State Complexity of Regular Languages*

Sheng Yu
Department of Computer Science, University of Western Ontario
London, Ontario, Canada N6A 5B7

January 16, 2000

Abstract

State complexity is a descriptive complexity measure for regular languages. We investigate the problems related to the state complexity of regular languages and their operations. In particular, we compare the state complexity results on regular languages with those on finite languages.

keywords Regular languages, finite languages, deterministic finite automata, state complexity.

1 Introduction

Regular languages and their implementations have been attracting more and more attention in recent years [18, 25] due to the increased applications of regular languages and finite automata in software engineering, programming languages, and other practical areas of computer science. Evidences of the increased applications include the popularity of the Regex ("regular expressions") as a programming-language construct in many recently-created programming languages such as Perl and Python, and the adoption of the statecharts as part of the object-oriented modeling and design methodologies such as OMT and UML [19, 2].

In recent years, quite a few software systems for manipulating formal language objects, with the emphasis on regular-language objects, have been developed. Examples include AMoRE, Automate, FIRE Engine, FSA, Grail, and INTEX [18, 25].

The applications as well as the implementations of regular languages require and also motivate the study of the complexity issues of regular languages. There are two kinds of complexity issues which are of interest: (1) time and spcace complexity issues and (2) descriptive complexity issues. In this article, the focus is on the *state complexity* issues. State complexity is a type of descriptive complexity for regular languages based on the deterministic finite automaton (DFA) model. The

*This research is supported by the Natural Sciences and Engineering Research Council of Canada grants OGP0041630.

state complexity of a regular-language operation also gives a lower-bound for the space as well as the time complexity of the same operation. In many cases, the bounds given are tight.

The investigation of the state complexity of regular languages and their operations was already going on in the sixties and seventies. Examples of early studies concentrated on this topic can be found in [17, 20, 21]. However, many problems remain. We are now back to those basic problems with much renewed motivation and interest.

The DFA model has at least the following advantages over other representations of regular languages such as nondeterministic finite automata (NFA) and regular expressions: (1) Checking the equivalence of two DFA can be done in almost linear time [1], while the same problem for NFA, respectively for regular expressions, is PSPACE complete. (2) For each regular language, the minimal DFA that accepts the language is unique up to an isomorphism. This is not the case in general for the other models. (3) There is an $O(n \log n)$-time algorithm for the minization of DFA. However, the same problem for the other models is not known to be polynomial. Thus, the size of a DFA is a natural and objective measure for the language it accepts.

However, there are many ways to measure the size of a DFA: the number of states, the number of transitions, or both the number of states and the number of transitions. One may notice that the number of states gives a linear upper bound on the number of transitions. In the case of a complete DFA, i.e., a DFA whose transition function is defined for each state and each letter in the alphabet, the number of transitions is totally *determined* by the number of states when the alphabet is given. Therefore, the number of states is the key measure on the size of a DFA as well as a natural complexity measure for the language that the DFA accepts. In the following, by a DFA we always mean a complete DFA.

A regular language is accepted by infinitely many different DFA. We use the number of states of the minimal automaton to measure the complexity of the given language. Thus, by the *state complexity* of a regular language L, denoted $C(L)$, we mean the number of states in the minimal DFA that accepts L. By the state complexity of a class $\mathcal{L}$ of regular languages, denoted $C(\mathcal{L})$, we mean the maximum among all $C(L)$, $L \in \mathcal{L}$. When we speak about the state complexity of an operation on regular languages, we mean the state complexity of the resulting languages from the operation. For example, we say that the state complexity of the intersection of an m-state DFA language, i.e., a language accepted by an m-state complete DFA, and an n-state DFA language is exactly mn. This means that mn is the state complexity of the class of languages each of which is the intersection of an m-state DFA language and an n-state DFA language. In other words, there exist two regular languages that are accepted by an m-state DFA and an n-state DFA, respectively, such that the intersection of them is accepted by a minimal DFA of mn states, and this is the worst case. So, in a certain sense, state complexity is a worst-case complexity.

State complexity is a complexity measure only for regular languages. However, it can be extended to cover other families of languages as well. For example, the automaticity studied by Shallit et al. [24] can be considered as an extension of the state complexity. We will not consider any extension

of state complexity in this article.

Examining the state complexity results on the basic operations, e.g., catenation, union, intersection, and complementation, on regular languages in [27], one would notice that all the worst cases are given by using infinite languages only. This observation raises the question: Are finite languages significantly different from (infinite) regular languages in state complexity of their operations? For example, would the state complexity of the union of two finite languages accepted by an m-state and an n-state DFA, respectively, be $O(m+n)$ instead of mn? We will investigate these questions in this article.

Finite languages are, perhaps, one of the most often used but least studied classes of languages. Finite languages are exactly the languages accepted by acyclic finite automata. It has been shown that there is a linear (time) algorithm for the minimization of an acyclic DFA by Revuz in 1992 [15]. However, for the minimization of a general DFA, the best known algorithm has a time complexity $O(n \log n)$ by Hopcropt in 1971 [8].

In this article, we compare the state complexity results for finite and infinite regular languages. We first consider the relatively simple cases, i.e., the operations on languages over a one-letter alphabet. Then we consider the general cases. In the one-letter cases, most of the operations on finite languages have a much lower state complexity than the corresponding operations on regular languages. However, in the general cases, only the catenation of two finite languages, when the first language is accepted by a DFA with a constant number of final states, has a much lower state complexity than its regular language counterpart.

Due to the not-so-positive results in the general cases, we resort to a different concept to try to reduce the number of states for DFA accepting finite languages. The concept of cover automata for finite languages is described in the last section of this article. In many cases, cover automata are a much more concise representation than DFA for finite languages.

2 Preliminaries

A deterministic finite automaton (DFA) is denoted by a quintuple $(Q, \Sigma, \delta, q_0, F)$ where Q is the finite set of states, Σ is the finite alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the start state, and $F \subseteq Q$ is the set of final states. In this paper, *all the DFAs are assumed to be complete DFAs.* By a complete DFA we mean that there is a transition defined for each letter of the alphabet from each state, i.e., δ is a total function. In contrast, a DFA is called an incomplete DFA if its transition function is a partial function.

For any $x \in \Sigma^*$, we use $\#(x)$ to denote the length of x and $\#_a(x)$ for some $a \in \Sigma$ to denote the number of appearances of a in x. The empty word is denoted by ε.

The transition function δ of a DFA is extended to $\hat{\delta} : Q \times \Sigma^* \to Q$ by setting $\hat{\delta}(q, \varepsilon) = q$ and $\hat{\delta}(q, ax) = \hat{\delta}(\delta(q, a), x)$ for $q \in Q$, $a \in \Sigma$, and $x \in \Sigma^*$. In the following, we simply use δ to denote $\hat{\delta}$ if there is no confusion.

A word $w \in \Sigma^*$ is accepted by a DFA $A = (Q, \Sigma, \delta, q_0, F)$ if $\delta(q_0, w) \in F$. The language accepted by A, denoted $L(A)$, is the set $\{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$. Two DFA are said to be equivalent if they accept the same language.

An incomplete DFA can be transformed to an equivalent complete DFA by adding a 'sink state' and transitions, which were undefined before, to the 'sink state', as well as transitions from the 'sink state' to the 'sink state'.

Let $A = (Q, \Sigma, \delta, s, F)$ be a DFA. Then

a) a state q is said to be accessible if there exists $w \in \Sigma^*$ such that $\delta(s, w) = q$;

b) a state q is said to be useful if there exists $w \in \Sigma^*$ such that $\delta(q, w) \in F$.

It is clear that for every DFA A there exists an equivalent DFA A' such that every state of A' is accessible and at most one state is useless (the 'sink state'). A DFA A' as above is called a *reduced* DFA. We will use only reduced DFA in the following.

A nondeterministic finite automaton (NFA) is denoted also by a quintuple $(Q, \Sigma, \eta, q_0, F)$ where $\eta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ is a transition relation rather than a function, and Q, Σ, q_0, and F are defined similarly as in a DFA. The words and languages accepted by NFA are defined similarly as for DFA.

For a set s, we use $|s|$ to denote the cardinality of s. For a language L, we define $L^{\leq l} = \bigcup_{i=0}^{l} L^i$.

For $L \subseteq \Sigma^*$, we define a relation $\equiv_L \subseteq \Sigma^* \times \Sigma^*$ by

$$x \equiv_L y \text{ iff } xz \in L \Leftrightarrow yz \in L \text{ for all } z \in \Sigma^*.$$

Clearly, $\equiv_L$ is an equivalence relation, which partitions Σ^* into equivalence classes. The number of equivalence classes of $\equiv_L$ is called the *index* of $\equiv_L$. The Myhill-Nerode Theorem [9] states that L is regular if and only if $\equiv_L$ has a finite index and the minimal number of states of a complete DFA that accepts L is equal to the index of $\equiv_L$.

For a rather complete background knowledge in automata theory, the reader may refer to [9, 22].

The following lemmas will be used in the subsequent sections. They can be proved rather easily. Thus, we omit the proofs to concentrate on our main results.

Lemma 1 *Let $R \subseteq \Sigma^*$ be a regular language. If there exists an integer n such that*

$$max\{\#(w) \mid w \in \Sigma^* \ \& \ w \notin R\} = n,$$

then any DFA accepting R needs at least $n + 2$ states. In particular, if Σ is a singleton, the minimal DFA accepting R uses exactly $n + 2$ states.

Lemma 2 *Let $m, n > 0$ be two arbitrary integers such that $(m, n) = 1$ (m and n are relatively prime).*

(i) The largest integer that cannot be presented as $cm + dn$ for any integers $c, d > 0$ is mn.

(ii) The largest integer that cannot be presented as $cm + dn$ for any integers $c > 0$ and $d \geq 0$ is $(m - 1)n$.

(iii) The largest integer that cannot be presented as $cm + dn$ for any integers $c, d \geq 0$ is $mn - (m + n)$.

3 Finite vs. regular languages over a one-letter alphabet

As we have mentioned in the introduction, we start our comparison of the state complexity of operations on regular and finite languages from the relatively easy cases, i.e., the languages over a one-letter alphabet.

We first list the basic results below and then give detailed explanations for some of the operations.

We assume that L_1 is an m-state DFA language and L_2 an n-state DFA language, $\Sigma = \{a\}$, and $m, n > 1$.

	Finite	Regular
$L_1 \cup L_2$	$\max(m, n)$	mn, for $(m, n) = 1$
$L_1 \cap L_2$	$\min(m, n)$	mn, for $(m, n) = 1$
$\Sigma^* - L_1$	m	m
$L_1 L_2$	$m + n - 1$	mn, for $(m, n) = 1$
L_1^R	m	m
L_1^*	$m^2 - 7m + 13$, for $m > 4$	$(m-1)^2 - 1$

Note that for **finite languages**, the state complexity for each of the union, intersection, and catenation operations is linear, while it is quadratic for infinite regular languages.

In the above table, all results for finite languages are relatively trivial except for L_1^*. We give an informal proof in the following. Let L_1 be accepted by an m-state DFA A_1 and A is a minimal DFA accepting L_1^*. It is clear that the length of the longest word accepted by A_1 is $m-2$. (Note that the m states include a 'sink state'.) We consider the following three cases (1) A_1 has one final state; (2) A_1 has two final states; or (3) A_1 has three or more final states. If (1), then A has $m-1$ states. For (2), the worst case is given by $L = \{a^{m-2}, a^{m-3}\}$. By (iii) of Lemma 2, the length of the longest word that is not in L_1^* is

$$(m-2)(m-3) - (2m-5) = m^2 - 7m + 11.$$

Then A has exactly $m^2 - 7m + 13$ states. In case (3), it is easy to see that A cannot have more than $m^2 - 7m + 13$ states.

For **regular languages**, we give a more detailed discussion below.

For the **union** operation, it is clear that mn states are sufficient for the resulting minimal DFA. To show that mn states are necessary, it suffices to show that there are at least mn distinct equivalence classes of the relation $\equiv_{L_1 \cup L_2}$. Let $L_1 = (a^m)^*$ and $L_2 = (a^n)^*$, $m, n > 1$ and $(m, n) = 1$. For positive integers p and q, let $m_p = p \bmod m$, $n_p = p \bmod n$, $m_q = q \bmod m$, and $n_q = q \bmod n$, $0 \leq m_p, m_q < m$, $0 \leq n_p, n_q < n$. It turns out that if $m_p \neq m_q$ or $n_p \neq n_q$, then a^p and a^q are not equivalent. However, this is not immediately clear for some cases. For example, let $m_p = -2 \bmod m$, $n_p = -1 \bmod n$, $m_q = -1 \bmod m$, and $n_q = -2 \bmod n$. (In order to explain easily, we use the

negative numbers.) Then neither $m_p = m_q$ nor $n_p = n_q$, but $m_p = n_q$ and $n_p = m_q$. So, both $a^p a^2, a^q a^2 \in L_1 \cup L_2$ and $a^p a, a^q a \in L_1 \cup L_2$. It then appears that $a^p \equiv_{L_1 \cup L_2} a^q$. However, this is not true because it can be proved that $a^p a^{2+m} \in L_1 \cup L_2$, but $a^q a^{2+m} \notin L_1 \cup L_2$ assuming that $m < n$.

The state complexity result for the **intersection** of two regular languages can be similarly proved.

The result for the **catenation** of two regular languages is more involved. We outline a proof in the following. A more detailed proof can be found in [27].

We first give a general example of an m-state DFA language and an n-state DFA language, $(m, n) = 1$, such that mn states are necessary for any DFA that accepts the catenation of the two languages. Let $L_1 = a^{m-1}(a^m)^*$ and $L_2 = a^{n-1}(a^n)^*$. Obviously, L_1 and L_2 can be accepted by an m-state DFA and an n-state DFA, respectively, and $L = L_1 L_2 = \{a^i \mid i = (m-1) + (n-1) + cm + dn$ for some integers $c, d \geq 0\}$. By (iii) of Lemma 2, for $(m, n) = 1$, the largest number that cannot be represented by $cm + dn$, $c, d \geq 0$, is $mn - (m + n)$. Then the largest i such that $a^i \notin L$ is $mn - 2$. So, the minimal DFA that accepts L has at least mn states. We show that mn states are sufficient in the following theorem.

Theorem 1 *For any integers $m, n \geq 1$, let A and B be an m-state DFA and an n-state DFA, respectively, over a one-letter alphabet. Then there exists a DFA of at most mn states that accepts $L(A)L(B)$.*

Proof. The cases when $m = 1$ or $n = 1$ are trivial. We assume that $m, n \geq 2$ in the following. Let $A = (Q_A, \{a\}, \delta_A, s_A, F_A)$ and $B = (Q_B, \{a\}, \delta_B, s_B, F_B)$. By a variation of the subset construction, we know that $L(A)L(B)$ is accepted by the DFA $C = (Q_C, \{a\}, \delta_C, s_C, F_C)$ where

$Q_C = \{< q, P > \mid q \in Q_A \ \& \ P \subseteq Q_B\}$;

$s_C =< s_A, \emptyset >$ if $s_A \notin F_A$ and $s_C =< s_A, \{s_B\} >$ if $s_A \in F_A$;

$\delta_C(< q, P >, a) =< q', P' >$ where $q' = \delta_A(q, a)$ and $P' = \delta_B(P, a) \cup \{s_B\}$ if $q' \in F_A$, $P' = \delta_B(P, a)$ otherwise;

and $F_C = \{< q, P > \mid P \cap F_B \neq \emptyset\}$.

Now we show that at most mn states of Q_C are reachable from s_C.

First we assume that in A there is a final state f in the loop of A's transition diagram. Then $\delta_A(s_A, a^t) = f$ and $\delta_A(f, a^l) = f$ for some nonnegative integers $t < m$ and $l \leq m$. Let $j_1, \ldots, j_r$, $0 < j_1 < \ldots < j_r < l$, be all the integers such that $\delta_A(f, a^{j_i}) \in F_A$ for each $1 \leq i \leq r$. Denote

$P_0 = \{s_B\}$,

$P_1 = \{\delta_B(s_B, a^l), \delta_B(s_B, a^{l-j_1}), \ldots, \delta_B(s_B, a^{l-j_r})\}$,

and for $i \geq 2$ we define

$P_i = \delta_B(P_{i-1}, a^l)$.

Let $\delta_C(s_C, a^t) =< f, S >$. Denote $S_0 = S - \{s_B\}$ and $S_i = \delta_B(S_{i-1}, a^l)$ for each $i \geq 1$. Then we have the following state transition sequence of C:

$$s_C \ \vdash_C^t \ < f, P_0 \cup S_0 > \tag{1}$$

$$\vdash_C^l \ < f, P_0 \cup P_1 \cup S_1 > \tag{2}$$

$$\ldots\ldots\ldots\ldots \quad (3)$$

$$\vdash_C^l \ < f, P_0 \cup P_1 \cup \ldots \cup P_{n-1} \cup S_{n-1} > \quad (4)$$

$$\vdash_C^l \ < f, P_0 \cup P_1 \cup \ldots \cup P_n \cup S_n > \quad (5)$$

Here $p \vdash_C^k q$ stands for $\delta_C(p, a^k) = q$. Denote $P_0 \cup \ldots \cup P_i$ by $\mathcal{P}_i$, $i \geq 0$. Let i be the smallest integer such that $\mathcal{P}_{i-1} = \mathcal{P}_i$. It is clear that $i \leq n$ since B has n states. If $i = n$, then $\mathcal{P}_{n-1} = Q_B$ and

$$< f, \mathcal{P}_{n-1} \cup S_{n-1} >=< f, \mathcal{P}_n \cup S_n >=< f, Q_B > .$$

Therefore, C needs at most $m + l(n-1) \leq m + m(n-1) = mn$ states. If $i < n$, consider the set $S'_{i-1} = S_{i-1} - \mathcal{P}_{i-1}$. Note that every state in S'_{i-1} is in the loop of the transition diagram of B. If for each element r of S'_{i-1}, there exists j, $0 \leq j \leq n-i$, such that $\delta_B(r, a^{jl}) \in \mathcal{P}_{i-1}$ (i.e. $\mathcal{P}_{n-1}$), then the proof is concluded as above. Otherwise, there is an element r_0 of S'_{i-1} and a transition sequence

$$r_0 \ \vdash_B^l \ r_1 \ \vdash_B^l \ \ldots \ \vdash_B^l \ r_{n-i}$$

such that, for some $j, k \leq n-i$ and $j < k$, $r_j = r_k$. (There are at most $n-i$ states not in $\mathcal{P}_{i-1}$.) Then it is easy to verify that $S_{i-1+j} = S_{i-1+k}$. Therefore, $< f, \mathcal{P}_{i-1+j} \cup S_{i-1+j} >=< f, \mathcal{P}_{i-1+k} \cup S_{i-1+k} >$. Thus, the number of states that are reachable from s_C is at most $t + 1 + l(n-1) \leq (m-1) + 1 + m(n-1) = mn$.

Finally we consider the case when no final states of A are in the loop. Let $Q_A = \{0, \ldots, m-1\}$ where $s_A = 0$ and $\delta_A(0, a^i) = i$ for $0 \leq i \leq m-1$. We can assume that $m-2$ is a final state and $m-1$ loops to itself. Otherwise, $L(A)$ can be accepted by a complete DFA with less than m states. Consider the following $m + n - 1$ transition steps of C

$$s_C \ \vdash_C^{m-2} < m-2, T > \vdash_C < m-1, T_0 > \vdash_C < m-1, T_1 >$$

$$\vdash_C \ \ldots \ \vdash_C < m-1, T_n >$$

Let the state $\delta_B(s_B, a^{i+1})$ be t_i, for each $i \geq 0$. Note that $s_B \in T$ and t_i is in T_i. It is clear that there exist j, k such that $0 \leq j < k \leq n$ and $t_j = t_k$. Then it is not difficult to see that $< m-1, T_j >=< m-1, T_k >$. Therefore, at most $m + n$ states are necessary for C. ($m + n < mn$ for $m, n \geq 2$.) □

For the union, intersection, and catenation operations, we have considered only the cases when $(m, n) = 1$. For $(m, n) = t > 1$, we have not obtained exact formulas for those cases. Note that neither $mn/(m, n)$ nor $lcm(m, n)$ (the least common multiple of m and n) is the solution. For example, $a(a^5)^*$ and $(a^9)^*$ are accepted by a 6-state and a 9-state DFA, respectively, but the union of them needs at least 45 states rather than $6 \times 9/3 = 18$ states.

We now consider the last operation on the table, i.e., the **star** operation on infinite regular languages. We find that the proofs for both directions are interesting.

Theorem 2 *The number of states which is sufficient and necessary in the worst case for a DFA to accept the star of an n-state DFA language, $n > 1$, over a one-letter alphabet is $(n-1)^2 + 1$.*

Proof. For $n = 2$, the necessity is shown by a 2-state DFA which accepts $(aa)^*$. For each $n > 2$, the necessary condition can be shown by the DFA $A = (\{0, \ldots, n-1\}, \{a\}, \delta, 0, \{n-1\})$ where $\delta(i, a) = i + 1 \bmod n$ for each i, $0 \leq i \leq n-1$. The star of $L(A)$ is the language $\{a^i \mid i = c(n-1) + dn$, for some integers $c > 0$ and $d \geq 0$, or $i = 0\}$. By (ii) of Lemma 2, the largest i such that $a^i \notin (L(A))^*$ is $(n-2)n$. So, the minimal DFA that accepts $(L(A))^*$ has $(n-2)n + 2$, i.e. $(n-1)^2 + 1$, states.

The proof for showing that $(n-1)^2 + 1$ states are sufficient is more interesting. Let $A = (Q, \{a\}, \delta, s, F)$ be an arbitrary n-state DFA, $n > 1$ and $R = L(A)$. If s is the only final state of A, then $R^* = R$. So, we assume that there is at least one final state f such that $f \neq s$. Clearly, R^* (excluding ε if $s \notin F$) is accepted by the NFA $A' = (Q, \{a\}, \delta', s, F)$ where $\delta' = \delta \cup \{(q, \varepsilon, s) \mid q \in F\}$. For any $X \subseteq Q$, denote by $closure(X)$ the set $X \cup \{q \in Q \mid (p, \varepsilon, q) \in \delta'$ for some $p \in X\}$. Now we follow the subset construction approach to build a DFA $B = (P, \{a\}, \eta, \{s\}, F_P)$ from A' to accept R^* such that $P \subseteq 2^Q$, $\eta(X, a) = closure(\{q \in Q \mid$ there exists $p \in X$ such that $(p, a, q) \in \delta'\})$, and $F_P = \{X \in P \mid X \cap F \neq \emptyset$ or $X = \{s\}\}$. Let f be the first final state from s in A and a^t is the shortest word such that $\delta(s, a^t) = f$. Then $\eta(\{s\}, a^t) = \{s, f\}$. Denote by p_{k_i} the state $\eta(\{s\}, a^{it})$ in P, $i \geq 0$, which is a subset of Q.

We claim that $p_{k_i} \supseteq p_{k_{i-1}}$ for all $i \geq 1$. It is true for $i = 1$ because $\eta(\{s\}, a^t) = \{s, f\}$, and also true for $i > 1$ since

$$p_{k_i} = \eta(\{s\}, a^{it}) = \eta(\{s, f\}, a^{(i-1)t}) = \eta(\{s\}, a^{(i-1)t}) \cup \eta(\{f\}, a^{(i-1)t})$$

$$= p_{k_{i-1}} \cup \eta(\{f\}, a^{(i-1)t}).$$

Then one of the following must be true:

(1) $p_{k_i} = p_{k_{i-1}}$ for some $i \leq n-1$;

(2) $p_{k_{n-1}} = Q$.

This is because if (1) is false, $p_{k_{n-1}}$ contains at least n states and, therefore, (2) is true. Note that if (2) is true, then $\eta(p_{k_{n-1}}, a) = p_{k_{n-1}}$. In any of the cases, the number of states of B is no more than $t(n-1) + 1$ which is at most $(n-1)^2 + 1$. □

For the transformation from an n-state NFA to a DFA, it is clear in the case of finite languages over a one-letter alphabet, at most n states are needed. However, in the case of an infinite regular language over a one-letter alphabet, the problem is still open.

4 Finite vs. regular languages over an arbitrary alphabet

For the one-letter alphabet case which we have discussed in the previous section, the state complexities for most operations on finite languages are of a lower order than their counterpart for regular languages. However, this is no longer true in the case when the size of the alphabet is arbitrary. Although none of the operations on finite languages, except the complementation, can reach the

exact bound for regular languages, most of them have a complexity that is of the same order as the corresponding operation on regular languages.

We list the state complexity of the basic operations for both finite and regular languages over an arbitrary alphabet below. All the results for regular languages are given as exact numbers. However, we use the big "O" notation for most of the results for finite languages due to the fact that either the formulas we have obtained are acutely nonintuitive or we do not have an exact result, yet. More detailed explanations follow the table.

We assume that L_1 and L_2 are accepted by an m-state DFA $A_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ and an n-state DFA $A_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$, respectively, and $m, n > 1$. We use t to denote the number of final states in A_1.

	Finite	Regular
$L_1 \cup L_2$	$O(mn)$	mn
$L_1 \cap L_2$	$O(mn)$	mn
$\Sigma^* - L_1$	m	m
$L_1 L_2$	$O(mn^{t-1} + n^t)$	$(2m-1)2^{n-1}$
L_1^R	$O(2^{m/2})$, for $\|\Sigma\| = 2$	2^m
L_1^*	$2^{m-3} + 2^{m-4}$, for $m \geq 4$	$2^{m-1} + 2^{m-2}$

For the **union** and the **intersection** of finite languages, it was expected that their state complexities would be linear, more specifically $O(m+n)$, but it turns out that both of them are of the order of mn although neither of them can reach the exact bound mn.

It is easy to show that mn states are sufficient for both union and intersection by the following simple argument. We can construct a DFA $A = (Q, \Sigma, \delta, s, F)$ which is the cross-product of A_1 and A_2, i.e., $Q = Q_1 \times Q_2$, $\delta = \delta_1 \times \delta_2$ that is $\delta((q_1, q_2), a) = (\delta(q_1, a), \delta_2(q_2, a))$, $s = (s_1, s_2)$. For the union operation, $F = \{(q_1, q_2) \mid q_1 \in F_1 \text{ or } q_2 \in F_2\}$ and for the intersection operation, $F = \{(q_1, q_2) \mid q_1 \in F_1 \text{ and } q_2 \in F_2\}$.

Note that the pairs of the form (s_1, q_2) where $q_2 \neq s_2$ and (q_1, s_2) where $q_1 \neq s_1$ are never reached from (s_1, s_2), and therefore, are useless. So, $mn - (m + n - 2)$ states are sufficient for both the union and intersection of two finite languages accepted by an m-state and an n-state DFA, respectively. However, this is a very rough upper bound. Much tighter upper bounds for the union and intersection of finite languages are given in [4], which unfortunately are in a very complicated and highly incomprehensible form. Thus, we will not quote them in this paper.

It is more interesting to show that the state complexities of those two operations are indeed of the order of mn but not lower. The following examples were originally given by Shallit [23]. Automaton-based examples are given in [4], which give better lower bounds than the examples below. We choose to present the following examples due to their clarity and intuitiveness.

For the intersection of two finite languages, consider the following example. Let $\Sigma = \{a, b\}$ and

$$L_1 = \{w \in \Sigma^* \mid \#_a(w) + \#_b(w) = 2n\},$$

$$L_2 = \{w \in \Sigma^* \mid \#_a(w) + 2\#_b(w) = 3n\}.$$

Clearly, L_1 is accepted by a DFA with $2n + 2$ states and L_2 by a DFA with $3n + 2$ states. The intersection $L = L_1 \cap L_2$ is

$$\{w \in \Sigma^* \mid \#_a(w) = \#_b(w) = n\}$$

One can prove that any DFA accepting L needs at least n^2 states by the Myhill-Nerode Theorem [9].

For the union of two finite languages, the example is slightly more complicated. Let $\Sigma = \{a, b\}$ and

$$L_1 = \{w \in \Sigma^* \mid \#(w) \leq 3t \text{ and } \#_a(w) + \#_b(w) \neq 2t\},$$

$$L_2 = \{w \in \Sigma^* \mid \#_a(w) + 2\#_b(w) < 3t\}.$$

It is clear that $L_1 \cup L_2$ includes all words in Σ^* of length less than or equal to $3t$ except those words w such that $\#(w) = 2t$ and $\#_b(w) \geq t$. One can prove that any DFA accepting $L_1 \cup L_2$ needs more than t^2 states by checking the number of the equivalence classes of $\equiv_{L_1 \cup L_2}$.

We now consider the **catenation** operation. Notice that for the finite language case, if the number of final states in A_1 is a constant, then the state complexity of catenation is a polynomial in terms of m and n. In particular, if A_1 has only one final state, then the state complexity is linear, i.e., $m + n$. In contrast, for infinite regular languages, there are examples in which A_1 has only one final state but any DFA accepting the catenation of the two languages needs at least $(2m - 1)2^{n-1}$ states [27, 26]. This is one of a few cases in which the state complexities for finite and infinite regular languages, respectively, are in different orders.

We now give the proof for the finite language case. For the general case for the catenation of regular languages, the reader may refer to [27] or [26].

Without loss of generality, we assume that all the DFA we are considering are reduced and ordered. A DFA $A = (Q, \Sigma, \delta, 0, F)$ with $Q = \{0, 1, \ldots, n\}$ is called an ordered DFA if, for any $p, q \in Q$, the condition $\delta(p, a) = q$ implies that $p \leq q$.

For convenience, we introduce the following notation:

$$\left(\begin{array}{c} n \\ \leq i \end{array} \right) = \sum_{j=0}^{i} \left(\begin{array}{c} n \\ j \end{array} \right).$$

Theorem 3 *Let $A_i = (Q_i, \Sigma, \delta_i, 0, F_i)$, $i = 1, 2$, be two DFA accepting finite languages L_i, $i = 1, 2$, respectively, and $\#Q_1 = m$, $\#Q_2 = n$, $\#\Sigma = k$, and $\#F_1 = t$. There exists a DFA $A = (Q, \Sigma, \delta, s, F)$ such that $L(A) = L(A_1)L(A_2)$ and*

$$\#Q \leq \sum_{i=0}^{m-2} \min\left\{k^i, \left(\begin{array}{c} n-2 \\ \leq i \end{array} \right), \left(\begin{array}{c} n-2 \\ \leq t-1 \end{array} \right)\right\}$$

$$+ \min\left\{k^{m-1}, \left(\begin{array}{c} n-2 \\ \leq t \end{array} \right)\right\}. \qquad (*)$$

Proof. The DFA A is constructed in two steps. First, an NFA A' is constructed from A_1 and A_2 by adding a λ-transition from each final state in F_1 to the starting state 0 of A_2. Then, we construct a DFA A from the NFA A' by the standard subset construction. Again, we assume that A is reduced and ordered.

It is clear that we can view each $q \in Q$ as a pair (q_1, P_2), where $q_1 \in Q_1$ and $P_2 \subseteq Q_2$. The starting state of A is $s = (0, \emptyset)$ if $0 \notin F_1$ and $s = (0, \{0\})$ if $0 \in F_1$. Let us consider all states $q \in Q$ such that $q = (i, P)$ for a particular state $i \in Q_1 - \{m-1\}$ and some set $P \subseteq Q_2$. Since A_1 is ordered and acyclic, the number of such states in Q is restricted by the following three bounds: (1) k^i, (2) $\left(\begin{array}{c} n-2 \\ \leq i \end{array} \right)$, and (3) $\left(\begin{array}{c} n-2 \\ \leq t-1 \end{array} \right)$. We explain these bounds below informally.

We have (1) as a bound since all states of the form $q = (i, P)$ are at a level $\leq i$, which have at most k^{i-1} predecessors. By saying that a state p is at level i we mean that the length of the longest path from the starting state to q is i.

We now consider (2). Notice that if $q, q' \in Q$ such that $\delta(q, a) = q'$, $q = (q_1, P_2)$ and $q' = (q_1', P_2')$, then $\delta_1(q_1, a) = q_1'$ and $P_2' = \{\delta_2(p, a) \mid p \in P_2\}$ if $q_1' \notin F_1$ and $P_2' = \{0\} \cup \{\delta_2(p, a) \mid p \in P_2\}$ if $q_1' \in F_1$. So, $\#P_2' > \#P_2$ is possible only when $q_1' \in F_1$. Therefore, for $q = (i, P)$, $\#P \leq i$ if $i \notin F_1$ and $\#P \leq i + 1$ if $i \in F_1$. In both cases, the maximum number of distinct sets P is $\left(\begin{array}{c} n-2 \\ \leq i \end{array} \right)$. The number $n-2$ comes from the exclusion of the sink state $n-1$ and starting state 0 of A_2. Note that, for a fixed i, either $0 \in P$ for all $(i, P) \in Q$ or 0 is not in any set P such that $(i, P) \in Q$.

(3) is a bound since for each state $i \in Q_1 - \{m-1\}$, there are at most $t-1$ final states on the path from the starting state to i (not including i).

For the second term of $(*)$, it suffices to explain that for each $(m-1, P)$, $P \subseteq Q_2$, $\#P$ is bounded by the total number of final states in F_1. $\Box$

Corollary 1 *Let $A_i = (Q_i, \Sigma, \delta_i, 0, F_i)$, $i = 1, 2$, be two DFA accepting finite languages L_i, $i = 1, 2$, respectively, and $\#Q_1 = m$, $\#Q_2 = n$, and $\#F_1 = t$, where $t > 0$ is a constant. Then there exists a DFA $A = (Q, \Sigma, \delta, s, F)$ of $O(mn^{t-1} + n^t)$ states such that $L(A) = L(A_1)L(A_2)$.*

It has been shown in [3] that the bound given in Theorem 3 can be reached in the case $|\Sigma| = 2$.

About the state complexity of the **reversal** of an m-state DFA language, one may easily have a misconception. Many thought, without any hesitation, that it should be linear (in terms of m), especially in the case of finite languages. In fact, it is not even polynomial for both finite and infinite regular languages. We break the misconception by giving two examples in the following: one for finite languages and the other for infinite regular languages. Note that a nontrivial proof for a tight upper bound on the state complexity of the reversal of finite languages can be found in [3].

Example 1 Let $m = 2n + 2$ and $L = \{a, b\}^n a \{a, b\}^{\leq n}$, where $\{a, b\}^{\leq n}$ denotes

$$\lambda \cup \{a, b\} \cup \{a, b\}^2 \cup \cdots \cup \{a, b\}^n.$$

It is clear that L is a finite language accepted by an m-state DFA. One can prove that any DFA accepting L^R needs at least 2^n states.

Example 2 An n-state DFA that accepts an infinite regular language is shown in Figure 1. A proof showing that any DFA accepting the reversal of this language requires at least 2^n states can found in [27].

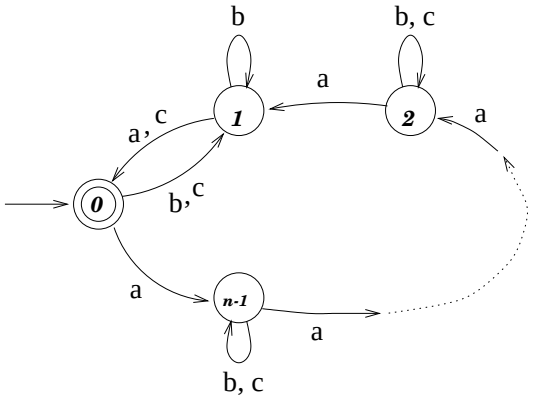

Figure 1: An n-state DFA such that $L(A)^R$ requires 2^n states

For the star operation, the difference between the state complexity for finite and infinite regular languages is that the latter is 4 times the former. Both bounds have been shown to be tight [3, 27]. Here we only give two examples to demonstrate that the bound given on the table can be reached.

Example 3 The n-state DFA A shown in Figure 2 accepts a finite language. It is shown in [3] that any DFA accepting $L(A)^*$ needs at least $2^{n-3} + 2^{n-4}$ states (assuming that n is even).

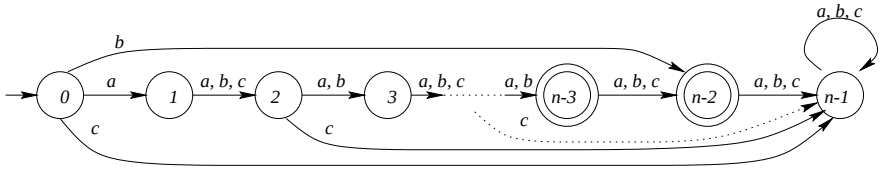

Figure 2: DFA A of n states such that $L(A)^*$ needs $2^{n-3} + 2^{n-4}$ states

Example 4 Let L be the language accepted by the DFA shown in Figure 3. It is shown in [26] that any DFA accepting L^* requires at least $2^{n-1} + 2^{n-2}$ states.

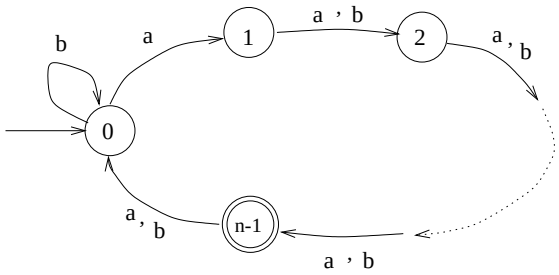

Figure 3: An n-state DFA A_n: The language $(L(A_n))^*$ requires $2^{n-1} + 2^{n-2}$ states

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[2] G. Booch, J. Rumbaugh, I. Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, Reading MA, 1999.

[3] C. Câmpeanu, K. Culik II, K. Salomaa, S. Yu, "State complexity of basic operations on finite languages", Proceedings of the Fourth International Workshop on Implementing Automata, WIA'99, to appear.

[4] C. Câmpeanu, K. Culik II, S. Yu, "Finite languages and cover automata", in preparation.

[5] C. Câmpeanu, N. Sântean, S. Yu, "Minimal cover automata for finite languages", *Proceedings of WIA'98,*, Lect. Notes Comput. Sci. **1660**, Springer-Verlag, 1999, pp. 33–42.

[6] A. K. Chandra, D. C. Kozen, L. J. Stockmeyer, "Alternation", *JACM* **28** (1981), 114-133.

[7] A. Fellah, H. Jürgensen, S. Yu, "Constructions for alternating finite automata", *Intern. J. Computer Math.* vol. 35 (1990) 117-132.

[8] J.E. Hopcropt, "An $n \log n$ algorithm for minimizing the states in a finite automaton", *The Theory of Machines and Computations (Z. Kohavi edited)*, pp. 189-196, Academic Press, New York.

[9] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison Wesley (1979), Reading, Mass.

[10] J. Kaneps, R. Freivalds, "Minimal Nontrivial Space Complexity of Probabilistic One-Way Turing Machines", *Proceedings of Mathematical Foundations of Computer Science*, Banská Bystryca, Czechoslovakia, August 1990, *Lecture Notes in Computer Science*, vol 452, pp. 355-361, Springer-Verlag, New York/Berlin, 1990.

[11] E. Leiss, "Succinct representation of regular languages by boolean automata", *Theoretical Computer Science* 13 (1981) 323-330.

[12] A.R. Meyer and M.J. Fischer. "Economy of description by automata, grammars, and formal systems", FOCS 12 (1971) 188-191.

[13] B. Ravikumar, "Some applications of a technique of Sakoda and Sipser", *SIGACT News*, 21.4 (1990) 73-77.

[14] B. Ravikumar and O. H. Ibarra, "Relating the type of ambiguity of finite automata to the succinctness of their representation", *SIAM J. Comput.* vol. 18, no. 6 (1989) 1263-1282.

[15] D. Revuz, "Minimisation of acyclic deterministic automata in linear time", *Theoretical Computer Science* 92 (1992) 181-189.

[16] T. Jiang and B. Ravikumar, "Minimal NFA Problems are Hard", *Proceedings of 18th ICALP*, Lecture Notes in Computer Science 510, Springer-Verlag (1991) 629-640.

[17] F.R. Moore, "On the Bounds for State-Set Size in the Proofs of Equivalence Between Deterministic, Nondeterministic, and Two-Way Finite Automata", *IEEE Trans. Computers* 20 (1971) 1211-1214.

[18] D. Raymond, D. Wood, S. Yu edited, *Automata Implementation*, — First Intern. Workshop on Implementing Automata, WIA'96, LNCS 1260.

[19] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-Oriented Modeling and Design*, Prentice Hall, Englewood Cliffs NJ, 1991.

[20] A. Salomaa, *On the Reducibility of Events Represented in Automata*, Annales Academiae Scientiarum Fennicae, Series A, I. Mathematica 353, 1964.

[21] A. Salomaa, *Theorems on the Representation of events in Moore-Automata*, Turun Yliopiston Julkaisuja Annales Universitatis Turkuensis, Series A, 69, 1964.

[22] A. Salomaa, *Theory of Automata*, Pergamon Press (1969), Oxford.

[23] J. Shallit, Private communication, Nov. 1996.

[24] J. Shallit and Y. Breitbart, "Automaticity I: Properties of a Measure of Descriptional Complexity", *Journal of Computer and System Sciences*, **53**, 10-25 (1996).

[25] D. Wood and S. Yu edited, *Automata Implementation*, — Second Intern. Workshop on Implementing Automata, WIA'97, LNCS 1436.

[26] S. Yu, "Chapter 2: Regular languages", *Handbook of Formal Languages*, G. Rozenberg and A. Salomaa eds., Springer, 1997.

[27] S. Yu, Q. Zhuang, K. Salomaa, "The State Complexity of Some Basic Operations on Regular Languages", *Theoretical Computer Science* 125 (1994) 315-328.