International Journal of Advances in Electronics and Computer Science, ISSN: 2393-2835 Volume-5, Issue-5, May.-2018
http://iraj.in

USING JFLAP IN ACADEMICS TO UNDERSTAND AUTOMATA THEORY IN A BETTER WAY

KAVITA TEWANI

Assistant Professor, Institute of Technology, Nirma University
E-mail: kavitatewani012@nirmauni.ac.in

Abstract - As it is been observed that the things get much easier once we visualize it and hence this paper is just a step towards it. Usually many subjects in computer engineering curriculum like algorithms, data structures, and theory of computation are taught theoretically however it lacks the visualization that may help students to understand the concepts in a better way. The paper shows some of the practices on tool called JFLAP (Java Formal Languages and Automata Package) and the statistics on how it affected the students to learn the concepts easily. The diagrammatic representation is used to help the theory mentioned.

Keywords - Automata, JFLAP, Grammar, Chomsky Heirarchy , Turing Machine, DFA, NFA

I. INTRODUCTION

The theory of computation is a basic course in computer engineering discipline. It is the subject which is been taught theoretically and hands-on problems are given to the students to enhance the understanding but it lacks the programming assignment. However the subject is the base for creation of compilers and therefore should be given much more emphasis not just on paper problems but also in simulations. The subjects mostly deals with the tools to recognize languages of Chomsky hierarchy. This hierarchy is divided into four classes: regular language, context free language context sensitive language and unrestricted language. To recognize these languages we have tools such as finite automata, pushdown automata, linear bounded automata and Turing machine. The JFLAP tool provides an interactive visualization for automaton constructs and it works as a simulator for automata constructs like Finite Automata for regular languages, Pushdown automata for context free languages and also for Turing machines.

II. LITERATURE SURVEY

As per the study done by Susan H. Rodger, Eric Wiebe and Kyung Min Lee[1,3], the survey done upon a group of 98 students having theory of computation as their subject and simulating them using JFLAP have helped them to engage themselves into the subject much more than simply solving the problems given as hands-on exercises. Almost 68% of students agreed that having access to JFLAP helped them to understand the concepts easily. Using the tool is easy once you get some experience of it, as per the survey done upon 134 responses 79% of the students found that the tool is easy to use and interactive. The statistics says it all that visualizing the concepts taught in the class helped them to get involved in the subject, so it is recommended to add the use of tools in the academics. Increasing visualization and interaction in the course is enhanced through the popular software tools[6,7]. The proper use of JFLAP and instructions are provided in the manual "JFLAP activities for Formal Languages and Automata" by Peter Linz and Susan Rodger. It includes the theoretical approach and their implementation in JFLAP. [10]

III. ABOUT JFLAP

The software tools like JFLAP are more useful for students to visualize theoretical concepts and also to understand annotations and terminology behind formal languages and automata theory.[9]
The JFLAP tool is an open source and easily available [2], in this paper the version 8 of JFLAP is used to demonstrate the examples. It deals with finite automaton, mealy machine, moore machine, pushdown automaton, Turing machine, grammar, regular expressions and pumping lemma. Once the JFLAP tool is been installed into the system, one gets menu having all the functionality mentioned above.

A. What a JFLAP can do?

The functionalities provided by JFLAP is huge, here we will mention some of the important features used in curriculum. Following is some of the tasks done by the tool.

- It allows to create finite automata and test for acceptance or rejection.
- The string can be parsed by giving the sample input, it can be done either step-by-step or directly to check the result.
- The finite automata can be converted into regular expression, grammar. Also the feature of combining two or more automata and minimizing is possible.
- It also helps to check whether the given FA is non-deterministic or deterministic, if non-deterministic the transition with non-determinism can be highlighted.
- One can check the multiple input string at the same time for the automata given.

- It can generate the possible strings accepted by the automaton given.
- Can check the type of grammar defined.
- It can generate the derivation steps and derivation tree of the given input string for the grammar.
- It can convert the grammar into PDA.
- Can build LL(1) parse table, SLR(1) parse table (both used while compiler design).
- It can allow to build Turing machine and can convert it into the grammar.
- One can check whether the input string(s) accepted by the given Turing machine or not.

The functionalities are not just restricted to the list given above, but these features are enough to simulate the concepts of automata[8].

IV. USE OF JFLAP TO UNDERSTAND THE WORKING OF AUTOMATA

Formally, finite automaton is defined as 5-tuple[4,5], given as (Q, ∑ , δ, q0, F) where,
Q: finite set of states,
∑: finite set of alphabet
δ: transition function
q0:is a start state and is element of Q
F: final states, it is the subset of Q
The Figure 1 shows the FA that accepts all the strings ending with "aba", it is a simple FA and now with the help of JFLAP it can be simulated to do further task.

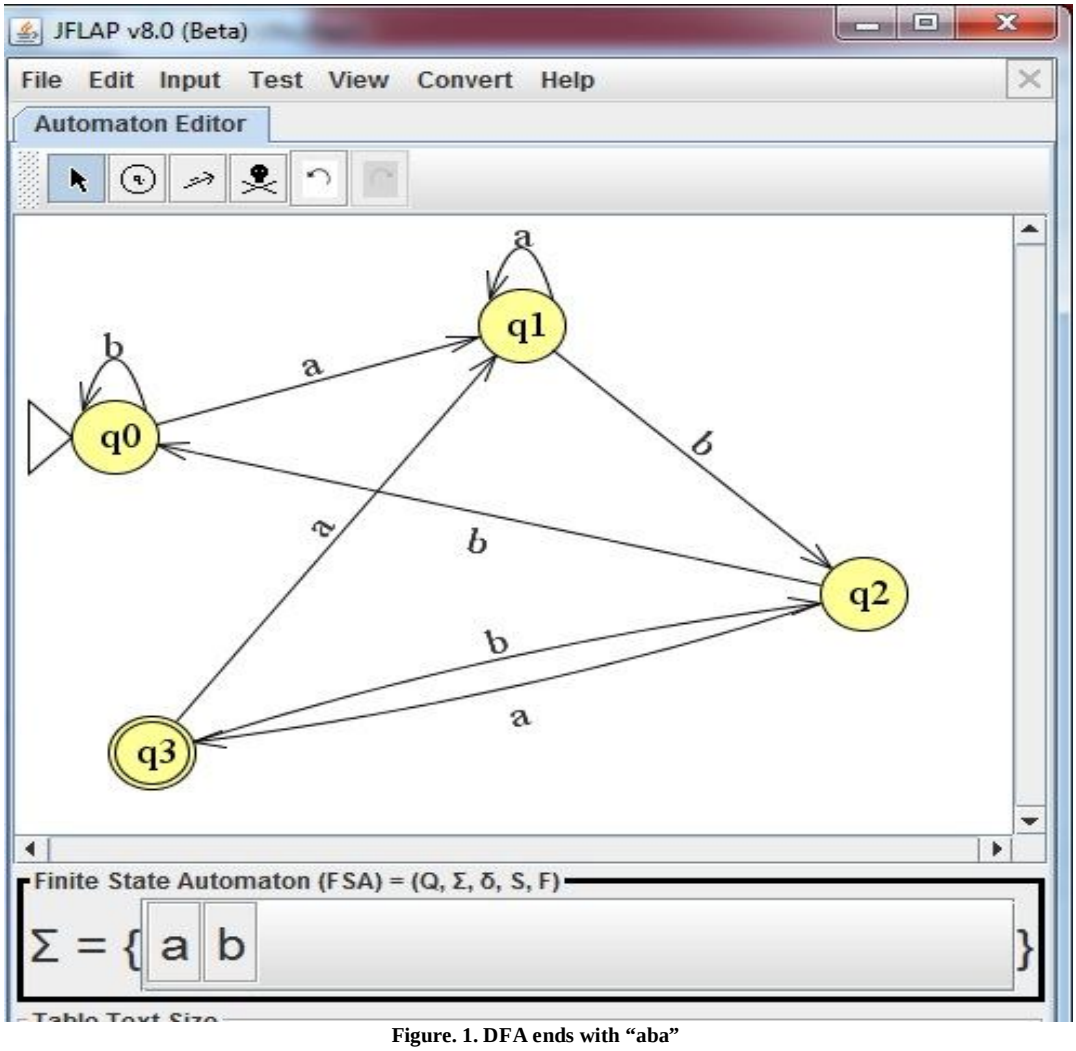

Figure. 1. DFA ends with "aba"

The options are given in the toolbar to create the automaton, one can also decide the initial state and final state on the basis of the problem given. Now with the help of input button one can check whether the input string given can be accepted or rejected. There are multiple options for the input, one can check the input step by step, by fast run or multiple run. The window will look like Figure 2 given here.

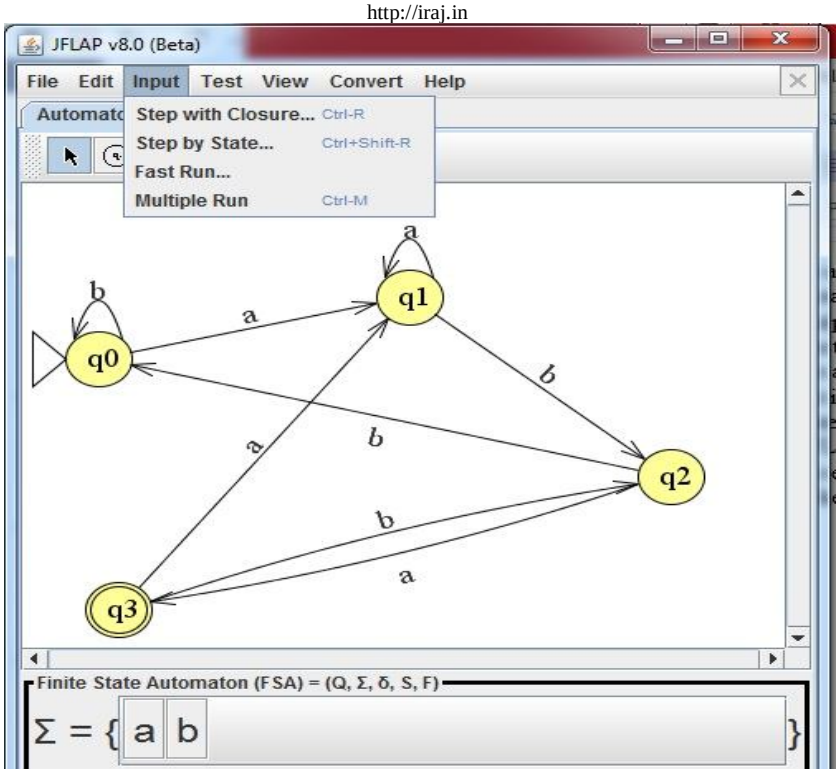

Figure. 2. Options to run the input for DFA given.

Let's check the string "bbaba", it can be observed that string will be loaded with the current state highlighted and can be traced step-by-step. Figure 3 shows the start state will be active as the first symbol becomes ready to read, and step-by-step the symbol is been read from the string and the corresponding active state will be highlighted. Figure 4 shows that for reading the symbol "b", the automata has state "q1" to validate it. Once the string gets exhausted and one reach to the final state after traversing, then it can be said that string is accepted else it will be rejected.

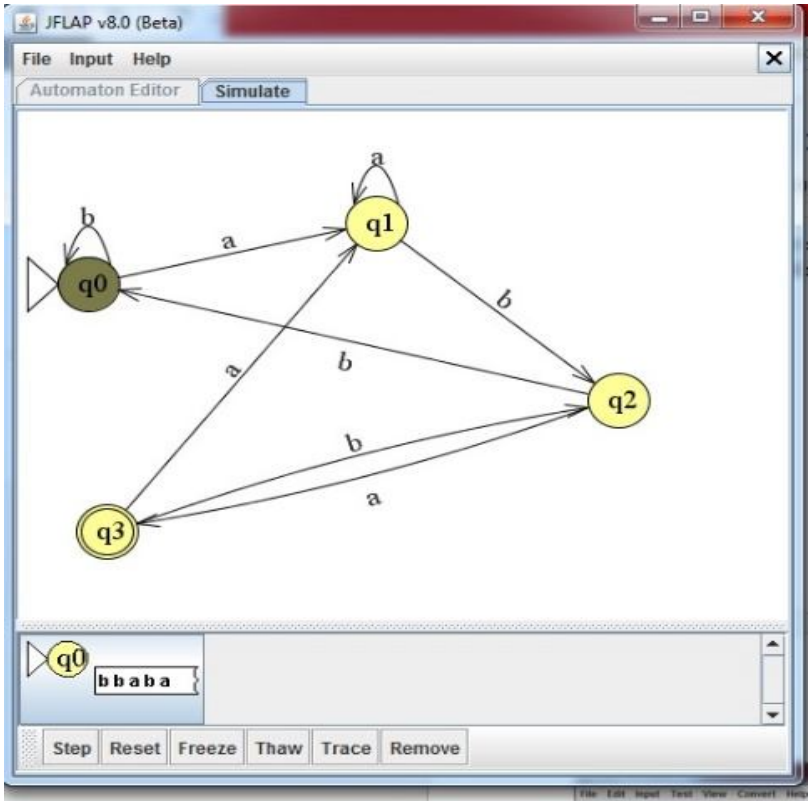

Figure. 3. The string "bbab" is loaded as input

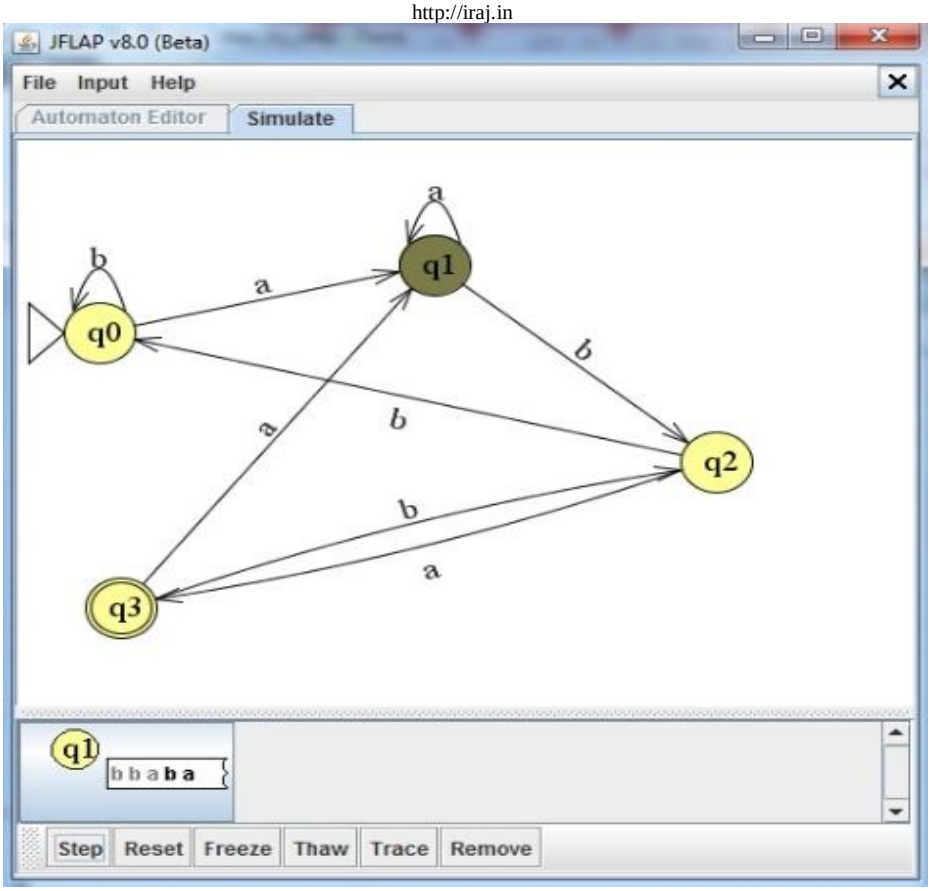

Figure 4. The active state for the corresponding symbol

Figure 5 and Figure 6 shows the multiple inputs and their corresponding results.

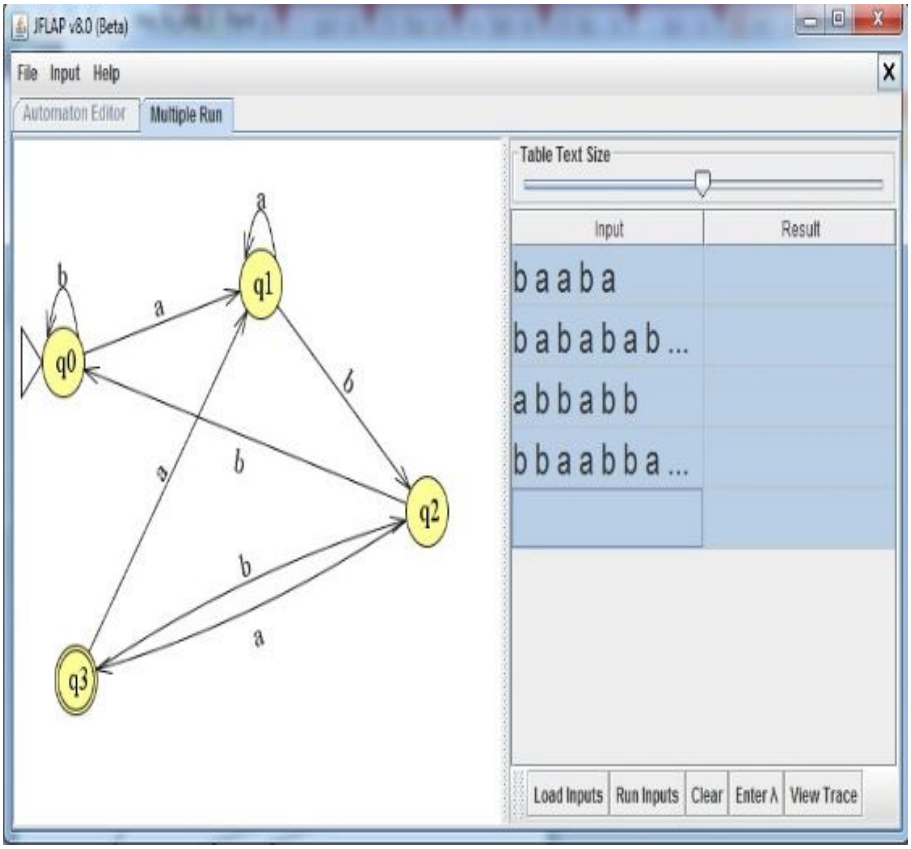

Figure. 5. Checking multiple inputs on the automaton

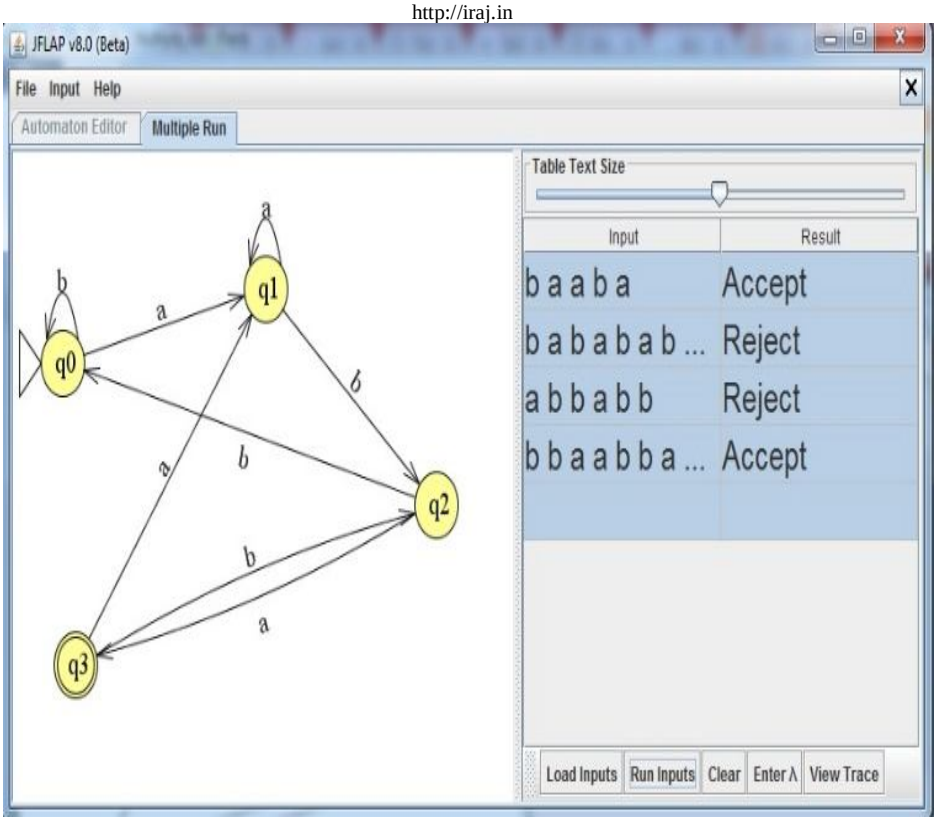

Figure 6. Shows the result of the inputs given

If the automata has to be converted into a grammar, then it just require a key press called convert.

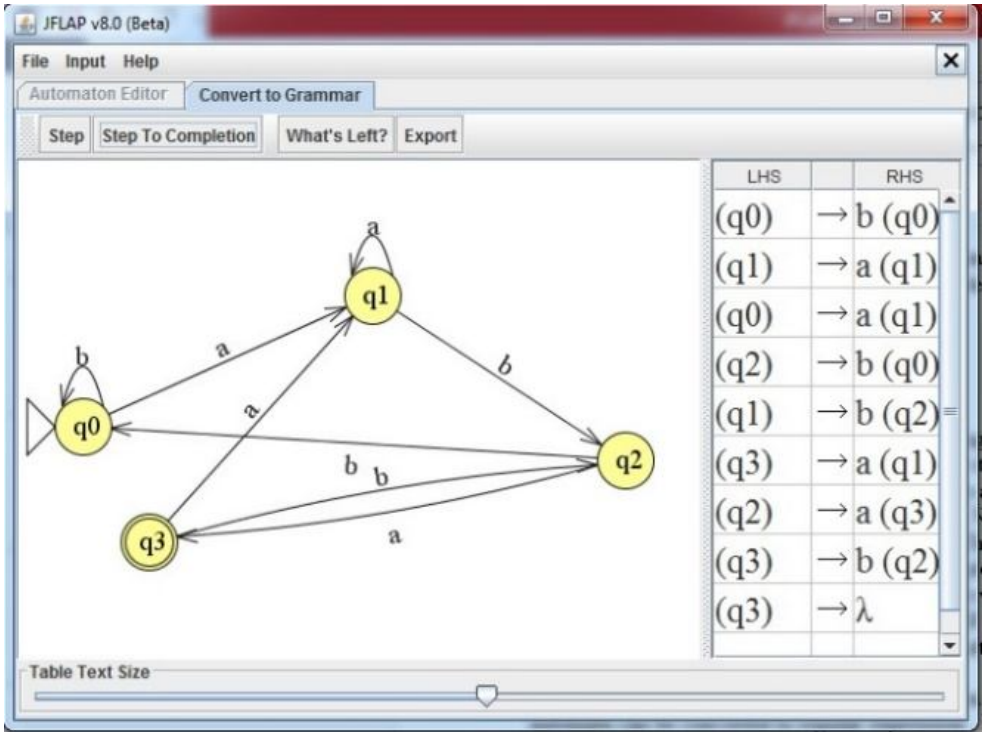

Figure 6. Corresponding grammar for the automaton

It is easy to work with grammar as well, a context free grammar is formally given as 4-tuple[4,5]
(V, T, P, S), where
V: finite set of variables
T: set of terminals
P: set of productions
S: starting variable

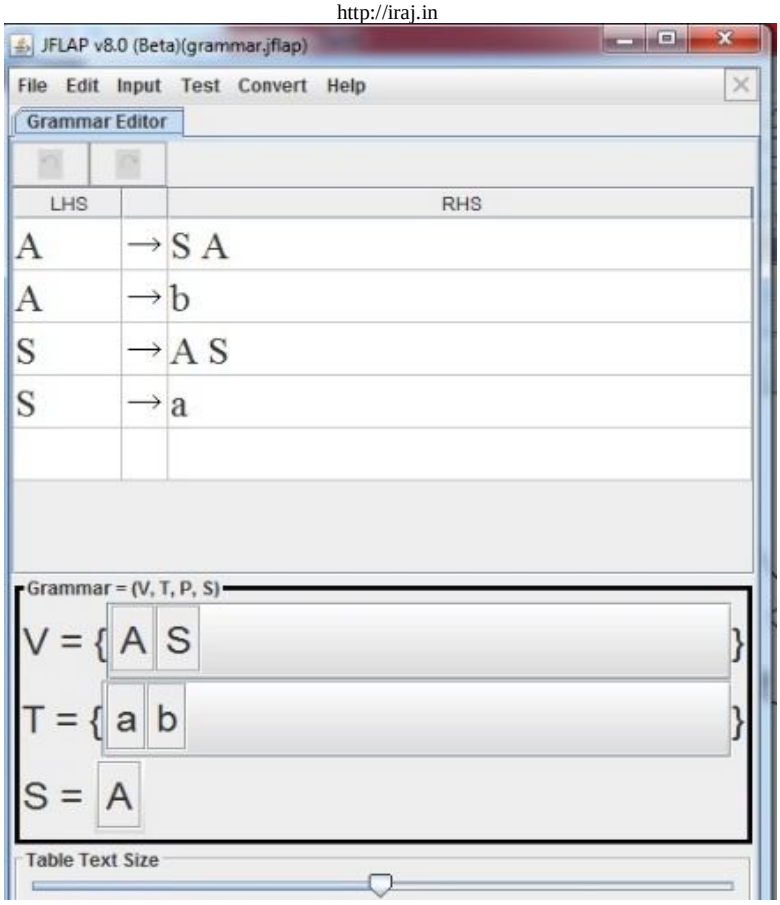

Figure 7. Grammar example in JFLAP

The tasks that can be done with the grammar is to see the derivation steps and derivation tree corresponding to the grammar given.

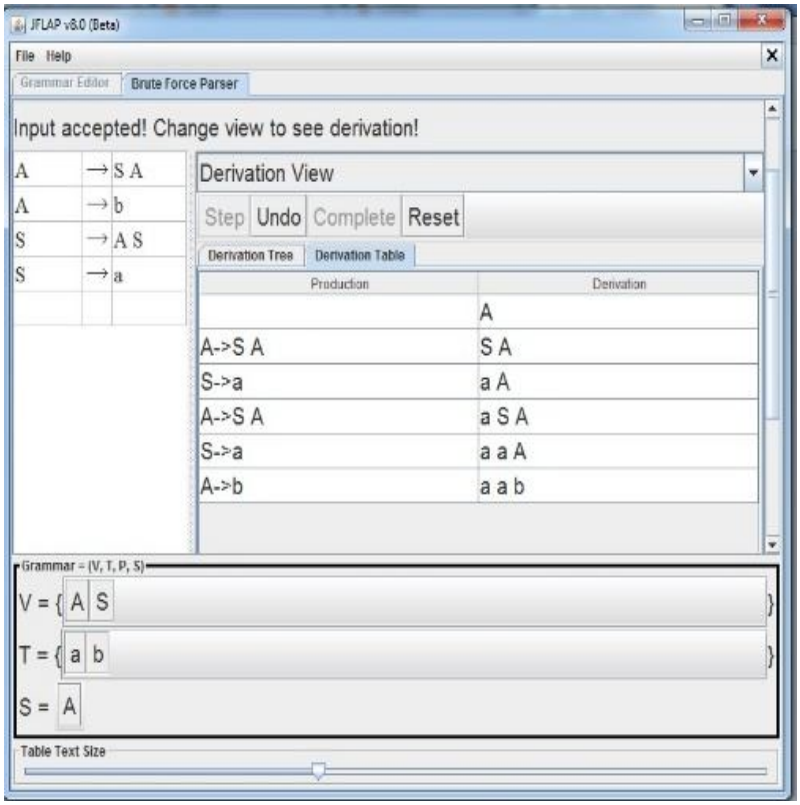

Figure 8. Derivation step to accept the string " aab"

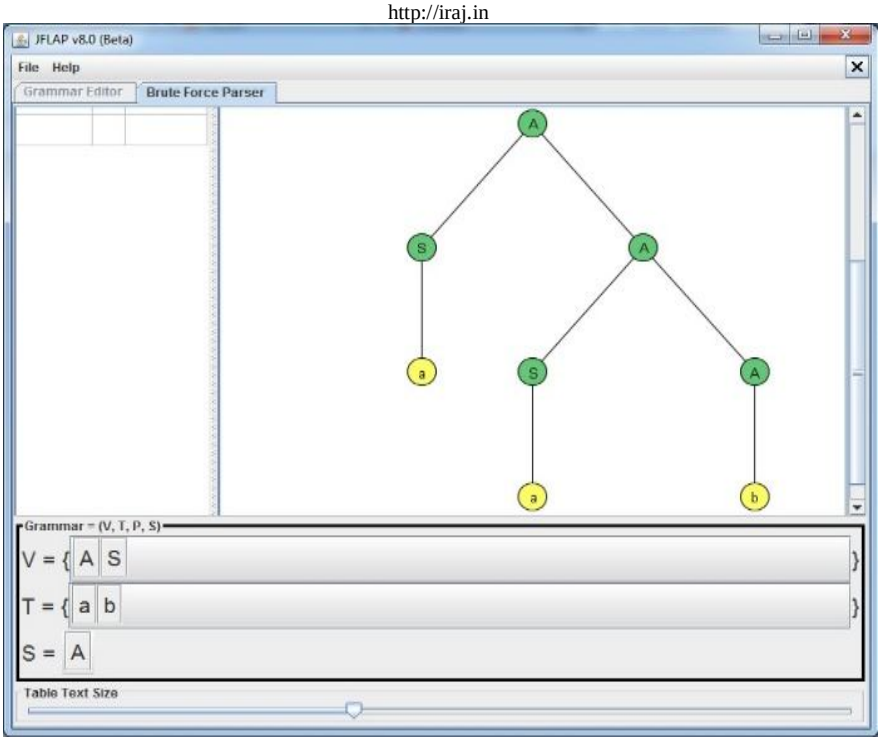

Figure 9. Derivation tree corresponding to string"aab"

To check the type of strings generated by the grammar one can click on the input and then generate language, this will give you the option of number of strings that you want to see.

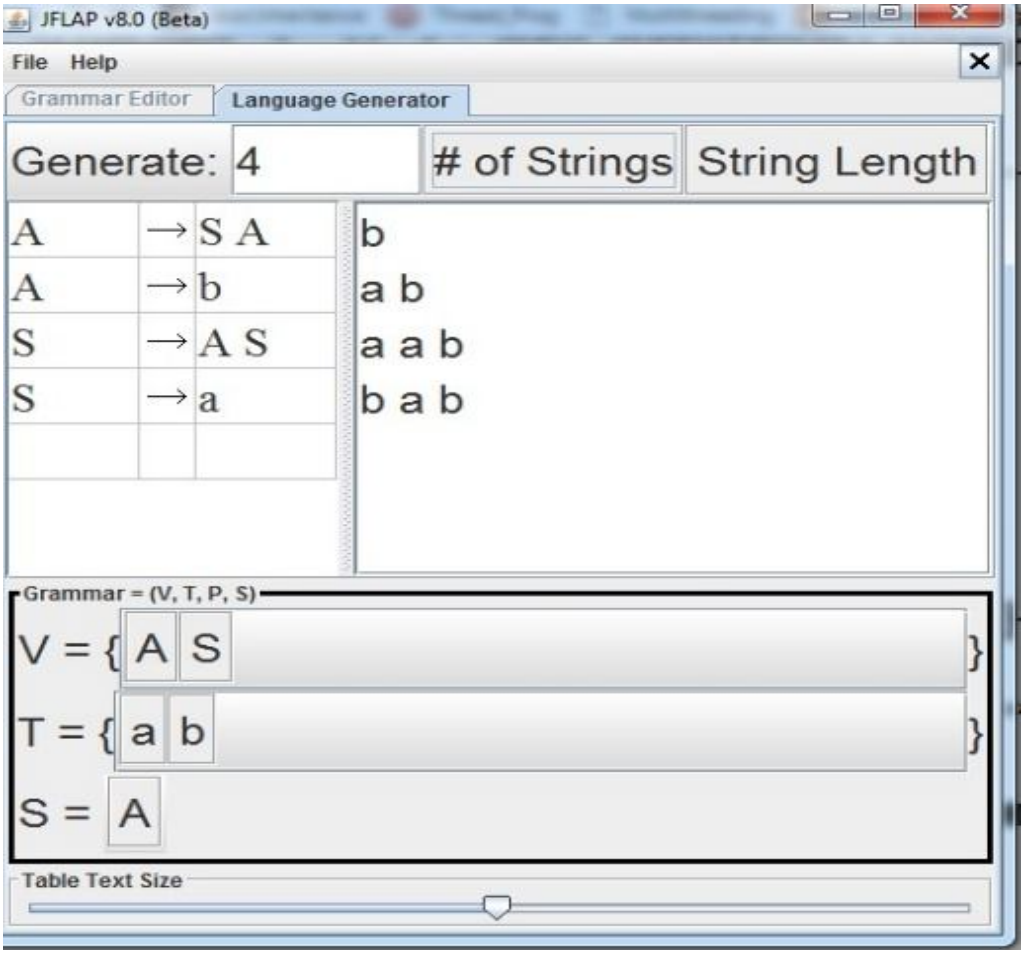

Figure 10. Strings generated corresponding to the grammar

One of the difficult task is to determine the type of grammar. And this can be done by simply feeding the grammar to the JFLAP tool and pressing the check button, it will give the type of grammar in a pop-up window.

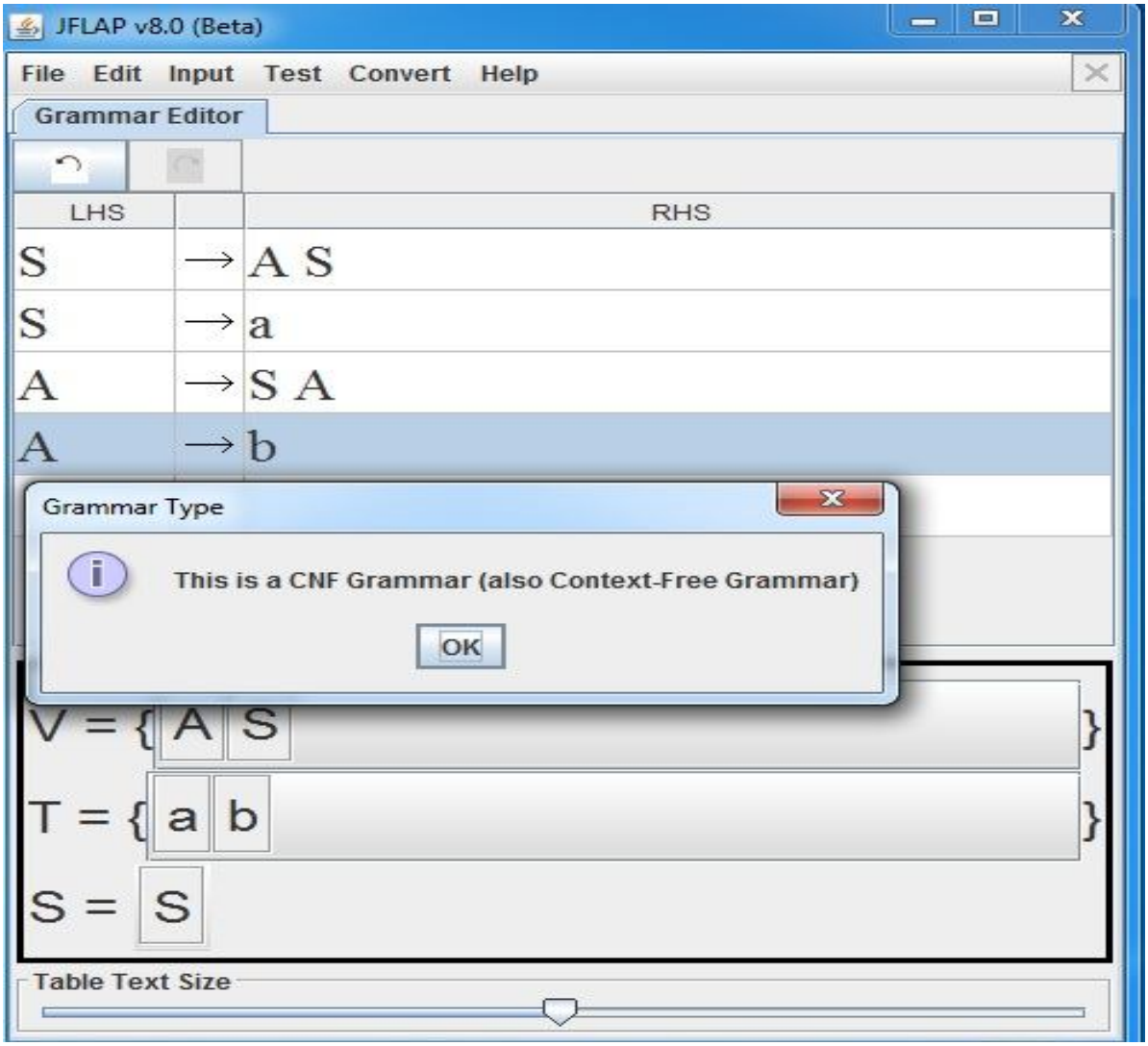

Figure 11. The pop-up window shows that grammar is CNF

The last concept to be discussed here is about Turing Machine, formally deterministic Turing Machine is a 7-tuple[4,5]:

M = (Q, Σ, Γ ,δ, q, qa, qr), where
Q: is the finite set of states
Σ: is the finite set of input alphabet
Γ: is a finite set of tape symbol
δ: is a transition function
q: is the start state, one of the elements of Q
qa: is called the accepting state , element of Q
qr: is called the rejecting state, element of Q

The figure given below demonstrate the Turing Machine accepting the language $a^nb^nc^n$, where n>0.

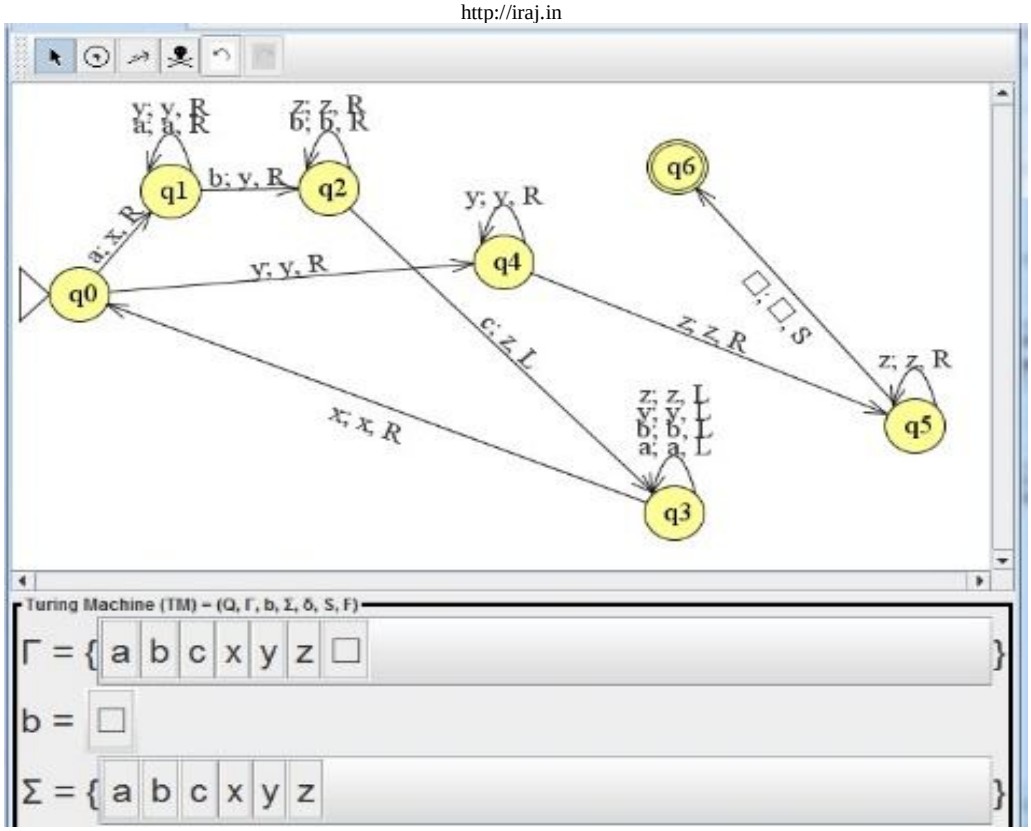

Figure 12. Turing Machine in JFLAP

Just like finite automaton, the Turing machine can be checked for different input string in either step-by-step style, fast run or multiple run.

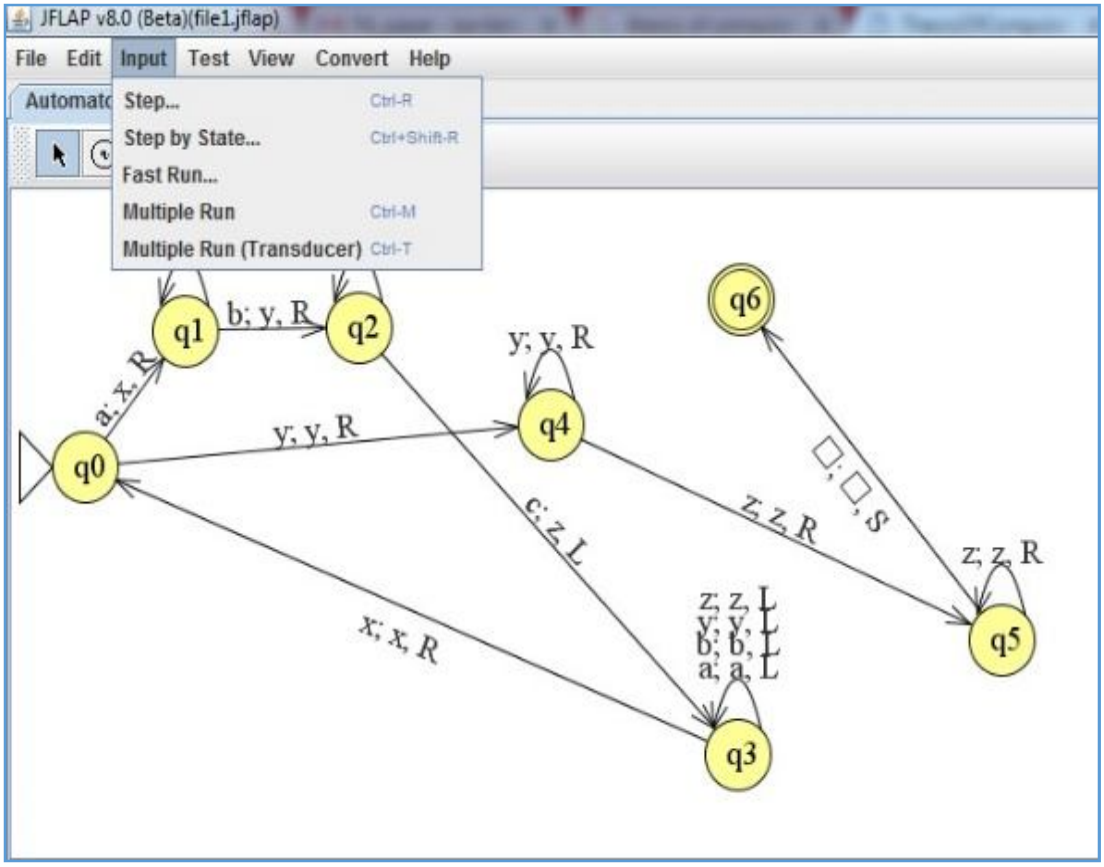

Figure 13. Options for taking inputs in Turing Machine

CONCLUSION

The aim of this paper is to include JFLAP tool into academics to understand the concept, however one has to be trained to have the basic knowledge of automata theory. To make it better the work has to be done to create the automata as per the conditions given. This will help the students to validate the solutions and help them to visualize what they study in the classroom.

REFERENCES

[1] S. H. Rodger, E. Wiebe, K. M. Lee, C. Morgan, K. Omar and J. Su, "Increasing engagement in automata theory with JFLAP", ACM Transactions, (2009)pp403-407.
[2] http://www.jflap.org/jflaptmp/
[3] S. Rodger, Integrating Hands-OnWork into the Formal Languages Course via Tools and Programming, FirstInternational Workshop on Implementing Automata, London, Ontario, 1996.
[4] Sipser M. Introduction to the Theory of Computation. Course Technology. 2005.
[5] Hopcroft J. E., Motwani R., Ullman J. D. Introduction to automata Theory, Languages, and Computation Addison Wesley; 2007
[6] Susan Rodger, A. Bilska, K. Leider, et al. "A collection of tools for making automata theory and formal languages comealive," SIGCSE '97 CA, USA, 1997.
[7] Daniela Chuda, "Visualization in Education of Theoretical Computer Science," International Conference onComputer Systems and Technologies - CompSysTech'07.
[8] Jonathan Jarvis, Joan M. Lucas, "Incorporating Transformations into JFLAP for Enhanced Understanding of Automata," SIGCSE'08, March 12–15, 2008, Portland, Oregon, USA.
[9] Enhancing the learning ability using JFLAP for Theory of Computation Course, Journal of Engineering EducationTransformations, Special Issue
[10] Peter Linz and Susan Rodger, "JFLAP activities for Formal Languages and Automata" manual, DukeUniversity, Februaury 2011.

★★★