IJCSES International Journal of Computer Sciences and Engineering Systems, Vol. 5, No. 2, April 2011
CSES International © 2011 ISSN 0973-4406

State Complexity of Combined Operations on Alternating Finite Automata

Kavitha Joseph

Department of Mathematics, CMR Institute of Technology, I. T. Park Road, Bangalore, INDIA
E-mail:kavijoseph_cmrit@rediffmail.com

Abstract: *This paper investigates the state complexity of combined operations on regular languages. In particular it investigates the state complexity of concatenation of complement and shuffle of complement of two regular languages represented by an alternating finite automaton.*

Keywords: *Alternating finite automata, state complexity and regular languages.*

1. INTRODUCTION

Motivated by several applications and implementation of finite automata in software engineering, programming languages and other practical areas in computer science, the state complexity of deterministic finite automata (DFA) and nondeterministic finite automata (NDFA) have been studied during the last decade.

The state complexity of a regular language is the minimal number of states of the automaton representing the language. The state complexity of an operation on regular languages is a function that associates the sizes of the automata representing the operands of the operation to the minimal number of states of the automata representing the resulting language.

Some early results concerning the state complexity of regular languages can be found in [3, 4, and 13]. Yu *et al.* [4] were the first to systematically study the complexity of regular language operations. Motivated by the result of Yu *et al.* [4] several authors have investigated the state complexity of finite languages operations and unary languages operations [refer 5, 6, 14 and 15]. State complexity results concerning operations on unary regular languages represented by DFAs are elaborated in the survey articles [7, 11]. The nondeterministic state complexity of regular languages operations was studied by Holzer and Kutrib in [10, 16]. The state complexity of some of the operations like concatenation, complement, star, reversal on regular languages was investigated in [8, 9, 10, 19, and 20]. However, almost all the operations which have been studied are individual operations. In applications of regular languages combined operations are quite often performed in addition to individual operations Yu *et al.* [17] were the first to systematically study the state complexity of combined operations on regular languages. In [17], the authors discussed the state complexity of concatenation and reversal combined with star operation for a DFA. After their work, the state complexity of combined operations for regular languages is studied in [18, 21]. Motivated by their work, in this paper I studied the state complexity of combined operations for an Alternating Finite Automata (AFA).

The notion of alternation is a natural generalization of nondeterminism. It received its first formal treatment by Chandra, Kozen and Stockmeyer in 1976 [1]. In this paper they proved that the AFA are precisely as powerful as DFA as far as language recognition is concerned. They have also shown that there exists k-state AFA such that any equivalent complete DFA has at least 2^{2^k} states. A more detailed treatment of AFA and their operations can be found in [2]. In [4], it has been shown that a language L is accepted by an n-state DFA if and only if the reversal of L, that is L^R, is accepted by a $\log n$-state AFA. So the use of an AFA instead of DFA guarantees a logarithmic reduction in the number of states. In addition, operations such as union, intersection, complement, difference and

Manuscript received May 25, 2010
Manuscript revised December 15, 2010

shuffle for an AFA are much simpler and more efficient to implement than the corresponding DFA operations.

2. ALTERNATING FINITE AUTOMATA

Let B denote the two element Boolean algebra $B = (\{0, 1\}, \wedge, \vee, \bar{}, 0, 1)$. Let Q be a set. Then B^Q is the set of all mappings of Q into B and $u \in B^Q$ can be considered as a vector of $|Q|$ entries, indexed by elements of Q, with each entry indexed by elements of Q, with each entry being from B. For $u \in B^Q$ *and* $q \in Q$, u_q to denote the image of q under u.

An AFA A is a quintuple A= (Q, Σ, s, F, g), where

Q is the finite set of states;

Σ is the input alphabet;

$s \in Q$ is the starting state;

$F \subseteq Q$ is the set of final states;

g is a function of Q into the set of all functions of $\Sigma \times B^Q$ into B. For each state $q \in Q$, $g(q)$ is a function from $\Sigma \times B^Q$ into B, which is often denoted by g_q in the sequel. For each state $q \in Q$ and $a \in \Sigma$, define $g_q(a)$ to be the Boolean function $B^Q \to B$ such that $g_q(a)(u) = g_q(a, u)$.

Thus, for $u \in B^Q$, the value of $g_q(a)(u) = g_q(a, u)$, is either 1 or 0. Define the function g_Q: $\Sigma\, B^Q \to B$ by putting together the $|Q|$ functions g_q: $\Sigma \times B^Q \to B$, $q \in Q$, as follows. For $a \in \Sigma$ and $u, v \in B^Q$, $g_Q(a, u) = v$ iff $g_q(a, u) = v$ for each $q \in Q$. *For* $f \in B^Q$, $f_q = 1$ *iff* $q \in F$. Extend g to a function of Q into the set of all functions Σ^* $B^Q \to B$ as follows.

$$g_q(w,u) = \begin{cases} u_q & \text{if } w = \lambda \\ g_q(a, g(w', u)) & \text{if } w = aw' \end{cases}$$

A word $w \in \Sigma^*$ is accepted by A iff $g_s(w, f) = 1$ where f is the characteristic vector of F. The language accepted by A is the set

$$L(A) = \{w \in \Sigma^* / g_s(w,f) = 1\}$$

Example. *Define an AFA* $A = (Q, \Sigma, s, F, g)$, *where*

$Q = \{q_0, q_1, q_2\}$,

$\Sigma = \{a, b\}$,

$s = \{q_0\}$,

$F = \{q_2\}$

and g is given by

Let w = aba. Then w is accepted by A as follows:

State	*a*	*b*
q_0	$q_1 \wedge q_1$	0
q_1	q_2	$q_1 \wedge q_2$
q_2	$q_1 \wedge q_2$	$q_1 \vee q_2$

$$\begin{aligned} & g_{q_0}(aba, f) \\ &= g_{q_1}(ba, f) \wedge g_{q_2}(ba, f) \\ &= (g_{q_1}(a, f) \wedge g_{q_2}(a, f)) \wedge (g_{q_1}(a, f) \vee \overline{g_{q_2}(a, f)}) \\ &= (g_{q_2}(\lambda, f) \wedge \overline{(g_{q_1}(\lambda, f))} \wedge g_{q_2}(\lambda, f))) \wedge (g_{q_2}(\lambda, f) \vee \overline{\overline{g_{q_1}(\lambda, f)} \wedge g_{q_2}(\lambda, f))} \\ &= (f_{q_2} \wedge (\overline{f_{q_1}} \wedge f_{q_2})) \wedge (f_{q_2} \vee \overline{(\overline{f_{q_1}} \wedge f_{q_2})}) \\ &= (1 \wedge (\bar{0} \wedge 1)) \wedge (1 \vee \overline{\bar{0} \wedge 1)}) \end{aligned}$$

3. OPERATIONS ON REGULAR LANGUAGES

The worst-case complexities of star of concatenation and star of reversal are discussed in [17] and in that paper the authors proved that the state complexity of the combined operations is very different from the combination of the state complexities of their individual operations. The worst-case complexity of an AFA for an individual operations concatenation, complement and shuffle are discussed in [12]. The coming next section gives the upper bound for the combined concatenation of complement operation. The next section discusses the state complexity of shuffle of complement.

4. CONCATENATION OF COMPLEMENT

Theorem 4. 1: *For any positive integers* m_1, $m_2 > 1$, *let* A_1 *be an* m_1*-state AFA without negation and let* A_2 *be an* m_2*-state AFA. Then* $m_1 + m_2$ *states are sufficient in the worst case for an AFA to accept the language* $\overline{L(A_1)}\,\overline{L(A_2)}$, *where* $L(A_1)$ *and* $L(A_2)$ *were defined on disjoint sets of alphabets.*

Proof. *Let* A_1 *be an AFA* $= (Q_1, \Sigma_1, s_1, F_1, g')$ *accepts the language* $L(A_1)$ *with* m_1 *states.*

Let A_2 *be an AFA* $A_2 = (Q_2, \Sigma_2, s_2, F_2, g'')$ *accepts the language* $L(A_2)$ *with* m_2 *states such that* $Q_1 \cap Q_2 = \Phi$ *and* $\Sigma_1 \cap \Sigma_2 = \Phi$.

The construction of an $m_1 + m_2$ *state AFA A which accepts* $\overline{L(A_1)}\,\overline{L(A_2)}$ *is defined as follows.*

$A = (Q, \Sigma, s_1, F, g)$

$Q = Q_1 \cup Q_2$,

$\Sigma = \Sigma_1 \cup \Sigma_2$,

$F = Q_2 - F_2$,

and g is defined as

$$g_q(a,u)=\begin{cases}\overline{g'_q(a,u)} & \text{if } q\in Q_1,\ a\in\Sigma_1 \quad (1)\\ \overline{g''_q(a,u)} & \text{if } q\in Q_2,\ a\in\Sigma_2 \quad (2)\\ q & \text{if } q\in F_1,\ a\in\Sigma_2 \quad (3)\\ \overline{g''_{s_2}(a,u)} & \text{if } q\notin F_1,\ a\in\Sigma_2 \quad (4)\end{cases}$$

where $\overline{g}$ is the dual of g.

Claim: $L(A) = \overline{L(A_1)}\,\overline{L(A_2)}$

Let $w \in L_1 L_2$. *Then* $w = xy$ *for some* $x \in L_1$ *and* $y \in L_2$.

Then by (2), $g''_{s_2}(y)=1[\because F=Q_2-F_2]$

By (4), $g_q(y) = 1$ *for all* $q \notin F_1$

By (1) and (3), $= g_{s_1}(xy,u)=\overline{g'_{s_1}(x)\,g_q(g)/q\notin F_1}$

$\because F=Q_2-F_2$ and $g_q(y)=1$, $\overline{g'_{s_1}(x)\,g_q(y)/q\notin F_1}=\overline{0}=1$

Therefore, $g_{s_1}(xy,u)=1$

ie., $g_{s_1}(w,u)=1$

$\Rightarrow w\ L(A)$.

Let $w\notin \overline{L(A_1)}\,\overline{L(A_2)}$.

We've to show that $w \notin L(A)$. *There are two cases.*

Case (1): There is no suffix y of w such that $y \in L(A_2)$

ie., w = xy such that $y \notin L(A_2)$

$$\because y\notin L(A_2),\ g''_{s_2}(y)=0[by(2)]$$

By (4), $g_q(y) = 0$ *for all* $q \notin F_1$.

By (1) and (3), $= g_{s_1}(xy,u)=\overline{g'_{s_1}(x)\,g_q(y)/q\notin F_1}$

$$\because F=Q_2-F_2 \text{ and } g_q(y)=1,\ \overline{g'_{s_1}(x)\,g_q(g)/q\notin F_1}=\overline{1}=0$$

$g_{s_1}(xy,u)=0$

$g_{s_1}(xy,u)=0$

Therefore, $w \notin L(A)$.

Case (2): Let $y = y_1 y_2 y_3 \ldots y_n \in \overline{L(A_2)}$ *are distinct suffixes of w. Then there are distinct*

words $x = x_1x_2\ldots x_n\Sigma^*$ *and* $x \notin \overline{L(A_1)}$ *such that* $w = xy$

$= x_1x_2\ldots x_ny_1y_2y_3\ldots y_n \notin \overline{L(A_1)}\,\overline{L(A_2)}$

$$\because y=y_1\,y_2\,y_3\ldots y_n\in\overline{L(A_2)}$$

We've $g''_{s_2}(y)=1$

Therefore, $g_{s_1}(xy,u)=\overline{g'_{s_1}(x)\,g_q(y)/q\notin F_1}=\overline{1}=0$

$[\because x\notin\overline{L(A_1)}]$

ie., $g_{s_1}(w,u)=0$

$\Rightarrow w \notin L(A)$

Hence the machine A accepts the concatenation of complement of two regular languages L_1 *and* L_2 *represented by two AFAs* A_1 *and* A_2.

Theorem 4.2 *For each AFA A = (Q, Σ, s, F, g) we can construct an equivalent AFA*

A_1 = $(Q_1, \Sigma_1, s_1, F_1, g')$ *with* $|Q'|\le 2|Q|$ *such that* $g'_q(a)$ *is defined with only the* Λ *and V operations for each* $q \in Q'$ *and* $a \in \Sigma$. *That is A′ is an AFA without negation.*

The proof of this theorem is given in [4].

Therefore as a consequence of the theorems 4.1 and 4. 2, we have the following theorem.

Theorem 4.3: *For any positive integers* $m_1, m_2 > 1$, *let* A_1 *be an* m_1*-state AFA and* A_2 *be an* m_2*-state AFA. Then at most* $2^{m_1} + m_2$ *states are sufficient in the worst case for an AFA A to accept the language* $\overline{L_1}.\overline{L_2}$.

5. SHUFFLE OF COMPLEMENT

5.1. Shuffle

This section first defines the shuffle operation then it establishes a sharp upper bound for the shuffle of complement of two regular languages represented by two AFAs.

Given two strings *x* and *y* the shuffle of *x* and *y*, denoted *x* || *y* is the set of strings obtained by taking the characters of *x* and interleaving them with the characters of y so that the relative order of the characters in each string is maintained.

The shuffle of two languages L_1 and L_2, denoted L_1 || L_2 is defined as

$$L_1 \Big\| L_2 = \bigcup_{\substack{x\in L_1\\ y\in L_2}} x \Big\| y$$

5.2. Shuffle of Complement

This section considers the state complexity of the shuffle of complement operation. That is the combination that includes first the complement of two regular languages and then the shuffle of the resulting languages. It gives the upper bound for the shuffle of complement of two regular languages.

Let $L_i = L(A_i)$ and Ai be an AFA of m_i states, $i = 1, 2$, then $\overline{L_1} \| \overline{L_2}$ is accepted by an AFA A with $m_1 + m_2 + 1$ states.

Let $A_1 = (Q_1, \Sigma_1, s_1, F_1, g')$ be an AFA accepting a regular language L_1 with m_1 states, $A_2 = (Q_2, \Sigma_2, s_2, F_2, g'')$ be an AFA accepting a regular language L_2 with m_2 states then an AFA A accepts the shuffle of complement of these two regular languages is defined as follows:

$A = (Q, \Sigma, s, F, g)$ be an AFA where
$Q = Q_1 U Q_2 U \{s\}$
$\Sigma = \Sigma_1 U \Sigma_2$

$$F = \begin{cases} \{Q_1 - F_1\} \cup \{Q_2 - F_2\} \cup \{s_1 \cup s_2\} & \text{if } s_1 \in F, s_2 \in F_2 \\ \{Q_1 - F_1\} \cup \{Q_2 - F_2\} \cup \{s_1\} & \text{if } s_1 \in F_1 \\ \{Q_1 - F_1\} \cup \{Q_1 - F_2\} \cup \{s_2\} & \text{if } s_2 \in F_2 \\ \{Q_1 - F_1\} \cup \{Q_2 - F_2\} & \text{if otherwise} \end{cases}$$

and g is defined as

$$g_q(a,u) = \begin{cases} \overline{g_{s_1}'(a,u) \wedge s_2} & \text{if } q = s, a \in \Sigma_1 \\ \overline{s_1 \wedge g_{s_2}''(a,u)} & \text{if } q = s, a \in \Sigma_2 \\ \overline{g_q'(a,u)} & \text{if } q \in Q', a \in \Sigma_1 \\ \overline{g_q''(a,u)} & \text{if } q \in Q'', a \in \Sigma_2 \\ q & \text{if } q \in Q', a \in \Sigma_2 \end{cases}$$

where $\overline{g}$ is the dual of g.

The construction of A is evident that it accepts the shuffle of complement of two regular languages L_1 and L_2 using not more than $m_1 + m_2 + 1$ states.

The computation of A begins by simulating both the computations of the machines A_1 and A_2. If $s_1 \notin F_1$ and $s_2 \notin F_2$ then the set of accepting states F has been chosen to be $\{Q_1\text{-}F_1\} U \{Q_2\text{-}F_2\}$. Using the transition rules of an AFA A, the machine A accepts the shuffle of complement of two regular languages L_1 and L_2. It is easy to verify that $L = \overline{L_1} \| \overline{L_2}$. In order to obtain an upper bound for the operation shuffle of complement, count the number of states of A. The $|Q_i| = m_i$, $i = 1, 2$, according to the construction, the cardinality of Q is $m_1 + m_2 + 1$.

Therefore we have the following theorem.

Theorem

Let A_i be an AFA with m_i states $i = 1,2$ accepts the language $L(A_i)$ and $m_i > 1$ then there exists an AFA A with $m_1 + m_2 + 1$ states which accepts the shuffle of complement of two regular languages $L(A_i)$.

Proof

Let L_i be a regular language recognized by an AFA A_i of size m_i, $i = 1, 2$ and $L_i = L(A_i)$ ie., $A_1 = (Q_1, \Sigma_1, s_1, F_1, g')$ be an AFA with m_1 states accepting the language L_1 and $A_2 = (Q_2, \Sigma_2, s_2, F_2, g'')$ be an AFA with m_2 states accepting the language L_2. Assume that Q_1 and Q_2 are disjoint.

Claim: $L(A) = \overline{L_1} \| \overline{L_2}$

Let $w \in L(A)$, $w \in (\Sigma_1 \cup \Sigma_2)^*$

Since $w \in L(A)$, $g_s(w, u) = 1$

By the construction of machine A, it is clear that $g_s(w,u) = \overline{g_{s_1}'(\alpha, u|_{Q_1 - F_1})} \wedge \overline{g_{s_2}''(\beta, u|_{Q_2 - F_2})}$, where α is the collection of alphabets which are in Σ_1 for the word w and β is a collection of alphabets which are in Σ_2 for the word w.

$$\because g_s(w,u) = 1$$

$$\Rightarrow \overline{g_{s_1}'(\alpha, u|_{Q_1 - F_1})} \wedge \overline{g_{s_2}''(\beta, u|_{Q_2 - F_2})} = 1$$

$$\Rightarrow \overline{g_{s_1}'(\alpha, u|_{Q_1 - F_1})} = 1 \text{ and } \overline{g_{s_2}''(\beta, u|_{Q_2 - F_2})} = 1$$

$$\Rightarrow g_{s_1}'(\alpha, u|_{Q_1 - F_1}) = 0 \text{ and } g_{s_2}'(\beta, u|_{Q_2 - F_2}) = 0$$

$$\Rightarrow \alpha \in \overline{L(A_1)}, \beta \in \overline{L(A_2)}$$

$$g_s(w,u) = 1 \text{ iff } \alpha \in \overline{L(A_1)}, \beta \in \overline{L(A_2)} \text{ and}$$

$$w = \alpha \| \beta \text{ iff } w \in \overline{L(A_1)} \| \overline{L(A_2)}$$

$$w \in L(A) \text{ iff } w \in \overline{L(A_1)} \| \overline{L(A_2)}$$

$$ie., L(A) = \overline{L(A_1)} \| \overline{L(A_2)}$$

Therefore by the construction of the machine A, it is clear that the machine A accepts the shuffle of complement of two regular languages $L(A_1)$ and $L(A_2)$ with $m_1 + m_2 + 1$ states

6. CONCLUSIONS

In this paper, I studied the state complexity of combined operations for an Alternating Finite Automata which were inspired by the work of Sheng Yu and Salomaa. Implementing combined operations for an AFA are much easier and much efficient compare to the corresponding DFA operations. Alternation adds perfect features to automata expressiveness, parallelism and succinctness which have practical applications in software system and has the potential to answer several open problems in formal languages and complexity. It remains to investigate the other combined operations for an AFA.

REFERENCES

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

[2] A. Fellah, H. Jurgensen, S. Yu, "Constructions for Alternating Finite Automata", *International Journal of Computer Mathematics*, **35**, 1990, 117-132.

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

[4] . . . . . Languages", in: G. Rozenberg, A. Salomaa (Eds), Handbook of Formal Languages Vol. **1**, Springer, Berlin, New York, 2, 1997, 41-110.

[5] C. Campeanu, K. CulikII, K. Salomaa, S. Yu, "State Complexity of Basic Operations on .nite Languages", in: O. Boldt, H. Jurgensen (Eds.), Proc. Fourth Internat. Workshop on Implementing Automata (WIA'99), Lecture Notes in Computer Science, Springer, Heidelberg, 2214, 2001, 60-70.

[6] G. Pighizzini , "Unary Language Concatenation and its States Complexity", in: S. Yu, A. Paun(Eds.), Implementation and Application of Automata: Fifth Internat. Conference, CIAA 2000, Lecture Notes in Computer Science, Springer, Heidelberg, 2088, 2001, 252-262.

[7] S. Yu , "State Complexity of Regular Languages", *Journal of Automata Languages and Combinatorics*, **6**, 2001, 221-234.

[8] J. Hromkovic, "Descriptional Complexity of Finite Automata: Concepts and Open Problems", *J. Automat. Lang. Comb*, **7**, 2002, 519-531.

[9] Galina Jiraskova, "State Complexity of Some Operations on Binary Regular Languages", *Theoretical Computer Science*, **330**, 2005, 287-298.

[10] Markus Holzer, Martin Kutrib, "On the Descriptional Complexity of Finite Automata with Modified Acceptance Condition , *Theoretical Computer Science*, **330**, 2005, 267-285.

[11] S. Yu, "State Complexity of Finite and Infinite Regular Languages", Bull. EATCS , **76**, 2002, 142-152.

[12] J. Kavitha, L. Jeganathan, G. Sethuraman, "Descriptional Complexity of Alternating Finite Automata", *DCFS*, 2006, 188-198.

[13] J. C. Birget," Intersection and Union of Regular Languages and State Complexity", *Inform. Process. Lett.*, **43**, 1992, 185-190.

[14] C. Campeanu, K. Salomaa, S. Yu, "Tight Lower Bound for the State Complexity of Shuffle of Regular Languages", *Journal of Automata Languages and Combinatorics*, **7**, 2002, 303-310.

[15] M. Holzer, M. Kutrib, "Unary Language Operations and their Nondeterministic State Complexity", *Lecture Notes in Computer Science*, 2450, 2003, 162-172.

[16] M. Holzer, M. Kutrib, "State Complexity of Basic Operations on Nondeterministic Automata", *Lecture Notes in Computer Science*, 2608, 2002, 148 -157.

[17] Y. Gao, K. Salomaa, S. Yu , "State Complexity of Catenation and Reversal Combined with Star" , *DCFS* 2006, 153-164.

[18] Arto Salomaa, Kai Salomaa, Sheng Yu, "State Complexity of Combined Operations", *Theoretical Computer Science*, **383**, 2007, 140-152.

[19] Yo-Sub Han and Kai Salomaa, "State Complexity of Basic Operations on Suffix-free Languages", *Lecture Notes in Computer Science*, **4708**, 2007, 501-512.

[20] Kai Salomaa, "Descriptional Complexity of Nondeterministic .nite Automata", *Lecture Notes in Computer Science*, **4588**, 2007, 31-35.

[21] 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.