Unit-I: Finite Automata and Regular Languages

Motivation for studying theory of computation

Automata theory (also known as **Theory Of Computation**) is a theoretical branch of Computer Science and Mathematics, which mainly deals with the logic of computation with respect to simple machines, referred to as automata.

Automata* enables the scientists to understand how machines compute the functions and solve problems. The main motivation behind developing Automata Theory was to develop methods to describe and analyse the dynamic behavior of discrete systems.

Automata is originated from the word “Automaton” which is closely related to “Automation”.

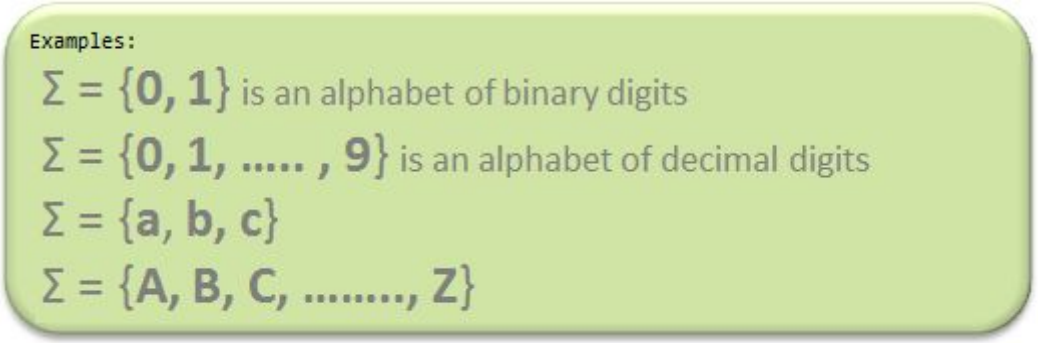
Formal grammar

In formal language theory, a **grammar** (when the context is not given, often called a **formal grammar** for clarity) describes how to form strings from a language's alphabet that are valid according to the language's syntax. A grammar does not describe the meaning of the strings or what can be done with them in whatever context—only their form. A formal grammar is defined as a set of production rules for strings in a formal language.

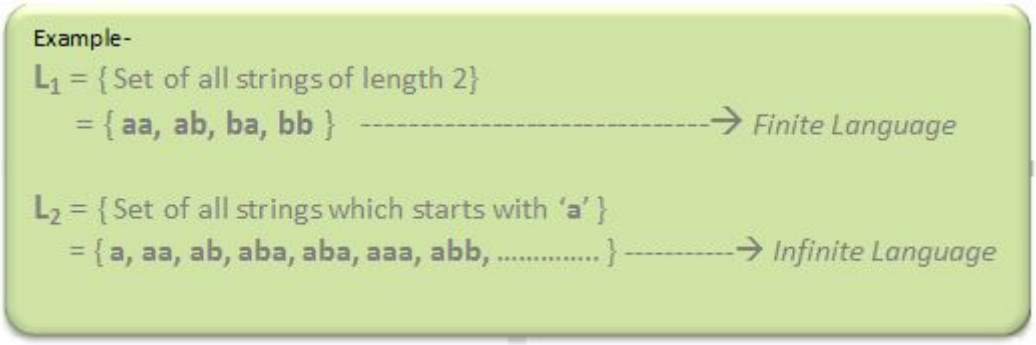
Now, let’s understand the basic terminologies, which are important and frequently used in Theory of Computation.

- **Symbol:** Symbol is the smallest building block, which can be any alphabet, letter or any picture.

Alphabets (**Σ**)**:** Alphabets are set of symbols, which are always ***finite***.



String: String is a ***finite*** sequence of symbols from some alphabet. String is generally denoted as ***w*** and length of a string is denoted as **|*w*|**.
Empty string is the string with zero occurrence of symbols, represented as ε.
Language: A language is a *set of strings*, chosen from some Σ* or we can say- 'A language is a subset of Σ* '. A language which can be formed over ' Σ ' can be Finite or Infinite.



Powers of ' Σ ' :
Say Σ = {a,b} then
Σ^0 = Set of all strings over Σ of length 0. {ε}
Σ^1 = Set of all strings over Σ of length 1. {a, b}
Σ^2 = Set of all strings over Σ of length 2. {aa, ab, ba, bb}
i.e. $|\Sigma^2|$= 4 and Similarly, $|\Sigma^3|$ = 8

Σ^* is a Universal Set.
$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2$ .....
= {ε} U {a, b} U {aa, ab, ba, bb}
= ..... //**infinite language**.

Kleene's Closure

Youtube: https://www.youtube.com/watch?v=Lp_FFsY5CLw

- **Definition** − The set $\sum^+$ is the infinite set of all possible strings of all possible lengths over ∑ excluding λ.
- **Representation** − $\sum^+ = \sum_1 \cup \sum_2 \cup \sum_3 \cup$…….
  $\sum^+ = \sum^* - \{ \lambda \}$
- **Example** − If ∑ = { a, b } , $\sum^+$ = { a, b, aa, ab, ba, bb,……..}

Regular Expressions and Regular languages

Yoututbe Series: https://www.youtube.com/watch?v=ksRzqKz2LQI
GeeksforGeeks:
https://www.geeksforgeeks.org/regular-expressions-regular-grammar-and-regular-languages

Regular Expressions are used to denote regular languages. An expression is regular if:

- ϕ is a regular expression for regular language ϕ.
- ε is a regular expression for regular language {ε}.
- If a ∈ Σ (Σ represents the input alphabet), a is regular expression with language {a}.
- If a and b are regular expression, a + b is also a regular expression with language {a,b}.
- If a and b are regular expression, ab (concatenation of a and b) is also regular.
- If a is regular expression, a* (0 or more times a) is also regular.

Regular Grammar : A grammar is regular if it has rules of form A -> a or A -> aB or A -> ε where ε is a special symbol called NULL.

Regular Languages : A language is regular if it can be expressed in terms of regular expression.

Closure Properties of Regular Languages

Union : If L1 and If L2 are two regular languages, their union L1 ∪ L2 will also be regular. For example, L1 = $\{a^n \mid n \geq 0\}$ and L2 = $\{b^n \mid n \geq 0\}$
L3 = L1 ∪ L2 = $\{a^n \cup b^n \mid n \geq 0\}$ is also regular.
Intersection : If L1 and If L2 are two regular languages, their intersection L1 ∩ L2 will also be regular. For example,
L1= $\{a^m b^n \mid n \geq 0$ and $m \geq 0\}$ and L2= $\{a^m b^n \cup b^n a^m \mid n \geq 0$ and $m \geq 0\}$
L3 = L1 ∩ L2 = $\{a^m b^n \mid n \geq 0$ and $m \geq 0\}$ is also regular.
Concatenation : If L1 and If L2 are two regular languages, their concatenation L1.L2 will also be regular. For example,
L1 = $\{a^n \mid n \geq 0\}$ and L2 = $\{b^n \mid n \geq 0\}$
L3 = L1.L2 = $\{a^m . b^n \mid m \geq 0$ and $n \geq 0\}$ is also regular.
Kleene Closure : If L1 is a regular language, its Kleene closure L1* will also be regular. For example,
L1 = (a ∪ b)
L1* = (a ∪ b)*
Complement : If L(G) is regular language, its complement L'(G) will also be regular. Complement of a language can be found by subtracting strings which are in L(G) from all possible strings. For example,
L(G) = $\{a^n \mid n > 3\}$
L'(G) = $\{a^n \mid n <= 3\}$

Note : Two regular expressions are equivalent if languages generated by them are same. For example, (a+b*)* and (a+b)* generate same language. Every string which is generated by (a+b*)* is also generated by (a+b)* and vice versa.

Finite Automata(FA)

It is the simplest machine to recognize patterns. A Finite Automata consists of the following :

Q : Finite set of states.
∑ : set of Input Symbols.
q : Initial state.
F : set of Final States.
δ : Transition Function.

Formal specification of machine is

{ Q, ∑, q, F, δ }.

Mealy machine and Moore machine

Prerequisite – Mealy and Moore Machines

Moore Machines: Moore machines are finite state machines with output value and its output depends only on present state. It can be defined as (Q, q0, ∑, O, δ, λ) where:

- Q is finite set of states.
- q0 is the initial state.
- ∑ is the input alphabet.
- O is the output alphabet.
- δ is transition function which maps Q×∑ → Q.
- λ is the output function which maps Q → O.

Mealy Machines: Mealy machines are also finite state machines with output value and its output depends on present state and current input symbol. It can be defined as (Q, q0, ∑, O, δ, λ') where:

- Q is finite set of states.
- q0 is the initial state.
- ∑ is the input alphabet.

- O is the output alphabet.
- δ is transition function which maps Q×∑ → Q.
- 'λ' is the output function which maps Q×∑→ O.

Unit-II: Nondeterminism and Minimization

In automata theory, a finite-state machine is called a deterministic finite automaton (DFA), if each of its transitions is *uniquely* determined by its source state and input symbol, and reading an input symbol is required for each state transition.

A nondeterministic finite automaton (NFA), or nondeterministic finite-state machine, does not need to obey these restrictions. In particular, every DFA is also an NFA. Sometimes the term NFA is used in a narrower sense, referring to an NFA that is *not* a DFA

Acceptance condition: *Acceptance of finite words*: Same as described in the informal definition above.

- *Acceptance of infinite words*: an *omega automaton* cannot have final states, as infinite words never terminate. Rather, acceptance of the word is decided by looking at the infinite sequence of visited states during the run.
- *Probabilistic acceptance*: An automaton need not strictly accept or reject an input. It may accept the input with some probability between zero and one. For example, quantum finite automaton, geometric automaton and metric automaton have probabilistic acceptance.

Kleene's Theorem in TOC:Youtube Series:

https://www.youtube.com/watch?v=qNqjkjIN4oo&list=PLrb70iTVZjZNkiZE4yfYn8Bf0t-AYXTkD

Myhill-Nerode relations: Youtube Series: https://www.youtube.com/watch?v=ZZg-Z2WMfow
Minimization Algorithm: Youtube Series: https://www.youtube.com/watch?v=uiFoT_N2J14
Non-Regular languages, Pumping Lemma for regular languages:
https://www.cs.wcupa.edu/rkline/fcs/re-pump.html
Youtube Series: https://www.youtube.com/watch?v=XeT8sJxU1ak

Made by @imnotshub