The International Journal Of Engineering And Science (IJES)
||Volume||2 ||Issue|| 3 ||Pages|| 56-59 ||2013||
ISSN: 2319 – 1813 ISBN: 2319 – 1805

Implementation of Unrestricted Grammar in To the Recursively Enumerable Language Using Turing Machine

[1,]Jainendra Singh, [2,] Dr. S.K. Saxena
[1,]*Department of Computer Science, Maharaja Surajmal Institute*
[2,]*Department of Computer Engineering, Delhi Technological University*

----------Abstract----------

This paper presents the implementation of the unrestricted grammar in to recursively enumerable language for JFLAP platform. Automata play a major role in compiler design and parsing. The class of formal languages that work for the most complex problems belongs to the set of Recursively Enumerable Language (REL).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, CFL as well as Recursive Enumerable Languages. Unrestricted grammar are much more powerful than restricted forms like the regular and context free grammars. In facts, unrestricted grammars corresponds to the largest family of languages so we can hope to recognize by mechanical means; that is unrestricted grammars generates exactly the family of recursively enumerable languages. Turing Machine is used to implementation of unrestricted grammar & RELs for JFLAP platform. JFLAP is most successful and widely used tool for visualizing and simulating all types of automata.

Keywords *- Automata, Compiler, CFG, JFLAP, PDA, REL, Unrestricted Grammar*

Date Of Submission:01,March, 2013 ⟺ Date Of Publication:15 March2013

I. INTRODUCTION

An automation is mathematical model for a finite state machine (FSM). A FSM is a machine that has a set of input symbols and transitions and jumps through a series of states according to a transition function. Automata play a major role in compiler design and parsing. Turing Machines are the most powerful computational machines. They possess an infinite memory in the form of a tape, and a head which can read and change the tape, and move in either direction along the tape or remain stationary. Turing Machines are equivalent to algorithms and are the theoretical basis for modern computers. JFLAP is software for experimenting with formal languages topics including nondeterministic finite automata, nondeterministic pushdown automata, multi-tape Turing machines, several types of grammars, parsing, and L-systems. JFLAP is extremely useful in constructing Turing Machine with multiple inputs. Complex Turing Machines can also be built by using other Turing Machines as components or building blocks for the same.

II. OVERVIEW OF JFLAP

JFLAP (Java Formal Languages and Automata Package) is instructional software for experimenting with automata and grammars, but goes further in allowing one to experiment with proofs and applications related to these topics.JFLAP's main feature is the ability to experiment with theoretical machines and grammars. With JFLAP one can build and run user-defined input on finite automata, pushdown automata, multi-tape Turing machines, regular grammars, context-free grammars (CFG), unrestricted grammars, and L-systems. After constructing the automaton or grammar, one can trace through a single input string or receive automatic feedback on multiple inputs.JFLAP's second feature is the ability to construct in steps the proof of the transformation of one form to another form.For example, after constructing a nondeterministic finite automata (NFA), one can step through its conversion to a deterministic finite automata (DFA), then to a minimal state DFA, and then to regular grammar. As another example, one can build a CFG and construct in steps the equivalent nondeterministic PDA. JFLAP's third feature is the experimentation with applications of theoretical material. One example is experimenting with parsing by constructing the SLR(1) parse table in steps, and then stepping through the parsing of input strings and the construction of the equivalent parse tree. The construction of the parse table includes building a special DFA that models the parsing process, thus seeing a use for a DFA.

Another application is building an L-system grammar of a plant, and rendering it to watch a simulation of the plant
growing.

III. TURING MACHINE IN JFLAP

A Turing machine is an automation whose temporary storage is tape. This tape is divided into cells, each of which is capable of holding one symbol. Associated with the tape is read-write head that can travel right or left on the tape and that can read and write a single symbol on each move. Turing Machines are the most powerful computational machines. The Turing Machine (TM) is the solution for the halting problem and all other problems that exist in the domain of computer science. Turing Machines provide an abstract model to all problems. It can work with Recursively Enumerable Language and Unrestricted Grammar.

3.1 Turing Machine

A Turing Machine *M* is defined by

$$M = (Q, \Sigma, \Gamma, \delta, q_s, \square, F)$$

where

Q is the set of internal states $\{q_i \mid i$ is a nonnegative integer$\}$
Σ is the input alphabet
Γ is the finite set of symbols in the tape alphabet
δ is the transition function
S is $Q * \Gamma^n \rightarrow$ subset of $Q * \Gamma^n * \{L, S, R\}^n$
□ is the blank symbol.
q_s (is member of *Q*) is the initial state
F (is a subset of *Q*) is the set of final states

3.2. Recursively Enumerable Language

A language L is said to be recursively enumerable if there exists a Turing Machine that accepts it. It implies that there exists a Turing Machine M, such that, for every w ∈ L

$$q_0 w \vdash^* M\ x_1 q_f x_2$$

with q_f a final state. The definition says nothing about what happens for w not in L; it may be that machine halts in a nonfinal state or that it never halts and goes into an infinite loop.Regular languages form a proper subset of Context Free Languages. So PDA are more powerful than finite automata. But CFLs are limited in scope because many of the simple language like $a^n b^n c^n$ are not context free. So to incorporate the set of all languages that are not accepted by PDAs and hence that are not context free, more powerful language families has been formed. This creates the class of Recursively Enumerable Languages (REL).

3.3. Unrestricted Grammars

A grammar G = (V,T,S,P) is called unrestricted if all the productions are of form

$$u \rightarrow v,$$

where u is in $(V \cup T)^+$ and v is in $(V \cup T)^*$.

In an unrestricted grammar, essentially no conditions are imposed on the productions. Any number of variables and terminals can be on left or right, and these can occur in any order. There is only one restriction: λ is not allowed as the left side of a production. Unrestricted grammars are much more powerful than restricted forms like the regular and context free grammars.In fact, unrestricted grammars correspond to the largest family of languages so we can hope to recognize by mechanical means; that is, unrestricted grammars generate exactly the family of recursively enumerable languages

IV. IMPLEMENTATION

Any language generated by an unrestricted grammar is recursively enumerable. For every recursively enumerable language L, there exists an unrestricted grammar G, such that L = L(G). Language L is said to be recursively enumerable if there exists a Turing Machine that accepts it.

We take unrestricted grammar, which is shown in figure 1.The various strings are applied to this unrestricted grammar (which is also shown in figure 2).

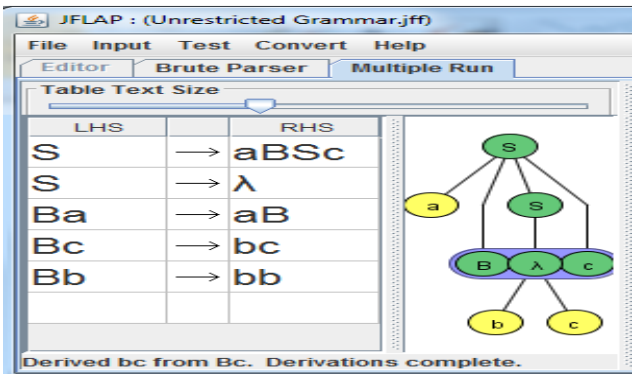

Figure 1: Unrestricted Grammar (with parse tree abc)

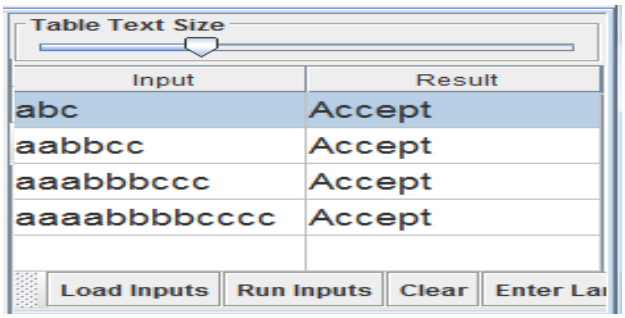

Figure 2: Accepted String by Unrestricted Grammar

This grammar drives the language L= $a^nb^nc^n$, n > 0. The language $a^nb^nc^n$ is a recursively enumerable language which cannot be implemented using a FA as well as a PDA. The standard Turing machine T_M for the language $a^nb^nc^n$ is given in figure 3. The various strings are applied to the Turing Machine with multiple run. A few results are shown in figure 4.

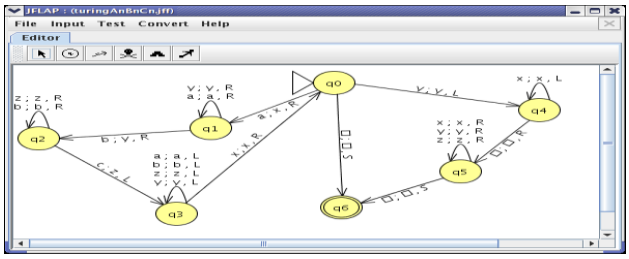

Figure 3: Turing Machine for $a^nb^nc^n$

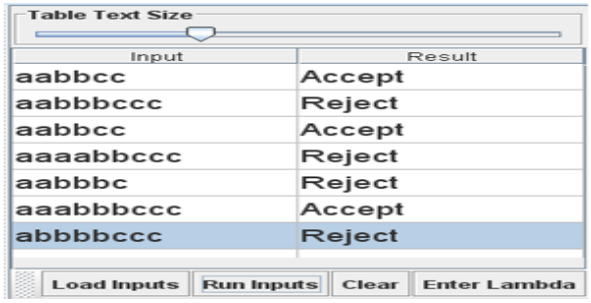

Figure:4 Multiple Run by Turing Machine

Any Turing machine can be converted REL into an unrestricted grammar. JFLAP defines an unrestricted grammar as a grammar that is similar to a context-free grammar (CFG), except that the left side of a production may contain any nonempty string of terminals and variables, rather than just a single variable. In an unrestricted

grammar, the left side of a production is matched, which may be multiple symbols, and replaced by the corresponding right hand side. There are 3 major steps that the algorithm follows in order to convert the Turing machine to an unrestricted grammar. The first step is applying the basic rules from the start variable. It allows the grammar to generate an encoded version of any string q0w with an arbitrary number of leading and trailing blanks. The second step is applying generating the production for each transition of Turing machine. The last step is applying additional rules to our productions if we enter one of final states of TM, so we can derive terminals.

V. CONCLUSION

Turing Machines are the most powerful computational machines. The Turing Machines provide an abstract model to all the problems. This paper describes the working of a Turing Machine for Recursively Enumerable Languages and Unrestricted Grammar for JFLAP platform. The Turing Machines differ from all other automata as it can work with Recursively Enumerable Languages and Unrestricted Grammar. Any language generated by an unrestricted grammar is recursively enumerable. The language $a^nb^nc^n$ is a recursively enumerable language which cannot be implemented using a Finite Automata or a PDA but can done using a Turing Machine. This requires more storage than for Context Free Languages and hence the Turing Machine with the infinite tapes, extendable in both directions are used for this.

VI. FUTURE WORK

[1] Developing a Universal Turing Machine for recursively
[2] enumerable languages
[3] Implement the concept of universality by including more
[4] symbols in the input alphabet as well as the tape alphabet

VII. ACKNOWLEDGEMENTS

We thank Dr. Susan Rodger of Duke University for the work on Formal Language and Automata for JFLAP platform. We also thanks Peter Linz of University of California for the meritorious work on the REL and Unrestricted Grammar.

REFERENCES

[1] Susan H.Rodger, Eric Wiebe, Kyung Min Lee, Chris Morgan, Kareem Omar and Jonathan Su, "Increasing engagment in automata theory with JFLAP", ACM Transactions,, 2009, 403-407.
[2] Eric Gramond and Susan H. Rodger, "Using JFLAP to interact with theorems in automata theory",ACM Poratl Proc. In SIGCSE,1999,336-340.
[3] Peter Linz, An Introduction to Formal Languages and Automata (3rd Edition, Narosa Publishing House,2003)
[4] J.Hopcroft, R.Motwani and J.Ullmann, Introduction to Automata Theory, Language and Computation(3rd Edition, Addison-Wesley, 2006)
[5] [Online]http://www.jfalp.org/tutorials
[6] Susan H.Rodger and Thomas W.Finley, JFLAP :An Interactive Formal Languages and Automata Package, ISBN 0763738344, Jones & Bartlett Publishers, 2006
[7] Susan H. Rodger, Jinghui Lim, and Stephen Reading, "Increasing Interaction and Support in the Formal Languages and Automata Theory Course", The 12th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2007), 2007 58-62.
[8] Sumitha C.H and Krupa Ophelia Geddam, "Implementation of Context Free Languages in Turing Machines",IEEE conf. on second International conference on Machine Learning and Computing,2010