Journal of Emerging Computer Technologies
Research Article
Received: *2021-08-20* | ***Reviewing:*** *2021-08-23* & *2021-09-15* | ***Accepted:*** *2021-09-18* | ***Online:*** *2021-09-18* | ***Issue Date:*** *2021-12-31*

Development of An Efficient Tool to Convert Regular Expressions to NFA

Mustafa Batar
Department of Computer Engineering
Burdur Mehmet Akif Ersoy University
Burdur, Turkey
mbatar@cs.deu.edu.tr
0000-0002-8231-6628

Kökten Ulaş Birant
Department of Computer Engineering
Dokuz Eylül University
İzmir, Turkey
ulas@cs.deu.edu.tr
0000-0002-5107-6406

***Abstract*—In the computing theory, while the term "Language" specifies the string set, the term "Regular Expressions" means the notation that builds, creates and generates these languages. Also, the term "Regular Expressions" creates the characters that structure and compose, which refers to the given strings, in order to search patterns for sample matching. In this context, this article tries to show how to convert "Regular Expressions" that is made up of characters into Nondeterministic Finite Automata (NFA), which is a character matching and character searching tool, by giving related algorithms and methods with their explanations in detail. Moreover, in this study, a new and efficient tool has been designed and developed in order to convert regular expressions to NFA. By the contribution of this application, an original conversion tool will have been gained in the computation area for benefiting it. As a natural result of this, an original NFA modelling tool will have been placed in the literature.**

Keywords—Computing theory, regular expressions, NFA, modelling.

I. Introduction

Today, humanity is in the middle of the information age. However, this age encompasses an intellectual power rather than a physical one. For the past 25-30 years, information technology has played a very important role in the world. But what this power can do is not yet known to us. If looked at the 1950s, only some of the people kept diaries, but now; most people post everything from international political events to their pet's favorite toy on their personal pages online for all to see. But still, for some reason, people still hold on to the idea that the concept of computation: numbers or punctuation or something like that [1].

Although the concept of computation does not only belong to the numbers or punctuation marks, the meaning of this concept is still not clearly and unambiguously known. Almost every person in the world thinks that if a person is dealing with accounting, s/he accepts that s/he is doing calculations. However, many also have different opinions about whether the brain is a computer or something else. If the brain is a computer, it turns out that all the thoughts are computations and all computations are thoughts. It is also known that; the virtual world in computer games is a reflection of the real world that has emerged as a result of calculations. Thus, a dilemma arises as to whether the real universe is a computer. This dilemma is addressed the question into mind: "If the universe is a computation, then what is not computation?" [1].

About 100 years ago, most people in the world thought that computing was a mental operation involving numbers, and therefore it was believed that information counting could only be performed by humans. But today, it is thought by the vast majority that the computation can only be performed by machines. In Western culture, the thinking capacity of humans is considered as the most important factor that distinguishes humans from animals. Also in this culture, private thoughts and feelings are considered as the main factor for the formation of personal identity. For this reason, thought or thinking is believed to be necessary both for living together and for personal development. It is pointed out that changes in ideas about the meaning of the concept of computation also cause changes in thinking and ideas about humanity. This reflects the change in concepts such as culture, excitement, anxiety or virtual. Despite hundreds of known articles and researches about the concept of computation so far, what this concept exactly is still preserves and continues to be confidential and its mystery [1].

II. Related Concepts

A. Automata Theory

The related theory is a developmental process that emerged to attract and distinguish people from a particular type of computer or a particular programming language. In automata theory, the emphasis is on the mathematical example of the concept of computation. In the theoretical sense; since it is easier to control and direct the machines mathematically, this theorem on automata creates a model, brings the computational mechanisms or machines to the simplest and plainest level, and reveals the bareness of these machines in order to demonstrate and also to prove the capacities of these machines [2].

The mathematical models are generally not suitable or not useful for solving practical programming problems. Real programming languages are better suited and more convenient for solving these problems. Although these programming languages are easy to use, it is rather difficult to evaluate them in a particular format. Theoretically, the main idea, that the relevant machines are designed from top to bottom to be made the simplest and mathematically operable, should definitely not to be forgotten [3].

B. DFA – Deterministic Finite Automata

The related automata is a finite state machine that accepts or rejects a certain number of strings, and in automata theory, determines a single computational or run-automation for each string input. The concept of "deterministic" means that the computation is unique and only one. When looked at the previous studies and researches in order to understand better the concept of "Deterministic Finite Automata (DFA)", it is seen that "McCulloch" and "Pitts" were the first researchers to put forward a concept similar to finite automation in 1943 [4].

DFA is defined as an abstract concept related to mathematics, but since it is "specific", "deterministic" and "certain", it has a structure and feature that can be applied and used in both software and hardware for solving various problems. For example, by means of a DFA, a software application can be modeled that reveals whether the entered e-mail addresses are valid or not. It can also identify and reveal a regular set of languages that help to do lexical analysis and pattern matching. With the help of various algorithms, transformation from nondeterministic finite automata to deterministic finite automata can be performed, carried out, and implemented [5].

C. NFA – Nondeterministic Finite Automata

The relevant automata is a generalized version of the deterministic finite automata defined above – DFA. In nondeterministic finite automata, there may be zero, one, or more connections between each state. In this automaton, if there is more than one connection coming out of a situation, it means that the branch is divaricating, but if this branch does not have any valid connections, this branch disappears, that is, it dies [6].

If the given input character string reaches the acceptance state, the NFA accepts this character string set, but if it does not reach the acceptance state at all, it rejects that string set. In this automaton, a single accept condition is sufficient, but for a reject condition all branches have to reject the character input string. This mechanism, which is carried out, is called the modeling of the computational concept [7].

D. Regular Grammar

Related grammar defines regular languages, and also has a specific format. If a grammar has one of the following forms, that grammar is regular, but if it does not, that grammar is irregular (not regular) [8] as defined below:

$S \rightarrow \varepsilon \mid S \rightarrow w \mid S \rightarrow T \mid S \rightarrow wT$

w: is any terminating character in a regular expression that cannot be defined as a null character.

T: is any character that will not terminate a regular expression.

The most important aspect of establishing grammar is to have the knowledge and information underlying the language. It should not be assumed that a language will have only one grammar because more than one grammar structure of the same language can be created and installed. The following table – Table 1 – shows the grammar structure of various languages. In this table (Table 1), each regular language is represented with the contribution of regular expressions [8]:

TABLE I. REGULAR EXPRESSIONS AND REGULAR GRAMMAR

Regular expressions	**Regular grammar**
a*	$S \rightarrow \varepsilon \mid aS$
(a+b)*	$S \rightarrow \varepsilon \mid aS \mid bS$
a*b	$S \rightarrow b \mid aS$
(ab)*	$S \rightarrow \varepsilon \mid abS$
ba*	$S \rightarrow bA$ $A \rightarrow \varepsilon \mid aA$

E. Regular Language

It is a language with the alphabet "Σ", and as a concept, it has emerged as the constant repetition of the following features and characteristics [9]:

The empty set "Ø", which is an empty language, is also a regular language.

A single letter (character) belonging to the alphabet is also a regular language ("a" $\in \Sigma$).

If languages "A" and "B" are regular languages, then the set "AUB" (union operation), set "AoB" (concatenation operation), and set "A*" (star operation) are also regular languages.

F. Regular Operation – Union Operation

Union operation takes all the sequences of the regular languages "A" and "B", and combines these sequences into a new language [10].

AUB = {x | x Є A V x Є B}; A = {hot, cold} and B = {woman, man}; then AUB = {hot, cold, woman, man}

G. Regular Operation – Concatenation Operation

Concatenation operation connects the beginning of the string of the regular language "B" to the end of the string of the regular language "A" [10].

A o B = {xy | x Є A Λ y Є B}; A = {hot, cold} and B = {woman, man}; then AoB = {hotwoman, hotman, coldwoman, coldman}

H. Regular Operation – Star Operation

Star operation applies to a single regular language, and adds any number of strings of that language together to assemble them into a new regular language [10].

A* = {$x_1x_2 \ldots x_k$ | k is positive integer Λ all x_i Є A}; A = {hot, cold}; then A* = {Ø, hot, cold, hothot, coldcold, hothothot, hotcoldhot,...}

İ. Regular Expressions

It is the sorted form of the characters that make up the search pattern, and used to make pattern matching with the given character strings. In regular expressions, each character either contains its own lexical meaning or is at the meta level and expresses a meaning other than itself. Also, regular expressions can easily reveal the textual meaning of the given sample pattern, and adhering to this meaning, can develop multiple examples that are exactly similar to this model. In addition, regular expressions plays an important role in designing automations for processing text files, creating and developing specified text formats, and receiving arbitrary string inputs [11].

The regular expressions processor reveals regular language information that is used to describe regular languages. There is a separate grammar and syntax for each regular language, and accordingly, a separate development system for each regular expression. Moreover, regular expressions operators compile the codes given to them, test the target character string, split it into smaller strings, and reveal whether that string belongs to the regular languages given to them. Furthermore, regular expressions have a very important place in the computational world because they create a common criterion in computation. For this reason, systems of regular expressions introduce basic and secondary criteria for grammar and syntax. Also, regular expressions operators can be found and included in various search engines, some text editors, some search and replace word dialogs, as well as various scripted text compilers [12].

III. Related Algorithms and Methods

Under this title, 6 main algorithms and methods – McNaughton and Yamada's Algorithm, Glushkov's Method, Thompson's Algorithm, Berry and Sethi's Algorithm, Chang and Paige's Algorithm, and Antimirov's Method – that convert and transfer regular expressions to nondeterministic finite automata (NFA) in the field of automata theory have been tried to be explained in detail.

A. McNaughton and Yamada's Algorithm

This method is an algorithm that is used to extract state graphs from regular expressions and has a high applicability. While applying this algorithm, fewer states than "2^P+1" are generally needed and the working time of this method is at most "$\Theta(m|x|)$". ($m = \Omega(s^2)$ – the highest possible number of connections in the state graph. s: number of alphabetic characters in regular expressions. x: bit vector.) To understand this method better and more clearly, be useful to look at the example in the following [13].

Let "P = 1(00U01)*0". In this expression, there are six positions, respectively: 1_1, 0_1, 0_2, 0_3, 1_2, 0_4. If "P" is rewritten with its positions, it becomes "$1_1(0_10_2U0_31_2)^*0_4$". {1_1} is the starting state of "P" while {0_4} is the ending state of "P". When creating the state graph, the first thing to do is to consider the initial state. Next, it is decided which state this initial state will combine with according to the given input: where to go in the "0" input character or where to go in the "1" input character. The initial state in this example goes to {1_1} state at input "1" and Λ – empty – state at input "0". If a state falls into this empty state, the automation will never be able to get out of this state, which is why it is called a "dead state". Continuing, {1_1} will be in the {0_1, 0_3, 0_4} state when the input "0" comes in, and "dead state" when the input "1" comes in. {0_1, 0_3, 0_4} becomes {0_2} when "0" is input, {1_2} when "1" is input. It goes to {0_1, 0_3, 0_4} state when the input "0" comes to the {0_2} state, and goes to the "dead state" when the input "1" comes. Also, state {1_2} has the same properties as state {0_2} in terms of inputs and destinations. The created nondeterministic finite automata (NFA) in the light of this information is demonstrated in Figure 1 [13] in the following.

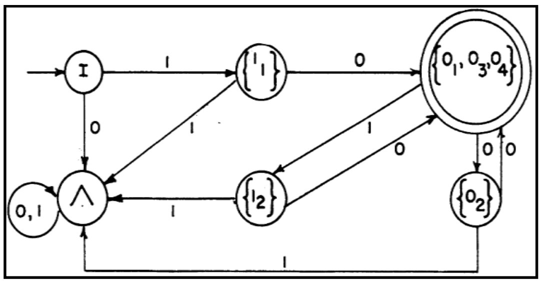


Fig. 1. NFA with McNaughton and Yamada's algorithm in the example.

B. Glushkov's Method

This method – also known as Glushkov's automation – transforms regular expressions to nondeterministic finite automata (NFA) using the subset construction algorithm. This algorithm creates a route path by matching the given string of characters with regular expressions with automation mechanism. It is decided how the states of this automation will be connected to each other with the help of this route. This algorithm uses a function on the nodes of regular expressions. With the contribution of this function, it reveals the empty and branching states in the automation, and thus all the necessary connection forms for a certain finite automata are revealed. Also, this method makes a complete transformation for NFA by including initial, acceptance and rejection states, and by eliminating redundant states in this automata [14].

C. Thompson's Algorithm

It is a method, also known as Thompson's configuration [15], which is among the algorithms that converts regular expressions to NFA. This algorithm breaks regular expressions into smaller regular expressions, and transforms these regular expressions into nondeterministic finite automata, and then combines these automata to produce the first given regular expressions translated into NFA. In addition, with this method, each character in the regular expressions is taken and tested in the order of their occurrence and transferred to the nondeterministic finite automata, thus avoiding the errors that may arise in the connections between the states and the route path. Also, this method starts parsing and splitting regular expressions from the left side and continues parsing and shredding in this way until the regular expressions are completed. In the Thompson's algorithm, the number of states is between "r-s()+1" and "2r", and the number of connections is between "r-s()" and "4r-3". (r: the length of the regular expressions. s(): number of parentheses in the regular expressions.) However, this method's runtime is up to maximum "O(r)" [15].

Based on the Thompson's algorithm, union operation is given in Figure 2, concatenation operation is demonstrated in Figure 3, and also star operation is shown in Figure 4 in the following.

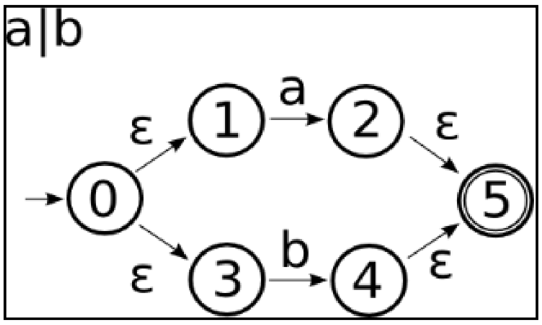


Fig. 2. Union operation based on Thompson's algorithm.

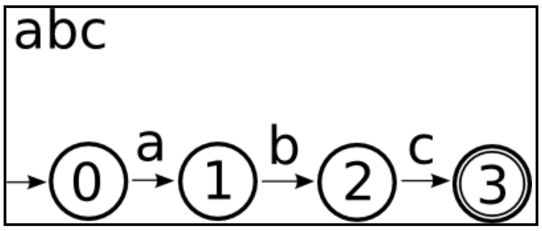


Fig. 3. Concatenation operation based on Thompson's algorithm.

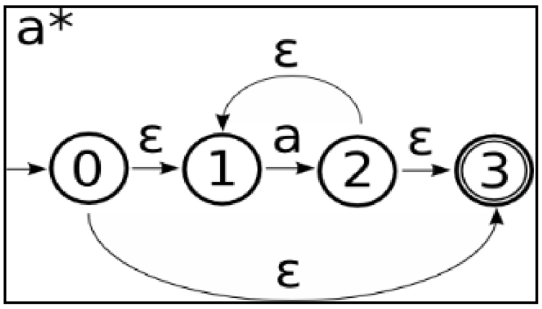


Fig. 4. Star operation based on Thompson's algorithm.

D. Berry and Sethi's Algorithm

This method is an algorithm that allows each symbol in regular expressions to appear in automation. The chart below (Figure 5 in the following) is an example of this algorithm, and reflects the regular expression "$(a_1b_2+b_3)*b_4a_5$". In the automation built with this algorithm in the following figure (Figure 5), the "C" state has connections derived from the "C" state under certain and particular symbols and characters. While creating this automation, each link derived from each state is labeled with the same symbol. For this reason, all sequences of the automation derived from the "C_3" initial state have to also be in the "wb_3" format. Thus, it turns out that the automation of regular expressions derived from the initial state "C_3", and the automation of regular expressions derived from the initial state "C_0" are the same. In the Berry and Sethi's algorithm [16], the number of states in the nondeterministic finite automata (NFA) is maximum "s+1" and the working time of this algorithm is maximum "O(r)". (r: the length of the regular expressions. s: the number of alphabetic characters in the regular expressions.)

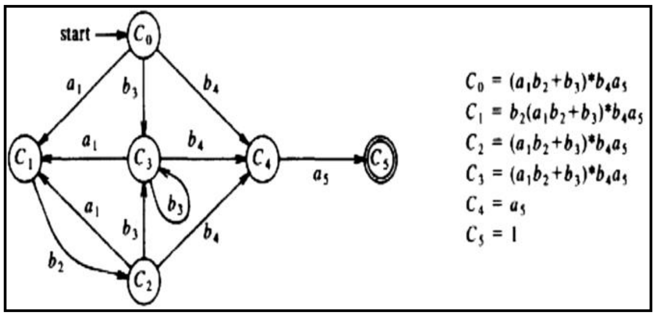

start
C_0
C_1
C_2
C_3
C_4
C_5
a_1
b_2
b_3
b_4
a_5
$C_0 = (a_1b_2+b_3)*b_4a_5$
$C_1 = b_2(a_1b_2+b_3)*b_4a_5$
$C_2 = (a_1b_2+b_3)*b_4a_5$
$C_3 = (a_1b_2+b_3)*b_4a_5$
$C_4 = a_5$
$C_5 = 1$

Fig. 5. NFA for the regular expression "$(a_1b_2+b_3)*b_4a_5$".

E. Chang and Paige's Algorithm

This method generates the transformation from regular expressions to NFA in the same asymptotic time as with Berry and Sethi's algorithm – "Θ(m)". However, it is a method that has improved the auxiliary memory – the amount of "Θ(s)" – held in the memory of Berry and Sethi's method (s: the number of alphabetic characters in the regular expressions) [17].

The McNaughton and Yamada's algorithm lacks the number of connections between states in automation. Because if it is taken the worst case, the number of connections is "$m = \Theta(s^2)$". Also, if it is taken the expression "s()" as the number of parentheses (right or left), Thompson's nondeterministic finite automata (NFA) is the case between the numbers "r-s()+1" and "2r", as well as it has a connection between "r-s()" and "4r-3" (r is the length of the regular expressions.). For these reasons, a new compressed data structure has emerged – "Compressed Nondeterministic Finite Automata: CNNFA". This structure transfers regular expressions to a NFA occupying "Θ(r)" time and "O(s)" amount of memory [17].

F. Antimirov's Method

The corresponding algorithm converts a regular expression "t" into a nondeterministic finite automata (NFA) [18]. It uses the most "|t|+1" status while doing this operation. The details of this algorithm are explained in detail in the following.

In the given regular expression "t" according to the alphabet "A", let the automata be "M" and also, let the state sets be "M = PD(t)". The initial state is "$\mu_0 = t$", the coupling function is "$\tau(p,x) = \eth_x(p)$" – p ϵ PD(t), x ϵ A – and the end state sets are "F = {p ϵ PD(t) | o(p) = λ}". Accordingly, the "M" automata is be able to recognize the "L(t)" regular language. To put this construction structure into practice, the functions "PD(t)" and "τ" have to be calculated, and the following operations have to be repeated sequentially [18].

$<PD_0, \Delta_0, \tau_0> := < \emptyset, \{t\}, \emptyset >$

$PD_{i+1} := PD_i \cup \Delta_i$

$\Delta_{i+1} := p \in \Delta_i \cup \{q \mid <x, q> \in if(p) \wedge q -\in PD_{i+1}\}$

$\tau_{i+1} := \tau_i \cup \{ <p, x, q> \mid p \in \Delta_i \wedge <x, q> \in if(p)\}$

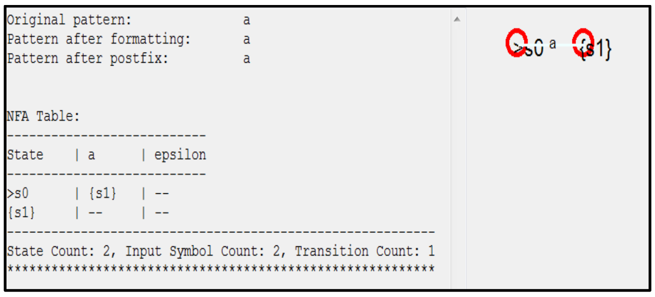
The working time of this algorithm is between "O(n)" and "$O(n^2)$" (n is the length of the given regular expressions.).

IV. Application – NFA Conversion Tool

A tool based on *Thompson Algorithm* [19-22] has been designed and developed in order to convert regular expressions to nondeterministic finite automata (NFA). This is a new and original application for modelling NFA according to given characters and expressions. NFA conversion tool works and performs based on the following directions.

With the arrival of the data, the NFA function starts to work. Each character in the regular expression is detected. These characters may be either a term or an action. Terms can be letters or numbers. Operations are union, concatenation and star operators. The union operation is given as the character "|", the concatenation operation is demonstared as the character "." and the star operation is also expressed with the character "*". When these characters are specified, it is understood that the terms will enter an operation. Also, the parentheses "(", ")" help determine the priority of the operations. According to the rules of these operations, it is determined that the terms, that are letters or numbers, go from which state to which state. When the NFA function completes running (finishes), all the states in the nondeterministic finite automata (NFA), and their trajectories appear exactly right. Thus, the necessary data for drawing the NFA are obtained.

Based on the application which have been designed and developed for modelling NFA, several regular expressions have been converted into nondeterministic finite automata in the study for showing its efficiency and usability: in Figure 6, the expression "a", in Figure 7, the expression "abc", in Figure 8, the expression "a*bc", and in Figure 9, the expression "(ab|bc)*" have separated, operated, transformed and drawn in the following.

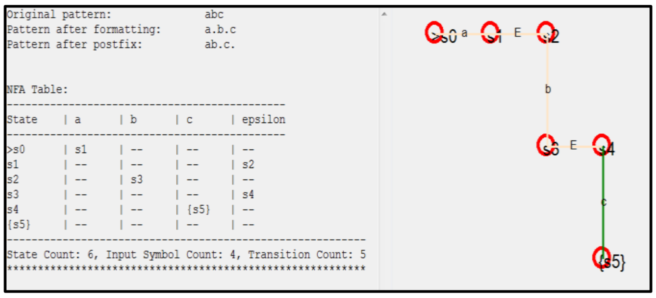


```
Original pattern:               a
Pattern after formatting:       a
Pattern after postfix:          a


NFA Table:
---------------------------
State    | a      | epsilon
---------------------------
>s0      | {s1}   | --
{s1}     | --     | --

----------------------------------------------------------
State Count: 2, Input Symbol Count: 2, Transition Count: 1
**********************************************************
```

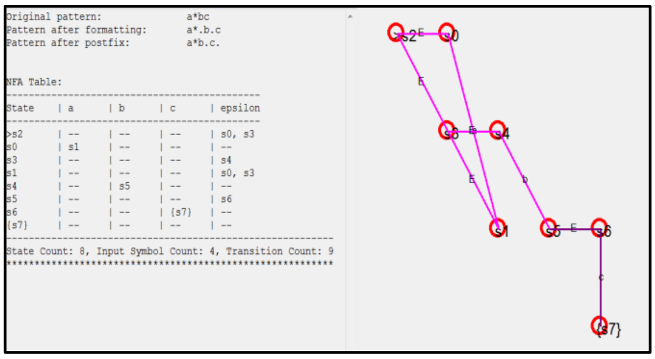
Fig. 6. NFA for the expression “a” in the developed tool in this study.



```
Original pattern:               abc
Pattern after formatting:       a.b.c
Pattern after postfix:          ab.c.


NFA Table:
---------------------------------------------
State    | a       | b       | c       | epsilon
---------------------------------------------
>s0      | s1      | --      | --      | --
s1       | --      | --      | --      | s2
s2       | --      | s3      | --      | --
s3       | --      | --      | --      | s4
s4       | --      | --      | {s5}    | --
{s5}     | --      | --      | --      | --
----------------------------------------------------------
State Count: 6, Input Symbol Count: 4, Transition Count: 5
**********************************************************
```

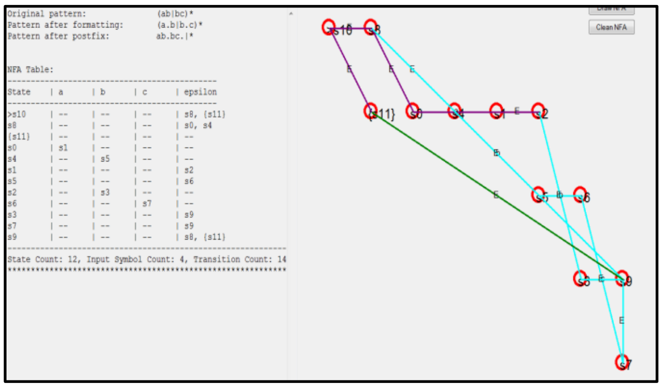
Fig. 7. NFA for the expression “abc” in the developed tool in this study.



```
Original pattern:               a*bc
Pattern after formatting:       a*.b.c
Pattern after postfix:          a*b.c.


NFA Table:
---------------------------------------------
State    | a       | b       | c       | epsilon
---------------------------------------------
>s2      | --      | --      | --      | s0, s3
s0       | s1      | --      | --      | --
s3       | --      | --      | --      | s4
s1       | --      | --      | --      | s0, s3
s4       | --      | s5      | --      | --
s5       | --      | --      | --      | s6
s6       | --      | --      | {s7}    | --
{s7}     | --      | --      | --      | --
----------------------------------------------------------
State Count: 8, Input Symbol Count: 4, Transition Count: 9
**********************************************************
```

Fig. 8. NFA for the expression “a*bc” in the developed tool in this study.



```
Original pattern:               (ab|bc)*
Pattern after formatting:       (a.b|b.c)*
Pattern after postfix:          ab.bc.|*


NFA Table:
---------------------------------------------
State    | a       | b       | c       | epsilon
---------------------------------------------
>s10     | --      | --      | --      | s8, {s11}
s8       | --      | --      | --      | s0, s4
{s11}    | --      | --      | --      | --
s0       | s1      | --      | --      | --
s4       | --      | s5      | --      | --
s1       | --      | --      | --      | s2
s5       | --      | --      | --      | s6
s2       | --      | s3      | --      | --
s6       | --      | --      | s7      | --
s3       | --      | --      | --      | s9
s7       | --      | --      | --      | s9
s9       | --      | --      | --      | s8, {s11}
------------------------------------------------------------
State Count: 12, Input Symbol Count: 4, Transition Count: 14
************************************************************
```

Fig. 9. NFA for the expression “(ab|bc)*” in the developed tool in this study.

V. Conclusion

Regular expressions are a representation in automata that make up regular languages. With the contribution of this notation, a given sample character string model can be tested; a text can be replaced with another; a smaller character string belonging to that string can be extracted from or added into a character string depending on the pattern match; word, text or symbol analysis can be performed. Also, regular expressions can be used effectively in search engines, natural language processing, parallel programming and neural networks. That means, regular expressions have an important role in computation area. With the help of this study, a tool has been designed and developed for that regular expressions are converted into nondeterministic finite automata (NFA). So, this tool may provide to understand more clearly and to benefit more effectively regular expressions for anyone who is interested in computing theory. In addition, this conversion application may be used in the studies and the researches about regular expressions. Thus, the tool, which have been designed and developed in this study, will have had a positive effect on both academic literature and computational industry.

Acknowledgment

Upon request, the executable file of the developed conversion tool and its source code within the scope of the study can be shared with the researchers who are interested in. Also, ethics committee approval form and document is not required for this study, and the authors declare that there is no conflict of interest.

References

[1] Horswill, I. (2008). What is computation? Obtained from https://users.cs.northwestern.edu/~ian/What%20is%20computation.pdf (Last access on 19th August 2021)

[2] Harvey, B. (1997). *Computer science logo style (1st ed.).* USA: The MIT Press.

[3] Dlugosch, P., Brown, D., Glendending, P., Leventhal, M., & Noyes, H. (2014). An efficient and scalable semiconductor architecture for parallel automata processing. *IEEE Transactions on Parallel and Distributed Systems, 25*(12), 3088-3098.

[4] Lucas, S. M., & Reynolds, T. J. (2005). Learning deterministic finite automata with a smart state labeling evolutionary algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*(7), 1063-1074.

[5] Holzer, M., & König, B. (2004). On deterministic finite automata and syntactic monoid size. *Theoretical Computer Science, 327*(3), 319-347.

[6] Yang, L. Karim, R., Ganapathy, V., & Smith, R. (2011). Fast, memory-efficient regular expression matching with NFA-OBDDs. *Computer Networks, 55*(15), 3376-3393.

[7] Pao, D, Lam, N., & Cheung, R. A memory-based NFA regular expression match engine for signature-based intrusion detection. *Computer Communications, 36*(10-11), 1255-1267.

[8] Hopcroft E., Motwani, R., & Ullman, D. (2007). *Introduction to automata theory, languages, and computation (3rd ed.).* Britain: Pearson.

[9] Konstantinidis, S. (2007). Computing the edit distance of a regular language. *Information and Computation, 205*(9), 1307-1316.

[10] Owens, S., Reppy, J., & Turon, A. (2009). Regular-expression derivatives re-examined. *Journal of Functional Programming, 19*(2), 173-190.

[11] Lee, T. (2009). Hardware architecture for high-performance regular expression matching. *IEEE Transactions on Computers, 58*(7), 984-993.

[12] Kumar, S., Dharmapurikar, S., Yu, F., Crowley, P., & Turner, J. (2006). Algorithms to accelerate multiple regular expressions matching for deep packet inspection. *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications,* 339-350.

[13] McNaughton, R., & Yamada, H. (1960). Regular expressions and state graphs for automata. *IRE Transactions on Electronic Computers, 9*(1), 39-47.

[14] Kurai, R., Yasuda, N., Arimura, H., Nagayama, S., & Minato, S. (2014). Fast regular expression matching based on dual Glushkov NFA. *Proceedings of PSC 2014,* 3-16.

[15] Thompson, K. (1968). Programming techniques: Regular expression search algorithm. *Communications of the ACM, 11*(6), 419-422.

[16] Berry, G., & Sethi, R. (1986). From regular expressions to deterministic automata. *Theoretical Computer Science, 48*(1), 117-126.

[17] Chang, C. (1992). *From regular expressions to DFA's using compressed NFA's.* PhD Thesis, New York University, New York.

[18] Antimirov, V. (1996). Partial derivatives of regular expressions and finite automaton constructions. *Theoretical Computer Science, 155*(2), 291-319.

[19] Brzozowski, J. A. (1964). Derivatives of regular expressions. *J. ACM 11*(4), 481-494.

[20] Kleene, S. C. (1956). *Representation of events in nerve nets and finite automata.* In Automata Studies, Ann. Math. Stud. No. 34. Princeton U. Press, Princeton, N. J., 3-41.

[21] IBM Corp. *IBM 7094 principles of operation.* File No. 7094-01, Form A22-6703-1.

[22] Kuno, S., & Oettinger, A. G. (1962). Multiple-path syntactic analyzer. Proc. *IFIP Congress,* Munich.