Conversion of Deterministic and Non-Deterministic Finite Automata to Regular Expression using Brzozowski Algebraic Method

[1] *Daniel, Musa Alih and* [2] *Akpa, Johnson*

[1] Department of Mathematical Sciences, Kogi State University, Anyigba, Kogi State.

[2] Department of Mathematical Sciences, Kogi State University, Anyigba, Kogi State.

ABSTRACT:

Regular expressions are widely used in the field of compiler design, text editor, search for an email- address, train track switches, pattern matching, context switching and in many other areas of computer science. Regular expressions are used to represent certain set of string in algebraic manner. The demand of converting regular expression into finite automata and vice versa motivates research into some alternative so that time taken for above is minimized. For conversion of deterministic and non-deterministic finite automata to regular expression, Brzozowski Algebraic method, also known as Arden's Theorem is used in this paper because of its simplicity and straight forwardness.

Keywords: Brzozowski, Regex, Automata, DFA, NDFA, Transition, State, Elimination.

I. INTRODUCTION

Regular expressions have been widely studied due to their expressiveness and flexibility for various applications [13]. Regular expressions are used to represent certain set of string in algebraic manner [6]. The conversion of regular expressions into finite state automata and finite state automata into regular expression is an important area of research in automata theory [5] and [16]. The notion of derivatives of regular expressions has been introduced to make the construction of finite state automata from regular expressions in a natural way [14]. A regular expression, regex or regexp (sometimes called a rational expression) is a sequence of characters that define a search pattern. Regular expressions are used to represent certain set of string in algebraic manner [10]. Finite state machines (or finite automata) provide a computational model for implementing recognizers for regular languages [8], [11] and [14]. Regular language and finite automata play a crucial role in pattern matching. Regular expression is used to specify certain pattern of interest and Non deterministic automata and Deterministic automata are the models to recognize the pattern. Deterministic Finite Automata plays a vital role in lexical analysis phase of compiler design, Control Flow graph in software testing, Machine learning [16], etc. Finite state machine or finite automata is classified into two. These are Deterministic Finite Automata (DFA) and non-deterministic Finite Automata(NFA).
In this paper, Regular expression can be converted from one form to another. For conversion of Deterministic Finite Automata (DFA) and non-deterministic Finite Automata (NFA) to regular expression, following methods have been introduced-Transitive Closure, State Elimination and Brzozowski Algebraic methods. The conversion technique or method adopted in this paper is Brzozowski Algebraic Method using Arden's Theorem [1], [3], [7] and [10]. Brzozowski [9], because it is a unique approach for converting both deterministic finite automata and non-deterministic finite automata to regular expressions. In this approach, characteristic equations for each state are created which represent regular expression for that state. Regular expression equivalent to deterministic or non-deterministic finite automata is obtained after solving the equations. If it has more one final states, the said states will be added together to obtain the regular expression accordingly.

II. BASIC DEFINATION

[A] Deterministic Finite Automata (DFA)

In DFA, for each input symbol, one can determine the state to which the machine will move. Hence, it is called Deterministic Automata. As it has a finite number of states, the machine is called Deterministic Finite Machine or Deterministic Finite Automata.

Formal Definition of a DFA

A DFA can be represented by a 5-tuple ($Q, \sum, \delta, q_{0,} F$) where −

- Q is a finite set of states.
- $\sum$ is a finite set of symbols called the alphabet.
- δ is the transition function where $\delta : Q \times \sum \to Q$.
- q_0 is the initial state from where any input is processed $(q_0 \in \mathrm{Q})$.
- F is a set of final state/states of Q $(\mathrm{F} \subseteq \mathrm{Q})$.

Graphical Representation of a DFA

A DFA is represented by digraphs called **state diagram**.

- The vertices represent the states.

- The arcs labelled with an input alphabet show the transitions.
- The initial state is denoted by an empty single incoming arc.
- The final state is indicated by double circles.

Transition functions can also be represented by transition table as shown in table 1.

Table 1: Transition Table representing transition function of DFA

Present/current State	Next State for Input 0	Next State for Input 1
x	x	y
y	z	x
z	y	z

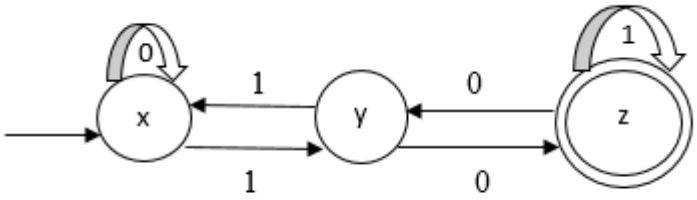

Fig. 1: Transition Diagram of DFA Table 1

[B] Non-Deterministic Finite Automata (NFA)

In DFA, for a particular input symbol, the machine can move to any combination of the states in the machine. In other words, the exact state to which the machine moves cannot be determined. Hence, it is called Non-deterministic Automaton. As it has finite number of states, the machine is called Non-deterministic Finite Machine or Non-deterministic Finite Automaton.

Formal Definition of an NDFA

An DFA can be represented by a 5-tuple ($Q, \sum, \delta, q_{0,} F$) where −

- Q is a finite set of states.
- $\sum$ is a finite set of symbols called the alphabets.
- δ is the transition function where $\delta : \mathrm{Q} \times \sum \to 2^{\mathrm{Q}}$

  (Here the power set of Q (2^{Q}) has been taken because in case of NDFA, from a state, transition can occur to any combination of Q states)
- **q₀** is the initial state from where any input is processed ($q_0 \in \mathrm{Q}$).
- **F** is a set of final state/states of Q $(\mathrm{F} \subseteq \mathrm{Q})$.

Graphical Representation of Deterministic Finite Automata (DFA):

An DFA is represented by digraphs called state diagram.

- The vertices represent the states.
- The arcs labelled with an input alphabet show the transitions.
- The initial state is denoted by an empty single incoming arc.
- The final state is indicated by double circles.

Example

Let a non-deterministic finite automaton be →

- $Q = \{x, y, z\}$
- $\sum = \{0, 1\}$
- $q_0 = \{x\}$
- $F = \{z\}$

The transition function δ as shown below

Table 2: Transition Table representing transition function of NFA

Present/current State	Next State for Input 0	Next State for Input 1
x	x, y	y
y	z	x, z
z	y, z	z

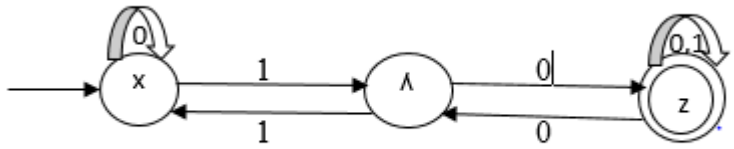

Fig. 2: Transition diagram of Table 2

[C] Regular Expression (RE)

A regular expression (RE) is a pattern that describes some set of strings. Regular expression over a language can be defined by [12] as:

1) Regular expression for each alphabet will be represented by itself. The empty string (ϵ) and null language (ϕ) are regular expression denoting the language {ϵ} and {ϕ} respectively.

2) If E and F are regular expressions denoting the languages $L(E)$ and $L(F)$ respectively, then following rules can be applied recursively.

- Union of E and F will be denoted by regular expression $E + F$ and representing language $L(E)$ ∪ $L(F)$.
- Concatenation of E and F denoted by EF and representing language $L(E * F) = L(E) * L(F)$.
- Kleene closure will be denoted by E^* and represent language (L (E))*.

3) Any regular expression can be formed using 1-2 rules only.

III. CONVERSIONS BETWEEN REGULAR EXPRESSION AND AUTOMATA

This section is mostly concerned with the description of technique or method used to convert deterministic and non-deterministic finite automata to regular expression.

First, Kleene [15] proves that every RE has equivalent DFA and vice versa. On the basis of this theoretical result, it is clear that DFA can be converted into RE and vice versa using some algorithms or techniques [2] and [4].

Brzozowski Algebraic Method

Brzozowski method [1] and [5] is a unique approach for converting deterministic and non-deterministic finite automata to regular expressions. In this approach, characteristics equations are created for each state which represent regular expression for that state by using Arden's Theorem [14]. This Theorem states that if P and Q are two regular expressions over ∑, and if P does not contain ε, then the following equation in R given by:

$R = Q + RP$ has a unique solution, i.e.

$R = QP^*$

Now, let's see how it works.

$R = Q + RP$

$= Q + \mathrm{QP} * \mathrm{P}$

$= Q(\varepsilon + \mathrm{P} * \mathrm{P})$, but [$\varepsilon + R * R = R^*$, and $\varepsilon R = \mathrm{R}$, identity law]

Also,

$R = \mathrm{Q} + \mathrm{RP}$

$= Q + (\mathrm{Q} + \mathrm{RP})\mathrm{P}$

$= Q + \mathrm{QP} + \mathrm{RP}^2$

$= Q + \mathrm{QP} + (\mathrm{Q} + \mathrm{QP})\mathrm{P}^2$

$= Q + \mathrm{QP} + \mathrm{QP}^2 + \mathrm{QP}^3$

\- - - -
\- - - -
\- - - -

$= Q + \mathrm{QP} + \mathrm{QP}^2 + \ldots + \mathrm{QP}^n + \mathrm{RP}^{n+1}$

$= Q + \mathrm{QP} + \mathrm{QP}^2 + \ldots + \mathrm{QP}^n + \mathrm{QP}^*\mathrm{P}^{n+1}$, but $R = \mathrm{QP}^*$

$= Q(1 + \mathrm{P} + \mathrm{P}^2 + \ldots + \mathrm{P}^n + \mathrm{P}^*\mathrm{P}^{n+1})$

$= Q(\varepsilon + \mathrm{P} + \mathrm{P}^2 + \ldots + \mathrm{P}^n + \mathrm{P}^*\mathrm{P}^{n+1}$

$= Q(\mathrm{P}^*), (\varepsilon + \mathrm{P} + \mathrm{P}^2 + \ldots + \mathrm{P}^n + \mathrm{P}^*\mathrm{P}^{n+1} = \mathrm{P}^*)$

$R = \mathrm{QP}^*$

Now, let us consider the state diagram in the figure 3 below.

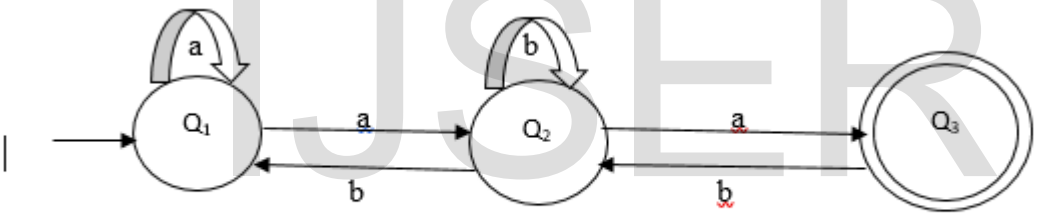

Fig. 3: Transition diagram of Deterministic Finite Automata (DFA)

From the above diagram in fig. 3 above, the first to do is to write down the characteristic equations that correspond transition diagram.
This implies that,

$Q_3 = \mathrm{Q}_2\mathrm{a}$ (1)

$Q_2 = \mathrm{Q}_1\mathrm{a} + \mathrm{Q}_2\mathrm{b} + \mathrm{Q}_3\mathrm{b}$ (2)

$Q_1 = \varepsilon + \mathrm{Q}_1\mathrm{a} + \mathrm{Q}_2\mathrm{b}$ (3)

From (1) above, substitute the value of Q_2.

$Q_3 = \mathrm{Q}_2\mathrm{a}$

Now becomes,

$Q_3 = (\mathrm{Q}_1 a + \mathrm{Q}_2\mathrm{b} + \mathrm{Q}_3\mathrm{b})\mathrm{a}$ (4)

$Q_3 = \mathrm{Q}_1\mathrm{aa} + \mathrm{Q}_2\mathrm{ab} + \mathrm{Q}_3\mathrm{ab}$

Also, putting the value of Q_3 in (1) into Q_2, we have

$Q_2 = \mathrm{Q}_1\mathrm{a} + \mathrm{Q}_2\mathrm{b} + (\mathrm{Q}_2\mathrm{a})\mathrm{b}$

$Q_2 = \mathrm{Q}_1\mathrm{a} + \mathrm{Q}_2\mathrm{b} + \mathrm{Q}_2\mathrm{ab}$

$$Q_2 = Q_1 a + Q_2 (b + ab) \quad (5)$$

By using Arden's Lemma or Theorem (where $R = Q + RP$ and $R = QP^*$), and compare the units of the equation, we also have,

$\underbrace{Q_2}_{R} = \underbrace{Q_1 a}_{Q} + \underbrace{Q_2}_{R}\underbrace{(b + ab)}_{P}$

Therefore,

$$Q_2 = Q_1 a (b + ab)^* \quad (6)$$

Also, from equation (3),

$Q_1 = \varepsilon + Q_1 a + Q_2 b$,

Putting the value of Q_2 from (5) into (3)

$$Q_1 = \varepsilon + Q_1 a + \{(Q_1 a)(b + ab)^*\} b$$

$$Q_1 = \varepsilon + Q_1 \{a + a (b + ab)^*\} b \quad (7)$$

Also by comparing (7) above using Arden's Theorem, we have,

$\underbrace{Q_1}_{R} = \underbrace{\varepsilon}_{Q} + \underbrace{Q_1}_{R}\underbrace{\{a + a (b + ab)*\}b}_{P}$

$$Q_1 = \varepsilon \{a + a (b + ab)^* b\}^* \quad (8)$$

And finally from final state in (3), and putting the value of Q_2 from (6) and the value of Q_1 from (7), we have,

$$Q_3 = Q_2 a$$

$$Q_3 = Q_1 a (b + ab) * a \quad (9)$$

$$Q_3 = \{a + a(b + ab) * b\} * a (b + ab) * a \quad (10)$$

The equation (10) above is the required Regular Expression (RE) from Deterministic Finite Automata (DFA).

Also, let us consider the transition diagram NFA in fig. 4 below.

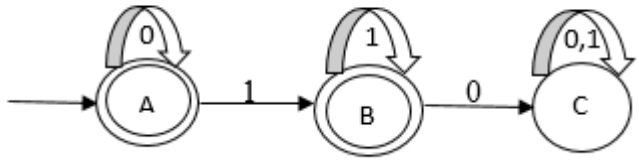

Fig. 4: Transition Diagram of Non-deterministic Finite Automata (NFA)

Again, from the transition diagram in fig. 4 above, the characteristics or corresponding equations are written down as follows.

$$A = \varepsilon + A0 \quad (11)$$

$$B = A1 + B1 \quad (12)$$

$$C = B0 + C0 + C1 \quad (13)$$

Now, let us consider the final state, A.

That is,

$$A = \varepsilon + A0$$

Looking at the equation above, it is of the form, $R = Q + RP$

$\underbrace{A}_{R} = \underbrace{\varepsilon}_{Q} + \underbrace{A}_{R}\underbrace{0}_{P}$

$$A = \varepsilon 0^*, (R = QP^*)$$

$$A = 0^*, (\varepsilon R = R) \quad (14)$$

Also, we consider the second part of the final states, (12).

$B = \mathrm{A1} + \mathrm{B1}$

By substituting the value of $A = 0*$ in (14) into (12), we have,

$B = 0*1 + \mathrm{B1}$, and compare with Arden's Theorem ($R = \mathrm{Q} + \mathrm{RP}$),

$\underset{R}{B} = \underbrace{0*1}_{Q} + \underset{R}{B}\underset{P}{1}$

$B = 0*1(1)*$ (15)

Lastly, to get the regular expression, we add both the final states (A and B) results.

Regular Expression (RE) = $0* + 0*1(1)*$

$= 0* + (\varepsilon + 11*)$

$= 0* + (\varepsilon + 1*1)$

$= 0*1*,$ $(\varepsilon + \mathrm{R}*\mathrm{R} = \mathrm{R}*)$

IV. CONCLUSION

This research paper provides a great insight into the technique or method adopted-Brzozowski Algebraic Method. The algebraic approach is elegant, leans toward a recursive approach, and generates reasonably compact regular expressions. Brzozowski's method is particularly suited for recursion-oriented languages, such as functional languages. This method has smoothened and narrowed down all the worries of researchers who particularly may find other methods of conversion more difficult and abstract as it is straight forward once the characteristic equations are formed according to the respective state transitions.

REFERENCES

[1] Alfred V. Aho, "Constructing a Regular Expression from a DFA", Lecture notes in Computer Science Theory, September 27, 2010, Available at http://www.cs.columbia.edu/~aho/cs3261/lectures.

[2] C. Neumann, "Converting Deterministic Finite Automata to Regular Expressions". Available at: http://neumannhaus.com/christoph/ papers 2005-03- 16.DFA_to_RegEx.pdf, 2005.

[3] Dean N. Arden. Delayed-logic and Finite-state Machines. In Theory of Computing Machine Design, Pages 1–35. U. of Michigan Press, Ann Arbor, MI, 1960.

[4] Ding-Shu Du and Ker-I Ko, "Problem Solving in Automata, Languages, and Complexity", John Wiley & Sons, New York, NY, 2001.

[5] G. Hermann and H. Markus, "From Finite Automata to Regular Expressions and Back—A Summary on Descriptional Complexity". 2014.
Available at: https://www.researchgate.net/publication/262568811

[6] Indu, Jyoti. "Technique for Conversion of Regular Expression to and from Finite Automata". International Journal of Recent Research Aspects, Vol. 3(2), Pages 62-64, 2016.

[7] J. Daintith, "Arden's Rule". A Dictionary of Computing.
https://www.encyclopedia.com

[8] J. Mahak, "Theory of Computation: Deterministic Finite Automata (DFA), 2018. Available at https://www.includehelp.com.

[9] Janusz A. Brzozowski, "Derivatives of Regular Expressions", J. ACM, 11(4) Pages 481-494, 1964.

[10] K. Neha and A. Sharma, "Conversion of Regular Expression in Finite Automata". International Journal of Scientific Research & Management, Vol. 3, No.5, 2015.

[11] K. Neha and A. Sharma, "Methods of Regular Expression". International Journal for Research in Applied Science & Engineering Technology (IJRASET), Vol. 3, Issue 9, 2015.

[12] L. W. Smith and S. S. Yau, "Generation of Regular Expressions for Automata by the Integral of Regular Expressions". The Computer Journal, Vol. 15(3), Pages 222-228, 1972.

[13] Lixiao Z., Shuai M., and Y. G. Wang. String Generation for Testing Regular Expression. The Computer Journal, 2019.
http://doi.org/10.1093/comjnl/bxy137.

[14] N. Murugesan and O.V. Shanmuga Sundaram, “Computation of Regular Expression Derivatives”. International Journal of Computing Science & Mathematics, Vol. 7(3), 2016.
[15] S. C. Kleene. Representation of Events in Nerve Nets and Nite Automata. In Automata Studies, Pages 3{40. Ann. Of Math. Studies No. 34, Princeton University Press, Princeton, NJ, 1956.
[16] Z. Jeilan and Z. Qian, “The Equivalent Conversion between Regular Grammar and Finite Automata”. Journal of Software Engineering & Application, Vol. 6(1), PP. 33-37, 2013.