

Partition Method for DFA Minimization

Abhishek Singh¹, Dr. Satwinder Singh²

¹M. Tech (Computer Science and Technology), Central University of Punjab, Bathinda.

²Assistant Professor, Dept. of Computer Science and Technology, Central University of Punjab, Bathinda.

Abstract - It is usually perceived that there are many DFAs which have those states which are not necessary for the machine. Removal of those states cannot do any harm to machine. In this paper, our priority is to merge those states who are equal in terms of transition. A method is proposed through which it can be done.

Key Words: DFA, Minimization.

1. INTRODUCTION

DFA is a mathematical conception, due to which strings are operated by taking an input and after processing they give some output in the form of string. The language accepted or recognized by DFA is Regular Language.

There are finite number of states in DFA, in which, there is at-most one outgoing edge/transition from one state to another. It has only one initial state but consists of a finite number of final states, means final state can be more than one.

DFA minimization is a process in which DFA is converted to an equivalent another DFA in which number of states are particularly less. All states are compared through an algorithm and states those have same transitions are merged as they give equivalent output for an equivalent input. The main advantages for a minimal DFA is as follows:

1. A minimal DFA executes faster in terms of time.
2. As number of states go less, the cost for designing a machine also go less.
3. Having less number of states helps in designing the less complex structure of a machine.

There are three types of non-productive states which can be merged or removed when minimization algorithm is applied on a given DFA. Two of them don't change the structure of DFA but one of them changed it, however, language recognized by the DFA will be the same after minimization. The explanation of these non-productive states is:

1. **Unreachable State:** It is a state which cannot be reached directly or indirectly from initial state. If it is removed then, no changes are done in the language of machine as well as structure of machine.
2. **Dead State:** It is a state from which we cannot come back to any state and it will go to itself number of times as it will trap in itself. If it is removed, then no changes are done in the language of machine but there is a change in structure of machine. The DFA will be converted into NFA after removing dead state.
3. **Equal State:** Two states are said to be equal when both of states go to the final or non-final state for the same input transition. If both are go to the final or non-final state, then they can be merged as give same output for the same input. If they are merged, there is no change in language of machine as well as structure of machine.

There is numerous minimization algorithm present to convert a DFA into minimal DFA. In this paper, we proposed a partition algorithm from which one can minimize the DFA.

2. PROPOSED WORK

In our proposed algorithm, we take partition of states for minimization in a particular pattern. After applying the method, equivalent states are merged and the process is continued till all the states are compared for minimization and we get a minimized DFA.

Algorithm

Input: Deterministic Finite Automata

Output: Minimized Deterministic Finite Automata

Algorithm:

- Take a DFA with Initial State Q_i and Final State Q_f .
- Take Partition in the form of $\lceil (\text{Total number of states}/2) + 1 \rceil$. Take floor value of division.
- Take first partition starting from initial State that is Q_0 .
- Check in the partition whether any state equivalent or not.
- If two state are equivalent then merge them and move to next partition.
- If any state from the current partition is not equivalent then move to next partition.
- States who have already merged cannot be the part of next partition but they will count in total number of states.
- Take next partition starting from next state in incremental numeric form for example: if in first partition if Q_0 is merge other state take next partition from Q_1 if Q_0 and Q_1 both are equivalent to some other states take partition from Q_2 and so on.
- Continue this process until all the states are covered.
- For last partition take the remaining ones
- After this process all the equivalent states are merged and DFA will become minimized DFA.

For equivalent state, here we consider that two state P and Q are said to be if both P and Q $\delta(P, x)$ and $\delta(Q, x)$ goes to final state or non-final state for every input string x.

3. EXPERIMENTS AND RESULTS

The experiments are done in the JFLAP tool in default settings.

Input: Given DFA

Table 1: Transition Table of DFA

States	0	1
Q_0	Q_0	Q_2
Q_1	Q_3	Q_1
Q_2	Q_3	Q_1
Q_3	Q_1	Q_4
Q_4	Q_1	Q_4

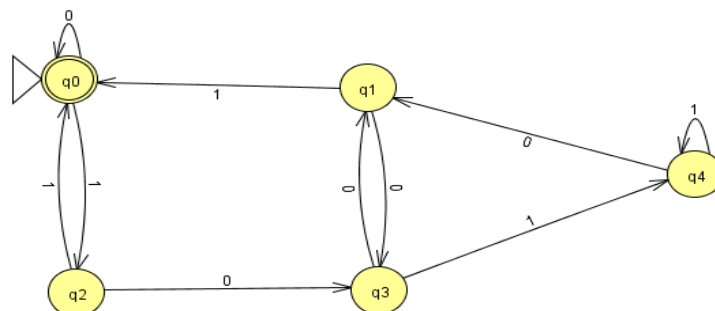


Fig 1: Transition Diagram of DFA

Now according to Algorithm, first partition is taken according to the formula and is in between three states and the table is given below

States	0	1
Q ₀	Q ₀	Q ₂
Q ₁	Q ₃	Q ₁
Q ₂	Q ₃	Q ₁

Table 2: First Partition Table

In this table, it is clear that states Q₁ and state Q₂ are equivalent since in both Q₁ and Q₂, transition of Q₁ on 0 goes to Q₃ and on 1 it goes to Q₁ && Q₂ on 0 goes to Q₃ and on 1 it goes to Q₁. Hence, they are equivalent state. So, we merge them (Q₁₂).

Now for the next partition we start from the state Q₃ the table is given below.

States	0	1
Q ₃	Q ₁	Q ₄
Q ₄	Q ₁	Q ₄
Q ₀	Q ₀	Q ₂

Table 3: Second Partition Table

Here in this partition, it is clear that state Q₃ and state Q₄ are equivalent states. Transition of Q₃ on 0 is Q₁ and on 1 it is Q₄ && Q₄ on 0 is Q₁ and on 1 it goes to Q₄.

Now, according to algorithm all the states cover we stop here and make a minimized DFA with the help Of the JFLAP tool. Now our minimized DFA is given below:

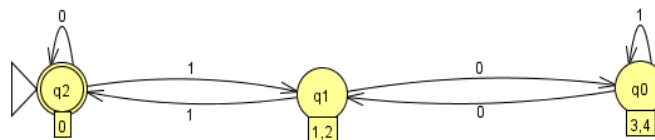


Fig 2: Minimized DFA

After applying partition, we get minimized DFA.

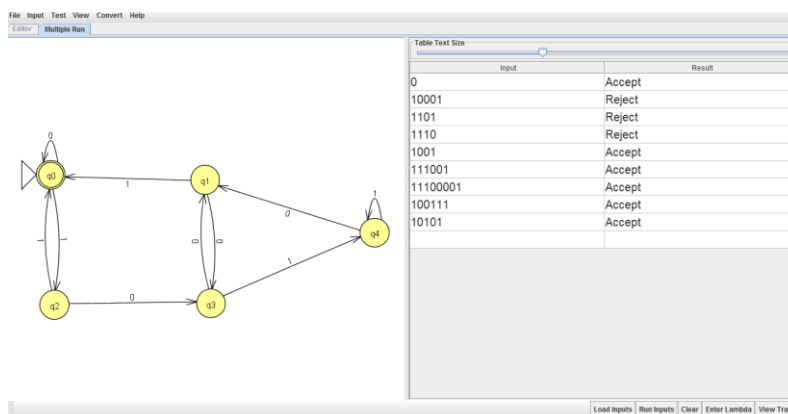


Fig 3: Multiple Run of Strings in DFA

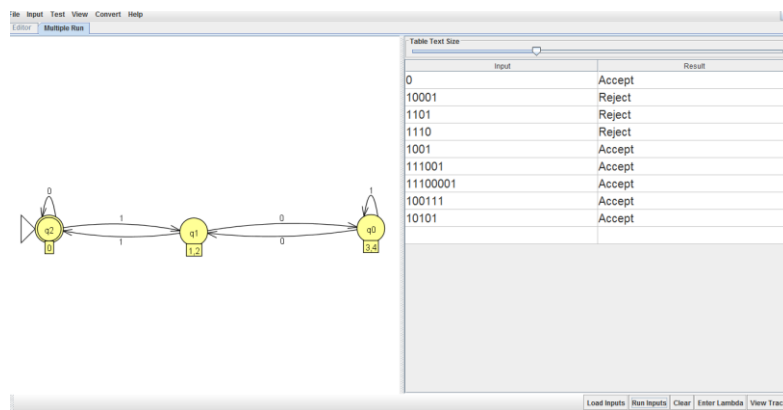


Fig 4: Multiple Run of Strings in DFA

Figure 3 and figure 4 represents the strings that are accepted before and after minimization on given DFA respectively. It is clear from both the figures that there is no change in language of DFA, strings that are recognized by DFA before minimization are also recognized even after applying minimization.

4. CONCLUSION

In this experiment, we proposed a minimization partition method from which by partitioning the states according to the formula one can minimize the DFA by merging equivalent states and removing dead and unreachable states. All the experimentation is done through JFLAP tool. After the minimization, it is checked that the DFA is successfully minimized with no change in structure and accepting language by running the few examples on DFA.

REFERENCES

- [1] lmeida, M., Moreira, N., & Reis, R. (2007). On the performance of automata minimization algorithms. *In Proceedings of the 4th Conference on Computation in Europe: Logic and Theory of Algorithms*, pp. 3-14.
- [2] Amit Kishor Shukla, A. S. (December 2015). State Minimization Approach in Deterministic Finite Automata using Inefficient State Elimination Approach. *International Journal of Advance Research in Computer Science and Management Studies*, Vol 3, Issue 12, pp 20-28.
- [3] Basten, H. J. S. (2011). Ambiguity detection methods for context-free grammars (*Doctoral dissertation, University of Amsterdam*).
- [4] Berry G. and Sethi R. (1986), From Regular Expressions to Deterministic Automata, *Theoretical Computer Science*, Vol. 48 pp. 117-126.
- [5] Bruggemann Klein A. (1992), "Regular Expressions into Finite Automata," *Springer link Lecture notes in Computer Science*, Vol. 583 pp. 87-98.
- [6] Champarnaud, J. M., Ponty, J. L., & Ziadi, D. (1999). From regular expressions to finite automata. *International journal of computer mathematics*, Vol 72(4), pp 415-431.
- [7] Chang, C. H., & Paige, R. (1992). From regular expressions to dfa's using compressed nfa's. *In Annual Symposium on Combinatorial Pattern Matching*, pp. 90-110. Springer, Berlin, Heidelberg.
- [8] Chang, C. H., & Paige, R. (1997). From regular expressions to DFA's using compressed NFA's. *Theoretical Computer Science*, Vol 178(1-2), pp 1-36.
- [9] Chhabra, T., & Kuymar, A. (2012). New heuristics for conversion of Deterministic finite automata to Regular expression. *International Journal of Advanced Research in Computer Science*, 3(4).
- [10] Choubey, A., & Ravi, K. M. (2013). Minimization of deterministic finite automata with vague (final) states and intuitionistic fuzzy (final) states. *Iranian Journal of Fuzzy Systems*, 10(1), 75-88.
- [11] Gruber, H., & Holzer, M. (2015). From finite automata to regular expressions and back—a summary on descriptonal complexity. *International Journal of Foundations of Computer Science*, Vol 26(08), pp 1009-1040.