A Novel Approach for construction of Minimal Deterministic Finite Automata - RC Algorithm

Ravirajsinh Chauhan[1], Chandan Trivedi[2]
[1]*School of Engineering, P. P. Savani University, Surat, Gujarat, India*
[2]*Department of Computer Science and Engineering, Institute of Technology Nirma University, Ahmedabad, Gujarat, India*

Abstract- Finite Automata are useful in wide variety of text processing tasks like internet search engines, construction of parsers, data validation, protocol analysis, Natural Language Processing, video games, CPU controllers, etc. The mathematical system that process a strings of Regular Language (RL) is called as Finite Automata (FA). By default FA is Deterministic Finite Automata (DFA) which includes finite number of states and transitions from all the states for each and every alphabet exactly once. There are many deterministic automata that accept a given language, among which there is a unique DFA that has a minimal number of states. This is called the Minimal Deterministic Finite Automata (MDFA) of the language. The DFA is minimal if it is free from equal states and unreachable states. In proposed algorithm, we have introduced a new approach in order to construct MDFA directly from infinite regular language which is free from equal states and unreachable states.
Keywords – Minimal DFA, Finite Automata, Minimization, RC Algorithm.

I. INTRODUCTION

Theory of Formal Languages provide the theoretical understanding for the study of different programming languages. The concepts of formal languages are widely used in the study of biological systems, networks of the computers, data compression and transmission, etc. The collection of the strings where we can put some restrictions and conditions in the formation of a string is known as Formal Language. According to Chomsky hierarchy, mainly 4 types of formal languages are there: Recursively Enumerable Language (REL), Context Sensitive Language (CSL), Context Free Language (CFL) and Regular Language (RL). All the formal languages have their specific advantages, uses and applications. In this paper we will focus on Regular Languages and their different representations. There are mainly three different mathematical representation of regular language: Finite Automata (FA), Regular Expression (RE) and Regular Grammar (RG). All three are equivalent to each other. Regular Expressions and Finite Automata are useful in wide variety of text processing tasks like internet search engines, construction of parsers, data validation, protocol analysis, Natural Language Processing, video games, CPU controllers, etc.

The Finite Automata is used to perform a predetermined sequence of tasks based on the occurring event. Examples are traffic lights, elevators, combination locks, etc. The FAs are of two types: FA with output and FA without output. Finite Automata may have outputs corresponding to each transition. There are two types of finite machines which generate output: Mealy Machine and Moore Machine. FA without output is characterized into two types: 1) Deterministic Finite Automata (DFA) and 2) Non Deterministic Finite Automata (NDFA/NFA).

DFA and NFA both consists of 5 tuples {Q: set of all the states, ∑: set of input symbols/alphabets, q0: initial state, F: set of final state, δ: Q ×∑ → Q for DFA and δ: Q ×∑ → 2Q for NFA (transition Function)}. The major difference between DFA and NFA lies in their transition function δ. In DFA, we can reach to exactly one state after reading a particular symbol from particular state. While in NFA, We can reach to any number of states (even zero) after reading a particular symbol from particular state. The transition path for each and every string is unique in DFA. The language accepted by DFA is Regular Language (The collection of strings formed using input alphabets is called language) [2]. There can be multiple DFAs possible for particular regular language. The DFA with minimum number of state is efficient among all possible DFAs for given regular language. The Myhill-Nerode theorem [1] states that among the many deterministic automata that accept a given language, there is a unique automata (excluding isomorphism) that has a minimal number of states. This is called the Minimal Deterministic Finite Automata (MDFA) of the language.

The process of detection and elimination of states whose presence or absence will not affect the language of automata is known as minimization of DFA. In general, the presence and absence of dead state, unreachable states and equal states will not affect the language of FA.

Equal states: Two states P and Q are said to be equal if both the transition δ (P, X) and δ (Q, X) goes to either final state or non-final state, ∀ X ∈ ∑*. Where ∑* = all possible strings from set of input alphabets ∑. Equal states are also called as non-distinguishable states.
Unreachable states: The state qun is called unreachable state if it is not reachable from the initial state q0 for any input string. i.e. δ (q0, X) ≠ qun, ∀ X ∈ ∑*. Where ∑* = all possible strings from set of input alphabets ∑.
Dead state: If we already know that the string is going to be rejected, then we create a rejecting state that is essentially a dead end. Once the machine enters a dead state, there is no way for it to reach an accepting state. The transition from dead state for every input alphabets happening to itself. i.e. δ (qD, X) = qD, ∀ X ∈ ∑. Where qD = Dead state and ∑ is set of all input alphabets.
The rest of the paper is organized as follows. Work related to Minimal DFA is explained in section II. Proposed RC algorithm and examples are presented in section III and section IV. Proof of correctness of proposed algorithm is presented in section V. Concluding remarks are given in section VI.

II. RELATED WORK

As the process of minimization of DFA, we need to remove dead state, unreachable states and equal states. But if we remove dead state from DFA then it becomes NFA. Hence if DFA contains dead state then it will be included in MDFA also. So the DFA which is free from unreachable states and equal states is known as Minimal DFA. Different authors have proposed variety of minimization algorithms to remove equal states and unreachable states. There exist numerous algorithms to minimize a deterministic automaton. Watson published a taxonomy on this topic [3]. Among the various possible constructions, Watson has used the result of Brzozowski's minimization algorithm [4] that it can take a nondeterministic automaton as input to design an algorithm which directly constructs a minimal deterministic automaton from a regular expression [5]. Hopcroft [6] has given an algorithm that computes the minimal automaton of a given deterministic automaton. Given a deterministic automaton A, Hopcroft's algorithm computes the coarsest congruence which saturates the set F of final states. It starts from the partition {F, FC} which obviously saturates F and refines it until it gets a congruence. These refinements of the partition are always obtained by splitting some class into two classes. The algorithm presented by Daciuk et al. in [7] is an incremental algorithm for the construction of a minimal automaton for a given set of words that is lexicographically sorted. In [8], Liu D. et al. have proposed efficient DFA minimization using backward depth information. J. Rot [9] has proposed coalgebraic Minimization of Automata by Initiality and Finality, where Hopcroft's algorithm and Brzozowski's algorithm are combined to get minimized DFA. In [10], S. Bhargava and G.N. Purohit proposed a method for constructing a minimal deterministic finite automaton (DFA) from a regular expression. Tulashiram B. Pisal and Archana A. Ghatule proposed method to give an easy way of learning and designing finite automata that accept a DFA which having different conditions specifically for starting and ending of the string [11]. E. A. Bondar and M. V. Volkov proposed approach of completely reachable automata [12]. In their approach for given DFA, one can easily decide whether or not it is completely reachable considering its power set automaton. A DFA is completely reachable if and only if every state Q is connected with every its non-empty subset by a directed path in the power set automata, and the latter property can be recognized by breadth-first search on power set automata starting at Q.
Hence, from the above study we can deduce that there is no comprehensive algorithm exist that can be used to eliminate both unreachable states and equal states at a time. There is one approach in which MDFA is directly constructed from given regular expression [10]. There is no such algorithm exist which can construct MDFA directly for given regular language. In proposed RC algorithm, we can directly construct MDFA which is free from both unreachable states as well as equal states at the same time. The proposed algorithm removes the dependency over the necessity of lengthy chain of conversion, that is, regular language → regular expression → NFA with ε-transitions → NFA without ε transitions → DFA → minimal DFA. Therefore the main advantage of the proposed minimal DFA construction algorithm is its minimal intermediate memory requirements and hence, the reduced time complexity.

III. PROPOSED RC ALGORITHM

In proposed algorithm, RC stands for Mr. Ravirajsinh Chauhan and Mr. Chandan Trivedi. The proposed algorithm is used to construct MDFA specifically for infinite regular language.
Notations:
∑: The set of all the alphabets (input symbols)
Q: The set of all the states.
F: The set of all the final states.
NF: The set of all the non-final states.
D: Boolean variable (Initially D = FALSE). Used to check whether dead state is part of FA or not.

q0: Initial State.

3.1 Algorithm

Step-1: List out the strings of given language starting with minimum length string(s).

Step-2: Create Finite Automata for minimum length string(s) and make the last state as final state accordingly and include it into set F and add other states into set NF. Name all the states as qi where i = 0, 1, 2…, N. Denote q0 as initial state.

If there are multiple strings available with minimum length then make sure that you create FA with minimum number of states considering given language in mind. Remember, FA must accepts all the strings which are part of given language and it must rejects all the strings which are not part of given language.

Step-3: Read the alphabet X from ∑ for the state qi exactly once and make transition as follows:

(3a) Find the minimum length string Sqipre required to reach at the state qi from initial state q0. If multiple values are possible for string Sqipre then consider all possible combinations while performing concatenation in step-3 (3c).

(3b) read the current alphabet X from ∑.

(3c) add the minimum length string Sqipost such that string Sqi will be the part of the given regular language. If multiple values are possible for string Sqipost then consider all possible combinations while performing concatenation. Where Sqi = Concatenation of strings Sqipre, X and Sqipost (i.e. S= Sqipre | X | Sqipost). In case of multiple Sqipre and/or Sqipost, there will be multiple values of Sqi.

(3d) the transition from qi for alphabet X is:

(i) If it is not possible to create a string which is part of given regular language then check whether dead state is already part of FA or not.

If D == FALSE then

Create new state qD (dead state) and change value of D = TRUE. The transition from qi for alphabet X is: qi × X → qD. The transition from qD for all the alphabets from ∑ is to qD only. Add qD in set NF. Go to Step-4.

Else

The transition from qi for alphabet X is: qi × X → qD and go to Step-4.

(ii) The transition X from state qi is: qi × X → qj

Where, qj = the state from where we can reach to any of the final state(s) after reading string Sqipost.

If qj is not available then create new state qj and add it into respective set (F or NF).

If qj leads to Dead state then instead of going on qj, create new state qk and add qk in the same set as of qj (F or NF). Create relation between qj and qk so that whenever the transition leads to qj, it will be redirected to qk.

(iii) If we have multiple values for string Sqi, then consider all Sqis while deciding transition. If it differs in the transition state qj (i.e. if we get multiple options for qj) then split the state qi accordingly and repeat step-3 again. Perform rollback if required.

In all other cases (if exist), it is not possible to create minimal DFA for the given regular language using RC algorithm. In such cases stop RC algorithm and follow traditional methods for DFA construction and apply any of the minimization algorithm to get minimal DFA.

Step-4: Repeat step-3 such that transition occurs for all the states (qi) from Q with each and every alphabet(X) from ∑ exactly once.

3.2 Special Cases

3.2.1 Complement of DFA:

If we need to construct MDFA where the nature of regular language is negative (i.e. if negative word such as "not" is used) then first construct MDFA of its opposite regular language (the language obtained after removal of negative word) using RC algorithm. After that obtain complement of constructed MDFA by interchanging final and non-final states (i.e. Interchange elements of F and NF) to get desired MDFA.

For Example: Construct the MDFA that accepts all the strings of 0's and 1's where every string does not contain a substring 101 (Here does not contain shows negative nature of given regular language). So, in such cases first construct MDFA that accepts all the strings of 0's and 1's where every string contains a substring 101 (opposite regular language) using RC algorithm, say FA1. Interchange final and non-final states of FA1, say FA2. FA2 will be the final MDFA which accepts given regular language.

3.2.2 Compound Automata:

Let L is the language with the conditions C1, C2, C3... CN. Let Li is the sub language satisfying the condition Ci (i.e. Li contains all the strings which satisfies the condition Ci). Let M1, M2, M3… MN are FAs for L1, L2, L3… LN respectively. The automata which is obtained by taking composition of M1, M2, M3… MN is called as

compound automata. In such case if you feel difficulty in direct MDFA construction then construct separate DFA and apply traditional method to find compound automata and minimize it using any minimization method.

IV. EXAMPLES

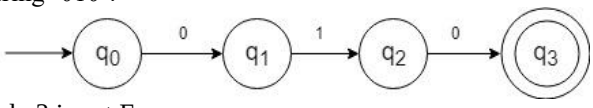
4.1 Construct MDFA that accepts all the strings of 0's and 1's where every string ends with substring '010'.

Construction of MDFA using RC algorithm:

Step-1: List out the strings of given language starting with minimum length string(s).

L = {010, 0010, 1010, 00010, 01010, 10010, 11010 …}

Step-2: Construct MDFA for string '010':



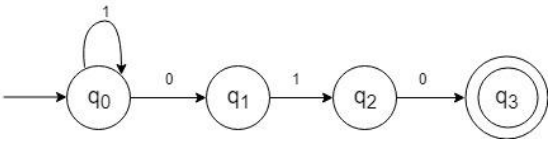
Add q0, q1 and q2 in set NF and q3 in set F.

Step-3: Read alphabet '1' from state q0:

According to Step 3a, 3b and 3c: Sq0pre= Null, X= 1 and Sq0post=010. So, Sq0= 1010.

According to Step 3d: The transition 1 from state q0 is: q0 × 1 → q0

Because the state from which we can reach to final state after reading Sq0post=010 is q0 only.



Step-4 Repeat step-3 such that transition occurs for all the states (qi) from Q with each and every alphabet(X) from ∑ exactly once.

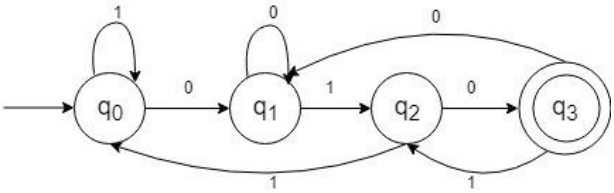
Step-3 for state q1 with alphabet '0': Sq1pre= 0, X= 0 and Sq1post=10. So, Sq1= 0010. So, the transition 0 from state q1 is: q1 × 0 → q1.

Step-3 for state q2 with alphabet '1': Sq2pre= 01, X= 1 and Sq2post=010. So, Sq2= 011010. So, the transition 1 from state q2 is: q2 × 1 → q0.

Step-3 for state q3 with alphabet '0': Sq3pre= 010, X= 0 and Sq3post=10. So, Sq1= 010010. So, the transition 0 from state q3 is: q3 × 0 → q1.

Step-3 for state q3 with alphabet '1': Sq3pre= 010, X= 1 and Sq3post=0. So, Sq1= 01010. So, the transition 1 from state q3 is: q3 × 1 → q2.

The final MDFA for given language is:



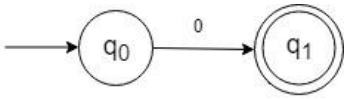
4.2 Construct MDFA that accepts all the strings of 0's and 1's where every string starts and ends with alphabet '0'.

Construction of MDFA using RC algorithm:

Step-1: List out the strings of given language starting with minimum length string(s).

L = {0, 00, 010, 000, 0000, 0010, 0110 …}

Step-2: Construct MDFA for string '0':

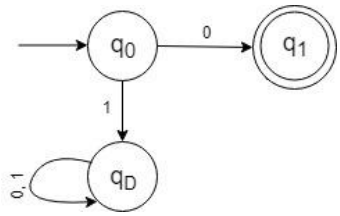


Add q0 in set NF and q1 in set F.

Step-3: Read alphabet '1' from state q0:

According to Step 3a, 3b and 3c: Sq0pre= Null, X= 1. But it is not possible to find any Sq0post such that string Sq0 can be part of given language L.

According to Step 3d: Check If D == FALSE (true) then Create new state qD (dead state) and change value of D = TRUE. The transition from q0 for alphabet '1' is: q0 × 1 → qD. The transitions from qD for alphabets '0' and '1' are to qD only. Add qD in set NF. Go to Step-4.



Step-4 Repeat step-3 such that transition occurs for all the states (qi) from Q with each and every alphabet(X) from ∑ exactly once.

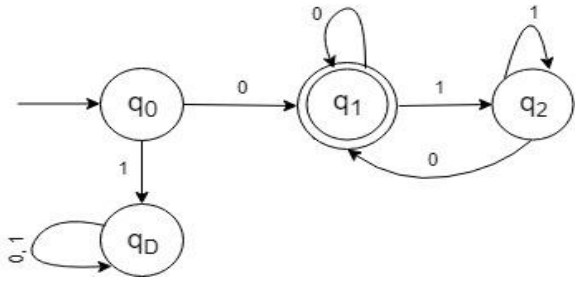
Step-3 for state q1 with alphabet '0': Sq1pre= 0, X= 0 and Sq1post=Null. So, Sq1= 00. So, the transition 0 from state q1 is: q1 × 0 → q1.

Step-3 for state q1 with alphabet '1': Sq1pre= 0, X= 1 and Sq1post=0. So, Sq1= 010. So, the transition 1 from state q1 is: q1 × 1 → q0. But state q0 leads to dead state qD. So instead of going on q0, create new state q2 and make relation between q2 and q0. Hence, the transition 1 from state q1 is: q1 × 1 → qD.

Step-3 for state q2 with alphabet '0': Sq2pre= 01, X= 0 and Sq1post=Null. So, Sq1= 010. So, the transition 0 from state q2 is: q2 × 0 → q1.

Step-3 for state q2 with alphabet '1': Sq2pre= 01, X= 1 and Sq1post=0. So, Sq1= 0110. So, the transition 0 from state q2 is: q2 × 0 → q2. (q0 is not selected as it leads to dead state)

The final MDFA for given language is:



V. PROOF OF CORRECTNESS

The finite automata constructed using RC algorithm is Deterministic Finite Automata as we make transition from all the states of Q with all the alphabets of ∑ exactly once (Refer Step-3 and Step-4 of RC algorithm) (Observation-1). The DFA is minimal if it is free from Equal States and Unreachable states. In proposed RC algorithm we are constructing directly minimal DFA from the regular language. In Step-2 of RC algorithm, we construct FA for minimum length string so it creates N+1 states, if length of smallest string is N>=0, which can't be reduced further (Observation-2).

In step-3(3a) of proposed algorithm, Sqipre is the minimum length string which is required to reach at current state qi from initial state q0. So the current state qi is reachable from initial state q0 (Observation-3). According to Step-4, in order to create DFA we need to follow step 3 for making the transition from each and every state with all the alphabets exactly once. So the string Sqipre is created for all the possible pairs of state qi and alphabet X exactly once (Observation-4). From Observation-3 and Observation-4, we can conclude that all the states are reachable from initial state q0. Hence, it is proved that the minimal DFA constructed using RC algorithm is free from unreachable states (Observation-5).

Initially we construct FA for minimum length string with N+1 number of states, Where N = length of smallest string. So, all N+1 states are unique. Step-3 is used to complete remaining transitions. While performing transition for state qi with alphabet X, we used to find Sqi which is concatenation of Sqipre, X and Sqipost (Refer Step-3 (3c)). If it is not possible to find Sqi which is part of given regular language then we need to construct dead state. But before constructing dead state we check whether dead state is already present in the FA or not using (Check D== False) condition. So there will be only one dead state (if available) in the final MDFA (Obervation-6). According to Step-3 (3d), the transition X from state qi is: qi × X → qj. Where, qj = the state from where we can reach to any of the final state(s) after reading string Sqipost. If such qj is not available in the system then and only then we create new state qj and add it into respective set (F or NF). In some cases if qj leads to dead state then we create new state qk from where we can reach to final state after reading same string. In such case, qj and qk will never be the same as qj leads to dead state after reading some string Sd and once we enter in the dead state the string will never get accepted. While we may reach to any of the final state after reading the same string Sd from state qk. Hence, the

states qj and qk are not equal states (Observation-7). From Observation-6 and Observation-7, we can prove that the FA constructed using RC algorithm is free from equal states (Observation-8).
Hence from Observation-5 and Observation-8, we conclude that the proposed RC algorithm ensures elimination of both unreachable states and equal states at a time from constructed FA which turned out to be MDFA.

VI. CONCLUSION

The FA constructed using RC algorithm is MDFA as it is free from equal states and unreachable states. In proposed algorithm we construct MDFA directly from RL which removes the dependency over the necessity of lengthy chain of conversion, that is, regular language → regular expression → NFA with ε-transitions → NFA without ε transitions → DFA → minimal DFA. Therefore the main advantage of the proposed RC algorithm is it eliminates intermediate memory requirements and reduces the time and space complexity. The proposed algorithm is specifically applicable for the construction of MDFA from infinite regular language. In future, we are going to propose extension of RC algorithm to cover finite regular languages also. The proposed algorithm is not fully automated/intelligent algorithm. We need to apply human intelligence at some specific places.

VII. REFERENCES

[1] J. E. Hopcroft and J. D. Ullman. Introduction to Automata Theory, Languages and Computation. Addison-Wesley Publishing Company, 1979.
[2] S.C. Kleene, Representation of events in nerve nets and finite automata, Automata Studies, Princeton Univ. Press (1956) 3–42.
[3] B. Watson, Taxonomies and Toolkits of Regular Languages Algorithms, PhD thesis, Eindhoven University of Technology, The Nederlands, 1995..
[4] C. Carrez, On the Minimalization of Non-deterministic Automaton, Research Report, Laboratoire de Calcul de la Faculté des Sciences de l'Université de Lille, 1970.
[5] B. Watson, Directly Constructing Minimal DFAs : Combining Two Algorithms by Brzozowski, in S. Yu and A. Paun, eds, CIAA 2000, London, Ontario, Lecture Notes in Computer Science, 2088(2001), 311–317, Springer.
[6] J. E. Hopcroft. An nlog n algorithm for minimizing states in a finite automaton. In Z. Kohavi and A. Paz, editors, Theory of Machines and Computations, pages 189–196. Academic Press, 1971. 2, 5, 8, 12
[7] J. Daciuk, S. Mihov, B. W. Watson, and R. E. Watson. Incremental construction of minimal acyclic finite-state automata. Comput. Linguist., 26(1):3–16, april 2000. 3, 20, 24, 26.
[8] Liu, D., Huang, Z., Zhang, Y., Guo, X., Su, S.: Efficient deterministic finite automata minimization based on backward depth information. PloS ONE 11(11), e0165864 (2016).
[9] J. Rot. Coalgebraic minimization of automata by initiality and finality. Elect. Notes in Theor. Comp. Sci., 325:253–276, 2016.
[10] S. Bhargava, G.N. Purohit, "Construction of a minimal deterministic finite automaton from a regular expression", International Journal of Computer 1845 Applications 15 (4) (2011) 16–27 (Published by Foundation of Computer 1846 Science).
[11] Tulashiram B. Pisal and Archana A. Ghatule, "AUTO GENERATION OF DFA WITH STARTING AND ENDING CONSTRAINTS", International Journal of Advances in Electronics and Computer Science, ISSN: 2393-2835, Volume-3, Issue-3, Mar.-2016.
[12] Bondar, E.A., Volkov, M.V.: Completely reachable automata. In: Câmpeanu, C., Manea, F., Shallit, J. (eds.) DCFS 2016. LNCS, vol. 9777, pp. 1–17. Springer, Cham (2016).