Generation of strings in language for given Regular Expression and printing its probable DFA state

Vaidehi Ligde[1], Rushikesh Malpani[2], Soham Borkar[3], Phinehas Mane[4]

[1,2,3,4] *Department of artificial intelligence and data science vishwakarma institute of technology pine, India*

ABSTRACT

Finite Automata (FA) is the simplest machine to recognize patterns. It is an abstract model of a digital computer. Regular Expression is a sequence of characters, that specifies a searching pattern in text. For “find” or “find and replace” string searching algorithms are used for operations on string or for input validation. Regular expressions are a powerful tool for specifying and manipulating strings and patterns in text. It can be used to generate finite automata, which are mathematical models that can recognize and process strings within a given language. There are various algorithms and techniques for generating finite automata from regular expressions, such as the Thompson construction algorithm and the Glouchkov construction algorithm. Regular Expression can be useful for tasks such as pattern matching and string manipulation, as it allows for efficient and precise specification of the desired strings.

KEYWORDS

Regular Expressions, Finite automata, Language generation String manipulation, Pattern matching Tautology, Thompson construction, Glushkov construction, Regular language.

1. INTRODUCTION

Regular Expressions are a fundamental concept in computer science and are used to specify and manipulate strings and patterns in text. They are a powerful tool for describing and processing a wide range of languages, including programming languages, natural languages, and database query languages. In the context of generating finite automata and language, regular expressions play a central role. Finite automata are mathematical models that can recognize and process strings within a given language. They are widely used in computer science and are often used to implement lexical analysers in compilers, to recognize patterns in text data, and to validate input in programs. There are various algorithms and techniques for generating finite automata from regular expressions, including the Thompson construction algorithm and the Glushkov construction algorithm. These algorithms take a regular expression as input and output a corresponding finite automaton, which can then be used to recognize and process strings within the language described by the regular expression. In addition to generating finite automata, regular expressions can also be used to generate the language itself, which is the set of all strings that can be recognized by the automaton. This can be useful for tasks such as string manipulation and pattern matching, as it allows for efficient and precise specification of the desired strings. Regular languages, which are languages that can be described using regular expressions, are a subclass of formal languages and are widely used in computer science and related fields. Overall, the ability to generate finite automata and languages using regular expressions is a valuable tool in computer science and related fields, enabling efficient specification and manipulation of strings and patterns.

2. LITERATURE REVIEW

[1] Deterministic finite automata (DFAs) are mathematical models that can recognize and process strings within a given language. They are widely used in computer science and are often used to implement lexical analysers in compilers, to recognize patterns in text data, and to validate input in programs. [2] One way to generate DFAs is by using regular expressions, which are a widely used tool in computer science for specifying and manipulating strings and patterns in text. [3] There are various algorithms and techniques for generating DFAs from regular expressions, including the Thompson construction algorithm and the Glushkov construction algorithm. [4] The Thompson construction algorithm, introduced by Ken Thompson in 1968, is a well-known method for generating DFAs from regular expressions. It is a simple and efficient algorithm that can be implemented using a stack-based data structure. [5] The algorithm takes a regular expression as input and constructs a corresponding nondeterministic finite automaton (NFA). [6] The NFA can then be converted to a DFA using the powerset construction, which allows the automaton to recognize and process strings within the language described by the regular expression. [7] The Glushkov construction algorithm, introduced by Anatoly Glushkov in 1962, is another method for generating DFAs from regular expressions. It is a more complex algorithm that involves constructing a direct acyclic graph (DAG) representation of the regular expression and then converting the DAG to a DFA using a bottom-up construction. [8] The Glushkov construction has the advantage of producing smaller automata than the Thompson construction, but it can be more

difficult to implement and may not be as efficient in practice.[9] In addition to generating DFAs, regular expressions can also be used to generate the language itself, which is the set of all strings that can be recognized by the automaton.[10] This can be useful for tasks such as string manipulation and pattern matching, as it allows for efficient and precise specification of the desired strings. Regular languages, which are languages that can be described using regular expressions, are a subclass of formal languages and are widely used in computer science and related fields. [11] There has been significant research in the area of generating DFAs and language using regular expressions. [12] Researchers have developed various algorithms and techniques for constructing DFAs from regular expressions, and have explored the properties and characteristics of regular languages. For example, researchers have studied the relationship between regular languages and other classes of formal languages, such as context-free languages and context-sensitive languages. [13] They have also investigated the computational complexity of various operations on regular languages, such as union, intersection, and complementation.[14] Overall the ability to generate DFAs and language using regular expressions is a valuable tool in computer science and related fields, enabling efficient specification and manipulation of strings and patterns.[15] The research in this area has contributed to a better understanding of the properties and capabilities of regular languages and has led to the development of effective algorithms and techniques for generating DFAs and language.

3. METHODOLOGY

1. A sequence a characters that specifies a search pattern in texts is regular expression. Most of the time string searching algorithms for finding and replacing operations this type of patterns are used.

2. Although regular expression can also be used for generating finite automata

3. Finite automata are mathematical models that can recognize and process strings within a given language there are various algorithm and techniques

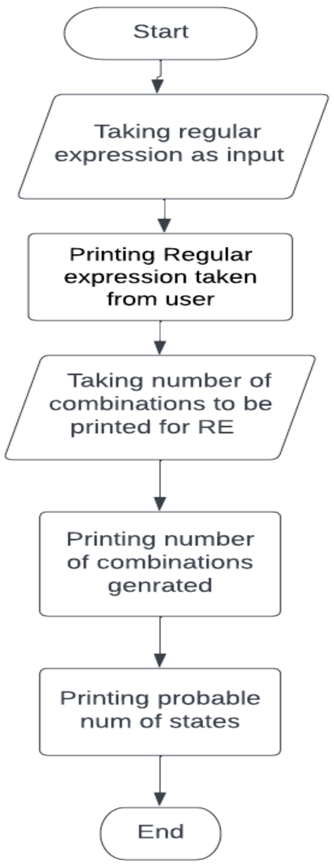
4. For generating finite automata from regular expression some algorithms like Thompson construction algorithm and the Glouchkov construction algorithms are used

5. We know regular expression can be a very useful for pattern matching and string manipulation, as it allows for efficients and precise specification of the desired strings.

3.1 PROPOSED SYSTEM

The proposed system will be a program that allows users to input a regular expression which may consisting of '*' symbol and generates its corresponding strings. The input to the system will be a regular expression in the form of a string. The system will then parse the regular expression through the program and construct a corresponding expression string which will be the part of language for given regular expression. After printing strings of language for regular expression, the program will print a probable number of states given expression. Overall, the proposed system will provide a convenient and user-friendly way for users to generate a number of strings from regular expressions. It will be a valuable tool for other people in computer science and related fields who need to generate a number of strings of language for a given regular expression and print probable states DFA of that regular expression. 'a*ba*b' may be the input we get from the user which will be a valid regular expression. After asking how many strings users want in a language, we process that regular expression by considering it as a string. Program will extract each element of the string and produce different combinations of strings and store them. At the end we print the probable number of states DFA for regular expression.

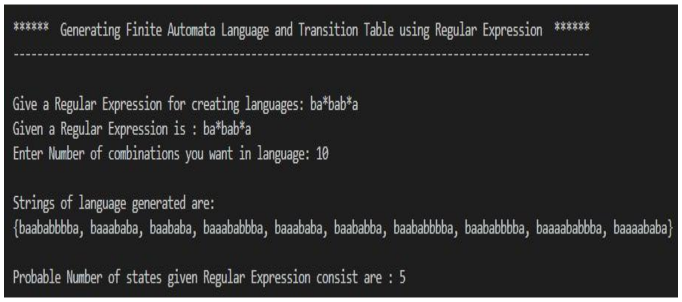
3.2 FLOWCHART


Start
Taking regular expression as input
Printing Regular expression taken from user
Taking number of combinations to be printed for RE
Printing number of combinations genrated
Printing probable num of states
End

4. RESULTS AND DISCUSSIONS

Making All above is the flow of a program used to generate strings of language for given regular expression and printing probable number of states DFA consist.

Users first needed to provide regular expressions as input to the program and then input a number of combinations of strings in language. If entered regular expression is valid regular expression then program will print string combination and provable states of DFA as output of program.

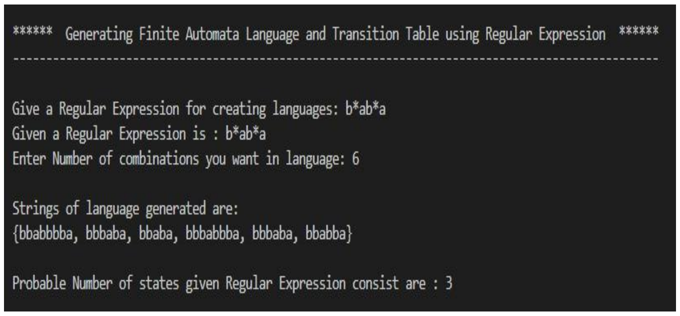


```
******  Generating Finite Automata Language and Transition Table using Regular Expression  ******
--------------------------------------------------------------------------------------------


Give a Regular Expression for creating languages: ba*bab*a
Given a Regular Expression is : ba*bab*a
Enter Number of combinations you want in language: 10

Strings of language generated are:
{baababbbba, baaababa, baababa, baaababbba, baaababa, baababba, baababbbba, baababbbba, baaaababbba, baaaababa}

Probable Number of states given Regular Expression consist are : 5
```

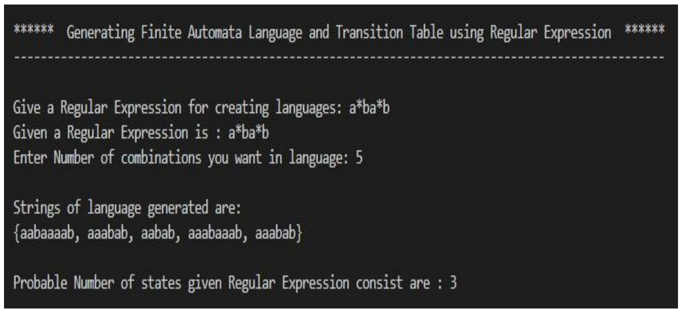


```
******  Generating Finite Automata Language and Transition Table using Regular Expression  ******
--------------------------------------------------------------------------------------------


Give a Regular Expression for creating languages: b*ab*a
Given a Regular Expression is : b*ab*a
Enter Number of combinations you want in language: 6

Strings of language generated are:
{bbabbbba, bbbaba, bbaba, bbbabbba, bbbaba, bbabba}

Probable Number of states given Regular Expression consist are : 3
```

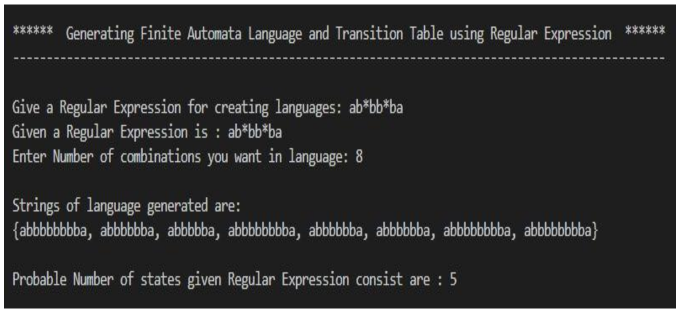


```
******  Generating Finite Automata Language and Transition Table using Regular Expression  ******
--------------------------------------------------------------------------------------------


Give a Regular Expression for creating languages: a*ba*b
Given a Regular Expression is : a*ba*b
Enter Number of combinations you want in language: 5

Strings of language generated are:
{aabaaaab, aaabab, aabab, aaabaaab, aaabab}

Probable Number of states given Regular Expression consist are : 3
```



```
******  Generating Finite Automata Language and Transition Table using Regular Expression  ******
--------------------------------------------------------------------------------------------


Give a Regular Expression for creating languages: ab*bb*ba
Given a Regular Expression is : ab*bb*ba
Enter Number of combinations you want in language: 8

Strings of language generated are:
{abbbbbbbba, abbbbbba, abbbbba, abbbbbbbba, abbbbbba, abbbbbba, abbbbbbbba, abbbbbbbba}

Probable Number of states given Regular Expression consist are : 5
```

5. LIMITATIONS

- Project isn't capable of producing DFA and NFA.
- Project only allows you to take regular expressions as input.
- States produced by a program for given regular expression are probable.

6. CONCLUSION

By developing the program we learnt numerous concepts of DFA, string generation, language, states in DFA, creation of DFA and its related field. So, we developed a program for generation of string which will be the part of language for given regular expression and print probable number of states DFA of given expression may consist of.

7. FUTURE SCOPE

So, this project gives output after compiling the whole result. In the future, we can represent our output in graphical format. Also, we can add a feature for creation of NFA as well as NFA using null transition.

8. REFERENCES

[1] Visual Designing and Debugging of Deterministic FiniteState Machines in FSM [2008.09254] Visual Designing and Debugging of Deterministic Finite-State Machines in FSM (arxiv.org)

[2] An Efficient Protocol for Oblivious DFA Evaluation and Applications An Efficient Protocol for Oblivious DFA Evaluation and Applications | SpringerLink

[3] A Collection of Tools for Making Automata Theory and Formal Languages Come Alive A collection of tools for making automata theory and formal languages come alive | ACM SIGCSE Bulletin

[4] Dynamic testing via automata learning Dynamic Testing Via Automata Learning | SpringerLink

[5] Application Research of Finite Automaton in Distance EducationApplication research of finite automaton in distance education | IEEE Conference Publication | IEEE Xplore

[6] A General Approach to DFA Construction (PDF) A General Approach to DFA Construction (researchgate.net)

[7] Turing machine and automata Simulators Turing Machine and Automata Simulators - ScienceDirect

[8] A Review Paper On Finite Automata Application in String Identification (PDF) EasyChair Preprint A Review Paper on Finite Automata Application in String Identification A Review Paper On Finite Automata Application in String Identification (researchgate.net)

[9] On Parallel Implementations of Deterministic Finite Automata On Parallel Implementations of Deterministic Finite Automata | SpringerLink

[10] 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.

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

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

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

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

[15] 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.

[16] Vayadande, Kuldeep, et al. "Modulo Calculator Using Tkinter Library." EasyChair Preprint 7578 (2022).

[17] VAYADANDE, KULDEEP. "Simulating Derivations of Context-Free Grammar." (2022).

[18] 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.

[19] 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).

[20] 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).

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

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

[27] 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.

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

[29] 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.