Elimination of States using State Minimization Approach

[1]Saroj Kumar, [2]Dr. Santosh Kumar, [3]Priya Singh
[1]Research Scholar, [2]Associate Professor - MUIT - Lucknow
[1]Computer Science and Engineering
[1]Maharishi University Information Technology, Lucknow, India

Abstract : ***This paper presents an entirely new technique for calculating and removing dead states, unreachable states and indistinguishable states with unreachable states, dead states and any other states that lead to final state(s). A central problem in Automata theory is to minimize a given Deterministic Finite Automata (DFA). DFA minimization is an important topic that can be applied both theoretically and practically. When there are dead states or unreachable states or indistinguishable states in a DFA, then the time complexity of to minimize the DFA is increases drastically. In DFA, it is not easy to determine dead states, unreachable states and even difficult to identify and remove indistinguishable states when it is attached to the some states that lead to the final state. And removing these useless states from deterministic finite automata is very necessary to generate useful strings. We proposed an algorithm which removes all the useless states that are not involved in the string generation and also remove all the indistinguishable and redundant states. Also if we follow given technique and algorithm, after selecting useful state we can minimize simply of deterministic finite automata.***

Keywords - Automata, Deterministic Finite Automata - DFA, Unreachable State, Dead State, Indistinguishable State, IS-Indistinguishable State, US-Unreachable State, DS-Dead State, DFA-Deterministic Finite Automata

I. INTRODUCTION

Automata- It is define as a system where some information, material or energy is transmitted, transformed or used to perform actions without the actual participation of man. And in other words we can describe as a machine for generating regular expression, context free grammar, context sensitive grammar and recursive endurable language. A finite automaton can be represented by a 5-tuple (Q, Σ, δ, q_o, F), where [3, 4, 5].

1. Q is a finite nonempty set of states.
2. Σ is a finite nonempty set of input called the input alphabet.
3. δ is the next state function, δ : D → 2Q where D is a finite subset of Q × Σ*
4. q_o: initial state: $q_o \subseteq Q$
5. F: set of final states: F⊆Q

Above definition is valid for both DFAs (deterministic finite automata), and NFAs (nondeterministic finite automata) [6,7].

Deterministic Finite Automata: DFAs are called deterministic because following any input string, we know exactly which state it's in and the path it took to get there. [8].**Deterministic finite automata (DFA)** can be described by 5tuples (Q, Σ, δ, q_0, F), where

1. Q is a finite non-empty set of states
Σ is a finite non-empty set of symbols
δ is the next state function, that is, δ: Q × Σ → Q. q0 is the initial state; $q_o \subseteq Q$ F is a set of final states of Q (i.e. F⊆Q) called accept states [9].

Transition functions can also be represented by transition table as shown in table 1.1. A finite automata is represented by ({0, 1, 2, 3}, {0}, δ, {0}, {3}) where, δ is shown in the following table [10].

State(Q)	Next State δ(q,0)
0	1
1	2
2	3
3	3

Table 1.1: Transition Table representing transition function of DFA

Transition function can also be represented by transition diagram as shown in figure 1.1.

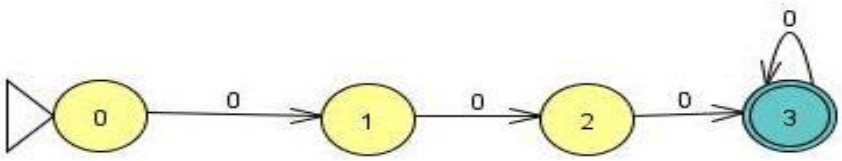

Figure 1.1: Deterministic finite automata corresponding to table 1.1.

2. PROBLEM

The problem is to identifying and removing the useless states that do not take part in string generation and redundant states from the deterministic finite automata. Indistinguishable state is the one of the major issue for DFA. First of all, we try to know what is the unreachable state, dead state and indistinguishable state.

Unreachable state: All those states which can never be reached from initial state are called inaccessible states or unreachable state.

Dead state: All those non final state which transit to itself for all input symbol in Σ are called Dead state.

Indistinguishable state: State p and q are indistinguishable if, staring in p and q, every string leads to the same state of "finality" (i.e., the strings fail or succeed together.)

- $\delta^*(p, w) \in F \Rightarrow \delta^*(q, w) \in F$, and
- $\delta^*(p, w) \notin F \Rightarrow \delta^*(q, w) \notin F$,
- for all string $w \in \sum^*$

There are some more partial indistinguishable states that uselessly increase the number of states in the DFA as we will see in the example from proposed algorithm.

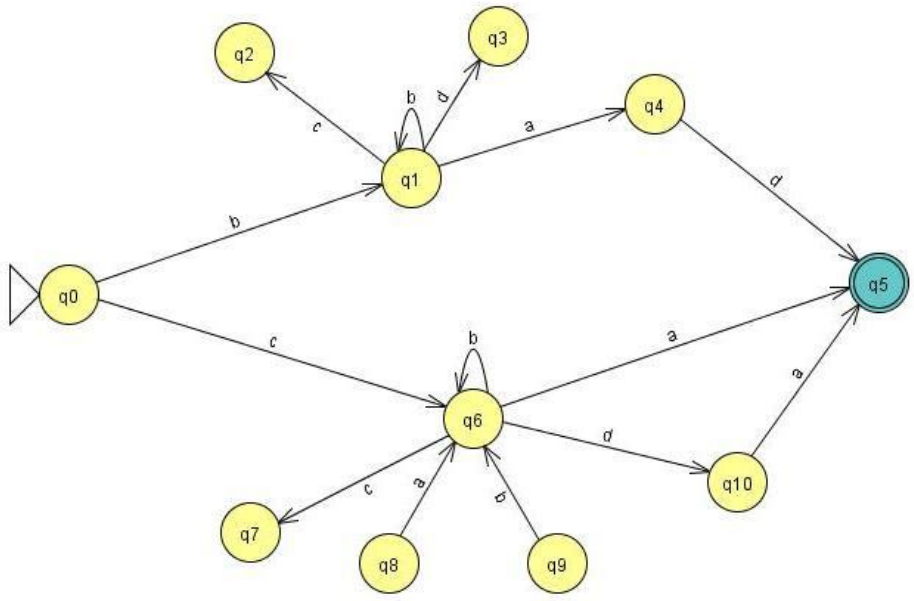

Figure 2.1: Deterministic finite automata with unreachable state, dead state and indistinguishable state.

In the above figure (that we take as example for illustrating our technique), we assume that the transitions from the states at unspecified input alphabet are connected to a dead state (not shown in above figure). In the above figure, states q_8 and q_9 are unreachable states because if we take any string from initial state q_0 to q_8 or q_9 then it is not possible. And states q_2, q_3 and q_7 are dead states. And q_1 and q_6 are somewhat redundant or partially indistinguishable states that is very difficult to detect by running techniques but with our proposed technique, it will be easy to remove any redundant state.

3. PROPOSED TECHNIQUE

In the proposed technique, we have to remove all the redundant or useless states before generating useful states for minimization.
In the first example, we take indistinguishable state having unreachable state and dead state. And we have to remove one of the indistinguishable states which have more unreachable state and dead state. Then reachable state and dead state are automatically removed from dead state.
In the second example, we take indistinguishable state having unreachable state and dead state and some other states that can lead to final state.

3.1 EXAMPLE1

In proposed approach first of all take only one input symbol from initial state and move simultaneously with changing accepting input symbol state as shown in below figure.

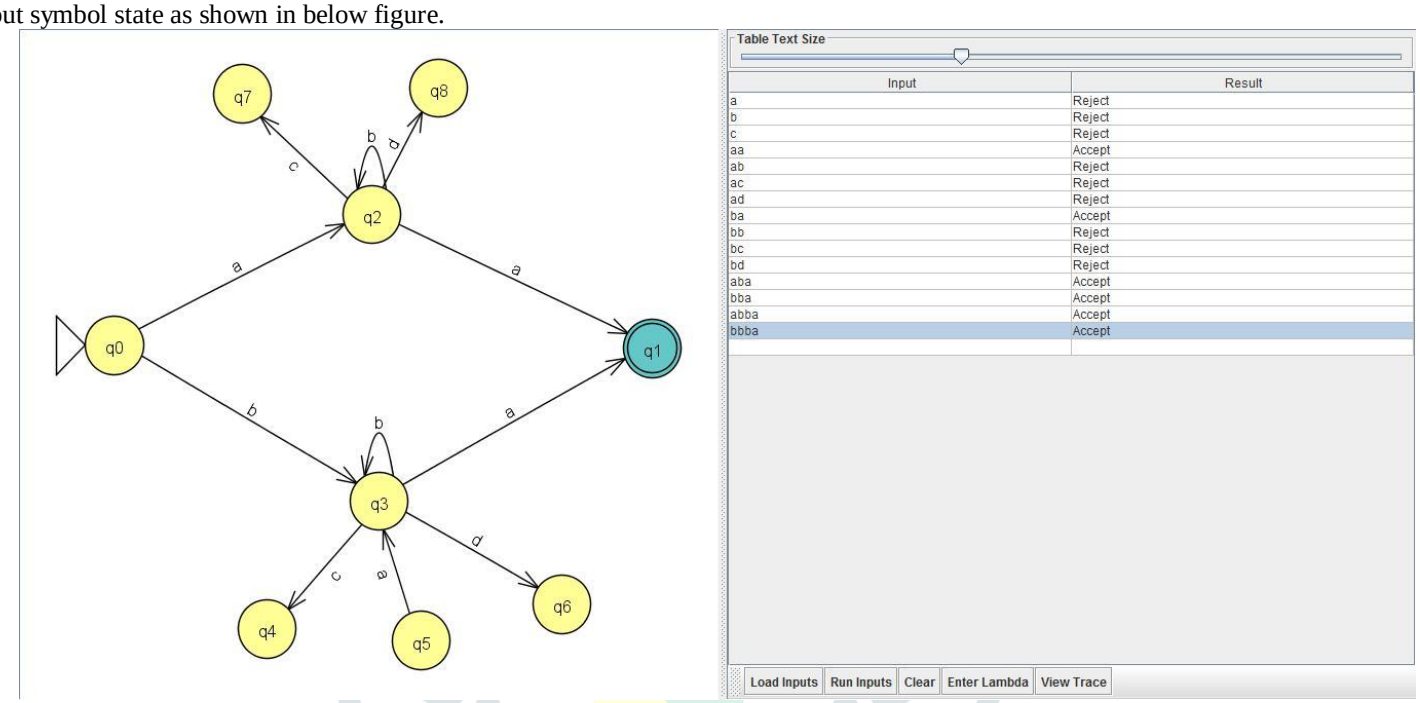

Figure 3.1.1: Deterministic finite automata.

Step1. Find Indistinguishable State.
In the above figure, taking outgoing input symbol from q_0 are a, b, c, aa, ab, ac, ad, ba, bb, bc, bd, aba, bba, abba, bbba and in these input symbols aa, ba, aba, bba, abba, bbba are accepting symbols and a, b, c, ab, ac, ad, bb, bc, bd are rejecting symbols. We can see view trace by JFLAP simulator [11]. In the accepting symbol, ba string is common other than aa string. States q_2 and q_3 are satisfy the condition for indistinguishable state.
i.e., $\delta(q_2, ba) \in F \Rightarrow \delta(q_3, ba) \in F$
$ba \in \sum^*$
So q_2 and q_3 are equivalent state then we have to remove one of the equivalent state for generating useful state.

Step2. Check which indistinguishable state have more unreachable and dead state:
Check for q_2:

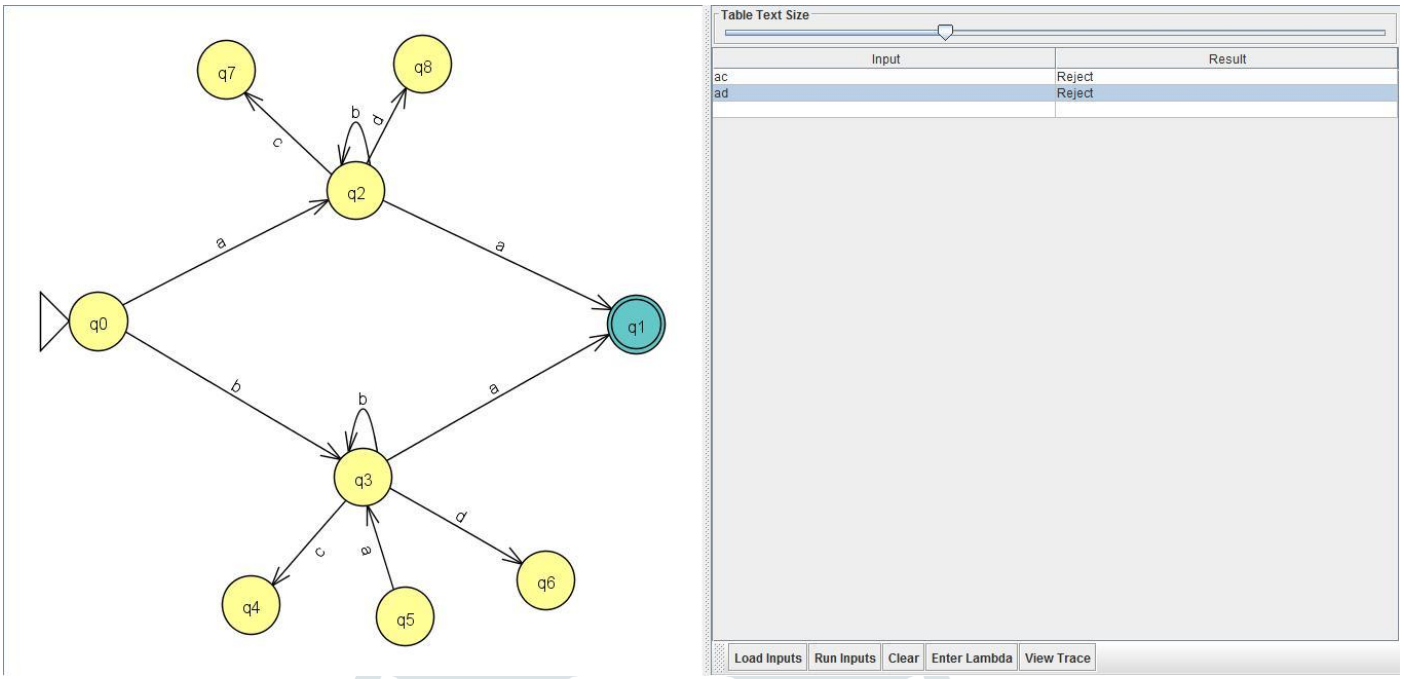

Figure 3.1.2: Deterministic finite automata with 'ac' and 'ad' input symbol.

In above figure, no any state generating accepting input symbol. 'ac' and 'ad' are rejecting input symbol. So we can see view trace by JFLAP simulator.
So both q7 and q8 are dead state.

Check for q3:

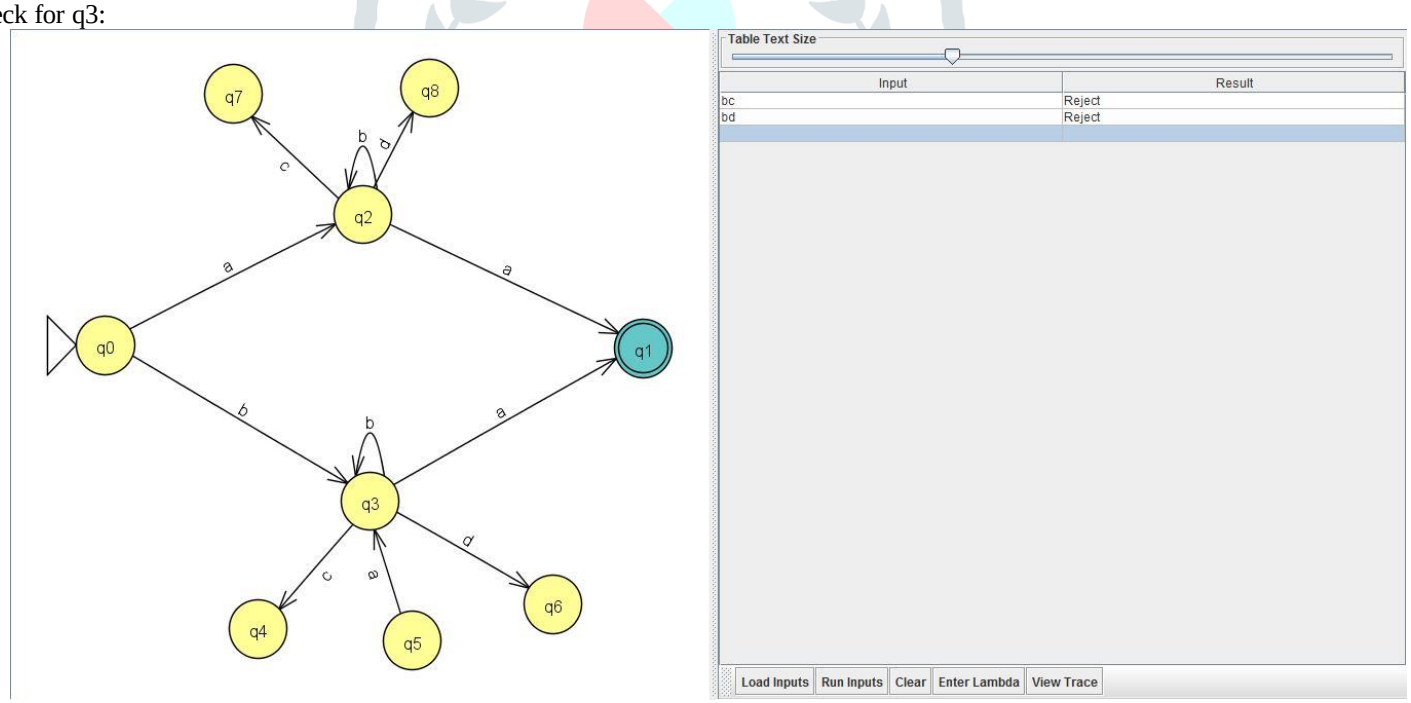

Figure 3.1.3: Deterministic finite automata with 'bc' and 'bd' input symbol.

In above figure, no any state generating accepting input symbol. 'bc' and 'bd' are rejecting input symbol. So we can see view trace by JFLAP simulator.
So both q_5, and q_6 are dead state. And finally q_2 and q_3 have same number of dead states but as we can find through proposed algorithm that state q_5 is unreachable state, so we have to remove state q_3 and merge the transition of q_3 in q_2 state.

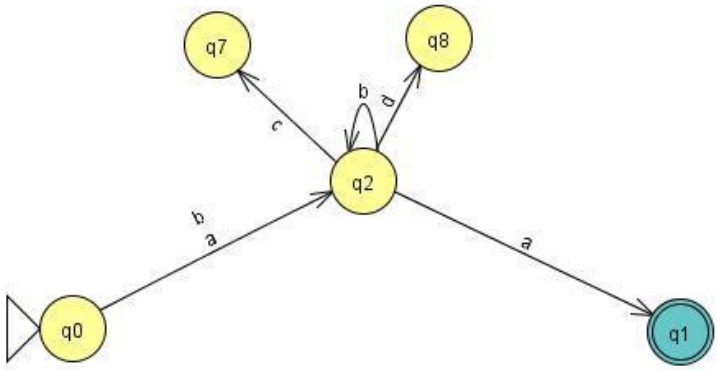

Figure 3.1.4: Deterministic finite automata with no any indistinguishable state.

Step3. Remove Dead state and unreachable state:

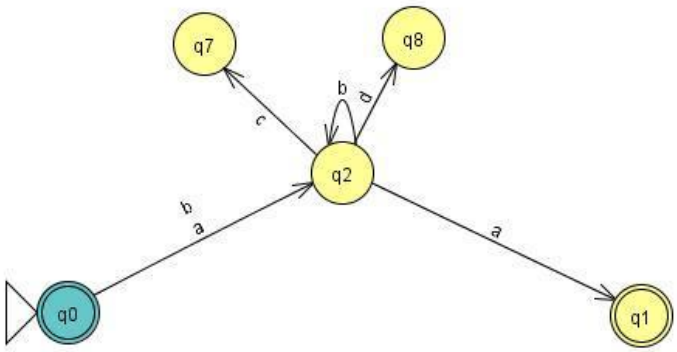

Figure 3.1.5: Deterministic finite automata with q_0 finalized state.

In above figure we finalize state q_0 because this state generating accepting input symbol with final state and take next input symbol.

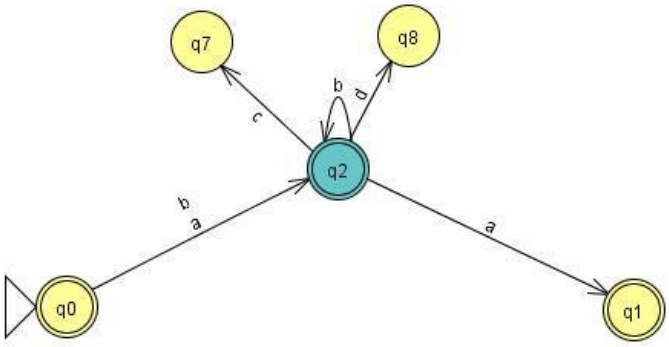

Figure 3.1.6: Deterministic finite automata with after finalized all the state including in accepting symbol.

In the figure, state q_1 is finalizing because this state including accepting input symbol 'ba', 'aa' again we will take next input symbol. If no any input symbol is accepting then we will take next possible input symbol. Again if no input symbols are accepting then repeat it whenever all possible input not finished. When all possible inputs are taken so we will remove all non-final state as shown in below figure. In above figure final state indicate accepting string generated by using these state and non- final state indicated no any string generated by using these state.

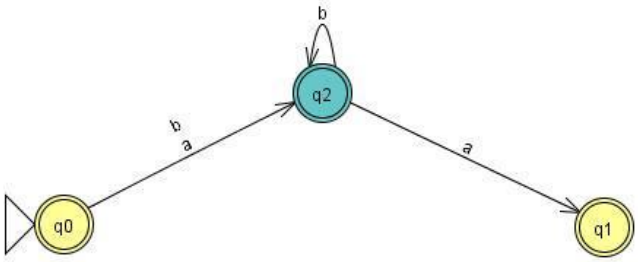

Figure 3.1.7: Deterministic finite automata without unreachable state and dead state.

In this figure 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 q_0 and final state was q_1 so these state are not same as a previous form and all updated label should be remove. As shown in below figure.

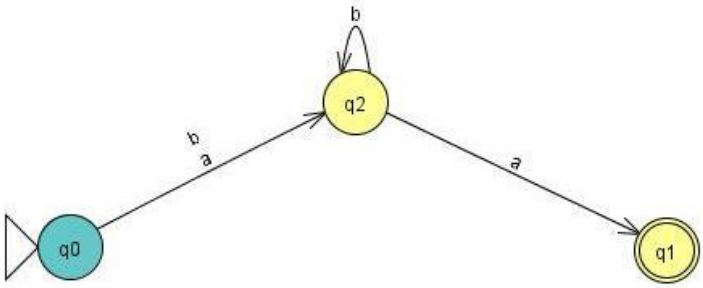

Figure 3.1.7: Deterministic finite automata with useful states

In above figure all useful state available means only whose state is available those are including in accepting string and all Dead state remove in this process. So this is a proposed technique for removing unreachable state, Dead state and indistinguishable state of deterministic finite automata.

3.2 EXAMPLE2

Let us take our problem statement example with are having dead state, unreachable state, and redundant states.

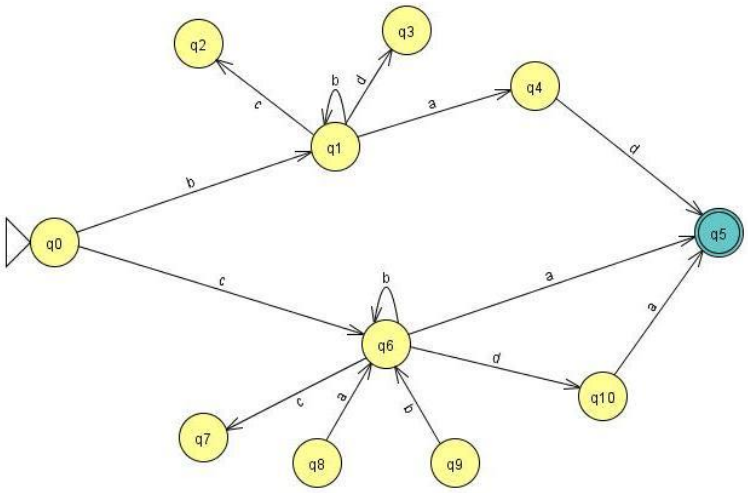

Figure 3.2.1: Deterministic finite automata with unreachable state, dead state and indistinguishable state.

In the above example, all steps of our proposed algorithm is utilized because this example do not contain direct indistinguishable states but having more complicated redundant states. Proceeding according to our proposed algorithm, we found states q_1 and q_6 as redundant states as follows:

Strings generated by q_1 ignoring loops and repeated strings is 'ad' and that by state q_6 is 'a' and 'da'.

As both these states are transited from same source and have self loops on same symbol, the strings generated by these states are compared. 'ad' and 'a' are same to one symbol and differ afterwards. Hence, according to our proposed algorithm and other steps of removal of dead state and unreachable states as that of previous example and hence we skip these procedures and the final DFA is as shown below.

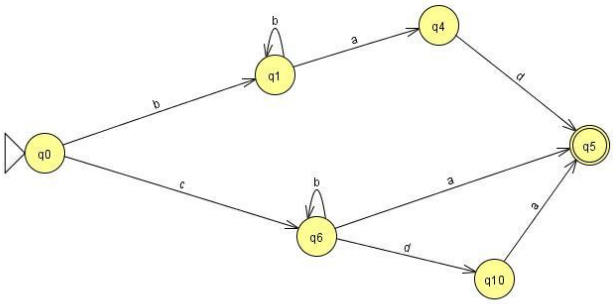

Figure 3.2.2: Deterministic finite automata with useful states

As we can see from the above two figures that the strings generated by these two DFAs are same.

Proceeding as our proposed algorithm hyper-minimization of DFA can also be achieved having some finite number of errors.

4. PROPOSED ALGORITHM

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

OUTPUT: A = (Q', Σ, δ', q_0', q_f') – Deterministic finite automata without useless and redundant state.

1. For (q ∈ Q) /*all state belong to given set of deterministic finite automata *\
2. Go to (q_i <- Initial State) /* go to initial state *\
3. If ($\delta_{i\text{-}Fi}$ ->q_f) /* this show transition function if any state reach to final state that means input string is accepted *\
4. If (δ (q_m, a) ∈ F = = (δ (q_n, a) ∈ F || δ (q_m, a) ∉ F = = (δ (q_n , a) ∉ F /* this show these state are indistinguishable. And n= 1,2,3,……; m= 1,2,3……; m ≠ n; a (input symbol) ∈ ∑*(non-empty finite set of input symbols) *\
5. THEN GOTO STEP 15
6. For (q ∈ Q) /*all state belong to given set of deterministic finite automata *\

7. Generate all the strings starting from each and every state that reach to final state of the DFA ignoring the repeated strings (eg. – self loops, loops via other states etc.) /* the number of strings will be finite as we do not consider strings generated from loops *\
8. END FOR
9. For each of the state q_i and $q_j \in \{Q - q_0\}$ (having the self loops and other loops on the same symbol (if any)), where $q_i \neq q_j$ and both the states q_i and q_j generated or transited from the same previous state.
10. Compare the first, second, third and so on...symbol of all the strings generated by q_i to the corresponding symbols of all the strings generated by q_j.
11. If first symbol matches
    THEN
12. If the string not matches after m symbols then make the transition as before. /* no redundant states exist*\
13. ELSE IF all symbols of all strings matches then GOTO STEP 18
14. END ELSE
15. END IF
16. ELSE CONTINUE
17. END ELSE
18. END IF
19. END FOR
20. If (q_m -> q_m next $\notin$ F > q_n -> q_n next $\notin$ F) /* this show q_n have more dead state.
21. q_n<-Remove, /* remove indistinguishable state and connecting input symbol from q_n to its previous state is merge with q_m state.*\
22. ELSE
    q_m<-Remove,/* remove indistinguishable state and connecting input symbol from q_m to its previous state is merge with q_n state.*\
23. Go to (q_i <- Initial State) /* go to initial state after removing the indistinguishable state for dead state *\
24. Else
    Go to (q_i <- Initial State) /* go to initial state *\
25. END ELSE
26. END IF
27. MAKE $q_{i\text{-}fi}$ <- Final State /* if any iteration for accepting state then all the state including in this iteration should be final state*
28. ELSE
    q_i -> q next /*if input string is not accepted then move to another state q next *\
29. END ELSE
30. END IF
31. END FOR
32. For (q' $\in$ q_f ') /*after checking all possible input string according to proposed technique all useful state should be final state*\
33. If (q' $\in$! q_f ') Then /* unreachable state*\
34. q' <- Remove /* remove unreachable state*\
35. END IF
36. END FOR
37. For (q' $\in$ q_f ')) /* after removing unreachable state all useful state present and all state shuld be final state *\
38. if($q_{initial}$ $\in$ q_f ') Then /* checking proposed initial state is final or not *\
39. $q_{initial}$ <- previous position /* if proposed initial state is final the it will form previous state*\
40. Else
    q' $\in$ q_f ' Then /* all proposed state is final state*\
41. q_f ' <- Previous position /* all proposed state become previous state*\
42. END ELSE
43. END IF
44. END FOR

5. CONCLUSION AND FUTURE WORK

It is very challenging in the automata theory to choose the useful state of Deterministic Finite Automata and hence it is also difficult at introductory level to understand and learn. In this paper, we remove almost each and every Indistinguishable State, Unreachable State, Redundant State and Dead State in DFA and hence the resultant DFA is almost minimal. Proposed algorithm is to find the useful state of the deterministic finite automata. JFLAP simulation for determine useless states and remove the states that are not useful in generating strings. Also it gives technique and algorithms, after selecting useful state minimize simply of deterministic finite automata.

In this paper all the useless states and indistinguishable states when it is attached to the useless states or some other states that lead to the final state but the removal of redundant states takes somewhat more time. So for future workc is work on this particular dimension to decrease the time taken in removing redundant states. We can work on minimization of deterministic finite automata in future time in order to extend our work.

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] Susan Rogers. Java Formal Language Automata Package (JFLAP), February 2006. http://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] Domenico Ficara, 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] Domenico Ficara, 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 saddle point approximation," Statistical Science, vol. 15, pp. 279–298, 2000.

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

[11] M. Procopiuc, O. Procopiuc, and S. Rodger, Visualization and Interaction in the Computer Science Formal Languages Course with JFLAP, 1996 Frontiers in Education Conference, Salt Lake City, Utah, p. 121125, 1996

[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] S. H. Rodger. Jap web site, 2011. 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.