Volume 3, No. 4, July- August 2012

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

New heuristics for conversion of Deterministic finite automata to Regular expression

Tamanna Chhabra*
CSED Thapar University
Patiala, India
tmnn_tanu@yahoo.com

Ajay Kuymar
Assistant prof., CSED
Thapar University Patiala, India
ajaykumar@thapar.edu

Abstract: Every deterministic finite automata can be converted into regular expression and vice versa.Various techniques like Transitive closure method, Brzozowski algebraic and State elimination exist for conversion of deteministic finite automata to regular expression. In state elimination method concept of bridge state, horizontal chopping and vertical chopping are used for obtaining smaller regular expressions from deterministic finite automata. In this research paper, new heuristic is discussed using which a smaller regular expression can be generated. Relationship between the size of deterministic finite automata and size of regular expression is estimated..

Keywords: Deterministic finite automata, regular expression ,circles

I. INTRODUCTION

The regular expressions are well known in the field of computer science. They are commonly used in lexical analyzers and text editors [1].

Regular language can be represented by regular expression or finite automata. Various techniques are used for conversion of deterministic finite automata(DFA) to regular expression(RE), but state elimination leads to shorter regular expressions from DFA [5]. For obtaining smaller RE from DFA, concepts of bridge state, horizontal chopping and vertical chopping are used in state elimination method. In this paper a new heuristic is proposed for obtaining smaller regular expression and relation between the size of deterministic automata and size of regular expression is estimated.

II. DEFINITIONS AND NOTATIONS

Regular languages can be described by regular expressions. If r is a regular expression, then the regular language corresponding to r is L(r) [3].

A. Def. 1: A regular expression (RE) [4] is a pattern that describes some set of strings such that:

a. Regular expression for each alphabet is represented by itself.
b. Null language (ϕ) and null string(ε) also represented by themselves.
c. If E and F are regular expressions, then following rules can be applied recursively.
    a) Union of E and F is denoted by E+F .
    b) Concatenation of E and F is denoted by EF.
    c) Kleene closure is denoted by E*.
d. Any regular expression can be formed by using 1-2 rules only.

Example 1: The given regular expression (a+b)* denotes the set of all strings {ε, a, b,aa,bb,ab,ba,aaa, bbb, ...}.

B. Def. 2: A deterministic finite automata (DFA) [4] M is a 5-tuple, $(Q, \Sigma, \delta, q_0, F)$, consisting of a finite set of states(Q),a finite set of input symbols called the alphabet (Σ),a transition function ($\delta : Q \times \Sigma \rightarrow Q$),a start state ($q_0 \in Q$),a set of final states ($F \subseteq Q$). DFA can be represented by transition diagram.

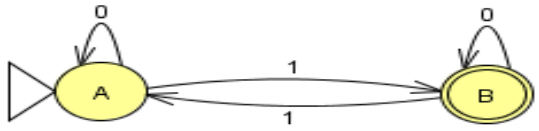

Figure 1: DFA accepting odd number of 1's

C. Def. 3: In state elimination method [2], states of the DFA are eliminated except the start and final state and the edges are replaced with regular expressions that includes the behavior of the eliminated states. At the end, we get a DFA with a start and final state. The advantage of this technique is that it is easier to visualize. State elimination is illustrated in Figure 2.

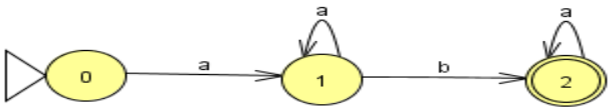

Figure 2: Example of state elimination

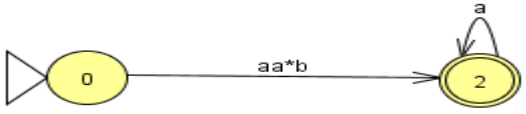

Figure 3: After eliminating state 1

The RE obtained after eliminating state 1 is aa*ba*.

III. PROBLEM OF STATE ELIMINATION

The problem with state elimination is that different removal sequences give different regular expressions corresponding to a deterministic finite automaton. We can

obtain a shorter regular expression by choosing a better removal sequence. Figure 4 illustrates this idea.

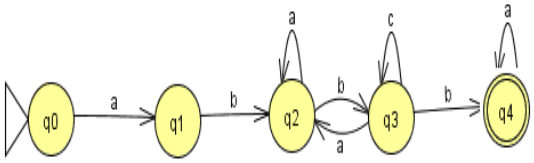

Figure 4: Problem of state elimination

Case 1*:* Removal sequence q1,q2 and q3.

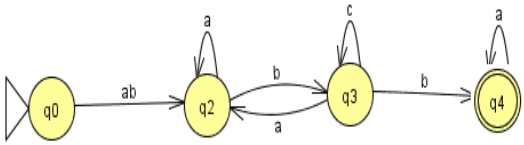

Figure 5: After removing state q1

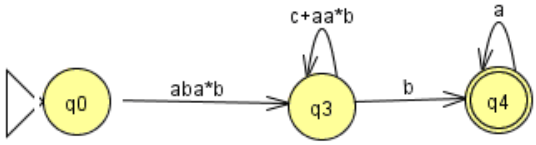

Figure 6: After removing state q2

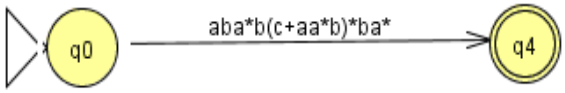

Figure 7: After removing state q3

Regular expression obtained is aba*b(c+aa*b)*ba*

Case 2*:* Removal sequence q3,q1 and q2

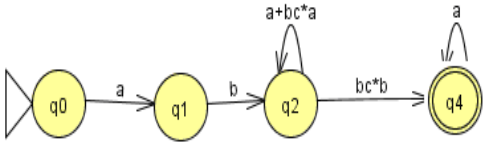

Figure 8: After removing state q3

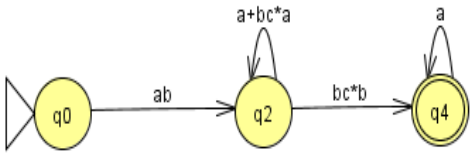

Figure 9: After removing state q1

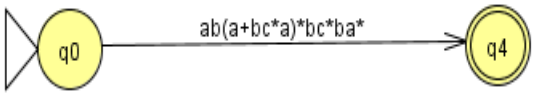

Figure 10: After removing state q2

Regular expression obtained is ab(a+bc*a)*bc*ba

Clearly |RE2|> | RE1|. Hence different removal sequence will generate different regular expressions. We need to choose optimal removal sequence. For chosing optimal removal sequence several heuristics are proposed. One such proposed heuristics is explained in following section.

IV. NEW HEURISTIC FOR CHOOSING OPTIMAL REMOVAL SEQUENCE IN STATE ELIMINATION METHOD

First we find out all the circles in the given DFA. corresponding to each states in the circles, we count the number of circles in which these states are involved. Self loops are also counted as circles. For each state leaving the initial and the final state we count the number of circles in which that state is involved. Eliminate the state, first which are involved in lesser number of circles.

Example 2: In the DFA (given in figure 11), we first count the number of circles. There are two circles 1-2-1 and 1-1. State 1 is involved in two circles, state 2 in one circle and state 3 is not involved in any circle.

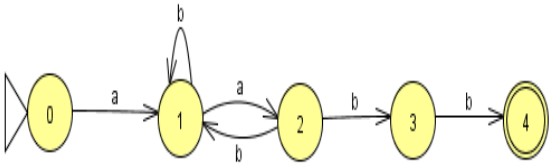

Figure 11: DFA with two circles, one circle and one self loop

Case 1*:* Eliminate those states first which are involved in the minimum number of circles. The removal sequence is 3-2-1

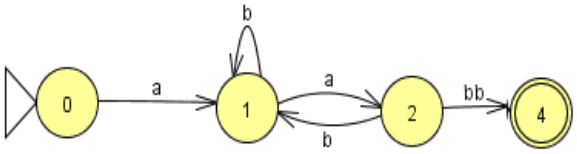

Figure12: After removing state 3

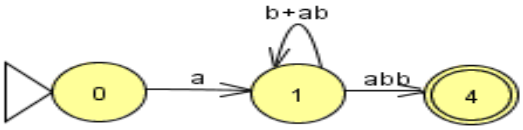

Figure 13: After removing state 2

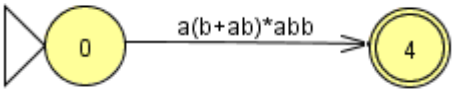

Figure 14: After removing state 1

Regular expression obtained is a(b+ab)*abb.

Case 2*:*- Removing those states first which are involved in the maximum number of circles using the sequence 1-2-3.

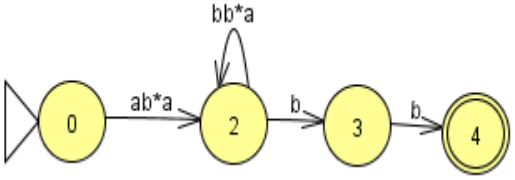

Figure 15: After removing state 1

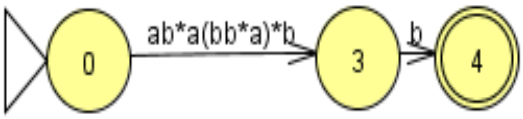

Figure 16: After removing state 2

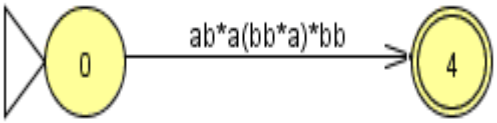

Figure 17: After removing state 3

Regular expression obtained is ab*a(bb*a)*bb.

Clearly |RE2| > | RE1|, Hence, if first we remove the states which are involved in lesser number of circles, we obtain a smaller regular expression.

V. RELATION BETWEEN THE SIZE OF A DFA AND REGULAR EXPRESSION

The size of DFA can be determined by counting the number of states and the edges in the DFA. Size of a regular expression is determined by counting the number of operators and the alphabets. Size of DFA (shown in figure 18) is six and size of regular expression is eight. Moreover, the size of the regular expression increases as the union and the concatenation operators increases. But size of a regular expression increases linearly with the size of a DFA.

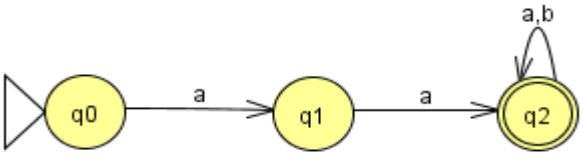

Figure 18: DFA accepting all the strings starting with aa

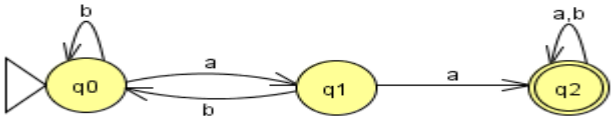

Figure 19: DFA accepting all the strings with substring aa
RE=(b+ab)*aa(a+b)*

Size of DFA is eight and size of regular expression is fifteen.

RE=(b+ab)*aa(a+b)*

Size of DFA is eight and size of RE is fifteen.

RE=(b+ab)*aa(a+b)*

Size of DFA is eight and size of regular expression is fifteen.

VI. CONCLUSION AND FUTURE WORK

In this paper,we have proposed a new heuristic for generating smaller regulae expression from DFA and relation between the size of DFA and RE is estimated.In future,some new heuristics can be proposed for getting minimal regular expression from a given DFA.

VII. REFERENCES

[1]. Alfred V. Aho, "Constructing a Regular Expression from a DFA", Lecture notes in Computer Science Theory, September 27, 2010, Available at http://www.cs.columbia.edu/~aho/cs3261/lectures.

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

[3]. Neumann Christoph, Converting Deterministic Finite Automata to Regular Expressions, Mar 16, 2005

[4]. Peter Linz, 2009 An Introduction to Formal Languages and Automata, Narosa publishers, fourth edition.

[5]. Yo-Sub Han and Derick Wood, Obtaining shorter regular expressions from finite-state automata, Theoretical Computer Science vol. 370(1-3) pages 110-120,2007.