1.2.3 Language

Now, let us see "What is a language? Give example"

Definition: A *language* can be defined as a set of strings obtained from Σ^* where Σ is set of alphabets of a particular language. In other words, a language is subset of Σ^* which is denoted by $L \subseteq \Sigma^*$. For example,

- A language of strings consisting of equal number of 0's and 1's can be represented as

  $\{\varepsilon, 01, 10, 0011, 1010, 0101, 0011, \ldots.\}$
- The language of strings consisting of n number of 0's followed by n number of 1's can be represented using the set as shown below:

  $\{\varepsilon, 01, 0011, 000111, \ldots\ldots\}$
- A language containing empty string ε is denoted by $\{\varepsilon\}$
- An empty language is denoted by ϕ.

Now, let us see "What is a sentence? Give example"

Definition: A string that belongs to a language is called word or language. For example, a lang

1.2.2 String

Now, let us see "What is a string? Explain with example"

Definition: The sequence of symbols obtained from the alphabets of a language is called a string. Formally, a *string* is defined as a finite sequence of symbols from the alphabet Σ. **Note:** An empty string is denoted by the symbol ε (pronounced as epsilon) or λ (pronounced as lambda) and note that $\varepsilon \notin \Sigma$ i.e., ε is not part of Σ.
Note: Let us use the symbol ε indicating an empty string instead of the symbol λ.

Example 1: Let $\Sigma = \{0, 1\}$ is set of alphabets. The various strings that can be obtained from Σ are

{0, 1, 00, 01, 10, 11, 010101, 1010,……..}

Note: Note that an infinite number of strings can be generated from Σ and once the string is generated, it has finite number of symbols in it and has a definite sequence.

Notations used: Normal notations used in this subject are shown below:

- The symbol ε is used to denote an empty string.
- The lowercase letters a, b, c, etc along with the symbols such as +,-,(,),{, and on are used to denote the symbols in Σ
- The lowercase letters such as u, v, w, x, y z are normally used to indicate strings. For example, we can write

= {3, 4, 5, 6}

1.2 Basic notations and terminologies used in FAFL

Before we see the definition of Finite Automata and Formal Languages (**FAFL**), let us have familiarity with basic notations and terminologies used in this subject.

1.2.1 Alphabet

First, let us "Define an alphabet with example"

Definition: A language consists of various symbols from which the words, statements etc., can be obtained. These symbols are called alphabets. Formally, an alphabet is defined as a finite non-empty set of symbols. The symbol Σ denotes the set of alphabets of a language.

Example 1: The alphabets of C language has the letters from A to Z, a to z, digits from 0 to 9, symbols such as +, -, *, /, (,), {, } , etc. and is denoted by

$$\Sigma = \{a, b, \ldots.z, A, B, \ldots Z, 0, 1, \ldots.9, \#, (,), \{, \}, <, >, !, [,], \ldots\ldots\}$$

Example 2: The machine Language is made up of only 0's and 1's and so, the alphabets of machine language is denoted by

$$\Sigma = \{0, 1\}$$

2.4 Difference between DFA, NFA and ε-NFA

Now, let us see "What are the difference between DFA, NFA and ε-NFA?" Strictly speaking the difference between DFA and NFA lies only in the definition of δ. Using this difference some more points can be derived and can be written as shown:

DFA	NFA	ε-NFA
The DFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where • Q is set of finite states • Σ is set of input alphabets • $\delta : Q \times \Sigma$ to Q • q_0 is the start state • $F \subseteq Q$ is set of final states	An NFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where • Q is set of finite states • Σ is set of input alphabets • $\delta : Q \times \Sigma$ to 2^Q • q_0 is the start state • $F \subseteq Q$ is set of final states	An ε-NFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where • Q is set of finite states • Σ is set of input alphabets • $\delta : Q \times (\Sigma \cup \varepsilon)$ to 2^Q • q_0 is the start state • $F \subseteq Q$ is set of final states
There can be zero or one transition from a state on an input symbol	There can be zero, one or more transitions from a state on an input symbol	There can be zero, one or more transitions from a state with or without giving any input
More number of transitions	Less number of transitions	Relatively more transitions when compared with NFA
Difficult to construct	Easy to construct	Easy to construct using regular expressions
Less powerful since at any point of time it will be in only one state	More powerful than DFA since at any point of time it will be in more than one state	More powerful than NFA since at any point of time it will be in more than one state with or without giving any input

Induction: If w has only 1's the machine will be in state A. So, w1 does not contain the substring 00 and hence the string is accepted by DFA.

If w ends with 0, the machine goes to state B and hence string ends with one 0. But, on 1 the machine enters into state A and thus the substring 00 is not accepted.

1.7 Applications of Finite Automata

Now, let us see "Why to study finite automata? or What are applications of finite automata?" [VTU-JULY 2007] Some of the applications where automata play an important role are shown below:

- Design of digital circuits: The FA is used during designing and checking the behavior of the digital circuits using software. The FA is very useful in hardware design such as circuit verification, in design of automatic traffic signals etc.
- Compiler construction: Used in the design of *lexical analyzer* (the first phase of compiler design) which breaks the input text into various units such as identifiers, keywords, punctuation etc.
- String matching: In designing a software for identifying the words, phrases and other patterns in large bodies of text (such as collection of web pages).
- String processing: To write software for processing the natural language (Ex: Speech processing). Large natural vocabularies can be described which includes the applications such as spelling checkers and advisers, multi-language dictionaries, indenting the documents etc.
- Software design: In building the software to verify the systems having finite number of states (for example, communication protocols in computer networks)
- Other applications: The FA are used in variety of applications in Artificial intelligence and knowledge engineering, in game theory and games, computer graphics, linguistics etc.,

SR.NO.	DFA	NFA
1	DFA stands for Deterministic Finite Automata.	NFA stands for Nondeterministic Finite Automata.
2	For each symbolic representation of the alphabet, there is only one state transition in DFA.	No need to specify how does the NFA react according to some symbol.
3	DFA cannot use Empty String transition.	NFA can use Empty String transition.
4	DFA can be understood as one machine.	NFA can be understood as multiple little machines computing at the same time.
5	In DFA, the next possible state is distinctly set.	In NFA, each pair of state and input symbol can have many possible next states.
6	DFA is more difficult to construct.	NFA is easier to construct.
7	DFA rejects the string in case it terminates in a state that is different from the accepting state.	NFA rejects the string in the event of all branches dying or refusing the string.
8	Time needed for executing an input string is less.	Time needed for executing an input string is more.
9	All DFA are NFA.	Not all NFA are DFA.
10	DFA requires more space.	NFA requires less space then DFA.