International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 22 (2017) pp. 11960-11964

New Heuristics for Obtaining Smaller Regular Expressions from Deterministic Finite Automata

[1]Kulwinder Singh, [2]Ajay Kumar and [3]Sunita Garhwal
[1]FSADM3, Infosys, Rajiv Gandhi Technology Park, Chandigarh, 160017, Punjab, India.
[2]Department of Computer Science and Engineering, Thapar University, Patiala 147004, Punjab, India.
[3]Department of Computer Science and Engineering, Thapar University, Patiala 147004, Punjab, India.

[1,2,3]Orcid: 0000-0001-7166-5497, 0000-0002-4452-4725 ,0000-0002-8959-3724

Abstract

State elimination method is a widely used efficient approach for the conversion of deterministic finite automata to regular expressions. Minimal regular expressions corresponding to deterministic finite automata is NP-complete. The paper aims to generate smaller regular expression from the deterministic finite automaton. New heuristics have been proposed for obtaining smaller regular expressions equivalent to deterministic finite automata using state elimination method. In this paper, heuristics are proposed using the structural properties of deterministic finite automaton, which leads to a smaller regular expression corresponding to a deterministic finite automaton.

Keywords: finite automata, regular expression, state elimination, heuristics.

INTRODUCTION

Regular expressions are used in compiler design, text editor, search for an email- address, grep filter of UNIX, context switching and in many areas of computer science [1-7]. Regular language can be expressed by regular expressions or finite automata. Kleene [8] proved that there exists an equivalent regular expressions corresponding to deterministic finite automata. Recently, the problem of obtaining smaller regular expression equivalent to deterministic finite automaton has been intensively investigated. For carry out this conversion, Kleene [8] proposed the first mathematical technique, known as transitive closure approach. Brzozowski [9] extended transitive closure approach, called Brzozowski algebraic approach. Brzozowski and McCluskey [10] introduced the concept of state elimination method for the conversion of deterministic finite automata to regular expressions. In state elimination method states are removed one by one except the starting and final state of deterministic finite automata. Neumann [11] carried out the comparisons of these proposed approaches and found that the transitive closure method is complex because it generates large regular expression as compared to Brzozowski algebraic method and state elimination method. Neumann [11] also found that Brzozowski method takes more time as compared to state elimination method. State elimination method leads to simpler computations and shorter regular expression. In state elimination method, different removal sequence gives different regular expression. We need to choose optimal removal sequence of states for obtaining smaller regular expression. Several heuristics have been proposed by various researchers [12-16] for choosing optimal sequence.

In this paper, we analyze the heuristics proposed by researchers, for choosing optimal removal sequence of states and propose new heuristics for the same. The paper is organized as follow: In section 2, some basic definitions and notations are reviewed. Section 3 gives some related work and describes state elimination ordering heuristics. In section 4, new heuristics are designed for choosing optimal removal sequence in state elimination method, and finally, section 5 includes conclusions and future scope.

BASIC DEFINITIONS AND NOTATIONS

An Alphabet is a finite non-empty set of symbols on which the language is defined. The alphabet is denoted by $\sum$. A language [17] is defined as a subset of $\sum^*$. Empty string and null language are denoted by ε and ϕ respectively. Various kinds of formal languages can be classified as regular, context-free, context-sensitive and recursive language. Regular language can be described by regular expression or finite automaton (Deterministic or Non-deterministic).

A regular expression (RE) [3] is a pattern that describes some set of strings. Regular expression for each alphabet will be represented by itself. The empty string (ε) and null language (ϕ) are regular expressions denoting the regular language $\{\varepsilon\}$ and $\{\phi\}$ respectively. If E and F are regular expressions, then $E+F, EF$ and E^* are the regular expressions denoting the regular language $L(E+F), L(EF)$ and $L(E^*)$ respectively.

Deterministic finite automata (DFA) [3] can be defined by 5-tuples $(Q, \sum, \delta, q_0, F)$ where Q is a finite set of states, $\sum$ is a finite set of symbols, δ is the transition function

$\delta : Q \times \Sigma \to Q$, q_0 is the starting state and F is a subset of Q called accepting states.

A Non-deterministic finite automaton (NFA) [3] is a same as DFA except the transition function. Transition function in NFA is defined as $Q \times \Sigma \to 2^Q$.

RELATED WORK

The state removal approach [11, 16] is widely used for converting DFA to a regular expression. In this approach states of DFA are removed one by one until we left with only starting and final state, for each removed state regular expression is generated.

Example 1: DFA, shown in figure 1, is converted to regular expression using state elimination method.

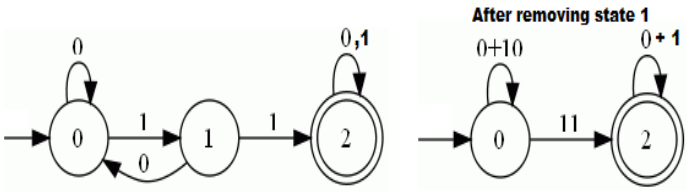

Figure 1: An example of state elimination method

Equivalent regular expression corresponding to given DFA is $(0+10)^*11(0+1)^*$. Using different removal sequences of states, we obtain different regular expressions for the same language. We require optimal removal sequence that gives smaller regular expression.

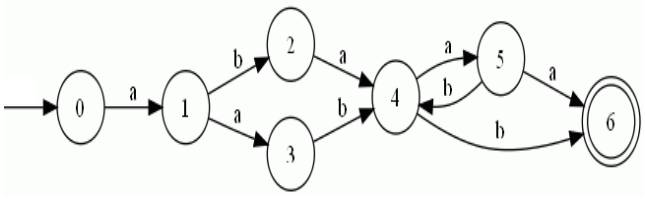

Figure 2: Different removal sequence will generate different regular expression

A regular expression R_1=a(ba+ab)(ab)*(b+aa) is obtained using state elimination sequence 2-3-5-1-4 .Similarly, regular expression R_2=(ab(ab+aa(ba)*(a+bb))+aa (bb+ba(ba)*(a+bb))) and R_3=[(aba+aab)b+(aba+aab) a(ba)*(a+bb)] are obtained using state elimination sequence 1-4-5-2-3 and 2-1-3-4-5 respectively. For n-state DFA excluding starting and final states, $n!$ removal sequences are possible in state elimination method. It is difficult to try all the possible removal sequence for obtaining smaller regular expression. Using structural properties of deterministic finite automata, researchers [12-16] introduced new heuristics for state elimination method that can give smaller regular expression equivalent to DFA. Han and wood [18] proposed the concept of bridge state for state elimination method. They found that removal of all non-bridge states before all the bridge states gives smaller regular expression.

In figure 2, state 1 and 4 satisfy the bridge state criterion. Regular expression R_1=a(ba+ab)(ab)*(aa+b) obtained if bridge states are removed after removing all non-bridge states. A regular expression R_2=(ab(ab+aa(ba)*(a+bb)) + aa (bb+ba(ba)* (a+bb))) is obtained if bridge states are removed before removing non-bridge states of deterministic finite automata and $|R_1| < |R_2|$.

Delgado and Morais's [19] proposed the concept of state weight in state elimination method. In each step of state elimination, weight of each state of deterministic finite automata is calculated using indegree, out degree and loop of the states.

Using Delgado and Morais [19] heuristic, we found the weights of different states of DFA given in figure 3.

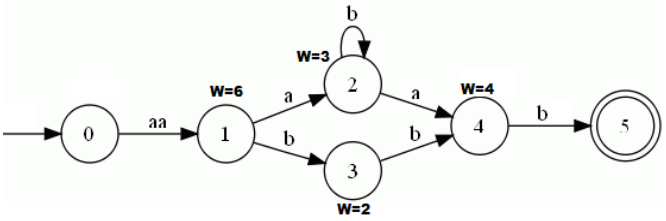

Figure 3: Weight Calculation using Delgado and Morais's formula

State 3 is eliminated first as it has the lowest weight. After removal of each state, weights of the states are calculated again. A regular expression aa(bb+a(b)*a)b is generated using Delgado and Morais's approach.

Sometimes removal sequence chosen by Delgado and Morais [19] heuristics give us larger regular expression as compared to other optimal removal sequence. Using Delgado and Morais heuristic regular expression $R_1 = a(ab)^* aa(aa + b(ab)^* aa)^* ab$ is obtained corresponding to DFA (in figure 4).

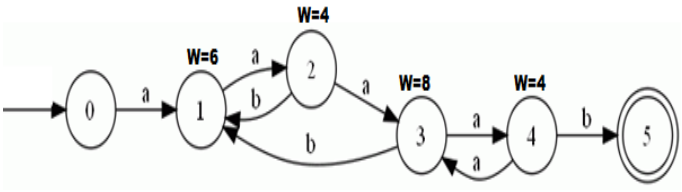

Figure 4: An example of removal sequence by Delgado and Morais heuristic gives larger RE

Regular expression $R_2 = aa(b+a(aa)^*b)^*aa(aa)^*ba$ is obtained if the states are removed in sequence of $4-3-2-1$ and $|R_1| > |R_2|$. Delgado and Morais heuristic gives the optimal removal sequence for obtaining smaller regular expression.

New Heuristics For Choosing Optimal Removal Sequence Of States In State Elimination Method

Cycle of a DFA: We call a cycle exists in a DFA A if it satisfies following conditions:

i. The sub-string x processed by the cycle is a sub-string of $y \in L(A)$.

ii. x may or may not belong to $L(A)$.

iii. The path traversed by x is such that it will visit the previously visited state q_v again, and Q' contains all these states that are traversed during processing of x from the first q_v to the second q_v.

iv. $Q' \subseteq Q$.

Sub-cycle of a Cycle in DFA: Any cycle with a set of states Q'' is called a sub-cycle of a cycle C with a set of states Q' if it satisfies following conditions:

i. Q'' satisfies the properties of a cycle of a DFA.

ii. $Q'' \subset Q'$.

Based on the structural properties of DFA sub-cycle of a cycle present in DFA can be found. For example, DFA shown below in figure 5 has two cycles $1 \to 2 \to 3 \to 4 \to 1$, and $2 \to 3 \to 2$ and these cycles satisfy all the conditions of a cycle. $2 \to 3 \to 2$ is sub-cycle of outer cycle $1 \to 2 \to 3 \to 4 \to 1$.

If cycle C is present in a DFA and any state present in that cycle has self-loop, then the self-loop is considered as sub-cycle of cycle ' C '.

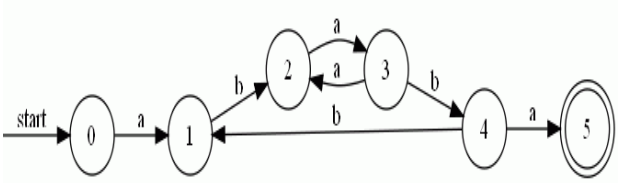

Figure. 5: An example of the cycle and sub-cycle

New Heuristics for choosing optimal removal sequence

If more than one, cycles are presented in a DFA, then the general procedure is to remove un-common states (which are not present in more than one, cycles) before common states (which are part of more than one, cycles). Based upon the structural properties of DFA following different cases are possible:

i. DFA is having a cycle and its sub-cycle with same starting state.

ii. DFA is having a cycle and its sub-cycle with different starting state. Following sub-cases can exist.

a) DFA has a direct transition from starting state of outer-cycle to the final state of the DFA.

b) DFA has no direct transition from starting state of outer-cycle to final state of the DFA.

Heuristic for DFA having cycle and sub-cycle with same starting state

If cycle and its sub-cycles are presented in a DFA with same starting state, then in each step of state elimination, common states are removed after removing all the un-common states, to obtain smaller regular expression. Removal sequence of un-common and common states should be chosen to use state weight heuristics. Figure 6 gives an example of DFA having a cycle and its sub-cycle with same starting state. Cycle $2 \to 3 \to 4 \to 2$ is a sub-cycle of $2 \to 3 \to 4 \to 5 \to 6 \to 2$ and state 2 is starting state of these cycles.

States $\{2,3,4\}$ are common states of cycle $2 \to 3 \to 4 \to 5 \to 6 \to 2$ and its sub-cycle $2 \to 3 \to 4 \to 2$. States $\{5,6\}$ are un-common states of these cycles. State 2 is common between cycles $1 \to 2 \to 1$ and $2 \to 3 \to 4 \to 2$. It is also common in between cycles $2 \to 3 \to 4 \to 5 \to 6 \to 2$ and $1 \to 2 \to 1$. State 1 of cycle $1 \to 2 \to 1$ satisfies all the conditions of bridge state. State 1 is a un-common state, so it should be deleted before non-bridge states of DFA for obtaining smaller regular expression.

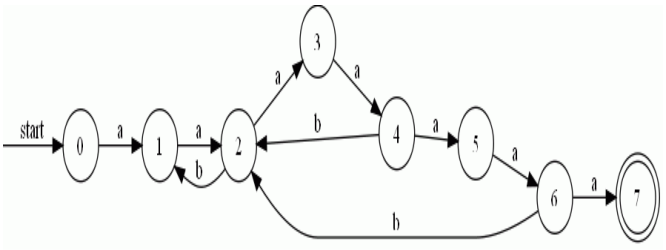

Figure 6: Another example of the cycle and its sub-cycle with same starting state

If un-common states of cycles are removed before common states, using removal sequence $1-5-6-3-4-2$, Regular expression $R_1 = aa(ba+aa(b+aab))^*aaaaa$ is obtained equivalent to DFA. This example demonstrates that the bridge

state removal before the non-bridge state will give us smaller regular expression.

If common states are removed before un-common states using removal sequences $1-3-4-5-6-2$ and $2-3-4-5-6-1$ then regular expressions $R_2 = aa(ba+aab+aaaab)^* aaaaa$ an $R_3 = (a((ab+aaa((baa)^*)bb+aaa((baa)^*)aa((baa((baa)^*)aa)^*)(bb+baa((baa)^*)bb))^*)aaa((baa)^*)aa((baa((baa)^*aa)))$ are obtained respectively. If all the un-common states are removed before common states of cycles and bridge state is removed at last. Using the removal sequence $5-6-3-4-2-1$, then regular expression $R_4 = aa(aa(b+aab))^* ba(aa(b+aab))^*(aaaaa)$ is obtained and $|R_1| \lessdot |R_2|$, $|R_1| \lessdot |R_3|$ and $|R_1| \lessdot |R_4|$. Hence in this case bridge state should be removed before non-bridge states for obtaining smaller regular expression.

Cycle and sub-cycle with different starting state

If cycle and its sub-cycle, present in deterministic finite automata have different starting states then following two cases must be considered.

1. DFA having direct transition from starting state of outer cycle to the final state of DFA

If cycle and its sub-cycle with different starting state are presented in a DFA, and the direct transition is presenting from the starting state of outer cycle 'C' to the final state of DFA. In each step of the state elimination, all other un-common states of the cycles except starting state of cycle 'C' should be removed before removing common states of these cycles. For example, following figure 7 shows a DFA having cycle 1→2→3→4→1 and it's sub-cycle 2→3→2. There is direct transition from starting state 1 of cycle $1 \to 2 \to 3 \to 4 \to 1$ to the final state 5 of DFA.

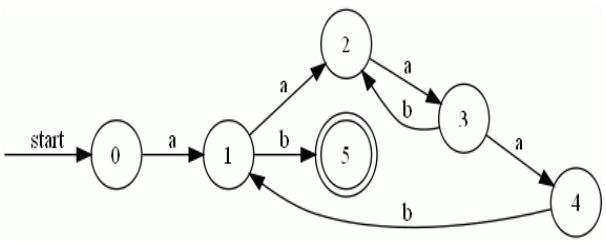

Figure 7: DFA with the direct transition from starting state of the outer cycle to the final state

States $\{2,3\}$ are common states of the cycles and states $\{1,4\}$ are un-common states. State 1 is a bridge state. In this case, bridge state or starting state of outer cycle 'state 1' should be removed after removing all other un-common states then common states of the cycles. State 4 is the un-common state of cycles, so it should be removed first. Regular expression $R_1 = a((aa((ba)^*)ab)^*)b$ is obtained using removal sequence $4-2-3-1$. In this case, the first removal of un-common states followed by common states elimination of cycles using removal sequence $1-4-3-2$, without considering the direct path from starting state of outer-cycle to the final state of DFA, gives regular expression $R_2 = ab + aa(a(b+aba))^* aabb$. Now $|R_2| > |R_1|$, so in this case starting state of the outer cycle should be removed after removing all other states of cycles.

2. DFA having no direct transition from starting state of outer-cycle to the final state

If cycle and its sub-cycle with different starting states are presented in a DFA, and no direct transition is presenting from the starting state of outer-cycle to the final state of DFA. Then rule for choosing removal sequence of states is same as a rule for DFA having a cycle and sub-cycle with same starting state, i.e. removal of all the un-common states before removing common states. For example, figure 8 shows DFA having cycles $1 \to 2 \to 3 \to 4 \to 1$ and its sub-cycle $2 \to 3 \to 2$ and no direct transition from starting state 1 of cycle $1 \to 2 \to 3 \to 4 \to 1$ to the final state of DFA.

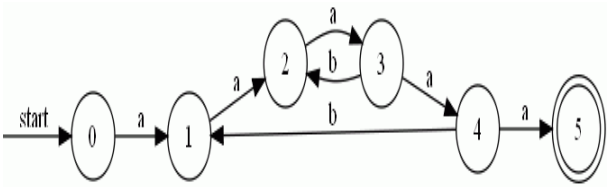

Figure 8: DFA without any direct transition from starting state of the outer cycle to the final state

State 1 is bridge state of given DFA, present in cycle $1 \to 2 \to 3 \to 4 \to 1$ and states $\{2,3\}$ are common states of cycles. States $\{1,4\}$ are un-common states of these cycles. To obtain smaller regular expression equivalent to DFA, state 1 and state 4 should be removed before common states of the cycles. Regular expression $R_1 = aa(a(b+aba))^* aaa$ is obtained using state removal sequence $1-4-3-2$.

Removing common states of cycles, present in the DFA shown in figure 8, before removing all un-common states using removal sequence $4-2-3-1$ gives regular expression $R_2 = a(aa(ba)^* ab)^* aa(ba)^* aa$ and $|R_1| \lessdot |R_2|$. In this case,

also, bridge state should be removed before non-bridge states for obtaining smaller regular expression.

CONCLUSION

In state elimination method, for a DFA having n states excluding starting and final state, $n!$ removal sequences are possible, and the different removal sequences of states give us different regular expression equivalent to same DFA. Using structural properties of given DFA, smaller regular expression can be generated by choosing optimal removal sequence of states in state elimination method. Based on structural properties of DFA, new heuristics are proposed in this paper using the concept of sub-cycle and cycle presented in the DFA. In general, removal of uncommon states before common states gives smaller regular expression. The removal of bridge state after removing all non-bridge states does not always give smaller regular expression equivalent to DFA. In some cases, removal of bridge state before non-bridge state gives smaller regular expression.

Conversion of DFA to RE is an NP-complete problem. Newly designed heuristics can be validated on some real-time applications. Still, there is a scope for generating smaller RE equivalent to a DFA. New heuristics can be designed to obtain smaller regular expression.

REFERENCES

[1] H. Hosoya, "Regular expression pattern matching - a simpler design", Technical Report 1397, RIMS, Kyoto University, 2003.

[2] H. Larkin, "Object-oriented regular expressions", 8th IEEE International Conference on Computer and Information Technology, 8-11 July 2008, pp. 491-496.

[3] J. E. Hopcroft, R. Motwani and J. D. Ullman, "Introduction to Automata Theory, Languages, and Computation", Pearson Education Inc, ISBN 0-201-44124-1. Addison Wesley, 2001.

[4] A. Kumar, "Design and Development of Algorithms for Converting Parallel Regular Expressions to Deterministic Finite Automata", PhD Thesis, Thapar University, 2013.

[5] N. Kalra and A. Kumar, "State grammar and deep pushdown automata for biological sequences of nucleic acids", Current Bioinformatics, vol. 11(4), pp. 470-479, 2016.

[6] S. Garhwal and R. Jiwari, "Parallel Fuzzy Regular Expression and its Conversion to Epsilon-Free Fuzzy Automaton", The Computer Journal, vol. 59(9), pp. 1383-1391, 2016.

[7] R. Kumar and A. Kumar, "Metamorphosis of fuzzy regular expressions to fuzzy automata using the follow automata", arXiv:1411.2865, 2014.

[8] S. C. Kleene, "Representation of events in nerve nets and finite automata", In C. E. Shannon & J. McCarthy, editors: Automata studies, Annals of mathematics studies 34, Princeton University Press, pp. 2-42, 1956.

[9] J. A. Brzozowski, "Derivatives of regular expressions", Journal of ACM, 11(4), pp. 481-494, 1964.

[10] J. A. Brzozowski and E. Jr. McCluskey, "Signal flow graph techniques for sequential circuit state diagrams", IEEE Transactions on Electronic Computers EC-12, pp. 67-76, 1963.

[11] C. Neumann, "Converting Deterministic Finite Automata to Regular Expressions", http://neumannhaus.com/christoph/ papers 2005-03-16.DFA_to_RegEx.pdf, 2005.

[12] H. Gruber and M. Holzer, "Provably shorter regular expressions from deterministic finite automata", LNCS, Springer, Heidelberg vol. 5257, pp. 383–395, 2008.

[13] J. J. Morais, N. Moreira, and R. Reis, "Acyclic automata with easy-to-find short regular expressions", In 10th Conference on Implementation and Application of Automata, volume 3845 of LNCS, Springer , June 2005, pp 349–350.

[14] S. Gulan and H. Fernau, "Local elimination-strategies in automata for shorter regular expressions, In Proceedings of SOFSEM, pp. 46-57, 2008.

[15] Y. S. Han and D. Wood, "Obtaining shorter regular expressions from finite-state automata", Theoretical Computer Science 370(1-3), pp. 110-120, 2007.

[16] K. Singh, "Conversion of deterministic finite automata to regular expression using bridge state", M.E. Thesis, Thapar University, 2011.

[17] Peter Linz, "An introduction to Formal Languages and Automata", Jones and Bartlett Publishers, Sudbury, MA, third edition, 2001.

[18] Y. S. Han and D. Wood, "The generalization of generalized automata: Expression automata", International Journal of Foundations of Computer Science 16 (3), pp. 499-510, 2005.

[19] M. Delgado and J. Morais, "Approximation to the smallest regular expression for a given regular language", In Proceedings of CIAA'04, Lecture Notes in Computer Science, vol. 3317, pp. 312-314, 2004.