ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

PROCESSING OF STRING THROUGH VARIOUS LANGUAGE IN AUTOMATA

Sheela*, Dr. Surender Jangra
* M.Tech Scholar,Computer Science and Engineering,Haryana College of Technology & Management,Kaithal.
Assoc. Professor,Computer Science Engineering,Haryana College of Technolsogy & Management, Kaithal.

ABSTRACT

Language plays an important aspect in the life of all beings. Language can be defined as verbal, physical, biologically innate, and a basic form of communication. Language helps to fulfill our wants and needs. Language is the ability to acquire and use complex systems of communication. Human beings can communicate with each other. We are able to exchange knowledge, beliefs, opinions, wishes, threats, commands, thanks, promises, declarations, feelings But when we share any sensitive information over the internet their must be some security threats from intruders .We can protect our information through security feature on internet. In this paper, we are introducing some string processing techniques in automata, which protect our data from intruders.

KEYWORDS: Simulator, DFA ,NDFA ,Context free Grammar ,Turing machine.

INTRODUCTION

As we know that the communication is the important aspect of all human beings. But when we communicate over the internet we have to protect our data from intruders. Here we are introducing Automata techniques to protect our data over internet. String in automata can be processed by following ways:
1)Regular Grammar
2)Context Free Grammar
3)Recursive enumerable Grammar.

In Regular language we can process the string through Deterministic Finite Automata(DFA) & Nondeterministic Finite Automata(NDFA)[15]. For implement the string in context free language we used.1.Context free grammar 2.Derivation Tree/Parse Tree For Implementation the string in Recursive Enumerable Language we used Intentions Description.

Recursively Enumerable Language (RELs) are accepted by the type of automata as Turing Machine. Turing Machines are the most powerful computational machines and are the theoretical basis for modern computers. Turing Machine works for all classes of languages including regular language, Context Free Languages as well as Recursive Enumerable Languages.[14]

REVIEW OF LITERATURE

Fang Yu, Muath Alkhalaf, Tevfik Bultan and Oscar H. Ibarra In this paper, I studied an automata-based approach for symbolic analysis of string manipulation programs.I studied the STRANGER,an automata-based string analysis tool for detecting string-related security vulnerabilities.

Abdulbaki Aydin, Muath Alkhalaf, and Tevfik Bultan In this paper, I studied an automated testing framework that targets testing of input validation in web applications for discovering vulnerabilities. In this paper,they show that vulnerability signatures computed for deliberately insecure web applications (developed for demonstrating different types of vulnerabilities) can be used to generate test cases for other applications.

Aakanksha Pandey , NilayKhare In this paper they present the area vs speed tradeoff of the Deterministic Finite Automata (DFA) and the delayed input Deterministic Finite automata(D2FA).They will see how area occupied by the state transition table get decreased by eliminating redundant transition in D2FAalong with how the time taken by the input pattern for processing get increased.

Len Sassaman, Meredith L. Patterson, Sergey Bratus, and Michael E. Locasto They present a formal language theory approach to improving the security roles of protocol design and message based interactions. In this paper, they argue that to finally get it right security wise, they need a new stronger computational-theoretic understanding of message-based interactions between components. They use formal language complexity arguments to explain why certain protocol and message format design decisions are empirically known to be wellsprings of vulnerabilities, and why the respective software components do not seem to yield to concerted industry efforts to secure them.

SIMULATORS TO BE USED TO PROCESS STRING IN AUTOMATA

Automaton Simulator:-Automaton Simulator is open-source software, under the terms specified below:-

Automaton Simulator is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version Drawing an automaton:-Start with a blank, new automaton by choosing from the new submenu of the File menu. Automaton Simulator will allow you to create following types of machines: a deterministic finite automaton, a non-deterministic finite automaton.

Select the state tool, pictured in the toolbar as a large red circle. Place the states on the canvas where you think they should go. Designate initial and final states by right-clicking (or command-clicking) the states. Draw transitions between states using the transition tool, pictured in the tool bar as a line connecting two small red circles. Designate to which letters the transitions apply by right-clicking (or command-clicking) the transitions.

To simulate the automaton on a string, click the simulate button (pictured as a green right-pointing triangle). Now type the characters of the string. As you type, the current state of the automaton will update.

Automaton Simulator tools

Using the Automaton Simulator toolbar involves three tools. Click on a tool button to select it; the program indicates your current selection by giving the button a dark background.
The four buttons, going left to right in the toolbar, are the following:

1. The state tool, for adding states to the automaton.
2. The transition tool, for adding transitions to the automaton.
3. The text tool, for adding text labels to the automaton.
4. The simulation buttons, for simulating the automaton on a string.

For an automaton, the toolbar contains a single Play button, represented by a green triangle in the toolbar. When you click the Play button, the automaton resets its current state to the initial state, displayed in green. As you type letters, the highlighted states will update to reflect the state transitions of the automaton, and the tape at the window's bottom will log the letters you type. If you want to simulate a different string entirely, click the Play button again. The tape will clear, and the automaton will reset to its initial state.

The backspace key will work as an ``undo:'' it restores the machine to its state before the last character typed. You can go back several steps.

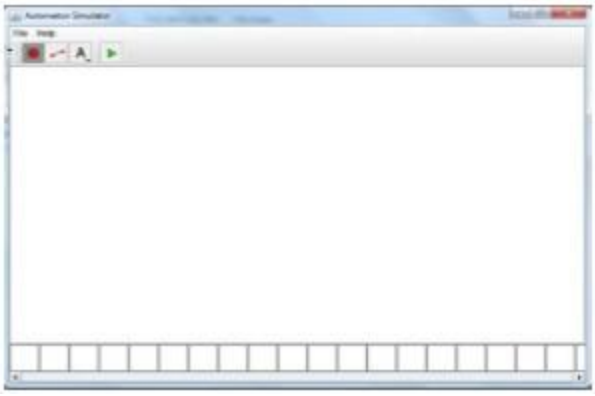


Fig:- Automaton Simulator.

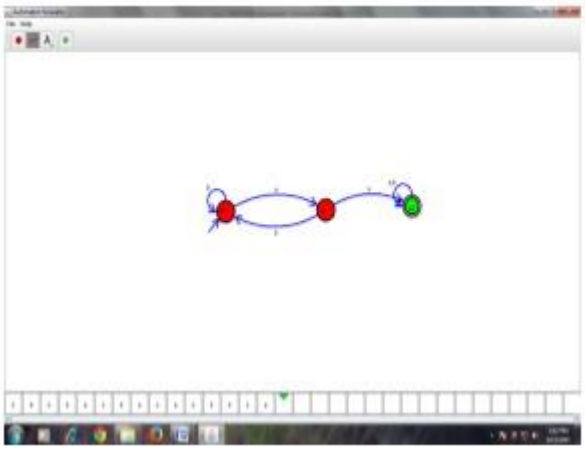
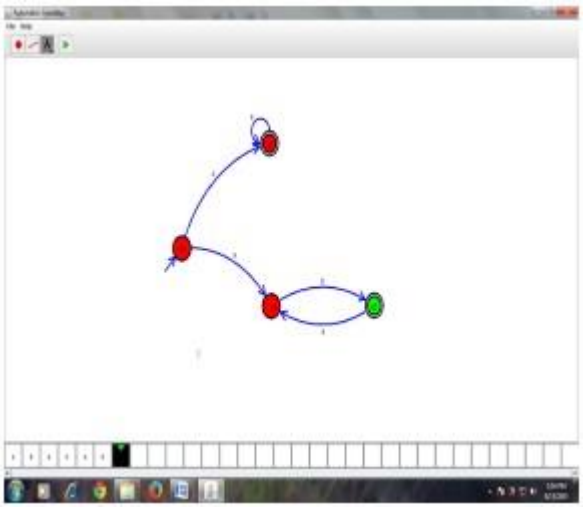


fig. Simulation of string of language containing aa as substring.

For the nondeterministic finite automata we implement the string represented by the language L=(a)* + (ab)*



Grammar Editor Simulator:

Grammar Editor should run on any platform supporting Java, version 1.4 or later.

1. Grammar Editor requires Java 1.4 or later. If you do not already have it on your computer, Java is available from java.sun.com.

2. Download grammar.jar (110 kilobytes).
3. On Windows systems, you might be able to start Grammar Editor by simply double-clicking on the grammar.jar file. On Unix-base systems, type ``java -jar grammar.jar''. You'll also have to do that on Windows systems if the double-clicking doesn't work.

Grammar Editor allows you to compose and use a context-free grammar. Write your grammar in the ``Grammar:'' text area. For example, you might type the following into the blank.
S → aSa | bSb | aa | bb

Each symbol (non terminals) should be in all capital letters; atoms (terminals) can be any combination of letters and digits, as long as they're not all capital letters. The symbol on the first rule's left side is assumed to be the root symbol in the grammar. At the bottom of the window are three buttons.

- The *Parse button* parses the contents of the ``Text:'' field based on the current grammar. It displays the resulting parse tree in another window. The parser uses the CYK parsing algorithm. (Unlike many popular parsing algorithms, such as LR(1) parsers, the CYK algorithm can work with any grammar.)
- The *Generate button* computes a randomly generated sentence from the language described by the current grammar, and it places this sentence into the ``Text:'' field.
- The *Clear button* clears the ``Text:'' field.

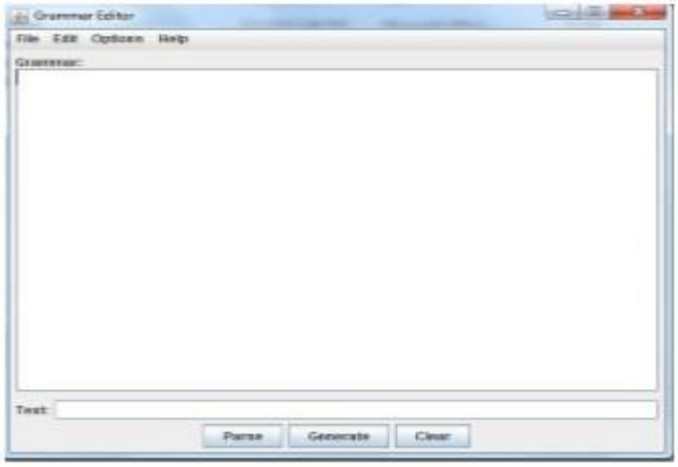


Fig. Simulation of CFG with grammar editor.

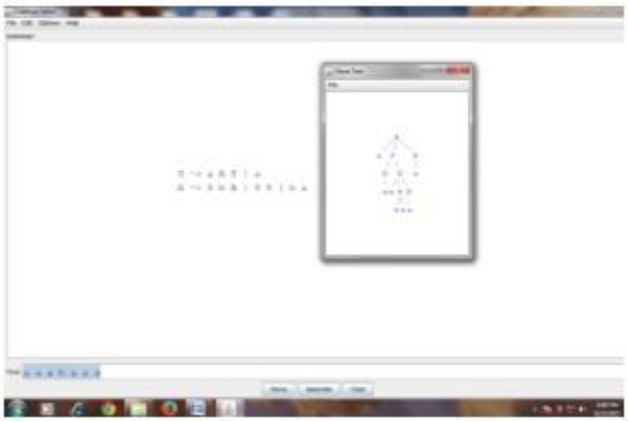


Fig: Implementation of string aaaaabaaaa with grammar editor.

Online Turing machine simulator

How a Turing Machine processes the input It depends on a set of transitions, on the input word, and on the initial state. Once the processing finishes, to know if at the end of an execution the machine accepted the input, we must specify a set of accepting states you must provide the transitions, the initial state and the accepting states. Before explaining how to define these three objects, we will show how to use the controls of the machine.Once your machine is programmed, you must click the green COMPILE button, If your code has any errors, the compiler will let you know one by one the lines you must change.Turing machines are intended to process inputs. Hence, before running your machine you must provide an input word in the bottom left corner of the machine panel and click the Load button. Now you are ready to start using the play, pause, stop and step buttons to run the machine.

Fig. On line Turing Machine simulator with string accepting state.

CONCLUSION

After implementing the string with different simulators a point comes is which one is better? Although every language have its particular advantage and disadvantage.

So if we talk about Regular Language We can simply process the string and the result is accepted or not and we can't store the string, we can moves in forward direction.

When we comes to Context free language it can recognize the string ,match the string, compare the string and store the string, moves to forward direction.

And in Recursive Enumerable language it can recognize the string ,match the string, compare the string and store the string. And more over it has a powerful capability to choose the direction for moving the string.

So by the above implementation we can say that the Recursive Enumerable language is the powerful and most usable language among the language.

REFERENCES

[1] James P. Schmeiser, David T. Barnard (1995). Producing a top-down parse order with bottom-up parsing. Elsevier North-Holland.
[2] D'Souza, D. and Shankar, P. (2012). Modern Applications of Automata Theory. World Scientific Publishing, Singapore.
[3] Danish Ather, Raghuraj Singh, Vinodani Katiyar. An Efficient Algorithm to Design DFA That Accept Strings Over Input Symbol a, b Having Atmost x Num-ber of a & y Number of b. JNIS World Science Publishers, U.S.A. Feb 2013.
[4] Lawson, Mark V. (2004). Finite automata. Chapman and Hall/CRC. ISBN 1-58488-255-7. Zbl 1086.68074. McCulloch, W. S.; Pitts, E. (1943). "A logical calculus of the ideas imminent in nervous activity". Bulletin of Ma-thematical Biophysics: 541–544.
[5] JFAP tool for simulating results and validation. [Online]. Available: http://www.jflap.org Deterministic finite automaton [Online] Available at http://www.wikipedia.org/wiki/Deterministic_finite_automaton.
[6] F. Yu, M. Alkhalaf, and T. Bultan. Stranger: An automata-based string analysis tool for php. In TACAS, pages 154{157, 2010.
[7] .R. Sommer and V. Paxson, "Outside the closed world: On using machine learning fornet work intrusion detection," in Proc. IEEE Symp. SecurityPrivacy, May 2010, pp. 305–316.
[8] J.Hopcroft, R.Motwaniand J.Ullmann ,Introduction to AutomataTheory,LanguageandComputation (3rd Edition, Addison-Wesley, 2006).
[9] M Handley, V. Paxson, and C. Kreibich, "Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics," in Proc. 10th USENIX Security Symp., 2001, p. 9.
[10] Michael Sipser (1997)Introduction to the Theory of Computation. PWS Publishing. ISBN 0-534-94728-X. Part One: Automata and Languages, chapters 1–2, pp. 29–122. Section 4.1: Decidable Languages, pp. 152–159. Section 5.1: Undecidable Problems from Language Theory, pp. 172–183.
[11] Daniel I. I. Cohen. Introduction to Computer Theory, 2nd ed. Wiley, 1997.
[12] A. BabuKaruppiah, Dr. S Rajaram "Deterministic Finite Automata for Pattern Matching in FPGA for intrusion Detection" in International Conference on Computer and Electrical Technoogy-ICCCET 2011,18th& 19th March,2011."
[13] Sipser, M., Introduction to the Theory of Computation, Brooks/Cole, Thomson Learning, London. 2003.
[14] Jainendra Singh, Dr. S.K. Saxena, " Implementation of Unrestricted Grammar in to the Recursively Enumerable Language using Turing Machine", The International Journal of Engineering And Science (IJES), ISSN 2319-1813 ISBN 2319-1805, Volume-2, Issue-3 (2013),pages 56-59.
[15] Adesh K. Pandey,Theory of Computation S.K.Kataria & Sons Publication,New Delhi 2nd Edition 2011.