International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 10 Issue: 01 | Jan 2023 www.irjet.net p-ISSN: 2395-0072

Regular Expression to Non-Deterministic Finite Automata Converter

Pratik Dhame[1], Siddhesh Shinde[2], Rushikesh Sanjekar[3], Soham Dixit[4]

[1]*Student, Dept. of Information Technology, Vishwakarma Institute of Technology, Maharashtra, India*
[2]*Student, Dept. of Information Technology, Vishwakarma Institute of Technology, Maharashtra, India*
[3]*Student, Dept. of Information Technology, Vishwakarma Institute of Technology, Maharashtra, India*
[4]*Student, Dept. of Information Technology, Vishwakarma Institute of Technology, Maharashtra, India*

Abstract - ***Regular Expression (RE) is an important notation for specifying patterns. It is a shortened way of writing the regular language from this we can know how a regular language is built. Every Regular expression contains a definite language. This way of describing is known as algebraic description. NFA is also important as it reduces the complexity of mathematical work required. NFA is easier to construct than DFA and it rejects the string whenever all branches refusing string. We require conversion of RE to NFA because to recognize a token. Another way of calling finite automata is token recognition. That's why we need to convert RE to NFA.***

Key Words: **Regular Expression, Non-Deterministic Finite Automata, Automata Theory, Deterministic Finite Automata, Transition Table, Automaton**

1. INTRODUCTION

Non-deterministic finite automata (NFA) is a type of finite automaton that allows for multiple changes from one input sign to another from a single state this means that, unlike a deterministic finite automaton (DFA), an NFA can have multiple possible next positions for a given current position and input symbol. Converting a regular expression to NFA is often done as a preliminary phase in the implementation process of regular expression check. NFA is a model of computation that can be used to match a string against a RE.

1) Well organized implementation: An NFA can be more structured to implement than RE in some cases, especially when the RE is more complex.

2) Interoperability: Many software libraries and tools that perform RE matching expect to receive an NFA as input, rather than an RE. Implementing RE in NFA can allow these tools to be used with RE.

3) Simplicity: An NFA is finite state machine, which means it can be represented an manipulated as a data structure. This can make it easier to work with than an RE, which is more brief concept.

Regular expression is nothing but the simpler form of representing a string. An expression over the alphabet ∑ using operator (+. *) is called regular expression. There are many methods for nondeterministic implementation of RE but the best among them are 2 methods one proposed by Mc Naughton & Yamada and second one is Thomson. There are few concepts that we need to know before creating NFA from the existing RE.

First is what means finite automata:

Finite automaton (FA) is a mathematical model used to recognize patterns within input. It is made up of a finite collection of states, a group of input symbols, transitions between those states, a starting state, and a group of accept states. Here is how it works

- The automaton reads an input symbol at a time.
- For each symbol, according to the transition function it changes states.
- If the automaton ever enters an accept state, it "accepts" the input, If the automaton enters a non-accept state or runs out of input symbols before entering an accept state it "rejects" the input.

FA can be divided in two distinct types: DFA and NFA

Before going forward there are few things that we need to keep in mind that are the 5 tuples used in finite automata.

- Finite collection of states `Q`: Set of states that automaton can be in.
- An input alphabet `∑`: This is the group of all input symbols that automaton can read.
- A transition function `δ`: This is an attribute that determines the next state of the automaton.
- A start state `q0`: This is the state that the automaton starts in when it begins processing an input.
- Group of accept state `F`: Once the pattern is successfully recognized in which automaton can be then, if the automaton ever enters one of these states, it "accepts" the input.
- δ: Transition function

DFA: DFA is a type of FA that has a unique transition for each state/input symbol combination. DFA could be represented graphically as a state diagram, with the states of the automaton represented as a circles and the transitions between states represented as labeled arrows.

Non-Deterministic finite automata: NFA is quite opposite and there are few changes that we can transit to the next state using the epsilon value and there is no need to just transfer the unique value you can transmit any number of states of a given value. This means that, unlike a deterministic finite automaton (DFA), an NFA can have N number of possible future states for a current state as well as input symbol. In contrast, a DFA has a unique transition for every state and input symbol combination.

Regular Expression: It is mainly used to match the pattern in the text and is sequence of characters defining the search pattern. Regular expressions can be used to match a variety of patterns, including simple strings, digits, and complex combinations of characters and symbols. Overall, regular expressions are a powerful and widely used tool for working with text and patterns in computer science and programming.

2. LITERATURE REVIEW

1. A Method for Converting RE into DFA published by International Journal of Applied Science and Engineering [1] tells about the method proposed by them where in they decompose the symbols one by one, and this experiment is done in JFLAP tool.

2. McNaughton and Yamada's Algorithm:

It is a general-purpose methodology for extracting state graphs from regular expressions.[2] this algorithm produces fewer states than 2p+1 which explains that while creating the graph first comes start state. Although the resulting NFA may have a large number of states. The NFA can then be determinized.

3. Thompson's Algorithm

This algorithm works by dividing the given regular expressions into smaller chunks and then transforming them into NFA later these small NFA are combined to form the complete NFA. In addition, it minimizes the error by taking single character at a time and converting it into an NFA, the characters are checked from left to right.

4. On the Minimization of State NFA published by IEEE Transactions on computer [2] tells us how to minimize the given NFA. And reversing it to check whether the given NFA is correct or not.

5. An improved Algorithm for Evaluating RE published by ACM Digital Library [3] this algorithm gives the description about how the DFA can be compressed so that it becomes easy to extract the Regular expression from it.

6. From Regular Expression to Deterministic Finite Automata published by Science Direct [4] states the best way to transform the given RE to FA and study of 3 algorithms by above mentioned researchers.

3. METHODOLOGY

Thompson's algorithm is a method for constructing NFA from a RE. The NFA produced by the algorithm can be used to recognize patterns in strings.

This algorithm works by constantly dividing the expressions into sub expressions, later from those subexpressions the NFA is formed.

Here's the algorithm:

1. Each symbol in the RE is treated as a separate NFA with a single state.
2. The NFA for the empty string ` ε` is a single state with an epsilon transition to itself and an accept state.
3. The NFA for the concatenation of two Res is constructed by combining the accept state of NFA for first to the beginning state of the NFA for second.
4. The NFA for union is constructed with a new initial state and a new accept state, as well as the addition of epsilon transitions from the new initial to the start state of NFAs.
5. NFA for Kleene star is constructed by new start and accept state by addition of epsilon transitions from the fresh start state to the NFA for 'R'.

There are 5 simple rules this algorithm follows for constructing the NFA from given regular expression.

1. An empty expression rule: Expression having e or epsilon will be converted to:

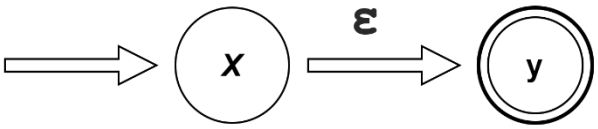


Fig. -1: Empty expression rule

2. A symbol rule: Symbols like 'a' or 'b' will transformed to:

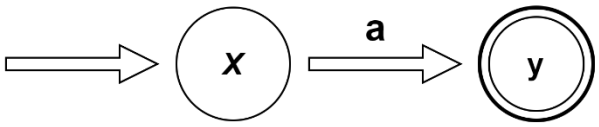


Fig -2: Symbol rule

3. Union Expression rule: Expressions like 'a+b' or 'a | b' which are known as union expressions will be converted to:

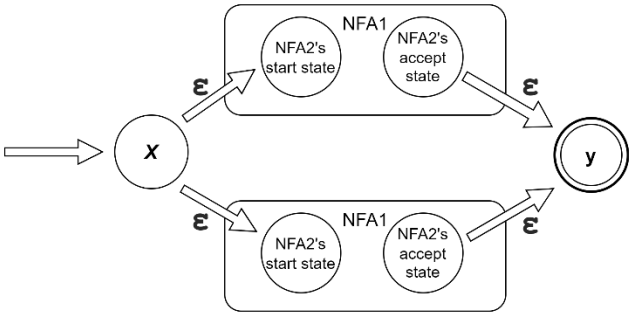


Fig -3: Union Expression rule

4. Concatenation Expression rule: An expressions like 'ab' or 'a.b' which are nothing but concatenated expressions will be converted to:

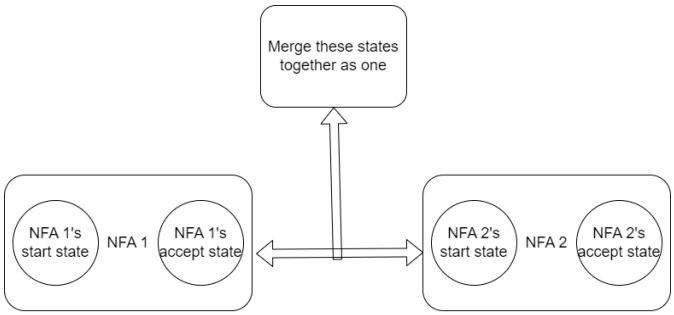


Fig -4: Concatenation Expression rule

5. A closure/kleen star Expression: An expression like 'a*' or 'b*' will be converted to:

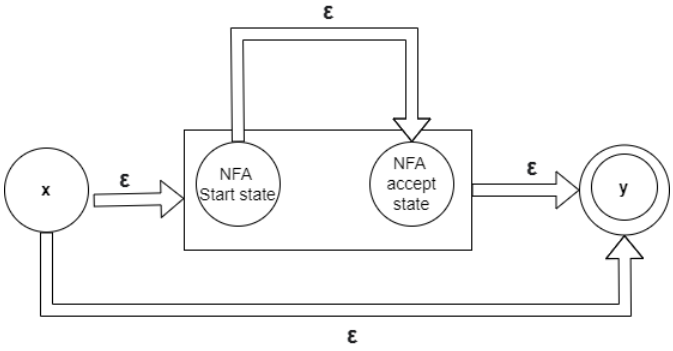


Fig -5: Closure/Kleen star Expression

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper.

3.1 PROPOSED SYSTEM

Steps required for converting regular expression to NFA have been explained above, according to that our proposed project is as follows.

1st step: To accept the regular expression string from the user.

2nd step: Defining a 2D matrix having 3 columns and 20 rows which will hold the transitions for particular sigma value.

3rd step: Traversing whole expression character by character and checking it with the conditions.

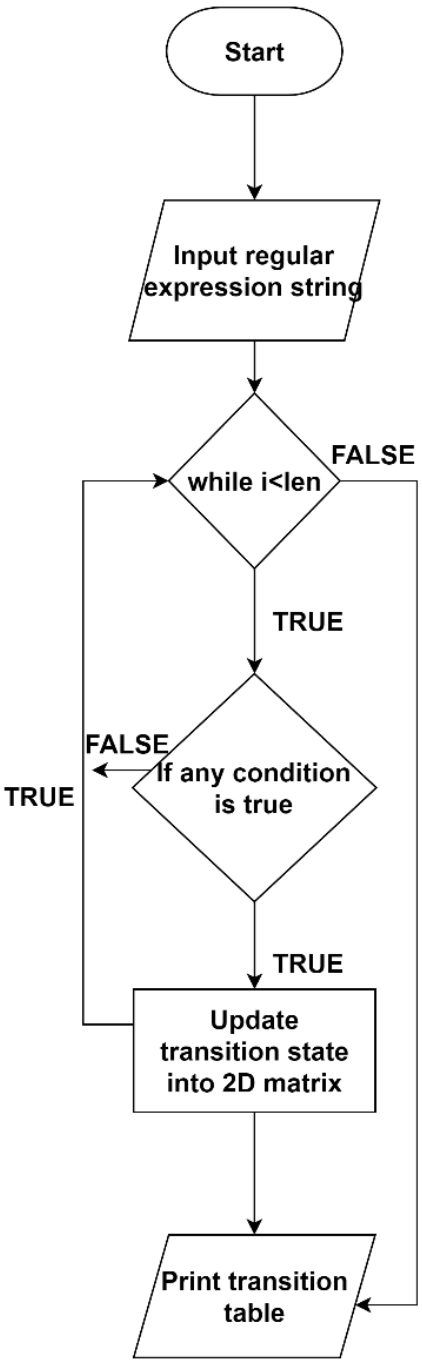
1. First condition checks whether the input character is single ‘a’.
2. Second condition checks whether the input character is single ‘b’.
3. Third condition checks whether the input string is ‘a+b’.

These are some of the conditions checked.

4th step: If the condition is matched, we will fill the 2D matrix with transition states.

5th step: Displaying the transition table to the user from the 2D matrix.

3.2 FLOWCHART



4. RESULTS AND DISCUSSIONS

From the resulting output, we get the transition table of NFA from the (a+b)* RE that is given in example below.

As seen below, the transition states have been increased due to epsilons according to the Thompson's construction algorithm.

This algorithm can be further developed to reduce the epsilons which will further reduce the transition states.

```
D:\Degree\SEM3\DSAAT\CP>main
Enter the regular expression: (a+b)*
Given regular expression: (a+b)*

        Transition Table
-------------------------------------
Current State | Input | Next State
-------------------------------------

  q[0]        |   e   |  q[4] , q[1]
  q[1]        |   a   |  q[2]
  q[2]        |   b   |  q[3]
  q[3]        |   e   |  q[4] , q[1]
-------------------------------------
```

5. LIMITATIONS

1. The code does not optimize the NFA for size or performance. The NFA that is produced may be unnecessarily large or may not be optimized for efficient matching against strings.

2. Program code does not provide input validation and error checking. If given input is invalid, it will produce undesired results.

6. CONCLUSION

Regardless of the specific algorithm used, the resulting NFA could be determinized (transformed into a DFA) using another algorithm, such as McNaughton and Yamada's algorithm for determinization. This can be useful for implementing efficient regular expression matching algorithms.

7. FUTURE SCOPE

1. Developing algorithms that can detect and handle errors in the regular expression, or that can validate the input to ensure it is a valid regular expression, would improve the reliability and usability of the program.

2. Optimization of the NFA size and performance, such as reducing the number of states and transitions or using more efficient data structures to represent the transitions.

REFERENCES

[1] Singh, Abhishek. (2019). A Method to Convert Regular Expression into Non-Deterministic Finite Automata. International Journal of Applied Science & Engineering. 7. 10.30954/2322-0465.2.2019.3.

[2] R. McNaughton and H. Yamada, "Regular Expressions and State Graphs for Automata," in IRE Transactions on Electronic Computers, vol. EC-9, no. 1, pp. 39-47, March 1960, doi: 10.1109/TEC.1960.5221603.

[3] B. Melnikov and A. Tsyganov, "The State Minimization Problem for Nondeterministic Finite Automata: The Parallel Implementation of the Truncated Branch and Bound Method," 2012 Fifth International Symposium on Parallel Architectures, Algorithms and Programming, 2012, pp. 194-201, doi: 10.1109/PAAP.2012.36.

[4] Michela Becchi and Patrick Crowley. 2007. An improved algorithm to accelerate regular expression evaluation. In Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems (ANCS '07). Association for Computing Machinery, New York, NY, USA, 145–154.

[5] Gerard Berry, Ravi Sethi, From regular expressions to deterministic automata, Theoretical Computer Science, Volume 48, 1986, Pages 117-126, ISSN 0304-3975.

[6] Domenico Ficara, Stefano Giordano, Gregorio Procissi, Fabio Vitucci, Gianni Antichi, and Andrea Di Pietro. 2008. An improved DFA for fast regular expression matching. SIGCOMM Comput. Commun. Rev. 38, 5 (October 2008), 29–40.

[7] Brüggemann, Anne. (1993). Regular Expressions into Finite Automata. Theoretical Computer Science. 120. 197-213. 10.1016/0304-3975%2893%2990287-4.

[8] 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.

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

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

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

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

[13] 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.

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

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

[16] 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.

[17] 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).

[18] 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).

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

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

[25]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.

[26]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.

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

[28] 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.