AUTOMATA THEORY FOR SOFTWARE DEVELOPMENT

Avdhesh Mann*

Dr.Rajneesh Talwar **

Dr.Bharat Bhushan***

Rakesh Gupta****

Abstract: *In this paper we discussed about a software i.e. Compiler. In each language users deal with compiler.But how the compiler will work that depend totally on language features. So here we discuss if someone want to design a new compiler what type of theory he or she should study before making that compiler according to the new features which he or she should use in his program.*

* B.Tech (Computer Science), U.I.E.T , Kurukshetra University, Kurukshetra (Haryana), INDIA

** Principal, SWIFT Group of Institutions, Rajpura (Patiala),Punjab, (INDIA)

*** Associate Professor & Head, G.N.K, Yamunanagar (Haryana),INDIA

**** Programmer , D.B.S.C.R., Government Polytechnic Sampla (Rohtak), Haryana, INDIA

INTRODUCTION

As we know, software development is directly related to customer need. Software is developed with systematic approach. Means with the different models of software engg. like spiral model, water fall model, RAD model, v-model, iterative model and so on .We take the water fall model now : it has different phases to develop the project and these phases are like this:

Specification

Design

Coding

Testing

Operation and maintenance

If we develop the software using this approach means before coding if we do the designing part and before designing if we clear our all specification than we can get a better quality product according to that customer need. However we can use other models by analyzing customer requirements.

HOW TO RELATE SOFTWARE DEVELOPMENT WITH AUTOMATA

It's a smart work .As every body knows that the software is a collection of programs that do need full task for the users. Now we talk about all system programs like translators e.g. compiler, interpreter, assembler, etc. Now here we deal only with one translator i.e. compiler. So it means compiler is also a software .A compiler is usually used to convert high level language in to machine level language. For different languages there can be different types of compilers like c compiler, java compiler etc.

$$\begin{tikzcd} \text{Source prog} \arrow[r] & \boxed{\text{Compiler}} \arrow[r] & \text{object prog.} \end{tikzcd}$$

Now the compiler software is also made up of different phases. Its different phases are

a) Lexical analysis

b) Syntax analysis

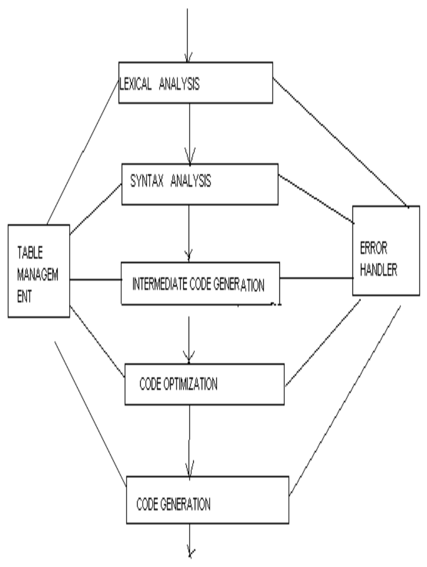
c) Intermediate code generation

d) Code optimization

e) Code Generation

f) Error Handling

g) Table Management



We can say all these phases behave like a software model. By studying all these phases we can design diifferet types of compiler according to language features. So these phases are analyzed by automata theory for making compilers.

I. WHAT IS AUTOMATA THORY

An automaton is generally called a Machine. It does all the work automatic. Like automatic packing m/c etc.

We just need to give input, we get the output. We can say that it is a basic machine.

Example:

OR Gate

Input	Output
(0, 0)	0
(0, 1)	1
(1, 0)	1
(1, 1)	1

Such type of basic m/c doesn't deal with states.

Computer is also a machine. We give the input, get the output. But in this m/c, we have to deal with states.

State: The configuration of machine at particular instant of time.

Like holding state, wait state, processing state etc. In this, record of state is kept. So we always have to use storage elements i.e. auxiliary memory. .

So such type of automaton (or m/c) has following characteristics:

1. Input: At each discrete instant of time t1, .t2…..,input values I1,I2,….. each of which can take a finite no. of fixed values from the input alphabet which are applied to the O/P side of mode.
2. Output: o1, o2----------which can take finite no's of fixed values from an o/p alphabet.
3. States: At any instant of time the automaton can be in one of the following states q1, q2………qn.
4. State Relation: The next state of an automaton at any instant of time is determined by the present state and the present input.
5. Output Relation: Output is related to either state only or to both the input and state.

An automata represented by 5 tuple $(Q,\Sigma, \delta, q0, F)$ where

Q is a finite nonempty set of states

Σ is a finite non empty set of states called input alphabet.

δ Is a transition function such that:

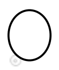
$\delta: (Q \times \Sigma) \to Q$

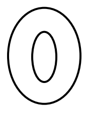
$q0 \in Q$ as the initial state.

F is final state.

Where $F \in Q$ is the set of final states.

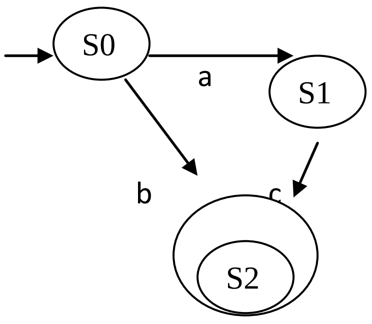
→ Initial state is represented by a circle with arrow head.





Final state is represented by two concentric circles

Example:



Q= {s0, s1, s2}

Σ = {a, b, c}

δ(s0, a)=s1

δ(s0, b)=s2

δ(s1, c)=s2

F= {s2} i.e. final states

Q0= {s0} i.e. initial state

We study here only 2 phases of compiler by automata theory:

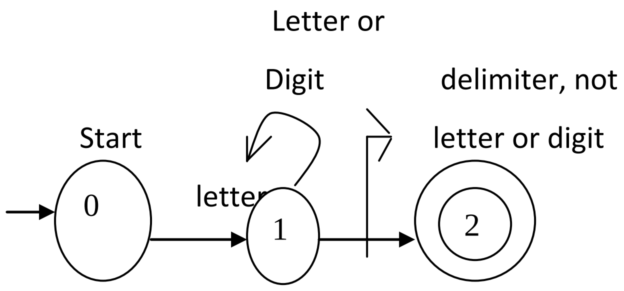
First is **LEXICAL ANALYZER**

Now, I will take an example in C program. I will relate it with C compiler and then relate, different features of C program with automata.

Suppose I have a program in C language .We use identifier, constants, operators, keywords in our program. Now the question arise how our compiler recognize that this particular string is identifier or keyword. The answer comes that it is up to the compiler.

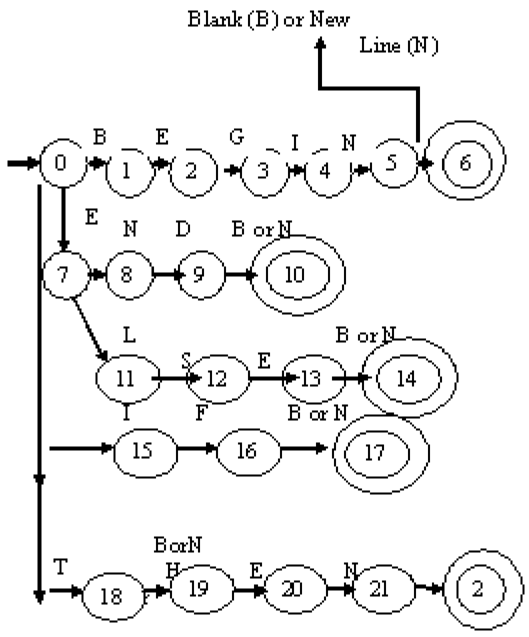
But compiler solves the problem by automata concept in which we design a machine called automata. In this machine we take a string and try to evaluate that string. If we reach on the final state of M/c using that string, we can say that string is acceptable by automata else not acceptable by automata

Suppose a M/C for the identifier is:



If the given string match by this machine that the given string will be identifier else it can check for keywords. Similarly, we can draw machine for keywords, constant etc.

For Keyword is:



REGULAR EXPRESSION:

These are very useful notations. We can say that regular expressions are language for the set of inputs.

Example

01

100

1100

For all these input string we can write a common language i.e.

(0+1)*

Similarly we can write language for 'identifier' is:

Identifier=letter(letter| digit)*

For keywords:

Keyword=BEGIN|END|IF|THEN|EL

NOW (2ND PHASE) SYNTAX ANALYZER WITH AUTOMATA:

Here we study about context free grammar.CFG are useful for describing arithmetic expressions with arbitrary nesting of balanced parentheses and block structure in programming languages. A CFG is denoted by G= (V, T, P, S) where V and T are finite sets of variables and terminals. P is a finite set of productions is of form:

A-> x

Where A is a variable and x is a string of symbols from(VU∑)*

And S is a special variable called the Start symbol.

Now a concept related to grammar is derivation tree. It imposes a structure on the words of a language that is useful in application such as the compilation of programming languages. The vertices of a derivation tree are labeled with terminal or variable symbols of the grammar. If an interior vertex n is labeled with A and the sons of n are labeled X1, X2…..Xk from the left then A->X1, X2, X3, Xk must be a production.

Example:

S->Aas|a

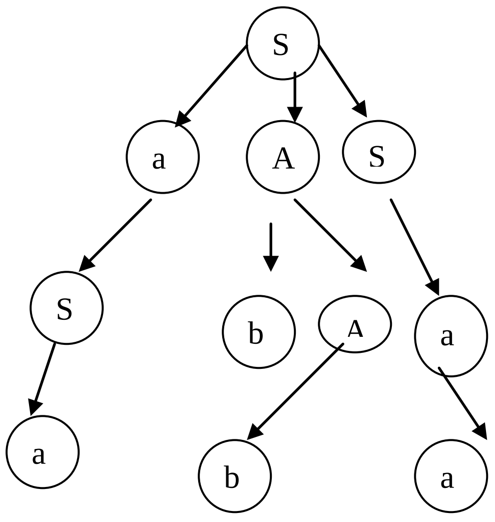
A-> SbA|ss|ba

We have to derive

S-> aabbaaa

Sol: S->aAS=>aSbAS=>aabAS=>aabbaS=>aabbaa

Its derivation tree is:



So according to our language feature we can design leftmost and rightmost derivation tree.

A derivation A=>w is called a leftmost derivation if we apply a production to the leftmost variable at every step.

A derivation A=>w is a rightmost derivation if we apply production to the rightmost variable at every step.

PROBLEM STATEMENT AND PROPOSED SOLUTION:

As we are seeing daily new applications are created and programmer are trying to find the ways to solve applications by software. Every software required a platform that fully satisfied the application. What the features we needed we map into the programs code. So for every new language we may required to develop a new compiler. So to map these new features we study automata theory which is a deep study of compiler making.

In this paper I took only two modules or phases of compiler and trying to relate it with program code. That is made up of different features like variables,identifier,keywords etc. and I tried to make it understand easily how our compiler understand all these features and give successfully compilation of program.

REFERENCES:

- "Animation of the generation and computation of finite automat a for learning software" Beatrix Braune, Stephan Diehl, Andreas Keeren and Rein hard Wilhelm
- "The theory of definite automata" Perles, M.Robin, M.O. Sharmir
- "Tree Automata Techniques and Applications" Hubert Common, Max, Dauchet, Remi Gilleren
- "Tree Walking Pebble Automata" Joost, Engelfriet and Hendrik Jan Hoogeboom
- "The general and logical theory of automata" John Von Neumann
- "Learning compiler design as a research activity" Francisco Moreno Seco, Mikel L.Forcada