Theory of Computation

Section (1)

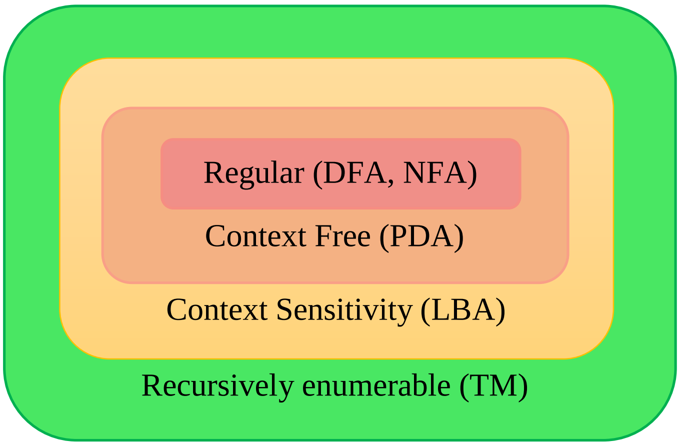
Theory'21

The Basic Concepts

- **Language:** a collection of sentences (word) of finite length all constructed from alphabet or symbols
  Example: An alphabet is a set of character {a, b}
  L1= {ab, ba, abab, aabb}
  L2= {aaab, bbaa, abb, bbba}
  L3= {λ, abbbaa, abababb}

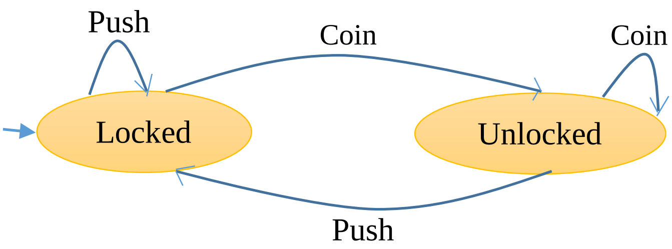
- **Grammar:** set of the rules that describe all possible string in the given language G= (V, T, S, P)
  V: variable or Non terminal alphabets: S, A, B
  T: terminal alphabets: {a, b}
  S: start symbol: S$\longrightarrow$
  P: production rule
  Example : L= $\{(a\ b)^n \ \ n > 0\}$
  L= {ab, abab, ababab, abababab, …...}
  S $\longrightarrow$ abS
  S $\longrightarrow$ λ

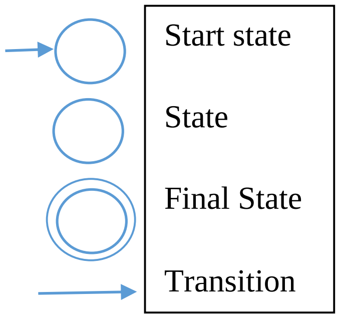
- **Automaton:** It is defined as self-operating machine that considered as an abstract model for digital computers with limited memory, we can say it simulate the parts of computer, to theory can be solve a model of computation by algorithm, is applied one of three fields:
  1-Automata theory
  2-Computability theory
  3-Computational complex theory



Chomsky Hierarchy

Simple Example





Context Free Grammar (CFG)

The Regular Grammar ⊆ CFG

Example: L= $\{a^n b^{n+1}\ ; n \geq 0\}$, Find the context free grammar ?

L= {b, abb, aabbb, aaabbbb, …...}

S $\rightarrow$ Ab

A $\rightarrow$ aAb

A $\rightarrow$ λ

Example: L= $\{w = n_a(w) = n_b(w)\}$, Find the context free grammar ?

L= {ab, ba, abab, baba, aabb, bbbaaa, …….}

S $\rightarrow$ aSb

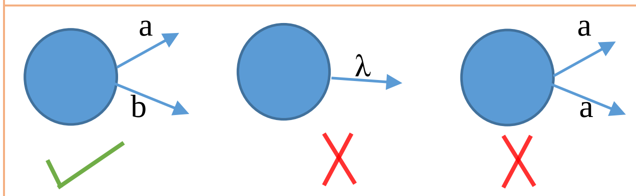
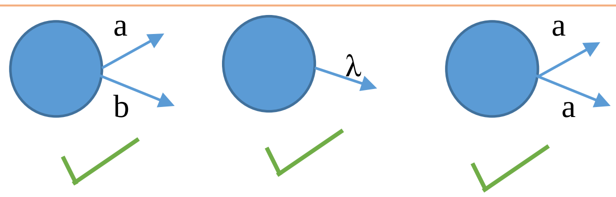
S $\rightarrow$ bSa

S $\rightarrow$ λ

Deterministic Finite Automata (DFA) & non- Deterministic Finite Automata (NFA)

M=(Q, ∑, δ, q_0, F)

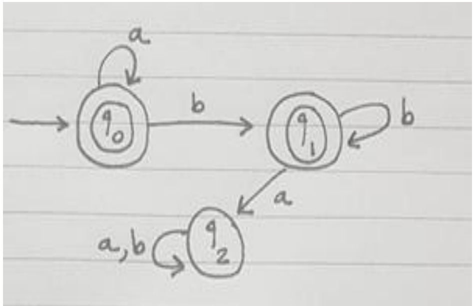
Q:States, ∑: Alphabets, δ :Transitions, q_0:Initaial state , F: Final state

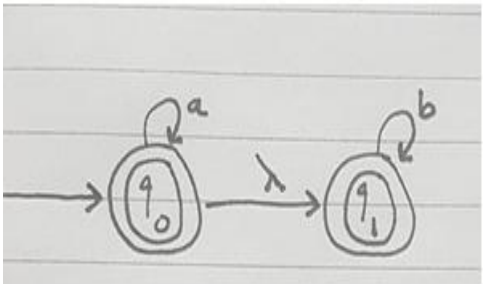
DFA	NFA
Each state has transition for all input (Alphabet)	Each state can has 2 transition that has same input (Alphabet)
Can't have λ transition	Can have λ transition
Each transition is uniquely determined	
	

DFA & NFA Examples

What's the grammar, DFA & NFA of the following languages?

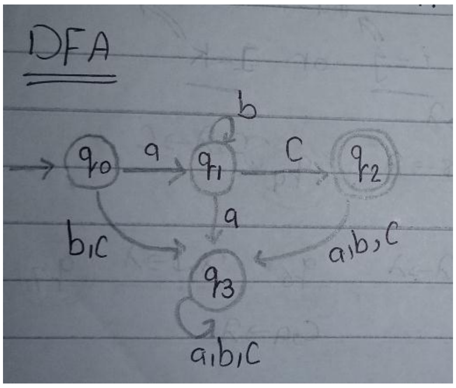
1. **L = {$a^n b^m$, n, m ≥ 0}**
   a. S → AB
   A → aA | λ
   B → bB | λ

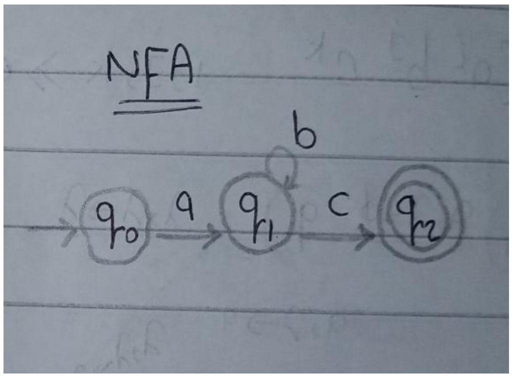
   b. 

   c. 

2. **L = {a b^n c, n ≥ 0}**

   a. S → aBc
   B → bB | λ

   b. 

   c. 

3. **L = {$a^n b^n$, n ≥ 0}**
   a. S → aSb | λ
   Using any memory space like stack may be helpful to save information.
   The above language is a non-regular language since this can't be described by neither DFA nor NFA.