AN ELEGANT TECHNIQUE FOR OBTAINING SHORTER REGULAR EXPRESSION

Dr. O. V. Shanmuga Sundaram
Associate Professor, Dept of PG Mathematics, Sree Saraswathi Thyagaraja College,
Pollachi – 642 107, Tamil Nadu, South India.
ovs3662@gmail.com

Abstract

In this paper, the conversion of an automaton into regular expression is investigated and three types of minimization techniques are discussed. An algorithm has devised for state – splitting procedure which enhances the state elimination procedure. Also this procedure has compared with other techniques of obtaining shorter regular expressions and illustrated with examples.

Keywords: **Regular expressions, NFA, Self–loop, State–splitting.**

1. INTRODUCTION

Many researchers formulated algorithms for construction of various types of automata like Thompson, Position, Partial derivatives, Canonical, Follow and some automata converted into small regular expressions. In the present research topics of Automata theory, the minimization of Finite Automata is burning topic by reducing the size of the automata. Especially, the construction from NFA to regular expressions is now more important topics in Automata Theory and investigations of the making the regular expression to be shorter one.

In the recent, researchers of automata theory discussed the problem of obtaining a shorter regular expressions from a given finite automaton. Reducing an automaton into regular expression, state elimination method or system of linear equations is most commonly used. In 1956, Kleene proved the theorem that the language accepted by an NFA can be represented by the regular expression. In 1960, McNaughton and Yamada developed an algorithm based on the transitive closure of the digraph which computed a regular expression from an NFA. In 1963, state elimination method was used by Brzozowski and McCluskey for conversion of the automaton into regular expression and later Derrick Wood in 1987 made some modification in the state elimination method to get a regular expression very simple. In removing states of the automaton, there was a drawback of selecting a state to eliminate first, because selection of different removing states gave different regular expressions. To overcome this situation by Delgado and Morais introduced the weight of a state concept in their automaton to reduce the size of the automaton. This method took more time for computation of each state weight. In 2007 Han and Wood introduced a bridge state concept in their construction to get a shorter regular expression quickly.

The authors J A Brzozowski and E J McCluskey was also used the concept of state – splitting in 1963 to eliminate unwanted states from the given automaton. It is the concept presently we use to get a regular expression from a given automaton very quickly.

There are three important methods to find a shorter regular expression from the given finite automata.

1.1 R McNaughton and H Yamada Regular Expression

In 1960, they developed an algorithm for obtaining a regular expression from finite automata [3]. This is the ancient and simplest method. In this method, initially, order of state removal should be assigned to all the states. Based on allotted orderings, considering the simplest order, we get the simplest regular expression. Following this way, step by step, we get a regular expression on order of removal states (different orderings gives different regular expressions). For complex set of states, those states having more transitions towards to it are considered later in the removal ordering successively. That is, the least transition order is considered first. Finally, the sum up of all sub regular expressions is yielding an equivalent regular expression to the given finite automata. The generated regular expression is relatively quite large of the given automaton.

1.2 Arden's Regular Expression

In 1960, Arden found the solution of Gaussian set of linear equations and proved that the problem containing $x = xa + b$ which was reduced as the solution as $x = ba^*$ provided a do not contain ε. In this method, given finite automata can be represented by set of linear equations. Before solving these equations, adding the empty symbol ε to the start state equation. Successively applying the above solution equation to the set of given linear equations, we obtain a regular expression. He imposed and applied the signal flow graph concept to the state diagram of the automata.

2. State Elimination Method

Brzozowski and McCluskey [1] devised an algorithm for constructing a regular expression from the given automaton by the application of signal flow graph theory. It is known as BMC algorithm. This is commonly used method. In this method, the states of an automaton are eliminated step by step until we kept with only initial and final states. For every eliminated state, regular expression is produced. The created regular expression operates as input for a state which is to be eliminated next. In elimination of the state, the regular expression is accumulated. On visualization of the diagram, we can easily understand how the states are eliminated.

The language of the automaton retains itself whenever the state elimination process is executed. Two important drawbacks of the state elimination method are (1) the size of the transition labels is increased in successive elimination of the states and (2) state elimination ordering gives the different regular expression in different ordering of states [4].

In this algorithm, a state is called a source if it has only outgoing edges and a sink if it has only incoming edges.

3. Brzozowski and McCluskey's State splitting method

Some automata have complex set of states and it is difficult to remove some of the states that consist of self – loop, in this situation, we can use state splitting method. An automaton may have any number of self – loop states and further we can eliminate self – loop states one by one using the following procedure. Before this, we define state – splitting.

3.1 Definition state – splitting

The self – loop of a state q can be omitted by adding two states q' and q'' similar to the given state q with the existing edge label (self – loop) between those two states is called a state – splitting. In the separated two states, q' and q'', one of the state is called source state and other state is called sink state. Here splitting a state eliminates the path through that particular state and all the edges connected to that node becomes normal (without self – loop) edges.

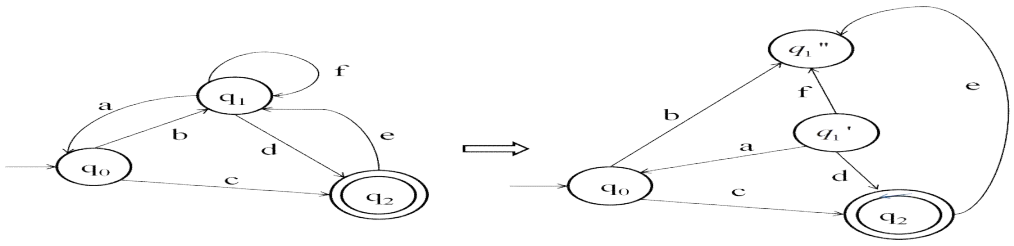


Fig 1: Illustration of State splitting (a) before splitting (b) after splitting

In the above diagram, state q_1 contains a self – loop so that it is separated into q_1' and q_1'' as shown in the diagram. Then, applying state elimination method, the above state diagram can be reduced by eliminating a set of states and accumulating the transitions into the sum of sub regular expressions for the paths by removed states. Subsequently, we can obtain a regular expression in the consolidated form of sub regular expression between two states consists of source and sink states.

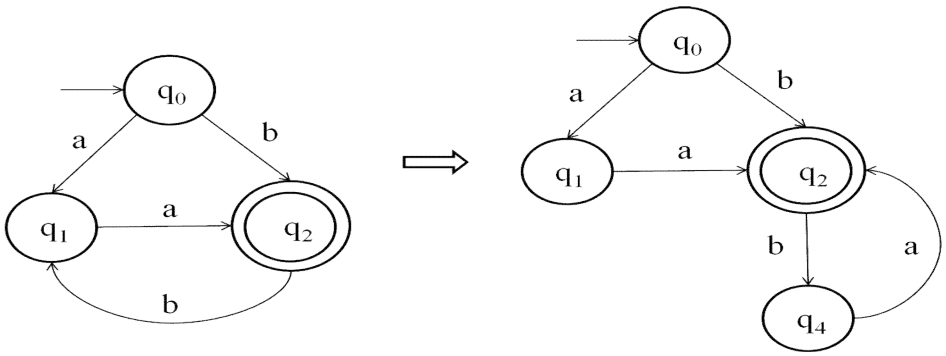


Fig 2: Another form of state splitting

In complicated set of self – loop states, we can have selection of state order in the following manner. Two types of state – splitting techniques are discussed with self – loop states. Of which, one self – loop state is modified by introducing two states with Kleene closure of the existing transition labeled of that state and by state elimination technique, we can get a regular expression. Another type of self – loop state, separating self – loop state into two states and taking other state as a base state which again forms a self – loop with base state in the same automaton, then we eliminate that base state by making a sub regular expression from source state to sink state in one path through the base state combining other direct path between source and sink states forms a self – loop gives the Kleene closure of the sum of the sub regular expression and finally we get a accumulated regular expression from the initial to the final state. Then we obtain a regular expression of the given automaton.

3.2 Example

The self – loop of a state changed by adding a state in addition that existing state and performed an usual operation, we get a regular expression for that particular state.

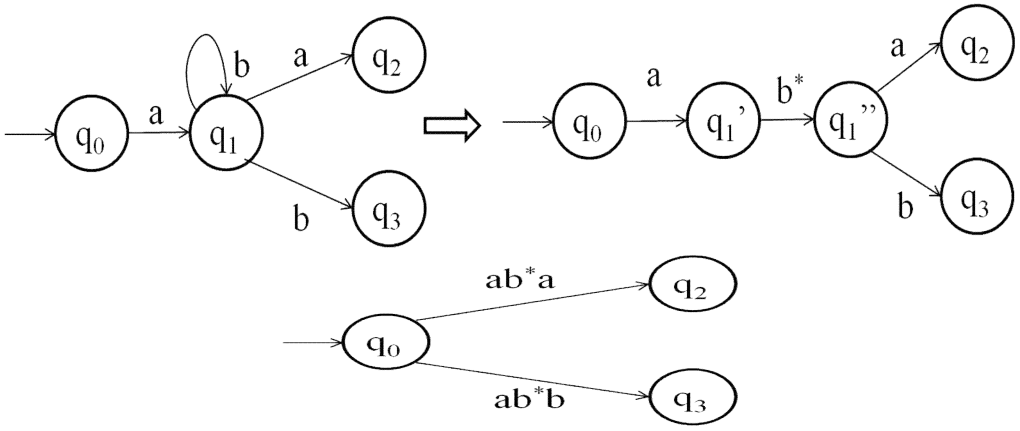


Fig 3: reduction of a state with self – loop

In the above figure, the state contains self – loop and adding an additional state with the same edge transition in the next diagram. Then by usual operations of SPL graph, the states are merged to get sub regular expression in the next diagram. This procedure eliminates self – loop of a state.

3.3 State Splitting Algorithm

1. Check whether the given automaton has more than one self – loop states or not. If so, begin with next step. Otherwise use SPL graph procedure to get the simplest regular expression.
2. In the SPL graph elimination process, parallel paths can be combined to form $a+b$ and all series paths in the form b^*.
3. From the given automaton, identify the initial and final states and if both the states have self – loop state.
4. Keeping the final state as a base and separating the initial state into source and sink states namely, for example, q_0' and q_0'' as the given self – loop state q_0 with transition labeled symbol will be set to between the above source and sink states.
5. Using SPL graph procedure and state elimination technique , q_0' and q_0'' through the final state, we get sub regular expression and already transition label between q_0' and q_0'' forms the combination, which the yields the sum of the sub regular expressions denotes $q_{s.re}$.
6. If the path between the original initial state and final state, use SPL graph procedure, we get final regular expression as q_{RE}.

3.4 Example

Consider two states self – loop automaton.

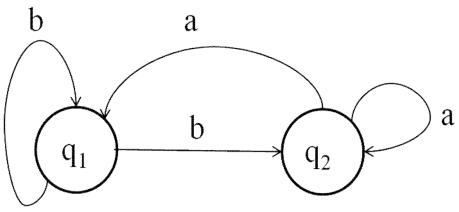
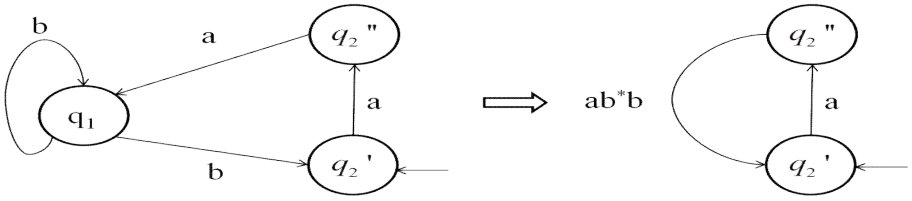
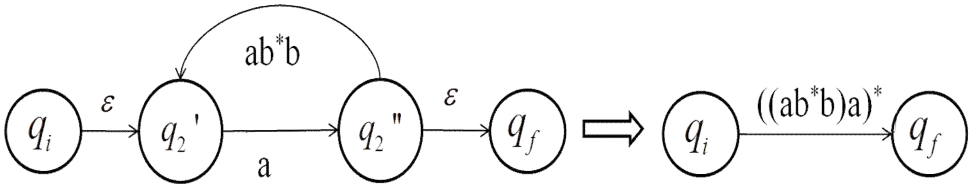


Fig 4: two state self – loop automaton

The above two state self – loop automaton can be reduced in two different state removal ordering, of which, we first find the regular expression of the state q_2 through the path of the state q_1 and the self – loop containing the state q_2. Here, we take q_1 as the base state, that is accepting state, then split the state q_2 into q_2' and q_2'' as source and sink states respectively as in the figure

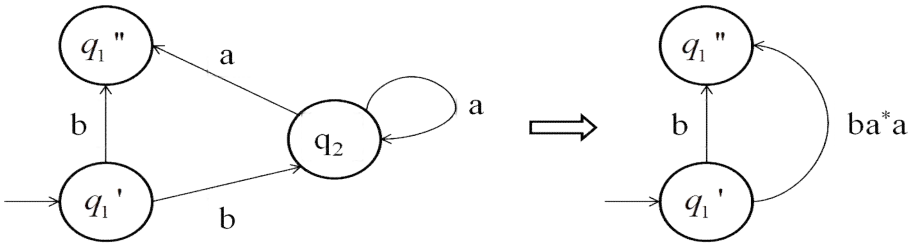


By SPL graph procedure and state elimination technique, q_1 is removing from the diagram and we obtain the sub regular expression ab^*b as in the above diagram.

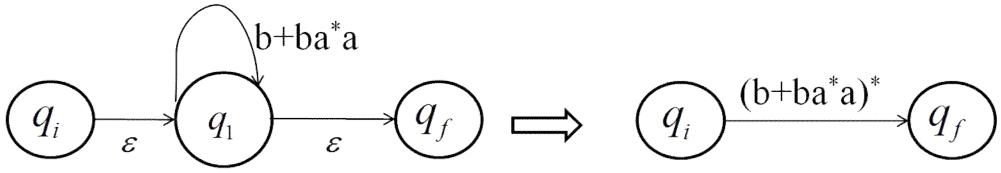


Finally, we get the regular expression of the state q_2 is $\left(\left(ab^*b\right)a\right)^*$. Here the path is between q_2 to q_1, so the entire regular expression of the automaton is $q_{RE} = q_2{}^*a^*a = \left(\left(ab^*b\right)a\right)^* a^*a$.

In another state removal ordering, we find the regular expression of the state q_1 through the path of the state q_2 and the self – loop containing the state q_1. Here, we take q_2 as the base state, that is accepting state, then split q_1 into q_1' and q_1'' as source and sink states respectively as in the figure



By SPL graph procedure and state elimination technique, q_2 is removing from the diagram and we obtain the sub regular expression ba^*a as in the above diagram.



Finally, we get the regular expression of the state q_1 is $\left(b+ba^*a\right)^*$. Here the path is between q_1 to q_2, so the entire regular expression of the automaton is $q_{RE}=q_1{}^*ba^*=\left(b+ba^*a\right)^*ba^*$.

By state removal ordering, the regular expression contains minimum number of stars is the desired regular expression [2].

3.5 Example

Suppose the given self – loop state may be singleton state that comprises of initial and final, then by state splitting procedure and SPL graph procedure, add two states before the self – loop state and add other two states after the self – loop state as shown in the diagram below. That is three edges adding to self – loop state with middle edge containing labeled a^* and other two edges containing labeled ε.

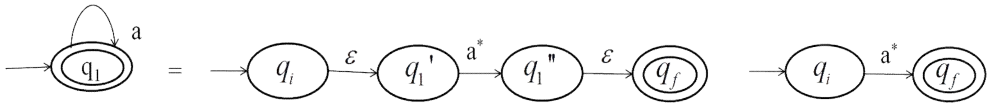


Fig 5: state splitting of singleton self – loop state.

3.6 Example

Suppose we want construct a regular expression from three state self – loop automaton. First we consider any two self – loop states of the given automaton and whether we have to use state split operation or not. If so, we analyze the self – loop states, and then apply the state split procedure. For example, the following automaton has three state self – loop in it. The operation is applying as below.

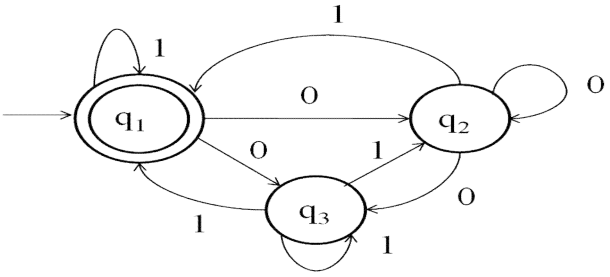
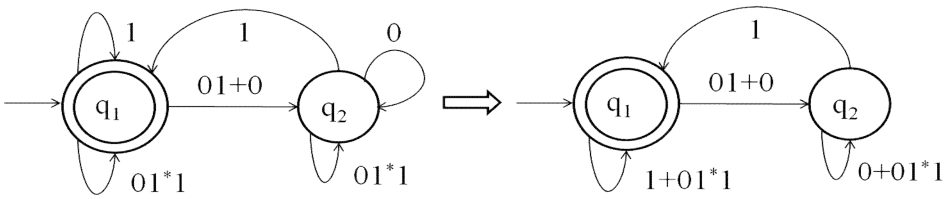
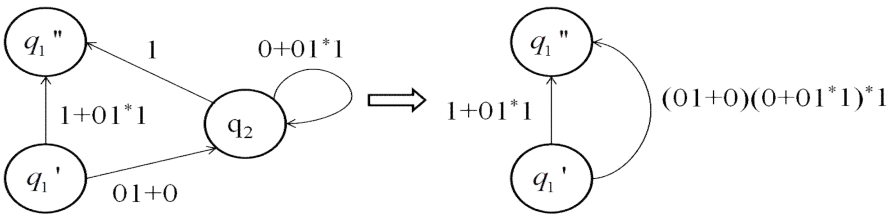


Fig 6: Three State Self – Loop Automaton

First, we eliminate the state q_3 by usual state elimination method, we get as below.



Then, we eliminate the state q_2 by state elimination method again, we get as below.



Finally, we get a regular expression as below

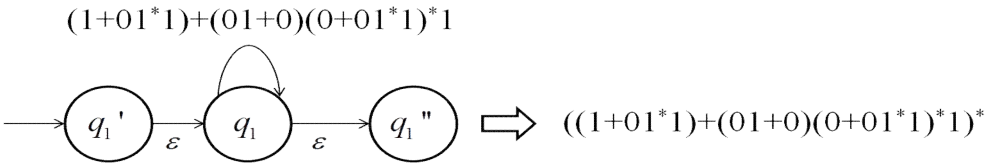
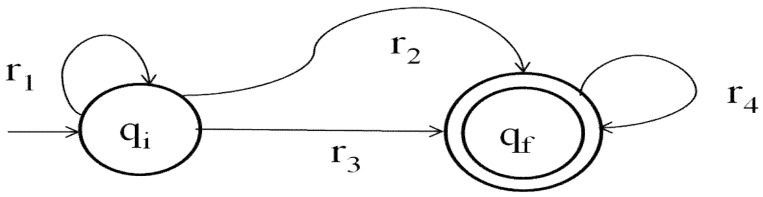


Fig 7: Illustration of 3–states self – loop automaton converted into a regular expression.

In general, for an NFA, the transition labels of the automaton contain regular expressions as seen below.

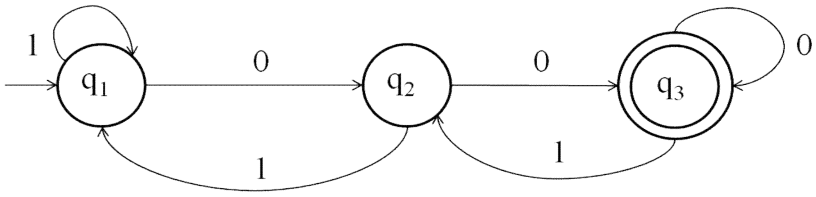


By state – splitting technique, we can get the regular expression as $R = r_1^* r_2 \left(r_4 + r_3 r_1^* r_2 \right)^*$. This is for complex set of regular expressions.

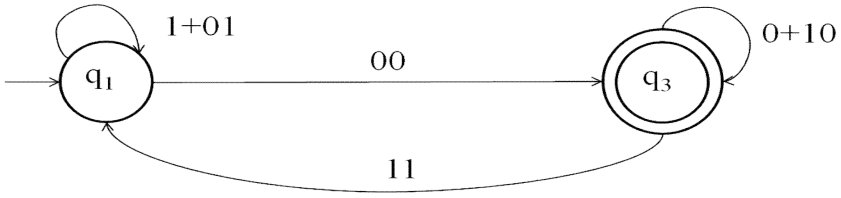
3.7 Example

The DFA can also be converted to the simple regular expression by state splitting procedure if it contains self – loop states. We will see the following example.

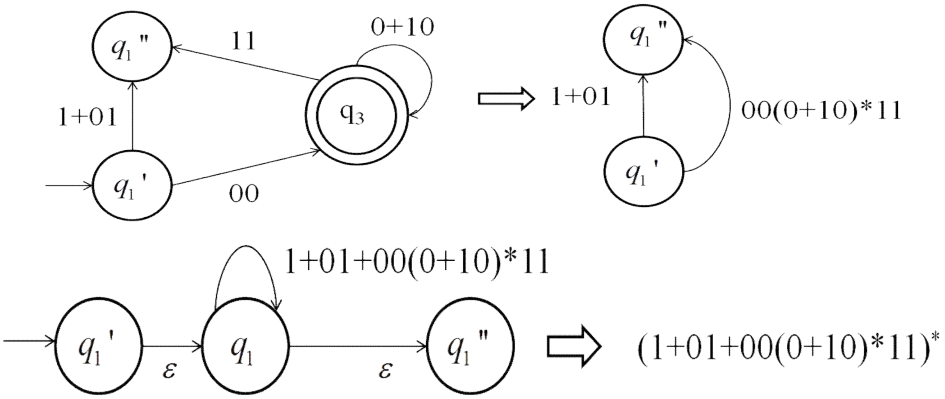
The given DFA is



After eliminating the state q_2 by state removal method, the modified DFA will be seen as below:



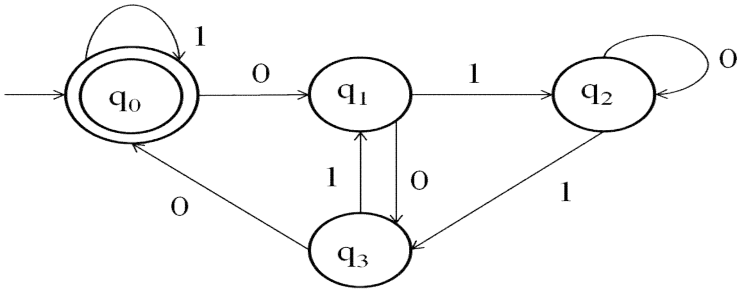
By applying State splitting procedure for the above diagram, we get the regular expression by successive diagrams.



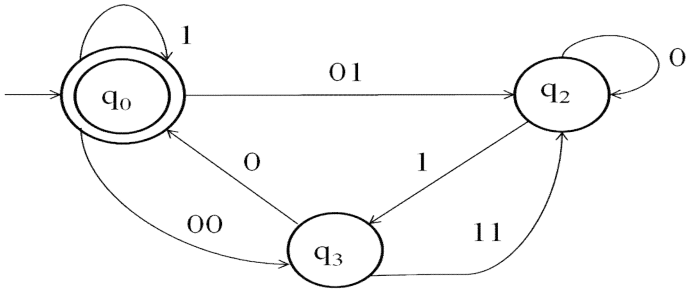
By the final state q_3 is absorbed by separating the state q_1 into q_1'and q_1", then we get sub regular expression for q_1 is $\left(1+01+00\left(0+10\right)^{*}11\right)^{*}$. Our accepting state is q_3, so the simplified regular expression is $\left(1+01+00\left(0+10\right)^{*}11\right)^{*}00\left(0+10\right)^{*}$.

3.8 Example

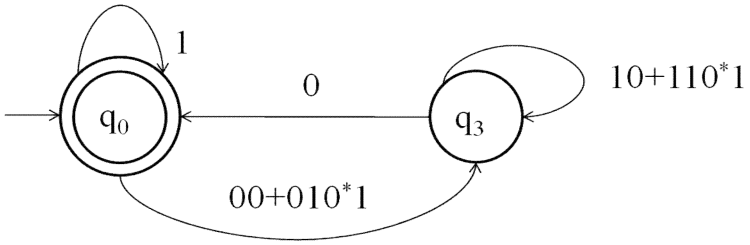
Convert the following DFA into a regular expression by state split procedure.



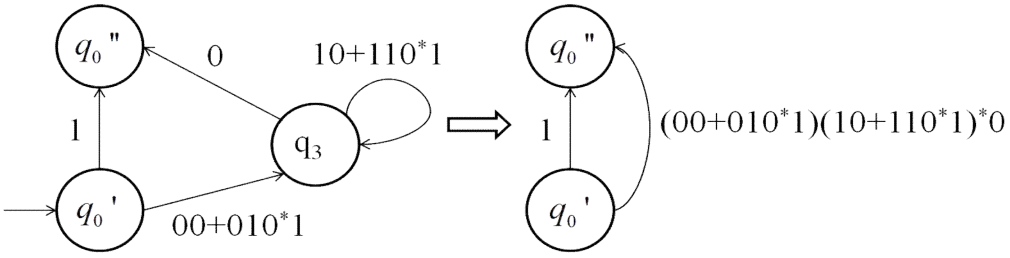
First we eliminate the state q_1 by state elimination technique. We get the following diagram as below.

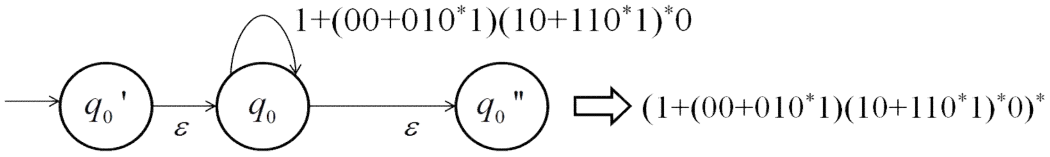


Also we eliminate the state q_2 by state elimination technique. We get the following diagram as below.



Now we can apply state split procedure for the above diagram.





In this way, we can find the regular expression for any type of automaton.

This method enhances the state elimination method quickly to get the regular expression. Also this method reduces the calculation as comparing with weight of the automaton.

4. Conclusion

It is the method for determining the desired regular expression in the quickest way. Even manually we can easily obtain a shorter regular expression through the state splitting method as comparing with other types of obtaining shorter regular expression of the automata.

References

[1]. Berry G, Sethi R, *From regular expressions to deterministic automata*, Theoretical Computer Science 48 (1986), 117–126.
[2]. Brzozowski J.A, McCluskey E.J, *Signal Flow Graph Techniques For Sequential Circuit State Diagrams*, IEEE Transactions on Electronic Computers, EC–12, 67–76, April 1963.
[3]. Delgado M, Morais J, *Approximation to the smallest regular expression for a given regular language*, In Proceedings of CIAA‟04, Lecture Notes in Computer Science, 3317, 2004, 312–314.
[4]. Han Y.S, Wood D, *Shorter Regular Expressions from Finite State Automata*, In: Proceedings of CIAA'05, Springer–Verlag (2006), 141–152 Lecture Notes in Computer Science, 3845.
[5]. McNaughton R, Yamada H, *Regular expressions and state graphs for automata,* IRE Trans. on Electronic Computers EC-9:1 (1960) 38–47.
[6]. Murugesan N, *Principles of Automata theory and Computation*, Sahithi Publications, 2004.
[7]. Sakarovitch Jacques, Elements of Automata Theory, Cambridge University Press, 111–118, 2009.