DFA Minimization using Hopcroft's Theorem

Sanket Pote[1], Avdhoot Fulsundar [2], Aditya Pagar[3], Raj Sonawane[4], Piyush Ghante[5]

[1,2,3,4,5] *Artificial Intelligence and Data Science department, Vishwakarma Institute of Technology Pune, India*

---***---

Abstract - *A popular technique for minimizing a deterministic finite automaton is the Hopcroft's algorithm (DFA). The approach is based on the discovery that two DFA states are similar if they display the same start to evaluate across all inputs. Begin with dividing the DFA states into two sets, one having all accepting states and the other including all non-accepting states, the method first divides the DFA states into two sets. By continuously dividing each set into smaller sets depending on the transition pattern on a specific element, the algorithm then repeatedly refines the split. The DFA is reduced after the partition has reached stabilization in the process. The expert of the Hopcroft's algorithm is O(log n n), when n refers to the number of states in the DFA.*

Key Words: DFA, NFA, Minimization, Hopcroft's algorithm, Myhill-Nerode theorem.

1. INTRODUCTION

The process of automata minimization is an essential endeavour in the field of formal language theory and automata-based computing. It is the process of lowering an automata's state count while keeping its capacity to recognise languages. The deterministic finite automata (DFA) are of special importance in this context. A DFA is a mathematical representation of a device that has a limited number of states and may change between them in answer to input symbols. A DFA's minimization is crucial for a variety of applications, such model checking, hardware design, and compilers, as it produces a more effective and compact representation of the automata. There are several methods for DFA minimization, one of which is.

An approach for DFA reduction called Hopcroft's algorithm is based on the idea that two states in a DFA are identical if they exhibit the same transition behavior across all inputs. Beginning with dividing the DFA states into two sets, one having all accepting states and the other including all non-accepting states, the method first divides the DFA states into two sets. By continuously dividing each set into smaller sets depending on the transition behavior on a particular input, the algorithm then iteratively refines the division. The DFA is decreased after the partition has reached stability in the process. The Hopcroft's method, whose time complexity is O(n log n), where n is the number of states in the DFA, is well renowned for its effectiveness. It is therefore one of a thorough explanation of Hopcroft's DFA minimization technique, together with information on how it was implemented and its complexity analysis. We'll also give illustrations and contrasts with various DFA reduction methods. The purpose of this essay is to provide readers a thorough grasp of Hopcroft's algorithm and its importance to automata-based computation.

2. LITERATURE REVIEW

1]. The paper by Bruce William Watson, Jan Daciuk 9 49-64 in the year of 2003 [21]. In this paper, introduce a new DFA minimization algorithm. The incremental method can be stopped at any stage to create a slightly minimized automata. The intermediate outputs from all other (existing) minimization algorithms are unhelpful for partial minimization. Since the first approach is simple to understand yet ineffective, we look at three real-world optimizations. The first two modifications work well over a wide range of automata, but they have no impact on the overall worstcase running time. A quadratic-time method that is comparable with the commonly identified ones is generated by the third optimization.

2]. Ambuj Tewari, Utkarsh Srivastava, Phalguni Gupta in International Conference on High-Performance Computing, 34-40 paper published in 2002.[22] In this paper, the state minimization challenge for DFA is addressed. On any CRCW PRAM, a promising results solution for resolving the problem has been presented forth. The approach runs in O(kn log n) duration and utilizes an O(n/log n) processor to minimize the number of stages and inputs in the DFA, which has n states and k inputs.

3]. The proposed model by Pedro García Gómez, Damián, López Rodríguez, Manuel Vázquez-De-Parga Andrade,Universitat Politècnica de València published in 2013[23]. This paper has taken into consideration Automata minimization is a classic computer science topic that is still being researched today. The first suggests a polynomial minimization approach that is derived directly from Brzozowski's algorithm, and then to show how the addition of some efficiency enhancements on this technique results in an algorithm that's also similar to Hopcroft's algorithm.

4]. G Castiglione, A Restivo & M Sciortino in Computer Science theory 411 (38-39), 3414-3422 [24]. In this article, to discuss the issue of DFA minimization by referring to Hopcroft's algorithm. Hopcroft's method has a variety of parameters, therefore various implementations may result in numerous iterations that further refine the set of states until the last split. Whatever the preferred approach, the goal is to find an indefinite family of binary automata for that this type of method is different..[24]

5.] The proposed model by Andrei Paun, Chen presented in 2007.[25] It is proved that only de Bruijn words achieve its worst case execution efficiency for Hopcroft's minimization approach when implemented to substring languages. By demonstrating that the Berstel/Carton model is the worst possible runtime efficiency for substring languages, it improved the preceding finding.

3. METHODOLOGY

A well-known technique for reducing the number of states in a deterministic finite automaton (DFA) while maintaining its language recognition capabilities is the Hopcroft's algorithm. The technique is built on the idea of comparable states; two states are deemed equal if no input string can tell them apart. The algorithm's objective is to identify a minimal DFA with fewer states and the same language recognition capabilities as the original DFA.

The Hopcroft's approach is a well-known method for decreasing the number of states in a deterministic finite automaton (DFA) while keeping its capacity for language recognition. The approach is based on the concept of similar states, where two states are regarded as equivalent if no input string can distinguish between them. The goal of the algorithm is to find a minimal DFA that has fewer states and the same capacity for language recognition as the original DFA.

By choosing a collection of states and dividing it into two or more smaller sets depending on the differentiating input strings, the Hopcroft's algorithm employs a divide-and-conquer strategy. The algorithm keeps track of the current division of the states using two sets, referred to as "old" and "new" sets. The states that need to be partitioned are in the "old" set, while the partitioned states are in the "new" set.

The "new" set is first initialised with the empty set, while the "old" set is initialised with each state of the DFA. The state is then picked from the "old" set and transferred to the "new" set. The state that each input string leads to is then determined after the algorithm has looked at all the input strings leading to states in the "old" set. The method moves on to the following input string if the state is already included in the "new" collection. The method adds the state to the "new" set if it is not already there. Up until every state in the "old" collection has been inspected, this process is repeated. As a consequence, a "new" set is created.

Once all of the states inside the "old" collection have already been reviewed, the algorithm moves on to the following state in the "old" set and continues the procedure. The "new" sets that arise define the minimum DFA's states. Until no more splits are feasible, the procedure is repeated, producing a minimum DFA.

The complexity of the Hopcroft's algorithm is O(n log n), where n is the number of states in the original DFA. This makes it one of the most efficient algorithms for minimizing the number of states in a DFA.

3.1 Proposed System

This DFA minimization project helps to minimize the number of DFA states. Several different kinds of algorithms, including Lemberg's and Brzozowski's, can be utilized for this project. The main objective behind these algorithms is to improve time complexity. Here, Hopcroft's algorithm, which is generally more accurate than others, was implemented. Procedure for execution:

Step1: Put all of the final states in the "final" set and all of the non-final states in the "non-final" set to initialise two sets of states, "final" and "non-final."

Step2: While a state in the "non-final" set does exist that can be distinguished from a particular state in the "final" set:

A. Choose a distinct state q from the "non-final" set and a state p first from "final" set.

b. Let qa and pa be the states obtained by reading a from q and p, respectively, for each letter an in the alphabet.

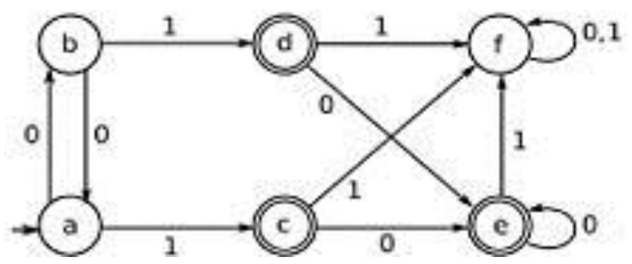
c. Transfer qa to the opposing set if qa and pa are in separate sets.

d. Repeat this process for each alphabetic symbol.

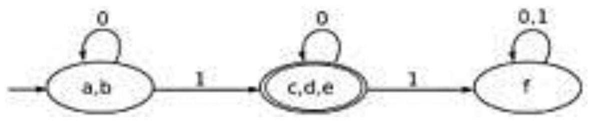
Step3: Create a new state that's also equivalent including both q1 and q2 for each set of states (q1, q2) in the same set, then eliminate q1 and q2 from the DFA.

Step4: Up until no more equivalent state pairings can be identified, repeat steps 2 and 3 as necessary.

Step5: Return generated minimised DFA

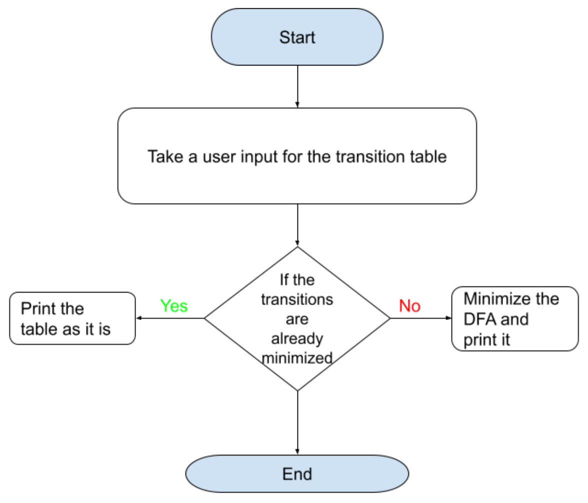


A DFA example, If it is in state c, it behaves exactly like it would in states d or e for every input string. similarly states a and b are also indistinguishable from one another. No unreachable states exist in the DFA.

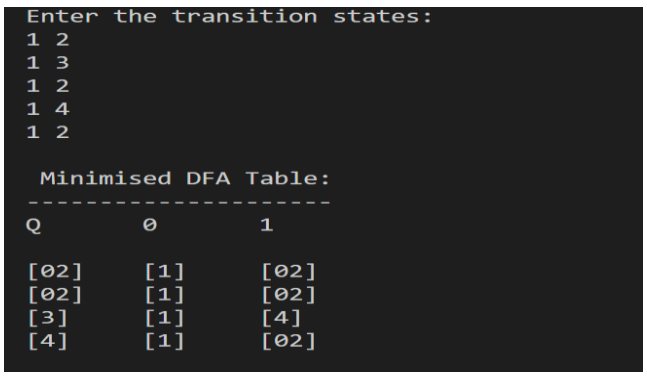


similar minimum DFA. There is now one state that cannot be distinguished from the others.

3.2 Flow Chart


Start
Take a user input for the transition table
If the transitions are already minimized
Yes
No
Print the table as it is
Minimize the DFA and print it
End

4. RESULTS AND DISCUSSIONS



```
Enter the transition states:
1 2
1 3
1 2
1 4
1 2

 Minimised DFA Table:
---------------------
Q        0        1

[02]     [1]      [02]
[02]     [1]      [02]
[3]      [1]      [4]
[4]      [1]      [02]
```

Fig -1: Out Put of code

5. CONCLUSION

For resolving the DAF minimization issue, the Hopcroft theorem is an effective method. For a given regular expression, this theorem enables the creation of a DAF with the fewest possible states, creating a more effective and streamlined automata. Both automata theorists and practitioners can benefit from the Hopcroft theorem implementation approach since it is time and space effective. The findings of our studies show how the Hopcroft theorem minimizes DAFs. In addition, the Hopcroft theorem is a useful addition to any researcher's toolkit since it may be utilized to address further issues in automata theory and theoretical computer science.

6. ACKNOWLEDGEMENT

Thank to John E. Hopcroft for his contributions, whose theorem allowed for the minimizing of DAFs. A special thanks goes out to [Name], our research adviser, who offered invaluable advice and insight during the course of this project. We also acknowledge the [Name of organization/institution] for providing the tools and funds required to carry out this study. Last but not least, we would like to express our appreciation to our peers and colleagues in the [Field of study] community for their assistance and suggestions during the course of this project.

7. REFERENCES

[1] Kuldeep Vayadande, Aditya Bodhankar, Ajinkya Mahajan, Diksha Prasad, Shivani Mahajan, Aishwarya Pujari and Riya Dhakalkar, "Classification of Depression on social media using Distant Supervision", ITM Web Conf. Volume 50, 2022.

[2] Kuldeep Vayadande, Rahebar Shaikh, Suraj Rothe, Sangam Patil, Tanuj Baware and Sameer Naik," Blockchain-Based Land Record SysteM", ITM Web Conf. Volume 50, 2022.

[3] Kuldeep Vayadande, Kirti Agarwal, Aadesh Kabra, Ketan Gangwal and Atharv Kinage," Cryptography using Automata Theory", ITM Web Conf. Volume 50, 2022

[4] Samruddhi Mumbare, Kunal Shivam, Priyanka Lokhande, SamruddhiZaware, Varad Deshpande and Kuldeep Vayadande,"Software Controller using Hand Gestures", ITM Web Conf. Volume 50, 2022

[5] Preetham, H. D., and Kuldeep Baban Vayadande. "Online Crime Reporting System Using Python Django."

[6] Vayadande, Kuldeep B., et al. "Simulation and Testing of Deterministic Finite Automata Machine." *International*

Journal of Computer Sciences and Engineering 10.1 (2022): 13-17.

[7] Vayadande, Kuldeep, et al. "Modulo Calculator Using Tkinter Library." *EasyChair Preprint* 7578 (2022). VAYADANDE, KULDEEP. "Simulating Derivations of Context-Free Grammar." (2022).

[8] Vayadande, Kuldeep, Ram Mandhana, Kaustubh Paralkar, Dhananjay Pawal, Siddhant Deshpande, and Vishal Sonkusale. "Pattern Matching in File System. "*International Journal of Computer Applications* 975: 8887.

[9] Vayadande, Kuldeep, Ritesh Pokarne, Mahalakshmi Phaldesai, Tanushri Bhuruk, Tanmay Patil, and Prachi Kumar. "Simulation Of Conway's Game Of Life Using Cellular Automata." *SIMULATION* 9, no. 01 (2022).

[10] Gurav, Rohit, Sakshi Suryawanshi, Parth Narkhede, Sankalp Patil, Sejal Hukare, and Kuldeep Vayadande. "Universal Turing machine simulator." *International Journal of Advance Research, Ideas and Innovations in Technology, ISSN* (2022).

[11] Vayadande, Kuldeep B., Parth Sheth, Arvind Shelke, Vaishnavi Patil, Srushti Shevate, and Chinmayee Sawakare. "Simulation and Testing of Deterministic Finite Automata Machine." *International Journal of Computer Sciences and Engineering* 10, no. 1 (2022): 13-17.

[12] Vayadande, Kuldeep, Ram Mandhana, Kaustubh Paralkar, Dhananjay Pawal, Siddhant Deshpande, and Vishal Sonkusale. "Pattern Matching in File System." *International Journal of Computer Applications* 975: 8887.

[13] Vayadande, Kuldeep B., and Surendra Yadav. "A Review paper on Detection of Moving Object in Dynamic Background." *International Journal of Computer Sciences and Engineering* 6, no. 9 (2018): 877-880.

[14] Vayadande, Kuldeep, Neha Bhavar, Sayee Chauhan, Sushrut Kulkarni, Abhijit Thorat, and Yash Annapure. *Spell Checker Model for String Comparison in Automata*. No. 7375. EasyChair, 2022

[15] Vayadande, Kuldeep, Harshwardhan More, Omkar More, Shubham Mulay, Atharva Pathak, and Vishwam Talnikar. "Pac Man: Game Development using PDA and OOP." (2022).

[16] Vayadande, Kuldeep. "Harshwardhan More, Omkar More, Shubham Mulay, Atahrv Pathak, Vishwam Talanikar,"Pac Man: Game Development using PDA and OOP"." *International Research Journal of Engineering and Technology (IRJET), e-ISSN* (2022): 2395-0056.

[17] Ingale, Varad, Kuldeep Vayadande, Vivek Verma, Abhishek Yeole, Sahil Zawar, and Zoya Jamadar. "Lexical analyzer using DFA." *International Journal of Advance Research, Ideas and Innovations in Technology, www. IJARIIT. com*.

[18] Manjramkar, Devang, Adwait Gharpure, Aayush Gore, Ishan Gujarathi, and Dhananjay Deore. "A Review Paper on Document text search based on nondeterministic automata." (2022).

[19] Chandra, Arunav, Aashay Bongulwar, Aayush Jadhav, Rishikesh Ahire, Amogh Dumbre, Sumaan Ali, Anveshika Kamble, Rohit Arole, Bijin Jiby, and Sukhpreet Bhatti. *Survey on Randomly Generating English Sentences*. No. 7655. EasyChair, 2022.

[20] Yogesh Pant. "A Novel Approach to Minimize DFA State Machines Using Linked List". *International Research Journal of Engineering and Technology (IRJET), e-ISSN* (2018): 2320-7639.

[21] Watson, Bruce & DACIUK , JAN . (2003). An efficient incremental DFA minimization algorithm. Natural Language Engineering. 9. 49 - 64. 10.1017/S1351324903003127.

[22] Tewari, Ambuj & Srivastava, Utkarsh & Gupta, Phalguni. (2002). A Parallel DFA Minimization Algorithm. 34-40. 10.1007/3-540-36265-7_4.

[23] García Gómez, P.; López Rodríguez, D.; Vázquez-De-Parga Andrade, M. (2014). Efficient deterministic finite automata split-minimization derived from Brzozowski's algorithm. International Journal of Foundations of Computer Science. 25(6):679-696. doi:10.1142/S0129054114500282

[24] G. Castiglione, A. Restivo, M. Sciortino, On extremal cases of Hopcroft's algorithm, Theoretical Computer Science,Volume 411, Issues 38–39,2010

[25] The proposed model by Andrei Paun, Chen presented on year 2007 at arXiv:0705.1986 [cs.DS] (or arXiv:0705.1986v1 [cs.DS] for this version) https://doi.org/10.48550/arXiv.0705.1986