Volume 5, Issue 12, December 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

State Complexity of Unary and Binary Operations on Regular Languages

Rajesh Kumar
CRM Jat College, Hisar,
Haryana, India

Abstract— It appears that the state complexity of each operation has its own special features. Thus, it is important and practical to calculate good estimates for some commonly used general cases. In this paper, the author consider the state complexity of combined Boolean operations on regular language and give an exact bound for all of them in the case when the alphabet is not fixed. Moreover, the author show that for any fixed alphabet, this bound can be reached in infinite cases.

Keywords— Alternating finite automaton, Automata, Combined operations, Estimation, Formal languages, Multiple operations, State complexity.

I. INTRODUCTION

Motivated by the increasing sizes of finite automata machine that are used in recent applications, state complexity has become one of the most important topics in automata and formal language theory [7]. A state complexity for regular languages is the the number of states needed by various finite-state automata(deterministic, non-deterministic) in order to accept the language. Often the state complexity influences the computational complexity of algorithms that use regular languages. All the most fundamental operations that preserve regularity are intersection, union and complementation. It is well known that if M languages are each recognized by deterministic (or non-deterministic) finite automata with n states, then their intersection is recognized by a deterministic (respectively non-deterministic) finite automaton with n^m states (obtained as the cartesian product of the M given automata[3]). Given n languages $L^{(0)}_n$, $L^{(1)}_n$, $L^{(2)}_n$,...,$L^{(n-1)}_n$ each of which is recognized by a DFA (deterministic finite automaton) with n states: what is the minimum number of states of the minimal DFA for the intersection in the worst case? The author claims a lower bound of n!+1; but, as Berstel[5] points out, this does not completely solve the problem, since the best known upper bound is n^n. Below the author show a lower bound of n^n; so the well-known upper bound is actually optimal. This can be extend result to non-deterministic finite automata. The author also consider the operation of union of regular languages. For notation and the definition of DFA (deterministic finite automaton) and NFA (non-deterministic finite automaton) the author will follow [3]. A partial DFA is an NFA whose next-state relation is a partial function (i.e., for every state and every input symbol there is one or no next state). An AFA (alternating finite automaton, see [2] and [1]) is a generalization of an Non-deterministic Finite Automata; an AFA has a state graph like an NFA but, in addition, every state has a boolean function of states attached to it; recogination of an input is decided by these boolean functions (in the case of an NFA these boolean functions are the logical OR of some states); for the exact definition (which is rather long) see [2] or [4]. In this paper AFA's(Alternating finite automaton) are only mentioned in passing; the reader can skip any mention of AFA's without loss of continuity.

II. DEFINITIONS AND NOTATION

Let Σ be an alphabet, i.e., a non-empty, finite set of letters(symbols). By Σ^* the author denote the set of all finite words(strings of letters) over Σ, and by λ, the empty word (a word having zero symbols). The operation of concatenation of two words x and y is denoted by x.y, or simply xy. For $w \in \Sigma^*$,the reverse order of symbols in w is denoted by w^R. A NFA over Σ, is a tuple $M=(Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, $\delta: Q \times \Sigma \rightarrow 2^Q$ is a next-state function, q_0 is an initial state and $F \subseteq Q$ is a set of final states. δ function is extended over $(Q \times \Sigma^*)$ in the usual way. M is deterministic (DFA) if $\delta: Q \times \Sigma \rightarrow Q$. We consider complete DFAs, that is, those whose transition function "δ" is a total function. The size of M is the total number of its states. The language of M, denoted by L(M), belongs to the family of regular languages and consists of those strings accepted by M in the usual way. For a background on finite automata and regular languages the author refer the reader to[11].

III. STATE COMPLEXITY

A. Star and Reversal

In [8,9], estimation based state complexity of nondeterministic automata was introduced. Briefly speaking, for a combined operation on regular languages, the method first estimates the state complexity(nondeterministic) of the combined operation using the mathematical composition of the state complexities of its component operations, and then converts it to an estimate of the deterministic state complexity. For example, for $(L(A) \cup L(B))^*$ where A and B are DFAs

of m states and n states, respectively, the state complexity(nondeterministic) of L=L(A)∪L(B) is m+n+1 and that of L^* is m+n +2, which is then converted to an estimate of the deterministic state complexity 2^{m+n+2}. It has been shown that this method can obtain good estimates for the combined operations: star of intersection, star of union, star of catenation, and star of reversal. However, this method clearly has its limitations. Using this method it is observed that the complexity of union of n_1-state, n_2-state, and n_3-state DFA languages are $2^{n1+n2+n3+2}$. However, the actual state complexity of this combined operation is $n_1n_2n_3$. It seems that this method may work well for all combined operations with the final component operation having an exponential state complexity, e.g., star or reversal. Indeed, it works well when a combined operation is ended with the star operation. However, it does not work well in general for combined operations that are ended with reversal. For example, the state complexity of reversal of intersection of an m-state DFA language and an n-state DFA language is $2^{m+n}-2^m-2^n+2$. However, the author would obtain an estimate 2^{mn+1} using this method.

The following result was obtained in [13], where a regular operation expression is an expression built from occurrences of binary operations union and concatenation, occurrences of the unary operation star, and variables, where each variable occurs at most once in the expression, and nsc(f) denotes the nondeterministic state complexity of the operation f denoted by a regular operation expression.

Theorem 1. Let f be an operation defined by a regular operation expression with k variables, and denote the sizes of the NFAs for the argument languages by $m_1, \ldots, m_k$. Then

$$nsc(f) \leq 1 + \sum_{i=1}^{k} \qquad (1)$$

Using the above result, the author can claim the following estimates.

Let f be an operation defined by a regular operation expression with k variables and denote the sizes of the NFAs for the argument languages by $m_1, \ldots, m_k$. Then the state complexity of f is no more than $2^{m1+\ldots+mk+1}$. So it is clear that when the unary star operation is the final operation of f, the upper bound is pretty tight.

There are many different combinations of two basic operations selected from catenation, star, reversal, intersection, and union. Note that the author consider $(L_1^R)^*$ and $(L_1^*)^R$ as the same combined operation because $(L_1^R)^* = (L_1^*)^R$. The combined operations $(L_1^*)^* = L_1^*$ and $(L_1^R)^R = L_1$ are not counted, either. Among these combined operations, the state complexities of the following ones have been studied in the literature:$(L_1 \cup L_2)^*$ in [11], $(L_1 \cap L_2)^*$ in [8], $(L_1L_2)^*$, $(L_1^R)^*$ in [9], $(L_1 \cup L_2)^R$, $(L_1 \cap L_2)^R$, $(L_1L_2)^R$, $L_1L_2^*$, $L_1L_2^R$ in [3], $L_1(L_2 \cup L_3)$, $L_1(L_2 \cap L_3)$ in [4], $L_1^* \cup L_2$, $L_1^* \cap L_2$, $L_1^R \cup L_2$, $L_1^R \cap L_2$ in [11], $L_1L_2L_3$, the combined Boolean operations $L_1 \cup L_2 \cup L_3$, $L_1 \cap L_2 \cap L_3$, $(L_1 \cup L_2) \cup L_3$, and $(L_1 \cap L_2) \cup L_3$ in [8], where L_1, L_2, and L_3 are three regular languages.

Although the state complexity of $(L_1L_2)^R$ has been considered in[8], only an upper bound has been obtained. In this paper, the author prove, by providing some witness DFAs, that the upper bound, $3 \cdot 2^{m+n-2}-2^n+1$, proposed in [10] is indeed the state complexity of this combined operation when m≥2 and n≥1.

The author also show that, unlike some other combined operations, the state complexities of $(L_1 \cap L_2)L_3$, $L_1L_2 \cap L_3$, and $L_1L_2 \cup L_3$ in general cases are equal to the compositions of the state complexities of their component operations, while the state complexities of $L_1^RL_2$, $L_1^*L_2$ and $(L_1 \cup L_2)L_3$ are close to the compositions.

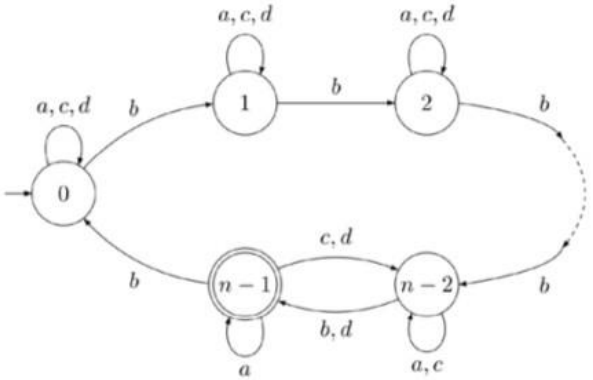

Fig. 1. Witness DFA N which shows that the upper bound of the state complexity of $(L(M)L(N))^R$, $3 \cdot 2^{m+n-2}-2^n+1$, is reachable when m, n≥2.

Theorem 2. For any integers m≥2 and n≥2, there exist a binary DFA A of m-states and a binary DFA B of n-states such that any DFA accepting the language L(A)L(B) needs at least $m2^n-2^{n-1}$ states.

Proof. Let m and n be arbitrary but fixed integers such that m≥2 and n≥2 . Let d=(m−n+1) mod (m−1) and let = {a,b}

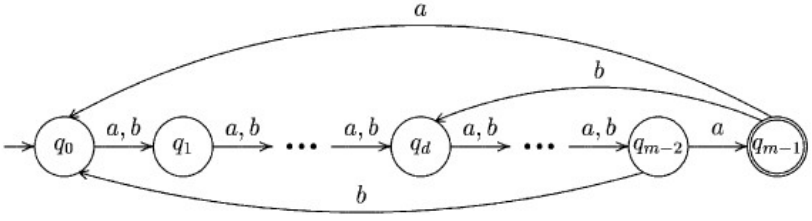

Fig. 2. The deterministic finite automaton A; d=(m−n+1) mod (m−1).

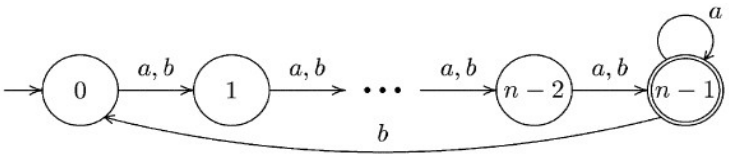

Fig. 3. The deterministic finite automaton B.

Define an m-state DFA A = (Q_A, Σ, δ_A, q_0, F_A), where Q_A = {q_0, q_1, . . . , q_{m-1}}, F_A ={q_{m-1}}, and for any i∈ {0, 1, . . . , m − 1},

$$\delta_A(q_i, X) = \begin{cases} q_j, j = (i+1) mod(\mathbf{m}) & if X = a, \\ q_{i+1} & if \mathbf{i} \leq m-3 \ \mathbf{and} X = b, \\ q_0 & if \mathbf{i} = m-2 \mathbf{and} X = b, \\ q_d, d = (m-n+1) mod(\mathbf{m}-\mathbf{1}) & if \mathbf{i} = m-1 \mathbf{and} X = b \end{cases}$$

Define an n-state DFA B=(Q_B, Σ, δ_B, q_0, F_B), where Q_B = {q_0 , q_1, . . . , q_{n-1}}, F_A = {q_{n-1}}, and for any i ∈ {0, 1, . . . , n − 1}

$$\delta_B(q_i, X) = \begin{cases} i+1 & if \mathbf{i} \leq n-2 \ \mathbf{and} X = a, \\ n-1 & if \mathbf{i} = n-1 \ \mathbf{and} X = a, \\ (i+1) mod(\mathbf{n}) & if X = b \end{cases}$$

The DFA A and B are shown in Figs. 2 and 3, respectively.

B. Intersection and union

Note that for the complement operation of an m-state DFA, it is easy to verify that m states are necessary and sufficient. In the following, the author consider the operations of intersection and union. Given two DFAs A and B, we can construct a DFA for the intersection of L(A) and L(B) based on the Cartesian product of states. For details on the Cartesian product construction, refer to Hopcroft and Ullman [6].

Theorem 3. Given two DFAs A = (Q_A, Σ, δ_A, q_0, F_A) and DFA B = (Q_B, Σ, δ_B, q_0, F_B)
let A∩B(Q_A×Q_B, Σ, δ,(q_A,q_B), F_A× F_B), where for all p∈ Q_A and q∈ Q_B and a∈ Σ,
δ((p,q),a) = (δ_A(p,a), δ_B(q,a)). Then, L(A∩B) = L(A)∩L(B).

Since the automaton A∩B constructed in theorem 3.2.1 is deterministic, it follows that mn states are sufficient for the intersection of L(A) and L(B), where |A|=m and |B|=n. Note that mn is a tight bound for the intersection of two regular languages[10]. The author assign a unique number for each state from 1 to m in A and from 1 to n in B, where |A|=m and |B|=n. Assume that the m^{th} state and the n^{th} state are the sink states in A and B, respectively. Let A∩B denote the resulting intersection automaton that the author compute using the Cartesian product of states. By the construction, A∩B is deterministic since A and B are deterministic. Therefore, a DFA for L(A)∩L(B) is obtain. Next, the author minimize A∩B by merging all equivalent states and removing unreachable states from the start state.

Given minimal DFAs A and B, all states (i,n) for 1≤i≤ m and all states (m,j) for 1≤j≤ n of A∩B are equivalent. Consider all states (1,j), for 1<j≤n, of A∩B. Since the start state of A has no in-transitions. It implies that (1,j) is not reachable from (1,1) in A∩B and, therefore, these states are useless as shown in Fig. 4. A similar result can be establish for the states (i,1), for 1<i≤ m.

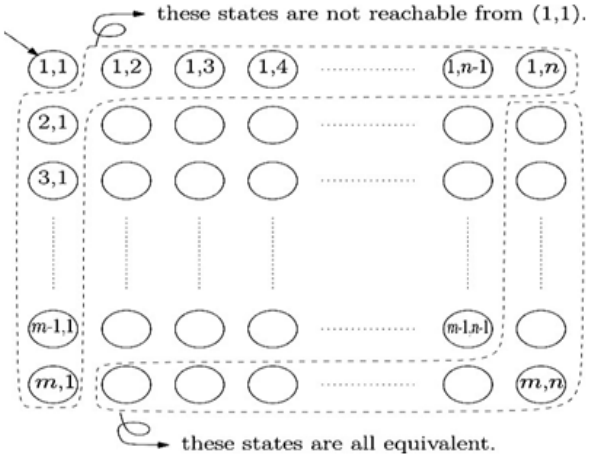

Fig. 4 The figure depicts the intersection automaton A∩B constructed for two suffix-free minimal DFAs A and B. Note that, by theorem 3.2.1, all states in the last row and in the last column are equivalent. Similarly, by theorem 3.2.2, all states, except for the start state (1,1), in the first row and in the first column are unreachable from (1,1).

Theorem 4. Given minimal DFAs A and B, all states(i,1), for 1<i≤ m, and all states(1,j) , for 1<j≤ n, are useless in A∩B. Once the author minimize A∩B based on theorem 3 and 4, the resulting minimal DFA has mn−2(m+n)+6 states.

Theorem 5. Given two minimal DFAs A and B, mn−2(m+n)+6 states are necessary and sufficient in the worst-case for the minimal DFA of L(A)∩L(B), where |Σ|≥3 and m,n≥3.

The previous consideration together with Fig. 4 shows that mn−2(m+n)+6 states are sufficient. We prove the necessary condition by giving two suffix-free minimal DFAs that reach the bound.

Assume that $\Sigma=\{a,b,\#\}$. Let A be the minimal DFA for $L=\{\#w|w\in\{a,b\}^*, |w|_a \equiv 0(\text{mod } m-2)\}$ and B be the minimal DFA for $L=\{\#w|w\in\{a,b\}^*, |w|_b \equiv 0(\text{mod } n-2)\}$

L(A) and L(B) are suffix-free since all strings have only one occurrence of # which may occur only as the first symbol in any string. It is easy to verify that |A|=m and |B|=n. Let $L=L(A)\cap L(B)$. We claim that the minimal DFA for L needs mn−2(m+n)+6 states. To prove the claim, it is sufficient to show that there exist a set R of mn−2(m+n)+6 strings over Σ that are pairwise inequivalent modulo the right-invariant congruence of L.

Let $R=R1\cup R2$, where

$R_1=\{\lambda, \#\#\}$,

$R_2=\{\#a^i b^j \,|1\le i\le m-2 \text{ and } 1\le j\le n-2\}$.

Any string $\#a^i b^j$ from R_2 is inequivalent with λ since $\#a^i b^j \cdot \#$does not $\in$ L but $\lambda\cdot\#\in L$[12]. Similarly, $\#a^i b^j$ is inequivalent with ## since $\#a^i b^j \cdot a^{m-2-i}b^{n-2-j} \in L$ but $\#\#.a^{m-2-i}b^{n-2-j}$ does not $\in$ L. The two strings λ and ## of R_1 are inequivalent as well.

Next, consider two distinct strings $\#a^i b^j$ and $\#a^k b^l$ from R_2. Since $\#a^i b^j \neq \#a^k b^l$, $\#a^i b^j \cdot a^{m-2-i}b^{n-2-j}\in$ L but $\#a^k b^l a^{m-2-i}b^{n-2-j}$ does not $\in$ L. Therefore, any two distinct strings from R_2 are inequivalent.

Thus, all mn−2(m+n)+6 strings in R are pairwise inequivalent. This concludes the proof.

IV. CONCLUSIONS

In this paper, the we studied the state complexities of operations like, e.g., union, intersection, complementation, and reversal, on finite languages . We obtained the state complexities of particular combined operations that are $(L_1L_2)^R$, L_1L_2, $L_1^* \cap L_2^*$ and $L_1^* \cup L_2^*$ where L_i an n_i-state DFA language, $n_i \geq 2$, $1 \leq i \leq k$, and $k \geq 2$. The state complexities of these combined operations are all less than the mathematical compositions of the state complexities of their component individual operations. Comparing with other known state complexities of combined operations, it is interesting to see that the state complexities of $L_1^* \cap L_2$ and $L_1^* \cup L_2$ are the same, and $L_1^* \cap L_2^*$ and $L_1^* \cup L_2^*$ share the same state complexity, whereas the state complexities of $(L_1 \cup L_2)^*$ and $(L_1 \cap L_2)^*$ are different.

REFERENCES

[1] A. Chandra, D. Kozen and L. Stockmeyer, Alternation, J. ACM 28 (1981) 114-133

[2] E. Leiss, Succint representation of regular languages by boolean automata, Theoret. Comput. Sci. 13 (1981) 323- 330.

[3] G. Liu, C. Martin-Vide, A. Salomaa, S. Yu, State complexity of basic language operations combined with reversal, Information and Computation 206 (2008) 1178–1186.

[4] G. Rozenberg, A. Salomaa, Handbook of Formal Languages, Springer-Verlag, Berlin, Heidelbergm, New York, 1997

[5] J. Berstel, D. Perrin, Theory of Codes, Academic Press Inc., 1985.

[6] J. Hopcroft, J. Ullman, Introduction to Automata Theory Languages and Computation, 2nd ed., Addison-Wesley, Reading, MA, 1979.

[7] K. Salomaa, S. Yu, On the state complexity of combined operations and their estimation, International Journal of Foundations of Computer Science 18 (4) (2007) 683–698.

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

[9] M. Domaratzki, K. Salomaa, State complexity of shuffle on trajectories, Journal of Automata Languages and Combinatorics 9 (2–3) (2004) 217–232.

[10] S. Yu, Q. Zhuang, K. Salomaa, The state complexities of some basic operations on regular languages, Theoretical Computer Science 125 (2) (1994)315–328

[11] S. Yu, Regular Languages, In [23] Ch.1 (1997) 41–110.

[12] S. Yu, State complexity: Recent results and open problems, invited talk at International Colloquium on Automata, Languages and Programming 2004 Formal Language Workshop, also appears in Fundamenta Informaticae 64 1–4 (2005) 471–480.

[13] S. Yu, On the state complexity of combined operations, in: invited talk at 11th International Conference on Implementation and Application of Automata, in: Lecture Notes in Computer Science, vol. 4094, Springer, 2006, pp. 11–22.

