FORMAL LANGUAGE

Formal Languages

Definition – A formal language consists of words whose letters are taken from an alphabet and are well-formed according to a specific set of rules.

This is a car

- Sentences
- Words / Strings
- Letters
- Alphabets
- Grammar

Mathematically Defined Formal Languages

Languages , L

- Symbols　　a, b, 1, 2 , Z,0,c
- Alphabet (Σ)
  - Finite and nonempty set of symbols

    {a,b,c}　　{0,1}
- Strings
  - Finite set of sequence from Σ

    aabbc abcccaaabb　　0110110 01000000001　　Є

Are these alphabets ?

A = { 1,2,d,v}

X={ a, b, …}

Y={}

Σ = { x,y,z}

Identify the symbols in the above alphabets !!!

Length of a string

Σ ={a,b}

Length of a string, n → |n|

aaabbaa

abb

a

Powers of Σ

Σ ={a,b}

Σ^k = {w| w is a string of length k , k>=1}

Σ^1 Set of all strings over Σ^k with a length of 1

$\Sigma^2 = \Sigma\,\Sigma$ Set of all strings over Σ^k with a length of 2

Σ^n

Σ^0

Σ ={a,b}

How many strings are possible with 2 symbols, of length 2 ?

How many strings are possible with 2 symbols, of length n ?

How many strings are possible with | Σ | symbols, of length n ?

Language (L)

Collection of strings

L_1 = Set of all strings with length 2

L_2= Set of all strings with length 3

L_3= Set of all strings starting with a

Σ ={a,b}

Finite languages
Infinite Languages

L is finite	L is infinite
Σ = {a,b}	Σ = {a,b}
L1 = Set of all strings with length 2	L2 = Set of all strings starting with a
{aa , ab , ba , bb}	{a, aa , ab, aba, abb , abba , abadb , …}
Check whether “ab” is valid	Check whether “ab” is valid
Check whether “bbab” is valid	Check whether “bbab” is valid

Kleene Closure

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots \cup \Sigma^n$

- Null allowed
- Universal Set
- Infinite too !!!

Positive Closure

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots \cup \Sigma^n$

- Null not allowed

Application of Formal Languages

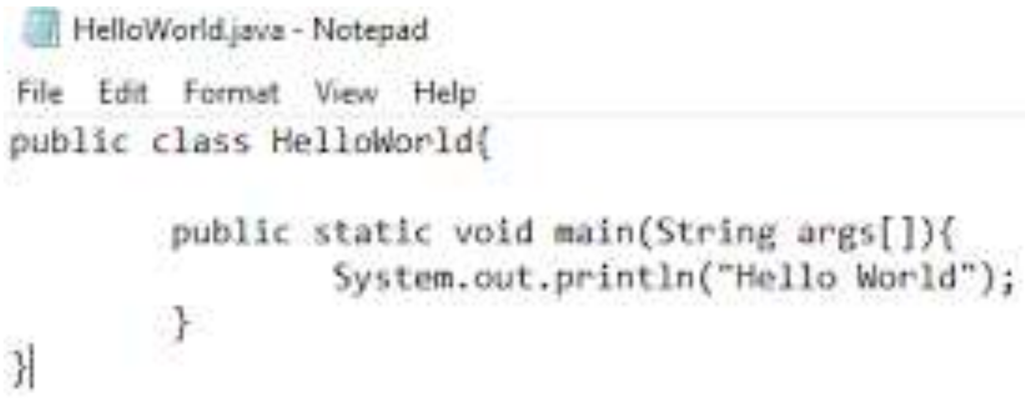
Java Programming

Σ

String –> Program

Language → All programs → **Infinite**

Is a given Program, P, present in the Language, L ????



L is finite	L is infinite
Σ = {a,b}	Σ = {a,b}
L1 = Set of all strings with length 2	L1 = Set of all strings starting with a
{aa , ab , ba , bb}	{a, aa , ab, aba, abb , abba , abadb , …}
Check whether “ab” is valid	Check whether “ab” is valid
Check whether “bbab” is valid	Check whether “bbab” is valid

Theory of Automata

Finite Automaton

Finite Automata(FA) is the simplest machine to recognize patterns

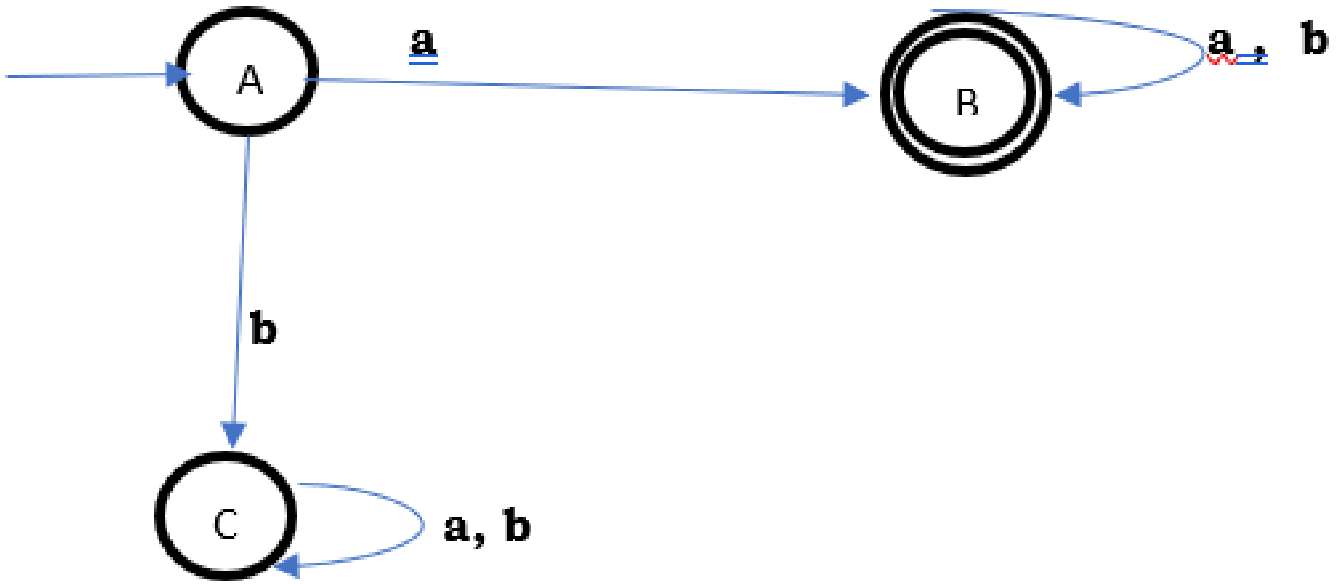
Basically it is an abstract model of a digital computer.

Used to recognize patterns.

How does a FA work ?

L is infinite
Σ = {a,b}
L1 = Set of all strings starting with a
{a, aa , ab, aba, abb , abba , abadb , …}
Check whether “abba” is valid
Check whether “bbab” is valid

State Transition Diagram



How does a FA work

It takes the string of symbol as input and changes its state accordingly.

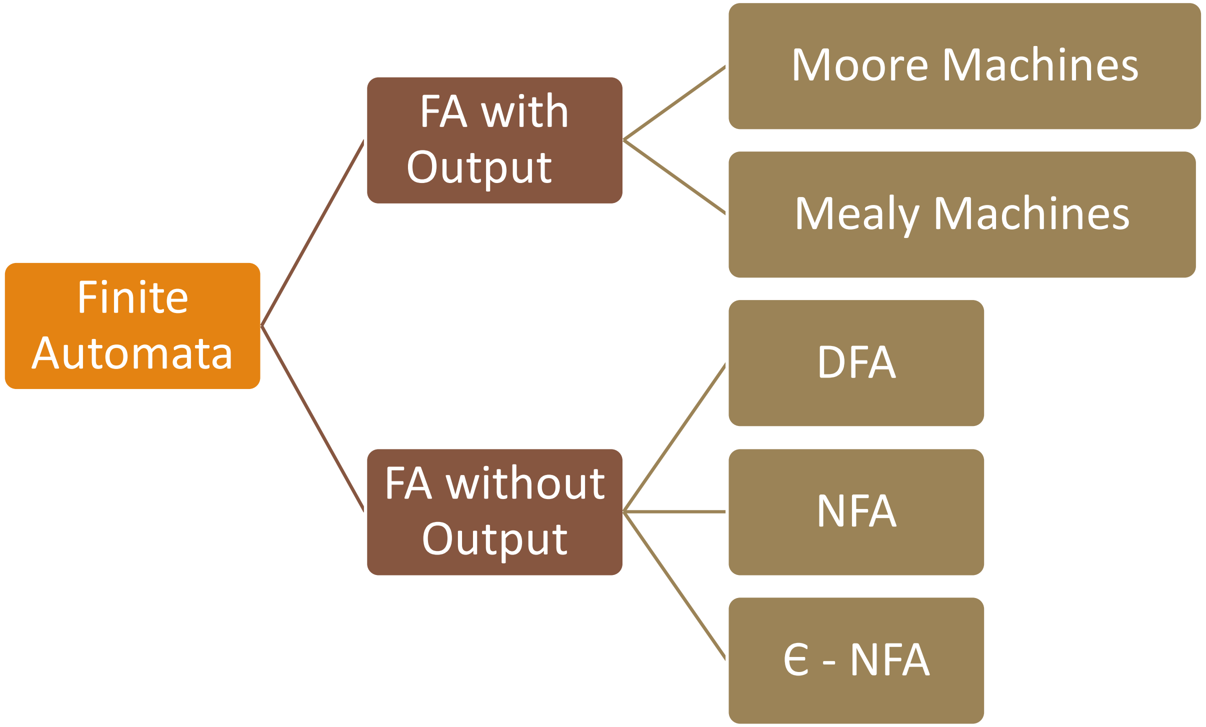
When the desired symbol is found, then the transition occurs.

At the time of transition, the automata can either move to the next state or stay in the same state.

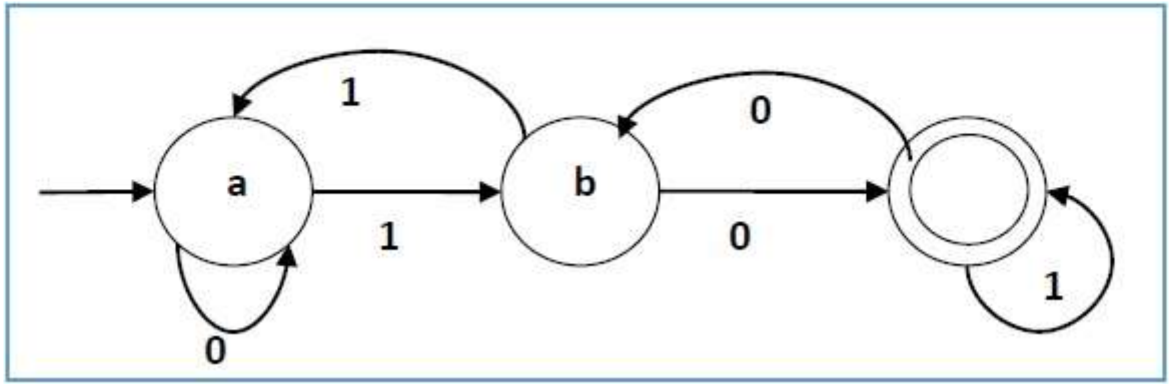
Finite automata have two states, **Accept state** or **Reject state**.

When the input string is processed successfully, and the automata reached its final state, then it will accept.

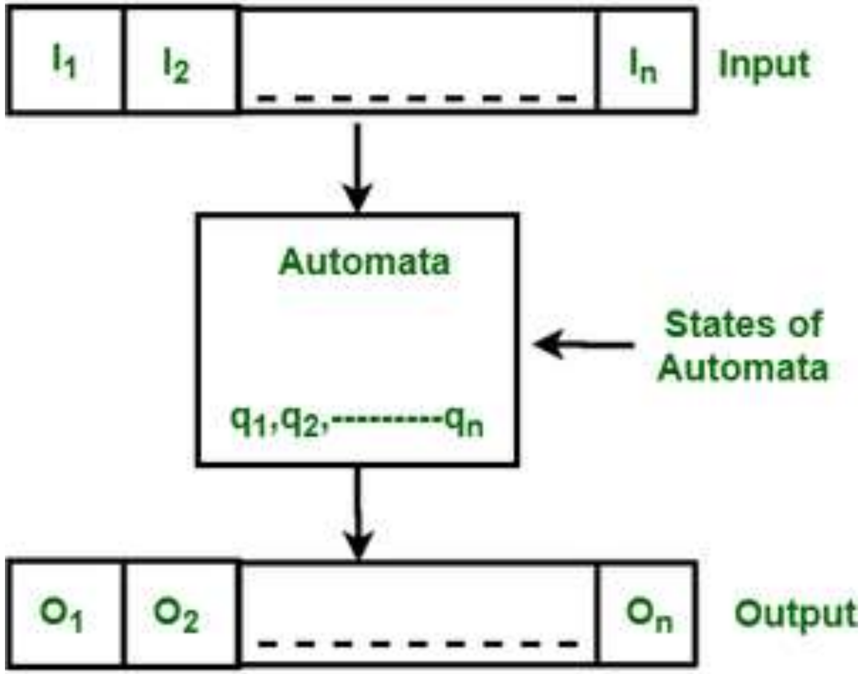
Families of Automata



DFA
Deterministic Finite Automata



Structure of DFA - Quintuple



1.Input
2.Output
3.States of automata
4.State relation
5.Output relation

Formal specification of machine is $\{ Q, \Sigma, \delta, q_0, F\}$.

Q : Finite set of states.

Σ : set of Input Symbols.

δ : Transition Function.

q_0 : Initial state.

F : set of Final States.

Formal specification of machine is $\{ Q, \Sigma, \delta, q_0, F\}$.

Q : Finite set of states.

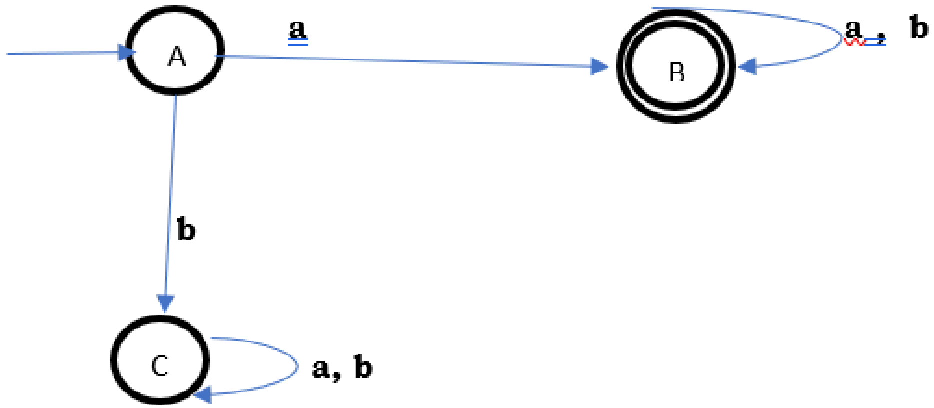
Σ : set of Input Symbols.

δ : Transition Function.

q_0 : Initial state.

F : set of Final States.

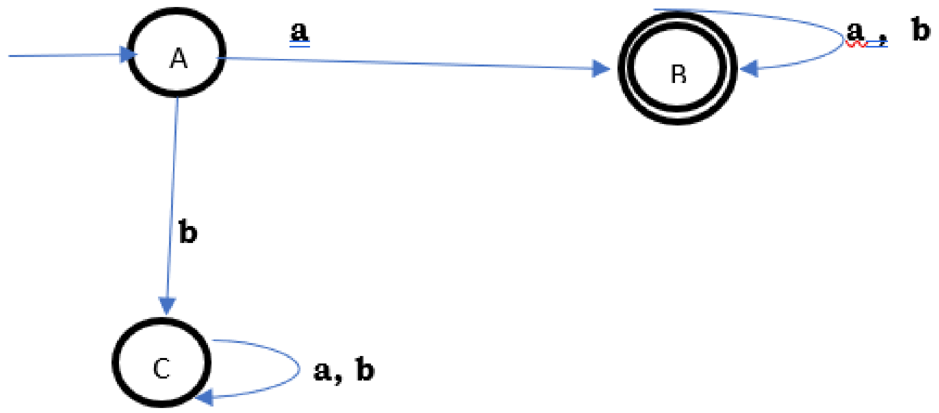
State Transition Diagram



1. Define the DFA
2. Check Validity

L is infinite
Σ = {a,b}
L1 = Set of all strings starting with a
{a, aa , ab, aba, abb , abba , abadb , …}
Check whether "abba" is valid
Check whether "bbab" is valid

State Transition Diagram



1. Construct a DFA that accepts set of all strings over

Σ = {a,b} of length 2.

1. Construct a DFA that accepts set of all strings over $\Sigma = \{a,b\}$ of length 2.

$\Sigma = \{a,b\}$

L={w| w is a string |w|=2}

L={aa, ab , ba , bb}

1. Construct a DFA that accepts set of all strings over $\Sigma = \{a,b\}$ of length 2.

$\Sigma = \{a,b\}$

L={w| w is a string |w|=2}

L={aa, ab , ba , bb}

Take the smallest string, and construct a skeleton automata
Check all possibilities by scanning the strings
Reach a final state
Check for a trap state
Construct a State Transition Table

String acceptance by a DFA

Start from Initial state

Scan the entire string

Reach the final state

Language acceptance by a DFA

Accept all strings in the language

Reject all strings not in the language

String acceptance by a DFA

Start from Initial state

Scan the entire string

Reach the final state

2. Construct a DFA that accepts set of all strings over $\Sigma = \{a,b\}$ of length >=2.

L= {aa, ab, ba, bb, aaa, aba , …}

Infinite → DFA can be defined

3. Construct a DFA that accepts set of all strings over **Σ = {a,b} of length <=2.**

L = {Є, a, b, aa,ab,ba, bb}

Relationship between string length & no: of states

Length of string	|w|=2	|w|>=2	|w|<=2
Number of states	4	3	4

Relationship between string length & no: of states

<table>
<tr><th>Length of string</th><th>|w|=2</th><th>|w|>=2</th><th>|w|<=2</th></tr>
<tr><td>Number of states</td><td>4</td><td>3</td><td>4</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>Length of string</td><td>|w|=n</td><td>|w|>=n</td><td>|w|<=n</td></tr>
<tr><td>Number of states</td><td>n+2</td><td>n+1</td><td>n+2</td></tr>
</table>

4. Construct a DFA that accepts set of all strings over $\Sigma = \{0,1\}$ which ends with 0

L= {0 , 00, 10 , 000, 1010, 110 ,... }

5. Construct a DFA over $\Sigma = \{0,1\}$ which accepts 101

$L = \{ 101 \}$

6. Construct a DFA over $\Sigma = \{0,1\}$ which accepts strings with three consecutive 1's

$L = \{ 111, 00111, 1110, 1110111, \ldots \}$

NFA – Non Deterministic Finite Automata

DFA	Vs	NFA
On an input w,		
q1 → q2		q1→q2
		q1→q3
		…
		q1→qn
Ends in a single state		Can end in more than one state

Need for NFA

Exhaustive searching

Backtracking

Formal specification of machine is $\{Q, \Sigma, \delta, q_0, F\}$.

Q : Finite set of states.

Σ : set of Input Symbols.

δ : Transition Function.

q_0 : Initial state.

F : set of Final States.

1. Construct an NFA that accepts set of all very string ending with ‘a’ over **Σ = {a,b}**

L ={ a,aa, ba, bba, ababbaaa, …}

2. L= {All string starting with ‘a’}

Σ = {a,b}

---

3. L= {All strings contain ‘a’}

Σ = {a,b}

4. L= {Every string starts with ‘ab’}

Σ = {a,b}

Equivalence of DFA and NFA

DFA is already NFA

NFA←→DFA

Principle of Subset Extraction

1. L= {Every string starts with ‘a’}

$\Sigma = \{a,b\}$

2. L= {Every string ends with ‘a’}

$\Sigma = \{a,b\}$

3. L= {Every string’s second symbol is ‘a’}

$\Sigma = \{a,b\}$

Regular Expressions

A sequence of characters that define a search pattern

Most effective way to represent any language.

Language accepted by finite automata can be easily described by simple expressions called Regular Expressions.

Regular Expressions are used to denote regular languages.

RE are mathematical expressions

Operations of Regular Expressions

Union (+)

Concatenate (.)

Kleene Closure (*)

Properties of Regular Expressions

1. Primitive RE
    - If Φ is the empty set, it is represented as {}
    - If Є is the empty string, it is represented as {Є}
    - If a Є Σ , it is represented as {a}
2. If r1 and r2 are primitive RE
    - r1+r2 is an RE
    - r1. r2 is an RE
    - r1*is an RE
3. If a and b are used several times, the result is also an RE

RE	Language
Φ	{}
Є	{Є}
a	{a}
a*	{Є,a ,aa, aaa , …}
a+	{a ,aa, aaa , …}
(a+b)*	{Є,a, b, aa, ab, ba, bb, …)

Examples of RE $\Sigma = \{a,b\}$

Language	Strings in the language	RE
{length of 2}		
{length atleast 2}		
{length atmost 2}		
{Even length string}		
{odd Length string}		
Length of string should be divisible by 3		
Length of string should be 2 mod 3		

Examples of RE

Σ ={a,b}

Language	Strings in the language	RE
Exactly 2 a’s		
Atleast 2 a’s		
Atmost 2 a’s		
Even number of a’s		
Odd number of a’s		
Starts with a		
Ends with a		
Containing a		
Start and end with a		
Start and end with different symbols		
Start and end with same symbols		

Regular Grammar

Grammar – Description of a language by rules.

A grammar **G** can be formally written as a 4-tuple (V, T, S, P) where −

- **V** is a set of variables or non-terminal symbols / vertices.
- **T** is a set of Terminal symbols.
- **P** is Production rules for Terminals and Non-terminals.
- **S** is a special variable called the Start symbol

Derivation

Derivation→ Getting a string from grammar, starting from S

S → aSB

S → aB

B → b

Derivation of aabb

2. Set of all strings of length 2

3. $a^n/n>=0$

4. Set of all strings over a , b

5. Set of all strings of at least length 2

6. Set of all strings of atmost length 2

7. Set of all strings start with ‘a’ and end with ‘b’

8. Set of all strings starting and ending with a different character

9. Set of all strings starting and ending with same character

10. $a^n b^n / n >= 1$

11. Even length strings

12. $a^n b^m / n >= 1$

13. $a^n b^n c^m / n >= 1$

14. $a^n c^m b^n / n >= 1$

Types of Languages

Regular Languages

- A language is regular if it **can be expressed** in terms of **regular expression**
- Finite State machines will recognize Regular Languages.

Non - Regular Languages

- A language is non regular if it **cannot be expressed** in terms of **regular expression**

Properties of Regular Languages

1. **Union :** If L1 and If L2 are two regular languages, their union L1 ∪ L2 will also be regular.

   Eg: $L1 = \{a^n \mid n \geq 0\}$ and $L2 = \{b^n \mid n \geq 0\}$
   $L3 = L1 \cup L2 = \{a^n \cup b^n \mid n \geq 0\}$ is also regular.

2. **Intersection :** If L1 and If L2 are two regular languages, their intersection L1 ∩ L2 will also be regular.

   Eg: $L1= \{a^m b^n \mid n \geq 0 \text{ and } m \geq 0\}$ and $L2= \{a^m b^n \cup b^n a^m \mid n \geq 0 \text{ and } m \geq 0\}$
   $L3 = L1 \cap L2 = \{a^m b^n \mid n \geq 0 \text{ and } m \geq 0\}$ is also regular.

3. **Concatenation :** If L1 and If L2 are two regular languages, their concatenation L1.L2 will also be regular.

   Eg: $L1 = \{a^n \mid n \geq 0\}$ and $L2 = \{b^n \mid n \geq 0\}$
   $L3 = L1.L2 = \{a^m . b^n \mid m \geq 0 \text{ and } n \geq 0\}$ is also regular.

4. **Kleene Closure :** If L1 is a regular language, its Kleene closure L1* will also be regular.

Eg: $L1 = (a \cup b)$

$L1^* = (a \cup b)^*$

5. **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.

Eg: $L(G) = \{a^n \mid n > 3\}$

$L'(G) = \{a^n \mid n <= 3\}$

Non Regular Languages

Languages which are not regular

Cannot be represented and identified by a FSM

Languages which require memory

- Memory of FSM is very limited
- It cannot **store** or **count** strings

Pumping Lemma for Regular Languages

Used to prove that a language is not regular

Cannot prove that a language is regular

- If A is a regular language, then
  - there is a number p (the pumping length), and
  - if s is any string in A of length at least p, then
    - s can be divided into three pieces, $s = xyz$, that satisfy the following conditions:

      1. $xy^iz \in A$, for each $i \geq 0$

      2. $|y| > 0$

      3. $|xy| \leq p$

Steps to Perform Pumping Lemma (Proof by Contradiction)

(We prove using Contradiction)

- -> Assume that A is Regular
- -> It has to have a Pumping Length (say P)
- -> All strings longer than P can be pumped $|S| \geq P$
- -> Now find a string 'S' in A such that $|S| \geq P$
- -> Divide S into x y z
- -> Show that $x y^i z \notin A$ for some i
- -> Then consider all ways that S can be divided into x y z
- -> Show that none of these can satisfy all the 3 pumping conditions at the same time
- -> S cannot be Pumped == CONTRADICTION

1. P. T $A=\{a^n b^n \mid n>=0\}$ is not regular

2. P. T $A=\{\mathbf{yy} \mid y \in (0,1)^*\}$ is not regular

Tools to prove that a Language is Regular:

DFA
NFA
GNFA
Regular Expression

Tools to prove that a Language is Nonregular:

The Pumping Lemma