ISSN: 2321-7782 (Online)
Volume 3, Issue 12, December 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

State Minimization Approach in Deterministic Finite Automata using Inefficient State Elimination Approach

Amit Kishore Shukla[1]
Department of Computer Science
Motilal Nehru National Institute Of Technology
Allahabad, UP (India)

Akanksha Singh[2]
Department of Computer Science
Babu Banarasi Das University
Lucknow, UP (India)

Gaurav Ojha[3]
Department of Computer Science
Kashi Institute of Technology
Varanasi, UP (India)

Abstract: ***It is often observed that DFA can have more states than the regular need but those states all not as efficient as they have to be, so those state increases the complexity. This paper is primarily concern about the removal of those states to make DFA more efficient .DFA contain unreachable states that cannot be reached from initial state. DFA's may also include dead states that cannot reached any accepting state that means neither unreachable states nor dead states can participate in scanning any valid string. Therefore we have to eliminated all such states in ordered to optimized processes.***

Keywords: Deterministic Finite Automata, Non Deterministic Finite Automata, Unreachable States, Dead States

I. INTRODUCTION

A DFA for which all edges with the same label lead to a single vertex. DFA automata 'accept' the class of local languages, those for which connected of a word in the language is finding by a "sliding window" of length two on the word. DFAs were invented to model 'real world' finite state machines in contrast to the concept of a Turing machine, which was too general to study properties of real world machines.

- DFAs are one of the most practical models of computation, since there is a trivial sequential time, constant-space, online algorithm to simulate a DFA on a maximum of input. Also, there are efficient algorithms to find a DFA recognizing :
- The complement of the natural language recognized by a given DFA.
- The union/intersection of the natural languages recognized by two given DFAs.

DFAs can be minimizing to a 'consequence form' (minimal DFAs), there are also effective algorithms to determine:

- Any DFA accepts any strings
- Any DFA accepts all strings
- Any two DFAs recognize the same language
- The DFA with a lowest number of states for a particular regular language

II. PROPOSED ALGORITHM

In this proposed algorithm first of all we have generated useful states before minimization. If we generate useful state then useless state (unreachable state) will remove simultaneously. Initially we proposed initial state as a final state and obtain only one input symbol from initial state and move simultaneously with changing accepting input symbol state. In the fig1 input

symbol is ‘a’, ‘b’ because outgoing symbol of state q0 is ‘a’, ‘b’. Input symbol ‘a’ and ‘b’ both is rejecting so we can see view trace by JFLAP simulator and finalized all state including accepting input symbol.

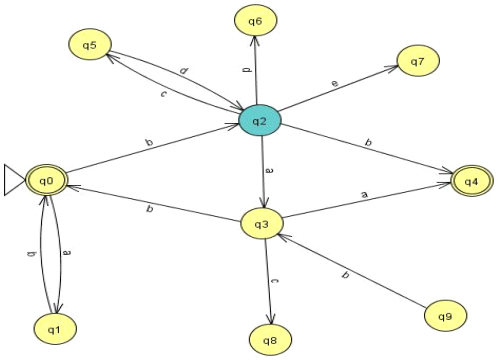

Fig 1:Deterministic Finite Automata with useless states

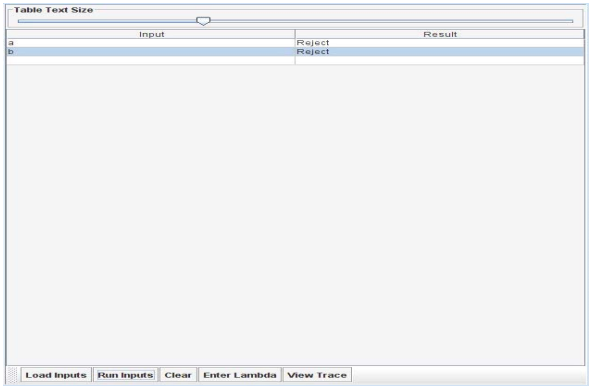

Fig 2: Multiple Run with ’a’ and ‘b’ input symbols

After that take another input string, taking input string as per proposed rule that means all input take simultaneously with string length is one that means only one symbol included at a time. As shown in below fig3:

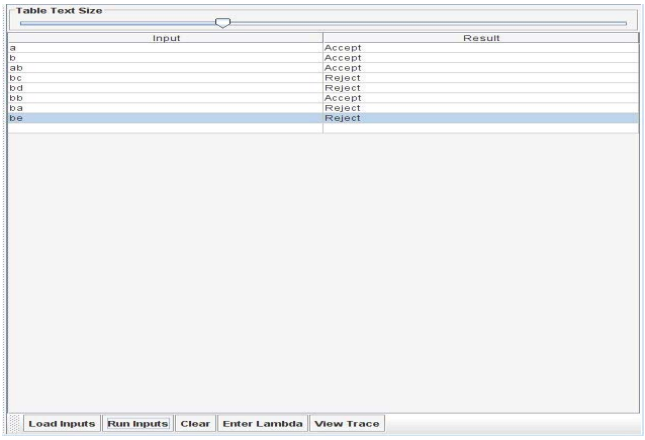

Fig 3: Multiple run with ‘one length’ and ’two length’ input symbols.

In this fig3 if we have taken input symbol ‘ab’, ‘bc’, ‘bd’, ‘be’, ‘ba’ and ‘bb’. Input symbol ‘bb’ and ‘ab’ are accepting but input symbol. ‘bc’, ‘bd’, ‘be’ and ‘ba’ are rejecting so we can see view trace by JFLAP simulator and finalized all state including accepting input symbol . Input symbol ‘bb’ and ‘ab’ are accepting but input symbol so state q2 and q1 is also a final state. Remembering that all the input symbol start with initial state. After that taking next input string as shown in below fig:

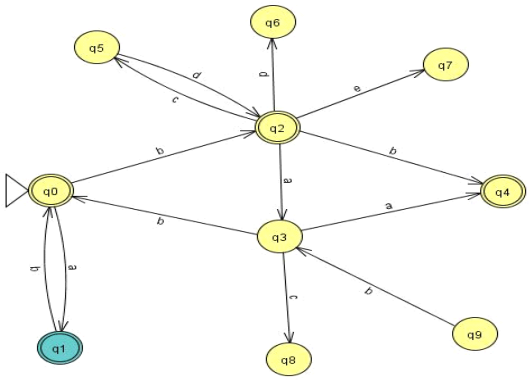

Fig 4:DFA with after finalized all state that include accepting input symbols

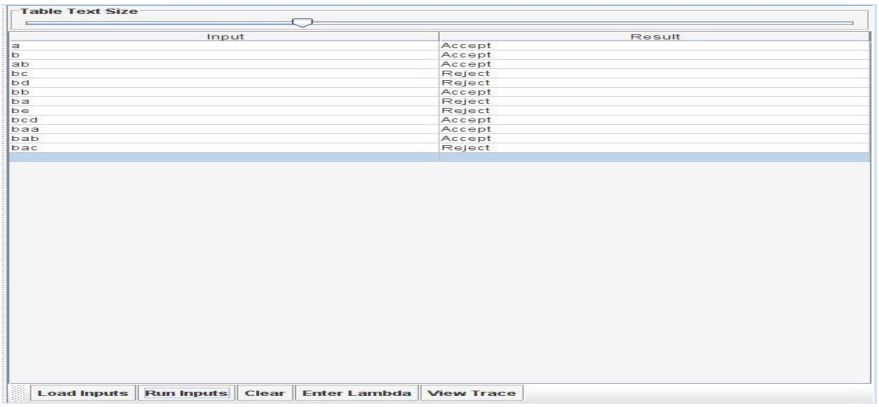

Fig 5: Multiple Run with 'One, Two and Three Length' Input Symbols.

In fig.4 some state is finalizing because this state including accepting input symbol again we have taken next input symbol. If no any input symbol is accepting then we have taken next possible input symbol. Again if no input symbol accepting then repeats it whenever all possible input not finished.

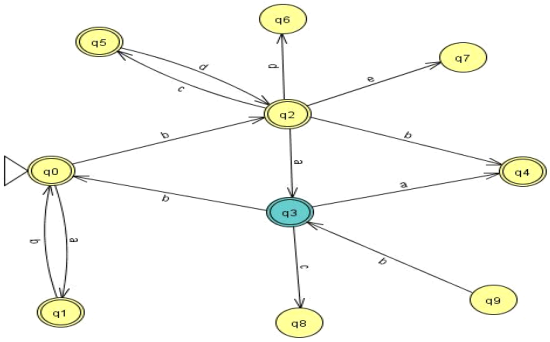

Fig 6: DFA with after finalized all state that include accepting input symbols

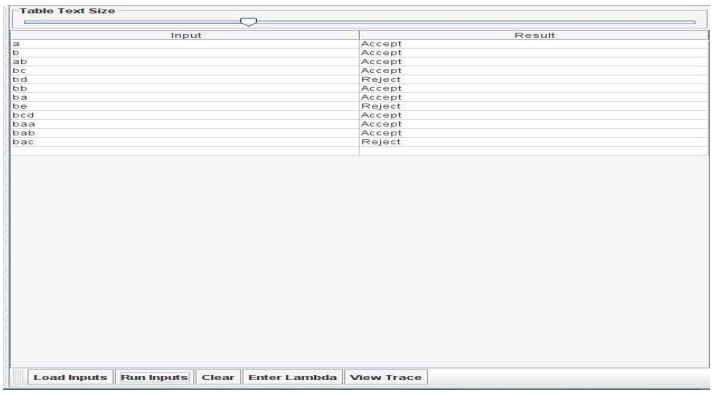

Fig 7: Multiple Run with All Possible Input Symbols

In fig.6 we can see all taking string is accepted that means no any useful state remaining for generating technique of useful state in deterministic finite automata. When all possible input taken so we will remove all non-final state.

In fig. 8 final state indicate accepting string generated by using these state and non- final state indicated no any string generated by using these state.

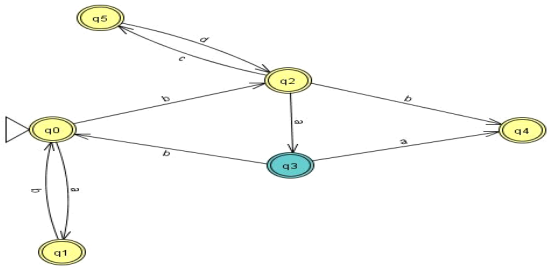

Fig 8:Deterministic Finite Automata With Without Unreachable State.

In the above fig.8 all final state available and no any non- final state available. Final state shows these state are including in accepting string . Now next step for proposing technique, remove all finalized label of state except initialized form means in initial form of deterministic finite automata initial state was q0 and final state was q1 so these state are not same as a previous form and all updated label should be remove. As shown in below fig.9

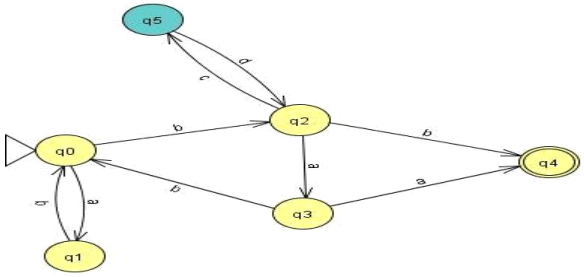

Fig 9:Deterministic Finite Automata with Useful State

Algorithm:

INPUT: A= (Q,∑,δ,q_0,q_f) – Deterministic finite automata.

OUTPUT: A'= (Q',∑,δ',$q_{0'}$,q_f') – Deterministic finite automata without unreachable state

- For ($q_{\epsilon Q}$) /*all state belong to given set of deterministic finite automata *\
- Go to (q_i <- Initial State) /* go to initial state *\
- And MAKE q_i<- Final State
- If ($\delta_{i\text{-}Fi}$ ->q_f) /* this show transition function if any state reach to final state that means input string is accepted *
- MAKE $q_{i+1\text{-}Fi}$ <- Final State /* if any iteration for accepting state then all the state including in this iteration should be final state*\
- Else
- q_{i} ->q_{next} /*if input string is not accepted then move to another state q_{next} *\
- END Else
- END if
- END For

» For (q' € q'$_f$) /*after checking all possible input string according to proposed technique all useful state should be final state*\

» If (q' €! q'$_f$) Then /* unreachable state*\

» q' <- Remove /* remove unreachable state*\

» END if

» END For

» For (q'$_i$ € q'$_f$) /* after removing unreachable state all useful state present and all state should be final state *\

» if($q_{initial}$ € q'$_f$) Then /* checking proposed initial state is final or not *\

» $q_{initial<-}$ previous position /* if proposed initial state is final the it will form previous state*\

» Else

» q'$_f$ ϵq_fThen /* all proposed state is final state*\

» q'$_f$<- Previous position /* all proposed state become previous state*\

» END Else

» END if

» END For.

III. Experimental Results

The objective of this paper is primarily done for removing unreachable states and dead states by using inefficient state elimination approach. In this paper the primarily focus for removing unreachable state and dead state by using proposed approach. In this particular approach, at a time only single outgoing symbol have to take. First of all we have moved from initial state and checked how many outgoing symbol present in this state after we have taken only one symbol from start state. We can check which state follows this symbol that means how many state present for generating accepting symbol by using view trace of JFLAP simulator. Primarily we know about how to generate Deterministic Finite Automata after that check for useful or unreachable state and dead state in deterministic finite automata.

Step 1:- First of all we clicked JFLAP simulator when simulator open in new window then we have selected state creator button. In this simulator if we want to create state then we have to drag state creator symbol. As shown in below fig it show state q0,q1,q2,q3,q4,q5,q6,q7, q8,q9 and q10 are the state if we want to change name of this state then we have changed name by selecting attribute editor and clicking right on symbol and go to set name.

Step 2: After generating state by state creator in JFLAP simulator then second important task is creating initial state and final state. In JFLAP simulator next editor uses that means we have used attribute editor button for creating initial state and final state. In given below example state q0 is an initial state and state q4 is a final state in below given deterministic finite automata. State q0 denote (add) arrow symbol after using attribute editor and state q4 denote double circle by attribute editor using JFLAP simulator.

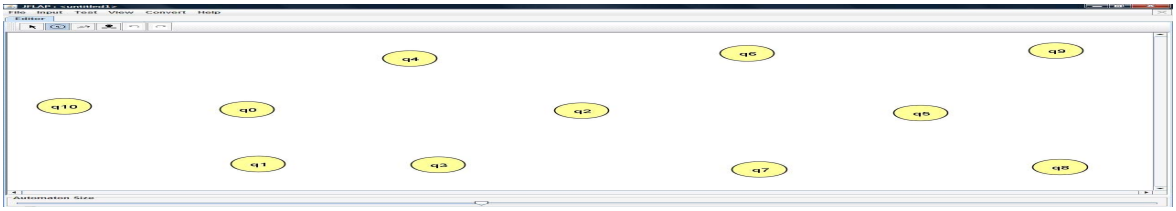

Fig 10: Creating state by JFLAP simulator

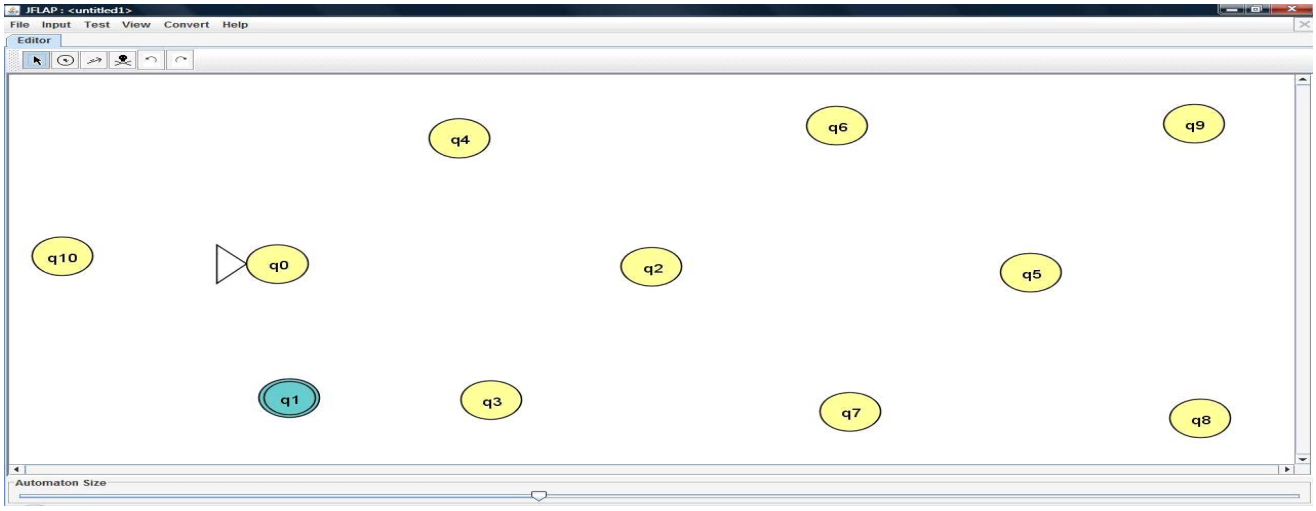

Fig 11: Creating initial and final state

In fig4.3 state q0 and q4 are initial and final state respectively.

Step 3: After step 2 another important task remaining that means how input symbol show in deterministic finite automata for this problem we can use JFLAP simulator for generating transition from one state to another state. In JFLAP simulator transition creator button available so we can select transition creator button and drag this button one state to another state. After clicking one box generated and we will put symbol in this box. In below example 'a,b,c' and 'd' are input symbol and this symbol use for transition from one state to another state.

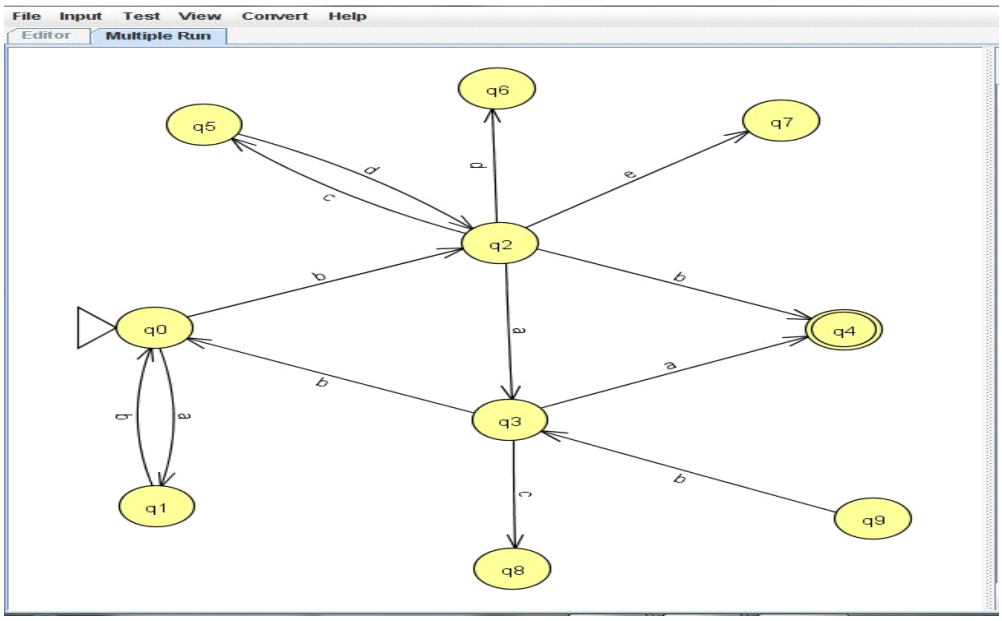

Fig 12: Generating Transition By Given Input Symbol 'a','b', 'c' and 'd'

In above fig from state q0 to q1 input symbol 'a' taking that means from state q0 to q1 transition symbol is 'a'. And this symbol using for generating any type of string. Also this symbol uses transition function.

Step 4: After using step 3 we will see deterministic finite automata generated but this type of deterministic finite automata is not a well form because in this automata unreachable state and dead state are available in proposed example state q6,q7,q8 are unreachable state. So next task is removing this type of state from deterministic finite automata. For this task we use JFLAP simulator in this simulator we will go to input box and select multiple run then one new window open as shown in below fig. in this window table text size given in this part to column present first is input and second is result.

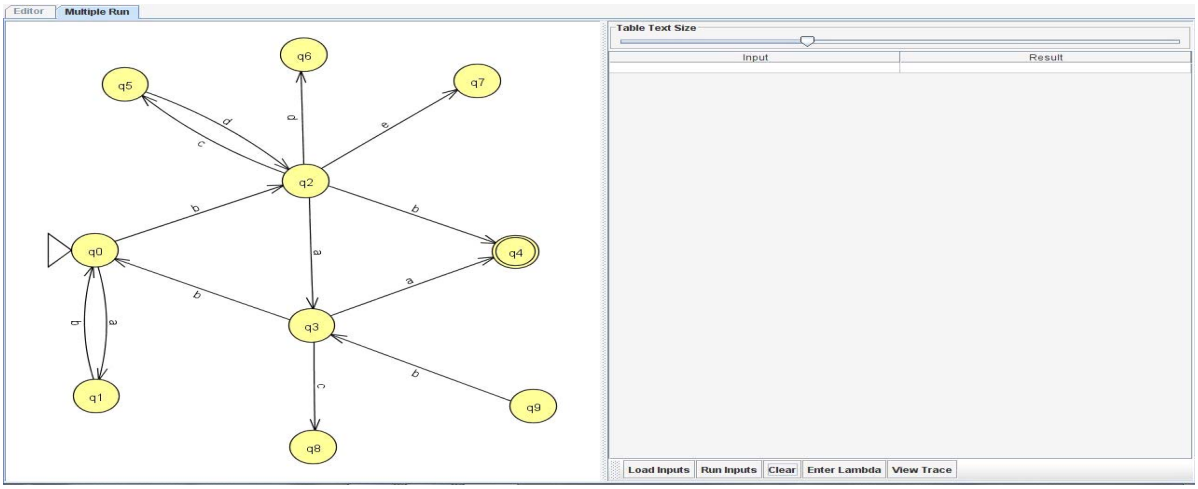

Fig 13: Open New Window for Putting Input Symbol and Check in Input Result

In above fig.13 we can see below part of this window is given five button we can use as a require task example for if we want to check input result is accepted or rejected then we can click run input.

Step 5: After opening new window we can put input in given option. In below fig input 'a', and 'b' taking because form initial state q0 outgoing transition is four 'a', and 'b'. After putting input symbol we will click 'RUN INPUTS' button for check in which input string is accepted or rejected. As shown in fig .14

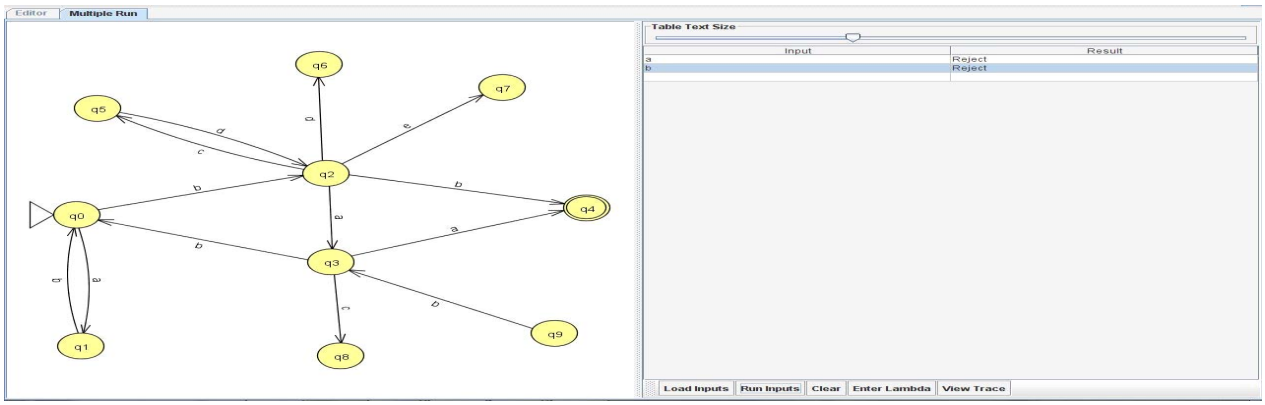

Fig 14: Taking Input Symbol 'a','b' From Initial State q0

Step 6: After putting all possible input we have selected or click 'RUN INPUTS' button and see all result of input symbol is accepted this is easily know about any researcher for which state is unreachable that means if any state didn't participate or not including in accepting state then this type of state will not be considered useful for deterministic finite automata. So in next state this type of state will be removed.

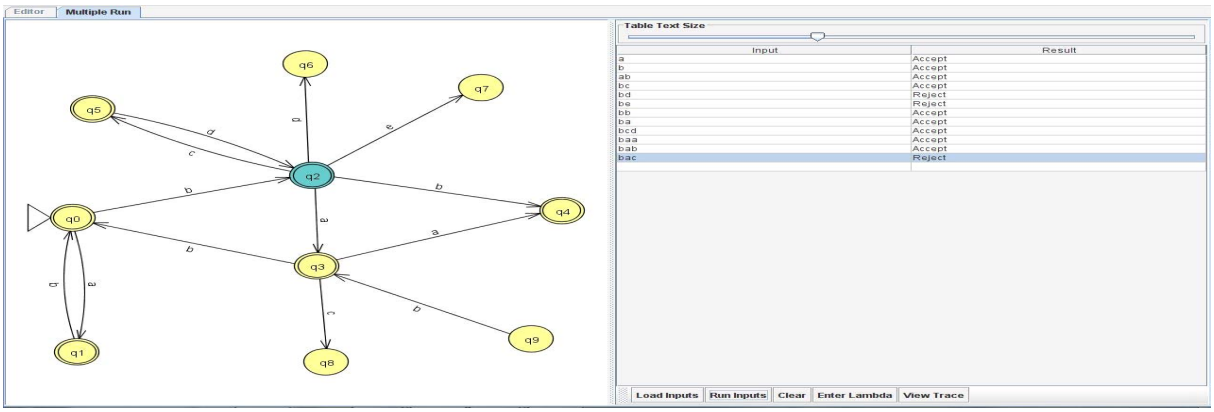

Fig 15: Taking All Possible Input Symbol

Step 7: After step 6 we get to know about which state is useful and which state is useless so finally we will remove unreachable state by using JFLAP simulator 'Delete' button. And we have seen all useful state is in final state so in next state we will remove final state and finalize in initial position with ought including unreachable state.

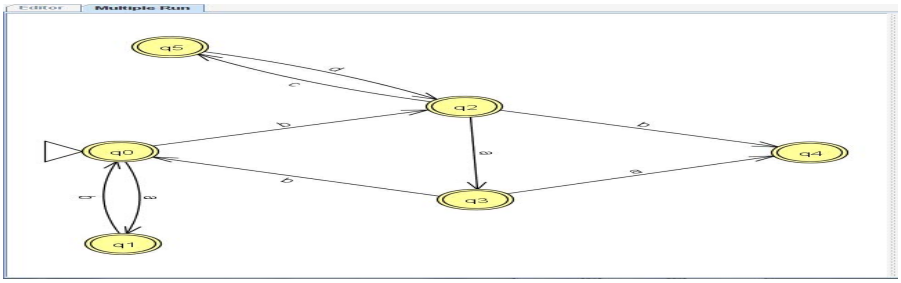

Fig 16: Removing All Unreachable State

Step 8:- When all non-final state got removed by step 7 after that initializing all the state of given deterministic finite automata that means remove label of modifiable state. After that remaining state is called a useful state. After that only useful state remaining. So, apply this approach for generating useful state. After applying this approach all unreachable state is removed and only useful state is remaining.

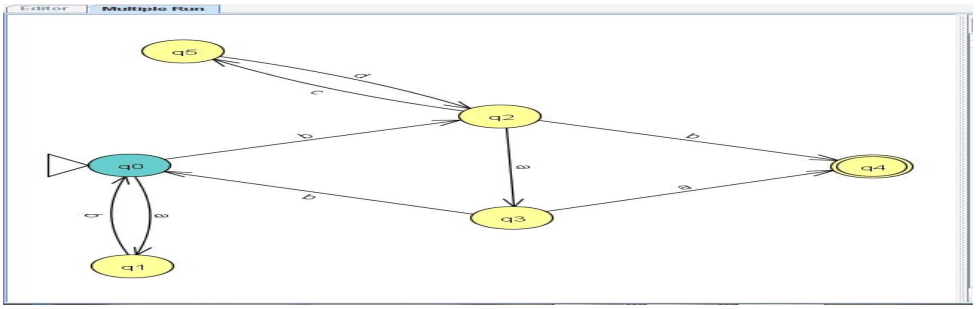

Fig 17: Useful State Remaining

IV. CONCLUSION

In this paper we primarily aimed at the removal of unreachable and dead state of deterministic finite automata. We have followed simple approach for generating useful state by proposed approach in this work. We have chosen JFLAP simulation for removal of inefficient state of deterministic finite automata. If the proposed technique apply for generating useful state then both unreachable state and dead state would simply removed. If we have gone through with contemplated technique or approach, then the number of steps for taking input is lesser than running technique. After selecting useful state we have minimized simply of deterministic finite automata.

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.cscolum bia.edu/ ~aho/cs3261/lectures.
2. S. H. Rodger. Jap web site, and tutorial, 2011. www.jflap.org
3. Hao Wang, Student Member, IEEE, Shi Pu, Student Member, IEEE, Gabe Knezek, Student Member, IEEE, and Jyh-Charn Liu, Member, IEEE, MIN-MAX: A Counter-Based Algorithm for Regular Expression Matching, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 24, NO. 1, JANUARY 2013.
4. DomenicoFicara, Member, IEEE, Andrea Di Pietro, Student Member, IEEE, Stefano Giordano, Senior Member, IEEE, Gregorio Procissi, Member, IEEE, Fabio Vitucci, Member, IEEE, and Gianni Antichi, Member, IEEE, Differential Encoding of DFAs for Fast Regular Expression Matching, IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 19, NO. 3, JUNE 2011.
5. DomenicoFicara, Member, IEEE, Andrea Di Pietro, Student Member, IEEE, Stefano Giordano, Senior Member, IEEE, Gregorio Procissi, Member, IEEE, Fabio Vitucci, Member, IEEE, and Gianni Antichi, Member, IEEE, Differential Encoding of DFAs for Fast Regular Expression Matching, IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 19, NO. 3, JUNE 2011.
6. Jean-Charles Delvenne and Vincent D. Blonde, Complexity of Control on Finite Automata, IEEE TRANSACTIONS ON AUTOMATIC CONTROL, VOL. 51, NO. 6, JUNE 2006.
7. Attila Csenki, Flowgraph Models in Reliability and Finite Automata: IEEE TRANSACTIONS ON RELIABILITY, VOL. 57, NO. 2, JUNE 2008.

8. R. W. Butler, "Reliabilities for feedback systems and their saddlepoint approximation," Statistical Science, vol. 15, pp. 279–298, 2000.
9. P. Linz "Theary of Computatuon Regular Expression from a DFA", Lecture notes in Computer Science Theory, September 27, 2008.
10. H. Gruber and J. Johannsen, "Optimal lower bounds on regular expression size using communication complexity", In Proceedings of the 11th International Conference Foundations of Software Science and Computation Structures, volume 4962 of LNCS, pages 273–286, Budapest, Hungary, March–April 2008. Springer.
11. Mishra K.L.P.& N. Chandrasekaran, "Theory of Computer Science (Automata Language and. Computation)", PHI, third edition, 2007.
12. Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., Mohri, M.: OpenFst: A general and efficient weighted finite-state transducer library. In: Proc. 12th Int. Conf. Implementation and Application of Automata (CIAA). LNCS, vol. 4783, pp. 11_23. Springer (2007).
13. Susan Rogers. Java Formal Language Automata Package (JFLAP), February 2006. http://www.jflap.org.
14. Susan H. Rodger, Eric Wiebe, Kyung Min lee, Chris Morgan, Kareem Omar, and Jonathan Su. Increasing engagement in automata theory with jap. In Fourtieth SIGCSE Technical Symposium on Computer Science Education, pages 403{407. SIGCSE, March 2009.
15. S. H. Rodger and T. W. Finley. JFLAP – An Interactive Formal Languages and Automata Package. Jones and Bartlett, Sudbury, MA, 2006.
16. J. Berry. Improving discrete mathematics and algorithms curricula with link. In ACM SIGCSE/SIGCUE Conference on Integrating Technology in Computer Science Education, 1997.
17. J. Stasko. Tango: A framework and system for algorithm animation. IEEE Computer, pages 27–39, Sept 1990.
18. T. Naps. Jhave: Supporting algorithm visualization. IEEE Computer Graphics, 25:49–55, 2005.
19. Gelade, W., Neven, F., "Succinctness of the complement and intersection of regular expressions", Symposium on Theoretical Aspects of Computer Science. Dagstuhl Seminar Proceedings, vol. 08001, pages 325–336. IBFI (2008).
20. 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, 2004, pages 312–314.
21. H. Gruber and M. Holzer, "Finite automata, digraph connectivity, and regular expression size", In Proceedings of the 35th International Colloquium on Automata, Languages and Programming, Iceland, July 2008. Springer.
22. Yo-Sub Han and Derick Wood, Obtaining shorter regular expressions from finite-state automata , Theoretical Computer Sciecce 370(1-3) (2007): pages 110-120.
23. Ulman, J., A. V. Aho and R. Sethi, "Compiler Design: Principles, Tools, and Techniques", Pearson Education Inc, ISBN 0-201-10088-6, 1986.
24. Y.-S. Han and D. Wood, "The generalization of generalized automata: Expression automata", International Journal of Foundations of Computer Science 16 (3) (2005) pages 499–510.
25. McNaughton R. and Yamada H., "Regular expressions and state graphs for automata". IEEE Transactions on Electronic Computers 9, pages 39–47 (1960).
26. Gaurav Ojha , Akash Niras ,Uday Singh "State Minimization Algorithm to Remove Useless State in Deterministic Finite Automata" International Journal of Enhanced Research in Management & Computer Applications ISSN: 2319-7471, Vol. 4 Issue 8, August-2015

