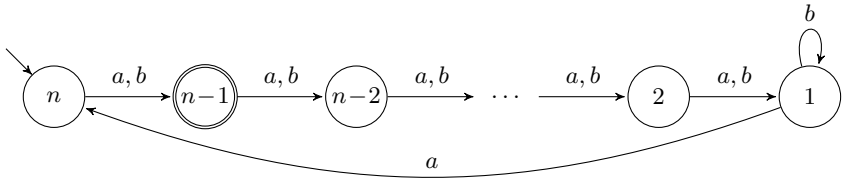

Fig. 4. The binary witness for reversal meeting the bound $2n+1$.

First, notice that the string a^{n-i} is accepted by the NFA A^R from a subset S of $[1,n]$ if and only if the state i is in the subset S. To show that $\mathcal{F}$ is an sv-fooling set for L, we have three cases to consider:

(1) Let $1 \leq i < k \leq n$. Then $u_{[1,i]} \cdot a^{n-k} \notin L^R$ and $u_{[1,k]} \cdot a^{n-k} \in L^R$.

(2) Let $1 \leq j < \ell \leq n$. Then $u_{[j+1,n]} \cdot a^{n-\ell} \in L^R$ and $u_{[\ell+1,n]} \cdot a^{n-\ell} \notin L^R$.

(3) Let $1 \leq i \leq n$ and $1 \leq j \leq n$. Here we have two subcases:

(3a) If $i \leq j$, then $u_{[j+1,n]} \cdot a^{n-i} \notin L^R$ and $u_{[1,i]} \cdot a^{n-i} \in L^R$.
(3b) If $i > j$, then $u_{[1,i]} \cdot a^{n-j} \in L^R$ and $u_{[j+1,n]} \cdot a^{n-j} \notin L^R$.

Hence we have shown that $\mathcal{F}$ is an sv-fooling set for the language L^R. Now, we use Lemma 9 to show that one more state is necessary for an SVFA to accept L^R. To this aim, let $w_i = a^{n-1}$ for $i = 1, 2, \ldots, n$, $w_{n+j} = \varepsilon$ for $j = 1, 2 \ldots, n-1$, and $w_{2n} = a$. Then we have

$\varepsilon \cdot a^{n-1} \notin L^R$ while $u_{[1,i]} \cdot a^{n-1} \in L^R$ if $1 \leq i \leq n$ and $n \geq 3$,
$\varepsilon \cdot \varepsilon \notin L^R$ while $u_{[j+1,n]} \cdot \varepsilon \in L^R$ if $1 \leq j \leq n-1$.
$\varepsilon \cdot a \in L^R$ while $u_\emptyset \cdot a \notin L^R$ since $n \geq 3$.

By Lemma 9, every SVFA for L^R has at least $2n+1$ states. □

3.2 Boolean Operations

To get a DFA for the complement of a given regular language, we only need to interchange the final and non-final states in a DFA for the given language. Formally, if a regular language L is accepted by a DFA $A = (Q, \Sigma, \delta, s, F)$, then the language L^c is accepted by the DFA $A' = (Q, \Sigma, \delta, s, Q \setminus F)$. Moreover, if A is minimal, then A' is minimal as well. It follows that the state complexities of a regular language and its complement are the same.

On the other hand, if a language is represented by an NFA, we first apply the subset construction to this NFA, and only after that we can interchange the final and non-final states. This gives an upper bound 2^n. This upper bound is known to be tight [2, 20], and witness languages can be defined over a binary alphabet [10]. Our first observation shows that the self-verifying complexity of a language and its complement are the same.

Then we consider the following four Boolean operations: intersection, union, difference, and symmetric difference. In the general case of all regular languages, the state complexity of all four operations is given by the function mn, and the worst-case examples are defined over a binary alphabet [3, 15, 19, 23]. The nondeterministic state complexity of intersection and union is mn and $m+n+1$, respectively, with witness languages defined over a binary alphabet [7].

The difference and symmetric difference on languages represented by NFAs have not been studied yet. Since both these operations require complementation, the nondeterministic state complexities $m \cdot 2^n$ and $m \cdot 2^n + n \cdot 2^m$ of difference and symmetric difference, respectively, could be expected. In the case of self-verifying state complexity, we obtain a tight upper bound mn for all four operations, with worst-case examples defined over a binary alphabet.

Theorem 12 ([9]). *Let K and L be languages over an alphabet Σ with* $\mathrm{svsc}(K) = m$ *and* $\mathrm{svsc}(L) = n$. *Then*

(i) $\mathrm{svsc}(L^c) = n$,

(ii) $\mathrm{svsc}(K \cap L), \mathrm{svsc}(K \cup L), \mathrm{svsc}(K \setminus L), \mathrm{svsc}(K \oplus L) \leq mn$, *and all the bounds are tight if* $|\Sigma| \geq 2$. □

Proof. (i) Let L be accepted by an SVFA A. To get an SVFA A' for the language L^c, we only need to interchange the accepting and rejecting states in the SVFA A. Moreover, if A is minimal, then A' is minimal as well.

(ii) Now we consider intersection. Let K and L be accepted by SVFAs $A = (Q_A, \Sigma, \delta_A, s_A, F_A^a, F_A^r)$ and $B = (Q_B, \Sigma, \delta_B, s_B, F_B^a, F_B^r)$ of m and n states. Construct the product automaton $A \times B = (Q, \Sigma, \delta, s, F^a, F^r)$, where

$Q = Q_A \times Q_B$; $s = (s_A, s_B)$;

$F^a = \{(p,q) \mid p \in F_A^a \text{ and } q \in F_B^a\}$ and $F^r = \{(p,q) \mid p \in F_A^r \text{ or } q \in F_B^r\}$;

$\delta((p,q),a) = \delta_A(p,a) \times \delta_B(q,a)$ for each (p,q) in Q and each a in Σ.

The product automaton $A \times B$ accepts $K \cap L$, and it is self-verifying.

For tightness, consider languages $K = \{w \in \{a,b\}^* \mid \#_a(w) \equiv 0 \bmod m\}$ and $L = \{w \in \{a,b\}^* \mid \#_b(w) \equiv 0 \bmod n\}$ accepted by an m-state and n-state DFAs, so also SVFAs, respectively. Then the set of pairs $\mathcal{F} = \{(a^i b^j, a^{m-i} b^{n-j}) \mid 0 \leq i \leq m-1 \text{ and } 0 \leq j \leq n-1\}$ is an sv-fooling set of size mn for $K \cap L$.

Let us continue with union and difference. Since $K \cup L = (K^c \cap L^c)^c$, and self-verifying state complexity of a language and its complement are the same, we can get an SVFA for the union of K and L as follows. We first construct SVFAs for K^c and L^c. Then we construct an SVFA for $K^c \cap L^c$. Finally, we take an SVFA for the complement of the resulting language. As witness languages, we can take the complements of the witnesses for intersection. Similar considerations can be done also for difference since $K \setminus L = K \cap L^c$.

Finally, we consider symmetric difference. To get the upper bound, we construct a product automaton for symmetric difference in a similar way as for intersection. However, now the sets of accepting and rejecting states are

$F^a = \{(p,q) \mid p \in F_A^a \text{ and } q \in F_B^r\} \cup \{(p,q) \mid p \in F_A^r \text{ and } q \in F_B^a\}$;

$F^r = \{(p,q) \mid p \in F_A^a \text{ and } q \in F_B^a\} \cup \{(p,q) \mid p \in F_A^r \text{ and } q \in F_B^r\}$.

This is an mn-state SVFA for the symmetric difference of given languages.

For tightness, let K and L be languages accepted by DFAs A and B shown in Fig. 5. Construct a product automaton for $K \oplus L$ as described above, and notice that the set $\mathcal{F} = \{(a^i b^j, a^{m-1-i} b^{n-1-j}) \mid 0 \leq i \leq m-1 \text{ and } 0 \leq j \leq n-1\}$ is an sv-fooling set for the language $K \oplus L$. □

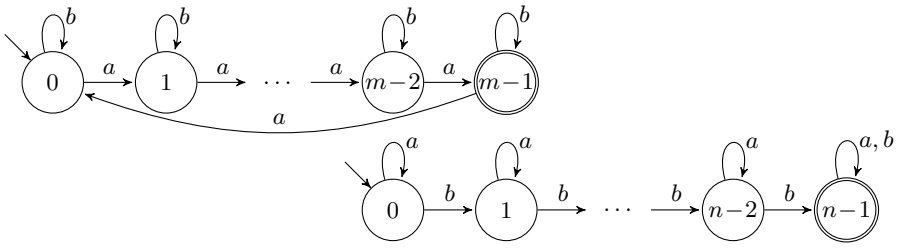

Fig. 5. The binary witnesses for symmetric difference meeting the bound mn.

3.3 Star

The state complexity of the star operation is $3/4 \cdot 2^n$ with binary witness languages [10, 15, 23]. In the unary case, the tight bound on the state complexity of star is $(n-1)^2+1$ [23, 24]. The nondeterministic state complexity of star is $n+1$, with witnesses defined over a unary alphabet [7]. In this section we show that the self-verifying state complexity of star is $3/4 \cdot 2^n$. Our worst-case examples are defined over an alphabet which grows exponentially with n. However, for a four-letter alphabet, we still get an exponential lower bound $2^{n-1}-1$.

Theorem 13 ([9]). *Let L be a language over Σ with* $\mathrm{svsc}(L) = n$. *Then*
(i) $\mathrm{svsc}(L^*) \leq 3/4 \cdot 2^n$, *and the bound is tight if* $|\Sigma| \geq 3/4 \cdot 2^n + 1$;
(ii) the bound $2^{n-1}-1$ *can be met by a quaternary language.*

Proof. (i) Let $A = (Q, \Sigma, \delta, s, F^a, F^r)$ be an SVFA for L. If only the initial state s is accepting, then $L^* = L$. Assume that A has k accepting states that are different from s. Construct an NFA A^* for L^* from A as follows. First, add a new initial and final state q_0 and for each symbol a in Σ, add a transition from q_0 to $\delta(s,a)$ if $\delta(s,a) \cap F^a = \emptyset$, and to $\{s\} \cup \delta(s,a)$ otherwise. Next, for each state q in Q and each symbol a, add a transition from q to s on a whenever $\delta(q,a) \cap F^a \neq \emptyset$. The initial state of A^* is q_0, and the set of final states is $\{q_0\} \cup F^a$. Now consider the subset automaton of A^*. Notice that no set containing a state in F^a and not containing s is reachable in the subset automaton. Next, we can show that the empty set is unreachable. Hence the subset automaton has at most $2^{n-1}+2^{n-1-k}$ reachable subsets. The maximum is attained if $k = 1$, and it is equal to $3/4 \cdot 2^n$.

To prove tightness, consider the following family of $3/4 \cdot 2^n - 1$ subsets: $\mathcal{R} = \big\{S \mid S \subseteq \{0,1,\ldots,n-1\} \text{ and } 0 \in S\big\} \cup \big\{S \mid \emptyset \neq S \subseteq \{1,2,\ldots,n-2\}\big\}$. Let $\Sigma = \{a,b\} \cup \{c_S \mid S \in \mathcal{R}\}$ be an alphabet consisting of $3/4 \cdot 2^n + 1$ symbols. Let L be accepted by an n-state DFA $A = (\{0,1,\ldots,n-1\}, \Sigma, \delta, 0, \{n-1\})$, where the transitions are defined as follows: $\delta(i,a) = (i+1) \bmod n$; $\delta(0,b) = 0$, $\delta(i,b) = i+1$ if $1 \leq i \leq n-2$, and $\delta(n-1,b) = n-1$; and for each set S in $\mathcal{R}$,

$$\delta(i, c_S) = \begin{cases} 0, & \text{if } i \in S, \\ n-1, & \text{if } i \notin S. \end{cases}$$

The transitions on a and b in A are shown in Fig. 6 (top-left), and the transitions on the symbol $c_{\{1,3\}}$ in the case of $n = 5$ are shown in Fig. 6 (bottom-right).

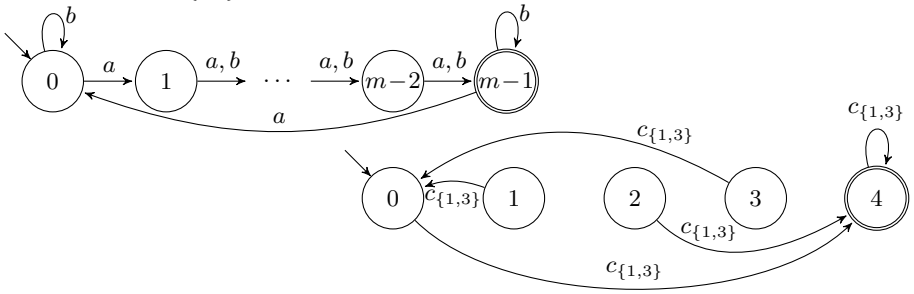

Fig. 6. The witness for star; symbols a a b (top-left) and symbol $c_{\{1,3\}}$ for $n = 5$.

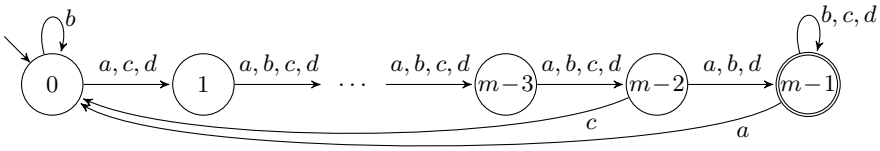

Fig. 7. The quaternary DFA of a language L with $\mathrm{svsc}(L^*) \geq 2^{n-1} - 1$.

Construct an NFA A^* for the language L^* as described above. Notice that each subset in $\mathcal{R}$ is reachable in the subset automaton of A^*, that is, for each subset S in $\mathcal{R}$, there is a string u_S, by which $\{q_0\}$ goes to the subset S. Then the set $\mathcal{F} = \{(u_S, c_S) \mid S \in \mathcal{R}\}$ is an sv-fooling set of size $3/4 \cdot 2^n - 1$ for L^*. Finally, by setting $w_S = \varepsilon$ if $n-1 \notin S$ and $w_S = b$ otherwise, we use Lemma 9 to show that one more state is necessary in every SVFA for the language L^*.

(ii) Consider the language L accepted by the quaternary DFA B shown in Fig. 7. Notice that the transitions on symbols a and b are the same as in the DFA A above. It follows that all the subsets of $\{0, 1, \ldots, n-1\}$, that have been shown to be reachable in the subset automaton of A^*, are reachable in the subset automaton of B^* as well. In particular, all the non-empty subsets of $\{0, 1, \ldots, n-2\}$ are reachable. Similarly as in the proof above, let u_S be a string over $\{a, b\}$ by which the initial subset $\{q_0\}$ goes to S in the subset automaton. Our aim is to describe an sv-fooling set for L^* of size $2^{n-1} - 1$. To this aim, for every non-empty subset S of $\{0, 1, \ldots, n-2\}$, define the string $v_S = v_0 v_1 \cdots v_{n-2}$ of length $n-1$ over $\{c, d\}$ as follows:

$$v_{n-2-i} = \begin{cases} c, & \text{if } i \in S, \\ d, & \text{if } i \notin S, \end{cases}$$

that is, the string v_S somehow describes the set S, however, in a reversed order: we can assign the symbol $\sigma(i) = c$ to each state i in S and the symbol $\sigma(i) = d$ to each state i outside the set S, and then we have $v_S = \sigma(n-2)\sigma(n-3)\cdots\sigma(1)\sigma(0)$. Then, for every set S, the string v_S is accepted by B^* from every state outside the set S, while v_S is rejected by B^* from every state in S. It follows that $\{(u_S, v_S) \mid \emptyset \neq S \subseteq \{0, 1, \ldots, n-2\}\}$ is an sv-fooling set for L^*. □

3.4 Left and Right Quotients

The left quotient of a language L by a string w is $w\backslash L = \{x \mid wx \in L\}$, and the left quotient of a language L by a language K is the language $K\backslash L = \bigcup_{w \in K} w\backslash L$. The state complexity of the left quotient operation is $2^n - 1$ [23], and its nondeterministic state complexity is $n + 1$ [10]. In both cases, the worst-case examples are defined over a binary alphabet.

The right quotient of a language L by a string w is $L/w = \{x \mid xw \in L\}$, and the right quotient of a language L by a language K is $L/K = \bigcup_{w \in K} L/w$. If a language L is accepted by an n-state DFA $A = (Q, \Sigma, \cdot, s, F)$, then the language L/K is accepted by a DFA that is exactly the same as the DFA A, except for the set of final states that consists of all the states q of A, such that there exists a string w in K with $q \cdot w \in F$ [23]. Thus $\mathrm{sc}(L/K) \leq n$. The tightness of this upper bound has been shown using binary languages in [23].

Here we show that the self-verifying complexity of the left quotient operation is 2^n-1. To prove tightness, we use an exponential alphabet. Then, using a four letter alphabet, we get a lower bound $2^{n-1}-1$. Finally, we show that the self-verifying state complexity of right quotient is given by the function $g(n)$, where $g(n)$ is the tight upper bound for SVFA-to-DFA conversion given in (1) on page 3.

Theorem 14 ([9]). *Let $K, L \subseteq \Sigma$, $\mathrm{svsc}(K) = m$, and $\mathrm{svsc}(L) = n$. Then*
(i) $\mathrm{svsc}(K\backslash L) \leq 2^n - 1$, and the bound is tight if $|\Sigma| \geq 2^n + 1$;
(ii) the bound $2^{n-1}-1$ can be met by quaternary languages.

Proof. (i) Let L be accepted by an SVFA $A = (Q, \Sigma, \delta, s, F^a, F^b)$. Then the language $K\backslash L$ is accepted by an NNFA $N = (Q, \Sigma, \delta, I, F^a)$, where a state q is in I if it can be reached from the initial state of A by a string in K. After applying the subset construction to the NNFA N, we get a DFA for $K\backslash L$, in which the empty set is unreachable. This gives the upper bound.

To prove tightness, consider the family $\mathcal{R}$ of all non-empty subsets of $\{0, 1, \ldots, n-1\}$. Let $\Sigma = \{a, b\} \cup \{c_S \mid S \in \mathcal{R}\}$ be an alphabet consisting of 2^n+1 symbols. Let $K = a^* \cup a^* b^{m-2}$ be a language over Σ. Then K is accepted by an m-state DFA, and the set $\{(b^i, b^{m-2-i}) \mid 0 \leq i \leq m-2\} \cup \{(b^{m-1}a, \varepsilon)\}$ is an sv-fooling set of size m for the language K. Hence $\mathrm{svsc}(K) = m$.

Let L be accepted by an n-state DFA $B = (\{0, 1, \ldots, n-1\}, \Sigma, \delta, 0, \{n-1\})$, where the transitions are defined as follows: $\delta(i, a) = (i+1) \bmod n$; $\delta(0, b) = \delta(1, b) = 0$, and $\delta(i, b) = i$ if $2 \leq i \leq n-1$; and for each subset S of $\{0, \ldots, n-1\}$, we have $\delta(i, c_S) = 0$ if $i \in S$, and $\delta(i, c_S) = n-1$ otherwise.

Construct an NNFA N for the language $K\backslash L$ from the DFA B by making all the states of B initial. Each subset S in $\mathcal{R}$ is reachable in the subset automaton of the NNFA N by a string u_S. Now, in the same way as in the proof of Theorem 13, we can prove that the set of pairs $\{(u_S, c_S) \mid S \in \mathcal{R}\}$ is an sv-fooling set of size $2^n - 1$ for the language $K\backslash L$.

(ii) The language K over $\{a, b, c, d\}$ is the same as in (i). The language L is accepted by the DFA B', in which the transitions on a and b are the same as in the DFA B above, and the transitions on c and d are the same as in Fig. 7. In a similar way as in the proof of Theorem 13 (ii), we can describe an sv-fooling set $\big\{(u_S, v_S) \mid \emptyset \neq S \subseteq \{0, 1, \ldots, n-2\}\big\}$ of size $2^{n-1}-1$ for $K\backslash L$. □

Theorem 15 ([9]). *Let $K, L \subseteq \Sigma$, $\mathrm{svsc}(K) = m$, and $\mathrm{svsc}(L) = n$. Then*
(i) $\mathrm{svsc}(L/K) \leq g(n)$, and the bound is tight if $|\Sigma| \geq g(n) + 2$;
(ii) the bound $\Omega(2^{n/3})$ can be met by quaternary languages. □

Proof. (i) Let a language L be accepted by an n-state SVFA. First, convert this SVFA to an equivalent minimal DFA. By Theorem 4, this DFA has at most $g(n)$ states. By making certain states final based on the language K, we get a DFA for L/K of at most $g(n)$ states.

For tightness, let $n = 1 + 3k$ and $k \geqslant 2$; the arguments can be extended to the other values of n in a straightforward way. Consider the grid $Q = \{(i, j) \mid 0 \leq i \leq 2 \text{ and } 1 \leq j \leq k\}$ of $3k$ nodes. Let $\mathcal{R}$ be the following family of 3^k subsets $\mathcal{R} = \big\{\{(i_1, 1), (i_2, 2), \ldots, (i_k, k)\} \mid i_1, i_2, \ldots, i_k \in \{0, 1, 2\}\big\}$, that is, each

subset in $\mathcal{R}$ corresponds to a choice of one element in each column of the grid Q. Let $\Sigma = \{a, b, c\} \cup \{d_S \mid S \in \mathcal{R}\}$ be an alphabet consisting of $3 + 3^k$ symbols.

Let $K = \{c^\ell \mid \ell \geq m-2\}$ be the language over Σ that contains all the strings in c^* of length at least $m-2$. We have $\mathrm{svsc}(K) = m$. Let L be accepted by a $(3k+1)$-state SVFA B, in which the transitions on a, b are the same as in the binary witness for SVFA-to-DFA conversion in Theorem 4. Next, symbol c performs the cyclic permutation on each row of the grid Q, and maps the initial state to each state in the first row. Finally, for each set S in $\mathcal{R}$, symbol d_S maps every state (i, j) of S to the state $(1, j)$, and it maps every state (i, j) outside S to $(0, j)$. Then we can show that $\mathrm{svsc}(L/K) = g(n)$.

(ii) Let $\Sigma = \{a, b, c, d\}$. Let $K = \{c^\ell \mid \ell \geq m-2\}$ be a language over Σ with $\mathrm{svsc}(K) = m$. Let L be accepted by an n-state SVFA B' in which the transitions on a, b are the same as in the SVFA B in case (i). By c and d, the state q_0 goes to $\{(0,1), \ldots, (0,k)\}$, and each state (i, j) with $j \leq k-1$ goes to $\{(i, j+1)\}$. The state $(0, k)$ goes to $\{(1,1)\}$ on both c, d. The state $(1, k)$ goes to $\{(0,1)\}$ on c, and it goes to $\{(2,1)\}$ on d. The state $(2, k)$ goes to $\{(2,1)\}$ on both c, d. Then we get $\mathrm{svsc}(L/K) \in \Omega(2^{n/3})$. □

3.5 Concatenation

The state complexity of concatenation is $m2^n - 2^{n-1}$, and its nondeterministic state complexity is $m + n$. In both cases, the worst-case examples can be defined over a binary alphabet [7, 10, 15, 23]. The aim of this subsection is to get asymptotically tight bound $\Theta(3^{m/3} \cdot 2^n)$ on the self-verifying state complexity of the concatenation operation. Recall that $g(n)$ is the tight upper bound for SVFA-to-DFA conversion given in (1) on page 3.

Theorem 16 ([9]). *Let $K, L \subseteq \Sigma$, $\mathrm{svsc}(K) = m$, and $\mathrm{svsc}(L) = n$. Then*
(i) $\mathrm{svsc}(KL) \leq g(m) \cdot 2^n$;
(ii) the bound $1/2 \cdot g(m) \cdot 2^n$ can be met if $|\Sigma| \geq g(m) + 2^n + 4$;
(iii) the bound $\Omega(2^{m/3} \cdot 2^n)$ can be met if $|\Sigma| \geq 8$.

Proof. (i) Let K and L be accepted by SVFAs A and B, respectively. First, convert the SVFA A to a minimal DFA A'. Then, construct an NNFA N for the language KL from automata A' and B in a usual way. Next, apply the subset construction to N. In the subset automaton of N, every reachable subset can be expressed as $\{q\} \cup T$, where q is a state of A' and T is a subset of the state set of B. Since A is an SVFA, the DFA A' has at most $g(m)$ states by Theorem 4. Thus the subset automaton of N has at most $g(m) \cdot 2^n$ reachable states.

(ii) For the sake of simplicity, we consider the case of $m = 1 + 3k$ a $k \geq 2$. Consider the grid $Q = \{(i, j) \mid 0 \leq i \leq 2 \text{ and } 1 \leq j \leq k\}$ of $3k$ nodes. Let $\mathcal{R} = \big\{\{(i_1, 1), (i_2, 2), \ldots, (i_k, k)\} \mid i_1, i_2, \ldots, i_k \in \{0, 1, 2\}\big\}$. Let $\Sigma = \{a, b, c, d, e\} \cup \{f_S \mid S \in \mathcal{R}\} \cup \{g_T \mid T \subseteq \{0, 1, \ldots, n-1\}\}$ be an alphabet consisting of $5 + 3^{\frac{m-1}{3}} + 2^n$ symbols. Let K be the language over Σ accepted by m-state SVFA $A = (Q \cup \{q_0\}, \Sigma, \delta, q_0, F^a, F^r)$, where the transitions on a, b, c are the same as in the case of right quotient, the symbols d, e, g_T are ignored, and transitions on f_S are defined by

$$\delta((i,j),f_S)=\begin{cases}\{(1,j)\}, & \text{if } (i,j)\in S,\\ \{(0,j)\}, & \text{if } (i,j)\notin S;\end{cases}$$

Let L be the language accepted by DFA $B=(\{0,1,\ldots,n-1\},\Sigma,\cdot,0,\{0\})$, where $i\cdot a=i\cdot b=i\cdot c=i\cdot f_S=i$; $i\cdot d=(i+1) \bmod n$; $0\cdot e=0$, $i\cdot b=i+1$ if $1\le i\le n-2$, and $(n-1)\cdot b=1$;

$$i\cdot g_T=\begin{cases}n-1, & \text{if } i\in T,\\ 0, & \text{if } i\notin T.\end{cases}$$

Construct an NFA N for KL and show that each set in the family $\mathcal{R}_N=\big\{S\cup T \mid S\in\mathcal{R} \text{ with } (0,k)\notin S, \text{ and } T\subseteq\{0,1,\ldots,n-1\}\big\}$ is reachable in the corresponding subset automaton. Then prove that $\mathcal{F}=\{(u_{S\cup T}, g_T\cdot f_S\cdot c^k) \mid S\cup T\in\mathcal{R}_N\}$ is an sv-fooling set for the language KL.

(iii) The idea of the proof is to define strings v_S and v_T over an eight-letter alphabet for some sets S in $\mathcal{R}$, namely, for those that consist only of the states in the first and second row of the grid Q, and for each subset T of $\{1,\ldots,n-2\}$. As a result, we get an sv-fooling set $\{(u_{S\cup T}, v_T\cdot v_S\cdot c^{2k}) \mid S\in\mathcal{R}', T\subseteq\{1,\ldots,n-2\}\}$, where $\mathcal{R}'$ contains all the sets in $\mathcal{R}$ which only have states in the the first or second row of the grid Q. This gives a lower bound in $\Omega(2^{m/3}\cdot 2^n)$. □

4 Conclusions

Table 1 summarizes the results on the self-verifying state complexity of considered operations, and compares them to the known results on their state complexity and nondeterministic state complexity. The last column of the table displays the size of an alphabet which was used to define witness languages. For star and quotients, an exponential lower bound can be obtained by using a four-letter alphabet. In the case of concatenation, a lower bound in $\Omega(2^{m/3}2^n)$ is met by languages defined over an eight-letter alphabet. The tight upper bound for the concatenation operation remains open even in the case of a growing alphabet.

	DFAs	NFAs	SVFAs	$\lvert\Sigma\rvert$
complement	n	2^n	n	1
intersection	mn	mn	mn	2
union	mn	$m+n+1$	mn	2
difference	mn	?	mn	2
symmetric difference	mn	?	mn	2
reversal	2^n	$n+1$	$2n+1$	2
star	$3/4\cdot 2^n$	$n+1$	$3/4\cdot 2^n$	$3/4\cdot 2^n+1$
left quotient	2^n-1	$n+1$	2^n-1	2^n+1
right quotient	n	n	$g(n)$	$g(n)+2$
concatenation	$(m-\frac{1}{2})\cdot 2^n$	$m+n$	$\Theta(3^{m/3}\cdot 2^n)$	$g(m)+2^n+4$

Table 1. The state complexity, nondeterministic, and self-verifying state complexity of basic regular operations.

References

1. Assent, I., Seibert, S.: An upper bound for transforming self-verifying automata into deterministic ones. Theor. Inform. Appl. 41, 261–265 (2007)
2. Birget, J. C.: Partial orders on words, minimal elements of regular languages, and state complexity. Theoret. Comput. Sci. 119, 267–291 (1993)
3. Brzozowski, J. A.: Quotient complexity of regular languages. J. Autom. Lang. Comb. 15, 71–89 (2010)
4. Ďuriš, P., Hromkovič, J., Rolim, J., Schnitger, G.: Las Vegas versus determinism for one-way communication complexity, finite automata, and polynomial-time computations. In: STACS. LNCS, vol. 1200, pp. 117–128. Springer (1997)
5. Ershov, U. L.: On a conjecture of W.U. Uspenskii. Algebra i logika (seminar) 1, 45–48 (1962) (in Russian)
6. Glaister, I., Shallit, J.: A lower bound technique for the size of nondeterministic finite automata. Inform. Process. Lett. 59, 75–77 (1996)
7. Holzer, M., Kutrib, M.: Nondeterministic descriptional complexity of regular languages. Internat. J. Found. Comput. Sci. 14, 1087–1102 (2003)
8. Hromkovič, J., Schnitger, G.: On the power of Las Vegas for one-way communication complexity, OBDDs, and finite automata. Inform. Comput. 169, 284–296 (2001)
9. Jirásek, J., Jr., Jirásková, G., Szabari, A.: Operations on self-verifying finite automata. In: CSR 2015. LNCS, vol. 9139, pp. 231–261. Springer (2015)
10. Jirásková, G.: State complexity of some operations on binary regular languages. Theoret. Comput. Sci. 330, 287–298 (2005)
11. Jirásková, G., Pighizzini, G.: Optimal simulation of self-verifying automata by deterministic automata. Inform. Comput. 209, 528–535 (2011)
12. Jirásková, G., Šebej, J.: Reversal of binary regular languages. Theoret. Comput. Sci. 449, 85–92 (2012)
13. Leiss, E.: Succinct representation of regular languages by Boolean automata. Theoret. Comput. Sci. 13, 323–330 (1981)
14. Lupanov, O. B.: A comparison of two types of finite automata. Problemy Kibernetiki 9, 321–326 (1963) (in Russian)
15. Maslov, A. N.: Estimates of the number of states of finite automata. Soviet Math. Doklady 11, 1373–1375 (1970)
16. Mirkin, B. G.: On dual automata. Kibernetika (Kiev) 2, 7–10 (1966) (in Russian).
17. Moon, J. W., Moser, L.: On cliques in graphs. Israel J. Math. 3, 23–28 (1965)
18. Moore, F.: On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. IEEE Trans. Comput. C-20, 1211–1214 (1971)
19. Rabin, M., Scott, D.: Finite automata and their decision problems. IBM J. Res. Develop. 3, 114–125 (1959)
20. Sakoda, W. J., Sipser, M.: Nondeterminism and the size of two-way finite automata. In: Proc. 10th Annual ACM STOC, pp. 275–286 (1978)
21. Sipser, M.: Introduction to the theory of computation. PWS Publishing Company, Boston (1997)
22. Yu, S.: Chapter 2: Regular languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages - Vol. I, pp. 41–110. Springer, Heidelberg (1997)
23. Yu, S., Zhuang, Q., Salomaa, K.: The state complexity of some basic operations on regular languages. Theoret. Comput. Sci. 125, 315–328 (1994)
24. Čevorová, K.: Kleene star on unary regular languages. In: Jürgensen, H., Reis, R. (eds.), DCFS 2013. LNCS, vol. 8031, pp. 277–288. Springer (2013)