ISSN No:-2456-2165

A Critical Analysis of Performance and Efficiency of Minimization Algorithms for Deterministic Finite Automata

Shweta
B.Tech
Department of Computer Science and Engineering
SRM Institute of Science and Technology
Chennai, Kattankulathur-603203
INDIA

K.Senthil Kumar
Asst. Prof(S.G)
Department of Computer Science and Engineering
SRM Institute of Science and Technology
Chennai, Kattankulathur-603203
INDIA

Abstract—This paper deals with the problem of minimization of Deterministic Finite Automata. There are various approaches available for converting the DFA into its minimized DFA for the given input strings. The most efficient algorithm for doing the minimization is the Hopcroft's Minimization Algorithm which aims at removing all the states which are unreachable i.e. which cannot lead to the goal state with the given set of input symbols. Thereafter, removing or merging all the equivalent states such that the resultant automaton will only have distinguishable states and this automaton would be the minimized automaton for the given input deterministic finite automata. Any two deterministic finite automata will have the same minimized DFA if they represent the same regular language. Here in this project I have attempted to implement the Hopcroft's algorithm with some parallelism in C language keeping the average time complexity to be logarithmic.

Keywords:- Deterministic finite Automata, NFA, compiler, transition function, equivalent states, distinguishable states.

I. INTRODUCTION

In compiler design, there are two types of automata which are used to accept strings which are accepted by a particular regular language NFAs (or) Non-Deterministic Finite Automata and DFAs (or) Deterministic Finite Automata. NFAs are not widely used because they can have multiple states to move to for a given state and same input symbol and hence they might not lead to the goal state for the input string. On the other hand, we have DFAs which have only single transition from a given state to the next state for the given input symbol, and hence they are unambiguous and give reliable output for the regular language accepted by them. Also, DFAs are very useful in some of the real world implementations such as doing lexical analysis, pattern matching, traffic sensitive traffic lights, vending machines, compiler design, logic optimization of various programs, verification of protocols etc. Therefore, to find a minimum DFA from a given DFA is very useful as they make the implementation of automata in real world more efficient.

DFAs can be defined by five tuples: <Q, Σ, δ, Qo, F>

Q: finite set of all states present in the language

Σ: finite set of all input symbols which could be accepted by the language

δ: transition function (δ=Q* Σ → Q)

Qo: start state of the automata

F: finite set of all goal\accepting states (F ⊆ Q)

- Equivalent States: the states which have same transition from one state to the next state for all strings.
- Distinguishable States: the set of states which have different transition from current state to the next state for the given set of inputs and thus are considered as different state. A DFA with only set of all distinguishable states is considered as minimized DFA.
- Partition: The set which comprises of subsets of all equivalent states as one subset which can be later merged to a single state to minimize the DFA with further refinement of the partition based on the backward depth information.

Example: Regular Expression with its equivalent NFA, DFA and Transition diagram.

- Regular Expression: (a\b)*abb
- Equivalent NFA:

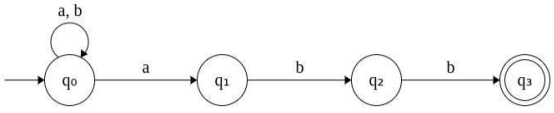

Fig. 1:-NFA for (a\b)*abb

- *Equivalent DFA:*

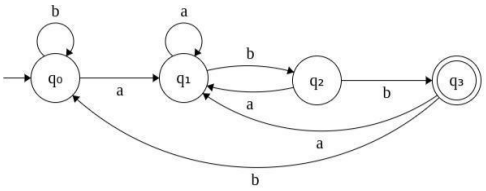

Fig. 2:- DFA for (a/b)*abb

State	a	b
->q_0	q1	q0
q_1	q_1	q_2
q_2	q_1	q_3
*q_3	q1	q0

Table 1: Transition Table for (a/b)*abb for DFA

The above example gives a brief about the construction of Non-Deterministic Finite Automata (fig-1), Deterministic Finite Automata (fig-2) and the transition table for the DFA (table-1). In the case of NFA there arises an ambiguity in state 'q_0' on the input symbol 'a' if whether to stay in the same state or to move to the next state that is 'q_1'. While, there is no such ambiguity in the case of Deterministic Finite Automata.

This paper is arranged in the following order i.e Section-II enlists some of the real-life implementations of Deterministic Finite Automata, Section-III discusses some of the implemented methods of DFA minimization while Section-IV gives the analysis of various existing algorithms based upon the above mentioned DFA minimization techniques.

II. APPLICATIONS OF DFAS

- Verification of protocols.
- Recognition of patterns.
- Searching for viruses
- Recognition of speech.
- CPU controllers.
- Text Parsing etc.

III. VARIOUS TECHNIQUES AVAILABLE FOR MINIMIZING A DFA INTO ITS MINIMIZED DFA

There are various methods available which are used to convert a DFA into its Minimized DFA. Any of the methods can be chosen based upon the application and type of DFA to be used.

A. Partition of sets method:

In this method there are two properties which are associated with the DFA i.e. states and their equivalence. This is an iterative approach in which the partition is refined each time
such that the states belonging to the same set in the final iteration are considered to be equivalent states and can be replaced by a single state.

Suppose there are two states (p,q), they will be considered as equivalent and can be replaced by a single state in the DFA if:

$$\delta(p,w) \in F \Rightarrow \delta(q,w) \in F \qquad (1)$$

and,

$$\delta(p,w) \notin F \Rightarrow \delta(q,w) \notin F \qquad (2)$$

where, $w \in \Sigma$

if |w|=0, then (p,q) are said to be 0 equivalent similarly, |w|=n, then (p,q) are said to be n equivalent where n=1,2,3,..

The zero equivalent state puts the final states in one state and all the non-final states in one set. After which every state is checked for its equivalency if for any input string, it goes to a state belonging to a different set. If yes, they are considered to be distinguishable states and are partitioned into different states

Otherwise, they are considered as equivalent states and can be replaced by a single state. The minimized DFA is finally constructed after removing all the unreachable and replacing all the equivalent states by a single state.

B. Table filling method:

This procedure constructs a minimized DFA based upon the table which is drawn for all pairs of states. Then all the pairs are marked where $p \in F$ and $q \notin F$. If there are any pairs (p,q) left such that [δ(p,w) and δ(q,w)] is marked, then mark [p,q]; where, $w \in \Sigma$. This step is repeated until there are no more pairs left to be marked in the table. The pairs which are not marked still are considered as equivalent states and can be replaced by a single state. After replacing all the unmarked states in the table by a single state minimized DFA is constructed.

C. A parallel DFA minimization method:

This is one more minimization method which provides a simpler implementation of the Moore's Algorithm. It makes use of O(n\logn) processors to obtain the average time complexity of O(log n), where n denotes the number of

states in the DFA to be minimized. In this method, the inner for loop is parallelized which iterates over the set of states. The labels which are obtained by this loop are hashed using a parallel algorithm for obtaining new block numbers for the states.

D. *Minimized DFA construction based upon Backward Depth Information- A hybrid approach:*

This a hybrid approach for DFA minimization based upon the backward depth information and hash tables. This method is divided into two phases: (i) backward depth information is built by partitioning the state set of DFA into different blocks, then (ii) hash tables are used for the refinement of the state set.
This method also has a naïve time complexity of $O(n^2)$, but this method provides a greater generality because it is independent of the topological complexity of the Deterministic Finite Automata. It is also faster than Hopcroft's Algorithm and provides greater scalability.

IV. ALGORITHMS BASED UPON THE ABOVE TECHNIQUES

There are various algorithms based upon the above mentioned techniques. This section provides a detailed analysis of these algorithms and their performance comparison for different types of DFAs is given below:

A. *Hopcroft's Algorithm*

Hopcroft's algorithm is based upon the refinement of partition of sets. It is the most widely used algorithm because of its worst case time complexity of O(n log n), where n: number of states in the DFA. It is based upon the successive refinement of sets which finally lead to the set of subsets comprising of distinguishable states which can be considered as the states of the minimized DFA. The only disadvantage with this algorithm is that it is not yet extended to deal with incomplete DFAs.

B. *Moore's Algorithm*

This algorithm is similar to Hopcroft's algorithm but it is easier to implement. It is also based upon the refinement of partition and terminates when no further refinement is possible. It starts with removing all the unreachable states and then starts constructing partition with set of 0 equivalent states. But the disadvantage of using this algorithm is that it has its worst case time complexity of $O(n^2)$ where, n is the number of states. This complexity is much higher than that of Hopcroft's Algorithm.

C. *Brzozowski's Algorithm*

This Algorithm is based upon the power-set construction method. In this algorithm, the edges if the DFA are reversed which produces a Non-deterministic Finite Automata for the original regular language after which only the reachable states are considered which leads to a minimal Deterministic Finite Automata. This algorithm works in the order of

$$D(R(D(R(A)))) \quad (3)$$

Where,
A- Automata to be minimized

D(A)- Determination of A by subset construction method

R(A)- Reversal of A

D. *Revuz's Algorithm*

This algorithm is applicable only for acyclic DFAs with linear average time complexity. This algorithm comprises of two steps namely: pseudo-minimization after which the actual minimization follows. This algorithm mainly focuses on finding a path taking help from the previous word suffix and tries to find the longest matching substring if the end of the string is not reached else it will clear the path if the end of the string is reached.

V. COMPARISON

The algorithms available to us can be used based upon their advantages for different types of automata. The performance of these algorithms towards DFA minimization in terms of average time complexity and the level of complexity for implementing that algorithm can be tabularised as below:

Algorithm	Avg. Time Complexity	Implementation Complexity
Hopcroft's Algorithm	O(n log n)	complex
Moore's Algorithm	$O(n^2)$	easier
Brzozowski's Algorithm	Exponential	complex
Revuz's Algorithm	Linear	complex

Table 2: comparison of various established algorithms for DFA minimization

VI. CONCLUSION

There are various algorithms discussed in this paper which gets us to the conclusion that Hopcroft's Algorithm is the most efficient Algorithm in terms of lowest time complexity and is applicable for most of the Deterministic Finite Automata, its only disadvantage is that it is complex to implement whereas Moore's algorithm is simpler to implement than Hopcroft's Algorithm but with exponential time complexity. Revuz's Algorithm and Brzozowski's Algorithm are two other algorithms which have better time complexity for some acyclic automata but they are not efficient for all different kinds of automata. So, the most efficient algorithm for the DFA minimization in terms of lowest time complexity is the Hopcroft's Algorithm.

REFERENCES

[1]. Introduction to the Theory of Computation-Textbook by Michael Sipserin the year 2017
[2]. G Efficient Deterministic Finite Automata Minimization Based on Backward Depth Information by Desheng Liu ,Zhiping Huang, Yimeng Zhang, Xiaojun Guo, Shaojing Su in the year 2016.
[3]. Weak minimization of DFA—an algorithm and applications by B.Ravikumara and G.Eismanb in the year 2004.
[4]. A Parallel DFA Minimization Algorithm by Ambuj Tewari, Utkarsh Srivastava, and P. GuptaDepartment of Computer Science & Engineering Indian Institute of Technology Kanpur 208016, INDIA, 2002.
[5]. Theory of Computation | Minimization of DFA by Geeks for Geeks at url https://www.geeksforgeeks.org/theorycomputationminimization- dfa.
[6]. DFA minimization: from Brzozowski to Hopcroft Pedro Garc´ıa, Dami´an L´opez and Manuel V´azquez de Parga Y in the year 2013.
[7]. Introduction to Theory of Computation Anil Maheshwari Michiel Smid School of Computer Science Carleton University Ottawa Canada in the year 2017.
[8]. Introduction to Automata Theory, Languages, and Computation 3rdEdition (English, Paperback, John E. Hopcroft, Rajeev Motwani, Jeffrey D Ullman)
[9]. Theory Of Computation ,Introduction to TOC, DFA, minimization of DFA- tutorials by Ravindra Babu Ravula at url: https://www.youtube.com/watch?v=eqCkkC9A0Q4.
[10]. Incremental algorithm for sorted data, described in: Jan Daciuk, Stoyan Mihov, Bruce Watson, and Richard Watson, Incremental Construction of Minimal Acyclic Finite State Automata, Computational Linguistics, 26(1), March 2000.
[11]. C. Incremental Construction and Maintenance of Minimal Finite State Automata Rafael Carrasco∗ Mikel L. Forcada in the year 2000.