International Journal of Computer Theory and Engineering, Vol. 1, No. 5, December, 2009
1793-8201

State Complexity of Shuffle and Intersection Combined with Complement

Kavitha Joseph, Member, CSI

***Abstract* - This paper investigates the state complexity of combined operations on regular languages. In particular it investigates the state complexity of shuffle of complement and the intersection of complement of two regular languages represented by alternating finite automata.**

***Index Terms*— Alternating finite automata, regular languages, shuffle, state complexity.**

I. 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 regular languages 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] and [11]. The nondeterministic state complexity of regular languages operations was studied by Holzer and Kutrib in [10] and [16].The state complexity of some of the operations like concatenation, complement, star and 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. 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] and [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 *logn*-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 shuffle for an AFA are much simpler and more efficient to implement than the corresponding DFA operations.

In this paper, I investigate the state complexity of some combined operations on regular languagues represented by an AFA. In the following sections, the paper first reviews the basic definitions and notations and then it proves the results on shuffle of complement and the intersection of complement.

II. 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 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 $A=(Q, \Sigma, s, F, g)$, 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 $B^Q \rightarrow B$ such that

Manuscript received on 4th June 2009.
Kavitha Joseph was with Anna University, Chennai, INDIA. She is now with the Department of Mathematics,
CMR Institute of Technology, Bangalore, INDIA.
(Author e-mail: kavijoseph_cmrit@rediffmail.com).

$g_q(a)(u) = g_q(a, u)$

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 $g_Q : \Sigma \times B^Q \to B^Q$ 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_q for each q$\in Q$. *For* $f \in B^Q$, f_q = *1 iff* $q \in$ F and f is the characteristic vector of F. Extend g to a function of Q into the set of all functions $\Sigma^* \times B^Q \to B$ as follows.

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

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, Σ, 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

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

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

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

III. Opertions 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 shuffle of complement operation. The next section discusses the state complexity of intersection of complement.

A..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

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

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 A_i 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*, Σ, *s*, *F*, *g*) be an AFA where
$Q = Q_1 \bigcup Q_2 \bigcup \{s\}$
$\Sigma = \Sigma_1 \bigcup \Sigma_2$

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

and g is defined as

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

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 $\{Q_1\text{-}F_1\} \bigcup \{Q_2\text{-}F_2\}$. 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 $L(A_i)$, i=1, 2.

Proof.

Let L_i be a regular language recognized by an AFA A_i of size m_i, i = 1, 2 and $L_i = L(A_i)$ i.e., $A_1 = (Q_1, \Sigma_1, s_1, F_1, g')$ 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

$g_s(w,u) = \overline{g'_{s_1}(a, u|_{Q_1-F_1})} \wedge \overline{g''_{s_2}(b, u|_{Q_2-F_2})}$,

where a is the collection of alphabets which are in Σ_1 for the word w and b is a collection of alphabets which are in Σ_2 for the word w.

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

$\Rightarrow \overline{g'_{s_1}(a, u|_{Q_1-F_1})} \wedge \overline{g''_{s_2}(b, u|_{Q_2-F_2})} = 1$

$\Rightarrow \overline{g'_{s_1}(a, u|_{Q_1-F_1})} = 1$ and $\overline{g''_{s_2}(b, u|_{Q_2-F_2})} = 1$

$\Rightarrow g'_{s_1}(a, u|_{Q_1-F_1}) = 0$ and $g'_{s_2}(b, u|_{Q_2-F_2}) = 0$

$\Rightarrow a \in \overline{L(A_1)}$, $b \in \overline{L(A_2)}$.

Therefore, $g_s(w, u) = 1$ iff $a \in \overline{L(A_1)}$, $b \in \overline{L(A_2)}$ and $w = a \,||\, b$ iff $w \in \overline{L(A_1)} \,||\, \overline{L(A_2)}$.

$w \in L(A)$ 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

B. Intersection Of Complement

In this section, I introduce a new state '*s*' to construct an AFA, which accepts the intersection of complement of two regular languages $L(A_1)$ and $L(A_2)$.

Theorem

For any positive integers m_1, m_2 let A be an m_1-state AFA and A_2 be an m_2-state AFA. Then $m_1 + m_2 + 1$ states are sufficient in the worst case for an AFA A to accept the language $\overline{L(A_1)} \cap \overline{L(A_2)}$.

Proof.

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

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

We construct an $m_1 + m_2 + 1$- state AFA $A = (Q, \Sigma, s, F, g)$ such that $L(A) = \overline{L(A_1)} \cap \overline{L(A_2)}$ as follows.

$Q = Q_1 \bigcup Q_2 \bigcup \{s\}$

$F = (Q_1 - F_1) \bigcup (Q_2 - F_2)$.

and g is defined as follows.

$$g_q(a,u) = \begin{cases} \overline{g'_{s_1}(a,u)} \wedge \overline{g''_{s_2}(a,u)} & \text{if } q = s, a \in \Sigma \\ \overline{g'_q(a,u)} & \text{if } q \in Q', a \in \Sigma \\ \overline{g''_q(a,u)} & \text{if } q \in Q'', a \in \Sigma \end{cases}$$

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

First we prove $\overline{L(A_1)} \cap \overline{L(A_2)} \subseteq L(A)$.

Consider a word $w \in \overline{L(A_1)} \cap \overline{L(A_2)}$

Then $w \in \overline{L(A_1)}$ and $w \in \overline{L(A_2)}$.

Star with $g_s(w, u) = 1$ for the machine A.

If q = s,

$g_s(w, u) = \overline{g'_{s_1}(w, u|_{Q_1-F_1})} \wedge \overline{g''_{s_2}(w, u|_{Q_2-F_2})}$.

Since $w \in \overline{L(A_1)}$, $\overline{g'_{s_1}(w, u|_{Q_1-F_1})} = 1$ and $w \in \overline{L(A_2)}$, $\overline{g''_{s_2}(w, u|_{Q_2-F_2})} = 1$.

$\overline{g'_{s_1}(w, u|_{Q_1-F_1})} \wedge \overline{g''_{s_2}(w, u|_{Q_2-F_2})} = 1$.

By the construction of the machine A

$g_s(w, u) = \overline{g'_{s_1}(w, u|_{Q_1-F_1})} \wedge \overline{g''_{s_2}(w, u|_{Q_2-F_2})}$

$= 1 \wedge 1$

$= 1$.

Therefore, $w \in L(A)$

ie., $\overline{L(A_1)} \cap \overline{L(A_2)} \subseteq L(A)$.

Conversely, let $w \in L(A) \Rightarrow g_s(w, u) = 1$.

ie., $\overline{g'_{s_1}(w, u|_{Q_1-F_1})} \wedge \overline{g''_{s_2}(w, u|_{Q_2-F_2})} = 1$.

$\Rightarrow \overline{g'_{s_1}(w, u|_{Q_1-F_1})} = 1$ and $\overline{g''_{s_2}(w, u|_{Q_2-F_2})} = 1$.

Therefore, $w \in \overline{L(A_1)}$ and $w \in \overline{L(A_2)}$,

That is $w \in \overline{L(A_1)} \cap \overline{L(A_2)}$.

$L(A) \subseteq \overline{L(A_1)} \cap \overline{L(A_2)}$.

Thus $L(A) = \overline{L(A_1)} \cap \overline{L(A_2)}$.

IV. Conclusion

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, pp. 114 - 133.
[2] A.Fellah, H.Jurgensen, S.Yu, Constructions for alternating finite automata, International journal of computer mathematics 35, 1990, *pp*.117 - 132.
[3] K.Salomaa, S.Yu, Q.Zhuang , The state complexity of some basic operations on regular languages, Theoretical computer science 125, 1994, *pp*. 315 - 328.
[4] S.Yu, Regular languages, in: G.Rozenberg, A.Salomaa(Eds), Handbook of Formal Languages Vol.1, Springer, Berlin, New york, 2,1997, *pp*. 41 - 110.
[5] C. Campeanu, K.CulikII, K.Salomaa, S.Yu, State complexity of basic operations on finite 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, *pp*. 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, *pp*. 252 - 262.
[7] S.Yu , State complexity of regular languages, Journal of automata languages and combinatorics 6, 2001, *pp*.221 - 234*:*
[8] J.Hromkovic, Descriptional complexity of finite automata: concepts and open problems. J.Automat.Lang.Comb. 7, 2002, *pp*. 519 – 531.
[9] Galina Jiraskova , State complexity of some operations on binary regular languages , Theoretical computer science 330, 2005, *pp*. 287 - 298*:*
[10] Markus Holzer, Martin Kutrib, On the descriptional complexity of finite automata with modified acceptance condition, Theoretical computer science, 330, 2005, *pp*. 267 - 285.
[11] S.Yu, State complexity of finite and infinite regular languages Bull.EATCS, 76, *2002*, *pp*. 142 -152.
[12] J.Kavitha, L. Jeganathan, G. Sethuraman, Descriptional complexity of Alternating finite automata, DCFS 2006, *pp*.188 -198.
[13] J. C. Birget, Intersection and union of regular languages and state complexity, Inform. Process. Lett. 43, 1992, *pp*. 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, *pp*. 303 - 310.
[15] M. Holzer, M. Kutrib, Unary language operations and their nondeterministic state complexity, Lecture Notes in computer Science 2450, *2003*, *pp*. 162 - 172.
[16] M. Holzer, M. Kutrib, State complexity of basic operations on nondeterministic automata, Lecture Notes in computer Science 2608, 2002, *pp*. 148 - 157.
[17] Y. Gao, K. Salomaa, S. Yu, State complexity of catenation and reversal combined with star, DCFS 2006, *pp*. 153 -164.
[18] Arto Salomaa, Kai salomaa, sheng Yu, State complexity of combined operations, Theoretical computer science 383, *2007*,*pp*. 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*, *pp*. 501 - 512.
[20] Kai Salomaa, Descriptional complexity of nondeterministic finite automata, Lecture Notes in computer Science 4588, 2007, *pp*. 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, *pp*.1178 - 1186.

Kavitha Joseph is an Assistant Professor in the Department of Mathematics at CMR Institute of Technology, Bangalore, INDIA. She received her Ph. D in Mathematics from Anna University, Chennai. Before joining CMR Institute of Technology, she was a teaching research associate in Madras Institute of Technology, Anna University. She has 10 years teaching experience and 8 years research experience in the field of DNA computing. theoretical computer science. Her research interests include automata theory, DNA computing, parallel computing and graph theory. She is a member of computer society of India