International Journal of Electrical Electronics & Computer Science Engineering
Special Issue - NCSCT-2018 | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
March, 2018 | Available Online at www.ijeecse.com

A Review: Chomsky Hierarchy of Languages

Dr. Megha Gupta[1], Pankaj Sharma[2], Rahul Patni[3]
[1]Associate Professor, [2-3]UG Scholar, Dept. of CSE, Poornima Institute of Engineering & Technology, Jaipur, Rajasthan
[1]megha.gupta@poornima.org , [2]2015pietcspankaj070@poornima.org,[3]2015pietcsrahul085@poornima.org

***Abstract:** This paper is concerned with learning of Chomsky classification, theory of pushdown automata, Turing machine, Formal language and computation. The Chomsky hierarchy is a containment hierarchy of classes of formal grammars. The automata provides us a new technology which can help us for the automation of every machine and Help to build software for designing and checking the behaviour of digital circuits that have finite number of distinct states. The aim of this paper is to discover things involved in Chomsky Hierarchy and describing the machine that are used in Chomsky Hierarchy.*

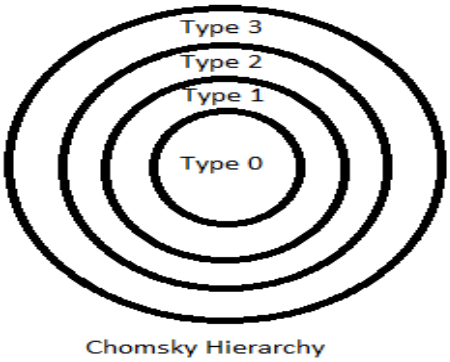
***Keyword**: Chomsky Hierarchy, Finite State Machine, Pushdown Automata, Turing Machine, Linear Bounded Automata.*

I. INTRODUCTION

Chomsky's Hierarchy - The Chomsky represents the categories of languages that are accepted by different machine. Chomsky classified the language into four types as given below-

Language class	Language	Grammar	Machine	Example
Type 3	Regular	Regular Grammar	FSM i.e. NFA or DFA	a*b*
Type 2	Context Free	Context Free Grammar	PDA	a^nb^n
Type 1	Decidable Language	Context Sensitive Grammar	Linear Bounded Automata	$a^nb^nc^n$
Type 0	Computable Language	Unrestricted Grammar	Turning Machine	n!

This is in hierarchy so every language of Type 3 is also of the Type 2, 1 and 0. Similarly Type 2 is also of the Type 1 and 0 etc.



Chomsky Hierarchy

Type 3: Regular Languages:

The Language which can be described using Regular expressions is called regular language. In this all the productions rules are in the given form :

A-> α or A-> α B

Here A,B∈ (non terminal) and α ∈ (terminal)

This language can be modelled by NFA and DFA.

Type 2: Context Free Languages:

The language which can be represented by context free Grammar (CFG) is called the context free language. The production rule is of the form:

A→α here A ∈ (single non terminal) and a ∈ (terminals and non terminals)

This language can represent by PDA.

Type 1: Context Sensitive Languages:

The language which can be represented by context sensitive grammar is called context sensitive language. The production rule is in the form of:

$\boldsymbol{\alpha}$ A $\boldsymbol{\beta}\rightarrow$ $\boldsymbol{\beta}$

Where A ∈(single non terminal) and , $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ ∈ (terminals and non terminals)*

This language can be represented by Linear Bounded Automata.

Type 0: Unrestricted Languages:

Type 0 grammar is a phase structure grammar without any restriction. All grammars are type 0 grammar. The production rule is of the form:

$\{L_c,NT,R_c\}$→ (String of terminal or non terminal or both)

L_c=left context, R_c=right context and NT=non terminal

This can be modelled by Turing machine.

II. FINITE STATE MACHINE (FSM)

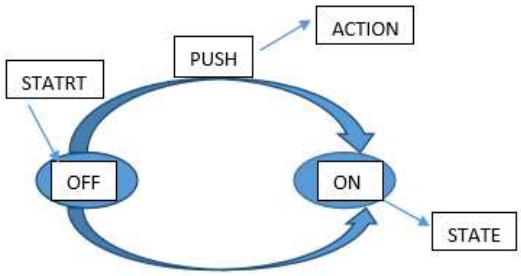
The Finite state machine system is represented in mathematical models of a system with respect to given input, it provides output.

Finite state machine contain five elements:

1. Input-finite automata takes input and produce output.

International Journal of Electrical Electronics & Computer Science Engineering
Special Issue - NCSCT-2018 | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
March, 2018 | Available Online at www.ijeecse.com

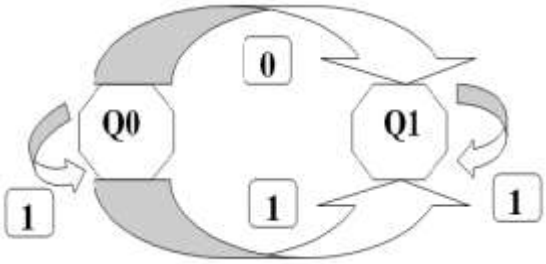
2. Output-finite automata takes input and produce output.
3. States- automata have finite no. of state.
4. State relation-the next state of automata is determined by the present state and the present input.
5. Output relation–it is related to either state only or both (input and state)



Types of Finite State Machine:

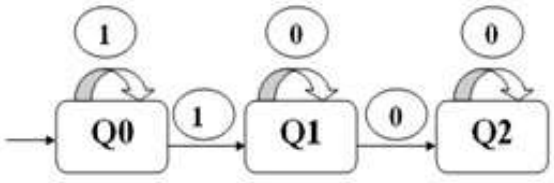
1. *Deterministic finite machine (DFA):*

If there is only one path for a specific input from current state to next state then it is called deterministic finite automata.



2. *Non Deterministic finite machine (NDFA)*

If there exists many paths for a specific input from current state to next state then it is called non deterministic finite automata



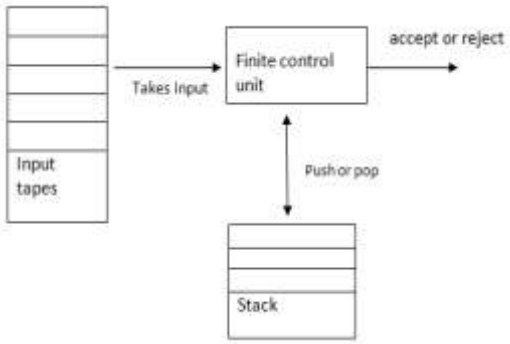
Non Deterministic Finite Automata

III. PUSH DOWN AUTOMATA (PDA)

The Pushdown Automaton is used to implement context free grammar.

Pushdown automata have three components:

1. Input Tape
2. Finite Control
3. Stack



Pushdown Automata Model

PDA can be defined as collection of 7- tuples (Q, ∑, S, **δ**, q_0, I, F)

1. **Q** is the finite set of state
2. ∑ is input set
3. **S** is stack alphabets
4. **δ** is the transition function
5. q_0 is the initial state here $q_0 \in Q$
6. **I** is the start symbol which is in S (I∈S)
7. **F** is the finite set of final states (F∈Q)

In PDA the transitional function (**δ**) is in the form = Q × (∑ ∪ {ε}) × S × Q × S*

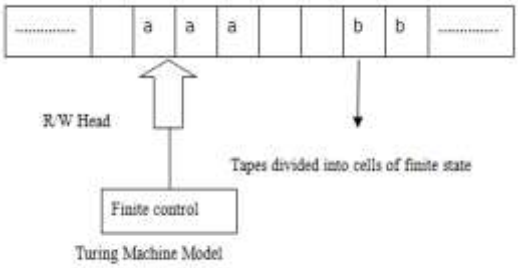
IV. TURNING MACHINE

A Turning machine is an accepting device. It accepts all the language that is generated by the Type 0 grammar.

It is a mathematical model and consists of infinite length tape. These tapes are sub-divided into cells on which input is given before. In which input tape is reads by head. Turing machine has state register in which store the states

It reads an input symbol and replaced it with another symbol, and then its internal state is changed, and it moves from one cell to the right or left. After that it reach at the final state and input string is accepted or rejected.

Turing machine can be defined as collection of 7-tuple (Q, X, ∑, δ, q_0, B, F) where-



Turing Machine Model

1. **Q** is a finite set of states
2. **X** is the tape alphabet

3. ∑ is the input alphabet
4. **δ** is a transition function
5. **q_0** is the initial state
6. **B** is the blank symbol
7. **F** is the set of final state

In PDA the transitional function (**δ)** is in the form= **δ**: Q x X → Q x X x {Left Shift, Right Shift)

V. LINEAR BOUND AUTOMATA

A linear bounded automaton is a non deterministic type of Turning machine. Tape of linear finite automata is bounded in finite length.

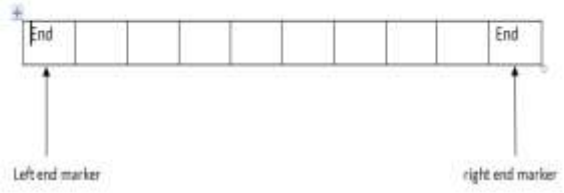
Length = function (Length of the initial input string, constant k)

Here the Memory Information ≤ k * Input Information

In this computation is restricted to only constant bounded area. The input alphabets contain two special symbols that are as follows:

1. Left end marker 2. Right end marker

It means transition neither move to the left to the left end marker nor right to the right end marker of the tape.



Linear Bounded Automata

It can be defined as collection of 8-tuple (Q, X, ∑, q_0, ML, MR, δ, F) where –

1. **Q** is a finite set of states
2. **X** is the tape alphabet
3. ∑ is the input alphabet
4. **q_0** is the initial state
5. **M_L** is the left end marker
6. **M_R** is the right end marker where $M_R \neq M_L$
7. **δ** is a transition function
8. **F** is the set of final states

In linear bounded automaton the transition function in the form = maps each pair (state, tape symbol) to (state, tape symbol, Constant 'k') where k can be 0 or +1 or -1

VI. CONCLUSION

In this paper we have tried to explain basic concepts of theory of computation which includes the Chomsky hierarchy which represent the class of languages that are accepted by different machine. Basic idea of what are Finite state machines, Automata, Pushdown Automata, Linear bound automata, turning machines have also been analysed in this paper.

VII. REFERENCES

[1] Automata, languages and computation by K.L.P Mishra and N. Chandrasekaran

[2] Theory of computation by A.A. Puntambekar

[3] http://www.info.univ-tours.fr

[4] http://www.tutorialpoint.com

[5] B. Khoussainov, A. Nerode, Automata Theory and Its Applications, Springer.

[6] W. Thomas, languages, automata and logics, handbook of formal languages, vol. 3, pages 389-455.