International Journal of Hybrid Information Technology
Vol.8, No.1 (2015), pp.193-202
http://dx.doi.org/10.14257/ijhit.2015.8.1.17

Enhancement of Turing Machine to Universal Turing Machine to Halt for Recursive Enumerable Language and its JFLAP Simulation

Tribikram Pradhan

Department of Information and Communication technology
Manipal Institute of Technology (MIT)
Manipal University, Manipal - 576014, Karnataka, India
tribikram.pradhan@manipal.edu

Abstract

Turing Machines are the most powerful computational machines. Turing machines are similar to algorithms, and are the theoretical basis for real computers. It is a tiresome task to build and maintain Turing Machines for all the problems. The Solution of this problem is The Universal Turing Machine (UTM) or simply a universal machine. The Formation of UTM is very difficult due to the fundamental complexities. There are so many existing tool those does not supporting formation of Universal Turing Machine that's why it is very difficult to accomplish the task. So that JFLAP platform is used for the creation of UTM. JFLAP is wrapping up of graphical tools that are used as an assist in learning basics of formal languages and automata theory. It is software to perform experiment using formal languages topics covered Nondeterministic finite automata, L-systems, parsing, multi-tape Turing machines, different kind of grammars and Nondeterministic Pushdown automata. As well as testing and making another example for these, JFLAP provide one to research with construction proofs from one form to another, such as converting an NFA to a DFA to a minimal state DFA to a regular expression or regular grammar. A Directed graph is be used to represent a Turing Machine, uses by JFLAP. It is widely used to simulate UTM as transducer of the Turing machine and it consists of multiple inputs.

Keywords: *Turing Machine, FSM, UTM, CFG, PDA, JFLAP, Recursive Enumerable grammar*

1. Introduction

The language which has proper alphabet, grammar and model to recognise is called formal language. The Collection of strings where we can put some restriction condition in the formation of string is called formal language. Mathematical representation of formal language is called automata, automata is a recognising devise. Mathematical representation of recursive enumerated language is called Turing machine which is having finite automata with memory and read write capability. Turing Machine can recognise the data through the tape symbols. The language which is accepted by Turing machine is called recursive enumerated language. Turing machine recognisable languages are two types such as recursive set, recursive enumerated. Turing machine has three component Infinite tapes, Tape Header, Finite control. The Infinite tape is a two way and it is divided into multiple cells. Each cell is capable of having only one input symbol. The tape header reads the data from infinite tape starting from left most end of the string.

The movement of tape header is bidirectional either from left to right or from right to left. Tape Header scans the symbols from the infinite tape and overwrites the symbols on the infinite tape. Finite control manages the transitions of Turing machine.Turing machines are

ISSN: 1738-9968 IJHIT

similar to algorithms, and are considered as an abstract model for real computers systems. A Turing machine which can be used replicate any other Turing machine is called a Universal Turing machine (UTM, or simply a universal machine). A UTM is the theoretical model for all computational models. A UTM is an automaton which has a set of input that can be used as the description of any Turing Machine TM and a string w, can reproduce the computation of M on w. JFLAP considered a Turing Machine as a directed graph. JFLAP is widely used in construct UTMs as the Turing machine transducers and having several inputs. Complex Turing machines can also be built by using other Turing machines we are able to create a complex Turing Machine.

2. Universal Turing Machine in JFLAP

Designing and Simulating Universal Turing Machine using JFLAP, that could simulate maximum number of halting problems (string accepted by Turing Machine) belongs to computer science domain and Developing a Universal Turing Machine for recursively enumerable languages. Turing Machine is an abstract model of real computer system and it is so dominant computational machine. The halting problem can be solved by Turing machine (TM) and it is very difficult to create and maintain TMs for all the problems. This Problem can be solved by Universal Turing machine. A UTM behaves like a single Turing Machine but provides a solution for all computational Problems.

2.1. Universal Turing Machine

A UTM that is capable of simulates all other TM, which provides a solution for all computational problem. It is an automation having input description of other Turing Machine (TM) and string w. also simulate the computation of M on w. That's why it minimizes space complexity as compared to a number of Turing machine. The Transition function is the central component of UTM. The UTM works on the following Rules.

$$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\} \quad (1)$$

The transition function δ is a partial function on Q X Γ and Turing Machine operates based on working of this Transition function δ.

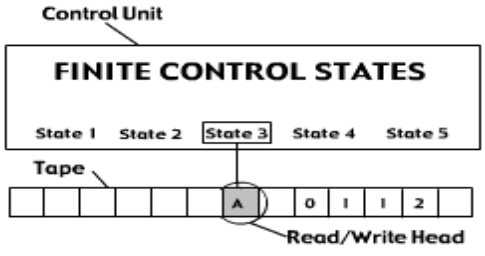

Figure 1. Finite Control States Of Turing Machine

The arguments of δ are the current state of the control unit and the current tape symbol being scanned. The result is a new state of the control unit, a new tape symbol which replaces the old one and a move symbol L or R.

Turing machine is an automaton whose temporary storage is a tape. This tape is divided into cells, each of which is capable of holding one symbol. Associated with tape is a READ-WRITE head that can travel right or left and can read –write a single symbol in each move.

It is defined by M= (Q, Σ, B, q0, F, δ, Ί)

Where,

Q =set of internal state,
Σ=input alphabet,
B=special symbol called blank,
Ί=finite set of tape alphabets,
δ=transition function,
F=final state,
q0=initial state.

And transition function according to which Turing Machine performs

Its operation is defined as:

$$\delta \; X \; Ί \rightarrow QX \; Ί \; X \; \{L,R\} \qquad (2)$$

2.2. Working of the UTM

The UTM is consisting of an infinite tape having multiple cells and each cell is capable of only one input symbol at a time. The blanks cells are filled with blank symbols B. At any point of time infinite tape contains only one input string. The movement of tape header is bidirectional from left to right or from right to left. Hence Turing machine works as both reader and writer. Tape header scans the symbols from the infinite tape and overwrites the symbols on the infinite tape. It also consist of two other tape which are used for processing. The description of original Turing machine can be present in the first tape and the internal state can present in the second tape of TM [3]. <TM, w> is used to give the input to the UTM TU. The input string w can be manipulated by this. The transition rule or delta rule is used for the transition. This is given below

$$\delta (qi , a) = (qj , b, R)$$

qi is the current state,
a is the current read symbol,
qj is the next state or destination state, b
b is the write symbol and
R is the direction to which the tape head has to move.

The Tape header of Turing machine and universal Turing machine is bidirectional. The movement of tape header is bidirectional from left to right or from right to left. Left represented by L and Right represented by R. One option presents to stay in the current position which is represented by S. The Encoded input are fed to the UTM. The Tape header scans symbol from infinite tape and reads current input symbol and internal state. It follow the transition rule which is present in the description tape complete the operation as per the transition rule of the original Turing machine. After reading all input symbol the TU enter the final state and check and perform the same as Turing machine.

2.3. Simulation of Turing Machines in the UTM

The universal Turing Machine, string is required as input to Turing Machine and the converted string of some random Turing Machine is given as input string. Operation performed by UTM is same as TU. When performing the emulation of TM, TU will require to keep trail of several things. TU maintains the current input symbol of the memory tape of Turing machine, the present state of TM, and the present location of Turing machine's read-write head.

2.4. Creation of input for the UTM

The B symbol is used to represent a blank symbol on the tape of TM. The infinite tape consists of symbol B at both start and end of the input string. The input is determined with three sections: (1) one final states of TM, (2) the transitions of TM, (3) the tape contents of TM just prior to the start of execution.

</Final State List/> : </Delta Rule/>[;</Delta Rule/>]*: B</Initial State ID/></Input String w/>B

The order of transition rules of TM are written in reversed format from that of the standard Turing Machine quintuple. From left-to-right, the five characters are: the read-head shift direction, the write character, the destination state, the read character, and the source state. The initial configuration of the tape is represented by the final section of execution.

2.5. Manipulation of Turing Machine in the UTM

TU keeps the present state-id-number which respective to the present state of Turing machine in cell to the left where the read-write head of TM is presently aligned. So it is uses only a single memory cell, TU is capable of keep trail of both the present state and the present read-write head place of the Turin machine that is being pretend. One step of TM will represent to multiple steps in TU, since the place of the present- state id digit might need to be changed with its adjacent to run the transition rules. The Part of memory which is hold two lists i.e. of final-states and list-of-transitions, is cannot be changed by TU. The contents of tape of TU will develop with time exactly in conformance with the simulated execution of Turing machine in the last division. As TU continues the execution of TM, In This portion of TU's tape duplicates the changing state of TM's tape that contains configuration of a tape. Other State do not changed as they hold two type of list i.e. list of final state and transition states of Turing machine which are assumed to be simulated. These sections are different from the others.

The classic von Neumann fetch-and-execute process is repeatedly carries out by the TM which delta-rule of TM to be emulated next will be determined by the fetch process. The initial configuration section of its tape begins first of a TM. It begins by taking the alphabet character the current state-id digit which lies one cell to its right in its copy of the tape of TM which is present in the last section of the tape. After that TU will locate the correct transition Rule of TM having a set of rules and it is enclosed by the by the two colon symbols, which is the middle section of the tape. If transition rule is unable to find or simply no matching found, TU will crash similar to what TM would have done [5]. TU enters the execution phase after getting correct matching. To execute a transition rule, write character, TM read-head shift direction and the destination state identifier must be stored. When executing delta-rule, TU determine if TM has entered a final state or not after executing the final state check section. TU has its read head positioned over the current state identifier of TM after performing the

final state check points. The branch into several states of a TM is depends on the state of TM and the total number of states in the TM. If the branch state identifier is equal to the state-id under then the TM goes to the final state list and halts. TU will replicate the fetch-and-execute process till it halts, If TM has not halted.

3. Literature Survey

In mid-1960 the major evolution in the field of computer science has been taken place. The very first immense step has taken by the computer scientists towards theory of computation. Which familiarize the foundation for principles of computer science, such as automata theory (finite state machines), formal languages and grammars, computability and computation?.

A formal language is an abstraction of the general characteristics of programming language whereas automata theory model the hardware of computer system and complexity addresses what can be done in practice by computer. This leads to the introduction and study of various computing models covering various problem domains. The important formal languages – Regular language, context free language were limited to scope in term string acceptance, memory, and computation.

Finite automata have no storage unit, Pushdown automata uses stack as storage unit. As time goes on various scientist enhances the capability by adding two stacks and more. This lead to the fundamental concept of "TURING MACHINE" or in turn it's generating the idea of a mechanical or algorithm computation. And it became the complete analogy of the present computer system. Turing machine is an automaton whose temporary storage is a tape.

4. Existing System

In the evolution of highly computable machine UTM come in to the picture. A Turing machine that is able to simulate any other Turing machine is called a Universal Turing machine (UTM, or simply a universal machine). A UTM is the abstract model for all computational models. A UTM TU is an automaton that, given as input the description of any Turing Machine TM and a string w, can simulate the computation of M. To simulate Turing Machine as UTM many scientists designed simulators.

The smallest known universal Turing machines simulated by Minsky, Rogozhin, Baiocchi and Kudlek. These were efficient (polynomial time) simulators of Turing machines. But the problem with these was of simulation speed. These simulations were exponentially slow. They had simulated CFL and some recursive languages but got low speed and non-halting condition for certain string.

5. Proposed System

In This paper we are taking a recursive enumerable language i.e. $a^nb^nc^n$ and simulate it with the help of JFLAP tool. Check this problem of speed we are using JFLAP simulator. And our aim is to simulate UTM, which simulate the behavior of Turing Machine in polynomial amount of time. And Developing a Universal Turing Machine for recursively enumerable languages, check its operation again by JFLAP. And also it represents a Turing Machine as a directed graph. The simulation working can be explained with an example $a^nb^nc^n$ that is solved using a standard Turing machine and the UTM (TU).The language $a^nb^nc^n$ is a Recursive Enumerated language which can be implemented using a Turing machine.

6. Result and Discussion

Let us take one example *i.e.* The language $L = \{ a^n b^n c^n \}$. Now we have to implement this language with the help of JFLAP Tool. So first we have to start JFLAP and click the Turing Machine option from the menu, the menu are as follows:

Figure 2. Initial Configuration of JFLAP

After clicking on the Turing machine we will get a blank screen the blank screen looks as follows. It consists of a number of buttons, menus, and features those are used for the complete operation of finite automata. The main idea behind this is to study and analyze the feature and option supported by Turing machine and how a turing machine differs from finite automata.

To implement this language $L = \{ a^n b^n c^n \}$ by the Turing machine we have to add a number of states. Assume there are seven states which are given below. And take q0 as the initial state and q6 as the final state then the screen will be look like as follows:

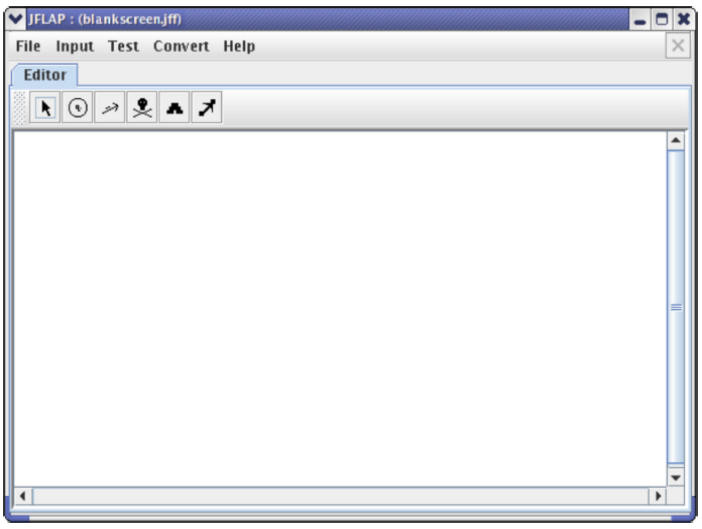

Figure 3. Initial Simulation Editor of JFLAP

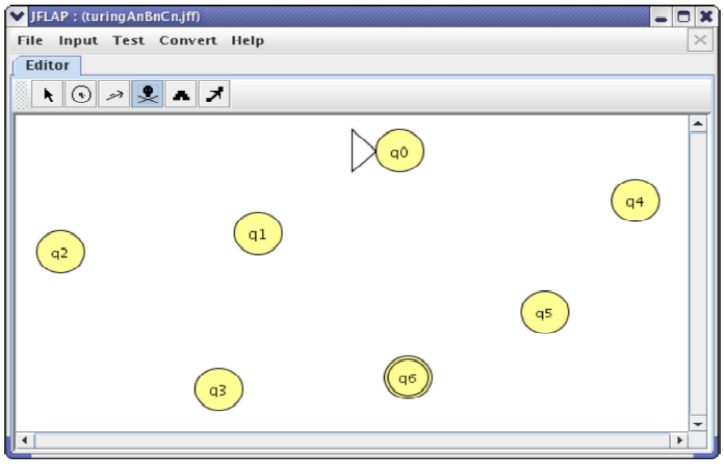

Figure 4. Configuration after Creating Number of States

Now we have to add the transitions between states. let us start with the state q0 and try to add one transition between q0 and q1.now you will see these transitions are something differ from the transitions of the finite automata and it consist of three input instead of one.

Figure 5. Configuration of Transition of Finite Automata

The first box represents the current value of the Turing machine. After completion of first step the second value will executed. The second value is the value replace the first value on the tape this step has been processed. The size of the values in these two boxes is limited to one character. The third value represents where the head will move after processing the step. It can be one of three values: 'R' (move right one square), 'L' (move left one square), and S (stay put and do not move the head). One could enter the value directly, or enter it from the pull-down menu that comes up when the third box is clicked on directly. Now, it's time to add input. To change the transition from the default, click on the first box. Enter a value of "a" for the first box, a value of "x" for the second box, and a value of "R" in the third box. Use Tab or the mouse to move between the boxes, and press enter or click the mouse on the screen outside the boxes when done. This transition has the following meaning. If the head is under an "a" and the machine is in state "q0", then replace the "a" with an "x" and move the head to the right. When done adding input, the area between q0 and q1 should resemble the example below.

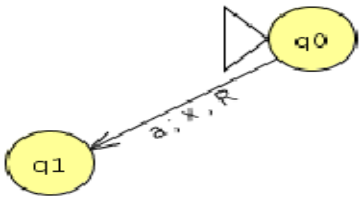

Figure 6. Configuration of Transition between States

Let's finish up the transitions. Add the transitions in your screen below to your Turing machine. If you would rather not add every transition directly and would prefer to load the file of the screen below, it is available at turing $a^nb^nc^n$.

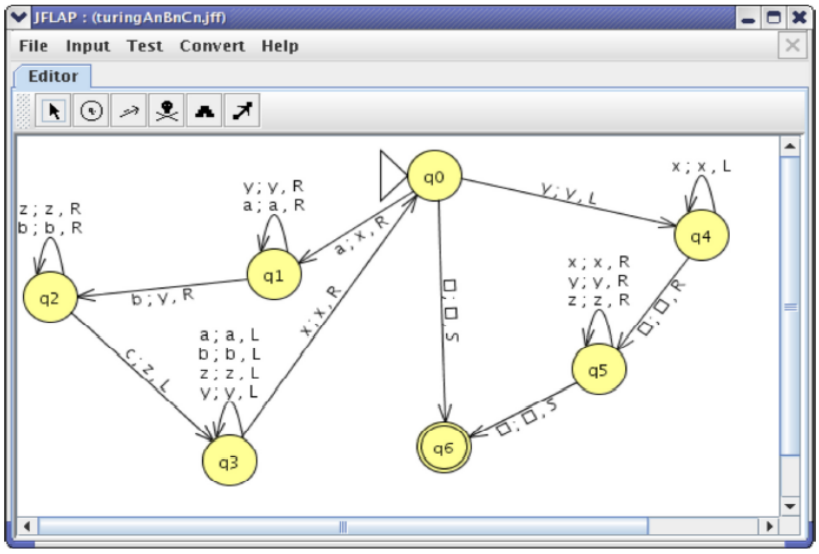

Figure 7. Configuration of JFLAP after Implementation

Now, let's try out our new Turing machine. Because of the number of steps, we will avoid the "Step" option we used with finite automata (although for finite automata titled "Step with Closure") and instead use the "Fast Run" option. To use this, click on the "Input" menu, and then click on "Fast Run". When it prompts you for input, enter "aabbcc", representing $a^2b^2c^2$. After clicking "OK" or pressing enter, the following screen should come up:

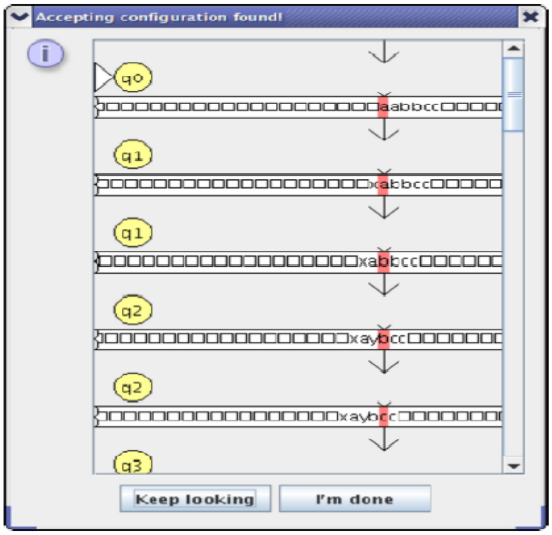

Figure 8. Configuration of JFLAP after pressing Fast Run

One can scroll down and see the tape, the current state, and the position of the head as the automaton processes the input step by step. One can see the algorithm at work, which is if the

head encounters an “a”, it replaces it with an “x”. Then, it replaces a corresponding “b” with a “y” and a corresponding “c” with a “z”. This repeats until it is no longer possible, and this loop is what makes up the cycle encompassing “q0”, “q1”, “q2”, and “q3”. Once this is done, the program makes sure that there is nothing but “x”s, “y”s, and “z”s left in the correct order. In the case of input with length zero, the program immediately goes to the final state.

The “Keep looking” button is for finding other possible paths in automata that aren't deterministic, which is not applicable here. When finished, click “I'm done.” Congratulations, you have built your first Turing Machine.

6.1. Transitions from Final States

Now we have to start with the final states .suppose the transitions should be enable in such a way that we can add one more final state efficiently. Now we have to consider two final states .so the figure will be modified from the previous figure.

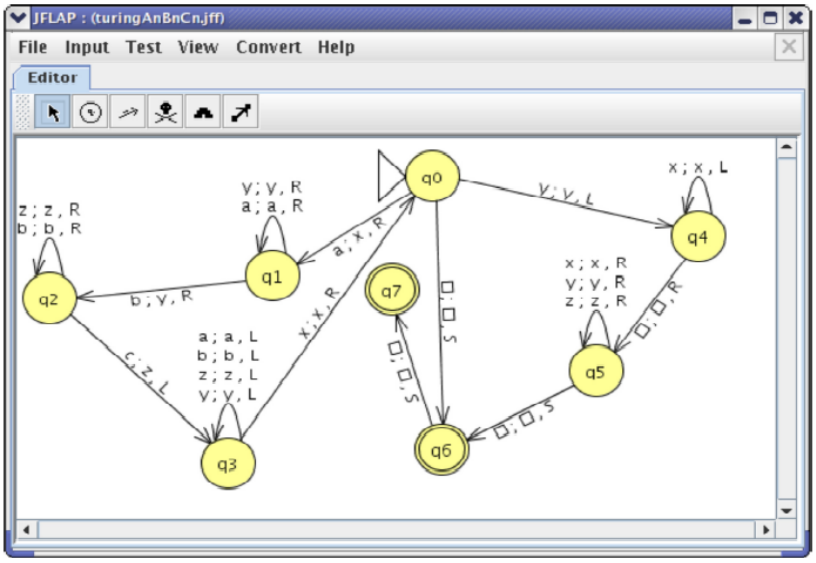

Figure 9. Final Configuration of JFLAP

Since there is no preference for enabled options and one edge extending from the final states which is q6 then one error will ding from the final state “q6”, the following error message will become visible. Suppose we want to create a machine then we have to enable the preference and after that eliminate all edges those are offending.

7. Conclusion and Future Work

The most powerful computational machines are the Turing machines. A Universal Turing Machine behaves like a single Turing machine but gives the solution for all computational problems. The Turing Machines gives an abstract model for real computer system. In this paper we are concentrating on both Turing machine as well as universal Turing machine. And we are taking the Recursive enumerable Languages and simulate it with the help of JFLAP .we are also concentrate on universal Turing machine of the recursive enumerable language. Actually other automata are different from the Turing machine is that a Turing machine linked with the recursive enumerable languages. Let us take the language $a^n b^n c^n$ is a recursively enumerable language and we are not able to simulate it through any other

automata except Turing machine. And Turing machine requires more storage space as compared to other automata like PDA, LBA etc. That's why the infinite tape of a Turing machine moves in both directions.

The future work includes

i. Developing a Universal Turing Machine for recursively enumerable languages with multiples tape
ii. Implement the concept of universality by including more symbols in the input alphabet as well as the tape alphabet
iii. Developing a Turing machine for recursively enumerable languages and converting the Turing machine to an unrestricted grammar.
iv. Create a block for the language $a^nb^nc^n$ and use this block to implement the language $a^nb^nc^nd^n$.
v. Developing a Turing machine for the language $a^nb^nc^nd^n$ and simulate it with the help of JFLAP.

References

[1] J. Jarvis and J. M. Lucas, "Understanding the Universal Turing Machine an implementation in JFLAP", ACM Portal, vol. 23, no. 5, (**2008**), pp. 180-188.
[2] E. Gramond and S. H. Rodger, "Using JFLAP to interact with theorems in automata theory", ACM Portal Proc. in SIGCSE, (**1999**), pp. 336-340.
[3] 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**), pp. 403-407.
[4] A. Rosen and D. Rosen, "The Turing Machine Revisited", MCon Inc., 2007. Online]http://www.jflap.org/tutorial/JFLAP tool and tutorials.
[5] J. Hopcroft, R. Motwani and J. Ullmann, "Introduction to Automata Theory", Languages and Computation, 3rd Edition, Addison- Wesley, (**2006**).
[6] S. H. Rodger and T. W. Finley, "JFLAP: An Interactive Formal Languages and Automata Package", ISBN 0763738344, Jones & Bartlett Publishers, (**2006**).
[7] Turing Machines with Finite Memory

Author

Tribikram Pradhan, he is currently working as an assistant professor in the Dept. of Information and Communication Technology, Manipal Institute of Technology, Manipal University, Karnataka, India. He has guided several students for their Graduate Studies. He has presented various research papers in National and International Conferences and Journals. And also He is a editor and reviewer for many International Journals and TPC members for 3 IEEE International Conferences. He received His B.Tech Degree in Computer Science and Engineering from Biju Pattnaik University of Technology (BPUT), Odisha, India and M.Tech Degree in Software Technology from VIT University, Vellore, Tamilnadu, India in 2010 and 2013 respectively. His research areas include Genetic Algorithm, Rough Set Theory, Data Mining, Computational Complexity, and Artificial Intelligence, etc.

