State Complexity of Union and Intersection of Finite Languages

Yo-Sub Han[1,⋆] and Kai Salomaa[2,⋆⋆]

[1] Intelligence and Interaction Research Center, Korea Institute of Science and Technology, P.O.Box 131, Cheongryang, Seoul, Korea
emmous@kist.re.kr
[2] School of Computing, Queen's University, Kingston, Ontario K7L 3N6, Canada
ksalomaa@cs.queensu.ca

Abstract. We investigate the state complexity of union and intersection for finite languages. Note that the problem of obtaining the tight bounds for both operations was open. We compute the upper bounds based on the structural properties of minimal deterministic finite-state automata (DFAs) for finite languages. Then, we show that the upper bounds are tight if we have a variable sized alphabet that can depend on the size of input DFAs. In addition, we prove that the upper bounds are unreachable for any fixed sized alphabet.

1 Introduction

Regular languages are one of the most important and well-studied topics in computer science. They are often used in various practical applications such as `vi`, `emacs` and `Perl`. Furthermore, researchers developed a number of software libraries for manipulating formal language objects with the emphasis on regular languages; examples are Grail [12] and Vaucanson [2].

The applications and implementations of regular languages motivate the study of the descriptional complexity of regular languages. The descriptional complexity of regular languages can be defined in different ways since regular languages can be defined in different ways. For example, a regular language L is accepted by a deterministic finite-state automaton (DFA) or a nondeterministic finite-state automaton (NFA). L is also described by a regular expression. Yu and his co-authors [1,13,14] regarded the number of states in the minimal DFA for L as the complexity of L and studied the state complexity of basic operations on regular languages and finite languages. Holzer and Kutrib [5,6] investigated the state complexity of NFAs. Recently, Ellul et al. [3] examined the size of the shortest regular expression for a given regular language. There are many other results on state complexity with different viewpoints [4,8,9,10,11]. We focus on the measure of Yu [13]: The *state complexity* of a regular language L is the number of states of the minimal DFA for L. The state complexity of an operation

⋆ Han was supported by the KIST Tangible Space Initiative Grant.
⋆⋆ Salomaa was supported by the Natural Sciences and Engineering Research Council of Canada Grant OGP0147224.

T. Harju, J. Karhumäki, and A. Lepistö (Eds.): DLT 2007, LNCS 4588, pp. 217–228, 2007.

on regular languages is a function that associates to the state complexities of the operand languages the worst-case state complexity of the language resulting from the operation. For instance, we say that the state complexity of the intersection of $L(A)$ and $L(B)$ is mn, where A and B are minimal DFAs and the numbers of states in A and B are m and n, respectively. It means that mn is the worst-case number of states of the minimal DFA for $L(A) \cap L(B)$.

Yu et al. [14] gave the first formal study of state complexity of regular language operations. Later, Câmpeanu et al. [1] investigated the state complexity of finite languages. Let A and B be minimal DFAs for two regular languages L_1 and L_2, and m and n be the numbers of states for A and B, respectively.

operation	finite languages	regular languages
$L_1 \cup L_2$	$O(mn)$	mn
$L_1 \cap L_2$	$O(mn)$	mn
$\Sigma^* \setminus L_1$	m	m
$L_1 \cdot L_2$	$(m-n+3)2^{n-2}-1^{\diamond}$	$(2m-1)2^{n-1}$
L_1^*	$2^{m-3}+2^{m-4}$, for $m \geq 4^{\diamond}$	$2^{m-1}+2^{m-2}$
L_1^R	$3 \cdot 2^{p-1}-1$ if $m=2p$ 2^p-1 if $m=2p-1$	2^m

Fig. 1. State complexity of basic operations on finite languages and regular languages [1,14]. Note that $\diamond$ refers to results using a two-character alphabet.

Fig. 1 shows the state complexity of basic operations on finite languages and regular languages. All complexity bounds, except for union and intersection of finite languages, in Fig. 1 are tight; namely, there exist worst-case examples that reach the given bounds. Clearly, mn is an upper bound since finite languages are a proper subfamily of regular languages. We also note that Yu [13] briefly mentioned a rough upper bound $mn-(m+n-2)$ for both operations. Therefore, it is natural to investigate the tight bounds for union and intersection of finite languages.

We define some basic notions in Section 2. In Section 3, we obtain an upper bound $mn-(m+n)$ for the union of two finite languages L_1 and L_2 based on the structural properties, where the sizes of L_1 and L_2 are m and n. Then, we prove that the bound is tight if the alphabet size can depend on m and n. We also examine the intersection of L_1 and L_2 in Section 4 and obtain an upper bound $mn-3(m+n)+12$. We again demonstrate that the upper bound is reachable using a variable sized alphabet. We conclude the paper in Section 5.

2 Preliminaries

Let Σ denote a finite alphabet of characters and Σ^* denote the set of all strings over Σ. The size $|\Sigma|$ of Σ is the number of characters in Σ. A language over

Σ is any subset of Σ^*. The symbol $\emptyset$ denotes the empty language and the symbol λ denotes the null string. A finite-state automaton (FA) A is specified by a tuple $(Q, \Sigma, \delta, s, F)$, where Q is a finite set of states, Σ is an input alphabet, $\delta : Q \times \Sigma \to 2^Q$ is a transition function, $s \in Q$ is the start state and $F \subseteq Q$ is a set of final states. Given a DFA A, we assume that A is complete; namely, each state has $|\Sigma|$ out-transitions and, therefore, A may have a sink (or dead) state. Since all sink states are always equivalent, we can assume that A has a unique sink state. Let $|Q|$ be the number of states in Q and $|\delta|$ be the number of transitions in δ. For a transition $\delta(p, a) = q$ in A, we say that p has an *out-transition* and q has an *in-transition*. Furthermore, p is a *source state* of q and q is a *target state* of p. A string x over Σ is accepted by A if there is a labeled path from s to a final state in F such that this path spells out the string x. Thus, the language $L(A)$ of an FA A is the set of all strings that are spelled out by paths from s to a final state in F. We say that A is *non-returning* if the start state of A does not have any in-transitions and A is *non-exiting* if all out-transitions of any final state in A go to the sink state.

Given an FA $A = (Q, \Sigma, \delta, s, F)$, we define the *right language* L_q of a state q to be the set of strings that are spelled out by some path from q to a final state in A; namely, L_q is the language accepted by the FA obtained from A by changing the start state to q. We say that two states p and q are *equivalent* if $L_p = L_q$.

3 Union of Finite Languages

Given two minimal DFAs A and B for non-empty finite languages L_1 and L_2, we can in the standard way construct a DFA for the union of $L(A)$ and $L(B)$ based on the Cartesian product of states.

Proposition 1. *Given two DFAs $A = (Q_1, \Sigma, \delta_1, s_1, F_1)$ and $B = (Q_2, \Sigma, \delta_2, s_2, F_2)$, let $M_\cup = (Q_1 \times Q_2, \Sigma, \delta, (s_1, s_2), F)$, where for all $p \in Q_1$ and $q \in Q_2$ and $a \in \Sigma$, $\delta((p, q), a) = (\delta(p, a), \delta(q, a))$ and $F = \{(p, f_2) \mid p \in Q_1 \text{ and } f_2 \in F_2\} \cup \{(f_1, q) \mid f_1 \in F_1 \text{ and } q \in Q_2\}$. Then, $L(M_\cup) = L(A) \cup L(B)$ and $M_\cup$ is deterministic.*

A crucial observation is that both A and B must be non-returning since L_1 and L_2 are finite. Therefore, as Yu [13] observed, if we apply the Cartesian product for union, all states (s_1, q) for $q \neq s_2$ and all states (p, s_2) for $p \neq s_1$ are not reachable from the start state (s_1, s_2) in $M_\cup$. Thus, we can reduce $(m + n) - 2$ states.

Another observation is that A must have a final state f such that all of f's out-transitions go to the sink state. Consider the right language of a state (i, j) in $M_\cup$.

Proposition 2 (Han et al. [4]). *For a state (i, j) in $M_\cup$, the right language $L_{(i,j)}$ of (i, j) is the union of L_i in A and L_j in B.*

Let d_1 and d_2 be the sink states of A and B and f_1 and f_2 be final states of A and B such that d_1 is the only target state of f_1 in A and d_2 is the only target state

of f_2 in B, respectively. Then, by Proposition 2, (f_1, f_2), (d_1, f_2) and (f_1, d_2) are equivalent and, thus, can be merged into a single state. It shows that we can reduce two more states from $M_\cup$. Therefore, we obtain the following result.

Lemma 1. *Given two minimal DFAs A and B for finite languages, $mn-(m+n)$ states are sufficient for the union of $L(A)$ and $L(B)$, where $m = |A|$ and $n = |B|$.*

We next examine whether or not we can reach the upper bound of Lemma 1.

Lemma 2. *The upper bound $mn - (m + n)$ for union cannot be reached with a fixed alphabet when m and n are arbitrarily large.*

Lemma 2 shows that we cannot reach the upper bound in Lemma 1 if $|\Sigma|$ is relatively small compared with the number states of the given DFAs. Then, the next question is what if $|\Sigma|$ is large enough?

Lemma 3. *The upper bound $mn - (m + n)$ for union is reachable if the size of the alphabet can depend on m and n.*

Proof. Let m and n be positive numbers (namely, $m, n \in \mathbb{N}$) and

$$\Sigma = \{b, c\} \cup \{a_{i,j} \mid 1 \le i \le m-2, 1 \le j \le n-2 \text{ and } (i,j) \ne (m-2, n-2)\}.$$

Let $A = (Q_1, \Sigma, \delta_1, p_0, \{p_{m-2}\})$, where $Q_1 = \{p_0, p_1, \ldots, p_{m-1}\}$ and δ_1 is defined as follows:

- $\delta_1(p_i, b) = p_{i+1}$, for $0 \le i \le m-2$.
- $\delta_1(p_0, a_{i,j}) = p_i$, for $1 \le i \le m-2$ and $1 \le j \le n-2$, $(i,j) \ne (m-2, n-2)$.

For all other cases in δ_1 that are not covered above, the target state is the sink state p_{m-1}.

Next, let $B = (Q_2, \Sigma, \delta_2, q_0, \{q_{n-2}\})$, where $Q_2 = \{q_0, q_1, \ldots, q_{n-1}\}$ and δ_2 is defined as follows:

- $\delta_2(q_i, c) = q_{i+1}$, for $0 \le i \le n-2$.
- $\delta_2(q_0, a_{i,j}) = q_j$, for $1 \le j \le n-2$ and $1 \le i \le m-2$, $(i,j) \ne (m-2, n-2)$.

Again, for all other cases in δ_2 that are not covered above, the target state is the sink state q_{n-1}. Fig. 2 gives an example of such DFAs A and B.

Let $L = L(A_1) \cup L(A_2)$. We claim that the minimal (complete) DFA for L needs $mn - (m + n)$ states. To prove the claim, it is sufficient to show that there exists a set R consisting of $mn - (m + n)$ strings over Σ that are pairwise inequivalent modulo the right invariant congruence of L, $\equiv_L$.

We show that $R = R_1 \cup R_2 \cup R_3$, where

$R_1 = \{b^i \mid 0 \le i \le m-1\}$.
$R_2 = \{c^j \mid 1 \le j \le n-3\}$. (Note that R_2 does not include strings c^0, c^{n-2} and c^{n-1}.)
$R_3 = \{a_{i,j} \mid 1 \le i \le m-2 \text{ and } 1 \le j \le n-2 \text{ and } (i,j) \ne (m-2, n-2)\}$.

Any string b^i from R_1 cannot be equivalent with a string c^j from R_2 since $c^j \cdot c^{n-2-j} \in L$ but $b^i \cdot c^{n-2-j} \notin L$. Note that $j \ge 1$ and hence also $b^0 \cdot c^{n-2-j} \notin L$.

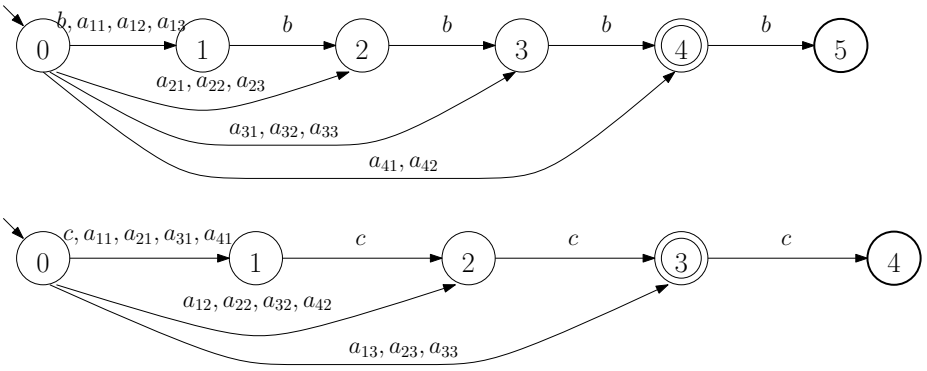

Fig. 2. An example of two minimal DFAs for finite languages whose sizes are 6 and 5, respectively, where state 5 above and state 4 below are sink states. Except for the b-transition to state 5 in A and the c-transition to state 4 in B, we omit all other in-transitions of the sink state.

Next consider a string b^i from R_1 and a string $a_{k,j}$ from R_3. There are four cases to consider.

1. $k \neq i$ and $0 \leq i \leq m-3$: It means that b^i and $a_{k,j}$ are inequivalent since $b^i \cdot b^{m-2-i} \in L$ but $a_{k,j} \cdot b^{m-2-i} \notin L$.
2. $k \neq i$ and $i = m-2$: It implies that $k < m-2$ and, thus, b^i and $a_{k,j}$ are inequivalent since $a_{k,j} \cdot b^{m-2-k} \in L$ but $b^i \cdot b^{m-2-k} \notin L$.
3. $k \neq i$ and $i = m-1$: The path for $b^i = b^{m-1}$ must end at the sink state for the minimal DFA for L since $b^{m-1} \notin L$. On the other hand, $a_{k,j}$ can be completed to be a string of L by appending zero or more symbols c.
4. $k = i$: Now we have strings b^i and $a_{i,j}$.
   (a) $j < n-2$: We note that $a_{i,j} \cdot c^{n-2-j} \in L$ but $b^i c^{n-2-j} \notin L$ since no string of L can have both b's and c's. Note that $k = i$ implies that $i \geq 1$.
   (b) $j = n-2$: Since $j = n-2$, $i < m-2$ by the definition of R_3. Now $b^i \cdot \lambda \notin L$ but $a_{i,j} = a_{i,n-2} \cdot \lambda \in L(B) \subseteq L$.
   Therefore, b^i and $a_{i,j}$ are inequivalent.

Symmetrically, we see that any string from R_2 cannot be equivalent with a string from R_3. This case is, in fact, simpler than the previous case since R_2 is more restrictive than R_1.

Finally we need to show that all strings from R_1 (respectively, from R_2 and from R_3) are pairwise inequivalent with each other.

1. R_1: By appending a suitable number of b's, we can always distinguish two distinct strings from R_1.
2. R_2: By appending a suitable number of c's, we can always distinguish two distinct strings from R_2.
3. R_3: Consider two distinct strings $a_{i,j}$ and $a_{x,y}$ from R_3. Without loss of generality, we assume that $i < x$. The other possibility, where j and y differ,

is completely symmetric. Since $a_{i,j} \cdot b^{m-2-i} \in L$ and $a_{x,y} \cdot b^{m-2-i} \notin L$, $a_{i,j}$ and $a_{x,y}$ are inequivalent. Note that $m-2-i > 0$ and, thus, the inequivalence holds even in the case when $y = n-2$.

This concludes the proof. □

In the construction of $R = R_1 \cup R_2 \cup R_3$ for Lemma 3, the size of Σ that we use is $mn - 2m - 2n + 5$. By using a more complicated construction, we might be able to reduce the size of Σ. On the other hand, we already know from Lemma 2 that $|\Sigma|$ has to depend on m and n.

We establish the following statement from Lemmas 1 and 3.

Theorem 1. *Given two minimal DFAs A and B for finite languages, $mn-(m+n)$ states are necessary and sufficient in the worst-case for the minimal DFA of $L(A) \cup L(B)$, where $m = |A|$ and $n = |B|$.*

Lemma 2 shows that the upper bound in Lemma 1 is unreachable if $|\Sigma|$ is fixed and m and n are arbitrarily large whereas Lemma 3 shows that the upper bound is reachable if $|\Sigma|$ depends on m and n. These results naturally lead us to examine the state complexity of union with a fixed sized alphabet. For easiness of presentation, we first give the result for a four character alphabet and afterward explain how the construction can be modified for a binary alphabet.

Lemma 4. *Let Σ be an alphabet with four characters. There exists a constant c such that the following holds for infinitely many $m, n \geq 1$, where $\min\{m,n\}$ is unbounded. There exist DFAs A and B, with m and n states respectively, that recognize finite languages over Σ such that the minimal DFA for the union $L(A) \cup L(B)$ requires $c(\min\{m,n\})^2$ states.*

The same result holds for a binary alphabet.

Proof. Let $\Sigma = \{a, b, c, d\}$. We introduce some new notations for the proof. Given an even length string $w \in \Sigma^*$, $\mathsf{odd}(w)$ denotes the subsequence of characters that occur in odd positions in w and, thus, the length of $\mathsf{odd}(w)$ is half the length of w. For example, if $w = adacbcbc$, then $\mathsf{odd}(w) = aabb$. Similarly, $\mathsf{even}(w)$ denotes the subsequence of characters that occur in even positions in w. With the same example above, $\mathsf{even}(w) = dccc$.

Let $s \geq 1$ be arbitrary and $r = \lceil \log s \rceil$. We define the finite language

$$L_1 = \{w_1 w_2 \mid |w_1| = 2r, w_2 = \mathsf{odd}(w_1) \in \{a,b\}^*, \mathsf{even}(w_1) \in \{c,d\}^*\}.$$

The language L_1 can be recognized by a DFA A with at most $10s$ states. For reading a prefix of length $2r$ of an input string, the start state of A has two out-transitions with labels a and b and the two corresponding target states are different. Then, each target state has two out-transitions with labels c and d where the target states are the same. This repeats in A until we finish reading the prefix of length $2r$. All other transitions go to the sink state. Fig. 3 illustrates the construction of A with $r = 3$.

The computations of A, which do not go to the sink state, on inputs of length $2r$ form a tree-like structure that branches into 2^r different states. Each of the 2^r states represents a unique string $\mathsf{odd}(u) \in \{a,b\}^*$, where u is the (prefix

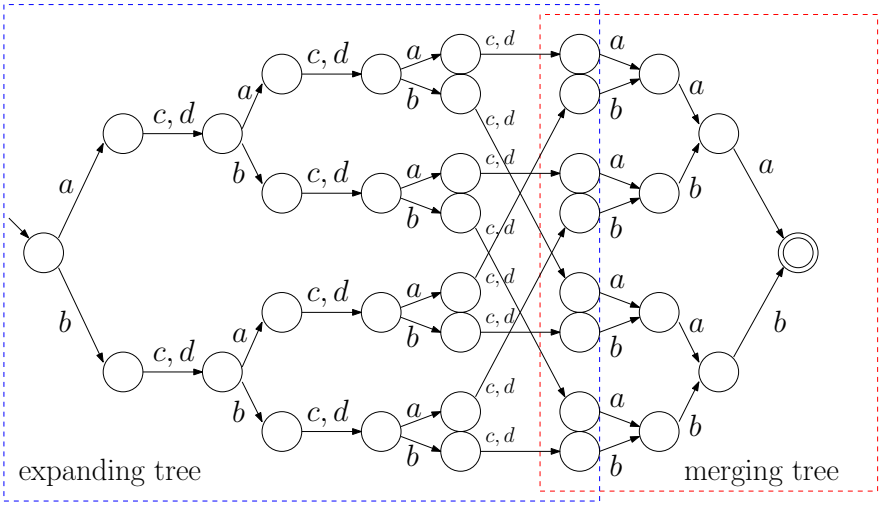

Fig. 3. A DFA A that recognizes L_1 when $r = 3$. We omit the sink state and its in-transitions.

of the) input of length $2r$. Then, the computation from each of these 2^r states verifies whether or not the remaining suffix is identical to the string $\mathsf{odd}(u)$. This can be accomplished using a tree that merges all the computations into a single final state. (See the right part of Fig. 3 for an example.) From each state, there is only one out-transition (either with symbol a or b), if we ignore transitions into the sink state. (The structure looks like a tree when we ignore transitions into the sink state.)

The computations of A on strings of length $2r$ branch into 2^r states. The first "expanding" tree (for instance, the left part of Fig. 3) uses less than $4 \cdot 2^r < 8s$ states[1] since we repeat each level with the c, d transitions in the tree and $s \leq 2^r < 2s$.

Finally, consider the number of states in the "merging" tree. (For example, we rotate the right part of Fig. 3.) Similarly, the merging tree has 2^r states and, therefore, the tree needs at most $2 \cdot 2^r < 4s$ states. However, we observe that the the last 2^r states of the expanding tree is the same state to the last 2^r states of the merging tree in A. Therefore, we only need $2s$ states for the merging tree.

$$\text{The total number of states in } A \text{ is less than } 10s. \tag{1}$$

Symmetrically, we define

$$L_2 = \{w_1 w_2 \mid |w_1| = 2r, \mathsf{odd}(w_1) \in \{a,b\}^*, w_2 = \mathsf{even}(w_1) \in \{c,d\}^*\}.$$

The language L_2 consists of strings uv, where $|u| = 2r$, odd characters of u are in $\{a,b\}$, even characters of u are in $\{c,d\}$ and $\mathsf{even}(u)$ coincides with v. Using an argument similar to that for equation (1), we establish that

[1] Note that a balanced tree with 2^r leaves has less than $2 \cdot 2^r$ nodes.

$$L_2 \text{ can be recognized by a DFA with less than } 10s \text{ states.} \tag{2}$$

Now let $L = L_1 \cup L_2$. Let u_1 and u_2 be distinct strings of length $2r$ such that $\mathsf{odd}(u_i) \in \{a, b\}^*$ and $\mathsf{even}(u_i) \in \{c, d\}^*$ for $i = 1, 2$.

If $\mathsf{odd}(u_1) \neq \mathsf{odd}(u_2)$, then $u_1 \cdot \mathsf{odd}(u_1) \in L_1 \subseteq L$ but $u_2 \cdot \mathsf{odd}(u_1) \notin L$. Hence, u_1 and u_2 are not equivalent modulo the right invariant congruence of L. Similarly, if $\mathsf{even}(u_1) \neq \mathsf{even}(u_2)$, then, $u_1 \cdot \mathsf{even}(u_1) \in L_2 \subseteq L$ but $u_2 \cdot \mathsf{even}(u_1) \notin L$.

The above implies that the right invariant congruence of L has at least $2^r \cdot 2^r \geq s^2$ different classes. Therefore, if $m = n = 10s$ is the size of the minimal DFAs for the finite languages L_1 and L_2, then from equations (1) and (2) we know that the minimal DFA for $L = L_1 \cup L_2$ needs at least

$$\frac{1}{100} n^2 \text{ states.} \tag{3}$$

Note that $|\Sigma| = 4$. The languages L_1 and L_2 can be straightforwardly encoded over a binary alphabet with the only change that the constant $\frac{1}{100}$ in equation (3) would become smaller. □

4 Intersection of Finite Languages

We examine the state complexity of intersection of finite languages. Our approach is again based on the structural properties of minimal DFAs of finite languages. We start from the Cartesian product of states for the intersection of two DFAs.

Proposition 3 (Hopcroft and Ullman [7]). *Given two DFAs* $A = (Q_1, \Sigma, \delta_1, s_1, F_1)$ *and* $B = (Q_2, \Sigma, \delta_2, s_2, F_2)$, *let* $M_\cap = (Q_1 \times Q_2, \Sigma, \delta, (s_1, s_2), F_1 \times F_2)$, *where for all* $p \in Q_1$ *and* $q \in Q_2$ *and* $a \in \Sigma$,

$$\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a)).$$

Then, $L(M_\cap) = L(A) \cap L(B)$.

Let $M_\cap$ denote the Cartesian product of states. Let m and n denote the sink states of A and B and $m-1$ and $n-1$ denote the final states whose target states are always the sink states of A and B, respectively. If we regard $M_\cap$ as a $m \times n$ matrix, then all states in the first row and in the first column are unreachable from $(1, 1)$ since A and B are non-returning and, thus, these states are useless in $M_\cap$. Moreover, by the construction, all remaining states in the last row and in the last column are equivalent to the sink state and, therefore, can be merged. Let us examine the remaining states in the second-to-last row and in the second-to-last column except for $(m-1, n-1)$.

Lemma 5. *A state* $(i, n-1)$ *in the second-to-last column, for* $1 \leq i \leq m-1$, *is either*

equivalent to $(m-1, n-1)$ *if state* i *is a final state in* A *or*
equivalent to (m, n) *if state* i *is not a final state in* A.

We can obtain a similar statement for the states in the second-to-last row in $M_\cap$. Therefore, all the remaining states at the second-to-last row and at the second-to-last column except for $(n-1, m-1)$ can be merged with either $(n-1, m-1)$ or (n, m). Thus, the number of remaining states is

$$mn - \{(m-1) + (n-1)\} - \{(m-2) + (n-2)\} - \{(m-3) + (n-3)\}$$

$$= mn - 3(m+n) + 12,$$

where $\{(m-1) + (n-1)\}$ is from the first row and the first column, $\{(m-2) + (n-2)\}$ is from the last row and the last column and $\{(m-3) + (n-3)\}$ is from the second-to-last row and the second-to-last column. We establish the following lemma from the calculation.

Lemma 6. *Given two minimal DFAs A and B for finite languages, $mn-3(m+n)+12$ states are sufficient for the intersection of $L(A)$ and $L(B)$, where $m = |A|$ and $n = |B|$.*

We now show that $mn - 3(m+n) + 12$ states are necessary and, therefore, the bound is tight. Let $m, n \in \mathbb{N}$ and choose $\Sigma = \{a_{i,j} \mid 1 \leq i \leq m-1 \text{ and } 1 \leq j \leq n-1\}$.

Let $A = (Q_1, \Sigma, \delta_1, p_0, \{p_{m-2}\})$, where $Q_1 = \{p_0, p_1, \ldots, p_{m-1}\}$ and δ_1 is defined as follows:

- $\delta_1(p_x, a_{i,j}) = p_{x+i}$, for $0 \leq x \leq m-2$, $1 \leq i \leq m-1$ and $1 \leq j \leq n-1$.

If the sum $x + i$ is larger than $m - 1$, then p_{x+i} is the sink state $(= p_{m-1})$. For all other cases in δ_1 that are not covered above, the target state is the sink state p_{m-1}.

Next, let $B = (Q_2, \Sigma, \delta_2, q_0, \{q_{m-2}\})$, where $Q_2 = \{q_0, q_1, \ldots, q_{n-1}\}$ and δ_2 is defined as follows:

- $\delta_2(q_x, a_{i,j}) = q_{x+j}$, for $0 \leq x \leq m-2$, $1 \leq j \leq n-1$ and $1 \leq i \leq m-1$.

Similarly, if the sum $x+j$ is larger than $n-1$, then q_{x+j} is the sink state $(= q_{m-1})$. For all other cases in δ_2 that are not covered above, the target state is the sink state q_{m-1}. Fig. 4 shows an example of such DFAs A and B.

Lemma 7. *Let $L = L(A) \cap L(B)$. The minimal (complete) DFA for L needs $mn - 3(m+n) + 12$ states.*

Proof. We prove the statement by showing that there exists a set R of $mn - 3(m+n) + 12$ strings over Σ that are pairwise inequivalent modulo the right invariant congruence of L, $\equiv_L$. We assume that $m \leq n$.

We choose $R = R_1 \cup R_2 \cup R_3 \cup R_4$, where

$R_1 = \{\lambda\}$.
$R_2 = \{a_{m-2,n-2}\}$.
$R_3 = \{a_{m-1,n-1}\}$.
$R_4 = \{a_{i,j} \mid \text{for } 1 \leq i \leq m-3 \text{ and } 1 \leq j \leq n-3\}$.

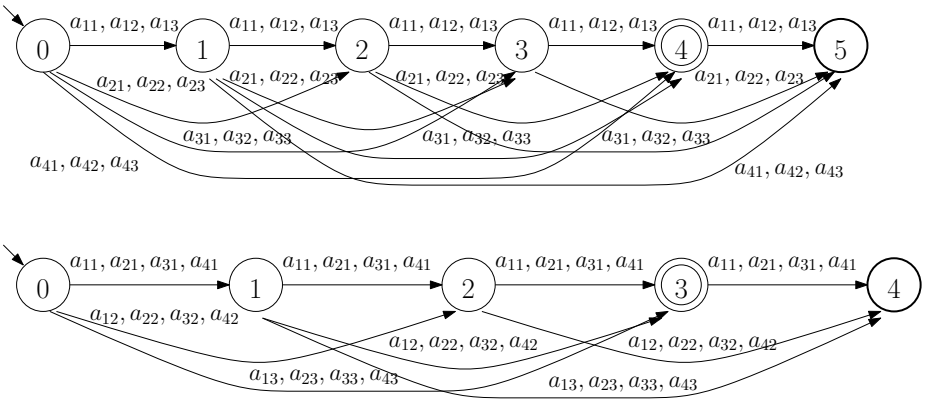

Fig. 4. An example of two minimal DFAs for finite languages whose sizes are 6 and 5, respectively, where state 5 above and state 4 below are sink states. We omit a large number of in-transitions of the sink state.

Any string x from $R_2 \cup R_3 \cup R_4$ cannot be equivalent with λ from R_1 since $\lambda \cdot a_{m-2,n-2} \in L$ but $x \cdot a_{m-2,n-2} \notin L$. Similarly, any string x from $R_1 \cup R_3 \cup R_4$ cannot be equivalent with $a_{m-2,n-2}$ from R_2 since $a_{m-2,n-2} \cdot \lambda \in L$ but $x \cdot \lambda \notin L$. Note that string $a_{m-1,n-1}$ from R_3 is not in L and it can never be in L by appending some string whereas any string x from $R_1 \cup R_2 \cup R_4$ can be in L by appending a suitable string. Therefore, R_1, R_2 and R_3 are inequivalent with each other including R_4.

Finally, we consider two strings $a_{i,j}$ and $a_{x,y}$ in R_4. The two strings are not equivalent since $a_{i,j} \cdot a_{m-2-i,n-2-j} \in L$ but $a_{x,y} \cdot a_{m-2-i,n-2-j} \notin L$ when $(i,j) \neq (x,y)$. Therefore, any two strings from R_4 are not equivalent.

Now we count the number of strings in R. We note that $|R_1| = |R_2| = |R_3| = 1$ and $|R_4| = (m-3)(n-3)$. Therefore, $|R| = mn - 3(m+n) + 12$. It implies that there are at least $mn - 3(m+n) + 12$ states in the minimal DFA for L. □

We obtain the following result from Lemmas 6 and 7.

Theorem 2. *Given two minimal DFAs A and B for finite languages, $mn - 3(m+n) + 12$ states are necessary and sufficient in the worst-case for the intersection of $L(A)$ and $L(B)$, where $m = |A|$ and $n = |B|$.*

Note that the upper bound $mn - 3(m+n) + 12$ is reachable when $|\Sigma|$ depends on m and n as shown in Lemma 7. On the other hand, we can prove that it is impossible to reach the upper bound with a fixed Σ using the same argument as in Lemma 2.

Let us investigate a lower bound for the state complexity of intersection of $L(A)$ and $L(B)$ over a fixed alphabet.

Lemma 8. *Let Σ be an alphabet with four characters. There exists a constant c such that the following holds for infinitely many $m, n \geq 1$, where* $\min\{m, n\}$

unbounded. There exist minimal DFAs A and B that recognize finite languages over Σ such that the minimal DFA for the intersection $L(A) \cap L(B)$ requires $c(\min\{m,n\})^2$ states, where $|A| = m$ and $|B| = n$.

The same result holds for a binary alphabet.

Proof. We omit the proof due to the space limit. The proof is similar to the proof for Lemma 4. □

5 Conclusions

The state complexity of an operation on regular languages is the number of states in the minimal DFA that recognizes the resulting language from the operation. Fig. 1 gives a summary of the results. We have noted that the precise state complexity of union and intersection cases have been open although rough upper bounds were given by Yu [13]. Based on the structural properties of minimal DFAs for finite languages, we have proved that

1. For union, $mn - (m + n)$ states are necessary and sufficient.
2. For intersection, $mn - 3(m + n) + 12$ states are necessary and sufficient.

We have also noted that the bounds are reachable if $|\Sigma|$ depends on m and n, where m and n are the sizes of minimal DFAs for two finite languages. If $|\Sigma|$ is fixed and m and n are arbitrarily large, then we have shown that the upper bounds for both cases are not reachable.

References

1. Câmpeanu, C., Culik II, K., Salomaa, K., Yu, S.: State complexity of basic operations on finite languages. In: Boldt, O., Jürgensen, H. (eds.) WIA 1999. LNCS, vol. 2214, pp. 60–70. Springer, Heidelberg (2001)
2. Claveirole, T., Lombardy, S., O'Connor, S., Pouchet, L.-N., Sakarovitch, J.: Inside Vaucanson. In: Farré, J., Litovsky, I., Schmitz, S. (eds.) CIAA 2005. LNCS, vol. 3845, pp. 116–128. Springer, Heidelberg (2006)
3. Ellul, K., Krawetz, B., Shallit, J., Wang, M.-W.: Regular expressions: New results and open problems. Journal of Automata, Languages and Combinatorics 9, 233–256 (2004)
4. Han, Y.-S., Salomaa, K., Wood, D.: State complexity of prefix-free regular languages. In: Proceedings of DCFS'06, pp. 165–176, Full version is submitted for publication (2006)
5. Holzer, M., Kutrib, M.: Unary language operations and their nondeterministic state complexity. In: Ito, M., Toyama, M. (eds.) DLT 2002. LNCS, vol. 2450, pp. 162–172. Springer, Heidelberg (2002)
6. Holzer, M., Kutrib, M.: Nondeterministic descriptional complexity of regular languages. International Journal of Foundations of Computer Science 14(6), 1087–1102 (2003)
7. Hopcroft, J., Ullman, J.: Introduction to Automata Theory, Languages, and Computation, 2nd edn. Addison-Wesley, Reading, MA (1979)

8. Hricko, M., Jirásková, G., Szabari, A.: Union and intersection of regular languages and descriptional complexity. In: Proceedings of DCFS'05, pp. 170–181 (2005)
9. Jirásek, J., Jiráskov, G., Szabari, A.: State complexity of concatenation and complementation of regular languages. In: Domaratzki, M., Okhotin, A., Salomaa, K., Yu, S. (eds.) CIAA 2004. LNCS, vol. 3317, pp. 178–189. Springer, Heidelberg (2005)
10. Nicaud, C.: Average state complexity of operations on unary automata. In: Kutyłowski, M., Wierzbicki, T., Pacholski, L. (eds.) MFCS 1999. LNCS, vol. 1672, pp. 231–240. Springer, Heidelberg (1999)
11. Pighizzini, G., Shallit, J.: Unary language operations, state complexity and Jacobsthal's function. International Journal of Foundations of Computer Science 13(1), 145–159 (2002)
12. Raymond, D.R., Wood, D.: Grail: A C++ library for automata and expressions. Journal of Symbolic Computation 17, 341–350 (1994)
13. Yu, S.: State complexity of regular languages. Journal of Automata, Languages and Combinatorics 6(2), 221–234 (2001)
14. Yu, S., Zhuang, Q., Salomaa, K.: The state complexities of some basic operations on regular languages. Theoretical Computer Science 125(2), 315–328 (1994)