International Journal of Applied Science and Engineering
Citation: *IJASE*: **7**(2): 129-133, December 2019
DOI: 10.30954/2322-0465.2.2019.3

A Method to Convert Regular Expression into Non-Deterministic Finite Automata

Abhishek Singh*, Dhwani Agrawal, Reena Chaudhary and Rashmi Chaudhary

Department of Computer Science and Technology, Central University of Punjab, India

*Corresponding author: abhi29101994@gmail.com

Abstract

Non-deterministic finite automata are a part of finite automata which can accept only valid strings to process. It is needed to accept valid strings. Regular expression is used for pattern matching for strings. Regular expression cannot give the number of states for a particular automaton. That's why, we need NFA through which we can state total number of states in any automata. There are several methods to convert non-deterministic finite automata into regular expression but there are few to convert regular expression to non-deterministic finite automata. In this paper, we proposed a method through which one can convert regular expression to non-deterministic finite automata by decomposing the input string with the help of JFLAP tool.

Keywords: Regular Expression, Operator, Non-Deterministic Finite Automata, Conversion

An expression over the alphabet Σ using the operator (* . +) is called as regular expression. Regular expression is a pattern to generate set of string every Regular expression generate only one regular language but a regular language can be generated by more than one form of regular expression that is regular expression is not unique. Regular expression is just like NFA and generates only the string of Regular expression (only valid string). There are many principal methods for converting regular expression to NFA one is due to Mc Naughton & Yamada and another is Thomson. In this paper only theoretical concept of converting Regular expression to NFA. In NFA no need to define the transition for each and every input symbol at each and every state. Transition path in NFA is not unique corresponding to any input string.

Regular operator

Operator	Description
*	Kleene Closure
.	Concatenation
+	Positive Closure

Received: 10 June, 2019 **Revised:** 12 Sept., 2019 **Accepted:** 25 Nov., 2019

Properties

Closure Property: Regular expression satisfies the closure properties with respect to Concatenation Positive Closure and Kleene Closure.

Associative Property: Regular expression satisfies associative properties with respect to Positive Closure and Concatenation but with not respect to Kleene Closure.

Identity Property: Let R is any Regular Expression if there exist a Regular expression ‘x’ such that,

R+ x = R

R.R = R

Here ‘*x*’ is called identity element.

Annihilator Property: Let ‘*r*’ is a Regular expression if there exist a Regular expression ‘*x*’

$R + x = x$

$R.\ x = x$ ($x = \phi$, R.$\phi = \phi$)

No annihilator with respect to ‘+’

Idempotent property: Regular expression satisfies Idempotent property with respect to Concatenation.

Commutative Property: Regular expression satisfies the Commutative property with respect to Positive Closure but not with Concatenation.

Distributive Property: Regular expression satisfies the distributive property with respect to Concatenation and Positive Closure.

Methodology

Method of decomposition (state creation method):

- ❒ Take a two-state machine by taking a regular expression as input.
- ❒ Start decomposing the regular expression string into symbols step by step.
- ❒ If ‘*r*’ is a Regular Expression then take ‘*r*’ as an edge for the system which has one initial state and one final state
- ❒ Decompose the string until it is divided into symbols by creating the strings.
- ❒ Using this mechanism, we directly obtained NFA from the Regular expression.

Experiment and Results

Input: Regular Expression

Output: Non deterministic Finite Automata

Regular expression: "b*abab"

According to algorithm take two states:

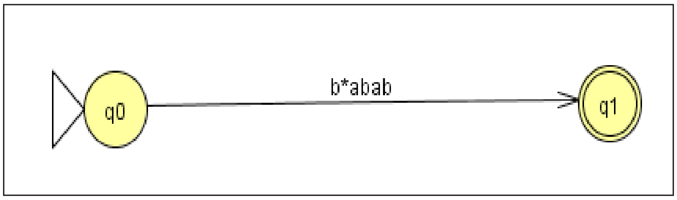


Fig. 1: Giving the regular expression to machine

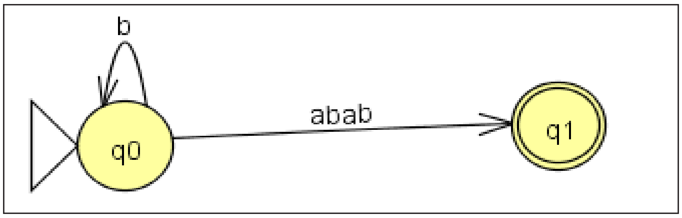


Fig. 2: Decomposing the regular expression (step 1)

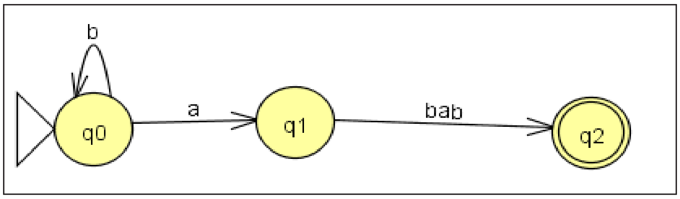


Fig. 3: Decomposing the regular expression (step 2)

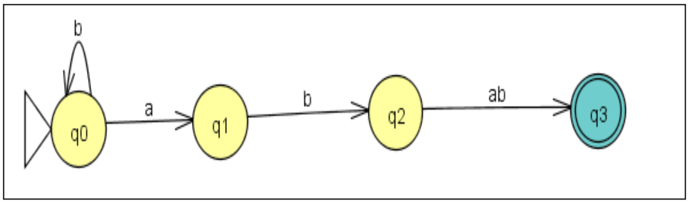


Fig. 4: Decomposing the regular expression (step 3)

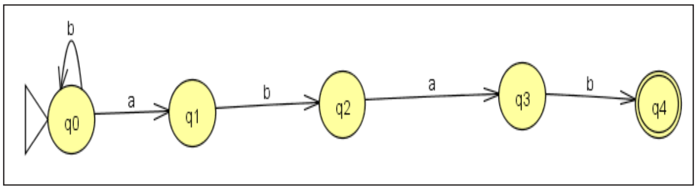
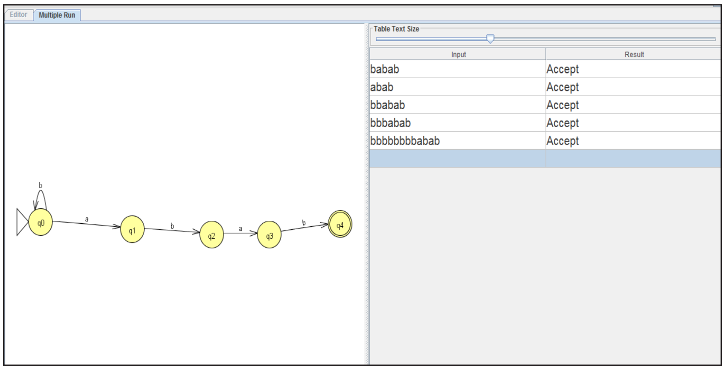


Fig. 5: Final NFA after decomposing regular expression into symbols



Input	Result
babab	Accept
abab	Accept
bbabab	Accept
bbbabab	Accept
bbbbbbbbabab	Accept

Fig. 6: String acceptance on converted NFA

From Figs. 1-6, it is clear that the method is successful to convert a regular expression into a non-deterministic finite automaton by decomposing the regular expression into each possible symbol.

CONCLUSION

In this paper, we have proposed a method to convert a regular expression into non-deterministic finite automata by decomposing the given regular expression into symbols one by one. All the experiments are done in the default settings in JFLAP tool. From the results, it is clear that the method is giving the accurate results and it does not change the language after the conversion. This is shown in the figure 5, as in this figure strings are shown that are accepted by the NFA in the form of starting with any number of 'b' and ending with 'abab', which was the input regular expression. The main advantage for this method is that it can also convert the regular expression into ε-NFA.

REFERENCES

1. Almeida, M., Moreira, N. and Reis, R. 2007. On the performance of automata minimization algorithms. *In Proceedings of the 4th Conference on Computation in Europe: Logic and Theory of Algorithms*, pp. 3-14.

2. Amit Kishor Shukla, A.S. 2015. State Minimization Approach in Deterministic Finite Automata using Inefficient State Elimination Approach. *International Journal of Advance Research in Computer Science and Management Studies*, **3**(12): 20-28.

3. Basten, H.J.S. 2011. Ambiguity detection methods for context-free grammars *(Doctoral dissertation, University of Amsterdam)*.

4. Berry, G. and Sethi, R. 1986. From Regular Expressions to Deterministic Automata, *Theoretical Computer Science*, **48**: 117-126.

5. Bruggemann Klein A. 1992. "Regular Expressions into Finite Automata," *Springer link Lecture notes in Computer Science*, **583**: 87-98.

6. Champarnaud, J.M., Ponty, J.L. and Ziadi, D. 1999. From regular expressions to Finite Automata. *International Journal of Computer Mathematics*, **72**(4): 415-431.

7. Chang, C.H. and Paige, R. 1992. From regular expressions to dfa's using compressed nfa's. *In Annual Symposium on Combinatorial Pattern Matching*, pp. 90-110. Springer, Berlin, Heidelberg.

8. Chang, C.H. and Paige, R. 1997. From regular expressions to DFA's using compressed NFA's. *Theoretical Computer Science*, **178**(1-2): 1-36.

9. Chhabra, T. and Kuymar, A. 2012. New heuristics for conversion of Deterministic Finite Automata to Regular expression. *International Journal of Advanced Research in Computer Science*, **3**(4).

10. Choubey, A. and Ravi, K.M. 2013. Minimization of deterministic Finite Automata with vague (final) states and intuitionistic fuzzy (final) states. *Iranian Journal of Fuzzy Systems,* **10**(1): 75-88.

11. Gruber, H. and Holzer, M. 2015. From Finite Automata to regular expressions and back—a summary on descriptional complexity. *International Journal of Foundations of Computer Science,* **26**(08): 1009-1040.