Chapter 3

Finite State Automaton: A Tool to Represent Formal language

3.1 Importance of Finite state automata:

A finite state automaton is an important notion for the study of formal languages. On one hand, it is a mathematical tool used for implementing regular expressions and on the other hand it is one of the most noteworthy devices of computational Linguistics.

Finite state automaton is important in various fields of interest such as it is one of the most significant tools of computational linguistics. Other varieties of automata such as finite-state transducers, Hidden Markov Models, and N-gram grammars are important components of applied linguistics. In speech recognition and synthesis, machine translation, spell-checking, and information-extraction, the standard notation for characterizing text sequences FSA is used frequently.

An automaton is a mathematical object that takes a word as input and decides either to accept it or to reject it. Since all computational problems are reducible into the accept/reject question on words, automata theory plays a

vital role in computational theory.

Formal language over an infinite set can be represented finitely by FSA. As mentioned before each model in automata theory plays an important role in several applied areas. Finite automata are used in text processing, compilers, and hardware design. Besides Computer Science, cellular automata a type of FSA are used in the field of Biology.

Finite automata, regular grammars and regular expressions are all equivalent ways of describing regular languages. The relation among these four theoretical constructions is sketched/outlined in the following figure.

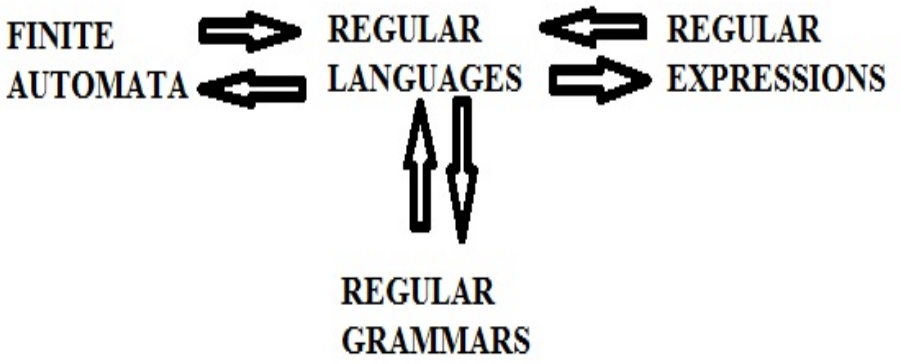

Figure 3.1: Relation of Finite automata, Regular grammars, Regular expressions and Regular languages

The automata exist in only one state at a time. They can change from one state to another when initiated by a triggering event or condition, this process is called a transition. The automaton is represented as a directed graph known as state graph which consists of a finite set of vertices known as nodes, together with a set of directed links between pairs of vertices called arcs. Vertices are represented by circles and arcs by arrows. An automaton can also be represented with a state-transition table. The state-transition table also represents the start state, the accepting states, and what transitions

leave each state with which symbols. There are two parts of Finite state automata. One part, whose behaviour during recognition is fully determined by the state it is in and there is one and only one state to which the automaton can transit from its current state is called "Deterministic". The other one, a "Nondeterministic" finite automaton has the power to be in several states at one time. This ability is often expressed as an ability to guess something about its input.

Organization of this chapter is as follows. Section 3.2 goes toward description of how automata are used to describe regular languages. Section 3.3, an algorithm to construct deterministic finite state automata is proposed. Section 3.4 gives result analysis of the above algorithm. We conclude the chapter in section 3.5 by summarizing the observations.

3.2 Use of Finite state automata to describe Regular languages:

Some illustrations about finite state automaton and state-transition table and their contribution to represent regular languages are given in this section.

1. Example

   The following (infinite) set gives the language of a singing bird:

   tuu!

   tuuu!

   tuuuu!

   tuuuuu!

   tuuuuuu!

. . .

The Finite state automata of this infinite input is

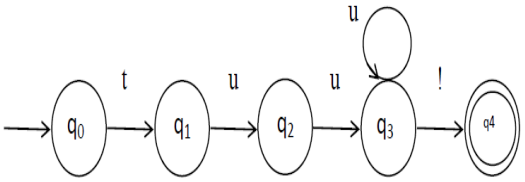

Figure 3.2: Finite state automaton for infinite word ‘tuu!’

The loop above the FSA denotes the infinite word ‘u’. The infinite set can be represented by a state transition table. From this table it is known about the start state, accepting states, the transitions from one state to another and about the symbols. There is a colon in the state 4 which

Table 3.1: State transition table for an infinite input ‘tuu!’

	input		
State	t	u	!
0	1	ϕ	ϕ
1	ϕ	2	ϕ
2	ϕ	3	ϕ
3	ϕ	3	4
4 :	ϕ	ϕ	ϕ

indicate that it is a final state. The symbol ϕ is used to indicate an illegal or missing transition. The first row indicates the state 0, when the input is ‘t’ the state is 1. It is a case of failure if in state 0 the input is ‘u’ or ‘!’. A deterministic finite automaton is defined by [48] the following five parameters:

$Q = q_0 q_1 q_2 ... q_{N-1}$ a finite set of N states

Σ a finite input alphabet of symbols

q_0 the start state

F the set of final states, $F \subseteq Q$

$\delta(q, i)$ the transition function or transition matrix between states. Given a state $q \in Q$ and an input symbol $i \in \Sigma$, $\delta(q, i)$ returns a new state $q' \in Q$. δ is thus a relation from $\delta : Q \times \Sigma \rightarrow Q$. For the language of a singing bird automaton in $Q = q_0 q_1 q_2 ... q_{N1}$, $\Sigma = t, u, !$, $F = q_4$ and $\delta(q, i)$ is defined by the transition table above.

The following two examples are based on finite inputs. Here there is no loop in the FSA. In the state transition table here also there is a colon in the last state, which indicates that it is a final state. The symbol ϕ is used to indicate an illegal or missing transition. The first row indicates the state 0 i.e. the starting state.

2. Example: Let us consider the word 'hi!' A Finite State Automaton for this word is

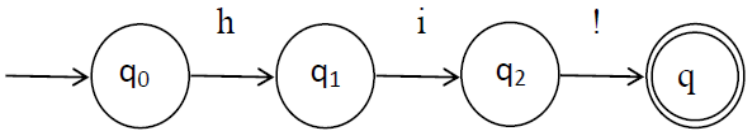

Figure 3.3: Finite state automaton for the word 'hi!'

Table 3.2: State transition table for a finite input hi!

	input		
State	h	i	!
0	1	ϕ	ϕ
1	ϕ	2	ϕ
2	ϕ	ϕ	3
3 :	ϕ	ϕ	ϕ

3. Example: Let us consider another word 'hello!' A Finite State automaton for this word is

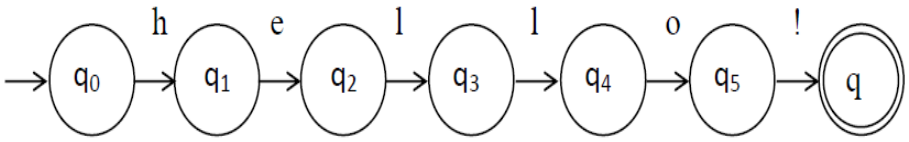

Figure 3.4: Finite state automaton for the word 'hello!'

Table 3.3: State transition table for a finite input 'hello!'

	input				
State	h	e	l	o	!
0	1	ϕ	ϕ	ϕ	ϕ
1	ϕ	2	ϕ	ϕ	ϕ
2	ϕ	ϕ	3	ϕ	ϕ
3	ϕ	ϕ	4	ϕ	ϕ
4	ϕ	ϕ	ϕ	5	ϕ
5	ϕ	ϕ	ϕ	ϕ	6
6 :	ϕ	ϕ	ϕ	ϕ	ϕ

3.3 An Algorithm to construct Deterministic Finite state automata:

It is discussed earlier how FSA is important for studying formal language, computational Linguistic and other branches of Computer Science, Biology etc. An algorithm which gives DFSA of a word will be helpful in those investigations. In the published paper [16] an algorithm which gives DFSA of a word is introduced. The following pseudo code gives instantly the deterministic finite state automata of a string.

Algorithm:

1. Initialize a word = $'w'$
2. Set len = length of w
3. **For** $i = 0$ to len do
4. Draw an Arrow mark $\longrightarrow$
5. **If** $(i = len)$ // finite state
6. Draw two circles [one is inner name it q_i, another is outer]
7. **else**
8. Draw a circle and name it q_i
9. **If** $(i \neq 0)$
10. Label the Arrow mark with character of the word at i^{th} position
11. **End**

3.3.1 Result Analysis:

The algorithm introduced gives finite state automaton of a word over regular languages instantly. Initial stage is preceded by an arrow. The final stage is denoted by double circle. The stages are named as q_i. Each stage is separated by arrow marks. The character of the word at the i^{th} position is marked just above the arrow mark. Using the above algorithm finite state automata of some words are given below.

- Example (i): The FSA of the word 'abcd' is

Figure 3.5: Finite state automaton of the word 'abcd'

- Example (ii): The FSA of the word 'book' is

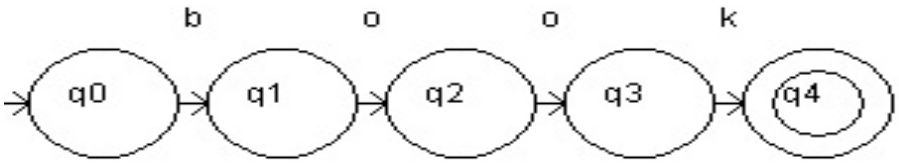

Figure 3.6: Finite state automaton of the word 'book'

- Example (iii): The FSA of the word 'technology' is

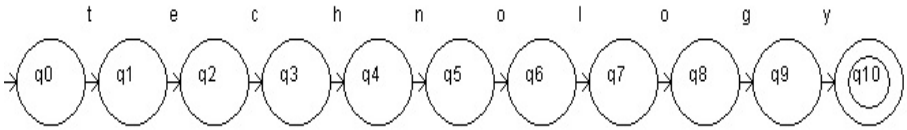

Figure 3.7: Finite state automaton of the word 'technology'

3.4 Conclusion of the Chapter:

A finite state automaton is a very important notion for the study of regular languages. In this chapter finite state automata and state transition table are discussed in the context of regular languages. An algorithm to construct deterministic finite state automata is introduced. This algorithm gives finite state automaton of a word over regular languages instantly. Initial and final stages are distinct. The stages are named as q_i. The stages are separated by arrow mark. The character of the word at the i^{th} position is marked above the arrow mark.

*****