

# FINITE AUTOMATA

## Basic of Toc

1. Symbol
2. Alphabet
3. String
4. Language

Symbol :

\* It is anything.

\* It is one length.

Symbol

o

gate

$|0|=1$

$|gate|=1$

1. English Language
2. Hindi Language
3. Decimal Language
4. Binary Language
5. C Language

Symbol,

<sup>letter</sup> a, b, ..., z

<sup>letter</sup> अ, आ, ..., त

<sup>digit</sup> 0, 1, 2, ...,  $\infty$

<sup>bit</sup> 0, 1

<sup>char</sup> 0...9, a, b, ..., z, +, -, ...

Alphabet :

\* It is a set.

Collection of Objects

well defined  $\rightarrow$  distinct,

unordered.

Alphabets in Toc

$\Sigma \rightarrow$  Input alphabet : Set of I/P Symbols

$\Delta \rightarrow$  Output alphabet : Set of output Symbols

$\Gamma \rightarrow$  Stack alphabet : Set of Stack Symbols

$T \rightarrow$  Tape Alphabet : Set of Tape Symbols

eg: English Alphabets  $\Sigma = \{a, b, \dots, z, A, B, \dots, Z\}$

Symbol

String :

\* It is a sequence of symbols.

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

- Zero length string:  $\emptyset$  or  $\epsilon$  empty string
- One length string: 0, 1
- Two length string: 00, 01, 10, 11
- Three length string: 000, 001, 010, 011, 100, 101, 110, 111

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

How many  $k$ -length strings?

$$\rightarrow 2^k = |\Sigma|^k$$

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

How many 100 length strings?

$$\rightarrow 3^{100}$$

Symbols

Alphabet

Strings

Languages

Set: Collection of objects

$\emptyset$

Empty Set

{ }

Set

$$|\emptyset| = 0$$

Size of  $\emptyset$

String: Sequence of symbols

$\epsilon$

Empty String

String

$$|\epsilon| = 0$$

Size of length of  $\epsilon$

Language [Set] over  $\Sigma$

\* Set

\* Set of Strings

\* Collection of Strings

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

Language over  $\Sigma$  =

Language with zero string:  $L = \emptyset = \{\}$  empty language.

Language with only 1 string:  $L_1 = \{e\}, L_2 = \{1\}, L_3 = \{0\}, L_4 = \{00\} \dots$

Language with only 2 strings:  $L_1 = \{e, 0\}, L_2 = \{0, 1\}, L_3 = \{0, \infty\} \dots$

Given  $\Sigma$ .

- No. of strings over  $\Sigma$  = Infinite
- No. of languages over  $\Sigma$  = Infinite
- No. of  $k$  length strings =  $|\Sigma|^k$
- No. of languages having 10 strings = Infinite

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

Finite Languages

$$\emptyset$$

$$\{0\}$$

$$\{0, 10\}$$

$$\{0^n 1^m | n \leq 10\}$$

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

Infinite Languages

$$\{0^n | n \geq 10\}$$

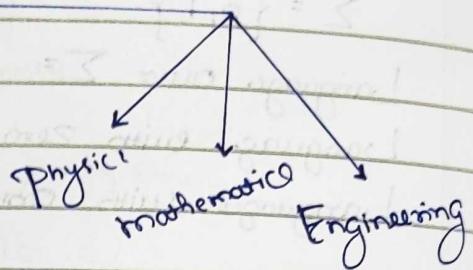
$$\{0^n 1^m | n \geq 1\}$$

form of a string.

	English	Binary N. System.	C. Language
Symbols	a, b, ..., z	0, 1	Character
Alphabets	{a, b, ..., z}	{0, 1}	Character Set.
String	words	numbers	tokens
Language	Sentences	binary language.	Programs.

## Applications of ToC :

- Compiler
- AI Applications
- Digital Circuits.



## Operations on String.

### 1. Reversal (funary)

$$w = abc$$

$$w^R = cba$$

$$|w| = |w^R|$$

### 2. Concatenation (Binary)

$$w_1 = abc$$

$$w_2 = bc$$

$$w_1.w_2 = abbc$$

$$w_2.w_1 = bcab$$

$$|w_1.w_2| = |w_2.w_1|$$

### 3. Prefix

$w = abcd$  → start is fixed

Prefixes of  $w =$

$\epsilon$   
a  
ab  
abc  
abcd

### 4. Suffix

$$w = abcd$$

Suffixes of  $w$ :

$\epsilon$ .

d

cd

bcd

abcd

### 5. SubString.

$$w = abcd$$

Substrings of  $w$  are:

$\epsilon$

4 { a      ab      abc      abc }  
       b      bc      bcd      bcd }  
       c      ca                     abcd }  
       d                               1

### Prefix ( $w$ )

$$= \{ v \mid uv = w \}$$

Beginning sequence  
of  $w$ .

### Suffix ( $w$ )

$$= \{ v \mid uv = w \}$$

Ending sequence  
of  $w$ .

### SubString ( $w$ )

$$= \{ y \mid xyz = w \}$$

$w = abcd$   
 $E. E. abcd = w$   
 $E. a. bcd = w$

\*  $w = \text{aaaa}$

Prefixes of  $w = \epsilon, a, aa, aaa, aaaa - 5$

Suffixes of  $w = \epsilon, a, aa, aaa, aaaa - 5$

Substrings of  $w = \epsilon, a, aa, aaa, aaaa - 5$

\*  $w = abcd$

Prefixes of  $w = \epsilon, a, ab, abc, abcd - 5$

Suffixes of  $w = \epsilon, d, cd, bcd, abcd - 5$

Substrings of  $w = \epsilon, a, b, c, d, ab, bc, cd, abc, bcd, abcd - 10$

How many Prefixes for  $n$ -length String?

→  $n+1$

How many Suffixes for  $n$ -length String?

→  $n+1$

How many Substrings for  $n$ -length String?

→  $n+1 \leq \text{No. of Substrings} \leq \frac{n(n+1)}{2} + 1$

If all Symbols are  
Same

If all Symbols in the String  
are distinct.

How many different length Prefixes for  $n$ -length String?

→  $n+1$

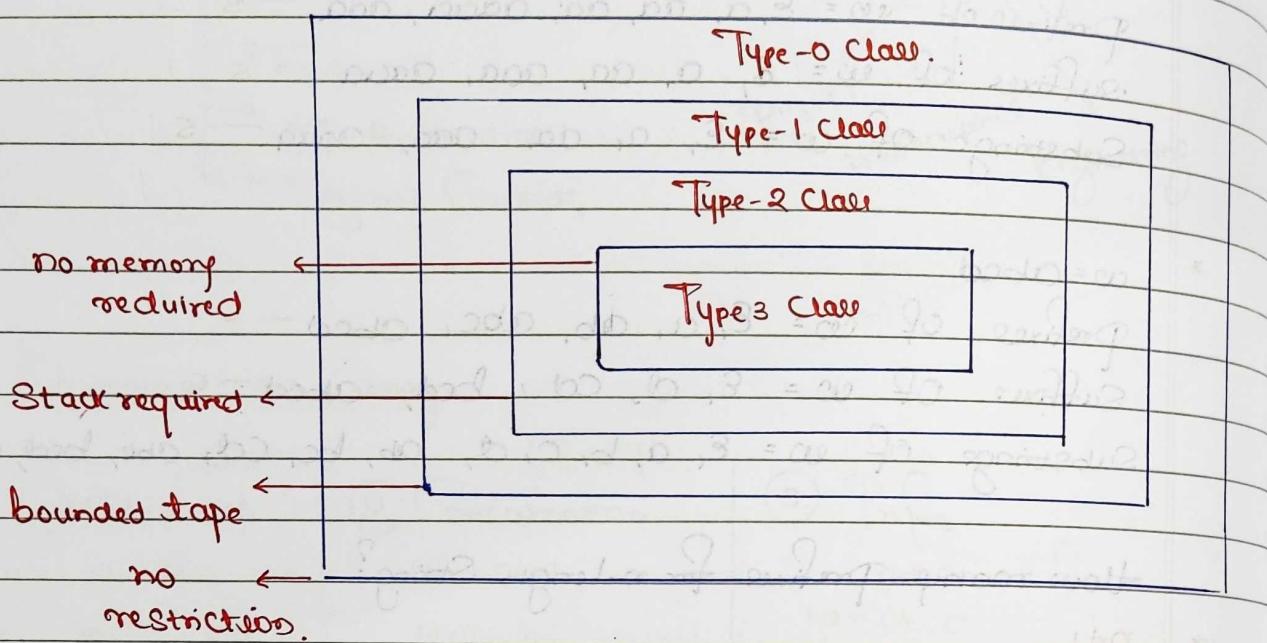
How many different length Suffixes for  $n$ -length String?

→  $n+1$

How many different length Substrings for  $n$ -length String?

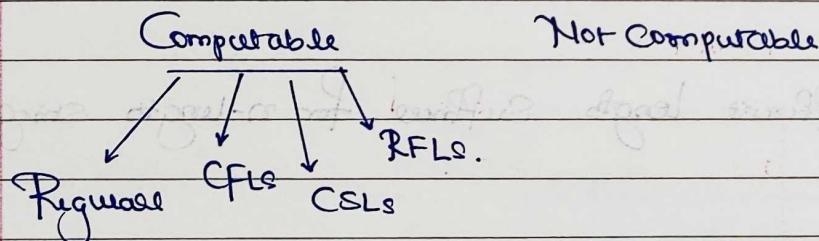
→  $n+1$

Chomsky hierarchy:  $T_3 < T_2 < T_1 < T_0$



Type 3 Class	Type 2 class	Type-1 Class	Type 0 class
↓ Set of all regular languages.	↓ Set of all context-free languages	↓ Set of all context-sensitive languages	↓ Set of all recursively enumerable languages.
Finite Automata (FA)	Push Down Automata (PDA)	Linear bound Automata (LBA)	Turing Machine (TM)

problems / Languages



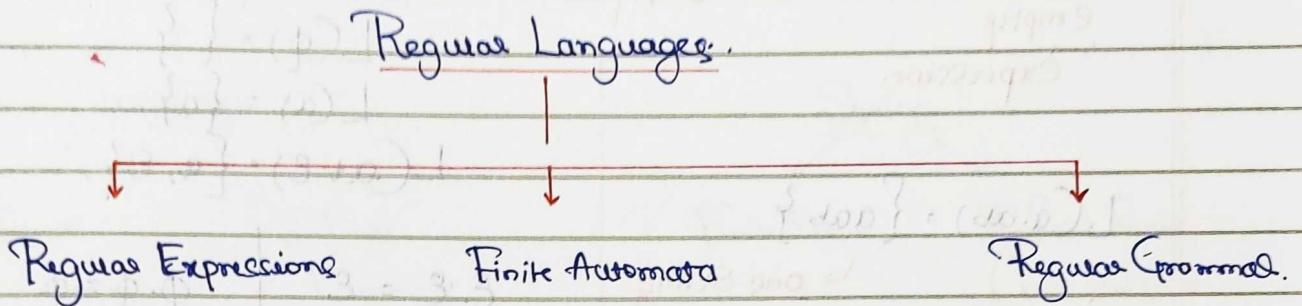
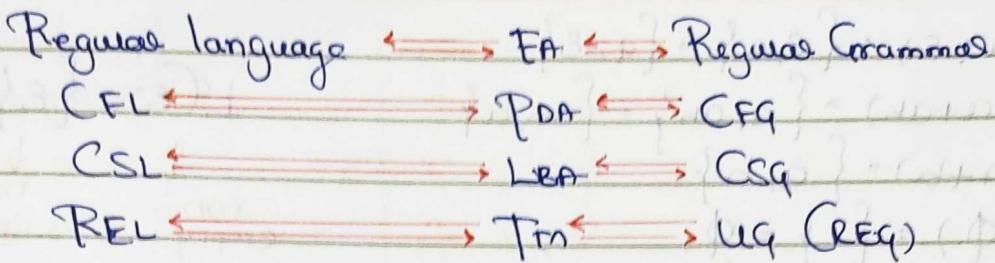
Language  
↳ set of strings

Automata  
↓ (Machine)

Grammars  
↳ Set of rules

String

Set of states & transitions



## Regular Expression:

- \* It represents a regular language.
- \* It uses four operators.

Binary	I. OR (+) II. Concatenation (.) III. Kleene Star [Kleene Closure] (*) IV. Kleene Plus [positive closure] (+)
--------	---

OR	Concatenation	Kleene Star	Kleene Plus.
$R_1 \cup R_2$	$R_1 \cdot R_2$	$R^*$	$R^+$

Either $R_1$ or $R_2$	$R_1$ followed by $R_2$	Zero or more occurrence of $R$	One or more occurrences of $R$ .
$a+b$ $\{a, b\}$	$a \cdot b$ $\{ab\}$	$a^* = a^{>0}$ $\{a^0, a^1, a^2, \dots\}$ $\{\epsilon, a, aa, aaa, \dots\}$	$a^+ = a^{>1}$ $\{a, aa, aaa, \dots\}$

$$L(a + \epsilon) = \{a, \epsilon\} = \{a, \epsilon\}$$

$$L(a + ab) = \{a, ab\} = \{ab, a\}$$

$$L(a + b) = \{a, b\}$$

$$L(\phi) = \{\} = \phi.$$

$\downarrow$   
empty  
expression

$$L(\epsilon) = \{\epsilon\}$$

$\downarrow$   
non  
empty  
expression

$\downarrow$   
nonempty  
set.

$$L(\phi) = \{\}$$

$$L(a) = \{a\}$$

$$L(a + \epsilon) = \{a, \epsilon\}.$$

$$L(a \cdot ab) = \{aab\}$$

$\downarrow$  one string

$$L(a \cdot \epsilon) = \{a\}.$$

$$\epsilon \cdot \epsilon = \epsilon$$

$$a \cdot \epsilon = a$$

$$\epsilon \cdot a = a.$$

$$\phi \cdot \phi = \phi$$

$$a \cdot \phi = \phi.$$

OR:

$$1. \epsilon + \epsilon = \epsilon$$

$$2. \phi + \phi = \phi$$

$$2. a + a = a$$

$$4. \epsilon + a = \{ \} = \epsilon + a = a + \epsilon$$

$$5. a + \epsilon = \{ \}$$

$$6. \epsilon + \phi = \epsilon$$

$$7. \phi + \epsilon = \epsilon$$

$$8. a + \phi = a$$

$$9. \phi + a = a.$$

Regular Expression  $\cong$  Set

$$\epsilon \cong \{\epsilon\}$$

$$\phi \cong \{ \} = \phi$$

$$a \cong \{a\}$$

$$R_1 + R_2 \cong L(R_1) \cup L(R_2)$$

$$R_1 \cdot R_2 \cong L(R_1) \cdot L(R_2)$$

$$\text{eg: } a + \epsilon = a + \epsilon$$

either  $a$  or  $\epsilon$ .

$$a \cdot \epsilon = a.$$

$$|a|=1$$

$$|\epsilon|=0.$$

$$L(a + \epsilon) = \{a, \epsilon\}.$$

$$a + \epsilon \neq a.$$

Concatenation.

$$1. \epsilon \cdot \epsilon = \epsilon.$$

$$3. a \cdot a = aa = a^2$$

$$2. \phi \cdot \phi = \phi.$$

$$4. \epsilon \phi = \phi.$$

5.  $\phi \cdot \mathcal{E} = \phi$

6.  $a \cdot \mathcal{E} = a$

7.  $\mathcal{E} \cdot a = a$

8.  $\phi \cdot a = \phi$

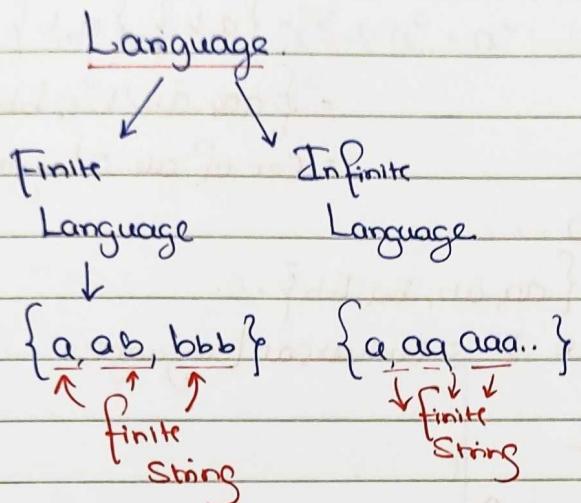
9.  $a \cdot \phi = \phi$

$\phi \cdot R = \phi$

$R \cdot \phi = \phi$

$R \cdot \mathcal{E} = R$

$\mathcal{E} \cdot R = R$



String

```

graph TD
    String[String] --> FiniteString[Finite String]
    String --> InfiniteString[Infinite String  
(not exist in  
Computation)]
  
```

$$(a+ab) \cdot (\mathcal{E} + aaa + b)$$

$$\{a, ab\} \cdot \{\mathcal{E}, aaa, b\} = \{a, ab, abb, a^4, abaaa\}$$

$$a \cdot \mathcal{E} \rightarrow a$$

$$a \cdot aaa \rightarrow aaaa = a^4$$

$$a \cdot b \rightarrow ab$$

$$ab \cdot \mathcal{E} \rightarrow ab$$

$$ab \cdot aaa \rightarrow abaaa$$

$$ab \cdot b \rightarrow abb$$

Kleene Star:

$$R^*$$

$$R^0 + R^1 + R^2 \dots$$

$$a^* = \{\mathcal{E}, a, aa, \dots\}$$

$$L(a^*) = \{\mathcal{E}, a, a^2, a^3, \dots\}$$

$$= \{a^n \mid n \geq 0\}$$

Language set.

$$1. b^* = \{\mathcal{E}, b, bb, bbb, \dots\} = \{b^n \mid n \geq 0\}$$

2.  $(ab)^* = \{ \epsilon, ab, abab, ababab \dots \} = \{ (ab)^n \mid n \geq 0 \}$
3.  $(ba)^* = \{ \epsilon, ba, baba, bababa \dots \} = \{ (ba)^n \mid n \geq 0 \}$
3.  $(aa)^* = \{ \epsilon, a^2, a^4, a^6 \dots \} = \{ a^{2n} \mid n \geq 0 \}$
5.  $(a+b)^* = \{ \epsilon, a, b, aa, ab, ba, \dots \} = \{ a, b \}^*$
6.  $(a+\epsilon)^* = \{ \epsilon, a, a^2, a^3 \dots \} = \{ a^n \mid n \geq 0 \} = a^*$
7.  $(b+\epsilon)^* = b^*$ .

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

$$= \{aa, ab, ba, bb\}$$

= Set of all 2 length strings

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots$$

$$= \{ \epsilon \} \cup \{ a, b \} \cup \{ aa, ab, ba, bb \} \dots$$

Set of all strings over  $\Sigma$  = Universal language

$a+b$

$(a+b)^*$

a ✓	$(a+b)^0 = \epsilon$
b ✓	$(a+b)^1 = a, b$
ab ✗	$(a+b)^2 = (a+b)(a+b)$
ba ✗	$= aa, ab, ba, bb$ .
bb, aa ✗	

$(a+\epsilon)^*$

$$(a+\epsilon)^0 = \epsilon$$

$$(a+\epsilon)^1 = a, \epsilon.$$

$$(a+\epsilon)^2 = (a+\epsilon)(a+\epsilon) \rightarrow a^2, a, \epsilon$$

$$(a+\epsilon)^3 = a^3, a^2, a, \epsilon$$

$$\epsilon, a, a^2, a^3 \dots a^*$$

$(a+b+\epsilon)^* = (a+b)^*$

Kleene Plus.

$R^+$

$$R^1 + R^2 + R^3 \dots$$

$$a^+ = \{a, a^2, a^3, \dots\} = \{a^n \mid n \geq 0\}$$

$$b^+ = \{b, b^2, b^3, \dots\} = \{b^n \mid n \geq 0\}$$

$$1. (a + \epsilon)^+ = a^+$$

$$2. (b + \epsilon)^+ = b^+$$

$$3. (a+b+\epsilon)^+ = (a+b)^+$$

$$4. a^+ + \epsilon = a^+ + \epsilon^0 = a^+ \quad (a+\epsilon) \xrightarrow{\epsilon} a$$

$$5. a^+ + \epsilon = a^+$$

$$R^+ = R^+ + \epsilon^0.$$

$$(a+\epsilon)^2 = (a+\epsilon)(a+\epsilon) = a, \epsilon, a^2$$

$a = a + \epsilon = a, \epsilon, a^2, a^3, \dots, a^*$

$$z = a + z = z + \epsilon$$

OR

Concatenation.

1. Identity.
2. Associative
3. Commutative
4. Dominator.  
(Annihilator)

$\emptyset \neq \emptyset$  (Empty expression)

$\epsilon$

✓ (both)

✓ (both)

✓ (neither)

✗ (neither)

$\Sigma^*$

$\emptyset$

Identity:

$$R + I = I + R = R$$

$$R + \emptyset = \emptyset + R = R$$

$I = \emptyset$   
for +

$\emptyset$  is identity for "OR" operation.

$I = \epsilon$   
for .

$$R \cdot I = I \cdot R = R$$

$$R \cdot \epsilon = \epsilon \cdot R = R$$

$\epsilon$  is identity for concatenation.

Associative:

$$(R_1 + R_2) + R_3 = R_1 + (R_2 + R_3)$$

Left associativity

Right associativity.

$$(a+b)+c = a+(b+c)$$

$\downarrow \{a, b, c\} \swarrow$

$$(R_1 \cdot R_2) \cdot R_3 = R_1 (R_2 \cdot R_3)$$

$$(a \cdot b) \cdot c = a \cdot (b \cdot c)$$

$\hookrightarrow \{a, b, c\} \leftarrow$

### Commutative :

$$R_1 + R_2 = R_2 + R_1 \quad \checkmark \text{ holds}$$

$$R_1 \cdot R_2 \neq R_2 \cdot R_1$$

(need not  
be)                  Does not hold.

### Dominator :

$$R + D = D + R = D \quad * \Sigma^* \text{ is dominator for}$$

$$R + \Sigma^* = \Sigma^* + R = \Sigma^*$$

$$R \cdot D = D \cdot R = D \quad \emptyset \text{ is dominator for.}$$

$$R \cdot \emptyset = \emptyset \cdot R = \emptyset.$$

### Distributive :

#### (I) OR over Concatenation

##### (i) Left distribution

$$a + (b \cdot c) ? (a+b) \cdot (a+c)$$

$\{a, bc\} \neq \{aa, ac, ba, bc\}$

} Does not hold.  
OR should not  
be distributed  
over concatenation.

##### (ii) Right distribution

$$(a \cdot b) + c ? (a+c) \cdot (b+c)$$

$\{ab, c\} \neq \{ab, ac, cb, cc\}$

#### (II) Concatenation over OR

##### (i) Left distribution :

$$a \cdot (b+c) = (a \cdot b) + (a \cdot c)$$

$\{ab, ac\} = \{ab, ac\}$

It holds.

Concatenation is  
distributed over OR.

##### (ii) Right distribution

$$(a+b) \cdot c = (a \cdot c) + (b \cdot c)$$

$\{ac, bc\} = \{ac, bc\}$

Simplification:

$$R^* = R' + R \dots$$

$$1. \phi^* = \phi$$

$$\phi^* = \phi^0 + \phi' + \phi^2 \dots = \phi + \phi = \underline{\underline{\phi}}$$

$$2. E^* = E$$

$$3. \phi^+ = \phi$$

$$^*(\text{Any})^0 = E$$

$$4. E^+ = E$$

$$^*\text{Any. } \phi = \phi$$

$$5. (a + \phi)^* = a^*$$

$$^*E^* = E$$

$$6. (a + \phi)^+ = a^+$$

$$7. (E + \phi)^+ = E^+ = E$$

$$13. a^* \cdot \phi^* = a^* \cdot E = a^*$$

$$8. (E + \phi)^* = E^+ = E$$

$$14. a^* \cdot \phi^+ = \phi$$

$$9. \phi^* \cdot \phi^+ = E \cdot \phi = \phi$$

$$15. a^* \cdot a = a^+$$

$$10. E^* \cdot E^+ = E \cdot E = E$$

$$\uparrow \downarrow \quad a^* = E, a, aa \dots \cdot a$$

$$11. \phi^* \cdot E^+ = E \cdot E = E$$

$$16. a \cdot a^+ = a^+$$

$$12. \phi^+ \cdot E^* = \phi \cdot E = \phi$$

$$17. a^* + a^* = a^*$$

$$23. a^* + a^+ + a + a^{100} = a^* \quad a - a \cdot a$$

$$18. a^* + a^+ = a^*$$

$$24. (a^+)^2 = a^* \cdot a^{**} = a^* \cdot a \cdot a = a \cdot a^+$$

$$19. a^+ + a^+ = a^+$$

$$25. (a^*)^2 = a^* \cdot a^* = a^*$$

$$20. a + a^+ = a + (a + a^2 + \dots) = a^+$$

$$26. (a^*)^{100} = a^*$$

$$21. a + a^* = a^*$$

$$27. a^* a^* a^+ a^* = a^*$$

$$22. a^{100} + a^+ = a^+$$

$$28. (a^*)^* = (a^0) \vee (a^1) \vee \dots = a^*$$

$$30. (a^+)^* = (a^0) \vee (a^1) \dots = E + a^+ = a^*$$

$$29. (a^*)^+ = (a^0) \vee (a^1)^2 \dots = a^*$$

$$31. (a^+)^+ = (a^1) \vee (a^2)^2 \dots = a^+$$

Note:

$$(I) R + \phi = R$$

$$(IV) a \cdot a^* = a^*$$

$$(II) R \cdot \phi = \phi$$

$$(V) a^* \cdot a = a^*$$

$$(III) R \cdot E = R$$

$$(VI) (a + E)^* = a^*$$

$$(IV) R^0 = E$$

$$(VII) (a + E)^+ = a^*$$

$$(V) R + \Sigma^* = \Sigma^*$$

$$(VIII) a^+ + E = a^*$$

Homework : Find the min string in the following expression :

1.  $(ab)^+ a a a (b+ab) \rightarrow \text{min} = ab a a a b$
2.  $(ab)^+ a a a (b+\epsilon) \rightarrow a a a$
3.  $(a+ab)^+ (bb+aaa)^+ (ab)^+ \rightarrow bb$
4.  $(ab)^+ + \epsilon \rightarrow ab$
5.  $(ab)(atb)^+ a a a + ba \rightarrow ba$
6.  $[(ab^+ + a^+ + ba) ab]^+ (a+ab)^+ \rightarrow a a b$
7.  $[( (ab)^+ a )^+ a a a ]^+ \rightarrow a a a$
8.  $[( (aba)^+ aba )^+ ab ]^+ \rightarrow \epsilon$
9.  $(a^+)^+ \rightarrow a$
10.  $((ab)^+ a^+)^+ bb^+ aa^+ \rightarrow abba$
11.  $[(ab)^+ aba ]^+ a a a \rightarrow ababa a a a$
12.  $(aaa^* aba^+)^+ \rightarrow a a a b$
13.  $(a+a a+a a a)^+ \rightarrow a$
14.  $a^+ b^+ \rightarrow ab$
15.  $a^+ b^+ \rightarrow \epsilon$
16.  $a^+ b^+ c^+ d^+ \rightarrow c$
17.  $(a+b^+ + c^+) \rightarrow a$
18.  $(a+b^+ + \epsilon)^+ \rightarrow \epsilon$
19.  $[(a+ab)aa+ab]^+ a a a \rightarrow abaaa$
20.  $(ab.aaa.a.ta^+)^+ \rightarrow \epsilon$

Priority : 1. Kleene plus, Kleene star

2. Concatenation

3. OR

Q1. How to write Regular expression?

→ Regular Language  $\cong$  Regular Expression.

$$1. L = \{ \} \text{ over } \Sigma = \{a\}$$

$R = \emptyset$ .

$$2. L = \{ \} \text{ over } \Sigma = \{a, b\}$$

$R = \emptyset$ .

Note: To represent one regular language, there are infinite regular expressions.

3.  $L = \{\epsilon\}$  over  $\Sigma = \{a\}$

$$R = E_+ = \phi = \phi^* \cdot a^* \dots$$

$$a^* = (a^*)^2 = (a^*)^* = (a^*)^+$$

$$= a^* + \epsilon$$

$$= (a + \epsilon)^*$$

$$= (a + \epsilon)^*$$

$$= \phi^* \cdot a^*$$

$$= a^* + \text{any other } \Sigma \text{ exp.}$$

4.  $L = \{\epsilon\}$  over  $\Sigma = \{a, b\}$

$$R = E_+ = \phi^* = a^* = b^* \dots \text{ (and many more).}$$

5.  $L = \text{set of all strings over } \Sigma = \{a\}$

$$R = a^* = \Sigma^*$$

$$ab^* \neq b^* a^*$$

$$(a^* b^*)^* = (b^* a^*)^*$$

6.  $L = \text{set of all strings over } \Sigma = \{a, b\}$

$$R = (a+b)^* = \Sigma^*$$

7.  $L = \{w \mid w \in \{a, b\}^*, w \text{ starts with } a\}$

$$= \{aw \mid w \in \{a, b\}^*\}$$

$$R = a(a+b)^* = a\Sigma^*$$

eg::  $A = \{x \mid x \in \mathbb{N}, x \geq 2\}$

Cond. 1      Cond. 2.

Every natural number greater than 2 = {3, 4, 5, ...}

8. Set of all strings starting with ab over  $\Sigma = \{a, b\}$

$$R = ab(a+b)^*$$

9.  $L = \{w \mid w \in \{a, b\}^*, w \text{ starts with aaa}\}$

$$R = aaa(a+b)^*$$

10.  $L = \{w \mid w \in \{a, b\}^*, w \text{ ends with a}\}$

$$R = (a+b)^* a$$

11.

## Group I.

1.  $a(a+b)^*$
2.  $b(a+b)^*$
3.  $(a+b)^*a$
4.  $(a+b)^*b$

## Group II.

- I.  $(ab^*)^* = \{a, aa, ab\dots\}$
- II.  $(b^*a)^* = (a+b)^*a$
- III.  $(ba^*)^* = b(a+b)^*$
- IV.  $(a^*b)^* = (a+b)^*b.$

12.

$$L = \{w_1 a w_2 \mid w_1, w_2 \in \{a, b\}^*\}$$

= Set of all strings containing 'a' as substring  
 $\{a, aa, ab\dots\}$

$(a+b)^* a (a+b)^*$  } Same.  
 $b^* a (a+b)^*$  }

13.

$$L = \{w \mid w \in \{a, b\}^*, \underbrace{|w|=2}\}$$

exactly 2 length string.

$$R_1 = ab + ba + aa + bb$$

$$= (a+b)(a+b)$$

$$= (a+b)^2 = \Sigma^2$$

14.

$$\{w \mid w \in \{a, b\}^*, \underbrace{|w| \leq 2}\}$$

at most 2 length string.

$|w|=k$

$$R_2 = \epsilon + a + b + aa + ab + ba + ba$$

$$= \Sigma^0 + \Sigma^1 + \Sigma^2 = (\epsilon + a + b)^2$$

$$= (a+b+\epsilon)^k$$

String	Length.
$\epsilon$	0
$a+b$	1
$ab + \epsilon$	0 or 1 $\leq 1$
$(a+b+\epsilon)^k$	$\leq 2 \text{ length}$

15.  $\{w \mid w \in \{a, b\}^*, |w| \leq 2\}$

At least 2 length

$$\begin{aligned}
 R &= \underbrace{(a+b)}_{\text{exactly}}^* \underbrace{(a+b)}_{\text{any}}^* \\
 &= (a+b)^* (a+b)^* \\
 &= (a+b) (a+b)^* \\
 &= (a+b)^* (a+b)
 \end{aligned}$$

16.  $\{w \mid w \in \{a, b\}^*, n_a(w) = 2\}$

$$\begin{aligned}
 R &= b^* ab^* ab^* \\
 &= (b^* a)^2 b^* \\
 &= b^* (ab^*)^2
 \end{aligned}$$

17.  $\{w \mid w \in \{a, b\}^*, n_a(w) \leq 2\}$

$$\begin{aligned}
 R &= b^* + b^* ab^* + b^* ab^* ab^* \\
 &= b^* (E+a) b^* (E+a) b^*
 \end{aligned}$$

$z \in a = b^*$

one  $a = b^* ab^*$

$2 a = b^* ab^* ab^*$

18.  $\{w \mid w \in \{a, b\}^*, n_a(w) \geq 2\}$

$$R = (a+b)^* a (a+b)^* a (a+b)^*$$

19.  $\{w \mid w \in \{a, b\}^*, |w| \text{ is even}\}$

$$R = \left[ (a+b)^2 \right]^*$$

20.  $\{w \mid w \in \{a, b\}^*, |w| \text{ is odd}\}$

$$\begin{aligned}
 R &= \left[ (a+b)^2 \right]^* (a+b) \\
 &\quad \underbrace{\text{even}}_{\text{odd}} \quad \underbrace{\text{odd}}_1 \\
 &= (a+b) \left[ (a+b)^2 \right]^*
 \end{aligned}$$

odd

21.  $L = \{xay \mid x, y \in \{a, b\}^*, |x|=3\}$

= 4<sup>th</sup> symbol from beginning is a.

$$\underbrace{(a+b)^3}_3 a \underbrace{(a+b)^*}_{4^{\text{th}} \text{ symbol}}$$

3 length 4<sup>th</sup> symbol.

22.  $\{w \mid w, y \in \{a, b\}^*, |y| = 2\}$   
any 2 length.  
 $= (a+b)^* a (a+b)^*$ . { 3rd symbol from end is a. }

23.  $\{a^m b^n \mid m, n \geq 0\}$   
 $R = a^* b^*$ .

24.  $\{a^m b^n \mid m \geq 0, n \geq 1\}$   
 $R = a^* b^+$

25.  $\{a^m b^n \mid m \geq 1, n \geq 1\}$   
 $R = a^+ b^+$ .

26.  $\{a^m b^n \mid m = \text{even}, n \geq 0\} = R = (aa)^* b^*$

27.  $\{a^m b^n \mid m = \text{odd}, n \geq 1\} = R = a (aa)^* b^*$

28.  $\{a^m b^n \mid m \geq 2, n = \text{even}\} = R = aaa^* (bb)^*$

29.  $\{a^m b^n \mid m \geq 2, n \geq 3\} = R = aaa^* bbbb^*$

30.  $\{a^m b^n \mid m \leq 1, n \geq 2\} = R = \underbrace{(e+a)}_{\text{Max fa.}} \dashv \underbrace{babab^*}_{\text{min 2 b's.}}$

31.  $\{w \mid w \in \{a, b\}^*, \text{na}(w) = \text{odd}\}$   
 $\begin{aligned} &= b^* (b^* ab^* ab^*)^* b^* ab^* \\ &= b^* ab^* (b^* ab^* ab^*)^* b^* ab^*. \\ &= b^* ab^* (babab^*)^* b^*. \end{aligned}$

~~two  
find our  
other  
equivalent  
equations.~~

32.  $\{w \mid w \in \{a, b\}^*, \text{na}(w) = \text{even}\}$   
 $\begin{aligned} &= b^* (b^* ab^* ab^*)^* b^* \\ &= b^* (babab^*)^* b^*. = (b^* ab^* a)^* b^*. \end{aligned}$

Finite Automata.

(Finite State Machine)

(Finite Machine)

It represents a regular language

(acceptor)

(recogniser).

Finite Automata.

without O/p.

(Acceptors)

input

FA

DFA

Deterministic FA.

Non-deterministic FA

Input &amp; move

FA

Output &amp; move

every O/p.

(Transducer)

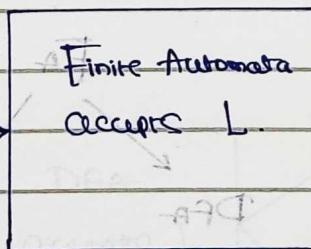
B/p

FA

O/p.

Moore m/c

Mealy m/c.

(w)  
Input String.

If  $w \in L$ , FA halts at final state.  
If  $w \notin L$ , FA halts at non-final state.

invalid string.

 $L = ab(ab)^*$ 

all strings start with ab.

$$\Sigma^* = \{ \epsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, \dots \}$$

## FA Configuration

$$FA = (Q, \Sigma, \delta, q_0, F)$$

↓  
Set of final states

↓  
Initial state of  $q_0 \in Q$

↓  
Transition Function

↓  
Input alphabet

Set of states

$Q = \text{Set of states}$

$$= \{1, 2\}$$

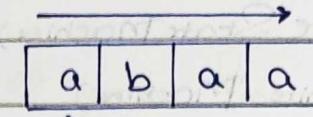
$\Sigma = \text{input alphabet}$

= Set of i/p symbol

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

$\delta = \text{Transition function}$

$q_0 = \text{Initial state}$



Read head.

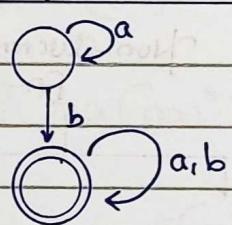
↓  
Finite Control.

↓  
Working

(contd.)

$$\begin{array}{|c|c|c|} \hline a & b & a \\ \hline \end{array}$$

↓  
↓



$$0 \leq |F| \leq |Q|$$

$F = \text{set of final states}$

$$F \subseteq Q$$

$$F = \{2\}$$

FA

DFA

NFA

$$\delta: Q \times \Sigma \rightarrow Q$$

without  $\epsilon$  move.

$$\delta: Q \times \Sigma \rightarrow Q$$

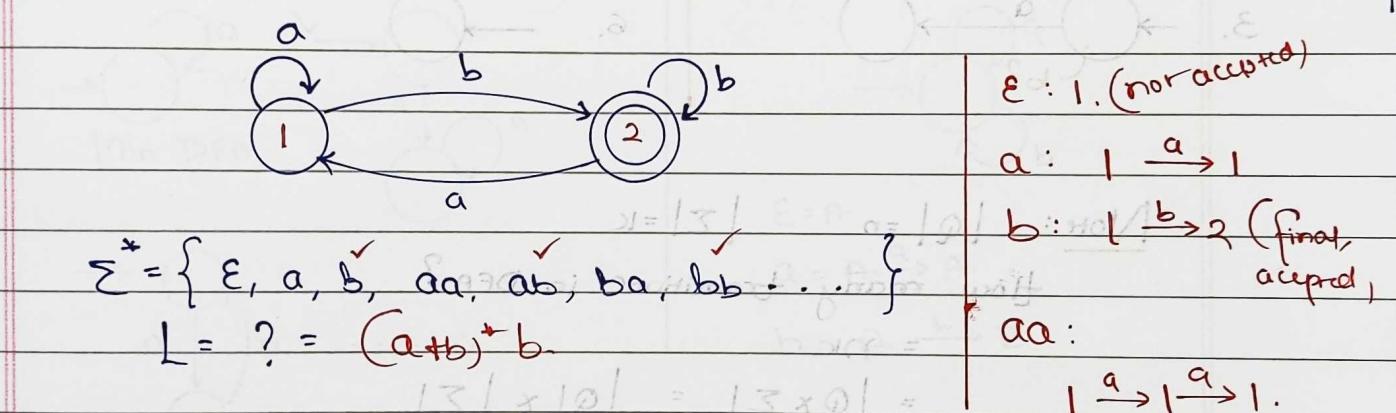
with  $\epsilon$  move.

$$\delta: Q \times \Sigma \cup \{\epsilon\} \rightarrow Q$$

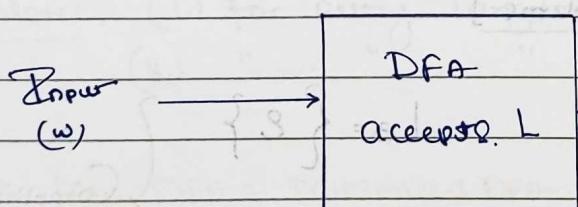
## FA Representations

1. State Diagram.
2. Transition Table.
3. Set Function / Relation.

Diagram	Table	Set.
<p>Initial State: <math>\rightarrow \textcircled{1}</math></p> <p>Final State / Non-Final State: <math>\textcircled{1} / \textcircled{2}</math></p>	$\Sigma$ $\delta$   $\Sigma$ $\delta$   a b $Q = \{1, 2\}$ $\Sigma = \{a, b\}$	$F_A = (Q, \Sigma, \delta, q_0, F)$ $Q = \{1, 2\}$ , $\Sigma = \{a, b\}$ , $\delta: Q \times \Sigma \rightarrow Q$ $\delta(1, a) = 1$ , $\delta(1, b) = 2$ , $\delta(2, a) = 2$ , $\delta(2, b) = 1$ $\text{I. } \delta = \{(1, a, 1), (1, b, 2), (2, a, 2), (2, b, 1)\}$
<p>Transition:</p> $P_S \xrightarrow{i} N_S$		



→ What is DFA?



If  $w \in L$ , exactly one path exists that ends at final state (therefore accepted)  
 If  $w \notin L$ , exactly one path exists that ends at non-final state (therefore rejected)

### DFA Definitions

I.  $\delta: Q \times \Sigma \rightarrow Q$

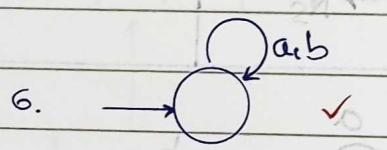
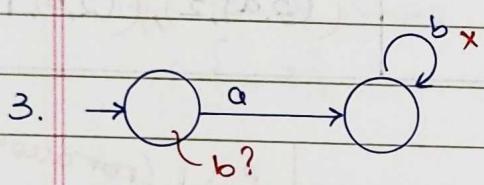
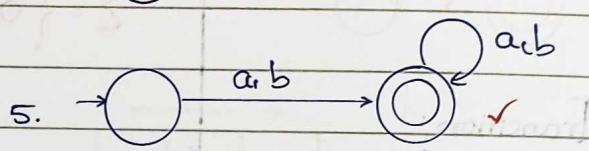
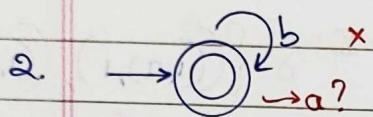
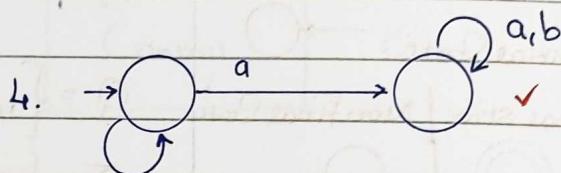
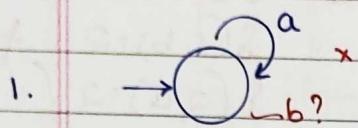
II. From every state, for every input symbol, exactly one transition to next state.

III.  $\forall w \in \Sigma^* \Rightarrow$  exactly one path exists.

[ valid string has a final state  
invalid string has an nonf. ]

Identify Correct DFAs

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



Note:  $|Q| = n$      $|\Sigma| = k$

How many transitions in DFA?

$$= nk$$

$$= |Q \times \Sigma| = |Q| \times |\Sigma|$$

## Modu 2. [Special problems]

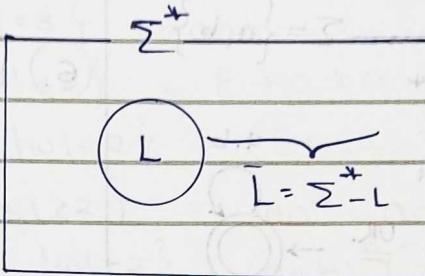
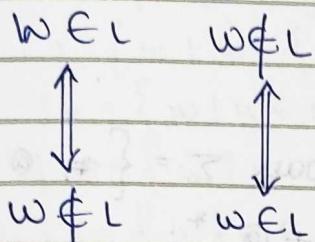
$$\left. \begin{array}{l} L_1 = \emptyset = \{\} \\ L_2 = \Sigma^* \\ L_3 = \{e\} \\ L_4 = \Sigma^+ \end{array} \right\} \text{Compliment to each other.}$$

Universal Set.

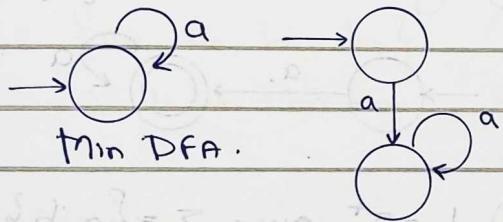
$$\rightarrow \bar{L} = \sum^+ L$$

= Universal Set - L

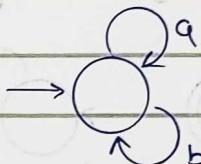
$$L \cup \bar{L} = \Sigma^*$$
$$L \cap \bar{L} = \emptyset$$



1.  $L = \{ \}$  over  $\Sigma = \{a\}$



$$(2) \quad L = \{ \} \quad \text{our } \Sigma = \{a, b\}$$



$$\Sigma = \emptyset$$

$$\alpha = A \xrightarrow{a} A$$

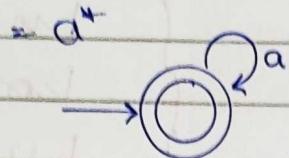
$$b = A \xrightarrow{b} B.$$

→ Min. DFA = Minimized DFA = minimal DFA = DFA with minimum no. of states

(iii) If DFA has only non final states  
then  $L(DFA)$  is  $\emptyset = \{ \}$

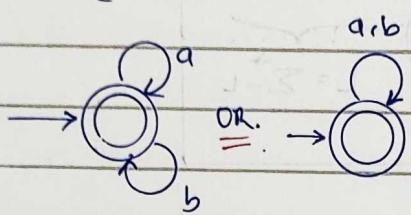
(iv) If DFA has only final state then  $L(DFA)$  is  $\underline{\Sigma}^*$

3.  $L = \Sigma^*$  over  $\Sigma = \{a\}$



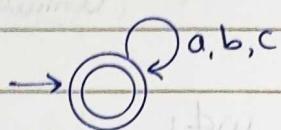
4.  $L = \Sigma^*$  over  $\Sigma = \{a, b\}$

$$= (a+b)^*$$



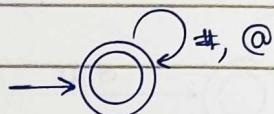
5.  $L = \Sigma^*$  over  $\Sigma = \{a, b, c\}$

$$= (a+b+c)^*$$

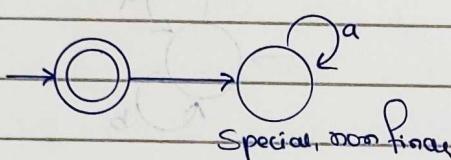


6.  $L = \Sigma^*$  over  $\Sigma = \{\#, @\}$

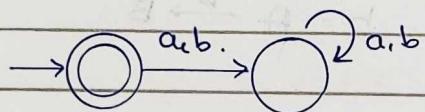
$$= (\# + @)^*$$



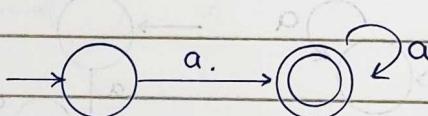
7.  $L = \{\epsilon\}$  over  $\Sigma = \{a\}$ .



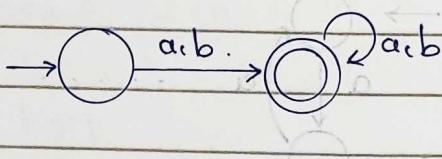
8.  $L = \{\epsilon\}$  over  $\Sigma = \{a, b\}$ .



9.  $L = \Sigma^*$  over  $\Sigma = \{a\}$

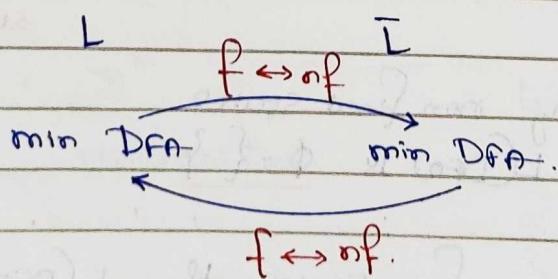


10.  $L = \Sigma^*$  over  $\Sigma = \{a, b\}$ .



Note:

If  $L$  has min DFA with n states, then  $\bar{L}$  has min states DFA with n states.



## Model-II. [Length based problems].

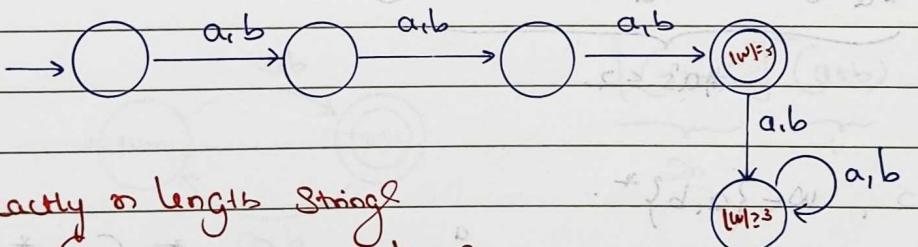
1.  $L_1 = \{ w \mid w \in \{a, b\}^*, |w| = 2 \} = (a+b)^2$
2.  $L_2 = \{ w \mid w \in \{a, b\}^*, |w| = 3 \} = (a+b)^3$
3.  $L_3 = \{ w \mid w \in \{a^*\}, |w| = 2 \} = aa$
4.  $L_4 = \{ w \mid w \in a^*, |w| = 3 \} = aaa$
5.  $L_5 = \{ w \mid w \in a^*, |w| \leq 2 \} = \epsilon + a + a^2 = (\epsilon + a)^2$
6.  $L_6 = \{ w \mid w \in (a+b)^*, |w| \leq 2 \} = (\epsilon + a+b)^2$
7.  $L_7 = \{ w \mid w \in a^*, |w| \geq 2 \} = aaa^* = aa^* = a^*a = a^*aa$
8.  $L_8 = \{ w \mid w \in (a+b)^*, |w| \geq 2 \} = (a+b)^2(a+b)^*$

$\rightarrow L_1 = \{ w \mid w \in \{a, b\}^*, |w| = 2 \} = (a+b)^2$   
 = set of all 2 length string.  
 =  $\{aa, ab, ba, bb\}$   
 =  $aa + ab + ba + bb = (a+b)(a+b) = (a+b)^2$

Note:

$$|w| = k \Rightarrow k+2 \text{ states in DFA}$$

$\rightarrow L = (a+b)^3$ .



Exactly n length string  
 $(a+b)(a+b) \dots n \text{ times}$

$\downarrow$   
 $n+2$  states in DFA.

Exactly n length      } Dead state in  $\Rightarrow (n+1) + 1$  dead state.  
 Atmost n length      } Required

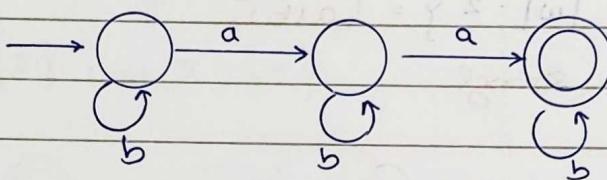
Atleast n length      } Dead state not required  $\Rightarrow n+1$

tlw

## Modu - III [No. of symbols]

1.  $\{ w \mid w \in \{a, b\}^*, \text{ na}(w) = 2 \}$
2.  $\{ w \mid w \in \{a, b\}^* \text{ na}(w) \leq 2 \}$
3.  $\{ w \mid w \in \{a, b\}^* \text{ na}(w) \geq 2 \}$
4.  $\{ w \mid w \in a^*, \text{ na}(w) = 2 \}$
5.  $\{ w \mid w \in a^*, \text{ na}(w) \leq 2 \}$
6.  $\{ w \mid w \in a^*, \text{ na}(w) \geq 2 \}$

1. No. of a's = 2.  $w \in \{a, b\}^*$



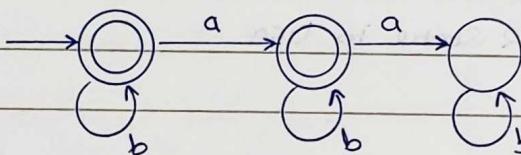
$$\#a^s = 2$$

$$\#b^s \geq 0$$

$$b^*ab^*ab^*$$

2. No. of a's  $\leq 2$   $w \in \{a, b\}^*$

$$b^*(a+\epsilon)b^*(a+\epsilon)b^*$$



$$\#a^s = 0$$

$$\#a^s = 1$$

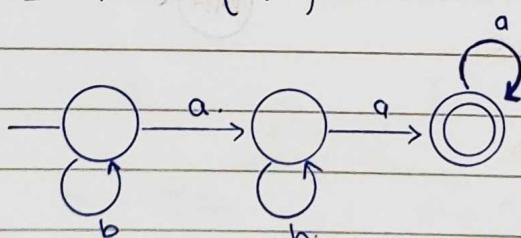
$$\#a^s = 2$$

$$\#a^s \leq 2$$

$$\#b^s \geq 0$$

$$\#a^s \leq 2$$

3.  $\#a^s \geq 2$ ,  $w \in \{a, b\}^*$ .

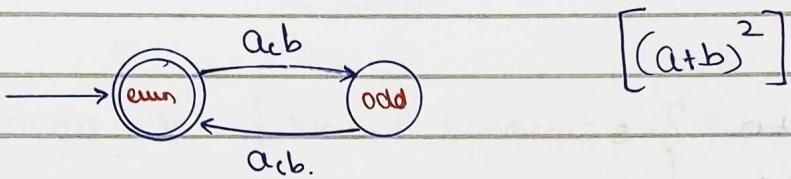


$$(a+b)^*a(a+b)^*a(a+b)^*$$

## Module - IV Length and Remainder

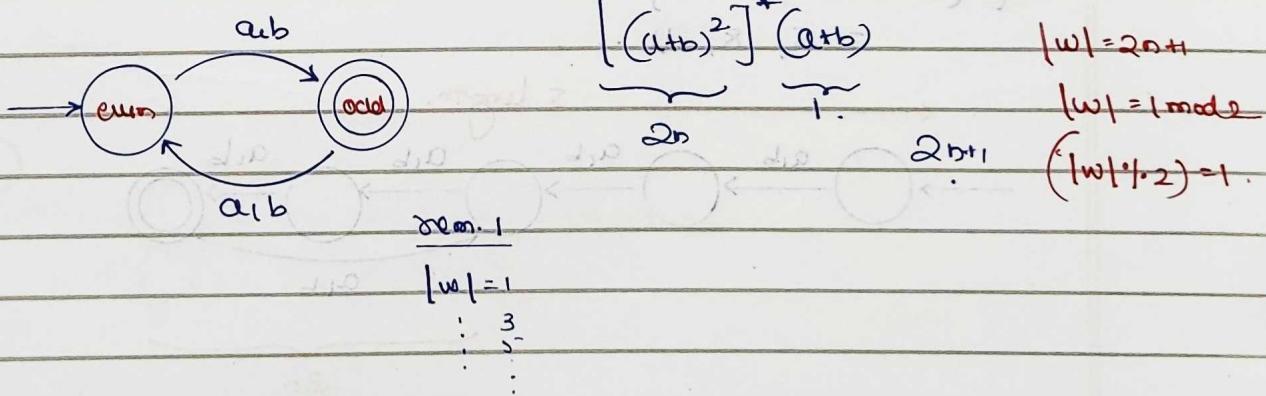
1.  $\{w \mid w \in \{a, b\}^*, |w| \text{ is divisible by } 2\}$  } Complement
2.  $\{w \mid " " \quad |w| \text{ is not divisible by } 2\}$  } to each other.
3.  $\{w \mid " " \quad |w| \text{ is divisible by } 3\}$
4.  $\{w \mid " " \quad |w| = 3n+2 \quad n \geq 0\}$
5.  $\{w \mid " " \quad |w| = 3n+5 \quad n \geq 0\}$

$\rightarrow \{w \mid w \in \{a, b\}^*, |w| = \text{even}\}$ .

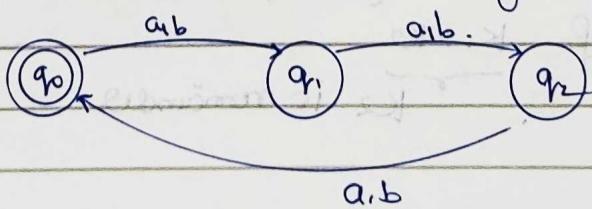


$ w =0$	$ w =1$
$ w =0$	$ w =1$
" 4	" 3
" 6	" 5
:	:

$\rightarrow \{w \mid w \in \{a, b\}^*, |w| = \text{odd}\}$ .



$\rightarrow \{w \mid w \in \{a, b\}^*, |w| \text{ is divisible by } 3\}$ .



Note:  $L = \{w \mid w \in \{a,b\}^*, |w| \text{ is divisible by } k\}$ .

No of states in min DFA =  $k$  states.

5 States

Final  
Start  
may  
differ.

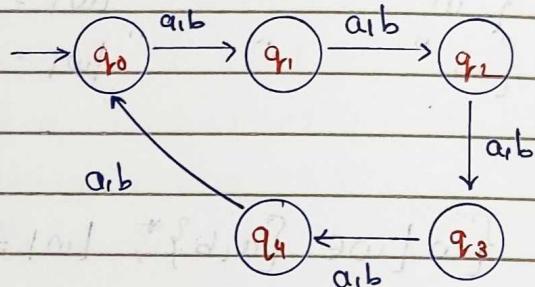
$|w| = 5n \Rightarrow \text{rem 0 when divide with } 5$ .

$$|w| = 5n+1 = \text{rem 1.}$$

$$|w| = 5n+2 = \text{rem 2.}$$

$$|w| = 5n+3 = \text{rem 3.}$$

$$|w| = 5n+4 = \text{rem 4}$$



$$|w| = 3n + 0 \quad \left\{ \begin{array}{l} \text{3 states.} \\ \text{= } 3n + 1 \end{array} \right.$$

$$= 3n + 2$$

$$= 3n + 3$$

$$= 3n + 4$$

$$= 3n + 5$$

$$= 3n + \text{constant (k)} \rightarrow k+1 \text{ State.}$$

If  $k_1 > k_2$ ,  $k_2$  is the remainder.

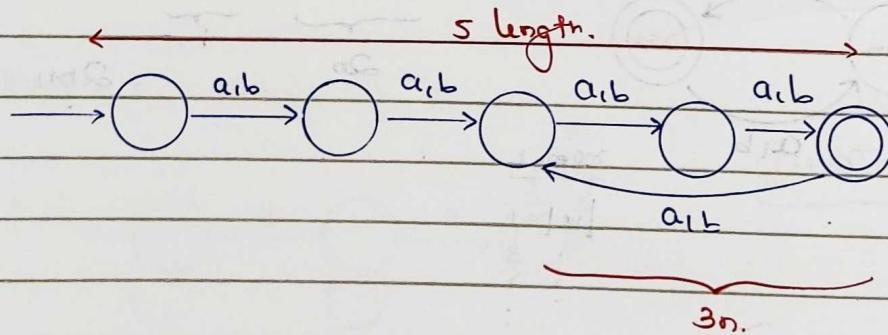
If  $k_1 \leq k_2$ .



$$\left\{ \begin{array}{l} |w| = 3n + 5. \\ \text{= } 5, 8, 11 \dots \end{array} \right.$$

$$3n+2, n \geq 0.$$

5 length.



Note:  $L = \{w \mid w \in \{a,b\}^*, |w| = k_1n + k_2, n \geq 0\}$

I.) If  $k_1 > k_2$

$\rightarrow k_2$  is remainder.

$\rightarrow k_1$  state

II) If  $\underbrace{k_1 \leq k_2}_{k_2 \text{ is not}} \Rightarrow k_2+1 \text{ states}$   
 remainder.

Note:  $L = \{w \mid w \in \{a,b\}^*, \#_a(w) = k_1, \#_b(w) = k_2 \geq 0\}$ .

I) If  $\underbrace{k_1 > k_2}_{k_2 \text{ is remainder}} \Rightarrow k_1 \text{ states.}$

II) If  $\underbrace{k_1 \leq k_2}_{k_2 \text{ is not}} \Rightarrow k_2+1 \text{ states}$   
 remainder.

### Model - IV. [No. of Symbols and Remainder]

1.  $\{w \mid w \in \{a,b\}^*, \#_a(w) = \text{even}\}$  } 2 states

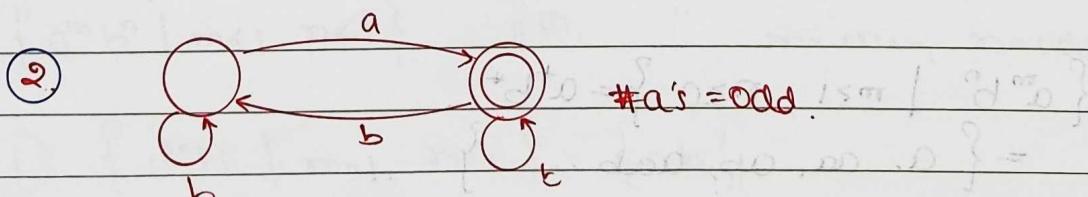
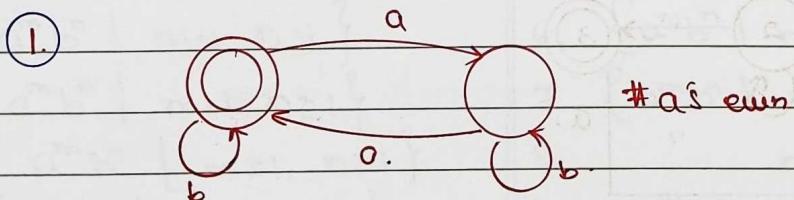
2.  $\{w \mid \text{" " " odd}\}$

3.  $\text{" " " divisible by 3?}$

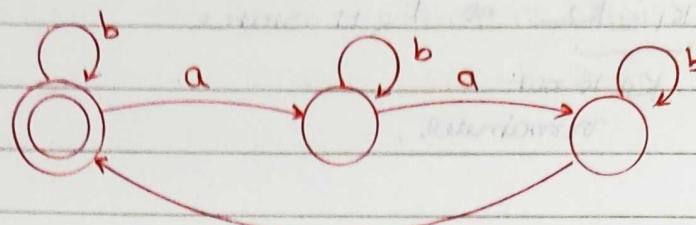
4.  $\text{" " " not divisible by 3?}$

5.  $\{w \mid \#_a(w) = 100n + 53\}$  }  $n \geq 0$  → 100 states

6.  $\{w \mid \#_a(w) = 123n + 1125\}$  }  $n \geq 0$  → 1126 states

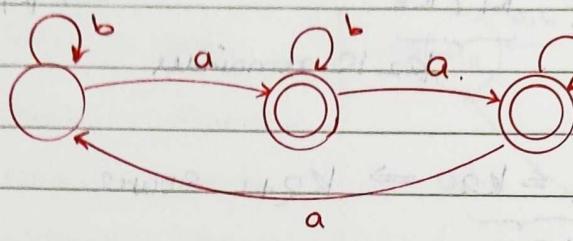


(3)



# a's in die by 3

(4)



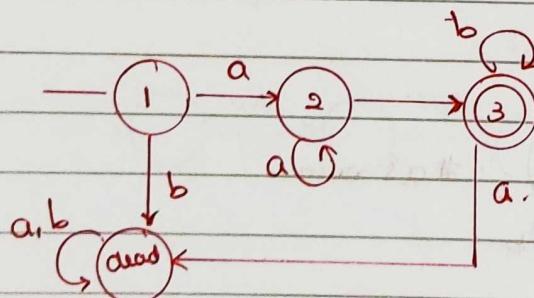
$$\begin{aligned} [\# \text{a's in } w] \mod 3 & \neq 0 \\ &= 1 \\ &= 2. \end{aligned}$$

#### Moar Vir. [Sequence]

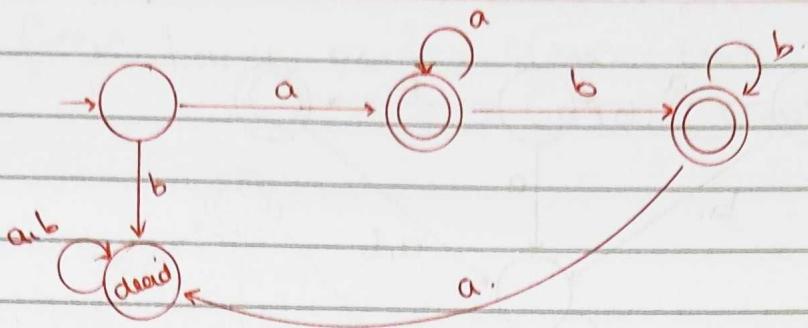
1.  $a^+ b^+$
2.  $a^+ b^*$
3.  $a^* b^+$
4.  $a^* b^*$ .

}  $a^*$  forwarded by  $b^*$ .

$$\begin{aligned} \rightarrow L = \{a^m b^n \mid m \geq 1, n \geq 1\} &= a^+ b^+ \\ &= aa^+ bb^* \quad \left. \begin{array}{l} a^* \text{ will never appear} \\ \text{after } b^*. \end{array} \right\} \\ &= \{ab, aab, abb, \dots\} = a^+ a b b^* \\ &= a^* a b^* b = (a^+)^+ (b^+)^+ \quad \left. \begin{array}{l} b^* \text{ will never} \\ \text{appear before } a^*. \end{array} \right. \end{aligned}$$



$$\begin{aligned} \rightarrow \{a^m b^n \mid m \geq 1, n \geq 0\} &= a^+ b^+ \\ &= \{a, aa, ab, aab, \dots\} \end{aligned}$$

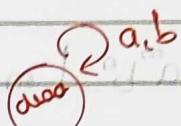


$$\rightarrow L = a^* b^*$$

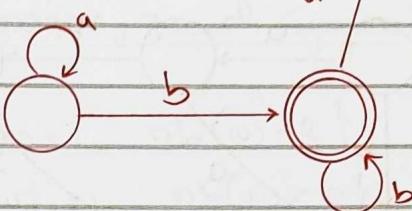
$$= \{ a^m b^n \mid m \geq 0, n \geq 0 \}$$

$$= \{ b \dots \}$$

min. a<sup>0</sup>b<sup>1</sup>



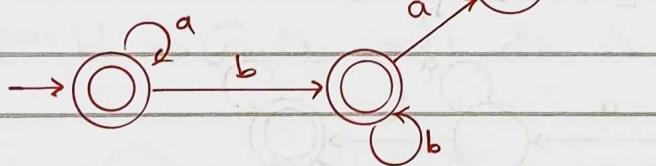
= 3 states.



$$\rightarrow a^* b^* = \{ a^m b^n \mid m \geq 0, n \geq 0 \}$$

$$= \{ \dots \}$$

min. a<sup>0</sup>b<sup>0</sup>



Homework

5.  $L = b^* a^*$

6.  $L = b^* a$

7.  $L = b^* a^*$

8.  $L = b^* a^*$

9.  $L = a^* b^* c$

10.  $L = b^* a^* c^*$ .

### Modu-7: [sequences, no. of symbols]

1.  $\{ a^m b^n \mid m=1, n=1 \}$

2.  $\{ a^m b^n \mid m=1, n \geq 1 \}$

3.  $\{ a^m b^n \mid m \geq 1, n \geq 1 \}$

4.  $\{ a^m b^n \mid m \leq 1, n=1 \}$

5.  $\{ a^m b^n \mid m \leq 1, n \leq 1 \}$

6.  $\{ a^m b^n \mid m=\text{even}, n=1 \}$

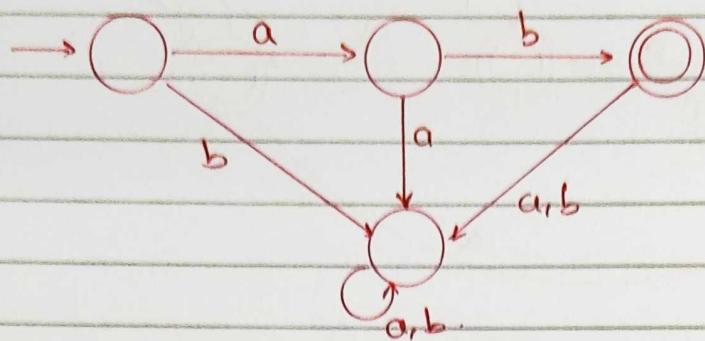
7.  $\{ a^m b^n \mid m=\text{odd}, n=1 \}$

8. "  $m=1, n=\text{even}$ "

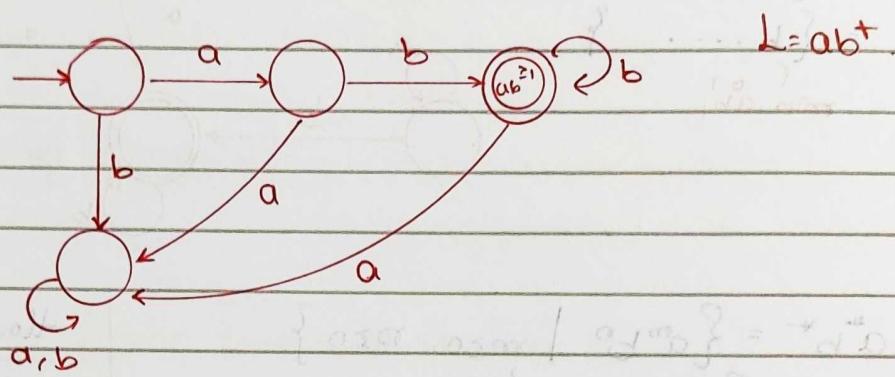
9. "  $m=\text{even}, n=\text{even}$ "

10. "  $m=\text{even}, n=\text{odd}$ "

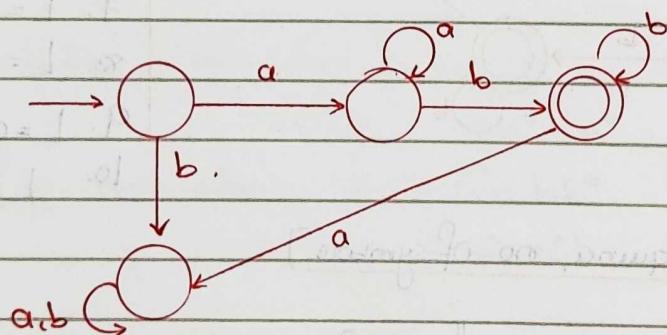
①  $\{ a^m b^n \mid m=1, n=1 \} = \{ ab \}$ .



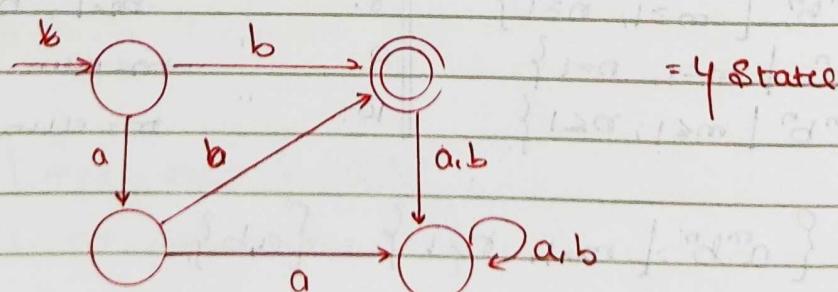
②  $\{a^m b^n \mid m \geq 1, n \geq 1\} = \{ab, \underbrace{ab}_{\text{min}}, \dots\}$



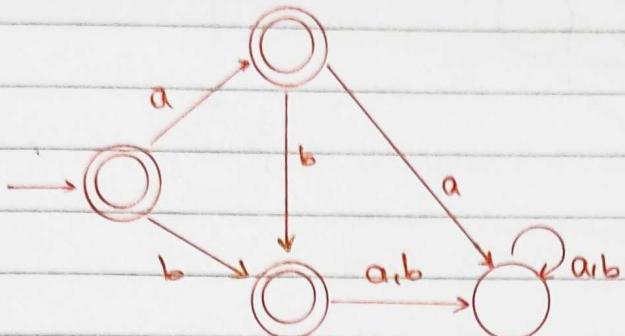
③  $L = \{a^m b^n \mid m \geq 1, n \geq 1\} = a^+ b^+$



④  $\{a^m b^n \mid m \leq 1, n \geq 1\} = (\epsilon + a)b = b + ab.$



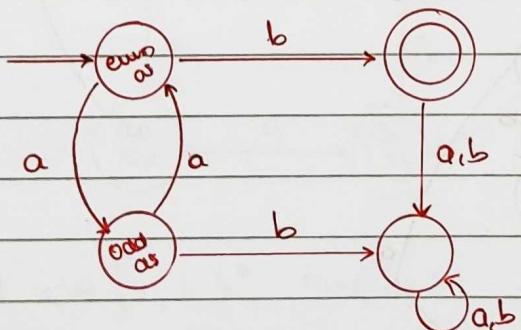
$$(5) \{a^m b^n \mid m \leq 1, n \leq 1\} = (E+a)(E+b) \\ = (a^0 + a^1)(b^0 + b^1) = \{E, a, b, ab\}.$$



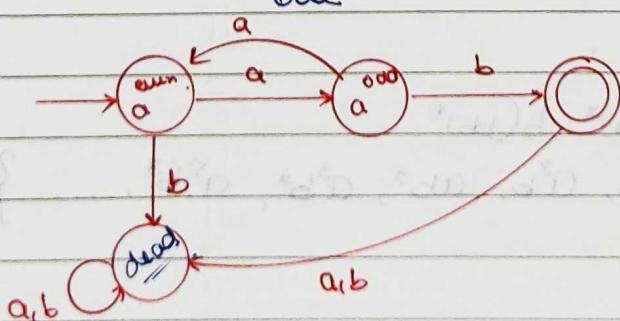
### Modu - VIII [Multiple Conditions, No. of Symbols]

1.  $\{\omega \mid \omega \in \{a, b\}^*, n_a(\omega) = 1, n_b(\omega) = 2\}$
2.  $\{\omega \mid \text{ " " } n_a(\omega) \geq 1, n_b(\omega) \geq 2\}$
3.  $\text{ " " } n_a(\omega) \leq 1, n_b(\omega) \leq 2\}$
4.  $\text{ " " } n_a(\omega) \leq 1, n_b(\omega) \leq 2\}$
5.  $\text{ " " } n_a(\omega) \geq 1, n_b(\omega) \leq 2\}$ .

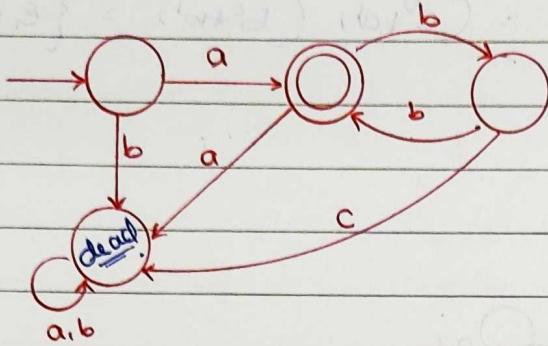
$$(6) a^{\text{even}} b^l \rightarrow (aa)^* b = \{b, ab, a^4b, a^6b, \dots\}$$



$$(7) a^{\text{odd}} b^l \rightarrow \underbrace{a(aa)^* b}_{\text{odd}} = \{ab, a^3b, a^5b, \dots\}$$

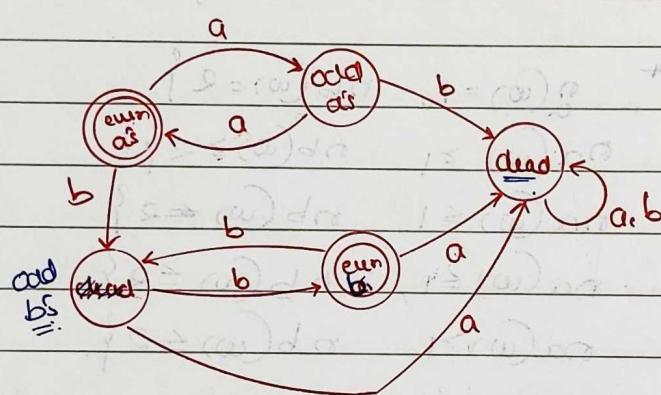


$$8 \quad a^{\text{even}} b^{\text{even}} = a(bb)^* \rightarrow \{a, ab^2, ab^4, \dots\}$$

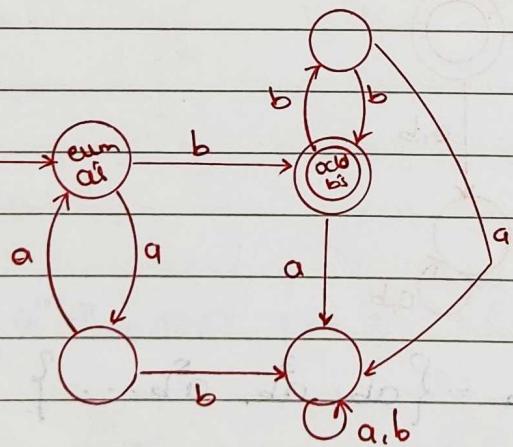


$$9 \quad a^{\text{even}} b^{\text{even}} = (aa)^* (bb)^*$$

$$= \{\epsilon, a^2, b^2, a^4, b^4, \dots\}$$

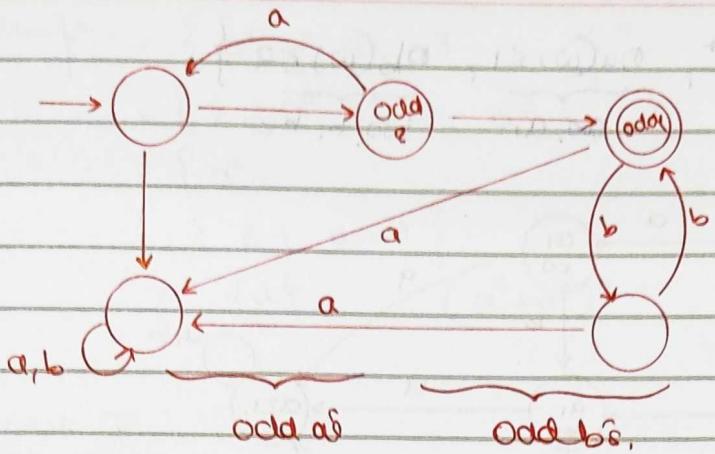


$$10 \quad a^{\text{even}} b^{\text{odd}} = (aa)^* b (bb)^*$$



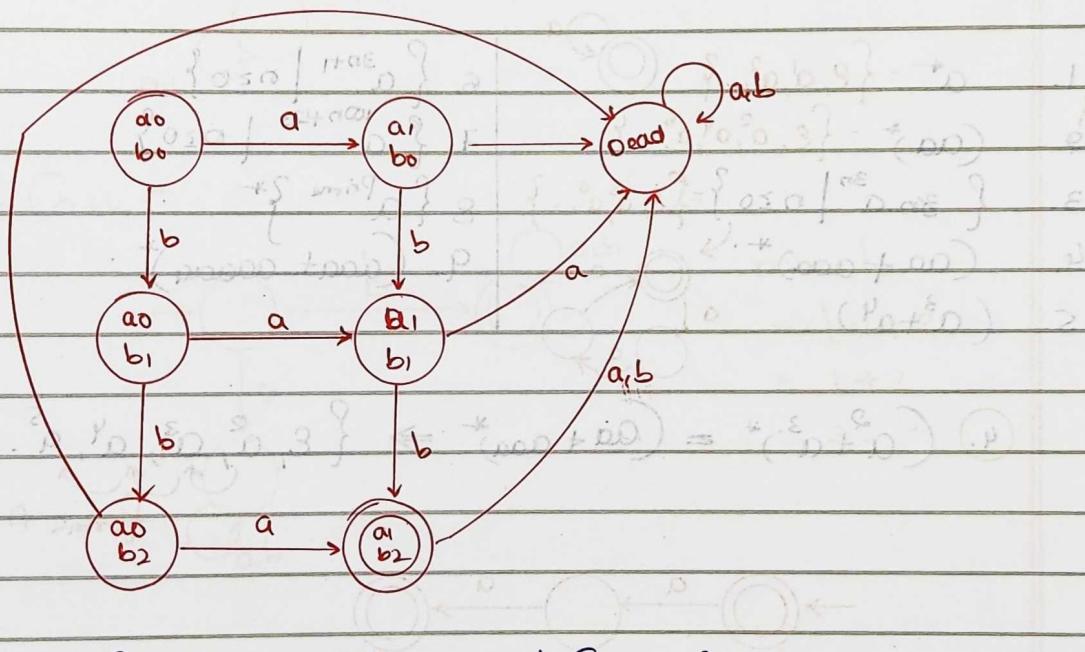
$$11 \quad a^{\text{odd}} b^{\text{odd}} = a(aa)^* b(bb)^*$$

$$= \{ab, a^3b, ab^3, a^3b^3, a^5b^5, \dots\}$$

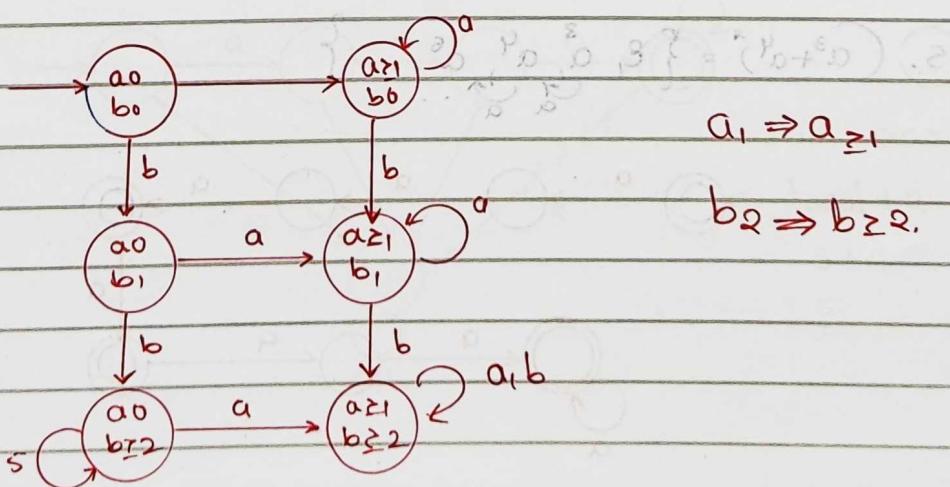


### Model VIII (Solutions)

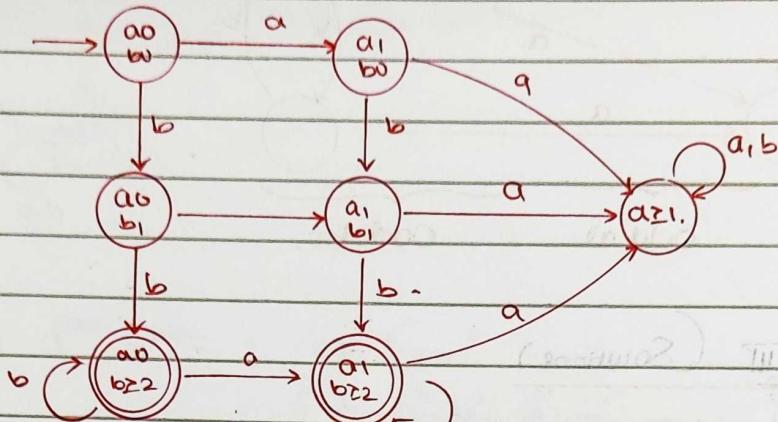
1.  $\{ w \mid w \in \{ab\}^*, n_a(w)=1, n_b(w)=2 \}$   
 $= \{ abb, bab, bba \}$



2.  $\{ w \mid w \in \{a, b\}^*, n_a(w) \geq 1, n_b(w) \geq 2 \}$



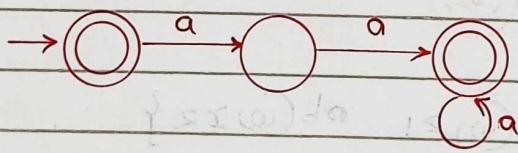
3.  $\{ w \mid w \in \{a, b\}^*, \underbrace{n_a(w) \leq 1}_{a_0, a_1}, \underbrace{n_b(w) \geq 2}_{b_0, b_1, b_2} \}$



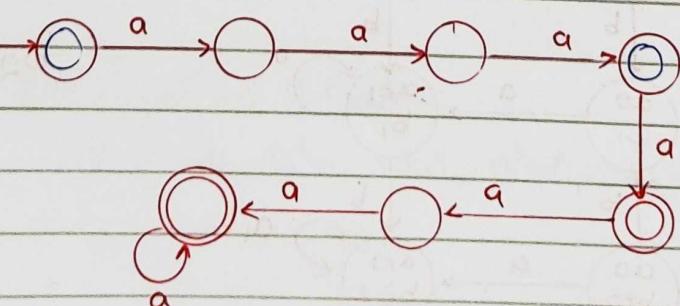
Model IV [Language over 1 symbol]

- |   |                                    |
|---|------------------------------------|
| 1. $a^* = \{\epsilon, a, a^2, \dots\}$  | $a$                                |
| 2. $(aa)^* = \{\epsilon, a^2, a^4, a^6, \dots\}$                                    | $a^{3n+1} \mid n \geq 0\}$         |
| 3. $\{\underbrace{a^n a^{3n}}_{3n} \mid n \geq 0\} = \{\epsilon, a^3, a^6, \dots\}$ | 7. $\{a^{100n+23} \mid n \geq 0\}$ |
| 4. $(aa + aaa)^*$   | 8. $\{a^{\text{prime}}\}^*$        |
| 5. $(a^3 + a^4)$  | 9. $(aaa + aaaa)^*$                |

4.  $(a^2 + a^3)^* = (aa + aaa)^* \Rightarrow \{\epsilon, a^2, a^3, a^4, a^5, \dots\}$   
forme AP.



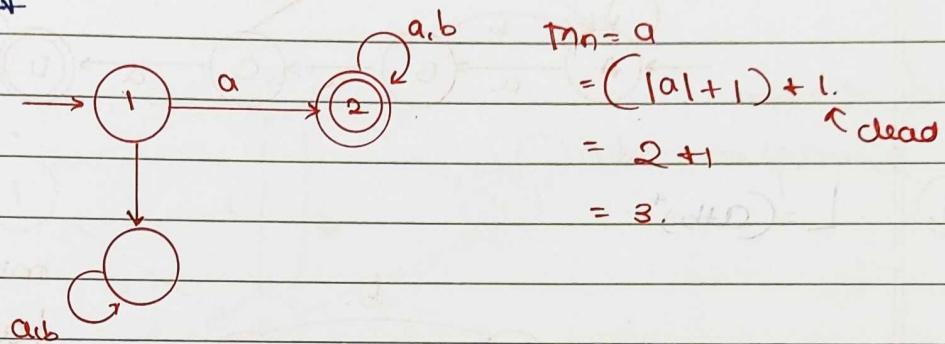
5.  $(a^3 + a^4)^* = \{\epsilon, a^3, a^4, a^5, a^6, \dots\}$



7.  
 $\{a^{\text{Prime}}\}^* = \{a^2, a^3, a^5, a^7, \dots\}$   
 $= \{\epsilon, a^2, a^3, a^4, a^5, \dots\}$   
 $= \{a^*\}$   
 $= \{\epsilon + aaa^*\}$   
 $= \{aa\} = (a^2 + a^3)^*$

Module 2

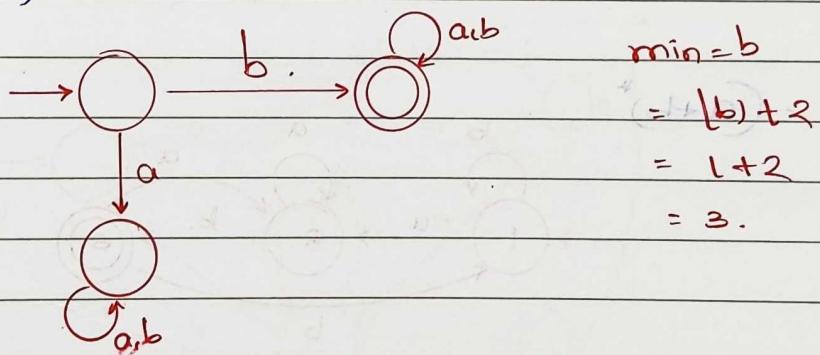
1.  $L = a(a+b)^*$



$$\begin{aligned} m_n &= a \\ &= (|a|+1)+1 \\ &= 2+1 \\ &= 3. \end{aligned}$$

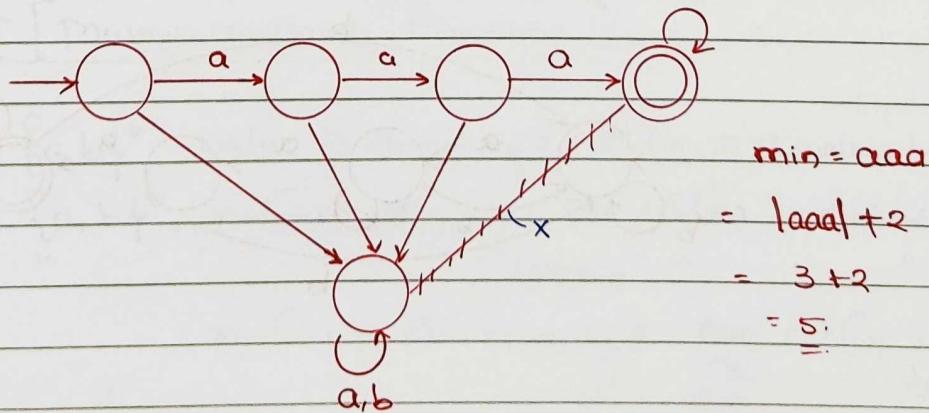
↑ dead

2.  $L = b(a+b)^*$



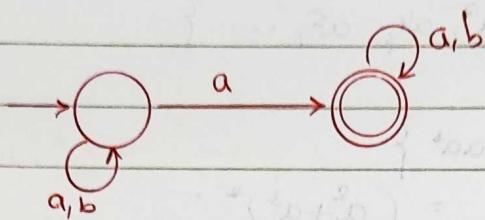
$$\begin{aligned} m_n &= b \\ &= (|b|+1)+2 \\ &= 1+2 \\ &= 3. \end{aligned}$$

3.  $L = aaa(a+b)^*$



$$\begin{aligned} m_n &= aaa \\ &= |aaa|+2 \\ &= 3+2 \\ &= 5. \end{aligned}$$

4.  $L = (a+b)^* a (a+b)^*$ .

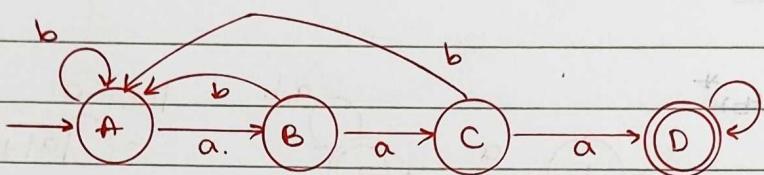


$\text{min} = a$

$|a| + 1 = 1 + 1$

= 2 states.

5.  $L = (a+b)^* aaa (a+b)^*$ .

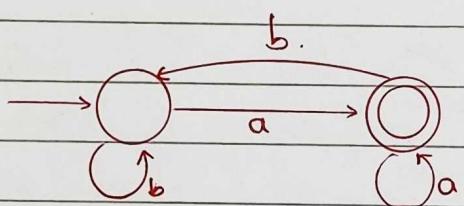


$\text{min} = aaa$

$|aaa| + 1$

= 3 + 1  
= 4.

6.  $L = (a+b)^*$ .

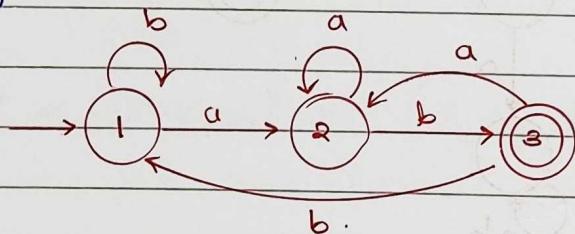


$\text{min} = a$

$|a| + 1$

= 2 states.

7.  $L = (a+b)^*$

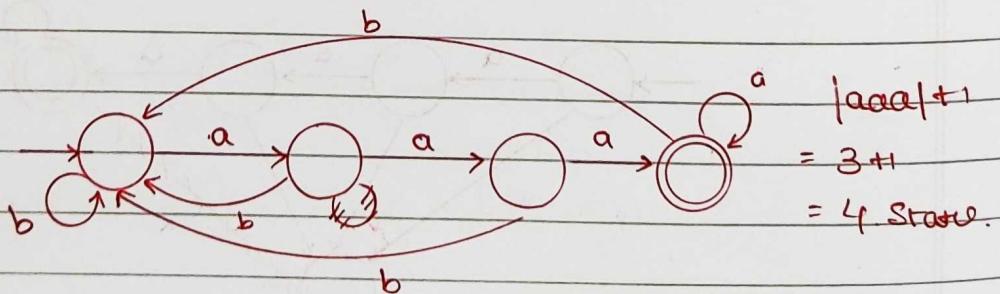


$\text{min} = ab$

$|ab| = 1$

2 + 1  
= 3 States.

8.  $L = (a+b)^* aaa$

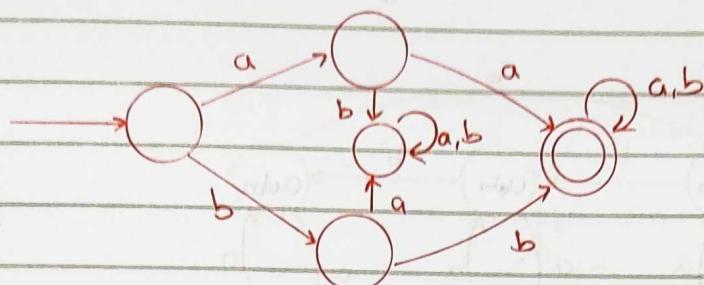


$|aaa| + 1$

$= 3 + 1$

= 4 States.

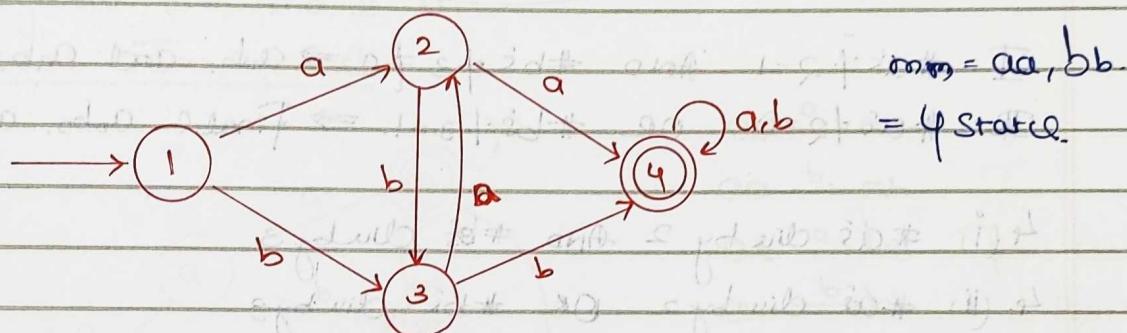
(9)  $L = (aa+bb)(a+b)^*$ .



$min = aa, bb$

= 5 States.

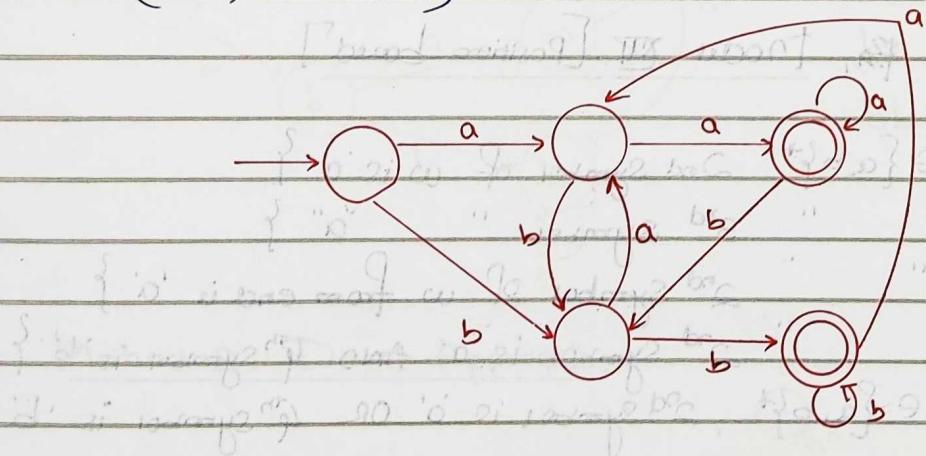
(10)  $L = (a+b)^*(aa+bb)(a+b)^*$ .



$min = aa, bb$

= 4 States.

(11)  $L = (a+b)^*(aa+bb)$

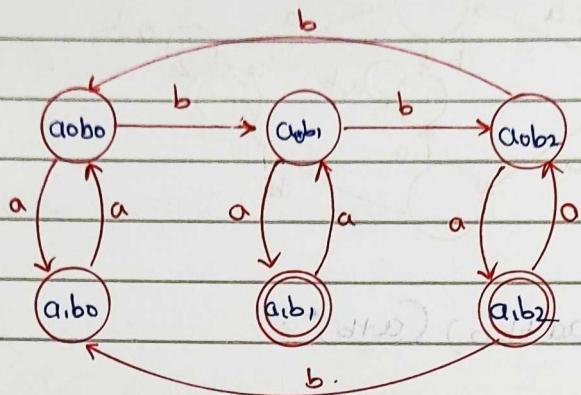


### Modulo - XI [Multiple conditions, Remainder]

1.  $\{w \mid w \in \{a,b\}^*, na(w) \text{ is divisible by } 2, nb(w) \text{ is divisible by } 2\}$
2.  $\{w \mid w \in \{a,b\}^*, na(w) \text{ is divisible by } 2 \text{ or } nb(w) \text{ is divisible by } 2\}$
3.  $\{\text{but not } \dots \text{ but not } \dots\}$
4.  $\{\text{na}(w) \text{ is divisible by } 3 \text{ AND nb}(w) \text{ is divisible by } 3 \text{ OR nb}(w) \text{ is divisible by } 3\}$

4.  $\{w \mid w \in \{a, b\}^*, \text{ #a}(w) \text{ is div by } 2 \text{ AND } \text{#a}(w) \text{ is div by } 3\}$ .

Method 1.



I: #a's / 2 = 1 AND #b's / 3 = 0  $\Rightarrow$  a1b, and a1b2 are final.

II: #a's / 2 = 0 OR #b's / 3 > 1  $\Rightarrow$  Final. a1bo, a1b1, ...

4(i) #a's div by 2 AND #b's div by 3

4(ii) #a's div by 2 OR #b's div by 3

4(iii), #a's div by 2 But Not #b's div by 3.

Done. Model 11. [Position based]

1.  $L = \{w \mid w \in \{a, b\}^*, \text{ 2nd symbol of } w \text{ is 'a'}\}$

2.  $L = \{w \mid \dots \text{ 3rd symbol } "a"\}$

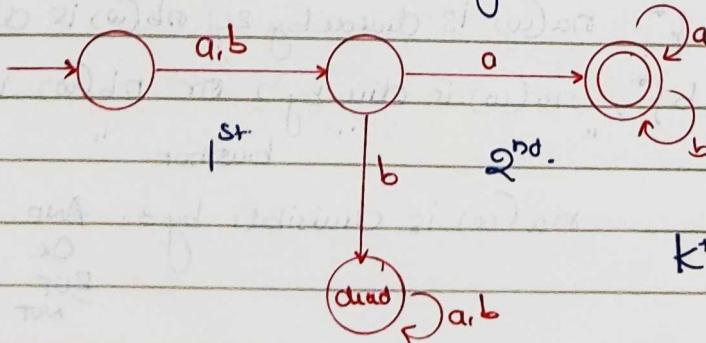
3.  $L = \{w \mid \dots \text{ 2nd symbol of } w \text{ from end is 'a'}\}$

4.  $L = \{w \mid \dots \text{ 2nd symbol is 'a' AND 4th symbol is 'b'}\}$

5.  $L = \{w \mid w \in \{a, b\}^*, \text{ 2nd symbol is 'a' OR 4th symbol is 'b'}\}$

1.  $(a+b)a(a+b)^*$

2<sup>nd</sup> symbol is 'a'.



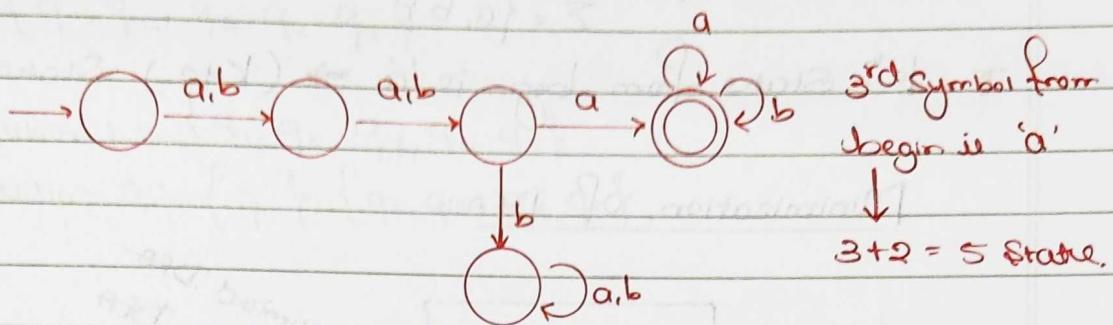
$$\min = aa$$

or

ba

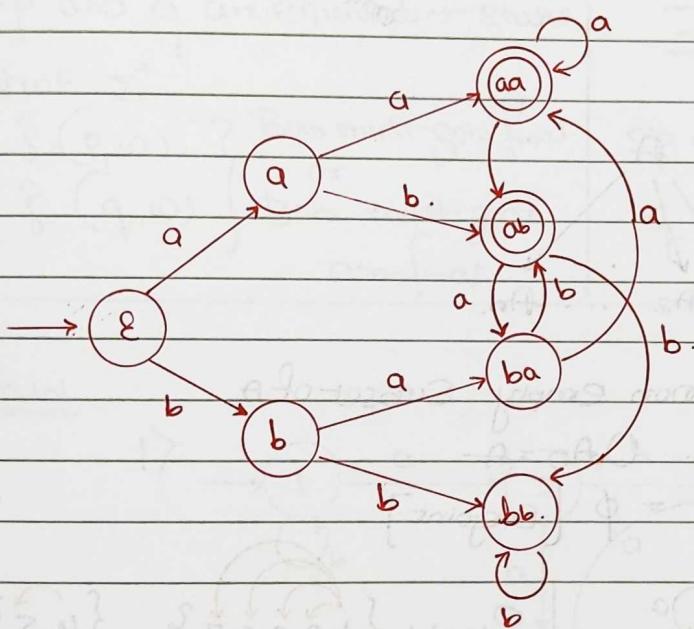
k<sup>th</sup> symbol is 'a'.

$$2. L = (a+b)(a+b) a (a+b)^*$$



3. 2<sup>nd</sup> Symbol from end ie 'a' :

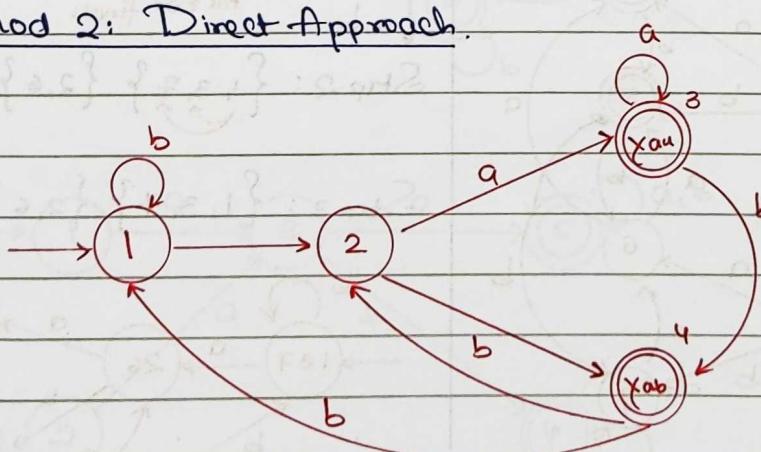
Method 1 : Tree Approach:



aa  $\xrightarrow{a}$  aa  
 aa  $\xrightarrow{b}$  ab  
 ab  $\xrightarrow{a}$  ba  
 ab  $\xrightarrow{b}$  bb  
 ba  $\xrightarrow{a}$  aa  
 ba  $\xrightarrow{b}$  ab  
 bb  $\xrightarrow{a}$  ba  
 bb  $\xrightarrow{b}$  bb

Traces last 2 symbols

Method 2: Direct Approach.

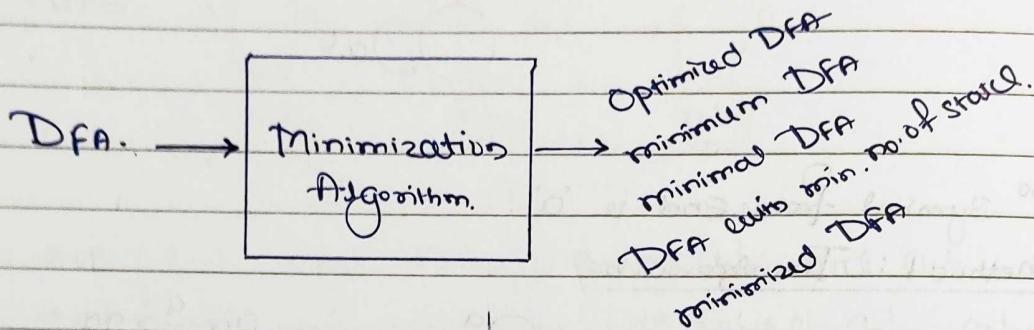


- 1: waits for a  $\frac{a}{b}$
- 2: waits for a/b
- 3: xaa
- 4: xab

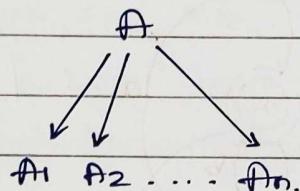
Note: I.  $K^{\text{th}}$  Symbol from end is 'd'  $\rightarrow 2^k$  state in DFA  
 $\Sigma = \{a, b\}$ .

II.  $K^{\text{th}}$  State from begin is 'a'  $\rightarrow (K+2)$  State.

### Minimization of DFA:



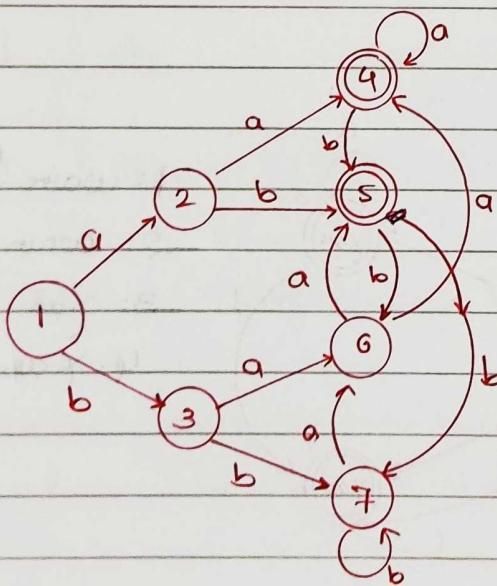
### Partition on Set A.



I. Every  $A_i$  is non empty subset of A

II.  $A_1 \cup A_2 \cup \dots \cup A_n = A$

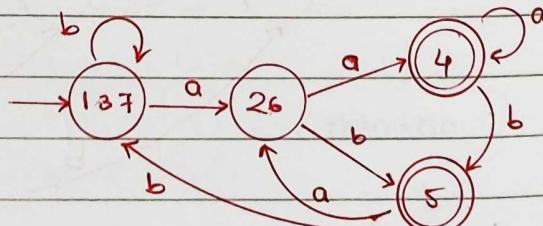
III.  $\forall i, j, A_i \cap A_j = \emptyset$  [Disjoint]



Step 1:  $\{1, 2, 3, 6, 7\}$   $\{4, 5\}$   
 ↓ a, b      All non finals      All finals.

Step 2:  $\{1, 3, 7\}$   $\{2, 6\}$   $\{4\}$   $\{5\}$ .

Step 3:  $\{1, 3, 7\}$   $\{2, 6\}$   $\{4\}$   $\{5\}$



### Partitions on Set Q

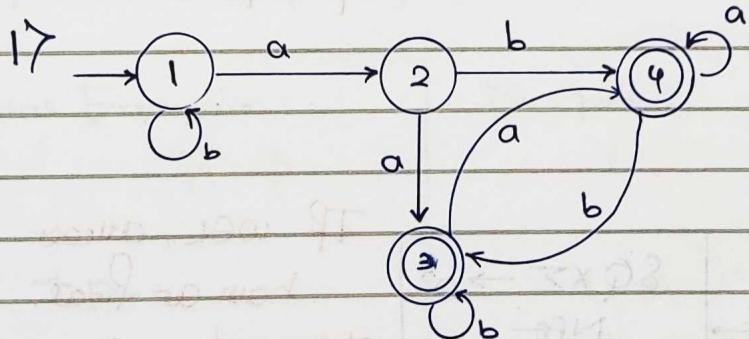
$$Q = \{q_1, q_2, q_3, q_4, q_5\}$$

Partition 1:  $\{q_1, q_2, q_3, q_4, q_5\}$

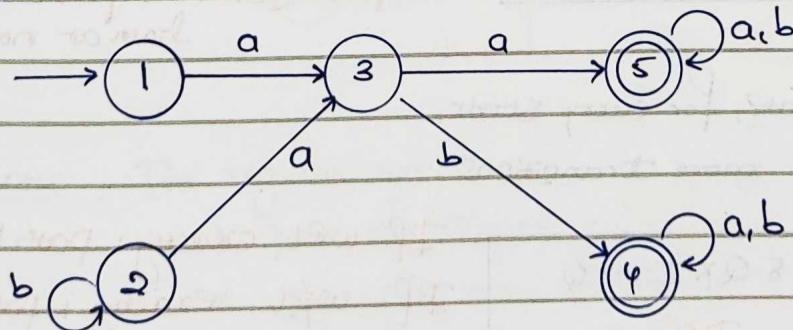
Partition 2:  $\{q_1\} \cup \{q_2, q_3, q_4, q_5\}$ .

Equivalent State. (non distinguishable)	Distinguishable State. (non equivalent)
<p><math>p</math> and <math>q</math> are equivalent states.</p> <p><math>t \in \Sigma^*</math></p> <p><math>\hat{s}(p, t) \cap \hat{s}(q, t) \neq \emptyset</math> Both must goto final or Both must goto non-final.</p>	<p><math>\exists w \in \Sigma^*</math></p> <p><math>p \xrightarrow{w} \text{final} \quad \text{or} \quad p \xrightarrow{w} \text{non final}</math></p> <p><math>q \xrightarrow{w} \text{final} \quad \text{or} \quad q \xrightarrow{w} \text{non final}</math></p>

t.i.w



2>



Model - XIII [Special]

1.  $\{w \mid w \in \{0,1\}^*, \text{ Decimal}(w) \text{ is divisible by } 8\}$

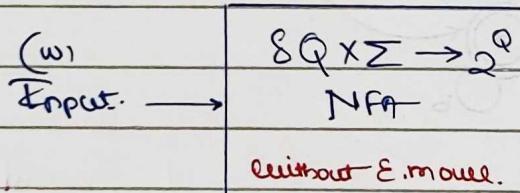
Div by  $2^k$

$2^3 = 3+1 = 4$  states.

2.  $\{w \mid w \in \{0,1\}^*, \#_0(w) = \#_{10}(w)\}$   
 $= \{\epsilon, 0, 1, 00, 11, 000, \dots\}$

3.  $\{w \mid w \in \{0,1\}^*, \text{ every prefix } p \text{ of } w \text{ satisfied } |\#_0(p) - \#_{10}(p)| \leq 1\}$   
 $\{\epsilon, 0, \dots\}$

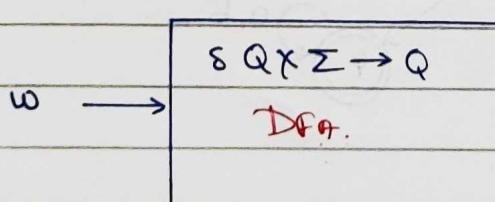
w	prefix	Condition
$\epsilon$	$\epsilon$	✓
0	0	$ 1-0  = 1$ ✓
00	00	$ 2-0  = 2$

NFA

If  $w \in L$ , atleast 1 path that have at final.

If  $w \notin L$ , there is no path have at final (either no path or every path have at non-final).

From every state, for every state,  
Zero or more transitions.



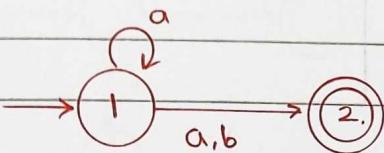
If  $w \in L$ , exactly 1 path have at final.  
If  $w \notin L$ , exactly 1 path have at non-final.

For every state, for every ip symbol, exactly 1 transition.

Note.

- I. Every DFA is NFA
- II. NFA may or may not be DFA
- III. Min. DFA need not be min NFA
- IV. For every regular language:  
 $n(\text{min DFA}) \geq n(\text{min NFA})$

eg:



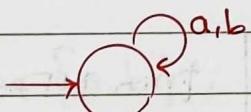
	No. of paths	valid / invalid.
e.	1	invalid
a	2	valid
b	1	valid
aa	2	valid
ab	1	valid
ba	0	invalid.

1.  $L = \{\}$  over  $\Sigma = \{a, b\}$ .

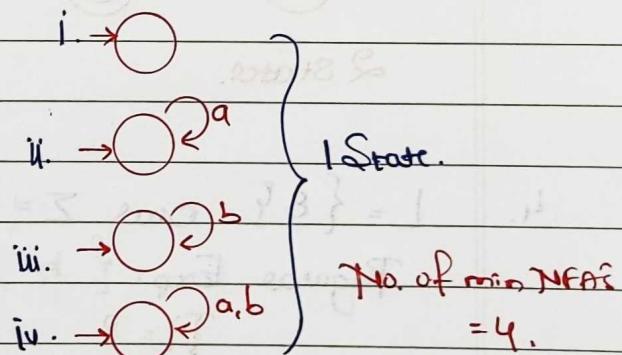
Regular Expression:

$$R = \emptyset$$

Min. DFA:



Min. NFA:



Note: For regular set,

i. No. of DFAs = Infinite

ii. No. of min DFAs = 1

iii. No. of NFAs = Infinite

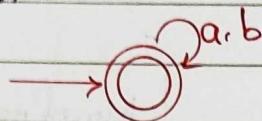
iv. No. of min NFAs = 1 or more.

2.  $L = \Sigma^*$  over  $\Sigma = \{a, b\}$

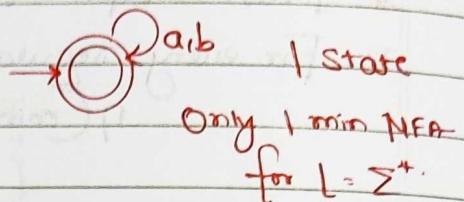
Regular Expressions:

$$\begin{aligned} R &= (a+b)^* \\ &= (a^*b^*)^* \cdot (b^*a^*)^* \end{aligned}$$

min DFA:



min NFA:

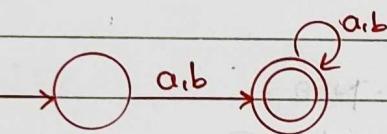


3.  $L = \Sigma^*$  over  $\Sigma = \{a, b\}$ .

Regular Expressions:

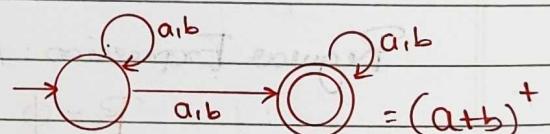
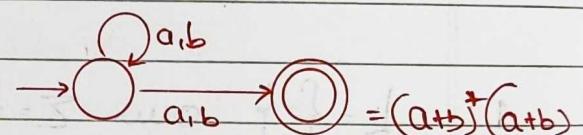
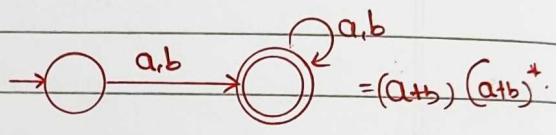
$$\begin{aligned} R &= (a+b)^+ = (a+b) \cdot (a+b)^* \\ &= (a+b)^* \cdot (a+b) \end{aligned}$$

min. DFA:



2 States.

min NFA: 2 states.

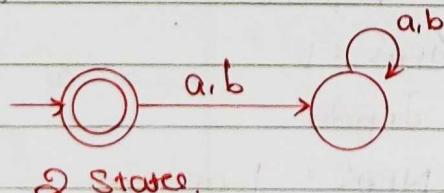


4.  $L = \{\epsilon\}$  over  $\Sigma = \{a, b\}$ .

Regular Exp:

$$\begin{aligned} R &= \epsilon \\ &= \phi^* = \epsilon^* = \epsilon^+ \end{aligned}$$

min DFA:



2 States.

min NFA



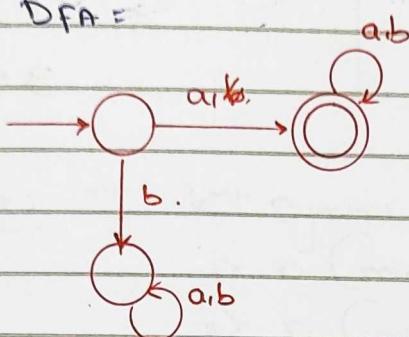
1 State.

5.  $L = \{w \mid w \in \{a,b\}^*, w \text{ starts with 'a'}\}$

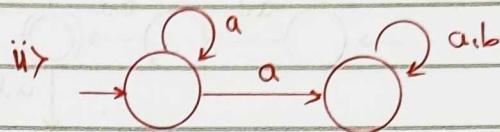
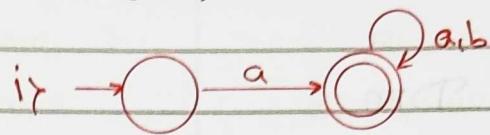
Regular Expression:

$$\begin{aligned} R &= a(a+b)^* \\ &= (ab^*)^+ \end{aligned}$$

Min DFA =



NFA (min):

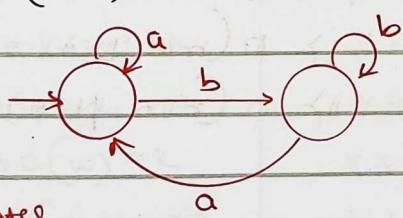


6.  $L = \{w \mid w \in \{a,b\}^*, w \text{ ends with 'b'}\}$

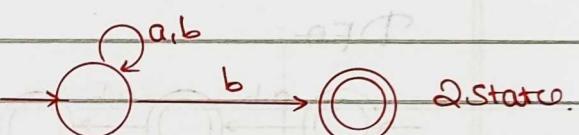
Regular Expression:

$$\begin{aligned} R &= (a+b)^* b \\ &= (a^* b)^+ \end{aligned}$$

DFA (min):



NFA (min):

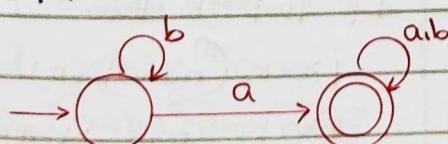


7.  $L = \{w \mid w \in \{a,b\}^*, w \text{ contains 'd'}\}$

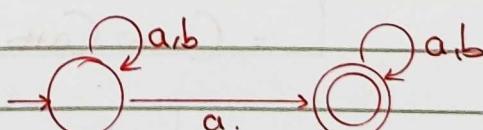
Regular Expression:

$$\begin{aligned} R &= \Sigma^* a \Sigma^* \\ &= b^* a (a+b)^* \end{aligned}$$

DFA:



NFA:



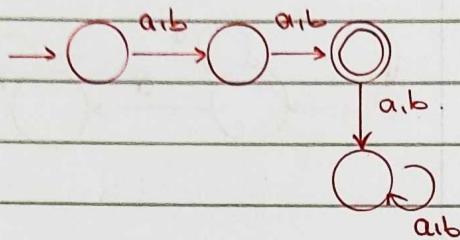
2 states.

8.  $L = \{w \mid w \in \{a,b\}^*, |w| \leq 2\}$

Regular Expression:

$$= (\epsilon + ab)^2$$

DFA:



NFA:



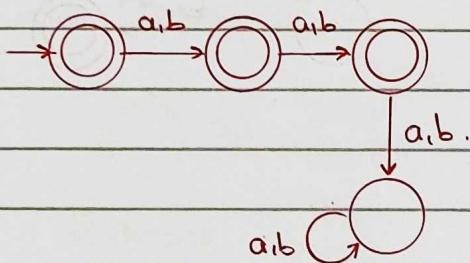
3 States

9.  $L = \{w \mid w \in \{a,b\}^*, |w| \leq 2\}$

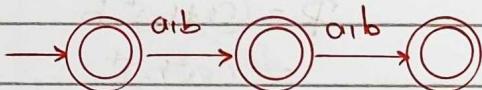
Regular Expression:

$$= (\epsilon + ab)^2$$

DFA:



NFA:



3 States.

If  $|w| \leq k$  then

$$\text{i)} n(\text{min DFA}) = k+2$$

$$\text{ii)} n(\text{min NFA}) = k+1,$$

10.  $L = \{w \mid w \in \{a,b\}^*, |w| \geq 2\}$

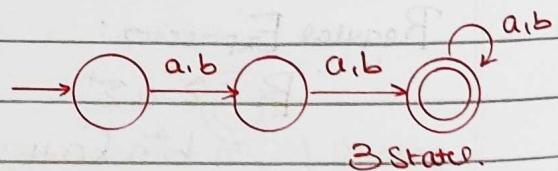
Regular Expression:

$$= (ab)^2 (ab)^*$$

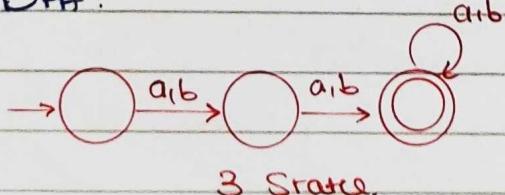
$$= (ab)^* (ab)^2$$

$$= (ab) (ab)^* (ab)$$

NFA:



DFA:



3 States.

If  $|w| \geq k$  then

$$\text{i)} n(\text{min DFA}) = k+1$$

$$\text{ii)} n(\text{min NFA}) = k+1.$$

	Min DFA	Min NFA
11. $L = (a+b)^2 a (a+b)^*$	5 States	4 States.
12. $L = (a+b)^* a (a+b)^*$	8 States	4 States.
13. $b^* a b^* a b^*$	4 States	3 States
14. $b^* (a+c) b^* (a+c) b^*$	4 States	3 States
15. $\Sigma^* a \Sigma^* a \Sigma^*$	3 States.	3 States,

Note: I.  $k^{th}$  symbol from begin 'a'  $\Sigma = \{a, b\}$   
 i)  $n(\text{min DFA}) = k+2$   
 ii)  $n(\text{min NFA}) = k+1$ .

II.  $k^{th}$  symbol from end is 'a'  
 i)  $n(\text{min DFA}) = 2^k$   
 ii)  $n(\text{min NFA}) = k+1$ .

$w \in \{a, b\}^*$	Min DFA	Min NFA
I. $ w  = k =$	$k+2$	$k+1$
II. $ w  \leq k \Rightarrow$	$k+2$	$k+1$
III. $ w  \geq k \Rightarrow$	$k+1$	$k+1$
IV. $\#a(w) = k$	$k+2$	$k+1$
V. $\#a(w) \leq k \Rightarrow$	$k+2$	$k+1$
VI. $\#a(w) \geq k \Rightarrow$	$k+1$	$k+1$
VII. $k^{th}$ symbol is 'a'	$k+1$	$k+1$
VIII. $k^{th}$ symbol from end is 'a'	$2^k$	$k+1$

16. $L = aaa (a+b)^*$	5	4
17. $L = (a+b)^* aaa$	4	4
18. $L = (a+b)^* aaa (a+b)^*$	4	4
19. $L = (a+b)^2 a (a+b)^3 b (a+b)^*$	9	8

20.  $L = (a+b)a(a+b)^7 a$   
 $(a+b)^*$ .

12

11.

21.  $L = (ab)^*$ .

3 States

2 states.

22.  $L = (ba)^*$ .

3 States

2 states.

23.  $L = a(ba)^*$ .

3 States

2 states.

NFA

without  $\epsilon$  moves.

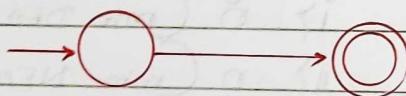
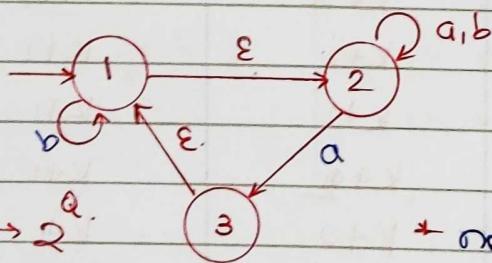
$$\delta : Q \times \Sigma \rightarrow 2^Q$$

not "NFA without  
 $\epsilon$  moves."

NFA

with  $\epsilon$  moves.

$$\delta : Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q$$

NFA with  $\epsilon$  moves.

$$\delta : Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q$$

\* not DFA

\* not NFA without  $\epsilon$  moves

\* It is NFA.

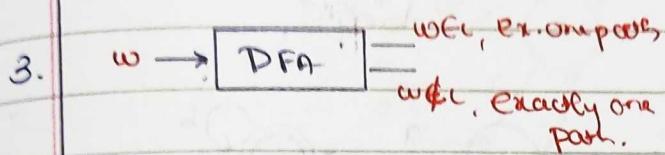
$(1, a)$	$\emptyset$
$(1, b)$	$\{1\}$
$(1, \epsilon)$	$\{2\}$
$(2, a)$	$\{3\}$
$(2, b)$	$\{1, 2\}$
$(2, \epsilon)$	$\{1, 3\}$
$(3, a)$	$\{2, 3\}$
$(3, b)$	$\{1, 2, 3\}$
$(3, \epsilon)$	

## DFA

## NFA

1. Every DFA is NFA

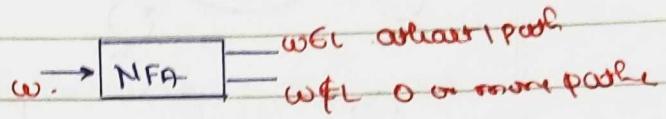
$$\delta: Q \times \Sigma \rightarrow Q$$



4. For every regular, unique min DFA exists.

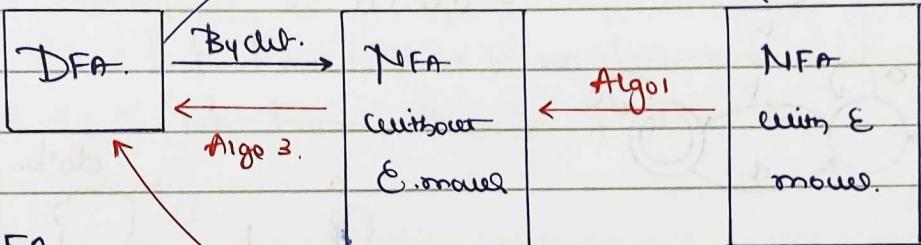
NFA may or may not be DFA

$$\delta: Q \times \Sigma \rightarrow 2^Q$$



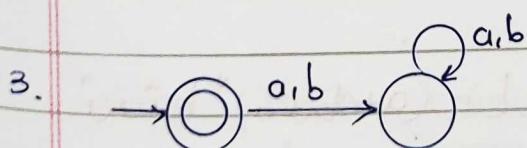
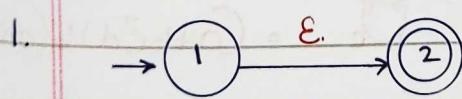
For every regular, one or more min NFA's exist.

By def.



$$\text{DFA} = \text{NFA}$$

NFA with  $\epsilon$  move:



$$\delta: Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q$$

$$\delta(\textcircled{1}, a) = \{\textcircled{2}\}$$

$$\delta(\textcircled{1}, b) = \{\textcircled{2}\}$$

$$\delta(\textcircled{1}, \epsilon) = \{\textcircled{1}\}$$

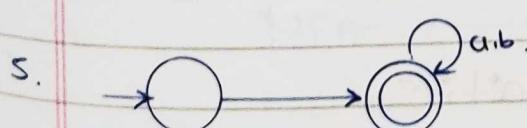
$$\delta(\textcircled{2}, a) = \{\textcircled{2}\}$$

$$\delta(\textcircled{2}, b) = \{\textcircled{2}\}$$

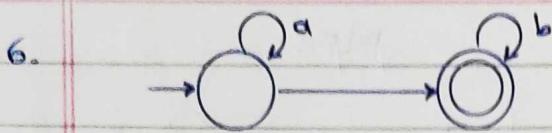
$$\delta(\textcircled{2}, \epsilon) = \{\textcircled{2}\}$$



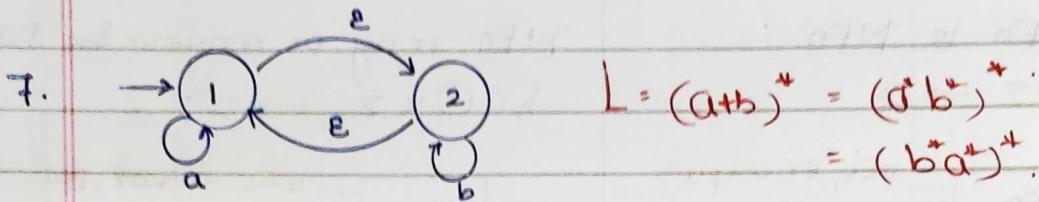
$$L = (a+b)^*$$



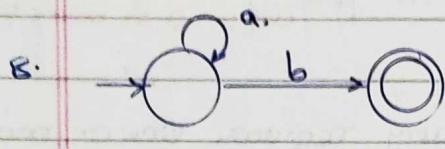
$$L = (a+b)^*$$



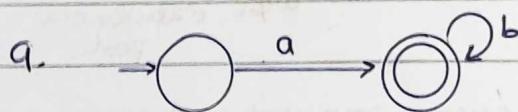
$$L = a^* b^*$$



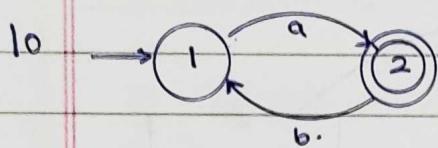
$$\begin{aligned} L &= (a+b)^* = (a^* b^*)^* \\ &= (b^* a^*)^* \end{aligned}$$



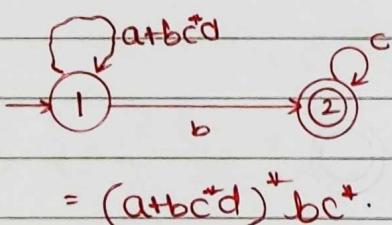
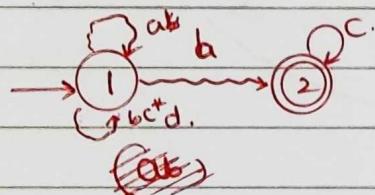
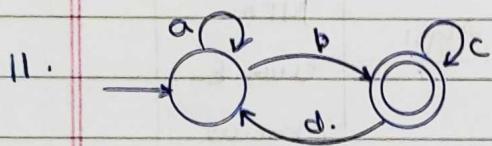
$$L = a^* b$$



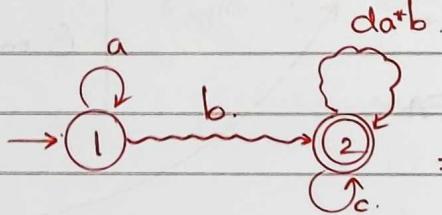
$$L = ab^*$$



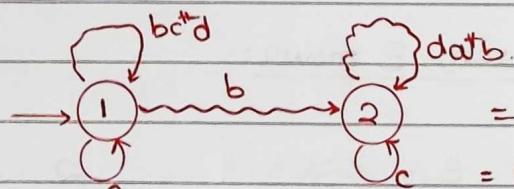
$$\begin{aligned} L &= \{ \bar{a}, \bar{a}\bar{b}a, \bar{a}\bar{b}\bar{a}\bar{b}a \dots \} = a(ba)^* \\ &= (ab)^* a \end{aligned}$$



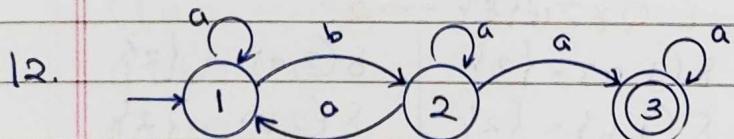
$$= (a+b+c+d)^* bc^*$$



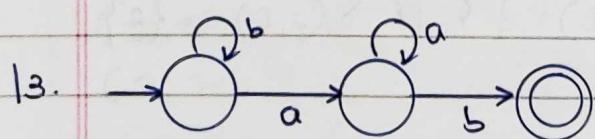
$$= ab(da^* b + c)^*$$



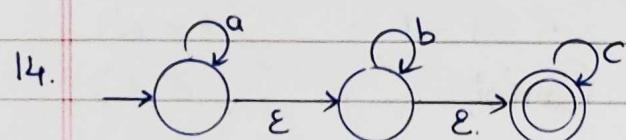
$$\begin{aligned} &= \bar{a}b (a\bar{a}b^* c)^* \\ &= (a+b+c)d b (da^* b + c)^* \end{aligned}$$



$$L = (a+ba^* a)^* b a^* a a^*$$



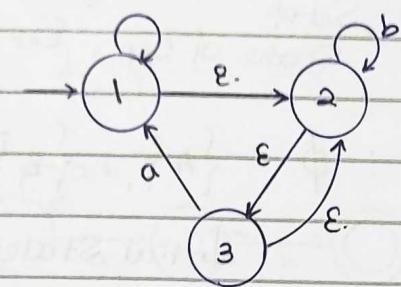
$$\begin{aligned} L &= b^* a^* b \\ &= b^* a^+ b \end{aligned}$$



$$L = a^* b^* c^*$$

NFA with E moves:

$\epsilon$ -closure (state):



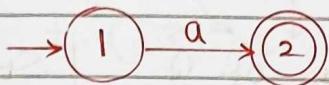
$$\epsilon\text{-closure}(1) = \{1, 2, 3\}$$

$$\epsilon\text{-closure}(2) = \{2, 3\}$$

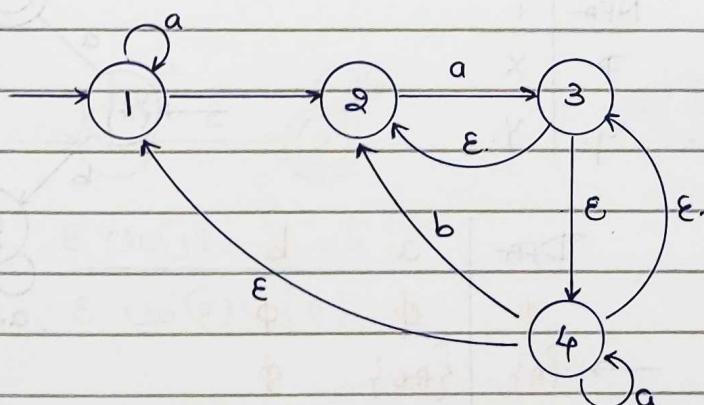
$$\epsilon\text{-closure}(3) = \{3, 2\}.$$

$\epsilon\text{-closure}(q) = \hat{\delta}(q, \epsilon)$   
 = Set of all states reachable  
 from state  $q$  without reading any  
 string.

e.g.:

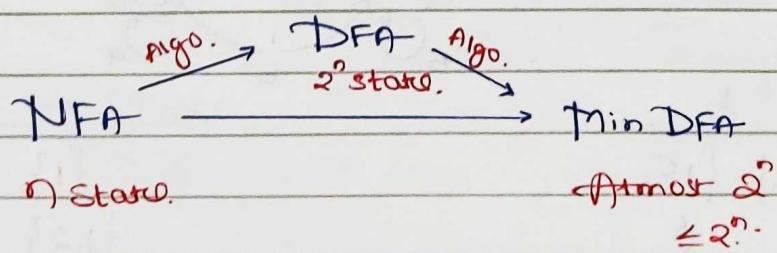


Transition Symbol |  $\epsilon$   
 $\delta(1, \epsilon) = \{\}$  From 1, there is no transition on  $\epsilon$ .  
 $\hat{\delta}(1, \epsilon) = \{1\}$  From 1, there is path on  $\epsilon$ .  
 Extended transitions.

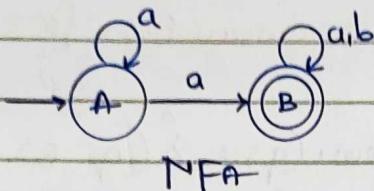


$$\begin{aligned}\epsilon\text{-closure}(1) &= \{1, 2\} \\ \epsilon\text{-closure}(2) &= \{2\} \\ \epsilon\text{-closure}(3) &= \{3, 1, 2, 4\} \\ \epsilon\text{-closure}(4) &= \{4, 1, 2, 3\}.\end{aligned}$$

→ Algo: NFA without  $\epsilon$  moves  $\Rightarrow$  DFA  
 , (Subset Construction)



Given NFA  $\rightarrow$  DFA  
 $(Q, \Sigma, \delta_{NFA}, q_0, F) \rightarrow (Q^*, \Sigma, \delta_{DFA}, \{q_0\}, F')$



Set of

States of DFA = Set of all Subsets of Q.

 $: \emptyset, \{A\}, \{B\}, \{AB\}$ 

2 States

$Q = \{A, B\}$

Initial States in DFA

$Q^* = \{\emptyset, \{A\}, \{B\}, \{A, B\}\}$

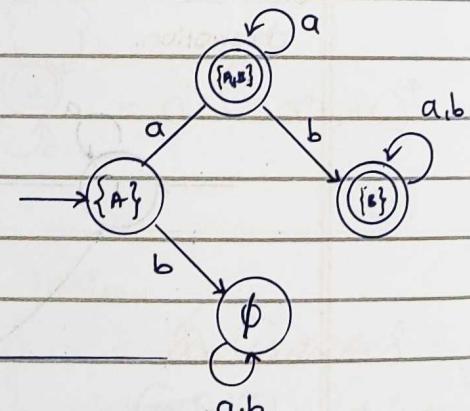
Subset of Q

Subset in DFA

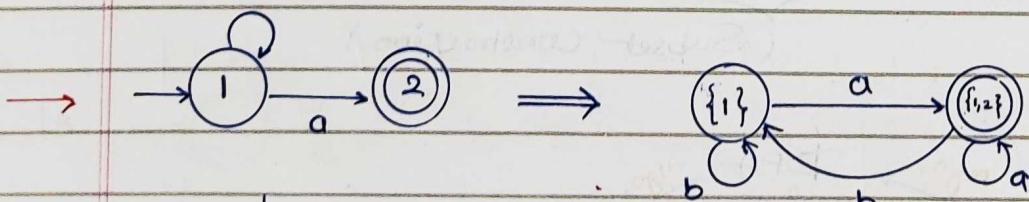
$\rightarrow \delta_{DFA}(\{p, q\}, i) = \delta_{NFA}(p, i) \cup \delta_{NFA}(q, i)$

$= X \cup Y$

NFA	i
p	x
q	y

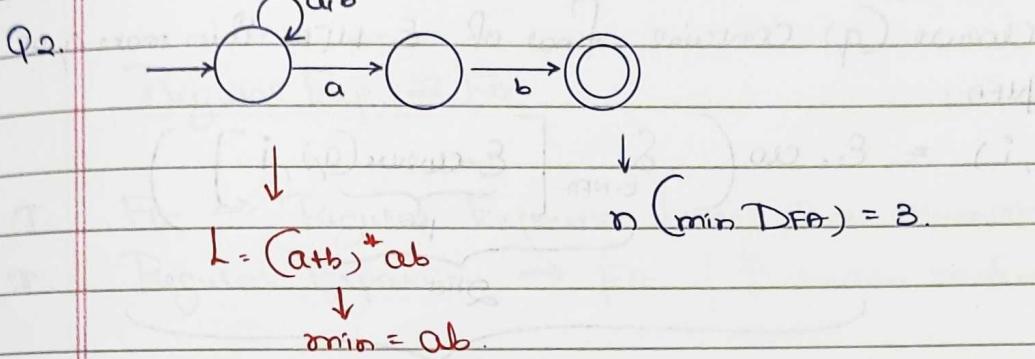
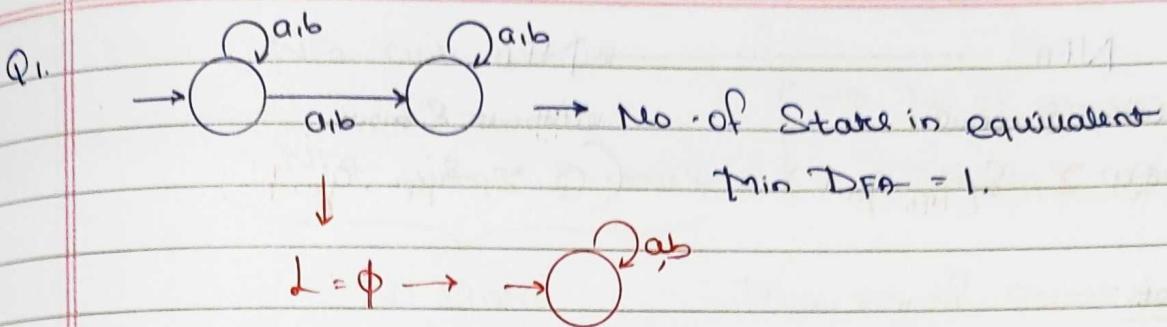


NFA	a	b	DFA	a	b
$\rightarrow A$	$\{A, B\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow B$	$\{B\}$	$\{B\}$	$\{A\}$	$\{AB\}$	$\emptyset$
			$\ast \{B\}$	$\{B\}$	$\{B\}$
			$\ast \{A, B\}$	$\{A, B\}$	$\{B\}$



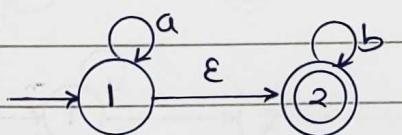
NFA	a	b
1	$\{1, 2\}$	$\{1\}$
2	$\emptyset$	$\emptyset$

	a	b
1	$\{1\}$	$\{1, 2\}$
2	$\{1, 2\}$	$\{1\}$



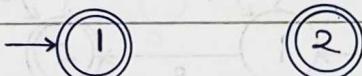
$\rightarrow$  NFA  $\xrightarrow{\text{without } \epsilon\text{-move.}}$  NFA  $\xrightarrow{\text{without } \epsilon\text{-move.}}$

$(Q, \Sigma, \delta_{\epsilon, \text{NFA}}, q_0, F)$   $(Q, \Sigma, \delta_{\text{NFA}}, q_0, F')$



$$\epsilon.\text{clo}(1) = \{1, 2\}$$

$$\epsilon.\text{clo}(2) = \{2\}.$$



without  $\epsilon$ -move.

$\rightarrow$  NFA  $\xrightarrow{\text{Min DFA.}}$

If  $\circ$  State present in NFA then.

$1 \leq$	no. of States in min. DFA	$\leq 2$ .
----------	------------------------------	------------

Always important.

$\rightarrow$  Min NFA  $\rightarrow$  Min DFA

$n \leq$	no. of States in min. DFA	$\leq 2^n$
----------	------------------------------	------------

$\rightarrow$  NFA  $\longrightarrow$  NFA.

with  $\epsilon$  move.

$$(Q, \Sigma, \delta_{\text{ENFA}}, q_0, F)$$

without  $\epsilon$  move.

$$(Q, \Sigma, \delta_{\text{NFA}}, q_0, F).$$

Note:

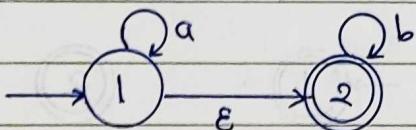
- I. Initial of NFA is same as initial of  $\epsilon$ -NFA.
- II. If  $\epsilon$ -closure ( $q_f$ ) contains final of  $\epsilon$ -NFA then make  $q_f$  as final in NFA.

III.  $\delta_{\text{NFA}}(q, i) = \epsilon \cdot \text{clo} \left( \underbrace{\delta_{\text{E.NFA}} \left[ \underbrace{\epsilon \cdot \text{closure}(q), i}_{\text{1st}} \right]}_{\text{2nd}} \right) \underbrace{\text{3rd.}}_{\text{3rd.}}$

$\epsilon \cdot \text{clo}$   
 $\epsilon, \epsilon \Sigma$ .

NFA with  $\epsilon$  move  $\longrightarrow$  DFA.

$$(Q, \Sigma, \delta_{\text{ENFA}}, q_0, F) \longrightarrow (\mathcal{Z}, \Sigma, \delta_{\text{DFA}}, \epsilon \cdot \text{clo}(q_f), F)$$



$$\delta(1, a) = \{1\}$$

$$\delta(1, b) = \emptyset. \quad \delta(2, a) = \emptyset$$

$$\delta(2, b) = \{2\}.$$

$$\epsilon \cdot \text{clo}(1) = \{1, 2\}$$

$$\epsilon \cdot \text{clo}(2) = \{2\}.$$

- Note: I. Initial state of DFA =  $\epsilon \cdot \text{clo}$  (Initial of NFA)

II.  $\delta_{\text{DFA}}(\{p, q\}, i) = \epsilon \cdot \text{clo} \left[ \delta_{\text{ENFA}}(\{p, q\}, i) \right]$

- III. Final of DFA: if any subset of  $\mathcal{Z}$  contains final of  $\epsilon$ -NFA of Q.

NFA with E-moves  
or

NFA without E-moves

$n$  States

$\Rightarrow$  No. of states in min.

DFA is at most  $2^n$ .

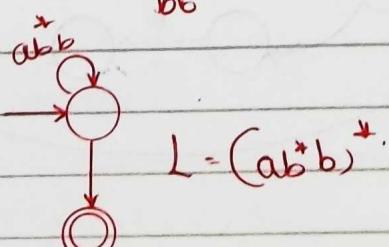
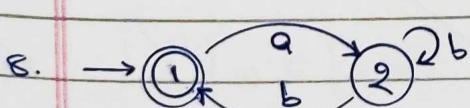
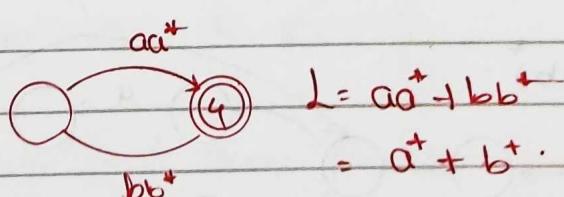
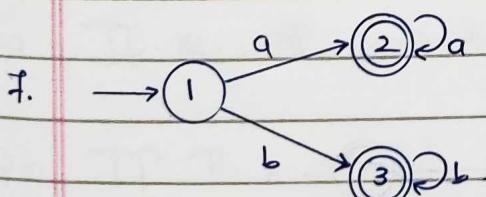
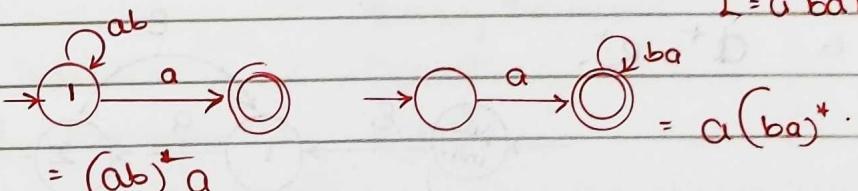
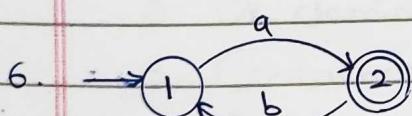
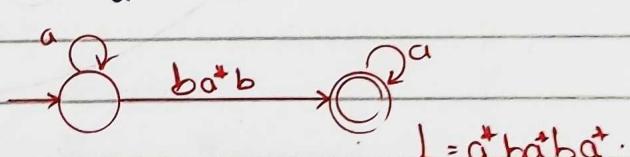
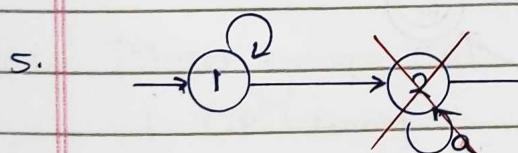
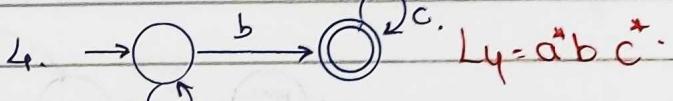
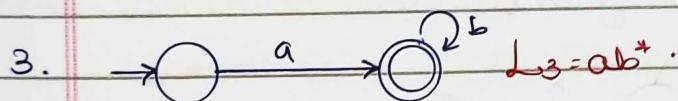
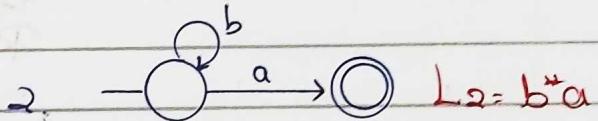
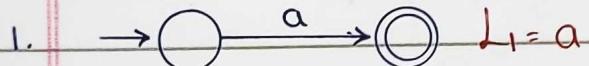
True no. of states in min DFA  
is  $2^n$

Regular Exp.  $\cong$  FA:

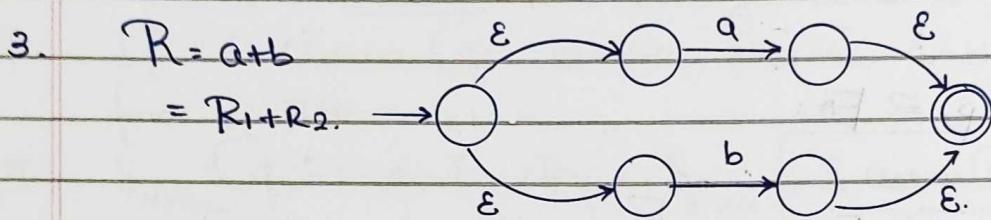
I. FA  $\Rightarrow$  Regular Expression [using State elimination]

II. Regular Expression  $\Rightarrow$  FA [Induction method].

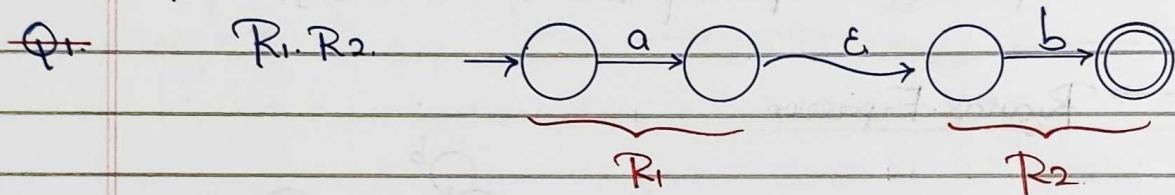
FA  $\rightarrow$  Regular Expression.



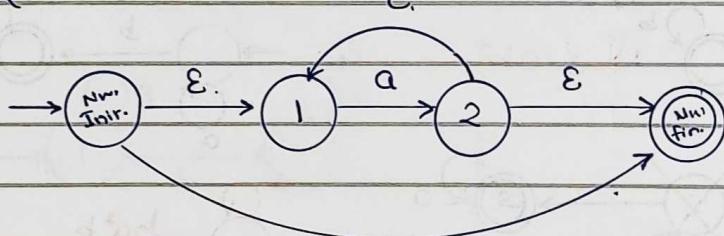
Regular Expression  $\rightarrow$  FA.



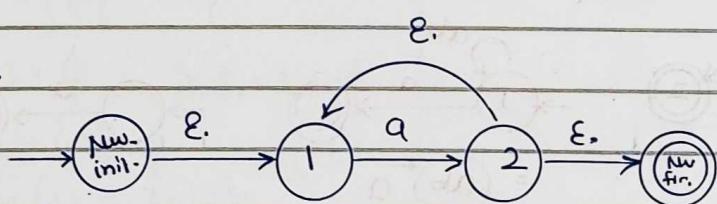
4.  $R = a.b$



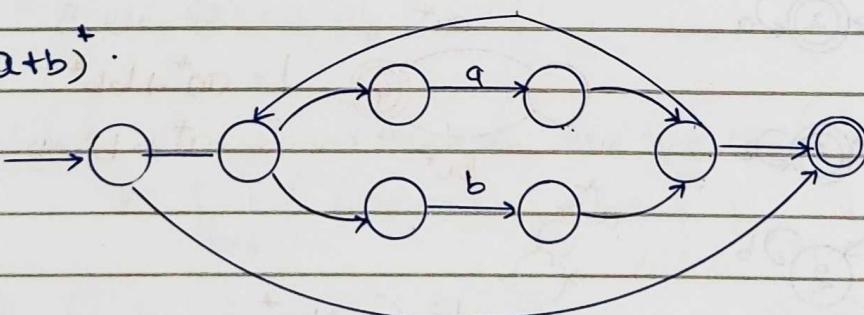
5.  $a^+ = R^+$



6.  $a^+ = R^+$



7.  $(a+b)^+$



Q1. Number of prefixes of "n" length string is  $n+1$

Q2.  $((ab)^* \cdot \phi)$  is equivalent to  $\phi$ .

Q3.  $(\phi^* \cup (bb^*)) = \underline{b^*}$ .

Q4. Which of the following is true?

$$(ab)^* 0 = a(ba)^*$$

Q5 OR operator in regular expressions satisfies  
Associative and Commutative

Q6. Concatenation Operator in regular expressions satisfies.  
Associative property

Q7. Which of the following distribution is valid in regular exp.?  
Concatenation over OR

Q8. Match groups over Sigma = {a,b}.

- 1. OR identity
  - 2. OR dominator
  - 3. Concatenation Identity
  - 4. Concatenation Dominator.
- Epsilon  
Empty Expression  
 $(ab)^*$   
 $(aa)^*$ .

Q9. If  $R_1 = a^*$ , and  $R_2 = (aa)^*$  then  $R_1 \cdot R_2 = \underline{R_1}$ .

Q10. If  $R_1 = a(aa)^*$ , and  $R_2 = (aa)^*$  then  $R_1 + R_2 = \underline{R_1^*}$ .

Q11.  $L = \{w \in \{a,b\}^*: \#_a(w) \leq 3\}$ .

$$b^* (a \cup \epsilon) b^* (a \cup \epsilon) b^* (a \cup \epsilon) b^*$$

Homework:

Find the min. string in the following expressions:

1.  $(ab)^+aaa(b+ab) \rightarrow \text{min. } abaaab$

2.  $(ab)^+aaa(b+\epsilon) \rightarrow aaa$

3.  $(a+ab)^+(bb+aaa)^+(ab)^* \rightarrow abb$

4.  $(ab)^* + \epsilon \rightarrow \epsilon$

5.  $(ab)(a+b)^+aaa + ba \rightarrow ba$

6.  $[(ab^* + a^* + ba)ab]^+ (a+b)^* \rightarrow aab$

7.  $\left[ ((ab)^*a)^*aaa \right]^+ \rightarrow aaa$

8.  $\left[ ((aba)^*aba)^*ab \right]^* \rightarrow \epsilon$

9.  $(a^*)^* \rightarrow \epsilon$

10.  $((ab)^*a^*)^*bb^*aa^* \rightarrow bbb^*aa^*$

11.  $[(ab)^*aba]^*aaa \rightarrow ababaaaa$

12.  $(aaa^*ab^*) \rightarrow aaab$

13.  $(a+aaa+aa) \rightarrow a$

14.  $a^*b^* \rightarrow ab$

15.  $a^*b^* \rightarrow \epsilon$

16.  $a^*b^*c^*d^* \rightarrow c$

17.  $(a+b^*+c^*)^* \rightarrow \epsilon$

18.  $(a+b+c)^* \rightarrow \epsilon$

Q12.  $L = \{w \in \{a,b\}^*: \#a(w) \geq 3\}$

$(aub)^* a (aub)^* a (aub)^* a (aub)^*$

Q13.  $L_1 = a^*b^*, L_2 = a^*b^*$

Find  $L_2 - L_1 \rightarrow \text{None}$

Q14.  $L_1 = a^* + b^* \text{ and } L_2 = a^*b^*$

Which of the following is true?  $\rightarrow L_1^* = L_2^*$

Q15.  $L = a^* + b^* \text{ and } L_2 = a^*b^*$

Which of the following is true?

$L_1$  is subset of  $L_2$ .

Q16.  $L_1 = a^+ b^+$  and  $L_2 = a^* b^*$ .

Which of the following is FALSE?  $L_1^* = L_2^*$ .

Q17.  $L_1 = a^+ + b^+$  and  $L_2 = a^* + b^*$ .

Which of the following is TRUE?  $L_1 \cup L_2 = L_2$ .

Q18.  $L_1 = a^*$  and  $L_2 = a^*$ .

Which of the following is TRUE?  $L_1^* = L_2^*$ .

Q19.  $A = a^*$  and  $B = b^*$ .

$$AB = ? \quad \{a^m b^n \mid m, n \geq 0\}$$

Q20.  $A = aa^*$  and  $B = bb^*$   $(A \cup B)^* = ? \quad (a+b)^*$

Q21. Given the language  $L = \{ab, aa, baa\}$ , which of the following strings are not in  $L^*$ ?

bbaaaabaaaaab.

Q22. The length of the shortest string NOT in the language (over  $\Sigma = \{a, b\}$ ) of the following regular expression is  $a^*(ba)^*a^*$  is 1.

Q23.  $L = a(a+b)^*$  is equivalent to  $(ab)^*$ ,  $(a^+b^*)^*$ ,  $a^*(ab^*)^*$ .

Q24.  $L = (a+b)^*b$  is equivalent to  $b^*(ab^*)^*b$

Q25.  $(b+ba)(b+a)^*(ab+b)$   $b(a+b)^*b$

Q26.  $\{w \in \{a, b\}^*: \#a(w) =_3 0\}$

$(b^*ab^*ab^*a)^*b^*$ .

Q27.  $(a+b)^*(a \vee \epsilon)b^* \rightarrow (a+b)^*$ .

Q28.  $L = \{w \in \{a,b\}^* \mid w \text{ has } bba \text{ as a substring}\}$

Which of the following describes  $L$ ?

$$(aub)^* bba (aub)^*$$

Q29.  $L = \{w \in \{a,b\}^*\}$

How many of below are equivalent to given  $L$ ?

$$(a+b)^*$$

$$(a^*b^*)^*$$

$$(a+b+\epsilon)^*$$

$$(b^*a^*)^*$$

$$\epsilon + (a+b)^*$$

$\therefore 5$  are equivalent.

### GATE PYQs

1. Which two of following regular expressions are valid?

$$\text{Ans. } (00)^*(\epsilon+0) = 0^* \rightarrow 0^* \quad \therefore (\text{i}) \text{ and } (\text{iii})$$

2. If the regular set  $A$  is represented by  $A = (01+1)^*$  and regular set 'B' is represented by  $B = ((01)^*1)^*$ , which of the following is true?

Ans.

$$01 = x$$

$$1 = y$$

$$A = (xy)^*$$

$$B = (x^*y^*)^* \quad \therefore A = B$$

3. The string 1101 does not belong to the set represented by

$$\text{Ans. c. } (10)^* (01)^* (00+11)^*$$

$$\text{d. } (00 + (11)^* 0)^*$$

4. Let  $S$  and  $T$  be languages over  $\Sigma = \{a,b\}$  represented by the regular expressions  $(a+b)^*$  and  $(a+b)^*$ , respectively. Which of the following is true?

Ans.

$$S \setminus T \quad (a+b)^* = (a+b)^*$$

$$S \supseteq T \quad (a^* + b)^* = (a+b)^* \quad \therefore S = T$$

$$T \subseteq S$$

HW

5. Consider the set  $\Sigma^*$  of all strings over the alphabet  $\Sigma = \{0, 1\}$ .  $\Sigma^*$  with the Concatenation operator for strings.
6. The regular expression  $0^*(10^*)^*$  denotes the same set as  
 Ans  $(1^*0)^*1^* = (0+1)^*$ .
7. Which of the following languages over alphabet  $\{0, 1\}$  is described by regular expression  $(0+1)^*0(0+1)^*0(0+1)^*$ ?  
 Ans The set of all strings containing atleast two 0's.
8. Consider the language  $L_1 = \emptyset$  and  $L_2 = \{a\}$ . Which one of the following represents  $L_1 L_2^* \cup L_1^* L_2$ ?  
 Ans  $\{\epsilon\}$ .
9. The lengths of the shortest strings NOT in the language (over  $\Sigma = \{a, b\}$ ) of the following regular expressions  $a^*b^*(ba)^*a^*$ .  
 Ans 3
10. Which of the following regular expression represents the language : the set of all binary strings having two consecutive 0's and two consecutive 1's?  
 Ans  $(0+1)^*(00(0+1)^*11 + 11(0+1)^*00)(0+1)^*$ .
11. Let  $r = 1(1+0)^*$ ,  $s = 11^*0$  and  $t = 1^*0$  be three regular expressions. Which one of the following is true?  
 Ans  $L(s) \subseteq L(r)$  and  $L(s) \subseteq L(t)$        $r = 1(1+0)^* \quad \left\{ \begin{array}{l} s \in r \\ s = 11^*0 \end{array} \right.$   
 $L(s) \subseteq L(t)$  and  $L(s) \subseteq L(r)$        $s \in t \quad \left\{ \begin{array}{l} t = 1^*0 = 0+1^*0 \\ s = 11^*0 \end{array} \right.$
12. Which of the following regular expression identified are true?  
 Ans  $(r^*s^*)^* = (r+s)^*$

13. Which one of the following regular expressions represents the set of all binary strings with an odd number of 1's?

Ans.  $= \emptyset^* (0^* 1 0^* 1 0^*)^* 0^* 1 0^*$ .

14. Which one of the following regular expressions over  $\{0, 1\}$  denotes the set of all strings not containing 100 as a substring?

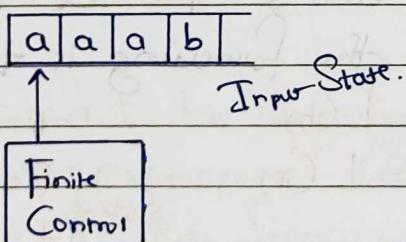
Ans.  $0^* (10 + 1)^*$ .

<u>Acceptor</u>	<u>Transducer</u>
Input: $\xrightarrow{\quad} \boxed{FA}$ Accepted/ Rejected.	Input: $\xrightarrow{\quad} \boxed{FA}$ $\xrightarrow{\quad} \text{Output}$
<u>FA without O/p.</u> <div style="border-left: 1px solid black; padding-left: 10px;">           NFA DFA.         </div>	<u>FA with O/p.</u> <div style="border-left: 1px solid black; padding-left: 10px;">           Moore M/c Mealy M/c         </div>

Applications: Token recogniser,  
Spell checker.

Applications: Sequence detector,  
Generators, Pref/Dec, Addition.. etc.

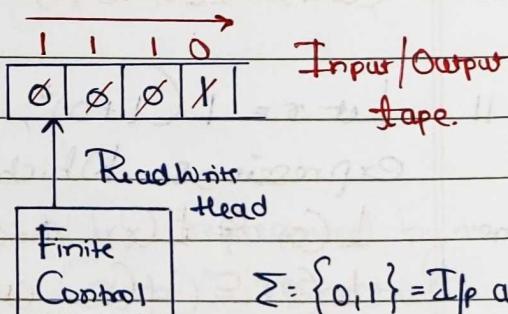
### FA Configuration.



$$FA = (Q, \Sigma, \delta, q_0, F)$$

$$\delta \text{ DFA } Q \times \Sigma \rightarrow Q$$

$$\delta \text{ NFA } Q \times \Sigma_E \rightarrow 2^Q$$



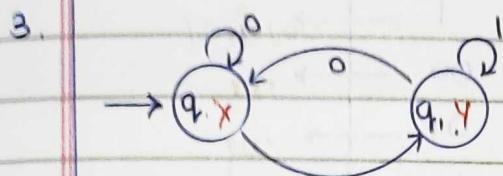
$$FA = (Q, \Sigma, \delta, q_0, \Delta, \lambda)$$

$$\underbrace{\delta Q \times \Sigma \rightarrow Q}_{\text{DFA}}$$

↓ O/p function  
O/p alphabet

Moore M/c

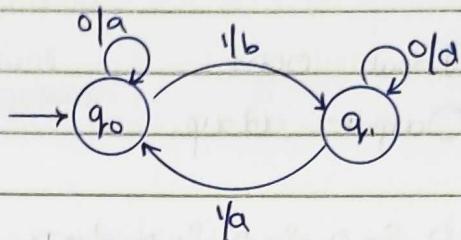
1.  $\lambda: Q \rightarrow \Delta$
2. Output is associated with State.



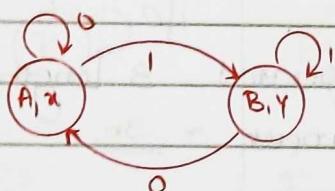
$$\Delta = \{x, y\} \quad \lambda(q_0) = x \\ \lambda(q_1) = y.$$

Mealy M/c.

1.  $\lambda: Q \times \Sigma \rightarrow \Delta$
2. Output is associated with transition.



$$\lambda(q_0, 0) = a \quad \lambda(q_1, 0) = d \\ \lambda(q_0, 1) = b \quad \lambda(q_1, 1) = a.$$

Moore Machine.

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

$$\Delta: A \xrightarrow{0} A$$

$$A \rightarrow B \rightarrow B$$

$$x \quad y \quad y$$

By default.

O/p associated with

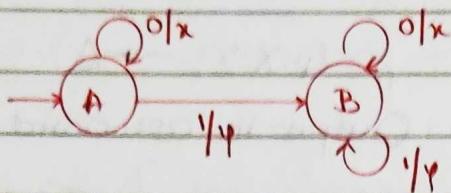
Initial State will be produced.

Input.	Output
$\epsilon$	$x$
0	$xy$
1	$zy$
00	$xxz$
01	$xxy$
10	$zyx$
11	$zyy$ .

- I. If  $n$  length input is given then Output length for given input will be  $n+1$

- II. If  $n$  length input is given and every state is associated with  $z$  length output then Output for the given i/p:  $z^*(n+1)$

## Mealy Machine:



Input: 0001

Output:  $xxyy$ .

$$A \xrightarrow{x} A \xrightarrow{0} A \xrightarrow{0} A \xrightarrow{1} B$$

I/P.	O/P
$\epsilon$	nothing
0	$x$
1	$y$
00	$xx$
01	$xy$
10	$yx$
11	$yy$

I. If  $n$  length input given then Output length for the given input =  $n$ .

II. If every transition is associated 3 length output then Output length for  $n$  length input =  $3n$ .

Moore M/C

Mealy M/C

Common points  
between both

$\delta: Q \times \Sigma \rightarrow Q$

Both are DFAs

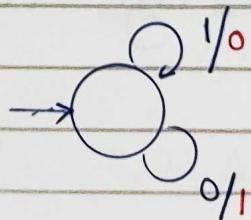
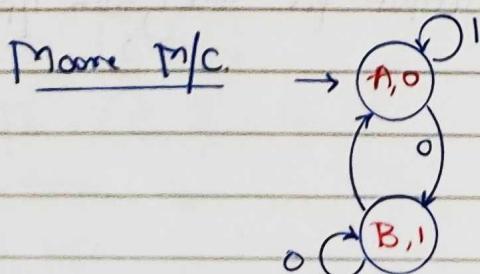
Final state can not there.

1.

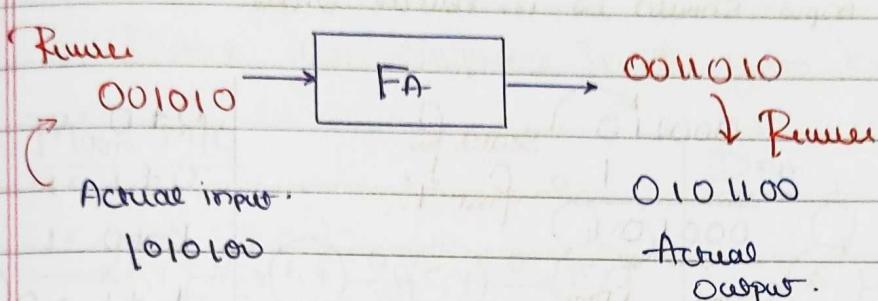
1's Complement.

$$011 \rightarrow F_A \rightarrow 100$$

Mealy M/C

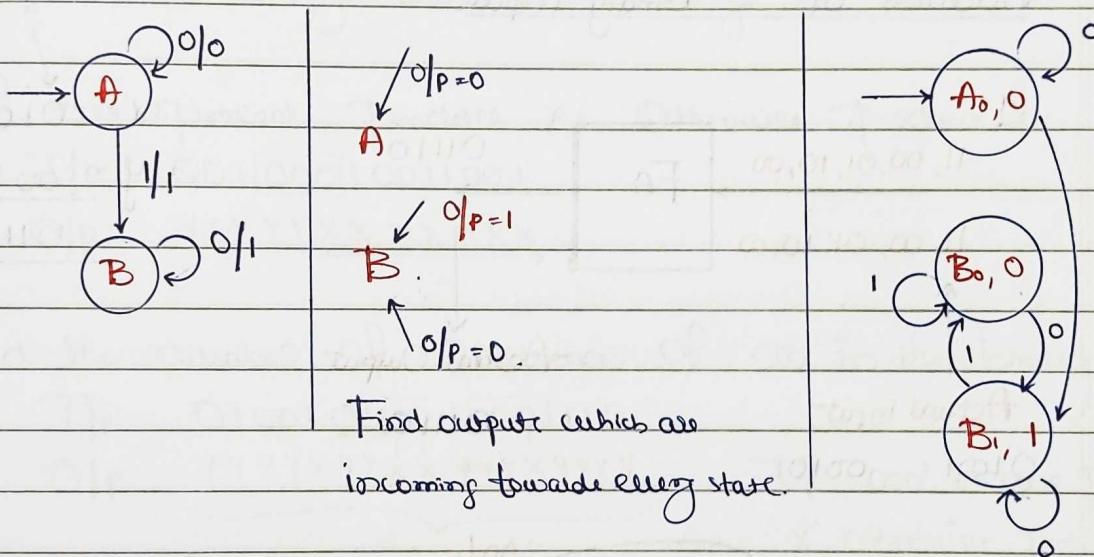


### ② 2's Complement.

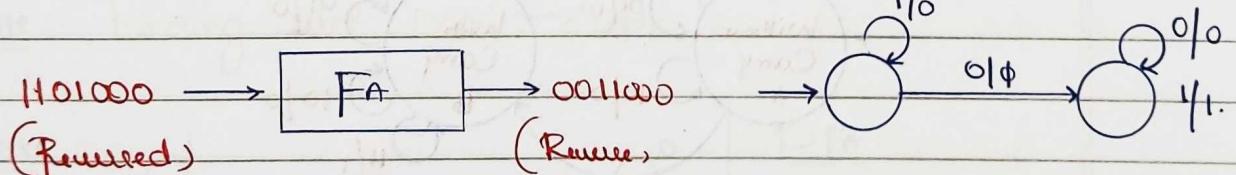


Mealy M/c

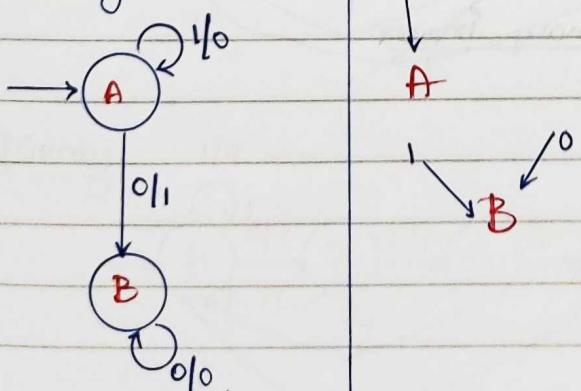
Mealy M/c  $\longrightarrow$  Moore M/c.



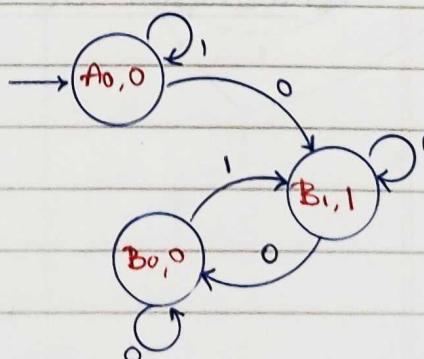
### ③ Increment of Binary.



Mealy:



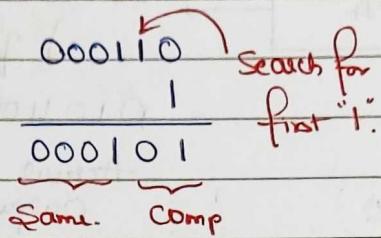
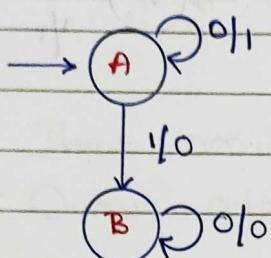
Moore M/c



4. Decrement of Binary Number

$$f(x) = x - 1$$

Note: Given input should be in reverse Order.



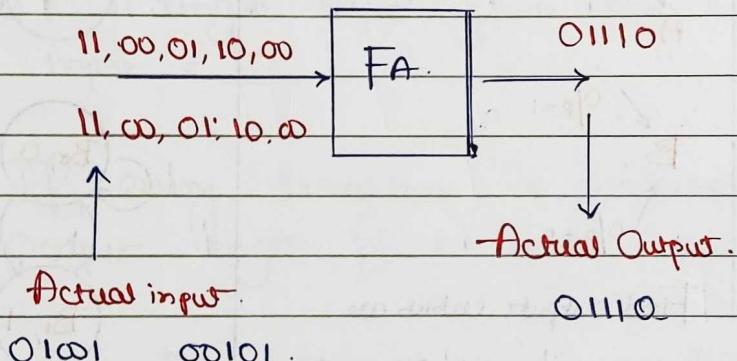
$$0 + 0 = 0$$

$$0 + 1 = 1$$

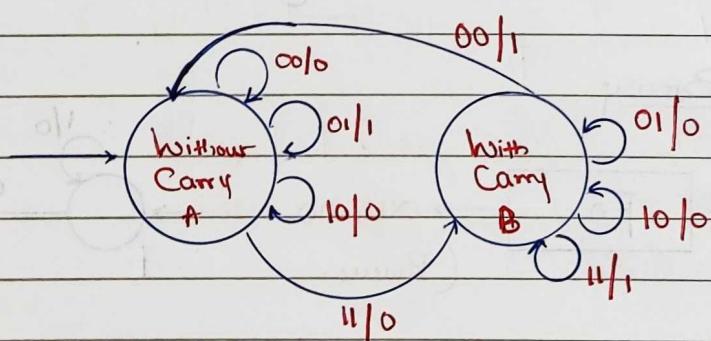
$$1 + 0 = 1$$

$$1 + 1 = 0 \text{ (Carry)}$$

5. Addition of 2 Binary inputs



$x = 01001$   
 $y = 00101$   
 $01110$   $\swarrow$  5<sup>th</sup>.  $\searrow$  1<sup>st</sup>.  
 with carry.



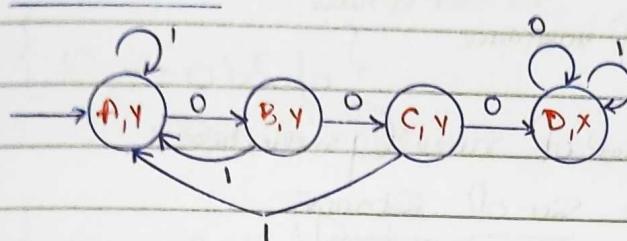
H.W.

6. Subtraction of two Binary inputs

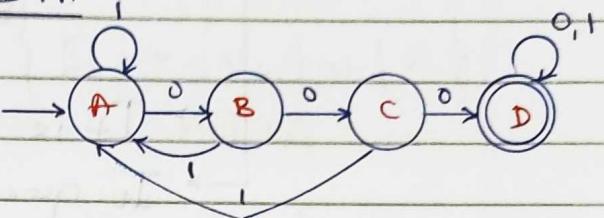
7. Sequence Detector.

Detec 'ooo' Sequence in the given binary input.

Moore M/C



DFA



Note: Containing 'ooo' as substring  $\rightarrow$  DFA.

8. If 'ooo' present, Produce x. Otherwise produce y.

I/P: 001 000 10011001.

O/P: YYYYYYXXXXXXX

Find the number of occurrence of 'ooo' in the binary input.

I/P: 01001 0000 10001110.

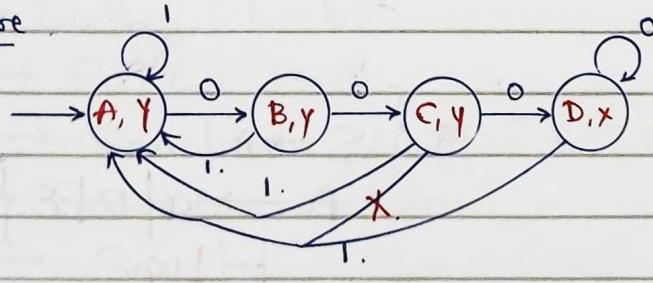
O/P: YYYYYYXXXXYYYYXYYYY,

3 times.

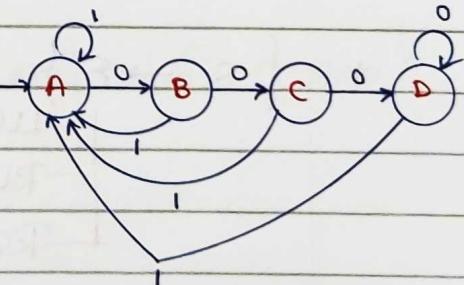
If 'ooo' occurs produce 'x' otherwise produce 'y'.

Note: Ending with 'ooo'  $\rightarrow$  DFA

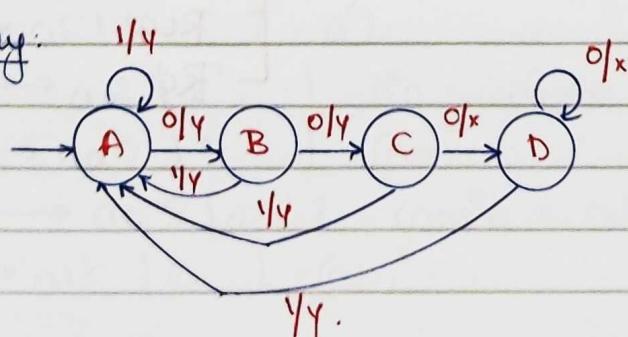
Moore



DFA:



Mealy:



HW

- Q. Produce the sum of present bit and previous bit of input.

$$\text{Grammar } (G) = (V, T, S, P)$$

↑                      ↑                      ↓  
 Set of terminals    Set of symbols    Set of rules  
 ↓                      ↓  
 Set of variables    Set of strings

- It is a set of rules (productions)
- It generates Set of Strings  
(Language)

### Regular Grammar (RG)

- It is left linear grammar (LLG) or Right Linear Grammar (RLG)

LLG

$$V \rightarrow V T^* | T$$

Example:

$$S \rightarrow Sab | \epsilon | ab$$

RLG

$$V \rightarrow T^* V | T$$

Example:

$$S \rightarrow E | a | ab | aabS$$

### Identify RG

$$1. \left\{ S \rightarrow \epsilon. \right\}$$

└── LLG  
└── RLG  
└── RG

$$2. \left\{ S \rightarrow a | bc | \epsilon. \right\}$$

└── LLG  
└── RG  
└── RLG

$$3. \left\{ \begin{array}{l} S \rightarrow A \\ A \rightarrow a | S | \epsilon. \end{array} \right\}$$

└── LLG  
└── RLG  
└── RG

4.  $\{ S \rightarrow Sa, S \rightarrow bs, S \rightarrow a \}$

*neither LRG or RQ.*

5.  $\{ S \rightarrow Sa | b \}$

*LRG ✓  
RQ ✗  
RG ✓*

6.  $\{ S \rightarrow abS | a \}$

*RIG and RG.*

7.  $\{ S \rightarrow aSb | ab | \epsilon \}$

*not RG*

8.  $\{ S \rightarrow AaB | \epsilon \}$

*not RG*

9.  $\{ S \rightarrow ABA | \epsilon \}$

*not RG*

Find Regular Language from Following RGs.

1.  $S \rightarrow \epsilon. \quad L = \{\epsilon\}$

2.  $S \rightarrow a1bb | \epsilon. \quad L = \{\epsilon, a, bb\}$

3.  $S \rightarrow \epsilon | a | aa | aaa. \quad L = \{\epsilon + a + aa + aaa\} = \{a^n | n \leq 3\}$

4.  $S \rightarrow Aa. \quad L = \{\} = \emptyset$

5.  $S \rightarrow Aa | b. \quad L = \{b\}$

6.  $S \rightarrow Aa \quad A \rightarrow \epsilon. \quad L = \{a\}$

7.  $S \rightarrow Aa \quad A \rightarrow ab | \epsilon. \quad L = (\epsilon + ab)a = a + aa + ba$

8.  $S \rightarrow Aa | Bb \quad A \rightarrow C \quad B \rightarrow D. \quad L = \{ca, db\}$

9.  $S \rightarrow AA | BB \quad A \rightarrow \epsilon | a \quad B \rightarrow \epsilon | b. \quad L = \{aa, a, bb, b\}$

10.  $S \rightarrow Sa | b. \quad L = ba^*$

11.  $S \rightarrow Sa | \epsilon. \quad L = a^*$

12.  $S \rightarrow Saa | \epsilon. \quad L = (aa)^* = \{a^{2n} | n \geq 0\} = \{a^n | n = even\}$

13.  $S \rightarrow Sab | \epsilon. \quad L = (ab)^*$

14.  $S \rightarrow Sab | c. \quad L = c(ab)^*$

15.  $S \rightarrow aS | b. \quad L = a^*b$

16.  $S \rightarrow aS | a. \quad L = a^*a = aa^* = a^+$

17.  $S \rightarrow aas | \epsilon. \quad L = (aa)^*$

18.  $S \rightarrow aas | a. \quad L = (aa)^*a = a(aa)^*$

19.  $S \rightarrow abs | \epsilon. \quad L = (ab)^*$

20.  $S \rightarrow Sa \quad L = \emptyset$
21.  $S \rightarrow Sa|b|c. \quad L = (b+c)a^* = ba^* + ca^*$
22.  $S \rightarrow abS|c|de|fgh. \quad L = (ab)^*(c+de+fgh)$
23.  $S \rightarrow Sa|b|c \quad L = (b+c)a^*$
24.  $S \rightarrow Sa|\varepsilon|b \quad L = (\varepsilon+b)a^*$
25.  $S \rightarrow Sa|Sb|c. \quad L = c(a+b)^*$
26.  $S \rightarrow Sa|Sb|\varepsilon \quad L = \varepsilon(a+b)^* = (a+b)^*$
27.  $S \rightarrow aS|bS| \quad L = (a+b)^*\varepsilon = (a+b)^*$
28.  $S \rightarrow Sa|Sb|a \quad L = a(a+b)^*$
29.  $S \rightarrow aS|bS|b \quad L = (a+b)^*b.$
30.  $S \rightarrow Fa \quad F \rightarrow Aa|Ab|\varepsilon. \quad L = (a+b)^*a.$
31.  $S \rightarrow aF \quad F \rightarrow ab|ba|\varepsilon. \quad L = a(a+b)^*$
32.  $S \rightarrow Sa|Sb|a \quad L = a(a+b)^*$
33.  $S \rightarrow aS|bS|a. \quad L = (a+b)^*a.$
34.  $S \rightarrow Sa|Sb|A \quad F \rightarrow aa|ba \quad L = S(a+b)^* = (aa+ba)(a+b)^*$
35.  $S \rightarrow Fa|Fb \quad F \rightarrow Ba \quad B \rightarrow Ba|Bb|\varepsilon.$   

$$\begin{aligned} S &= L = F(a+b) \\ &= (a+b)^*a(a+b) \end{aligned}$$
36.  $S \rightarrow aS|bS|cS|\varepsilon. \quad L = (a+b+c)^*$
37.  $S \rightarrow Fa|Ab|\varepsilon \quad F \rightarrow a|b|\varepsilon$   

$$\begin{aligned} L &= (a+b+\varepsilon)a + (a+b+\varepsilon)b + \varepsilon \\ &= \{\varepsilon, a, b, aa, ab, ba, bb\} \end{aligned}$$
38.  $S \rightarrow Sa|Sb|F \quad F \rightarrow Bab \quad B \rightarrow Ba|Bb|\varepsilon.$   

$$L = S.(a+b)^* = F.(a+b)^* = (a+b)^*ab(a+b)^*$$
39.  $S \rightarrow aS|bS|A \quad F \rightarrow abB \quad B \rightarrow aB|bB|\varepsilon.$   

$$L = (a+b)^*A = (a+b)^*ab(a+b)^*$$
40.  $S \rightarrow Fa|ba \quad F \rightarrow ab|bB \quad B \rightarrow aB|bB|\varepsilon.$   

$$S = (a+b)^n = (a+b)^2(a+b)^*$$
41.  $S \rightarrow Fa|Ab \quad F \rightarrow Ba|Bb \quad B \rightarrow Ba|Bb|\varepsilon$   

$$S = F.(a+b) = (a+b)^*(a+b)^2 = (a+b)^2(a+b)^*$$
42.  $S \rightarrow Sa|Sb|\varepsilon|A \quad F \rightarrow Fa|Ab|\varepsilon.$   

$$S = S.(a+b)^* = (\varepsilon+A)(a+b)^* = (a+b)^*(a+b)^* = (a+b)^*$$

$$43. S \rightarrow Sa | Ab | c \quad A \rightarrow Sb | d.$$

$$L = S(a+b)^* = (db+c)(a+b)^*.$$

$$44. S \rightarrow Sa | Ab \quad A \rightarrow Sb | Ac | e.$$

$$\text{Step 1: } A \rightarrow Ac | Sb | e$$

$$A = (Sb+c)c^* = Sbc^* + ec^*.$$

$$\text{Step 2: } S \rightarrow Saaa | Ab$$

$$S \rightarrow Saa | Sbc^* b | ec^* b \quad L = ec^* b (aa+bc^* b)^*.$$

$$45. S \rightarrow Sa | Saa | \epsilon \quad L = a^* = (a+aa)^*$$

$$46. S \rightarrow Sa | Sb | a | b \quad L = (a+b)^*$$

$$47. S \rightarrow aS | bS | a | b | \epsilon \quad L = (a+b)^* \cdot S = (a+b)^* \cdot (a+b+\epsilon)$$

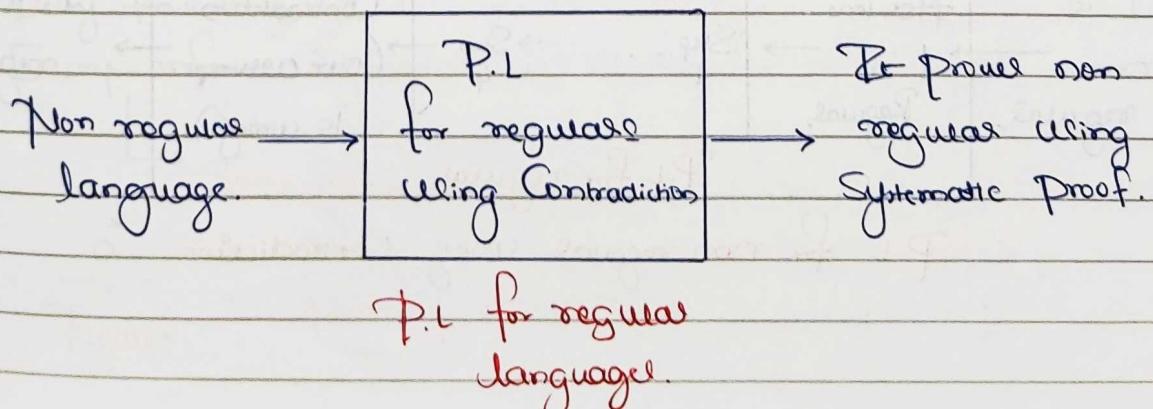
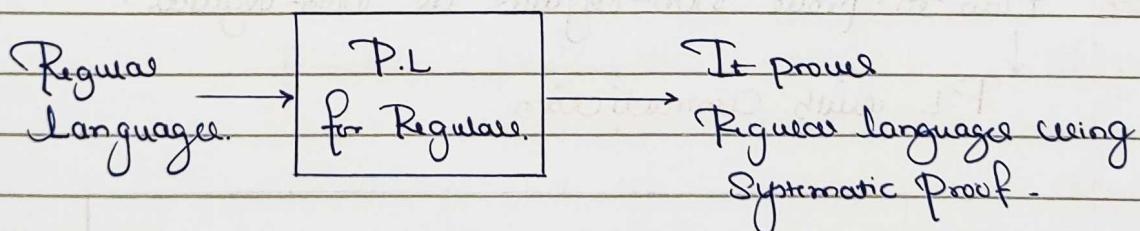
$$48. S \rightarrow aS | bs | abs | \epsilon. \quad L = (a+b+ab)^* = (a+b)^*$$

$$49. S \rightarrow Sa | Ab | a \quad A \rightarrow Ab \quad B \rightarrow Ba | Bb | \epsilon. \quad L = S = a^*$$

### Pumping Lemma for Regular Languages:

- \* It is a Lemma (proof).
- \* It satisfies regular language.
- \* It can also be used to prove non regulars using contradiction.

Note: This pumping lemma can't be used to check unknown language is regular or not.



## Pumping Lemma for Regular Language:

Step 1: Choose any constant  $P$ .

Step 2: Select any string  $w \in L$   
Such that  $|w| \geq P$ .

Step 3: Divide the string  $w$  into 3 parts

$$w = xyz \quad |y| \geq 1, |xy| \leq P$$

$\underbrace{y}$  + ε

Step 4:  $xz^0 yz^2 \in L$  if  $L$  is regular.

Q1. P.L satisfies Regular language.

Q2. P.L proves Non-Regular language using Contradiction.

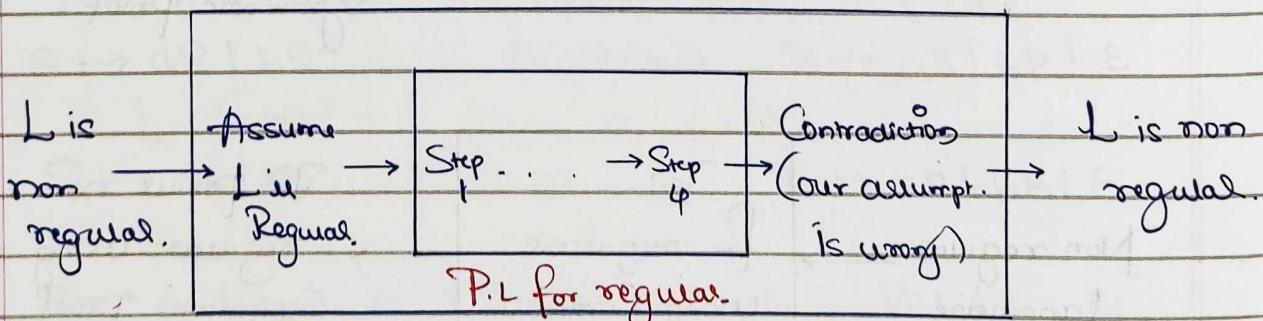
Q3. P.L uses Pigeon-hole principle.

Q4. For  $L = (atb)^*aaa(atb)^*$ , which of the following can't be pumping constant?

- |                     |      |                             |
|---------------------|------|-----------------------------|
| Can be<br>constant. | A. 1 | Min length = 3              |
|                     | B. 2 | # min states in min DFA = 4 |
- 
- |      |      |       |
|------|------|-------|
| C. 5 | D. 9 | P ≥ 4 |
|      |      |       |

→ How to prove non-regular as non-regular?

↓  
P.L with Contradiction.



P.L for non-regular using Contradiction.

Pumping Lemma for Non-regular using P.L for regular with Contradiction.

Step 1: Assume  $L$  is regular.

Step 2: Choose  $p$ .

Step 3: Select  $w \in L$ ,  $|w| \geq p$ .

Step 4:  $w = xyz$ ,  $|y| \neq \Sigma$ ,  $|ay| \leq p$

Step 5: If  $xyz^p \in L$  if  $L$  is regular.  
But if fail, sorry,  $xyz^p \notin L$

Step 6: Contradiction. So  $L$  is non-regular.

Assumptions

Choose  $p$

String  $w$

Divide

Regular

Contradiction.

### Language (Set)

Finite Languages

Regular Language

Infinite Languages

Lower 1 Symbol

Lower  $\geq 1$  Symbol

Form A.P.

Regular

Not Form A.P.

Not Regular

No. Inf.

Dependence

Inf. Dependency or  
Not Form A.P.

Regular

Non-Regular

\* Every finite language is Regular [TRUE]

\* Every infinite language is Regular [FALSE]

Some infinite languages are regular and some are non-regular.

$a^*$

Regular

$a'$

Not regular

## Identify Regular and not Regular:

1.  $\{a^m b^n\} = a^* b^* \rightarrow \text{Regular.}$
2.  $\{a^m b^n \mid m > n\} \rightarrow \text{Not Regular. language}$
3.  $\{a^m b^n \mid m < n\} \rightarrow \text{Not Regular language.}$
4.  $\{a^m b^n \mid m = even, n = even\} = (aa)^* (bb)^*$
5.  $\{a^m b^n \mid m = n = even\} = \{a^{2n} b^{2n}\} \Rightarrow \text{Not Regular.}$
6.  $\{a^m b^n \mid n \geq 0\} = \{\epsilon, ab, aabb, aaabbb \dots\} \rightarrow \text{Not Regular.}$
7.  $\{a^m b^{m+1}\} \rightarrow \text{Not Regular.}$
8.  $\{a^{n+2}, b^{n+100}\}$
9.  $\{a^m b^n \mid m = n \text{ or } m + n\} = a^* b^* \rightarrow \text{Regular}$
10.  $\{a^m b^n \mid m < n < 100\} \rightarrow \text{Finite language} \rightarrow \text{Regular}$
11.  $\{a^m b^n \mid m > n > 100\} \rightarrow \text{Not Regular.}$
12.  $\{a^m b^n \mid \gcd(m, n) = 1\} = \{a^2 b^3, a^2 b^5, \dots\} \rightarrow \text{Not Regular.}$
13.  $\{a^m b^n \mid \text{lcm}(m, n) = 1\} = \{a^1 b^1\} = \text{finite set} \rightarrow \text{Regular.}$
14.  $\{a^m b^n \mid m+n = 1\} = \{b, a\}$
15.  $\{a^m b^n \mid m \neq n = 1\} = \{a^1 b^1\}$
16.  $\{a^m b^n \mid m = n = 1\} = 15 = 13.$
17.  $\{a^m b^n \mid m=2, n=\text{odd}\} = aa b(bb)^* = \text{Regular}$
18.  $\{a^m b^n \mid \text{if } (m = even) \text{ then } (n = \text{odd})\}$   
 $= \{a^m b^n \mid m = \text{odd} \text{ or } n = \text{odd}\} = a(aa)^* b^* + a^* b(bb)^* \rightarrow \text{Regular}$
19.  $\{a^m b^n \mid m \leq 100, n > 100\} = (\epsilon + a + a^2 \dots) b^{101} b^* \rightarrow \text{Regular}$
20.  $\{a^m b^{2n}\} \rightarrow \text{Not Regular.}$
21.  $\{abc^n\}$
22.  $\{a^m b^{2n} c^{3n}\}$
23.  $\{a^m b^{m+1} c^{m+2}\}$
24.  $\{a^m b^n c^*\}$
25.  $\{a^m b^m c^2\}$
26.  $\{a^m b^{m+1} c^{2n}\}$
27.  $\{a^n \mid n \geq 0\} = a^* \rightarrow \text{Regular.}$
28.  $\{a^{2n}\} = (aa)^*$
29.  $\{a^{3n+100}\}$
30.  $\{a^{\text{Prime}}\}$
31.  $\{a^{n!}\}$
32.  $\{a^{2^n}\}$
33.  $\{a^{n^2}\}$
34.  $\{a^{100^n}\}$
35.  $\{a^{n^{45}}\}$
36.  $\{a^{100n}\}$
37.  $\{a^{2024^{n+1}}\}$
38.  $\{a^{100^n}\} \cup \{b^{100k}\}$

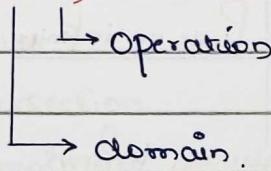
39.  $\{a^{2n}\}^*$  } Regular
40.  $\{a^{\text{Prime}}\}^*$
41.  $\{a^n\} = \{a, a^2, \dots\} \rightarrow \text{Not Regular}$
42.  $\{a^{m^n}\} = \{\epsilon, a, a^2, \dots\} \rightarrow \text{Regular}$
43.  $\{a^{2^n}\}^* = a^*$
44.  $\{a^{n^2}\}^* = a^*$  } Equal.
45.  $\{a^n\}^* = a^*$  } Regular.
46.  $\{a^{m^n}\}^* = a^*$ .

47.  $\{a^*, b^{\text{Prime}}\} \rightarrow \text{Not Regular}$
48.  $\{a^{2n}, b^{3m}\} \rightarrow \text{Regular}$
49.  $\{a^{2n}, b^{m^2}\}$
50.  $\{a^n, b^{\text{Prime}}\}$  } Not Regular
51.  $\{a^{n^2}, b^{2m}\}$
52.  $\{a^n, b^{n^2}\}$
53.  $\{a^n, b^{2^n}\}$  } Not Regular.
54.  $\{a^n, b^{n!}\}$

55.  $\{ww \mid w \in a^*\} = a^{2n} \rightarrow \text{Regular}$
57.  $\{w \# w \mid w \in a^*\} = a^* \# a^* \rightarrow \text{Not R}$
58.  $\{ww^R \mid w \in a^*\} = 56. \text{ Regular}$
59.  $\{w \# w^R \mid w \in a^*\} = 57. \text{ Non-R}$
60.  $\{ww \mid w \in \{ab\}^*\} \rightarrow \text{Non Reg}$
61.  $\{w \# w \mid w \in \{a, b\}^*\}$  } Not
62.  $\{ww^k \mid \dots\}$  } Regular
63.  $\{w \# w^k \mid \dots\}$
64.  $\{w_1 w_2 \mid w_1, w_2 \in a^*\} = a^* \rightarrow \text{Regular}$

What is Closure Property?

$(D, o)$  is closed.



if.  
 $\forall x_1, x_2 \in D \exists x_1, o x_2 \in D$   
 even 2 elements  
in D.

$(D, o)$  is not 'closed'.

if.

$\exists x_1, x_2 \in D \exists x_1, o x_2 \notin D$

some 2 elements  
in D.

For Finite Language, which of the following are closed?

I) Union  $\rightarrow$  Closed

II) Intersection

Finite,  $\cup$  Finite  $\rightarrow$  Finite

Finite,  $\cap$  Finite  $\rightarrow$  Finite

Domain: Set of finite

languages.

	Union	Intersection	Complement
For Finite Language.	closed	closed	not closed.
For Infinitive Language.	closed	not closed	not closed.

$F_1 \cup F_2 = \text{Finite}$      $F_1 \cap F_2 = \text{Finite}$      $\overline{F_2} = \text{Infinite.}$

$\text{Inf.} \cup \text{Inf}_2 = \text{Inf.}$      $\text{Inf} \cap \text{Inf}_2 = \begin{cases} \text{either finite or infinite.} \\ \text{or infinite.} \end{cases}$      $\overline{\text{I}} = \begin{cases} \text{either finite or infinite.} \end{cases}$

→  $\text{Inf}_1 \cap \text{Inf}_2 = \text{either finite or infinite}$   
(need not be infinite)

Example 1:  $a_{\text{Inf}}^* \cap a_{\text{Inf}}^* = a_{\text{Inf}}^*$

Example 2:  $a^* \cap b^* = \{\epsilon\}$   
 $\begin{array}{ccc} \text{Inf.} & \text{Inf.} & \text{finite lang} \\ \{\epsilon, \dots\} & \{\epsilon, \dots\} & \end{array}$

\*\*\*

1. Complement of infinite language is either finite or infinite.
2. Complement of finite language is always infinite.
3. Complement for infinite language is not closed.
4. Complement for finite language is not closed.

### Type of Domains:

- Set of finite languages
- Set of infinite languages
- " " regular "
- " " not regular " "
- " " languages.

Operations	Finite languages	Infinite languages.
Union	✓	✓
Intersection	✓	✗
Complement	✗	✗
Difference	✓	✗
Concatenation	✓	✓
Reversal	✓	✓
Kleene Star	✗	✓
Kleene Plus	✗	✓
Subset	✓	✗
Symmetric Difference	✓	✗
Devu(L) = L ∪ L <sup>rw</sup>	✗	✓

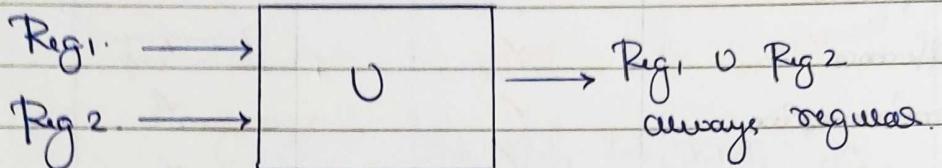
## Closure properties for regular languages:

1. Union	16. $\epsilon$ -free homomorphism.	
2. Intersection	17. Reverse homomorphism.	
3. Complement	18. Quotient	
4. Difference	19. $\frac{1}{2}(L) = \text{half}(L)$	30. Infinite U
5. Concatenation	20. Second half(L)	31. Infinite n
6. Reversal	21. One-third(L) = $\frac{1}{3}(L)$	32. Infinite -
7. Kleene plus	22. Middle $\frac{1}{3}(L)$	33. Infinite .
8. Kleene star.	23. Last $\frac{1}{3}(L)$	34. Infinite $\Sigma$
9. Symmetric diff.	24. Finite U	35. Infinite F
10. Subset. ✗	25. Finite D	
11. Prefix	26. Finite -	
12. Suffix	27. Finite .	
13. Substring	28. Finite $\subseteq$	
14. Substitution	29. Finite f.	
15. Homomorphism.		

# ① Union for regular language.

→ Closed.

$\text{Reg}_1 \cup \text{Reg}_2 \rightarrow \text{Always Regular.}$



Proof:

$\underbrace{\text{Reg. Lang}}_{\text{Regexp1.}} \cup \underbrace{\text{Reg. Lang}}_{\text{Regexp2.}} \rightarrow \text{Reg Lang.}$

Add '+' in between

$\underbrace{\text{Regexp1} + \text{Regexp2.}}$

" new Regexp."

eg:

$$L_1 = \phi, L_2 = \Sigma^* \rightarrow L_1 \cup L_2 = \Sigma^*$$

$$L_1 = a^*, L_2 = (aa)^* \rightarrow L_1 \cup L_2 = a^*$$

$$L_1 = \text{Any}, L_2 = \phi \rightarrow L_1 \cup L_2 = L_1$$

$$L_1 = \Sigma^*, L_2 = \text{Any} \rightarrow L_1 \cup L_2 = \Sigma^*$$

$$L_1 = a^*b^*, L_2 = a^* \rightarrow L_1 \cup L_2 = L_1$$

$$L_1 = a(a+b)^*, L_2 = a^*b^* \rightarrow L_1 \cup L_2 = L_1$$

$\text{Reg} \cup \text{Non Regular} \rightarrow \text{Either Reg or non-regular.}$

Case 1:  $\Sigma^* \cup \text{Any non-regular} \rightarrow \text{Regular.}$

Case 2:  $\phi \cup \text{Any non-regular} \rightarrow \text{Non regular.}$

$\text{Non Regular} \cup \text{Non Regular} \rightarrow \text{Either regular or non-regular.}$

Case 1:  $a^*b^* \cup \overline{a^*b^*} \rightarrow \underbrace{(a+b)^*}_{\text{Regular.}}$

Case 2:  $a^*b^* \cup a^*b^* \rightarrow a^*b^*$

Not regular.

HW

1. Reg. Language  $\cup$  Reg. Language  $\rightarrow$  Regular
2. Regular Language  $\cup$  Non Regular Language  $\rightarrow$  either reg / non regular
3. Non regular Language  $\cup$  Non regular Language  $\rightarrow$  either reg / non regular
4. If  $L_1 \cup L_2$  is Regular then  $L_i$  is Either reg or non regular.  
 $L_1 \cup L_2 \rightarrow$  Regular possible  
 $\text{Non regular } \cup L_2 \rightarrow$  Regular possible.
5. If  $L_1 \cup L_2$  is Not Regular than  $L_i$  is Either regular / not regular.  
 $L_1 \cup L_2 \rightarrow$  Not regular possible  
 $\text{Not Regular } \cup L_2 \rightarrow$  Not regular possible

## ② Intersection for regular language

$\hookrightarrow$  closed

Reg.  $\cap$  Reg.  $\rightarrow$  Always Regular.

- (i.)  $\emptyset \cap L \rightarrow \{\} \cap L \rightarrow \{\} \rightarrow \emptyset$
- (ii.)  $\Sigma^* \cap L \rightarrow$  set of all strings  $\cap L \rightarrow L$
- (iii.)  $a^* \cap b^* \rightarrow \{\epsilon, a, a^2, \dots\} \cap \{\epsilon, b, b^2, \dots\} \rightarrow \{\epsilon\}$
- (iv.)  $a^* \cap b^* \rightarrow \emptyset$
- (v.)  $a^*b^* \cap a^* \rightarrow a^*$ .

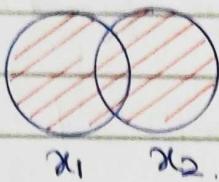
I. If  $L_1$  is Regular and  $L_2$  is Regular then  $L_1 \cap L_2$  is Regular.

II. If  $L_1$  is Regular and  $L_2$  is not regular then  $L_1 \cap L_2$  is Either regular or non regular.

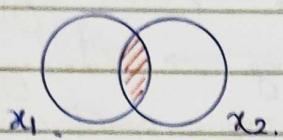
III. If  $L_1$  is not regular and  $L_2$  is not regular then  $L_1 \cap L_2$  is Either regular or not regular.

IV. If  $L_1 \cap L_2$  is regular then  $L_1$  is either regular or not regular.

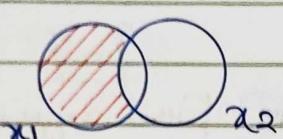
V. If  $L_1 \cap L_2$  is not regular then  $L_2$  is either reg or non regular.



$$x_1 \cup x_2 = \{ x \mid x \in x_1 \text{ or } x \in x_2 \}$$



$$x_1 \cap x_2 = \{ x \mid x \in x_1 \text{ and } x \in x_2 \}$$



$$x_1 - x_2 = \{ x \mid x \in x_1 \text{ and } x \notin x_2 \}$$

*x is only in x<sub>1</sub>.*

If  $x_1 \cap x_2 = \emptyset$  then  $x_1$  and  $x_2$  are Disjoint sets

### ③ Complement for Regular Language

*Closed.*

Regular  $\rightarrow$  Always Regular.

I.  $L$  is Regular iff  $\bar{L}$  is Regular

II.  $L$  is not regular if  $\bar{L}$  is not regular.

III. Not regular  $\rightarrow$  Not regular.

Proof:

$L$  is regular.



Design DFA



Modified DFA



$\bar{L}$  is regular.

$$i. L = \emptyset \rightarrow \bar{L} = \Sigma^*$$

$$ii. L = \Sigma^* \rightarrow \bar{L} = \emptyset$$

$$iii. L = a\Sigma^* \rightarrow \bar{L} = \epsilon + b\Sigma^*$$

$$iv. L = b\Sigma^* \rightarrow \bar{L} = \epsilon + a\Sigma^*$$

$$v. L = \Sigma^*a \rightarrow \epsilon + \Sigma^*b$$

$$vi. L = \Sigma^*a\Sigma^* \rightarrow \bar{L} = b^*$$

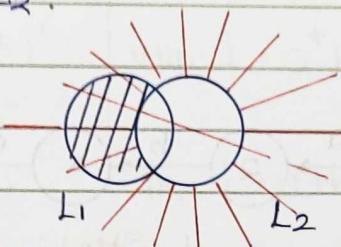
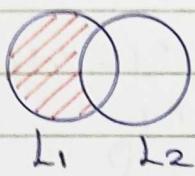
$$vii. L = \Sigma^*b\Sigma^* \rightarrow \bar{L} = a^* = \epsilon + a^*$$

(4) Difference for regular language.

→ closed.

$L_1 - L_2 \rightarrow \text{Always Regular.}$

Dif I  $L_1 - L_2 = L_1 \cap \bar{L}_2.$



$$a^* - b^* \rightarrow a^*$$

Dif II  $L_1 - L_2 = L_1 - (L_1 \cap L_2)$

i.  $L_1 = \emptyset, L_2 = \text{Any} \rightarrow L_1 - L_2 = \emptyset, L_2 - L_1 = L_2.$

ii.  $L_1 = \Sigma^*, L_2 = \text{Any} \rightarrow L_1 - L_2 = \Sigma^* - L_2 = \bar{L}_2$   
 $L_2 - L_1 = L_2 - \Sigma^* = \emptyset$

iii.  $L_1 = a^*, L_2 = b^* \rightarrow L_1 - L_2 = a^* - b^* \rightarrow a^*$   
 $L_2 - L_1 = b^* - a^* \rightarrow b^*$

iv.  $L_1 = a^*, L_2 = (aa)^* \rightarrow L_1 - L_2 = a(aa)^*$   
 $L_2 - L_1 = \emptyset.$

(5) Concatenation for regular languages.

→ closed

$L_1 \cdot L_2 \rightarrow \text{Always Regular.}$

reg. reg.

(i)  $L_1 = \emptyset, L_2 = \text{Any} \rightarrow L_1 \cdot L_2 = \emptyset, L_2 \cdot L_1 = \emptyset.$

(ii)  $L_1 = \Sigma^*, L_2 = \{a\} \rightarrow L_1 \cdot L_2 = \Sigma^* a, L_2 \cdot L_1 = a \Sigma^*.$

(iii)  $L_1 = a^*, L_2 = b^* \rightarrow L_1 \cdot L_2 = a^* b^*, L_2 \cdot L_1 = b^* a^*.$

(iv)  $L_1 = a^*, L_2 = (aa)^* \rightarrow L_1 \cdot L_2 = L_1, L_2 \cdot L_1 = L_1.$

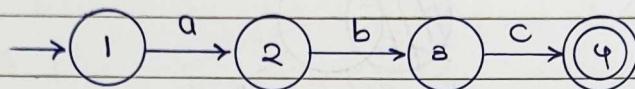
(v)  $L_1 = \Sigma^*, L_2 = a \Sigma^* \rightarrow L_1 \cdot L_2 = \Sigma^* a \Sigma^*, L_2 \cdot L_1 = L_2.$

⑥ Regular Language Reversal. / Reversal for regular language.  
 ↗ closed.

If  $L$  is Regular,  $L^{Rev}$  = Regular.

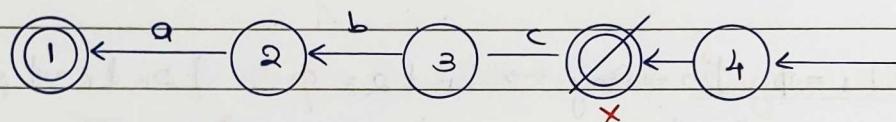
$(Reg)^{Rev}$  = Regular

e.g.:  $L = \{abc\}$ .



$$L^{Rev} = \{cba\}$$

- ↓
- Step 1: make 1. as final
  - Step 2: Transition  $\leftrightarrow$  Final
  - Step 3: Reverse all transitions.



i.  $L = \emptyset \rightarrow L^{Rev} = \emptyset = L$

ii.  $L = \Sigma^* \rightarrow L^{Rev} = \Sigma^* = L$

iii.  $L = a\Sigma^* \rightarrow L^{Rev} = \Sigma^*a$

iv.  $L = \Sigma^*b \rightarrow L^{Rev} = b\Sigma^*$

v.  $L = ab^* \rightarrow L^{Rev} = b^*a$ .

vi.  $L = \Sigma^*a\Sigma^* \rightarrow L^{Rev} = \Sigma^*\bar{a}\Sigma^*$

vii.  $L = a^*b^* \rightarrow L^{Rev} = b^*a^*$ .

viii.  $(L^{Rev})^{Rev} = L$

ix.  $(\bar{L}) = L$

I. Reversal of regular language is Regular

II. - Reversal of non-regular language is non regular.

Note: \*  $L$  is Regular if  $\bar{L}$  is regular

\*  $L$  is Regular if  $L^{Rev}$  is regular.

⑦ Kleene Star. (Kleene closure)

If  $L$  is Reg.  $L^+$  is Regular

Note: \* If  $L^+$  is regular then  
 $L$  is either reg/non regular

\* If  $L^+$  is regular then  $L$  is  
 either reg/non regular.

⑧ Kleene Plus (positive closure)

If  $L$  is Reg.  $L^+$  is Regular.

$\left\{ \begin{array}{l} \{a^{\text{prime}}\}^+ \text{ is Regular} \\ \text{but } \{a^{\text{prime}}\} \text{ is not} \end{array} \right.$   
 regular.

$$\text{i. } L = \phi \rightarrow L^+ = \{\epsilon\}$$

$$L^+ = \phi = L$$

$$\text{ii. } L = \Sigma^+ \rightarrow L^+ = \Sigma^+ = L$$

$$L^+ = \Sigma^+ = L$$

$$\text{iii. } L = \{a, b\} \rightarrow L^+ = (a+b)^*$$

$$L^+ = (a+b)^*$$

$$\text{iv. } L = a^* + b^* = L^+ = (a+b)^*$$

$$L^+ = (a+b)^*$$

$$\text{v. } L = a^* b^* \rightarrow L^+ = (a+b)^* \cdot L^+ = (a+b)^*$$

$$\text{vi. } L = b^* a^* \rightarrow L^+ = (a+b)^*$$

$$L^+ = (a+b)^*$$

$$\text{vii. } L = a^+ + b^+ \rightarrow L^+ = (a+b)^*$$

$$L^+ = (a+b)^*$$

$$\text{viii. } L = a^+ b^+ \rightarrow L^+ = (a+b)^* \quad \{ \neq (a+b)^*$$

$$L^+ = (a+b)^*$$

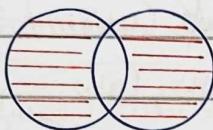
$$\text{ix. } L = a+b^+ \rightarrow L^+ = (a+b)^* \quad L^+ = (a+b^+)^*$$

9. Symmetric Difference for regulars.

→ closed.

$$L_1 \oplus L_2 = L_1 \Delta L_2 = (L_1 - L_2) \cup (L_2 - L_1)$$

$L_1, L_2$ .



$$\text{eg.: } \begin{cases} L_1 = a^* \\ L_2 = b^* \end{cases} \quad L_1 \Delta L_2 = a^* + b^*$$

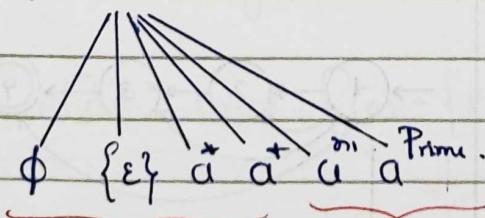
$$\text{ex. } \begin{cases} L_1 = a\Sigma^* \\ L_2 = b\Sigma^* \end{cases} \quad L_1 \Delta L_2 = a\Sigma^* + b\Sigma^*$$

10. Subset for Regulars.

→ Not closed.

Every Subset of a regular language is Either regular or non regular.

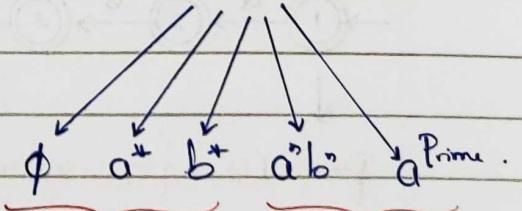
$$L = a^*$$



Regular

Non  
regular.

$$L = (a+b)^*$$



Regular

Non  
regular.

- I. Every subset of finite language is Finite Language (Regular)
- II. Every subset of infinite language is either infinite or not infinite.
- III. Every subset of regular language is either regular or nonregular
- IV. Every subset of nonregular language is either regular or nonregular

For Language.

$$\text{prefix}(L) = \{ u \mid w \in L, w=uv \} = \{ u \mid u \in L \}$$

$$\text{Suffix}(L) = \{ v \mid uv \in L \}$$

$$\text{SubString}(L) = \{ y \mid xyz \in L \}$$

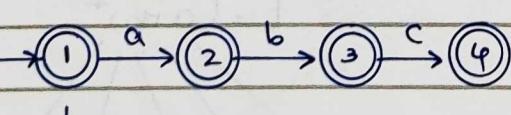
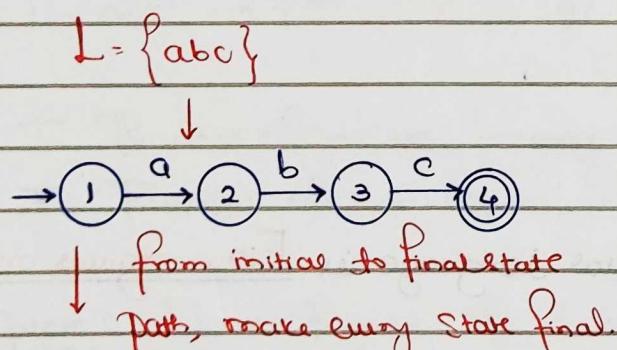
For String:

$$\text{Prefix}(w) = \{ u \mid uu=w \} \quad \text{prefix}(ab) = \{ \epsilon, a, ab \}$$

$$\text{Suffix}(w) = \{ v \mid uv=w \} \quad \text{suffix}(ab) = \{ \epsilon, b, \underline{\overline{ab}} \}$$

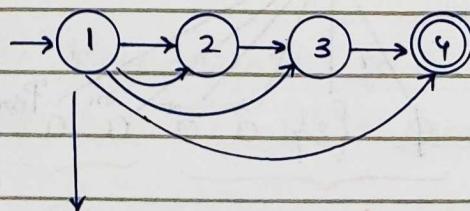
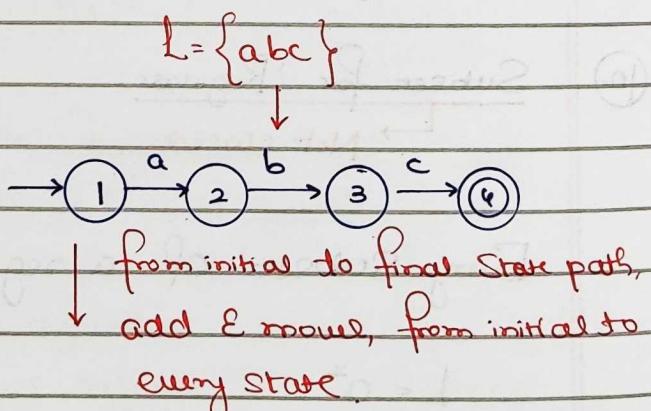
$$\text{SubString}(w) = \{ y \mid xyz=w \} \quad \text{substring}(ab) = \{ \epsilon, a, b, ab \}$$

(11) Prefix (Reg Language) is Regular



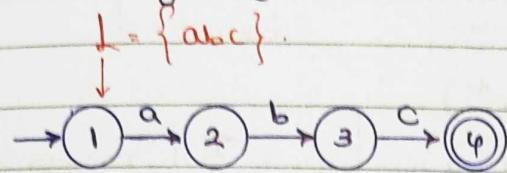
$$\text{prefix}(L) = \{ \epsilon, a, ab, abc \}$$

(12) Suffix (Regular) is Regular.

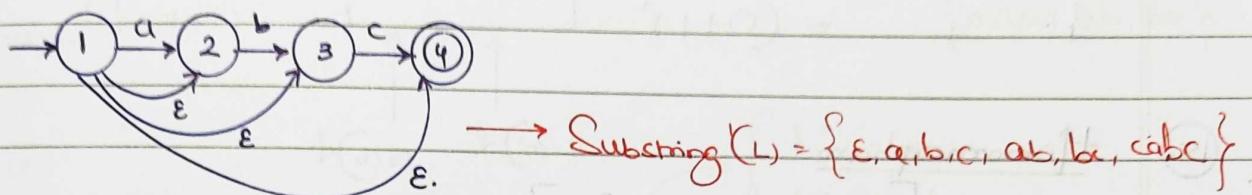


$$\text{suffix}(L) = \{ \epsilon, c, abc, abc \}$$

(13) Substring (Reg) is Regular.



Combine both prefix algorithm and suffix algorithm.

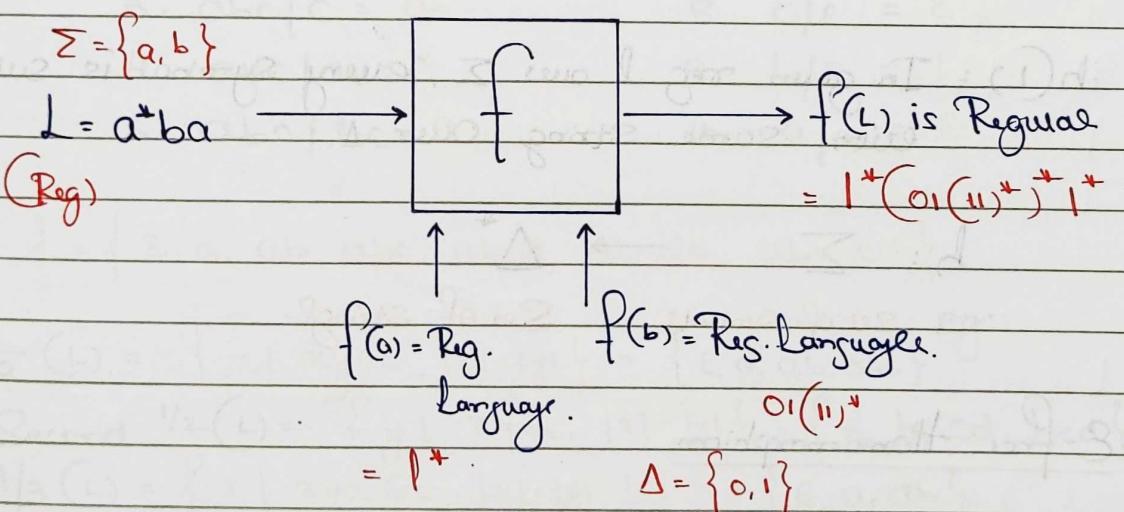


(14) Substitution for Regular Language.

→ closed.

Substitution of a regular language is Regular Language.

(Regular Substitution).



1.  $L = (ab)^*$  and  $f(a) = \{\epsilon\}$ ,  $f(b) = 0^*$ .

$$\begin{aligned}
 f(L) &= (f(a) \cdot f(b))^* \\
 &= (\epsilon \cdot 0^*)^* \\
 &= 0^*.
 \end{aligned}$$

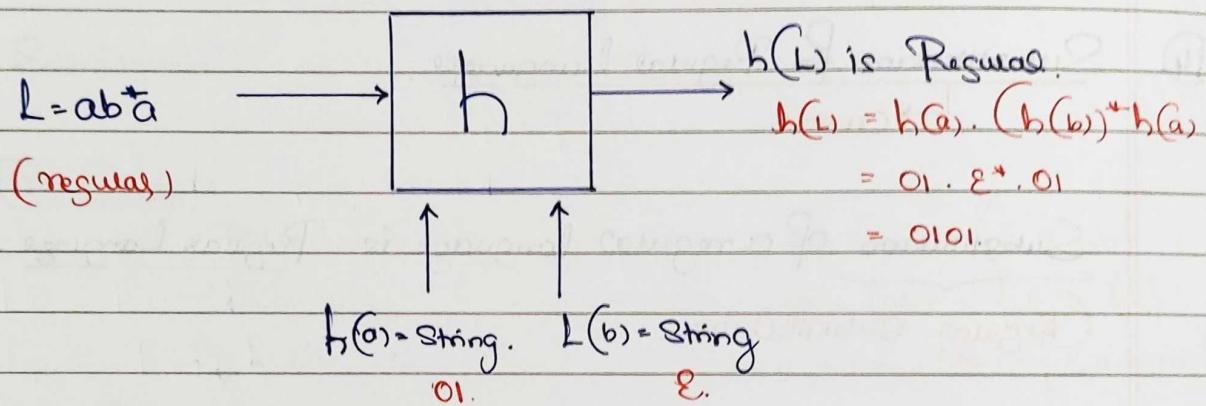
2.  $L = a(a+b)^*$ ,  $f(a) = 0^*$ ,  $f(b) = 1^*$ .

$$\begin{aligned} f(L) &= f(a(a+b)^*) \\ &= f(a) \cdot (f(a) + f(b))^* \\ &= 0^* (0^* + 1^*)^* \\ &= (0+1)^* \end{aligned}$$

$\Sigma$  { may or may not  
 $\Delta$  } be same.

(15) Homomorphism

[String Substitution]



$h(L)$ : In given reg  $L$  over  $\Sigma$ , every symbol is substituted with some string over  $\Delta$ .

$$h: \Sigma \longrightarrow \Delta^*$$

Set of symbols      Set of strings

(16)  $\epsilon$ -Free Homomorphism

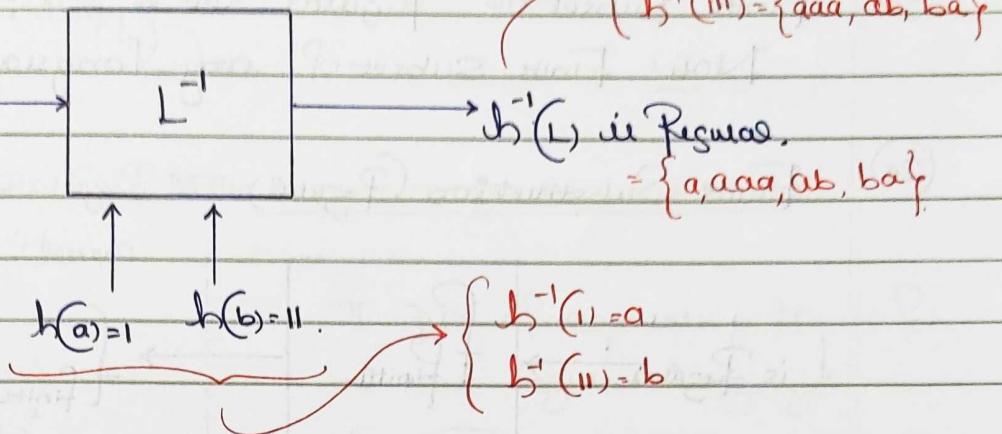
$$h: \Sigma \longrightarrow \Delta^+$$

In gives reg  $L$  over  $\Sigma$ , every symbol is substituted with some non empty string (other than  $\epsilon$ ) over  $\Delta$ .

(17) Inverse Homomorphism for Regular Language

$\hookrightarrow$  closed.

$L = \{1, 111\}$ ,  
given. (Regular)



(18) Quotient (1)

$$L_1 / L_2 = \{ u \mid uv \in L_1, v \in L_2 \}.$$

- eg::
- |                           |                                     |
|---------------------------|-------------------------------------|
| 1. $abc / \epsilon = abc$ | 6. $abc / abc = \epsilon$           |
| 2. $abc / a = \times$     | 7. $\epsilon / abc = \times$        |
| 3. $abc / c = ab$         | 8. $\epsilon / \epsilon = \epsilon$ |
| 4. $abc / ab = \times$    | 9. $L / \epsilon = L$               |
| 5. $abc / bc = a$         | 10. $a / a = \epsilon$ .            |

$$L = \{\epsilon, a, ab, abc, abcd, abcd\epsilon, abbbcd\}.$$

(19)  $\frac{1}{2}(L) = \{x \mid xy \in L, |x|=|y|\} = \{\epsilon, a, ab, abb\}$  eg::

(20) Second  $\frac{1}{2}(L) = \{y \mid xy \in L, |x|=|y|\} = \{\epsilon, b, cd, bcd\}$

(21)  $\frac{1}{3}(L) = \{x \mid xy_2 \in L, |x|=|y_2|\} = \{\epsilon, a, ab\}$

(22) Modulo  $\frac{1}{3}(L) = \{y \mid " " " \} = \{\epsilon, b, bb\}$

(23) Last  $\frac{1}{3}(L) = \{z \mid " " " \} = \{\epsilon, c, cd\}$

(24) Finite union is closed for regular languages.

$L_1 \cup L_2 \cup L_3 \dots \cup L_k \rightarrow$  Regular.

(25) Finite intersection is closed for regulars.

$L_1 \cap L_2 \cap L_3 \dots \cap L_k \rightarrow$  Regular.

(26)

 $L_1 - L_2 - L_3 \dots - L_K \rightarrow \text{Regular}$ 

(27)

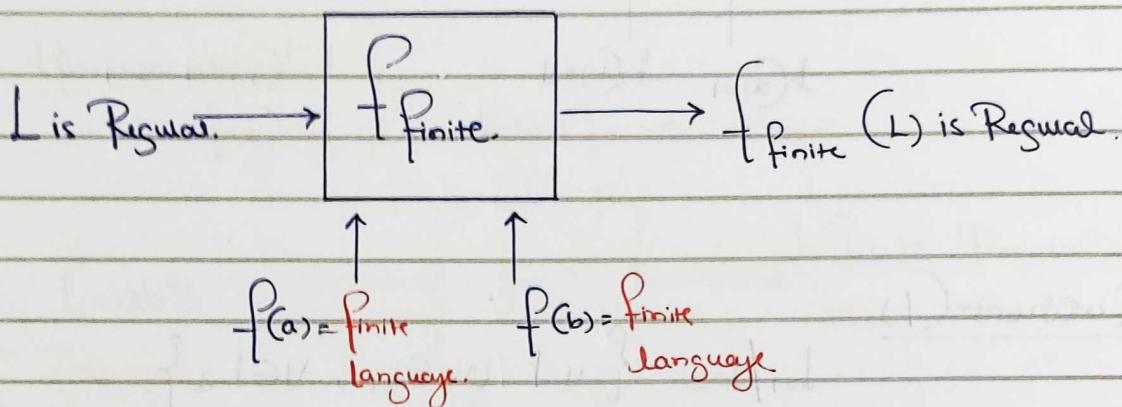
 $L_1 * L_2 * L_3 \dots * L_K \rightarrow \text{Regular}$ 

(28)

Finite Subset of Regular Set is finite set (Regular)

Note: Finite subset of any language is finite

(29)

Finite Substitution (Regular)  $\rightarrow$  Regular.

(30)

 $R_1 \cup R_2 \cup R_3 \dots \rightarrow$  either regular or non regular

(31)

 $R_1 \cap R_2 \cap R_3 \dots \rightarrow$  "

(32)

 $R_1 - R_2 - R_3 \dots \rightarrow$  "

(33)

 $R_1 \cdot R_2 \cdot R_3 \dots \rightarrow$  "

(34)

Infinite Subset of regular language  $\rightarrow$  either regular or non regular

(35)

Infinite Subset of regular language  $\rightarrow$  " Substitution.

## Equivalence Classes [Myhill - Nerode equivalence classes]

I. No. of states in min DFA for Reg = no. of equivalence classes for regulars.

Find no. of equivalence class for  $L = ab(a+ab)^*$

Regular Language

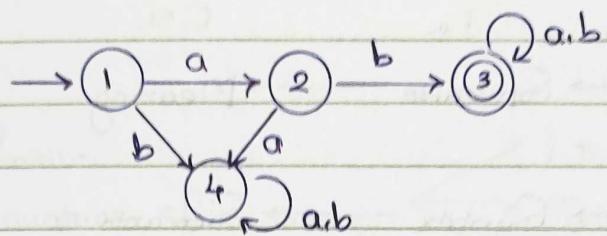
$\rightarrow$  Step 1: Construct min DFA

## PUSH DOWN AUTOMATA

classmate

Date \_\_\_\_\_

Page \_\_\_\_\_



Step 2: Each state represents one equivalence class.

= 4 equivalence classes

$$[1] = \emptyset$$

$$[2] = a$$

$$[3] = ab (a+b)^*$$

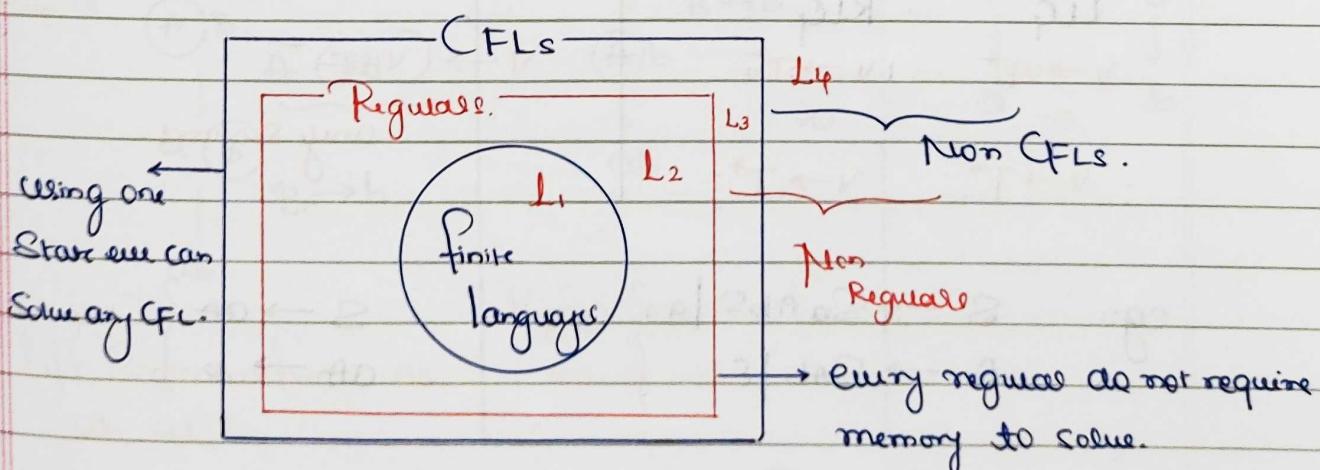
$$[4] = (b+a)(a+b)^*$$

II. No. of equivalence classes for non-regular languages.

$a^*b^*$  → finite no. of equivalent classes.  
Regular

$a^n b^n$  → infinite no. of equivalent classes.  
Non Regular

## Context Free Language



- I.  $L_1$  is finite language (Reg., CFL)
- II.  $L_2$  is regular but not finite (Infinite Resources)
- III.  $L_3$  is CFL but not regular. (CFL and not regular)
- IV.  $L_4$  is not CFL. (not regular and not finite)

Regular Language.

CFL

CSL

Words  $\rightarrow$  Structure  $\rightarrow$  Meaning

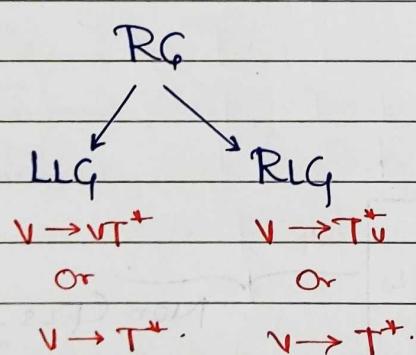
Tokens  $\rightarrow$  Syntax  $\rightarrow$  Semantic

Words  $\rightarrow$  Sentences  $\rightarrow$  English language meaning

### CFL Applications:

1. Every regular application
2. Messages
3. Call history
4. Web page history
5. DFS
6. Balanced Parenthesis

$$Q = (V, T, P, S)$$



CFG.

It represents CFL.

$$V \rightarrow (V \cup T)^*$$

any sequence.

eg::  $S \rightarrow SaAbS \mid a \quad \{ \text{CFG } \checkmark$

 $A \rightarrow Sab \mid E. \quad \{$

$S \rightarrow aA \quad \{ \text{CFG } \times$ 
 $aA \rightarrow b \quad \}$

Note: (i) Every RG is CFG

(ii) CFG need not be RG

(iii) Every Regular Language is CFL

(iv) CFL is may or may not be regular.

CFG.

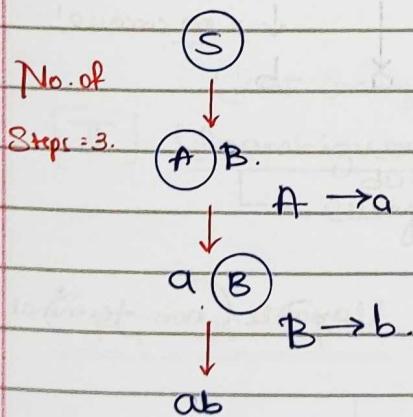
- Definition
  - Derivations of a string.
  - Type of CFG.
  - CFG vs CFL
  - Inherently ambiguous CFL.
- (left most derivation)
- LMD
- RMD (Right most derivation)
- Parse tree (Derivation tree)
- Ambiguous cfg
- Unambiguous cfg.

eg::

CFG.	$S \rightarrow AB   \dots$ $A \rightarrow a   \dots$ $B \rightarrow b   \dots$	String ab.
------	--	------------

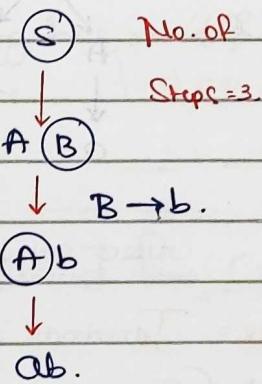
Derivation:

① LMD.



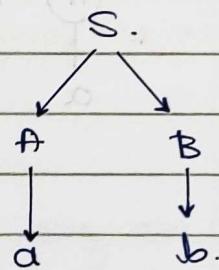
Left Sequential forms: AB,  
ab, AB.

② RMD.



Right sequential forms  
AB, AB, ab.

③ Parse Tree.



LMD: To derive a string, in every Sequential form "left most non terminal" is Substituted.

Q1. left most symbol = 'a',    Q2. left most terminal = 'a'  
Q3. left most non terminal = 'A'.

RMD: To derive a string, in every sequential form "right-most non-terminal" is substituted.

Note: In general, LMD and RMD need not be same.

$$\text{eg: } S \rightarrow Aa \mid AB$$

$$A \rightarrow c$$

$$B \rightarrow d$$

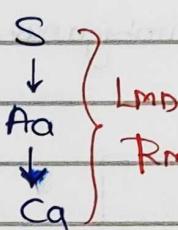
$$w = ca$$

LMD and

RMD are same!

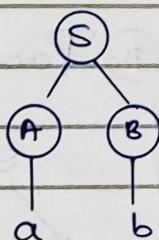
$$w = cd$$

LMD, RMD are different.

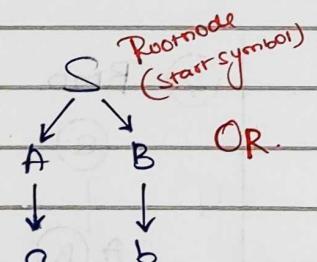


LMD	RMD
S	S
↓	↓
Aa	AB
↓	↓
ca	cd

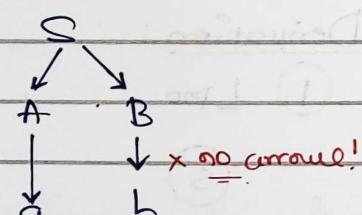
### Parse Tree (Derivations Tree)



OR.



OR.



Root node  
(start symbol)

x no closure!

→ collect all leaf nodes = ab

Leaf node = Terminal or E.

Non leaf node (Intermediate node) = Variable (non-terminal).

For a given string: (If only one derivation exists).

No. of derivation steps = No. of steps in LMD  
 = No. of steps in RMD  
 = No. of non-leaf nodes in parse tree.

No. of Lnodes  
No. of Rnodes  
No. of parent.

No. of Derivations  $\rightarrow$  How many ways string can be derived.

$\neq$

No. of Derivation Step : for a particular derivation, how many steps?

$\rightarrow$  It is possible to answer if we have Only one derivation.

$\rightarrow$  No. of Substitution in Rno.

" " in Lno

" " non leaf node Parent.

### Type of CFGs:

#### [I.] Ambiguous CFG

$\rightarrow$  Some String has more than 1 derivation. (you have to find one string which has  $> 1$  Poststring).

(not ambiguous).

#### [II.] Unambiguous CFG.

$\rightarrow$  every string derived from CFG has only one derivation.

### CFG vs CFL.

$$\begin{aligned} 1. \quad S &\rightarrow AB \\ A &\rightarrow \epsilon \\ B &\rightarrow b \\ L &= \{b\}. \end{aligned}$$

$$2. \quad S \rightarrow SS \mid \epsilon. \quad L = \{\epsilon\}$$

$$3. \quad S \rightarrow Sa \mid \epsilon. \quad L = a^*$$

$$4. \quad S \rightarrow aS \mid \epsilon. \quad L = a^*$$

$$5. \quad S \rightarrow Sa \mid a. \quad L = a^*$$

$$6. \quad S \rightarrow aS \mid a. \quad L = a^*$$

7.  $S \rightarrow aS|bS|\epsilon.$   
 $L = (a+b)^*$

11.  $S \rightarrow aaS|\epsilon$   
 $L = (aa)^*$

15.  $S \rightarrow AB$   
 $A \rightarrow aA|\epsilon$   
 $B \rightarrow Bb|\epsilon$

$B=b^*$   
 $A=a^*$

8.  $S \rightarrow aS|bS|a|b$   
 $L = (a+b)^*$

12.  $S \rightarrow aS|bS|cS|\epsilon$   
 $L = (a+b+c)^*$

It is not regular.  
 grammar but is  
 generates a regular  
 language.

9.  $S \rightarrow Sa|Sb|\epsilon.$   
 $L = (a+b)^*$

13.  $S \rightarrow abS|aab|\epsilon$   
 $L = (ab+aa)^*$

$L = a^*b^*$ .

10.  $S \rightarrow Sa|Sb|a|b$   
 $L = (a+b)^*$ .

14.  $S \rightarrow aS|bs|b.$   
 $L = (a+b)^*b.$

16.  $S \rightarrow aS|Sb|\epsilon.$   
 $\downarrow$   
 $a^*Sb^* = a^*b^*$ .

17.  $S \rightarrow bs|sa|\epsilon \rightarrow b^*a^*$ .

18.  $S \rightarrow aS|Sb|a \rightarrow L = a^*ab^* = a^*b^*$ .

19.  $S \rightarrow aS|Sb|b \rightarrow L = a^*bb^* = a^*b^*$ .

20.  $S \rightarrow aS|Sb|a|b \rightarrow L = a^*(a+b)b^* = a^*b^* + a^*b^*$ .

21.  $S \rightarrow aS|Sb|a|b \rightarrow L = a^*(a+b)b^* = a^*b^* + a^*b^*$ .

22.  $S \rightarrow aS|Sb|ab \rightarrow L = a^*abb^* = a^*b^*$ .

$S \rightarrow aSb|\epsilon \rightarrow L = a^*b^n = a^*b^n$ .  
 $= \{a^n b^n \mid n \geq 0\} = \{a^n b^n \mid \#a\epsilon = \#b\epsilon\}$

23.  $S \rightarrow bsae. L = \{b^n a^n \mid n \geq 0\}$

24.  $S \rightarrow aSb|a. L = \{a^n ab^n \mid n \geq 0\} = \{a^{n+1} b^n\}$

25.  $S \rightarrow aSb|b. L = \{a^n b^{2n} \mid n \geq 0\}$

26.  $S \rightarrow aaSbb|\epsilon. L = \{a^{2n} b^{2n}\}$

27.  $S \rightarrow aaaSbbb|\epsilon. L = \{a^{3n} b^{3n}\}$

28.  $S \rightarrow aSbb|\epsilon. L = \{a^n b^{2n} \mid n \geq 0\}$

29.  $S \rightarrow aaSb|\epsilon. L = \{a^n b^n\} = \{a^m b^n \mid m=2n\}$

30.  $S \rightarrow aaSbbb|\epsilon. L = \{a^{2n} b^{3n} \mid n \geq 0\}$   
 $= \{a^i b^j \mid i=2n, j=3n\}$

31.  $S \rightarrow aSb | ab$

$$L = \{ab, a^2b^2, a^3b^3, \dots\} \quad \left\{ \begin{array}{l} L = \{a^n ab | n \geq 0\} \\ = \{a^{n+1}, b^{n+1} | n \geq 0\} \\ = \{a^n, b^n | n \geq 0\} \\ = \{a^n b^n | n \geq 0\} \end{array} \right.$$

Note:  $L_1 = \{a^n | n \geq 0\} = a^*$

$$L_2 = \{b^n | n \geq 0\} = b^*$$

$$\begin{aligned} L_1 \cdot L_2 &= \{a^n\} \cdot \{b^n\} \\ &= \{\epsilon, a^1, a^2, \dots\} \cdot \{\epsilon, b, b^2, \dots\} \\ &= \{a^m b^n | m, n \geq 0\} \\ &= \{a^n b^m | n, m \geq 0\} \end{aligned}$$

32.  $S \rightarrow aSb | A$

$$A \rightarrow aA | \epsilon.$$

$$L = \{a^n a^i b^n\} = \{a^i b^n | i \leq j\}.$$

32.  $S \rightarrow aSb | A$

$$A \rightarrow bA | \epsilon \rightarrow b^*$$

$$L = \{a^n b^{m+n}\} = \{a^i b^j | i \leq j\}.$$

34.  $S \rightarrow aSb | A \quad \left\{ \begin{array}{l} S \rightarrow aSb | \epsilon \\ A \rightarrow bB | \epsilon \end{array} \right.$

$$L = a^n b^n. \quad \left\{ \begin{array}{l} S \rightarrow aSb | \epsilon \\ A \rightarrow bB | \epsilon \end{array} \right.$$

35.  $S \rightarrow Sa | \epsilon$

$$A \rightarrow a$$

$$L = A^* = a^*$$

36.  $S \rightarrow Sa | \epsilon.$

$$A \rightarrow aa. \quad L = A^* = (aa)^*$$

37.  $S \rightarrow Sa | \epsilon$

$$A \rightarrow a/b. \quad L = A^* = (a+b)^*$$

38.  $S \rightarrow Sa | \epsilon.$

$$A \rightarrow aAb | \epsilon.$$

$$L = \{a^n b^n\}^* = \{a^n b^n | n \geq 0\}$$

39.  $S \rightarrow Sa | \epsilon$

$$A \rightarrow ba | \epsilon.$$

$$L = \{b^n a^n\}^*$$

40.  $S \rightarrow Sa | \epsilon$

$$A \rightarrow ab. \quad L = (ab)^*$$

41.  $S \rightarrow aS_1|bS_2|\epsilon \rightarrow L = \{ww^R \mid w \in \{a,b\}^*\}$ .

= set of all even length palindromes

=  $\{x \mid x \in \{a,b\}^*, x = x^R, |x| = \text{even}\}$ .

42.  $S \rightarrow aS_1|bS_2|a|b \rightarrow L = \text{length of all odd length palindromes}$

43.  $S \rightarrow aS_1|bS_2|a|b|\epsilon \rightarrow L = \text{set of all palindromes}$

=  $\{x \mid x = x^R, x \in \{a,b\}^*\}$

=  $\{ww^R \mid w \in \{a,b\}^*, w \in \{a,b,\epsilon\}^*\}$

44.  $S \rightarrow aS_1|bS_2|*$

$L = \{w \# w^R \mid w \in \{a,b\}^*\}$ .

45.  $S \rightarrow aS_1|A$

$A \rightarrow aAc| \epsilon \rightarrow L = a^k a^k c^k b^k = \{a^{k+k} c^k b^k\}$ .

46.  $S \rightarrow AB$

$A \rightarrow aAb| \epsilon \quad B = C^*d^* \quad L = \{a^k b^k c^* d^*\}$ .

$B \rightarrow cBd| \epsilon. \quad A = a^k b^k$

47.  $S \rightarrow aS_1|A \quad \left. \begin{array}{l} A = c^*b^* \\ A \rightarrow cab| \epsilon \end{array} \right\} L = a^k c^* b^k b^k = \{a^k c^* b^{k+k}\}$

$A \rightarrow cab| \epsilon \quad L = a^k c^* b^k b^k = \{a^k c^* b^{k+k}\}$

48.  $S \rightarrow ABC$

$A \rightarrow aA| \epsilon - a^*$

$B \rightarrow bB| \epsilon - b^*$

$C \rightarrow Cc| \epsilon. - c^*$

$L = a^* b^* c^*$

49.  $S \rightarrow SS| (S) | \epsilon.$

$L = \text{set of all balanced parenthesis}$

50.  $E \rightarrow E+E| E \cdot E | (E) | a$

$L = \text{set of all arithmetic expressions using } +, \cdot$

} functionality  
(meaning)

51.  $S \rightarrow aS_1|bS_2|SS|aS| \epsilon$

$L = \{w \mid w \in \{a,b\}^*, n_a(w) = n_b(w)\}$

52.  $S \rightarrow aS_1|bS_2|SS|aS| \epsilon$

$L = \{w \mid w \in \{a,b\}^*, n_a(w) \geq n_b(w)\}$ .

53.  $S \rightarrow aS_1|bS_2|SS|bS| \epsilon.$

$L = \{w \mid w \in \{a,b\}^*, n_a(w) \leq n_b(w)\}$ .

54.  $S \rightarrow aS_1|bS_2|SS|aS|a$

$n_a(w) > n_b(w)$ .

ss.  $S \rightarrow S_aS_bS \mid S_bS_aS \mid \epsilon.$

$$L = \{ w \mid w \in \{a, b\}^*, n_a(w) = n_b(w) \}$$

→ Every Regular Language is CFL.

→ Some non regular languages are CFLs.

$$\{ a^n b^n \} \leftarrow \text{non-regular} \rightarrow \text{CFL.}$$

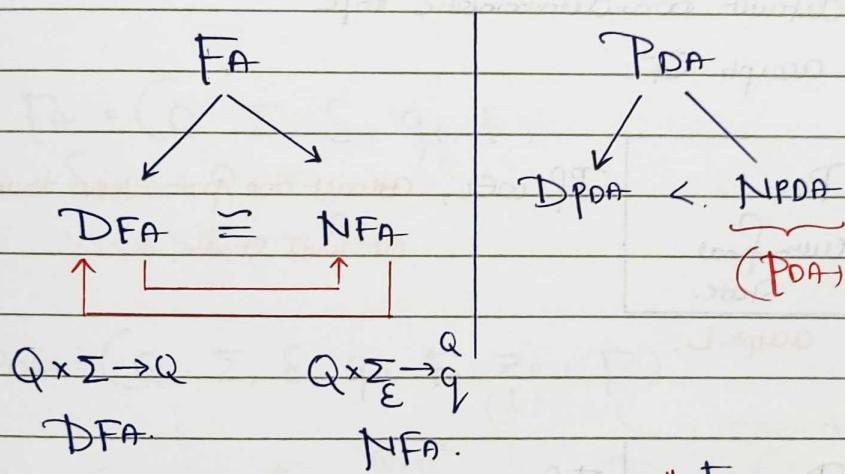
$$\{ w w^* \mid w \in \{a, b\}^* \}$$

$$\{ w \# w^R \mid " " \}$$

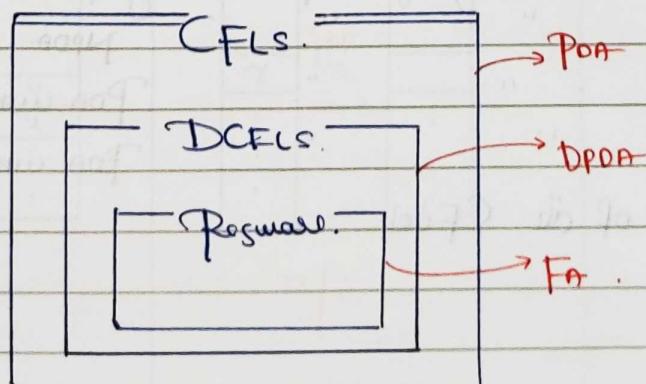
$$\{ a^n b^{2n} \}$$

$$\{ a^n b^n \}$$

## Push Down Automata



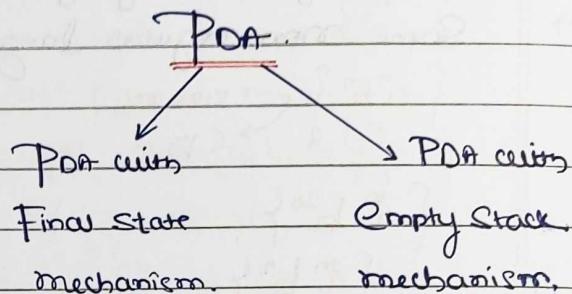
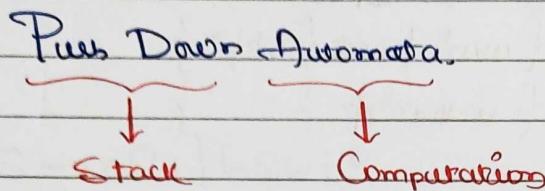
- \* Every DCFL is CFL
- \* CFL need not be DCFL
- \* Every DPDA is PDA
- \* PDA need not be DPDA.



$$FA < DPDA < PDA$$

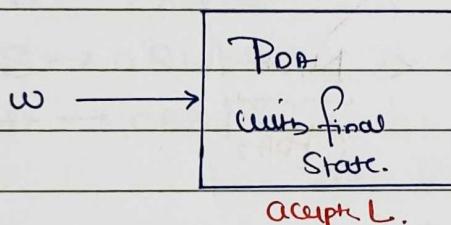
→ Which of the following statements is "True"? //

1. Regular Language is subset of CFL. [False]
  2. Set of all regulars is subset of set of all CFLs. [True]

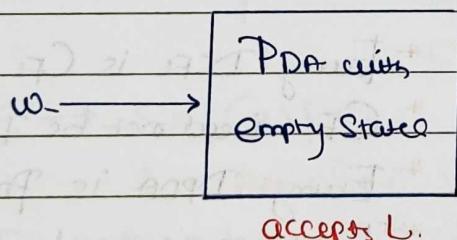


PDA (NPDA)

→ by default non-deterministic m/c  
→ it accepts CFL.



If well, after one pass exist that half  
at final state.



If we, atleast one path exist than  
makes empty stack

$\rightarrow L_1$  = Set of all languages represented by PFA

$\rightarrow L_2 = "NPDA"$

$$\rightarrow J_3 = " " " T$$

$\rightarrow$   $\text{H}_2\text{O}$

$\rightarrow L_S$  = Set of all CFLs.

PDA with final state

PDA with earth stat

John Lewis Company

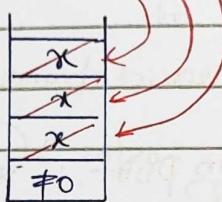
$$\left. \begin{array}{l} J_1 = L_2 \\ J_3 = L_4 \\ \hline J = L \end{array} \right\} \text{all are same.}$$

Push Down Automata.

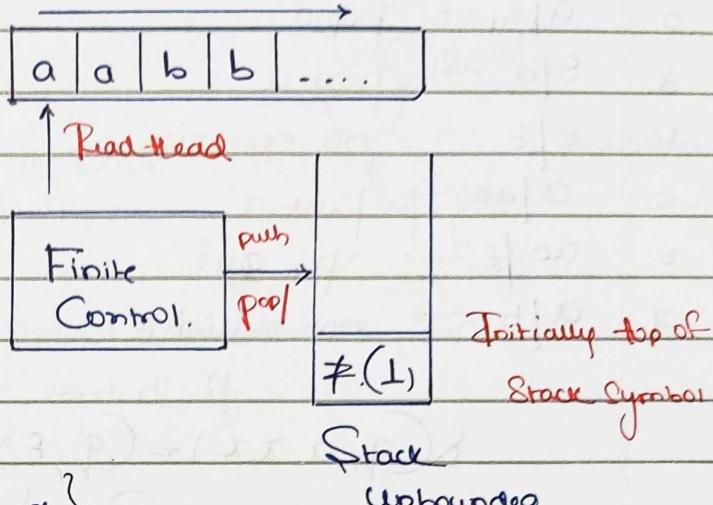
$\overline{ab}^n$ .  $\rightarrow$  input alphabet.

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

aaabbba  
Push



$$\Gamma = \{z_0, x\}$$



Stack Alphabet.

$$FA = (Q, \Sigma, \delta, q_0, F)$$

$$\delta_{NFA}: Q \times \Sigma \rightarrow 2^Q$$

$$\delta_{DFA}: Q \times \Sigma \rightarrow$$

$$POA = FA + \text{Stack}$$

$$POA = (Q, \Sigma, \delta, q_0, F, z_0, \Gamma)$$

$$SPOA = Q \times \Sigma \times \Gamma_E^* \rightarrow 2^{Q \times \Gamma^*}$$

$$\delta_{DPDA} = Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma^*$$

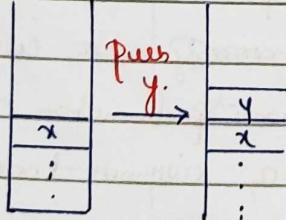
Stack Alphabet,

(Set of Stack Symbols)

initial top of stack symbol

(bottom of stack symbol).

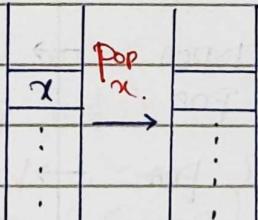
Push.



$$x \rightarrow yx$$

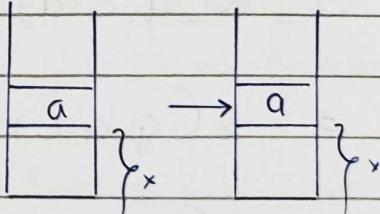
$$x/yx$$

Pop.



$$y/x$$

No Operations



$$a/a$$

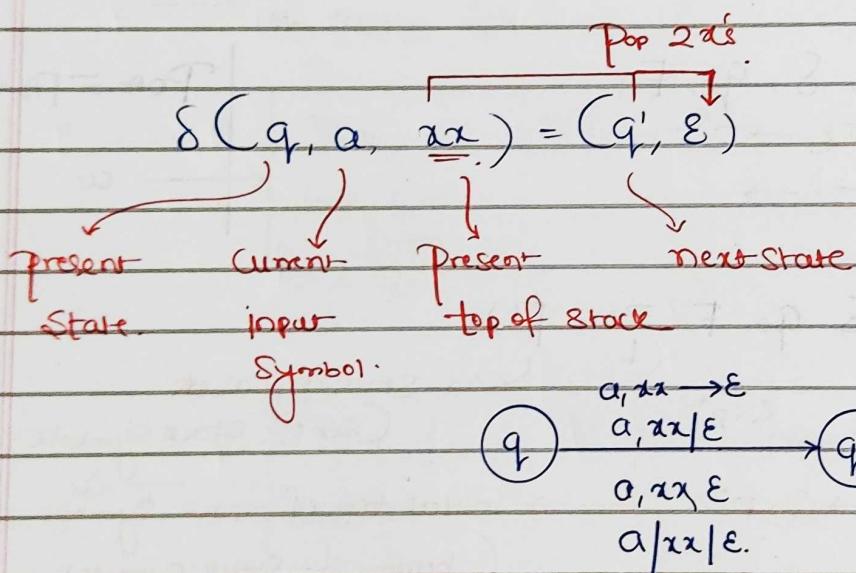
$$\epsilon/\epsilon$$

1.  $a/a \rightarrow$  no operation performed on stack
  2.  $a/E \rightarrow$  pop a
  3.  $E/a \rightarrow$  push a
  4.  $E/E \rightarrow$  no operations " "
  5.  $a/aa \rightarrow$  push a
  6.  $aa/E \rightarrow$  pop aa
  7.  $a/b \rightarrow$  not valid

$\delta(q_1, a, z) = (q_1', \epsilon) \rightarrow$  not valid transition in DFA

$$D_{\text{POA}} = Q \times \varepsilon \times F \rightarrow$$

$$P_{OA} = Q \times \sum_{\mathcal{E}} \times \Gamma^+ -$$



1.  $\delta(q, \epsilon, \epsilon) = (q', a)$  PDA ✓ DPDA ✗ → Without reading input, without looking at tos, push 'a'.

2.  $\delta(q, \epsilon, a) = (q', a)$  { DPDA ✗ PDA ✓ } → Without reading input, when tos=a, no operation performed on stack

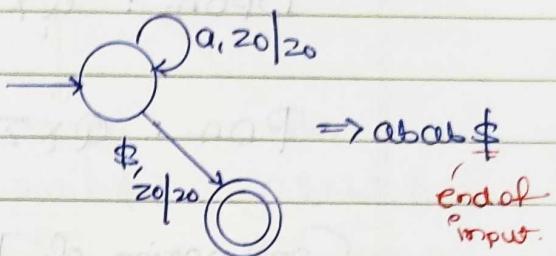
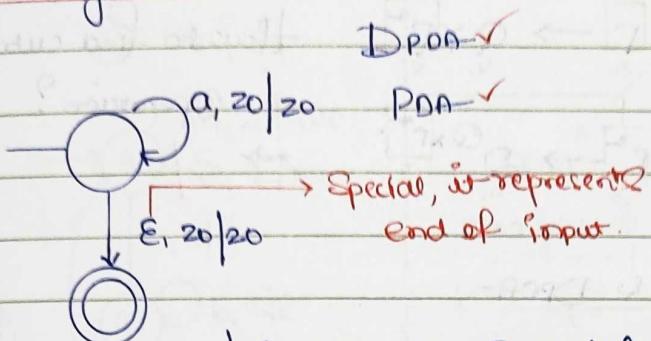
3.  $\delta(q, a, \epsilon) = (q', a)$  { PDA ✓ DPDA ✗ } → When if p=a, without looking at stack push 'a'

4.  $\delta(q, a, b) = (q', ab)$  { PDA ✓ DPDA ✓ } → When if p=a, tos=b, push a

5.  $\delta(q, a, b) = (q', aaab)$  PDA ✓ DPDA ✓ → When if p=a, tos=b, push 3ab

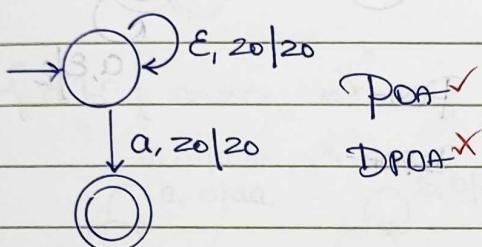
## Identify DPDA or PDA

1.

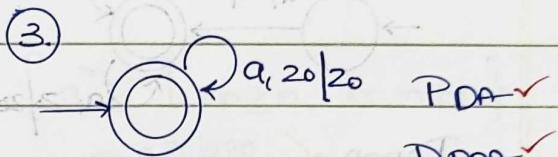


We assume special transitions required to represent end of i/p.

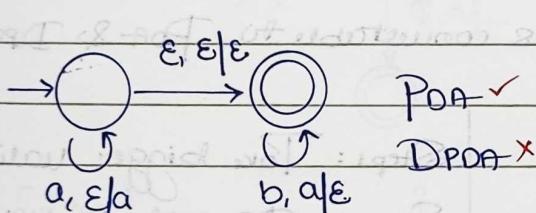
2.



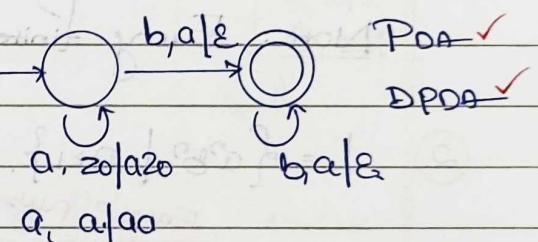
3.



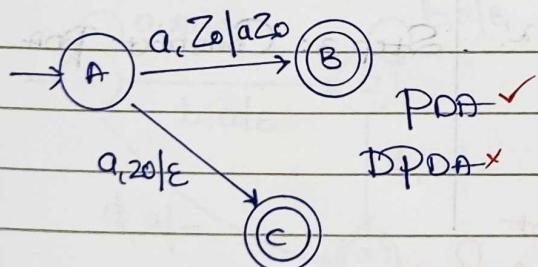
4.



5.



6.



$$Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma^*$$

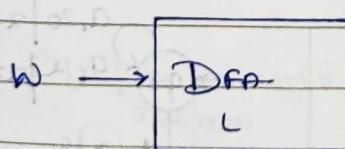
Fail

$$Q \times \Sigma \times \Gamma^* \rightarrow 2$$

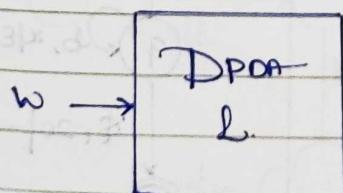
$$\delta(A, a, z_0) = (B, a z_0)$$

Q \times \Gamma^\*

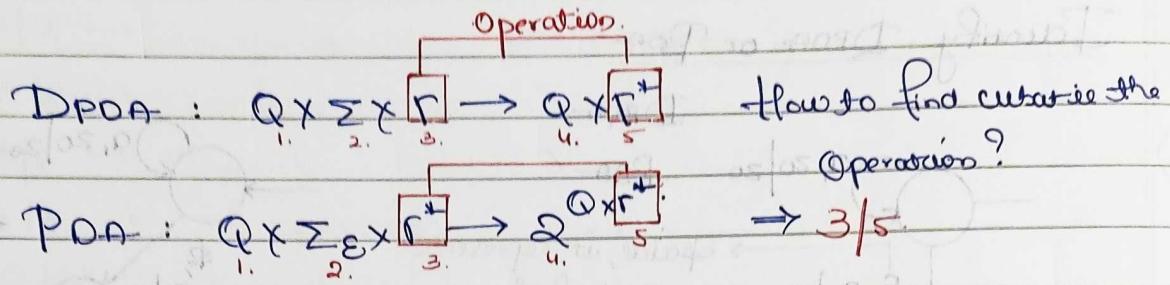
(C, E)



If w ∈ L, exactly 1 path has at final state.  
 If w ∉ L, exactly 1 path has at non-final state.



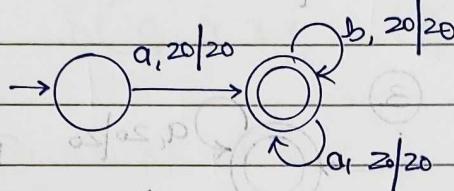
If w ∈ L, exactly 1 path has at final state.  
 If w ∉ L, exactly 1 path has at non-final state  
 or zero paths.



### Construction of POA & DPOA

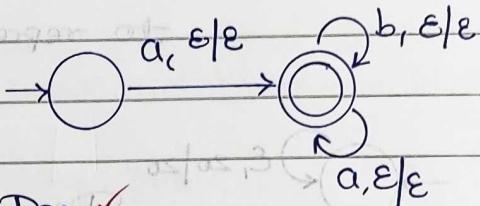
1.

$$L = a(a+b)^*$$



DPOA ✓

POA ✓



POA ✓

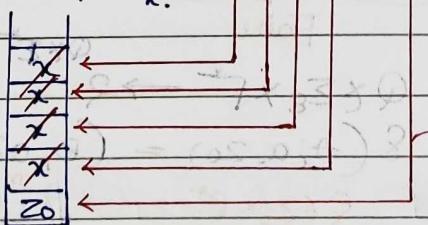
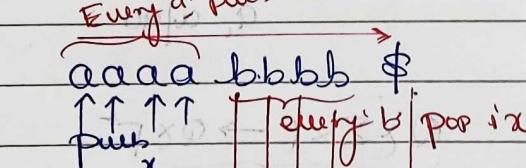
DPOA ✗

Note: Every finite automata is convertible to POA & DPOA.

2.

$$L = \{a^n b^n \mid n \geq 1\}$$

Every 'a' push 'x'.

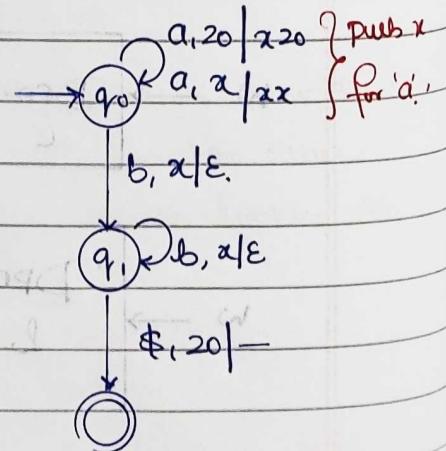
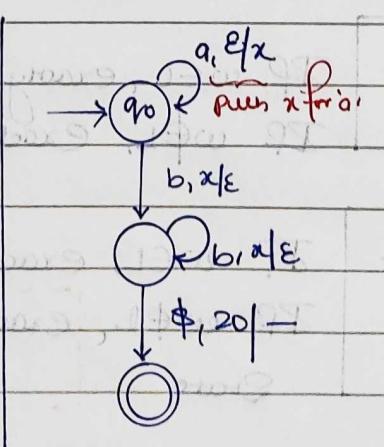
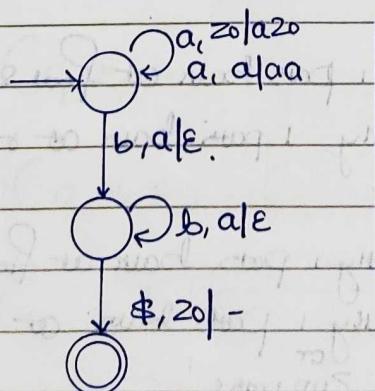


Accept (go to final).

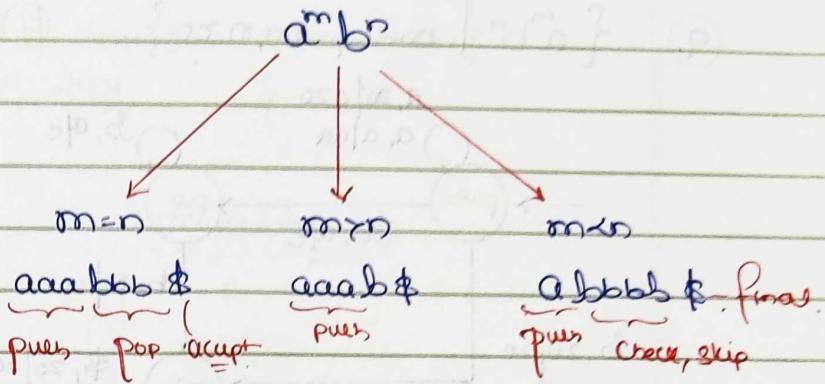
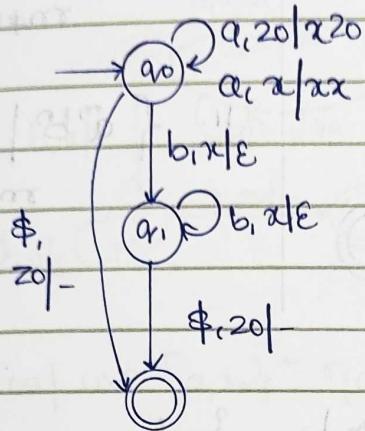
Step 1: Take bigger valid string

Step 2: Make logic using Stack

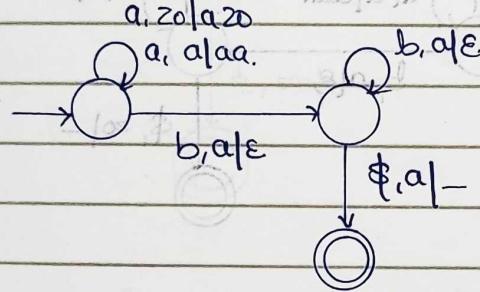
Step 3: Construct POA.



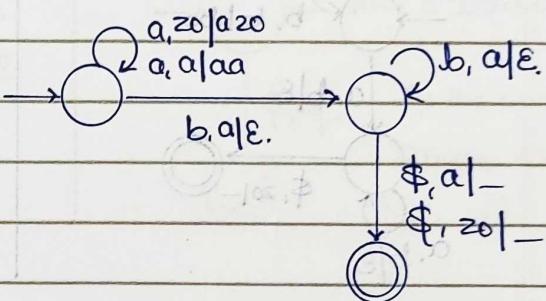
③  $L = \{a^n b^n \mid n \geq 0\}$   
 $= \textcircled{2} \cup \{\epsilon\}$



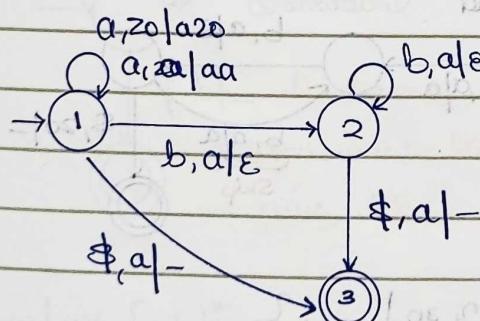
④  $\{a^n b^n \mid m > n, m, n \geq 1\}$



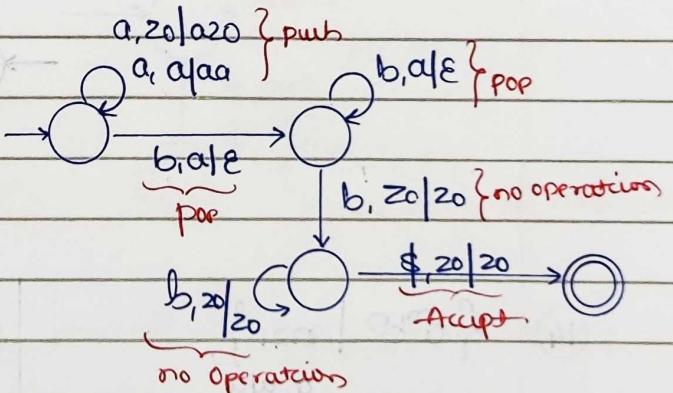
⑤  $\{a^n b^n \mid m \geq n, m, n \geq 1\}$



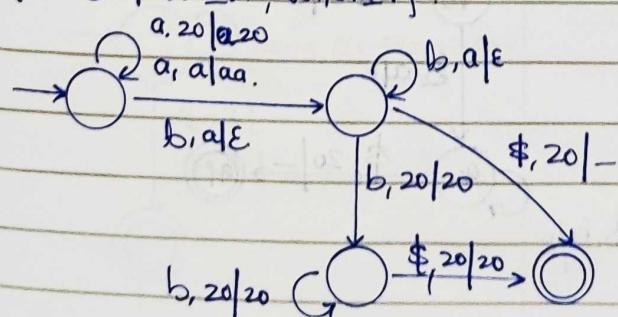
⑥  $\{a^n b^n \mid m > n, m, n \geq 0\}$



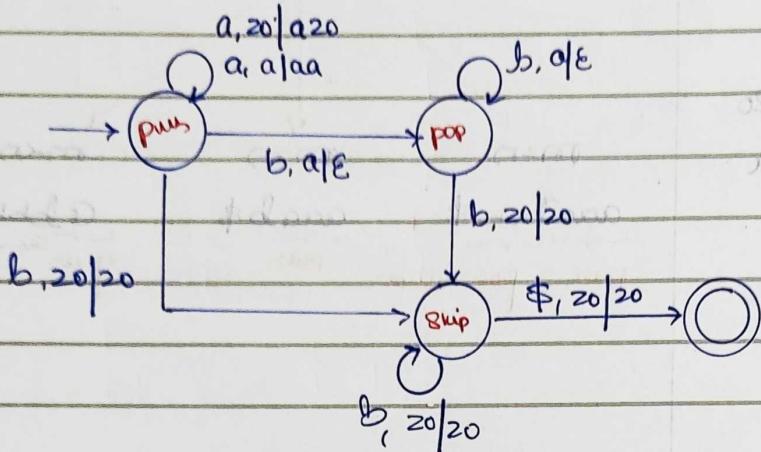
⑦  $\{a^n b^n \mid m < n, m, n \geq 1\}$



⑧  $\{a^n b^n \mid m \leq n, m, n \geq 1\}$



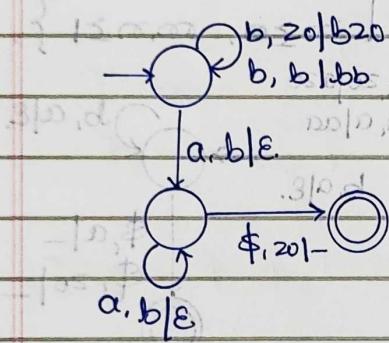
9.  $\{ a^m b^n \mid m, n \geq 0 \} = \text{L} \cup \{ b^+ \}$



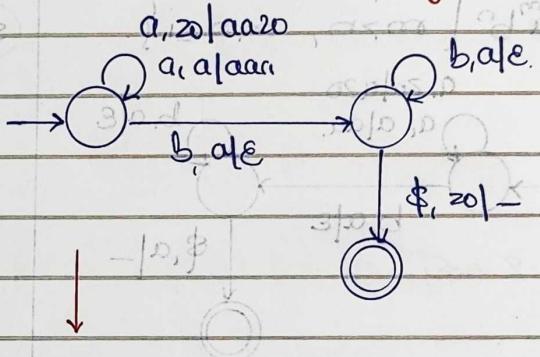
10.  $\{ a^m b^n \mid m, n \geq 1, m \neq n \}$

11.  $\{ a^m b^n \mid m, n \geq 0, m \neq n \}$

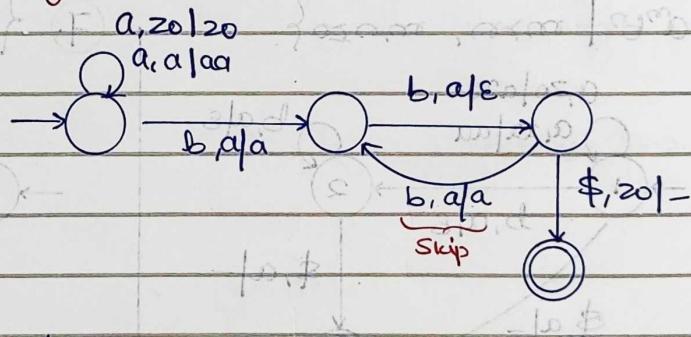
12.  $\{ b^n a^n \mid n \geq 1 \}$



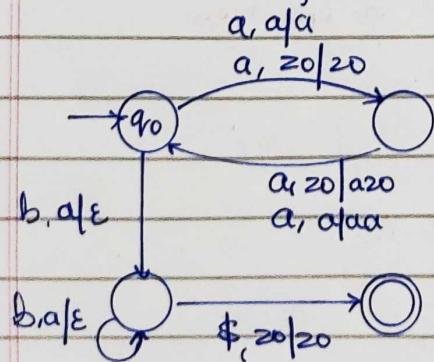
13.  $\{ a^n b^{2n} \mid n \geq 1 \}$  Logic 1:



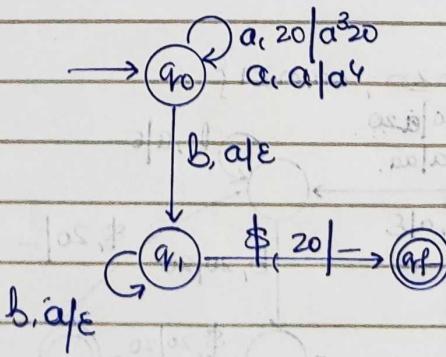
Logic 2:



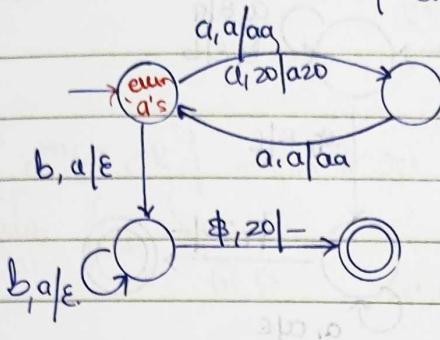
14.  $\{ a^n b^n \mid n \geq 1 \}$



15.  $\{ a^n b^{3n} \mid n \geq 1 \}$



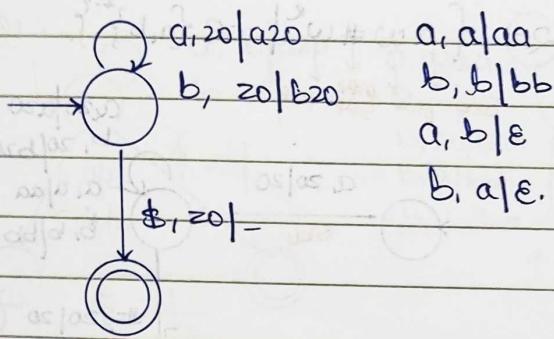
16.  $\{a^m b^n \mid m, n \geq 1, m = n \text{ even}\} = \{a^m b^n \mid m, n \geq 1, m = n \text{ even}\}$   
 $= \{a^2 b^2, a^4 b^4, \dots\}$



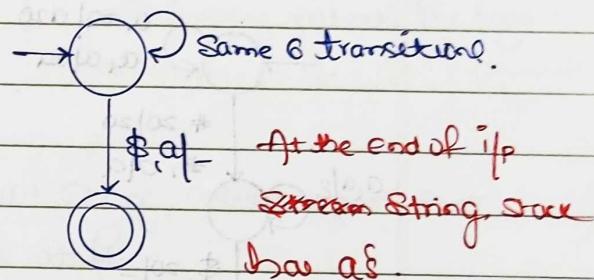
aaaaabbbbb \$

push a every b, pop 1 a  
for each a &  
guarantees even's

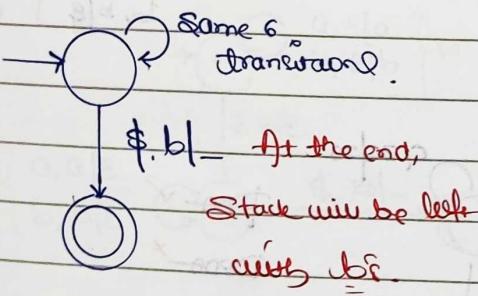
17.  $\{w \mid w \in \{a, b\}^*, \eta a(w) = \eta b(w)\}$



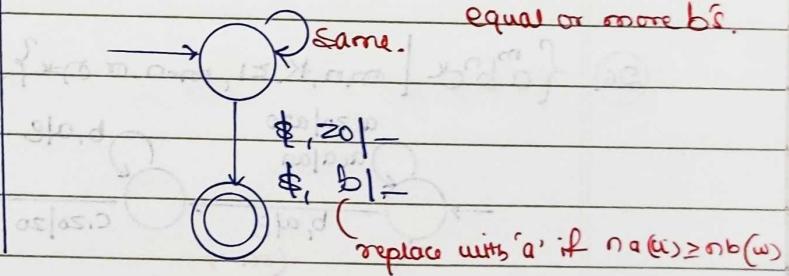
18.  $\{w \mid w \in \{a, b\}^*, \eta a(w) > \eta b(w)\}$



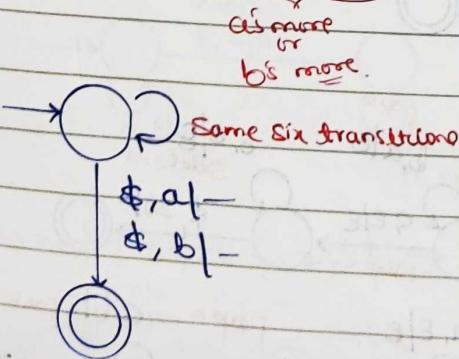
19.  $\{w \mid w \in \{a, b\}^*, \eta a(w) < \eta b(w)\}$



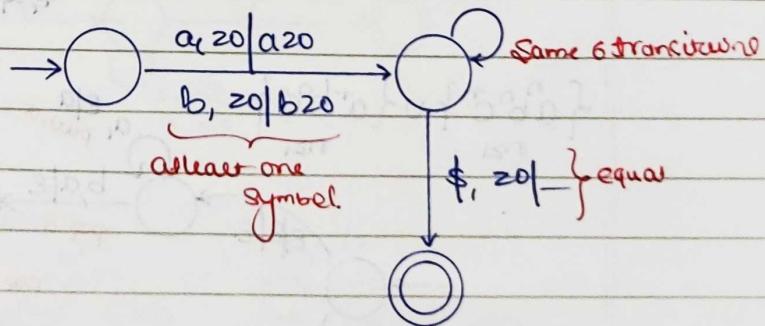
20.  $\{w \mid w \in \{a, b\}^*, \eta a(w) \leq \eta b(w)\}$



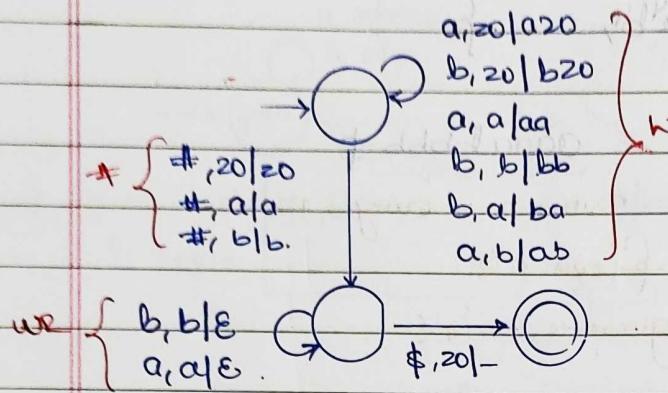
21.  $\{w \mid w \in \{a, b\}^*, \eta a(w) \neq \eta b(w)\}$



Note:  $L = \{w \mid w \in \{a, b\}^*, \eta a(w) = \eta b(w)\}$

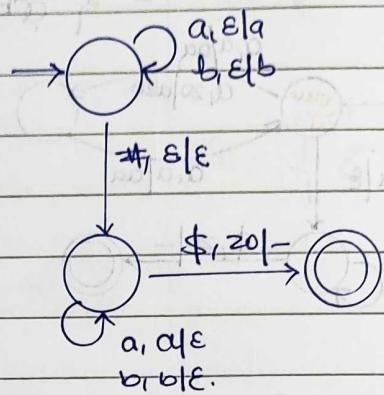


23.  $L = \{ w\#w^R \mid w \in \{a, b\}^*\}$ .

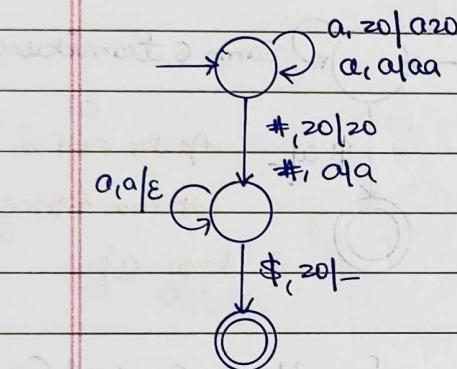


PDA ✓

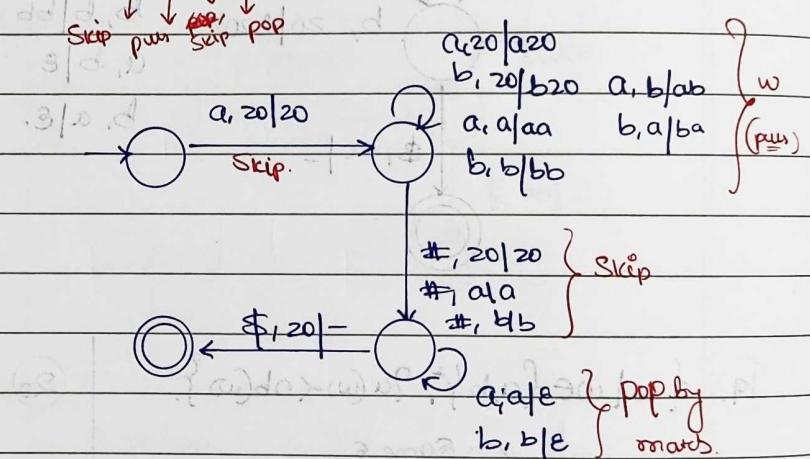
DPDA ✗



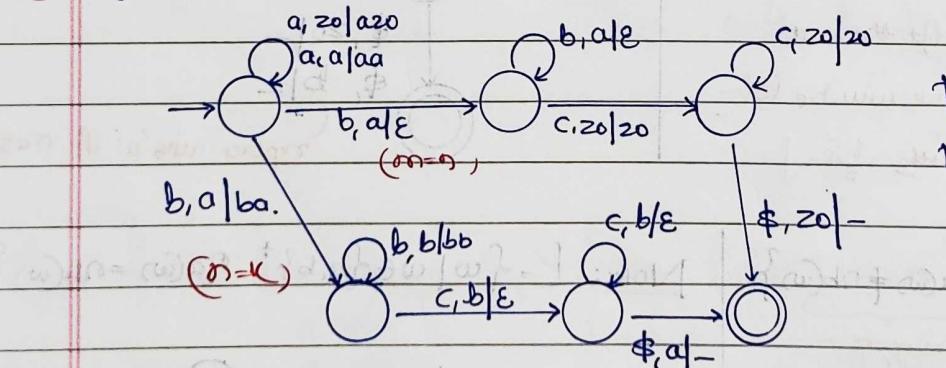
24.  $L = \{a^n \# a^n \mid n \geq 0\}$ .



25.  $\{a^m \# w^R \mid w \in \{a, b\}^*\}$ .

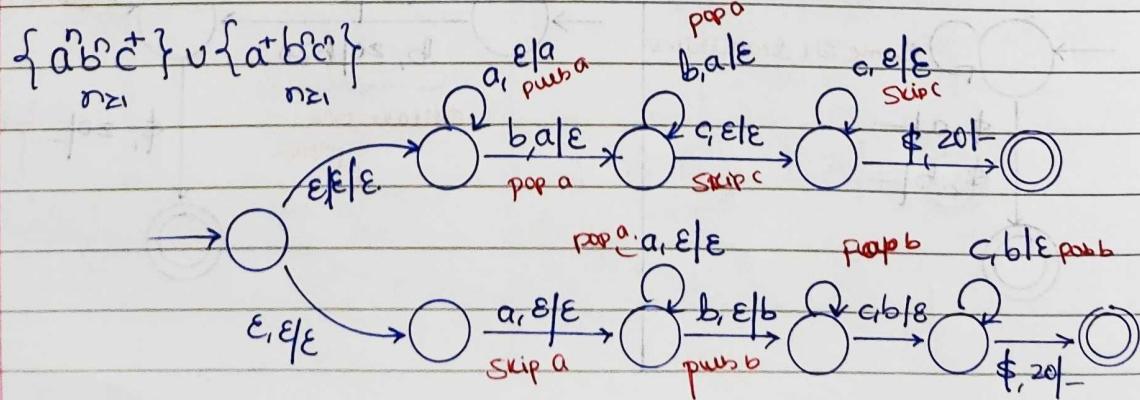


26.  $\{\overbrace{a^m b^n c^k}^{\text{m,n,k} \geq 1, m=n, \text{or } n=k}\}$ .



PDA ✓

DPDA ✗



(27)  $\{a^m b^n c^k \mid m, n, k \geq 1, m=n \text{ or } m=k\}$   
 pop  $a^i$  skip  $b^i$ .  
 $a^i = b^i$   
 $a^m b^n c^k$   
 $m \geq 1$   
 $a^i = c^i$   
 $a^m b^n c^k$   
 $m \geq 1$

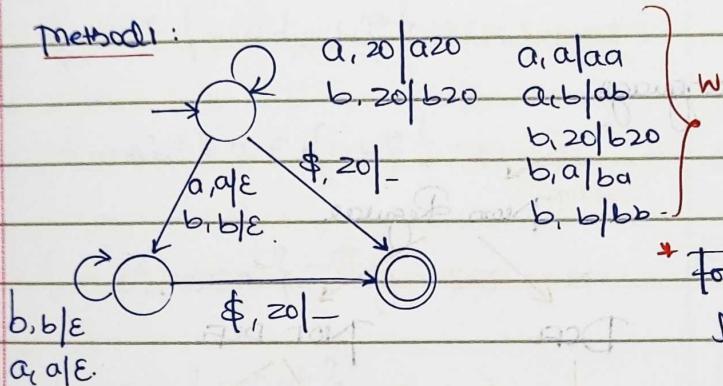
CFLs but  
not DCFLs.

(28)  $\{a^m b^n c^k \mid m, n, k \geq 1, m=k \text{ or } n=k\}$   
 push  $a^i$  push  $b^i$  for  $c^i$ .  
 $a^i = c^i$   
 $a^m b^n c^k$   
 $n \geq 1$   
 $a^i = b^i$   
 $a^m b^n c^k$   
 $n \geq 1$

without reading  $c^i$  pop  $b^i$ , then every  $c^i$  pop  $a^i$ .

(29)  $\{ww^* \mid w \in \{a, b\}^*\}$

Method 1:



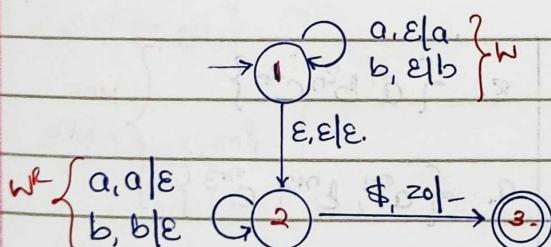
When can  $w^*$  begin?

Don't know.

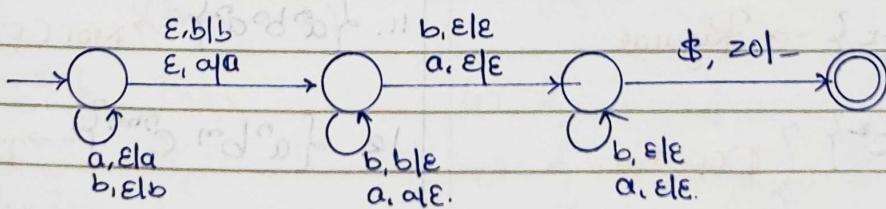
If  $w^*$  begins, where the logic?  
 present i/p = tos } previous i/p.

\* For valid string, atleast 1 path that starts at final.

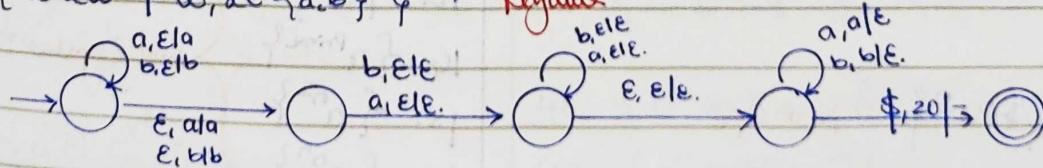
$\{ww^* \mid w \in \{a, b\}^*\} \rightarrow \text{CFL but not DCFL}$



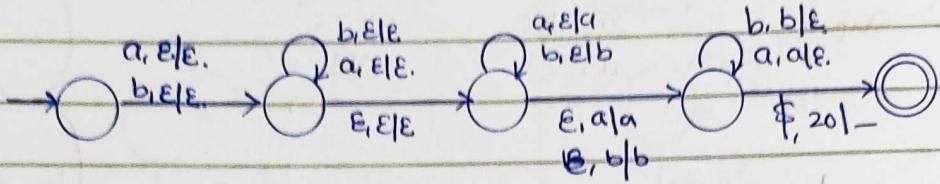
(30)  $\{ww^*x \mid w, x \in \{a, b\}^*\}$   $\rightarrow \text{CFL but not DCFL}$



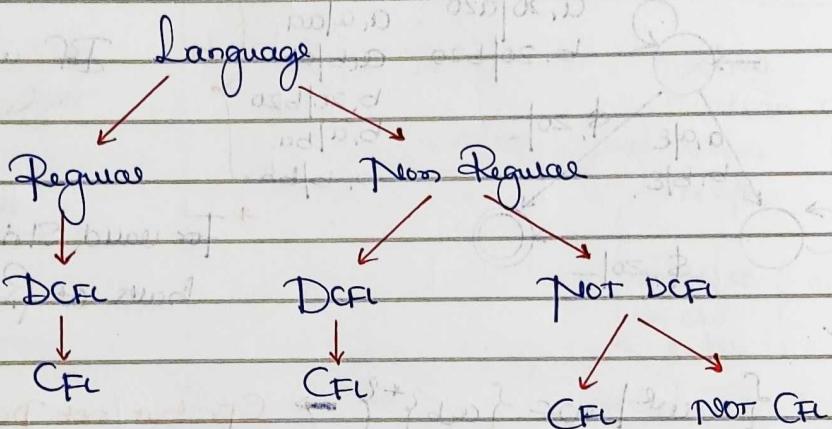
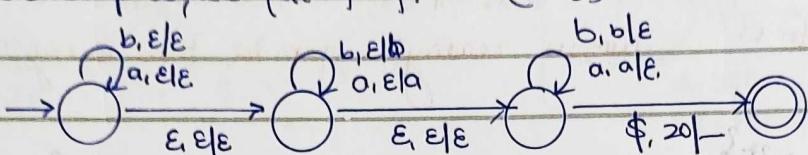
(31)  $\{w x w^* \mid w, x \in \{a, b\}^*\}$   $\rightarrow \text{Regular}$



32.  $\{xww^R \mid w, x \in \{a, b\}^*\}$ .  $\rightarrow$  CFL but not DCFL



33.  $\{xww^R \mid w, x \in \{a, b\}^*\}$ .  $\rightarrow (ab)^*$ .



1.  $\{a^n b^n\} \Rightarrow ab^* = \text{Reg.} \Rightarrow \text{DCFL} \Rightarrow \text{CFL}$

2.  $\{a^n b^n\}$  } DCFL but not regular

3.  $\{a^n b^{2n}\}$  } not regular

4.  $\{a^n b^n c^n\} \rightarrow$  Regular

5.  $\{a^n b^n c^m\}$  } DCFL but not regular

6.  $\{a^n b^n c^n\}$  } not regular

7.  $\{a^n b^n c^n\} \rightarrow$  NOT CFL

8.  $\{a^n b^{2n} c^{3n}\}$  } NOT CFL

a.  $\{a^{m1}, b^{m2}, c^{m3}\}$

10.  $\{a^{m1}, b^{m2}, c^{m3}\} \rightarrow$  Regular

11.  $\{a^n b^n a^n\} \rightarrow$  NOT CFL

12.  $\{a^n b^n c^{n+m}\} \rightarrow$  DCFL but ! Reg

13.  $\{a^n\}$

14.  $\{a^{\text{prime}}\}$

15.  $\{a^{n^2}\}$

16.  $\{a^{n^n}\}$

} NOT Regular  $\Rightarrow$  NOT CFL.

Note: \* If L is over 1 symbol  $\Rightarrow$  FA = PDA  
All non regulars are not CFLs.

$$18. \{a^n\} = a^* = \text{Regular.}$$

$$19. \{ww \mid w \in \{a,b\}^*\}$$

$\left. \begin{array}{l} \text{NOT} \\ \text{CFL} \end{array} \right\}$

$$20. \{w\#w \mid w \in \{a,b\}^*\}$$

$$21. \{ww^R \mid w \in \{a,b\}^*\} \rightarrow \text{CFL but not DCFL}$$

$$22. \{w\#w^R \mid w \in \{a,b\}^*\} - \text{DCFL but not Regular.}$$

$$23. \{ww \mid w \in a^*\} - \text{Reg} \Rightarrow \text{DCFL} \Rightarrow \text{CFL}$$

$$24. \{w\#w \mid w \in a^*\} - \text{DCFL but not Reg.}$$

$$25. \{a^nb^nc^nd^k\}$$

$\left. \begin{array}{l} \text{DCFL} \end{array} \right\}$

$$26. \{a^nb^kc^ld^m\}$$

$$27. \{a^n b^k c^m d^k\} - \text{not CFL}$$

$$28. \{a^m b^n c^k d^l\} - \text{DCFL}$$

$(m+n=k+l)$

$$29. \{a^2 b^3\} - \text{DCFL}$$

$$30. \{a^n b^k\}$$

$$31. \{a^2 b^k\}$$

$\left. \begin{array}{l} \text{NOT} \\ \text{CFL} \end{array} \right\}$

$$32. \{a^nb^nc^{k^2}\}$$

$$33. \{w \mid w \in \{0,1\}^*, n_0(w) = n_1(w)\}$$

$\left. \begin{array}{l} \text{DCFL} \end{array} \right\}$

$$34. \{wwx \mid w, x \in \{a,b\}^*\}$$

$\left. \begin{array}{l} \text{Also: } wzw, \quad wwx, \\ \quad xww, \quad xwx \\ \quad wwx, \quad wwx \end{array} \right\} \begin{array}{l} \text{Regular} \\ \downarrow \\ \text{DCFL} \\ \downarrow \\ \text{CFL} \end{array}$

$$35. \{wwz \mid w, z \in \{a,b\}^*\}$$

$$36. \{wzw \mid " " "\}$$

$\left. \begin{array}{l} \text{NOT} \end{array} \right\}$

$$37. \{xww \mid " " "\}$$

$\left. \begin{array}{l} \text{CFL} \end{array} \right\}$

$$38. \{wwz \mid " " "\}$$

$\left. \begin{array}{l} \text{CFL but not} \\ \text{DCFL} \end{array} \right\}$

$$39. \{xww \mid " " "\}$$

$\left. \begin{array}{l} \text{DCFL} \end{array} \right\}$

$$40. \{wzw \mid " " "\} - \text{Regular}$$

$$41. \{ww^R z z \mid w, z \in \{a,b\}^*\} - \text{CFL but not DCFL}$$

$$42. \{wwxx \mid w, x \in \{a,b\}^*\}$$

$$43. \{ww^R w \mid " " "\}$$

$\left. \begin{array}{l} \text{NOT} \\ \text{CFL} \end{array} \right\}$

$$44. \{www \mid " " "\}$$

$$45. \{w\#w\#w \mid " " "\}$$

$$46. \{a^n \# a^n \# a^n\}$$

$\left. \begin{array}{l} \text{NOT CFL} \end{array} \right\}$

$$47. \{a^n \# b^n \# c^n\}$$

$\left. \begin{array}{l} \text{NOT CFL} \end{array} \right\}$

48.  $\{w\#w^R \# x \# x^R \mid w, x \in \{a, b\}^*\}$  - DCFL

49.  $\{a^n \# b^n \# c^k\}$  } Regular

50.  $\{a^n \# a^n \# a^k\}$

51.  $\{a^m b^n c^k \mid m=n \text{ or } m=k\}$

52.  $\{a^m b^n c^k \mid m < n \text{ or } m > k\}$

53.  $\{a^m b^n c^k \mid m \neq n \text{ or } m \leq k\}$

54.  $\{a^m b^n c^k \mid m \neq n \text{ or } m \neq k\}$

55.  $\{a^m b^n c^k \mid \text{if } (m \neq n) \text{ then } m=k\}$

CFL but not

DCFL

56.  $\{a^m b^n \mid m \neq n \text{ or } m = n\} = a^+ b^+ - \text{Regular}$

57.  $\{a^m b^n c^k \mid m=n \text{ and } m=k\} = a^+ b^+ c^+$

not

CFL

58.  $\{a^m b^n c^k \mid m \neq n \text{ and } m \leq k\}$

59.  $\{a^m b^n c^k \mid m \neq n \text{ or } (m \leq k)\}$  - CFL but not DCFL

60.  $\{a^m b^n\}$  - DCFL

61.  $\{a^m b^n c^k\}$  - CFL but not DCFL

62.  $\{ww \mid w \in \{ab\}^*\}$  - CFL but not DCFL

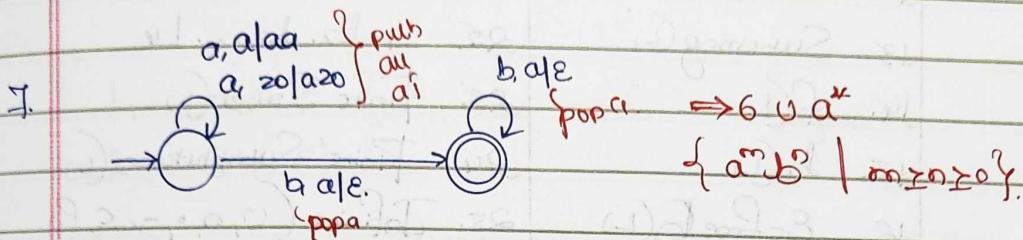
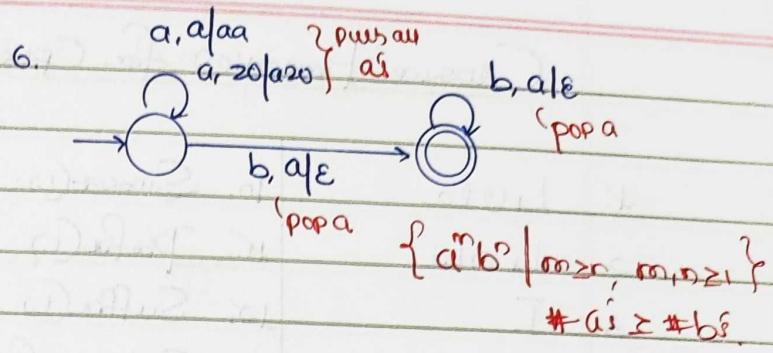
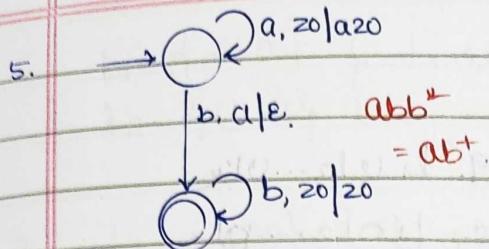
POA uses final state

1.  $\rightarrow \textcircled{O} \xrightarrow{a, z_0/z_0} \textcircled{O} \quad L = \{\epsilon, a\}$

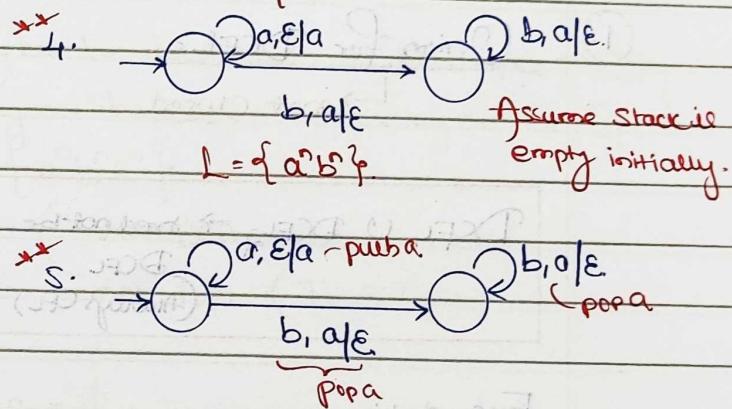
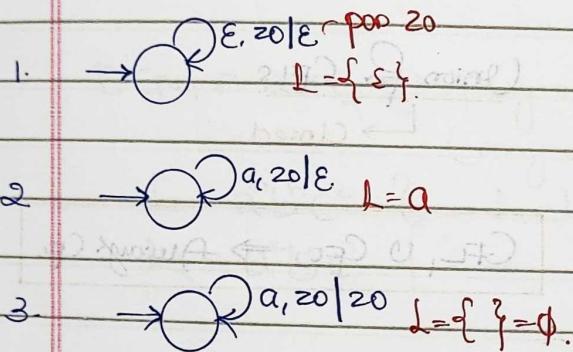
2.  $\rightarrow \textcircled{O} \xrightarrow{a, z_0/z_0} \textcircled{O} \quad L = a^*$

3.  $\rightarrow \textcircled{O} \xrightarrow{a, \epsilon/a} \textcircled{O} \quad \text{push } a \quad L = a^*$

4.  $\rightarrow \textcircled{O} \xrightarrow{a, z_0/z_0} \textcircled{O} \quad L = \{ab\}$   
 $\text{It pushes only } 1^{\text{st}} \text{ 'a'}$   
 $\text{pop } a \quad \text{pop } a$



POA uses empty stack: (If string is valid, show that stack is empty at the end of 'i(p.)')



Closure Properties for DCFNL (Remember Closed Operations).

1. $L_1 \cup L_2$	12. Suffix( $L$ )	23. Finite Subset( $L$ )
2. $L_1 \cap L_2$	13. Substring( $L$ )	24. Finite Substitution ( $L$ )
3. $\bar{L}$	14. $f(L)$	25. $\{\text{finite}\} \cup \{\cup, \cap, -, \circ, \subseteq, +\}$
4. $L_1 - L_2$	15. $\bar{L}(L)$	
5. $L_1 \cdot L_2$	16. $\epsilon$ -Free ( $L$ )	
6. $L_1^{\text{Rev}}$	17. $\bar{L}^{-1}(L)$	
7. $L_1^*$	18. $L_1 / L_2$	
8. $L_1^+$	19. $L_1 \cup L_2 \cup L_3 \dots \cup L_K$	
9. $L_1 \Delta L_2$	20. $L_1 \cap L_2 \cap L_3 \dots \cap L_K$	
10. Subset( $L$ )	21. $L_1 - L_2 - L_3 \dots - L_K$	
11. Prefix( $L$ )	22. $L_1 \cdot L_2 \cdot L_3 \dots \cdot L_K$	

## Closure Properties for CFLs

- |                     |                      |   |
|---------------------|----------------------|---|
| 1. $L_1 \cup L_2$   | 10. Subset( $L$ )    | 19. $L_1 \cup L_2 \dots \cup L_k$       |
| 2. $L_1 \cap L_2$   | 11. Prefix( $L$ )    | 20. $L_1 \cap L_2 \dots \cap L_k$       |
| 3. $\bar{L}$        | 12. Suffix( $L$ )    | 21. $L_1 - L_2 - L_3 \dots - L_k$       |
| 4. $L_1 - L_2$      | 13. Substring( $L$ ) | 22. $L_1, L_2, L_3 \dots, L_k$          |
| 5. $L_1 \cdot L_2$  | 14. $f(L)$           | 23. Finite Subset( $L$ )                |
| 6. $L^{\text{Rev}}$ | 15. $h(L)$           | 24. Finite Substitution( $L$ )          |
| 7. $L^+$            | 16. E-free( $L$ )    | 25. Infinite ( $0, 0^*, -, \Sigma, f$ ) |
| 8. $L^{\perp}$      | 17. $h^{-1}(L)$      |   |
| 9. $L_1 \Delta L_2$ | 18. $L_1 / L_2$      |   |

① Union for DCFLs.

→ not closed

$\text{DCFL}_1 \cup \text{DCFL}_2 \rightarrow$  need not be  
DCFL  
(Always CFL)

Example 1:

$\text{DCFL}_1 \cup \text{DCFL}_2 \rightarrow$  not DCFL possible  
 $\{a^n b^m c^n\} \cup \{a^m b^n c^n\} = \{a^m b^n c^n\}$   
 $m=n$  or  $n=k\}$ . (not DCFL)

Example 2:

$\text{DCFL}_1 \cup \text{DCFL}_2 \rightarrow$  DCFL possible

$a^* \cup a^* \rightarrow a^*$  DCFL.

Union for CFLs

→ Closed

$CFL_1 \cup CFL_2 \rightarrow$  Always CFL

Algorithm:

$CFL_1 \rightarrow$  write CFG, with  $S_1$  as start

$CFL_2 \rightarrow$  " " " " "  $S_2$  "



Add new start  $S$

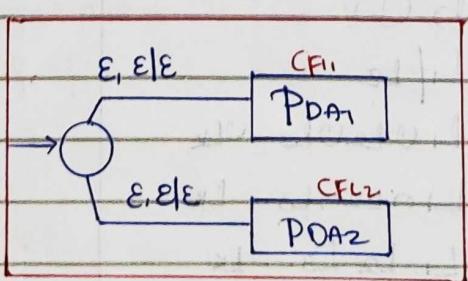
and Append

$S \rightarrow S_1 | S_2$

$S \rightarrow S_1 | S_2$

$CFG_1$

$CFG_2$



$CFL_1 \cup CFL_2$

$$\left. \begin{array}{l} L_1 = \{a^n b^n\} \\ L_2 = \{b^n a^n\} \end{array} \right\} L_1 \cup L_2 = \{a^n b^n\} \cup \{b^n a^n\} = DCFL$$

$$\left. \begin{array}{l} L_1 = \{a^n b^n\} \\ L_2 = (a+b)^n \end{array} \right\} L_1 \cup L_2 = (a+b)^n \rightarrow \text{Regular.}$$

$$\left. \begin{array}{l} L_1 = a^* b^* c^* \\ L_2 = a^* b^* c^+ \end{array} \right\} L_1 \cup L_2 = a^* b^* c^* \rightarrow \text{Regular}$$

(2) Intersection for DCFLs  
 ↳ not closed

Intersection for CFLs  
 ↳ not closed

$DCFL_1 \cap DCFL_2 \rightarrow$  May or may not be DCFL  
 (may or may not be CFL)

$$\left. \begin{array}{l} L_1 = a^* b^* c^* \\ L_2 = a^* b^* c^* \end{array} \right\} L_1 \cap L_2 = \{a^n b^n c^n\}$$

(not DCFL  
 not CFL (It is CSL))

$CFL_1 \cap CFL_2 \rightarrow$  may or  
 may not be CFL.

$$1. L_1 = \emptyset, L_2 = \{a^n b^n\} \rightarrow L_1 \cap L_2 = \emptyset$$

$$2. L_1 = \Sigma^*, L_2 = \{a^n b^n\} \rightarrow L_1 \cap L_2 = L_2 = \{a^n b^n\}$$

$$3. L_1 = \{a^n b^n c^n\}, L_2 = \{a^n b^n c^n\} = L_1 \cap L_2 = \{a^n b^n c^n\}$$

(3) Complement for DCFLs  
 ↳ Closed

$\overline{DCFL} =$  always DCFL

$L$  is DCFL

↓  
 Construct DPDA

↓  $f \leftrightarrow nf$ .

Modified DPDA

↓  
 $\overline{L}$  is DCFL

Complement for CFLs  
 ↳ not closed

$\overline{CFL} =$  either CFL or not CFL

$L = \{a^n b^n c^n\}$  is CFL

$\overline{L} = \{a^n b^n c^n\}$  is not CFL  
 (it is CSL)

$$\{a^nb^m\} = (a+b)^* - \{a^n b^n\}$$

$$= \sum^* ba \Sigma^* \cup \{a^m b^l \mid m \neq l\}$$

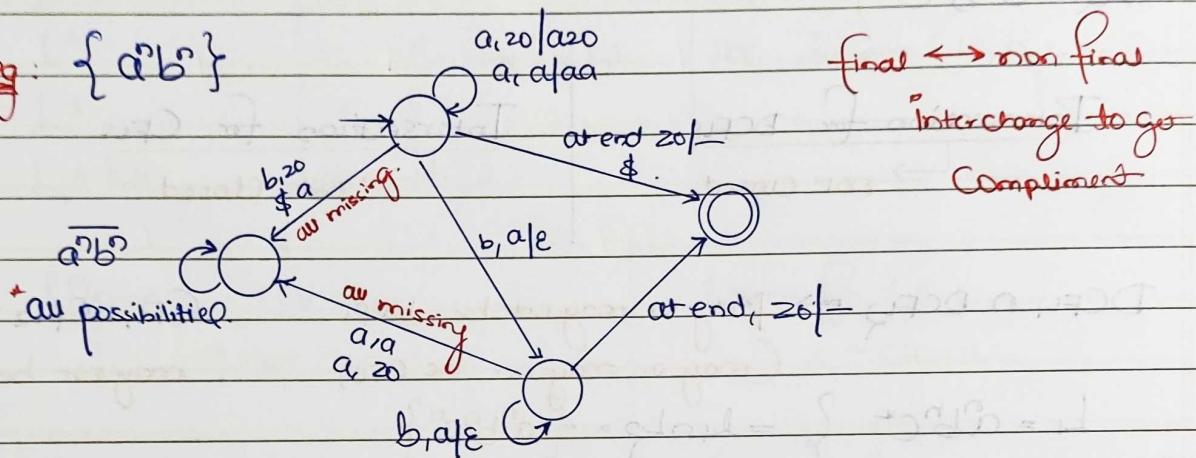
If the string  
is not in  $a^nb^n$

If the string is  
in  $a^nb^n$ .

$$L = a^* b^* = \{a^m b^n \mid m=n \text{ or } m \neq n\}$$

$$\hookrightarrow I = \sum^* ba \Sigma^*$$

Ex:  $\{a^n b^n\}$



$$L = \{a^n b^n c^k\} = (a+b+c)^* - \{a^n b^n c^k\}$$

$$= (a+b+c)^* - \{a^m b^n c^k \mid m=n=k\}$$

= CFL but not DCFL

1.  $\{a^n b^n\} \rightarrow$  DCFL

	DCFLs	CFLs
U	X	✓
N	X	X
I	✓	X

2.  $\{a^n b^n c^k\} \rightarrow$  CFL but not DCFL

3.  $\{a^nb^*\} \rightarrow$  Regular

4.  $\{a^m b^n c^k \mid m < n < k\} \rightarrow$  CFL but not DCFL

Note: I. Complement of a CFL is either CFL or not CFL.

II. Complement of not CFL is may or may not be CFL.

III. Complement for CFLs is not closed.

④

Difference

- Not closed for DCFLs
- Not closed for CFLs

$$L_1 - L_2 = L_1 \cap \bar{L}_2$$

- I.  $DCL_1 - DCL_2 \rightarrow DCL_1 \cap \bar{DCL_2} \Rightarrow DCL_1 \cap DCL_3 \rightarrow$  need not be DCL  
 II.  $CFL_1 - CFL_2 \rightarrow CFL_1 \cap \bar{CFL_2} \rightarrow$  need not be CFL

$L_1$  is DCL

$L_2$  is CFL

$$L \cup \bar{L} = \Sigma^*$$

1.  $L \cup \bar{L}$  is always CFL
2.  $L_1 \cap \bar{L}_2$  is need not be CFL
3.  $\bar{L}_1 \cup \bar{L}_2$  is need not be CFL
4.  $\bar{L}_1 \cup L_1$  is  $\Sigma^*$  is regular
5.  $\bar{L}_2 \cup L_2$  is  $\Sigma^*$  is regular.

(5) Concatenation:

- Not closed for DCLs
- Closed for CFLs

$DCL_1 \cdot DCL_2 \rightarrow$  need not be DCL (Always CFL)

$CFL_1 \cdot CFL_2 \rightarrow$  Always CFL

$$L_1 = \{a^n b^n\}$$

$$L_2 = \{a^n b^{2n}\}$$

$L_1$  is CFL = CFG<sub>1</sub>(S<sub>1</sub>)

$L_2$  is CFL = CFG<sub>2</sub>(S<sub>2</sub>)

[Add  $S \rightarrow S_1 S_2 \rightarrow S_1 \rightarrow S_2$  CFG<sub>1</sub>, CFG<sub>2</sub>.]

$\underbrace{aaaab}_{\text{push.}}$

⇒  $\{a^n b^n a^k b^k\} \rightarrow$  DCL

⇒  $\{a^n b^n a^k b^{2k}\} \rightarrow$  CFL but not DCL

⇒  $\{a^n b^n a^k b^{2k} \mid n \geq 1\} \rightarrow$  DCL

always comes

(6) Reversal

- Not closed for DCLs
- Closed for CFLs

I.  $(DCL)^{\text{rev}} \rightarrow$  need not be DCL

$L = \{a^n b^n a^k b^{2k} \mid n \geq 1, k \geq 0\}$  "is DCFL".

$L^{Rev} = \{b^{2k} a^k b^2 a^n \mid n \geq 1, k \geq 0\}$  is not DCFL

$L$  is CFL

$L = a^n b^n$

$S \Rightarrow aSb \mid \epsilon$

every rule, recursive

$S \Rightarrow bSa \mid \epsilon$

↓

$L = b^n a^n$  CFL

⑦ Kleene Star: → Not Closed for DCFLs

→ Closed for CFLs

⑧ Kleene Plus: → Not closed for DCFLs

→ Closed for CFLs

$L$  is CFL  $\rightarrow L^*$  need not be CFL

$L = \{a^n\}$

$S \Rightarrow a$

add new start  $x$  and Add

$x \rightarrow xS \mid \epsilon$

$S \Rightarrow a$

$L = a^* \text{ is CFL}$

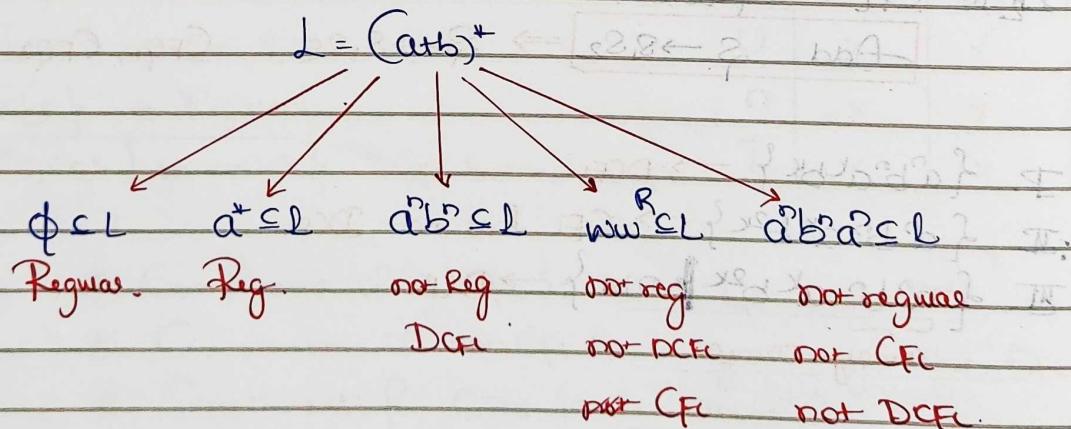
$$L_1 \Delta L_2 = (L_1 - L_2) \cup (L_2 - L_1)$$

$$= (L_1 \cup L_2) - (L_1 \cap L_2)$$

→ Not closed for DCFLs & CFLs

⑩ Subset of DCFL is need not be DCFL.

Subset of CFL need not be CFL.



# Push Down Automata (Continued)

Regular Closure for DCFLs:

1. DCFL  $\cup$  Regular
  2. DCFL  $\cap$  Regular
  3. DCFL - Regular
  4. Regular - DCFL
- $\left. \begin{array}{l} \text{Always DCFL} \\ (\text{need not be regular}) \end{array} \right\}$

Note: \* DCFL  $\cap$  Regular  $\rightarrow$  Always DCFL

\* Set of all DCFLs  $\cap$  Set of all regulars  $\rightarrow$  Set of all regulars

Regular Closure for CFLs:

1. CFL  $\cup$  Regular
  2. CFL  $\cap$  Regular
  3. CFL - Regular
  4. Regular - CFL  $\rightarrow$  need not be CFL.
- $\left. \begin{array}{l} \text{Always CFL} \\ \text{CFL} \end{array} \right\}$

I. DCFL

U
n
-
1.

Regular = DCFL

II. CFL

U
n
-
1.

Regular  $\Rightarrow$  CFL

$$1. L = \{a^n b^n\} \Rightarrow \text{prefix}(L) = \{a^m b^n \mid m \leq n\}$$

$$\text{suffix}(L) = \{a^m b^n \mid m \leq n\}$$

$$\text{substring}(L) = \{a^* b^*\}$$

$$2. L = \{a^n b^n\}, f(a) = \{0^n\}, f(b) = \{1^n\} \Rightarrow f(L) = 0^n 1^n = \{0^n 1^{2n}\}$$

$$3. L = \{ww^T \mid w \in \{a, b\}^*\} \Rightarrow \text{Prefix}(L) = (a+b)^*$$

$$\text{suffix}(L) = (a+b)^*$$

$$\text{substring}(L) = (a+b)^*$$

Prefix (DCFL) is DCFL

DCFL  $\rightarrow$  DPDA  $\rightarrow$  mod DPDA  $\rightarrow$  pref(DCFL)

From initial to final, \$ can come from any state.

Note: Language  $L$  satisfies prefix property if every  $w$  in  $L$  is not prefix of any other string in  $L$ .

Q1.  $L = a^*$ , does this satisfy Prefix property?

→ No

Q2.  $L = \{a^n b^n \mid n \geq 1\}$

→ Satisfies prefix property

\* If any DCFL satisfies prefix property then we can make DPDA with empty stack.

$\boxed{\text{DPDA} \equiv \text{DPDA with final state}}$

$\neq \text{DPDA with empty stack}$

$\boxed{\text{DPDA with empty stack} \subset \text{DPDA}}$

→ Set of all DCFLs

$\subseteq$

Set of all languages accepted by DPDA

$\subseteq$

Set of all languages accepted by DPDA with final state

$\subseteq$

Set of all languages generated by LR(1) CFGs

PDA  
with FS  $\equiv$  with E.S.

DPDA  
with FS  $\neq$  with E.S.  
Same as  
DPDA  
(less powerful)

$\text{CFG} \equiv \text{CFL} \equiv \text{PDA} \equiv \text{PDA with FS} \equiv \text{PDA with E.S.}$

$\text{DCFL} \equiv \text{DPDA} \equiv \text{DPDA with FS} \equiv \text{LR}(k) \text{ CFG}$   
( $k \geq 1$ )

### Inherently Ambiguous CFL:

1.  $a^*$  - Regular but not InL
  2.  $a^nb^n$  - DCFL but not InL
  3.  $\{ww^R \mid w \in \{ab\}^*\}$  - CFL, not DCFL, and not InL
  4.  $\{a^m b^n c^k \mid m=n \text{ or } n=k\}$  - CFL but not DCFL, it is InL
- there is no Unambiguous CFG in the world.

### Ambiguous CFG

$\exists w, \geq 1 \text{ PT}$

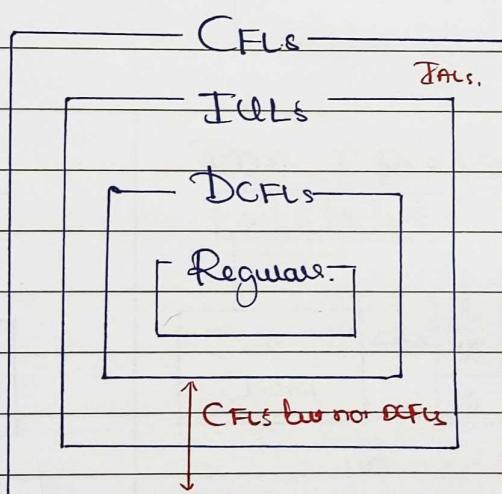
### InL

$L$  is InL if every CFG that represents  $L$  is Ambiguous.

$L$  has no Unambiguous CFG

### Per

$L$  is Per if some Unambiguous CFG exist for  $L$ .



- \* Every Regular is Per
- \* Every DCFL is InL
- \* CFL is either Per or InL.

$\{ww^R \mid w \in \{ab\}^*\}$

- CFL but not DCFL
- Unambiguous CFG

We can show Unambiguous CFG

$S \rightarrow aSa \mid bSb \mid \epsilon$

Unambiguous CFG

$\{a^m b^n c^k \mid m=n \text{ or } n=k\}$

- CFL but not DCFL
- Ambiguous CFL

\* There is no Unambiguous CFG for this language

If  $L$  is Unambiguous CFL  $\rightarrow$  Show atleast one Unambiguous CFG

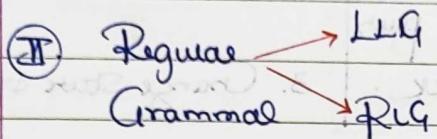
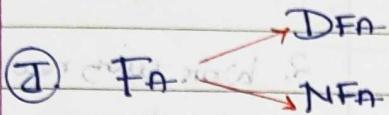
If  $L$  is Ambiguous CFL  $\rightarrow$  we don't have Unambiguous CFG

Every CFG is ambiguous

1.  $L = a^* b^* \rightarrow \text{Reg}$  } True
2.  $L = a^* b^n \rightarrow \text{DCFL}$  }
3.  $L = \{a^m b c^k \mid m=n \text{ or } m=k\} \rightarrow \text{CFL, but not DCFL, PSL}$
4.  $L = \{a^m b^n c^k\} \rightarrow \text{DCFL}$
5.  $L = \{w \# w^r \mid w \in \{a, b\}^*\} \rightarrow \text{DCFL}$  } True
6.  $L = \{ww^r \mid w \in (a+b)^*\} \rightarrow \text{CFL, but not DCFL}$

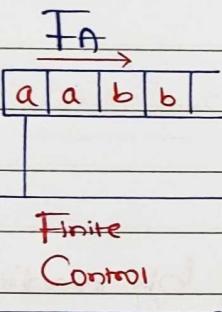
# TURNING MACHINE Recursively Enumerable

Regular



III Regular Expressions

\* These problems can be solved without using memory.



$$FA = (Q, \Sigma, S, q_0, F)$$

$$SDFA: Q \times \Sigma \rightarrow Q$$

$$SNFA: Q \times \Sigma \xrightarrow{\epsilon} 2^Q$$

CFs

PDA

CFG

Recursively Enumerable Languages (REs)

I TM

II Unrestricted Grammars

\* Can be solved using 1 stack with non-determinism



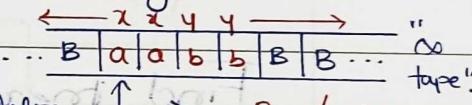
$$PDA =$$

$$(Q, \Sigma, S, q_0, F, Z_0, \delta)$$

$$\delta_{PDA} = Q \times \Sigma \times T \xrightarrow{*} 2^{Q \times r^+}$$

$$\delta_{DPDA} = Q \times \Sigma \times T \xrightarrow{*} Q \times T^*$$

Turing Machine



Finite Control

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

$$T = \{a, b, B, z, y\}$$

$$\text{alphabet: } \Sigma \subseteq T$$

$$TM = (Q, \Sigma, S, q_0, F, B, T)$$

$$S_{TM}: Q \times T \rightarrow Q \times T \times \{L, R\}$$

$$S_{NTM}: Q \times T \rightarrow 2^{Q \times T \times \{L, R\}}$$

B → Blank Symbol.

\* PDA that uses finite stack  $\equiv FA \equiv$  Regular Language

\*  $FA + R/W \text{ tape} + \text{Bidirectional} + \text{Infinite tape} \equiv \text{Turing m/c}$

Fa

To remember one symbol 'a':



1. Change State.

PoF

1. Change State  
or

2. Push/pop into stack  
or

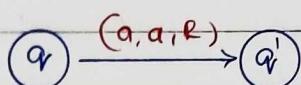
3. Change State and push  
or pop into stack

Tm

1. Change State  
or

2. Write with new symbol  
on tape

3. Change State & write



a  
a

↑↑  
Read 'a'.

$$\delta(q, a) = (q', a, R)$$

↓      ↓      ↓      ↓      ↓      ↓  
 present state    Read    new state    write    direction

x  
a

↑  
q   q

Read 'a' but write with 'x'

$$\delta(q, a) = (q', x, R)$$

### Find meaning of transitions

1.  $\delta(q, B) = (s, a, \leftarrow_{\text{left}})$  → From state q, by reading B, goes to s, by writing with a and move left.

2.  $\delta(q, a) = (q', a, \rightarrow_{\text{right}})$  → Read 'a' move right.

3.  $\delta(q, a) = (q', B, L)$  →  $|p=a$ , Replacing a with B left direction.

### Turing Machine

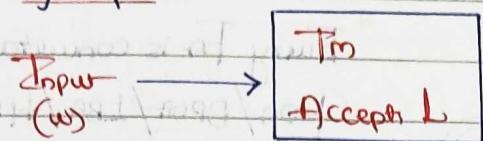
\* It represents REI (Recursively Enumerable Language)

\* It can accept REI

OR

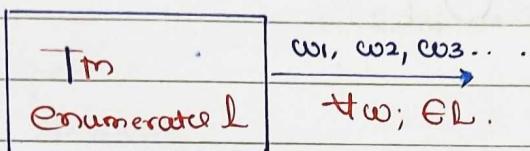
\* It can enumerate REI

Acceptor:



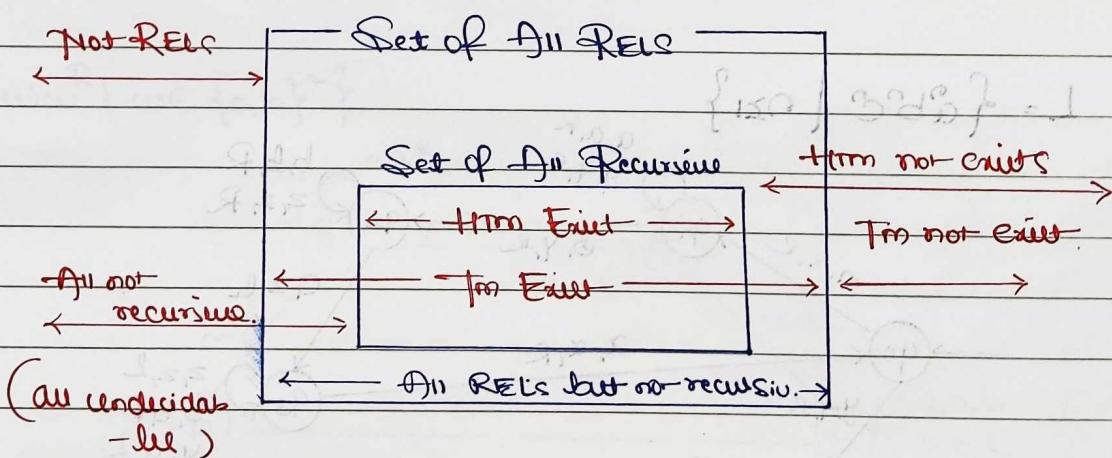
If  $w \in L$ ,  $Tm$  halts at final state.  
If  $w \notin L$ , either halts at non-final  
or never halts.

Enumerator:



$Tm \Rightarrow$  logic exist for valid strings  
Accepts REL (logic may or maynot exist for invalid string)

Having  $Tm$   $\leftarrow HTM \Rightarrow$  logic exist for valid and invalid  
Accepts Recursive language (Decidable)       $\downarrow$        $\downarrow$   
    final or non final

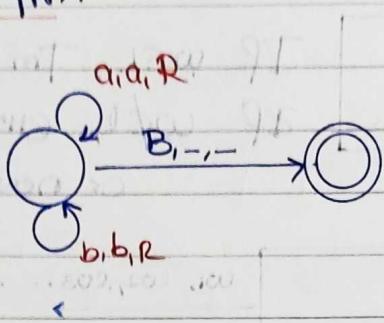


1. Recursive Language (decidable language)
2. REl (semi-decidable language)
3. REl but not recursive
4. NOT REl
5. Undecidable language

	Tm	HTM
1. Recursive Language (decidable language)	✓	✓
2. REl (semi-decidable language)	✓	Don't know.
3. REl but not recursive	✓	Don't know.
4. NOT REl	✗	✗
5. Undecidable language	Don't know	✗

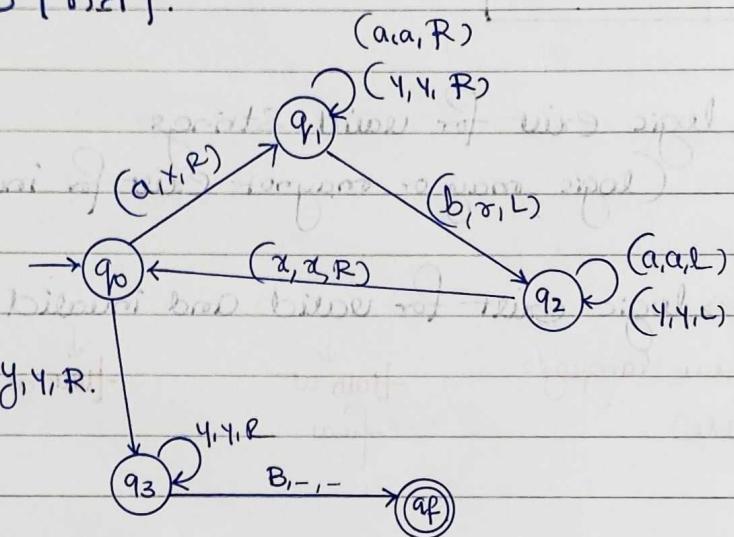
### Construction of Tm:

1.  $L = (a+b)^*$ .

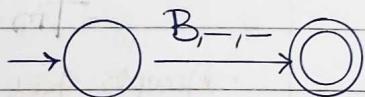


\* Every FA is convertible to Tm/  
PDA/DPA/LBA/TMs

2.  $\{a^n b^n \mid n \geq 1\}$ .

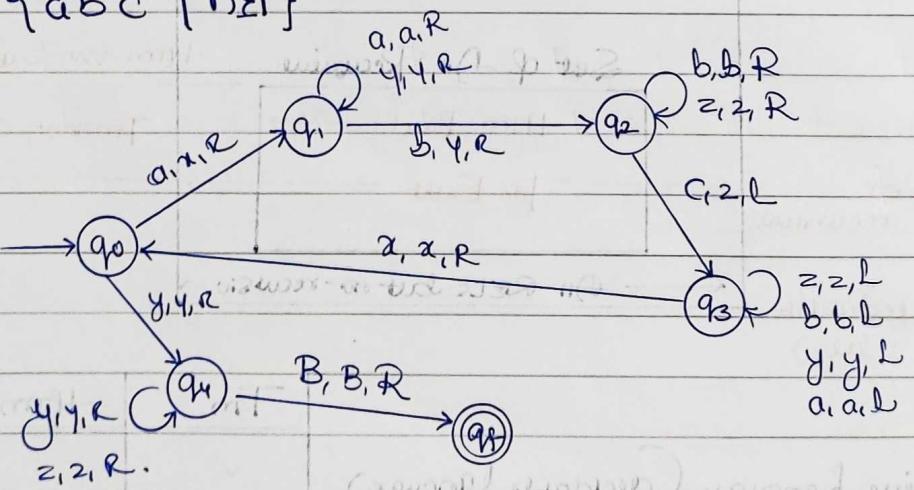


3.  $L = \{\epsilon\}$

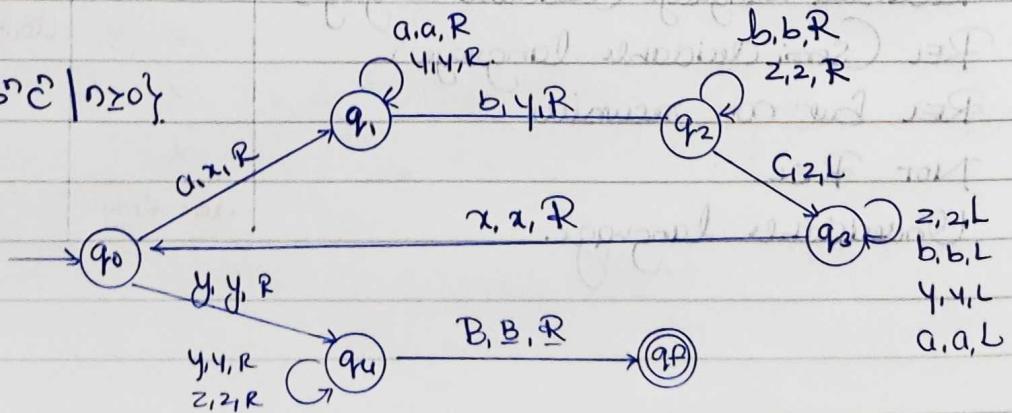


Tm accepts only ε

4.  $L = \{a^n b^n c^n \mid n \geq 1\}$



5.  $L = \{a^n b^n c^n \mid n \geq 0\}$



Trn.

initial state

11

- Acceptance  
Hans at final  $\rightarrow$  Accepted  
Hans at { }  $\rightarrow$  Non-Accepted.  
non final  $\rightarrow$  Accepted.

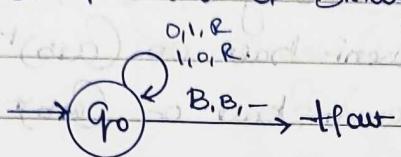
Functionality

I's compliment  
Q's compliment

Having & non-having

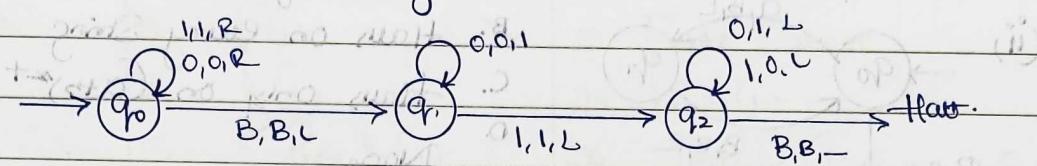
Behaviour of  
Tr/c

6. I's Compliment of Binary.

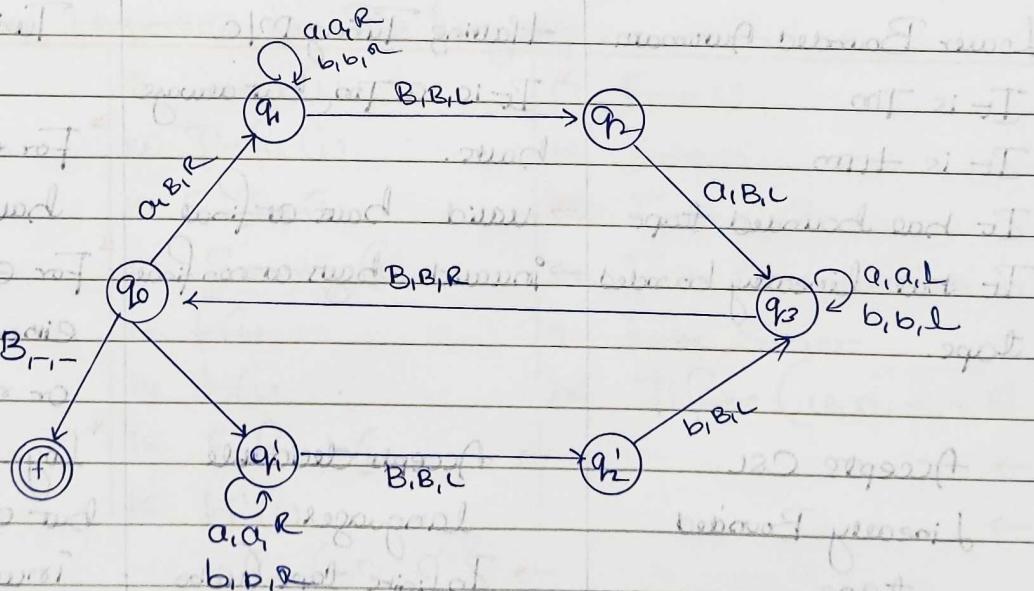


\* We do not need final state. There is no concept of valid and invalid input.

7. Q's Compliment Of Binary.



8.  $\{ww^R \mid w \in \{a,b\}^*\}$ .

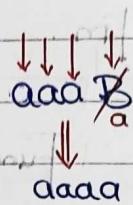
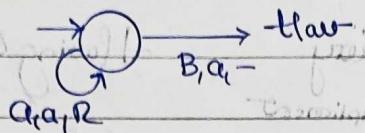


9.  $\{w#w^R \mid w \in \{a,b\}^*\}$ .

10.  $\{w#w \mid w \in \{a,b\}^*\}$ .

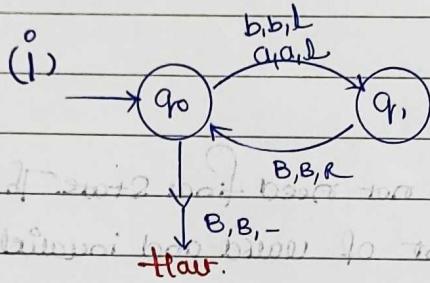
11.

## Unary addition

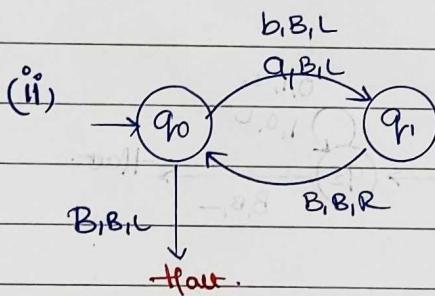


Q1. What is the language accepted by Tm? What is the functionality?

Ans



A. Halts only at E

B. Halts only on  $(ab)^+$ C. Doesn't halt on  $(ab)^+$ D. Doesn't halt on  $(ab)^*$ 

A. Halts only on E

B. Halts on every String

C. Halts only on  $(ab)^*$ 

D. None

- \* Lower Bounded Automata
- \* It is Tm
- \* It is LTM
- \* It has bounded tape
- \* It has linearly bounded tape.

- Accepts CSL
- Linearly Bounded tape
- Always halts

## Halting Turing M/c

- \* It is a Tm, but always halts.

- valid halts at final

- invalid halts at non final

- Accepts decidable languages

- Infinite tape (unbounded).

- Always halts

## Turing M/c

- \* For every valid string halts at final

- \* For every invalid string, either halts at non final or never halts

- \* Logic exist for valid but don't know about invalid.

- Accepts REI
- Infinite tape (unbounded)

### CSLs

- \* Every regular
- \* Every CFL.
- \*  $a^p b^q c^r$
- \*  $a^{\text{prime}}$
- \*  $\{ww \mid w \in \{a,b\}^*\}$
- \*  $a^n!$

### Recursives

- \* Every CSL
- {  $\langle x_1, x_2 \rangle \mid x_1, x_2 \text{ are}$
- regular expression
- $L(x_1) = L(x_2)$  }

### RELS.

- \* Every CSL
- \* Every Recursive
- \* Set of all regular languages.

### Closure Properties for Recursive Languages

1. $L_1 \cup L_2$	✓. Subset ( $L$ )	17. Finite U
2. $L_1 \cap L_2$	10. Prefix ( $L$ )	18. Finite n
3. $\bar{L}$	11. Suffix ( $L$ )	19. Finite Diff
4. $L_1 - L_2$	12. Substring ( $L$ )	20. Finite Concatenation
5. $L_1 \cdot L_2$	✓. $f(L)$	21. Finite Subset
6. $L^{\text{Rev}}$	✓. $h(L)$	22. $f_{\text{finite}}(L)$
7. $L^*$	15. $\epsilon$ -free $h(L)$	✗. Infinite ( $U, n, -, \cdot, c.f.$ )
8. $L^+$	16. $h^{-1}(L)$	

### Closure Properties for RE Languages

1. $L_1 \cup L_2$	✗. Subset ( $L$ )	17. Finite U
2. $L_1 \cap L_2$	10. Prefix ( $L$ )	18. Finite n
✗. $\bar{L}$	11. Suffix ( $L$ )	✗. Finite diff
✗. $L_1 - L_2$	12. Substring ( $L$ )	20. Finite Concatenation
5. $L_1 \cdot L_2$	13. $f(L)$	21. Finite Subset
6. $L^{\text{Rev}}$	14. $h(L)$	✗. Infinite ( $U, n, -, \cdot, c.f.$ )
7. $L^*$	15. $\epsilon$ -free $h(L)$	
8. $L^+$	16. $h^{-1}(L)$	

✗. Infinite ( $U, n, -, \cdot, c.f.$ )

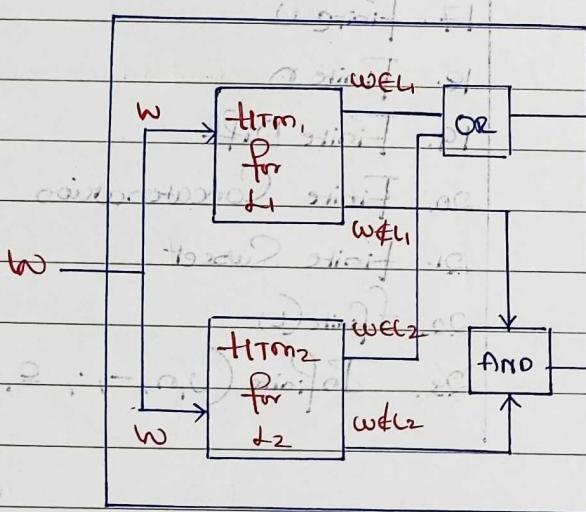
Note: \* 'Subset' are not closed for Regular languages, DCFLs, CFs, CSLs, Recursives, RELs.

\*  $h^{-1}$  and finite subset are closed for regular, DCFLs, CFs, Recursives, RELs.

1. Regulars  $\rightarrow \Sigma, \text{Infinite } (\cup, \cap, -, \circ, \subseteq, f)$
2. DCFLs  $\rightarrow L, \text{prefix, } \Sigma^*, \text{ finite subset}$
3. CFLs  $\rightarrow \cap, L, =, \subseteq, /, \text{ Fd, Tm}$
4. Recursive  $\rightarrow \subseteq, f, h, \text{Tof } (\cup, \cap, -, \circ, \subseteq, f)$
5. REs  $\rightarrow L, -, \subseteq, \text{Finite diff, Tof}(a)$

### 1. Union

Closed for recursive languages (decidable):  $\text{Rec}_1 \cup \text{Rec}_2 = \text{Rec}$   
 Closed for RE languages (semi-decidable):  $\text{REL}_1 \cup \text{REL}_2 = \text{REL}$



$w \in (L_1 \cup L_2)$

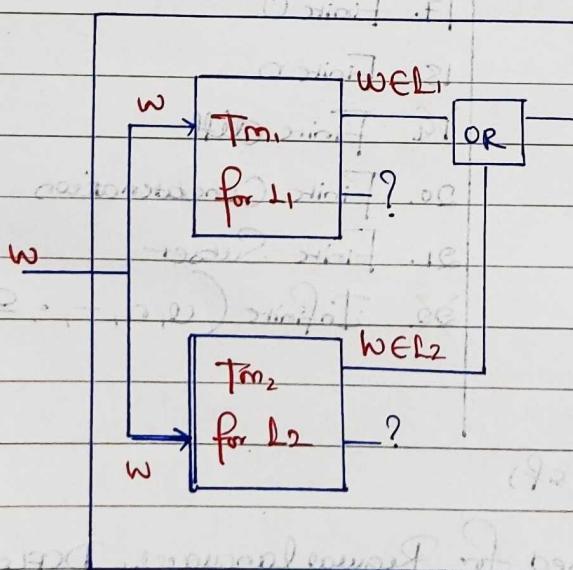
If any turing halt at final then new

turing halt at final.

$w \notin (L_1 \cup L_2)$

If both turing halt at non-final  
then new turing halt at non-final.

Tm for  $L_1 \cup L_2$



$w \in (L_1 \cup L_2)$

If any Tm halt at final then new  
Tm halt at final.

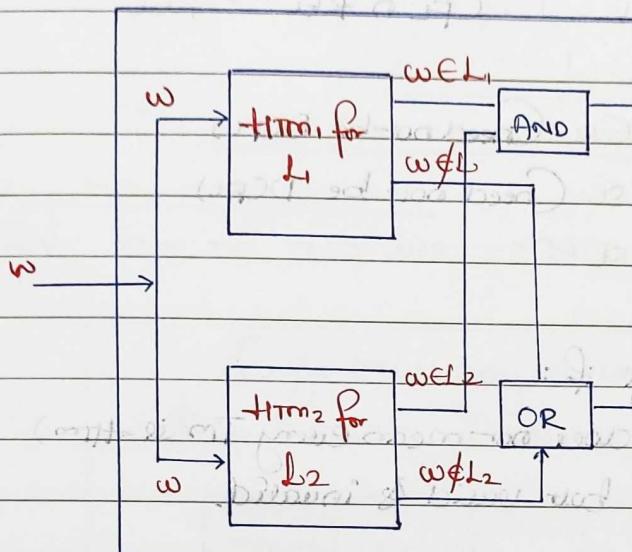
Tm for  $L_1 \cup L_2$

Q2.

Intersection:

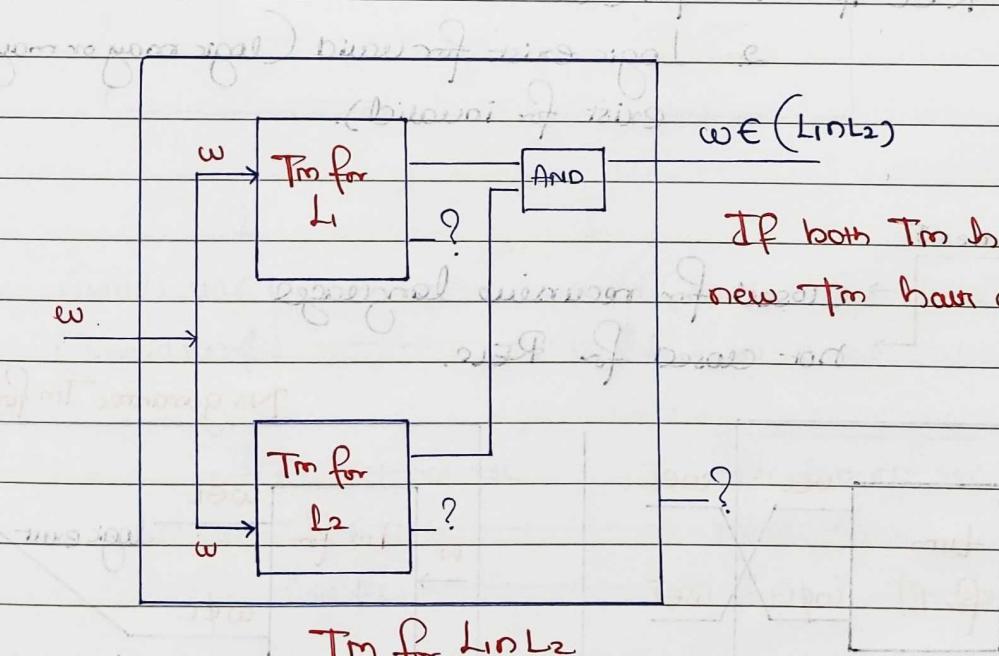
→ Closed for Recursive languages:  $\text{Rec}_1 \cap \text{Rec}_2 = \text{Rec}$

→ Closed for RE languages:  $\text{REL}_1 \cap \text{REL}_2 = \text{REL}$



$w \in L_1 \cap L_2$

If both Tm have at final then new Tm have at final.



$w \in (L_1 \cup L_2)$

If both Tm have at final then new Tm have at final

Q1.  $A_1, A_2$  are Recursive

$B_1, B_2$  are REs.

1.  $A_1 \cup A_2$  is Recursive

2.  $B_1 \cup B_2$  is RE

3.  $A_1 \cup B_1$  is RE

4.  $A_1 \cap A_2$  is Recursive

5.  $B_1 \cap B_2$  is RE

6.  $A_1 \cap B_1$  is RE

Note:  $\text{Reg} \cap \text{DCFL} \rightarrow \text{DCFL}$

$\text{Reg} \cap \text{CFL} \rightarrow \text{CFL}$

$\text{DCFL} \cap \text{CFL} \rightarrow \text{CSL}$

$\text{CFL}_1 \cap \text{CFL}_2 \rightarrow \text{CSL}$

$\text{CFL} \cap \text{CSL} \rightarrow \text{CSL}$

$\text{CSL} \cap \text{Recursion} \rightarrow \text{Recursion}$

$\text{Rec} \cap \text{REL} \rightarrow \text{REL}$

$\text{CFL} \cap \text{REL} \rightarrow \text{REL}$

\*  $\text{DCFL}_1 \cap \text{DCFL}_2 \rightarrow \text{CSL}$  (need not be DCFL)

\*  $\text{CFL}_1 \cap \text{CFL}_2 \rightarrow \text{CSL}$  (need not be CFL)

\*  $\text{DCFL}_1 \cap \text{CFL} \Rightarrow \text{CSL}$

→ L is recursive language if:

1. Tm exist (it does not mean every Tm is Tm)

2. Logic exist for both valid & invalid.

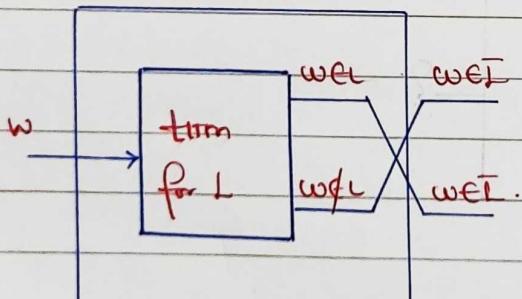
→ L is REL if: 1. Tm exist.

2. Logic exist for valid (logic may or may not exist for invalid).

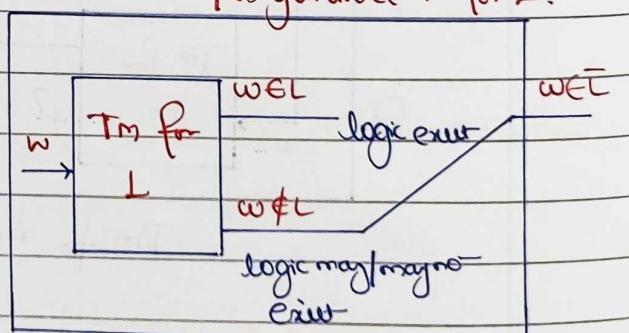
### ③ Complement

→ Closed for recursive languages  
not closed for RELs.

No guarantee Tm for L.



Tm for L.



I. Recursive Language → Recursive Language

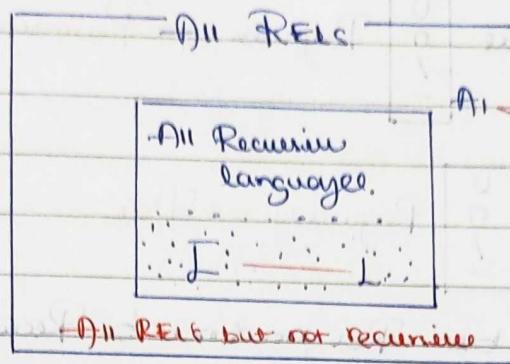
Never be "REL but not rec".

II. REL

Need not be REL

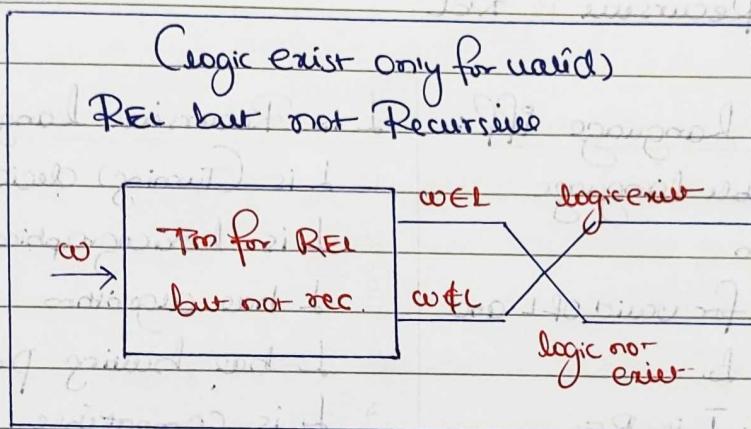
Either recursive lang or "Not REL"

$A_1$  is REL.



(ii) REL but not recursive

III. REL but not recursive  $\rightarrow$  NOT REL.



L

J

(valid) wEL  $\leftrightarrow$  wEL (invalid of L)

(invalid) wFL  $\leftrightarrow$  wFL (valid of J)

REL but not rec

Rec but not rec

✓ wEL  $\leftrightarrow$

wFL ✓

✗ wFL  $\leftrightarrow$

wEL ✗ Tm not exist.

Recursive

Recursive

✓ wEL  $\leftrightarrow$

wFL ✓

✓ wFL  $\leftrightarrow$

wEL ✓

REL

REL

✓ wEL  $\leftrightarrow$

wFL ✓

(don't know) ? wFL  $\leftrightarrow$  wEL (Don't know) ?

Tm may/maynot exist.

Note: I. Recursive       $\boxed{\begin{matrix} U \\ n \\ / \end{matrix}}$  Regular  $\rightarrow$  Recursive

II. REl       $\boxed{\begin{matrix} U \\ n \\ / \end{matrix}}$  Regular  $\rightarrow$  REl

- \* Recursive is REl but REl need not be Recursive.
- \* Decidable is REl but REl need not be decidable.
- \* Recursive  $\neq$  Decidable
- \* Recursive is REl

$\rightarrow$  L is Recursive language iff:

1. L is decidable language
2. L has TTM
3. Logic exist for valid of L and invalid of L.
4. L is REl & L is RE
5. L has TM & L has Tm.

L is Recursive language if:

1. L is (Turing) decidable
2. L is Lexicographically Enumerable
3. L has algorithm
4. L has halting program
5. L is compatible.

$\rightarrow$  L is REl iff:

1. L has TM
2. Logic exist for valid members of L
3. L is semi-decidable
4. L is enumerable
5. L has program
6. L is turing recognisable
7. L is turing acceptable
8. L is Recognizable (acceptable)
9. L has Unrestricted grammar.

L is REl but not Recursive if:

1. L has TM but not recursive
2. L has TM but L has no TTM
3. L has TM but L has no TM
4. L is semi-decidable & undecidable
5. Only valid of L has logic.
6. L is REl & L is not REl

L is not REl iff:

1. L is not semi-decidable
2. L has no TM
3. Valid strings of L has no logic

L is undecidable iff:

1. L is not decidable,
2. L is not recursive & no TTM.

# DECIDABILITY

Undecidable Languages

Decidable Languages	REc but not recursive (SDUDs)	Not REc.
REc (SOS)		

	D	SDUD	NR
Members	valid $\rightarrow$ logic exist invalid $\rightarrow$ logic exist	valid $\rightarrow$ logic exist invalid $\rightarrow$ logic not exist	valid $\rightarrow$ logic not exist
Machine	Hm exist	Thm exist but Hm not exist	Thm not exist
Language	Decidable language.	SDUD (REc but not rec)	NOT REc (not so)
Program (Computer)	Halting program exit (Algorithm exist)	Program exist but no algorithm	Program not exist

## Decision Properties Table (Decision Problems)

Problems	FA, regular lang. Reg. Exp. R.L.	DPA, DCFLs	PDA, CFs	Thm Reqd. lang.	Thm Reqd. lang.
1. Halting	D	D	D	D	UD
2. Membership	D	D	D	D	UD
3. Emptiness	D	D	D	UD	UD
4. Finiteness	D	D	D	UD	UD
5. Totality	D	D	UD	UD	UD
6. Equivalence	D	D	UD	UD	UD
7. Disjoint	D	UD	UD	UD	UD
8. Set Containment	D	UD	UD	UD	UD

### 1. Having Problem

Is  $m$  have on  $w$ ?

Or

" $m$  have on  $w$ ".

### Non-Having Problem.

Is  $m$  doesn't have on  $w$ ?

Or

" $m$  doesn't have on  $w$ ".

### 2. Membership

Is  $m$  accept string  $w$ ?

Or Is  $w$  accepted by  $m$ ?

Or Is  $w \in L(m)$ ?

Or "Is  $m$  accept  $w$ ".

### Non membership

Is  $m$  doesn't accept  $w$ ?

Or Is  $w$  not accepted by  $m$ ?

Or Is  $w \notin L(m)$ ?

Or "Is  $m$  doesn't accept  $w$ ".

### 3. Emptiness

Is  $m$  accepts  $\emptyset$ ?

Or Is  $m$  accept nothing?

Or Is  $L(m) = \emptyset$ ?

### Non Emptiness

Is  $m$  not accepts  $\emptyset$ ?

Or Is  $m$  accept something?

Or Is  $L(m) \neq \emptyset$ ?

### 4. Finiteness

Is  $m$  accepts finite language?

Or  $L(m) = \text{finite language}$ ?

### Non-Finiteness

Is  $L(m) \neq \text{finite language}$ ?

Is  $L(m) = \text{Infinite language}$ ?

### 5. Totality

Is  $m$  accepts everything?

Or Is  $L(m) = \Sigma^*$ ?

### Non Totality

Is  $m$  not accepts everything?

Is  $m$  not accepting atleast one string?

Or Is  $L(m) \neq \Sigma^*$ ?

### 6. Equivalence

Is  $m_1 \equiv m_2$ ?

Or Is  $L(m_1) = L(m_2)$ ?

### Non Equivalence

Is  $m_1 \neq m_2$ ?

Or Is  $L(m_1) \neq L(m_2)$ ?

### 7. Disjointness

Is  $L(m_1) \cap L(m_2) = \emptyset$ ?

### Non Disjointness

Is  $L(m_1) \cap L(m_2) \neq \emptyset$ ?

8.

Set ContainmentIs  $L(m_1) \subseteq L(m_2)$ ?Is  $L_1 \subseteq L_2$ ?Non Set ContainmentIs  $L_1 \neq L_2$ ?Halting Problem for Tm.

Is  $Tm$  halt on  $\epsilon$ ?

Undecidable      SDUD

Not Halting: Is  $Tm$  doesn't halt on  $w$ ?

Undecidable      NOT RE

Yes: logic not exist  
No

Membership for FA/Regl/RegExp/RG

FA → Is  $w \in L(FA)$ ?      w →

(logic exist)

Yes:  $w \in L(FA)$ : FA has at final  
No:  $w \notin L(FA)$ : FA has at non-final (logic exist)

Similarly, membership problem is decidable for FA/DPA/PDA/LB/tm

Membership for Tm: → SDUD

Tm → Is  $Tm$  accept  $w$ ?      w →

(logic exist)

Yes:  $w \in L(Tm)$ , given Tm has at final.  
No:  $w \notin L(Tm)$ , given Tm either has at non-final or never halts

Non membership for Tm

NOT RE

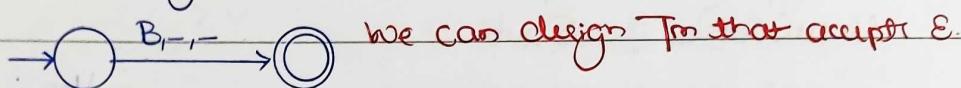
(logic not exist)

Problems:

1. Is FA accepts  $\epsilon$ ? → Membership for FA → Decidable
2. Is PDA accepts  $\epsilon$ ? → D
3. Is Tm accepts  $\epsilon$ ? → SDUD

4. Is  $FA_1 \approx FA_2$ ?  $\rightarrow$  SDUD  $\rightarrow$  Some String  $w$ ,  $PT$  (Logic exist)
5. Is CFG Ambiguous?  $\rightarrow$  Yes: CFG is ambiguous  
No: CFG is unambiguous  
 $\rightarrow$  Every string,  $PT$  (Logic not exist)
6. Is grammar CFG?  $\rightarrow$  Decidable
7. Is CFG unambiguous  $\rightarrow$  Not RE

(I) Can we design Tm that accepts  $E$ ?



We can design Tm that accepts  $E$ .

(II). Can we verify arbitrary Tm, that accepts  $E$ ?  $\rightarrow$  Yes: logic ✓  
 $\downarrow$  Undecidable & Semi-decidable  $\rightarrow$  No: logic ✗

### Language:

1.  $\{ FA \mid FA \text{ accepts } E \} \rightarrow D$
2.  $\{ \langle FA, w \rangle \mid FA \text{ accepts } w \} \} \rightarrow$  Decidable
3.  $\{ FA \# w \mid FA \text{ accepts } w \}$
4.  $\{ Tm \mid Tm \text{ accepts } E \} \} \rightarrow$  SDUD
5.  $\{ Tm \mid Tm \text{ accepts } w \}$
6.  $\{ Tm \mid Tm \text{ accepts some string} \} \rightarrow$  SDUD
7.  $\{ Tm \mid Tm \text{ accepts only } w \} \rightarrow$  Yes: Tm is accepting only  $w$ .  
 $\downarrow$  SSUD  $\rightarrow$  No:  $L(Tm) \neq \{w\}$  Tm nor only accepts  $w$ .
8.  $\{ Tm \mid Tm \text{ accepts 3 length strings} \} \} \rightarrow$  SDUD
9.  $\{ Tm \mid Tm \text{ reaches state } q \} \} \rightarrow$  SDUD
10.  $\{ Tm \mid Tm \text{ reaches state } q \text{ within } 5 \text{ steps} \} \rightarrow$  Decidability.
11.  $L = a^*b^*$
12. Finite Language
13. Regular language  $\rightarrow$  Decidable
14. DCFL
15. CFL
16. CSL
17. Recursive language  $\rightarrow D$
18. REI  $\rightarrow D$  or SDUD
19. REI but not Recursive  $\rightarrow$  SDUD
20. Not REI  $\rightarrow$  NR
21. Not Recursive  $\rightarrow$  NR or SDUD
22. Non-Recursive & Language  $\rightarrow D$  or SDUD or NR

23. Not CFL  $\rightarrow$  D / SDUO / NR
24.  $a^n b^n \rightarrow$  D
25. Set of (all) Finite languages } SDUO
26. Set of (all) regular languages }
27. Set of DCFLs, CFLs, CSLs, Decidable languages, REs, SDUOs } SDUO
28. Set of not REs }
29. Set of infinite languages } NR
30.  $\{ L | L \text{ is decidable} \} \cup \{ L | L \in \text{NR} \}$

Reducibility:  $A \leq B$

$A$  is reducible to  $B$

\* Every instance of  $A$  is mapped into some instance of  $B$

\*  $B$  is equal or harder than  $A$

\*  $B$  is atleast as hard as  $A$

$\rightarrow A \leq_p B$  ( $A$  is polynomially reducible to  $B$ )

$\rightarrow A \leq_m B$  ( $A$  is many-one reducible to  $B$ )

Let  $A \leq B$ :

1. If  $A$  is undecidable then  $B$  is undecidable.
2. If  $B$  is decidable then  $A$  is decidable.
3. If  $B$  is undecidable then  $A$  is decidable/undecidable.
4. If  $A$  is decidable then  $B$  is decidable/undecidable.
5. If  $B$  is RE then  $A$  is RE.
6. If  $A$  is not RE then  $B$  is not RE.
7. If  $A$  takes 100 days then  $B$  takes min 100 days.
8. If  $B$  takes 100 days then  $A$  takes max 100 days.

If  $A \leq B$  and  $C \leq B$ :

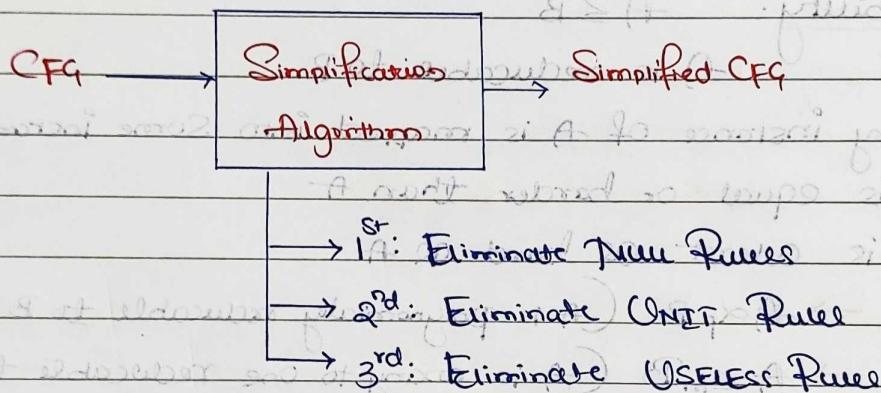
1. If  $A$  is decidable then  $B$  is ? and  $C$  is ?
2. If  $B$  is decidable then  $A$  is decidable and  $C$  is decidable.
3. If  $C$  is decidable then  $A$  is ? and  $B$  is ?

$\rightarrow$  If  $A \leq B$  and  $B \leq A$  then  $A = B$

Let  $A \leq B \leq C$  then:

1. If  $C$  is decidable then  $A$  is decidable and  $B$  is decidable.
  2. If  $A$  is decidable then  $B$  is undecidable and  $C$  is undecidable.
- $L$  is SDUD  $\rightarrow \bar{L}$  is NR
- $L$  is Set of all SDUD  $\rightarrow \bar{L} = \text{Set of all decidable} \cup \text{Set of all NRs}$

### Simplifications of CFG



eg:  $S \rightarrow AB | \epsilon$   
 $A \rightarrow aA | bB | c$   
 $B \rightarrow Ba | \epsilon | bs$

Step 1:  $S \rightarrow \epsilon$

$S \rightarrow AB$

$A \rightarrow aA | bB | c$

$B \rightarrow Ba | \epsilon | bs$

Step 2:  $B \rightarrow \epsilon$

$S \rightarrow AB | A$

$A \rightarrow aA | bB | c | b$

$B \rightarrow Ba | bs | b$

eg 2:  $S \rightarrow Sa | Ab | Bc | AB$   
 $A \rightarrow Ae | \epsilon$   
 $B \rightarrow Bb | \epsilon$

Step 1:  $A \rightarrow \epsilon$  delete

$S \rightarrow Sa | Ab | Bc | AB | b | B$

$A \rightarrow Ae | \epsilon$

$B \rightarrow Bb | \epsilon$

Step 2: Delete  $B \rightarrow \epsilon$

$S \rightarrow Sa | Ab | Bc | AB | b | B$

$A \rightarrow Ae | \epsilon$

$B \rightarrow Bb | b$

Step 3: Delete  $S \rightarrow \epsilon$

$S \rightarrow Sa | Ab | Bc | AB | b | B | C | A | a$

$A \rightarrow Ae | \epsilon$

$B \rightarrow Bb | b$

$X \rightarrow \epsilon$  delete

In whole grammar, add all new production by putting  $X$  with  $\epsilon$

eg3:  $S \rightarrow aN$   
 $A \rightarrow bB \mid aS \mid B$   
 $B \rightarrow aN \mid S$

Step 1: Delete  $S \rightarrow A$   
 $S \rightarrow aN \mid bB \mid aS \mid B$   
 $A \rightarrow bB \mid aS \mid B$   
 $B \rightarrow aN \mid S$

Step 2: Delete  $S \rightarrow B$   
 ~~$S \rightarrow aN \mid bB \mid aS \mid aN \mid S$~~   
 $A \rightarrow bB \mid aS \mid B$   
 $B \rightarrow aN \mid S$

Step 3: Delete  $A \rightarrow R$   
 $S \rightarrow aN \mid bB \mid aS$   
 $A \rightarrow bB \mid aS \mid aN \mid S$   
 $B \rightarrow aN \mid S$

Step 4: Delete  $A \rightarrow S$   
 $S \rightarrow aN \mid bB \mid aS$   
 $A \rightarrow bB \mid aS \mid aN$   
 $B \rightarrow aN \mid S$

Step 5:  
 $S \rightarrow aN \mid bB \mid aS$   
 $A \rightarrow bB \mid aS \mid aN$   
 $B \rightarrow aN \mid bB \mid aS$

\* In given G, if we delete  $x \rightarrow y \rightarrow$  Replace  $y$  using all  $y$  productions.

Eliminate USELESS Productions:

Step 1: Find the nonterminals which derive nothing and eliminate it.

Step 2: Find the unreachable nonterminals and delete them.

$$S \rightarrow aN \mid b \Rightarrow S \rightarrow b$$

$$S \rightarrow aN \mid Bb \mid cde$$

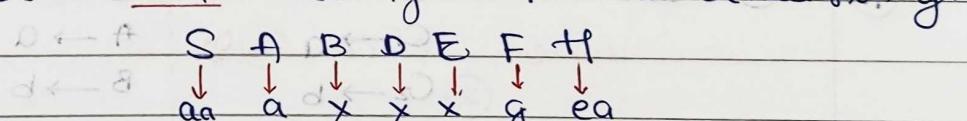
$$A \rightarrow alsa$$

$$B \rightarrow bB$$

$$D \rightarrow bD \mid ee$$

$$F \rightarrow g$$

$$H \rightarrow ea$$



Delete all productions associated with B, D, E.

Step 2: Delete all unreachable symbols

$$S \rightarrow aH$$

$$A \rightarrow a \mid sa$$

CFG  $\rightarrow$

Elimination of null,  
UNIT & USELESS  
Rules

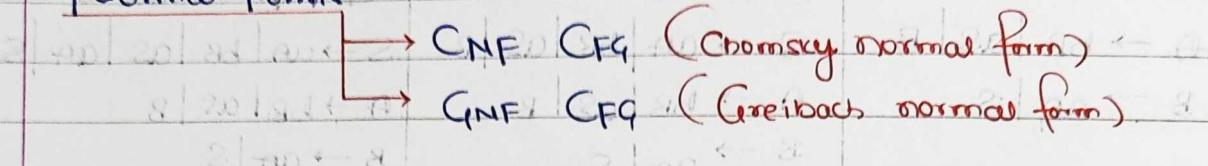
$\rightarrow$  Simplified CFG

CFG  $\rightarrow$

Elimination of  
USELESS Rules

$\rightarrow$  Reduced CFG.

### Normal Form:



CNF CFG	GNF CFG
$V \rightarrow vV_1V_2 \dots V_n$ only 1 terminal exactly 2 non-terminal	$V \rightarrow TV^*$ Example: $S \rightarrow a \mid assSA$ $A \rightarrow bA \mid CAAA SA$
Example: $S \rightarrow SS \mid AS \mid a$ $A \rightarrow ab \mid SA$	

	CNF	GNF
1. $S \rightarrow a$	✓	✓
2. $S \rightarrow ab$	✗	✗
3. $S \rightarrow SS \mid a$	✓	✗
4. $S \rightarrow as \mid bsc \mid c$	✗	✓

### Conversion of CFG to CNF CFG:

① $S \rightarrow ab \rightarrow S \rightarrow G_1 G_2 \rightarrow S \rightarrow AB$ $G_1 \rightarrow a \quad A \rightarrow a$ $G_2 \rightarrow b \quad B \rightarrow b$	③ $S \rightarrow as \mid bca \mid bB$ $A \rightarrow abc \mid e$ $B \rightarrow Aa \mid b$
② $S \rightarrow abc \rightarrow S \rightarrow G_1 G_2 G_3 \rightarrow S \rightarrow G_1 X \rightarrow G_1 G_2 G_3$ $X \rightarrow G_2 G_3$ $G_1 \rightarrow a$ $G_2 \rightarrow b$ $G_3 \rightarrow c$	$S \rightarrow G_1 G_2 G_3$ $X \rightarrow G_2 G_3$ $A \rightarrow G_1 \mid e$ $B \rightarrow G_2 G_3$ $C \rightarrow G_3$

$X \rightarrow \leftarrow$

if in CNF ✓

if not  $\rightarrow$  make every symbol as non-terminal then divide into

2 lengths

CFG  $\rightarrow$  Simplified  $\rightarrow$  CNF CFG

X → [ ]

if terminal → make non terminals

if non terminal, substitute with production

### Conversion of CFG to GNF CFG?

CFG → Simplified CFG → non left Recursive CFG → GNF CFG

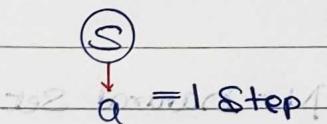
$$1. S \rightarrow ab \rightarrow S \rightarrow aC \\ C \rightarrow b$$

$$2. S \rightarrow abc \rightarrow S \rightarrow aCG_2 \\ C \rightarrow b \\ C_2 \rightarrow c$$

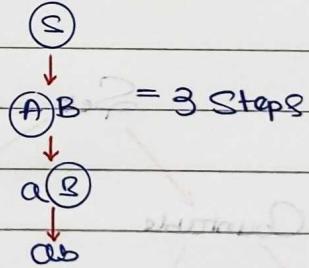
$$3. S \rightarrow f(ab) | baB \Rightarrow S \rightarrow fG_1G_2 | bG_3B \\ A \rightarrow f \\ B \rightarrow gh \\ C_1 \rightarrow a \\ C_2 \rightarrow b \\ C_3 \rightarrow c$$

→ How long does it take the derivation takes place for deriving a length string using CNF CFG?

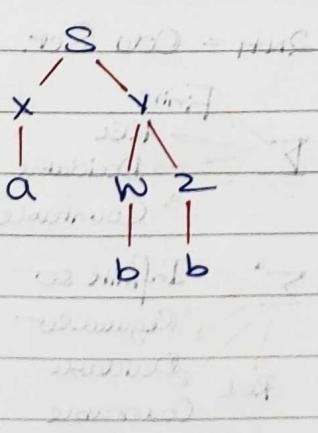
$$1. n=1, w=a$$



$$2. n=2, w=aa$$



$$3. n=3, w=abb$$



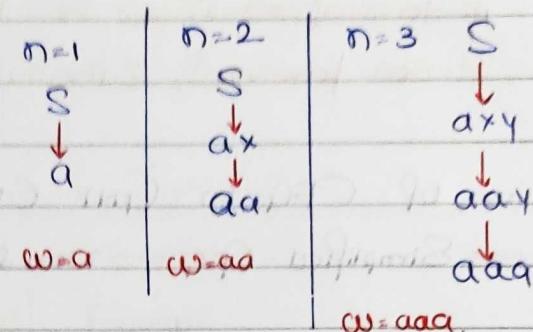
= 5 Steps  
= 5 non leaf nodes  
= 5 Substitutions

\* n length String → (2n-1) steps using CNF CFG

→ How long the derivations take place for deriving n length strings

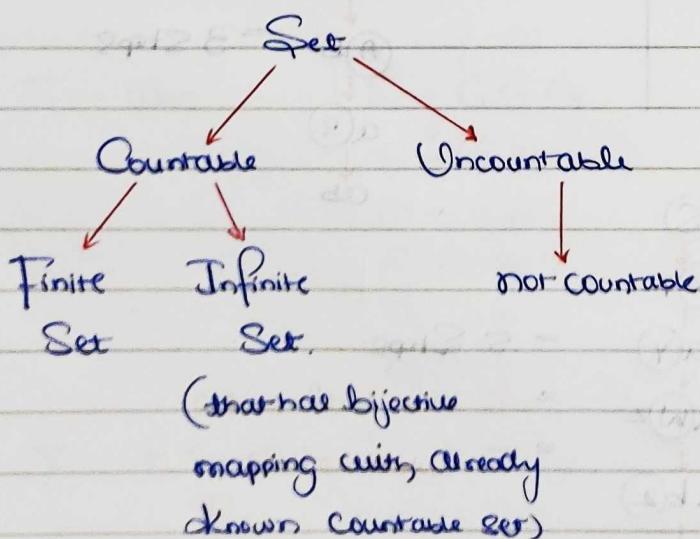
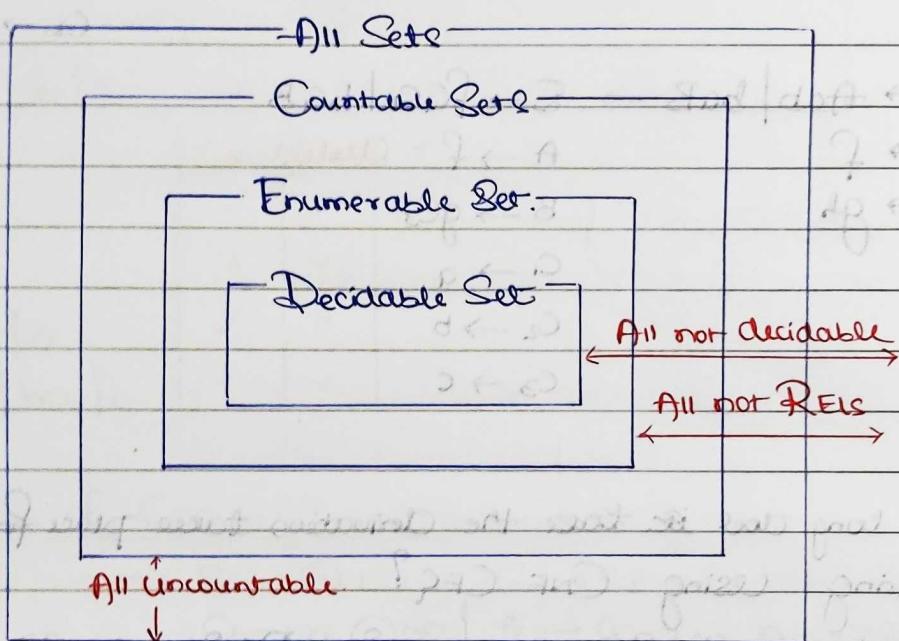
Using GNF CFG?

1.  $n=1 \rightarrow \# \text{Steps} = 1$
2.  $n=2 \rightarrow \# \text{Steps} = 2$
3.  $n=3 \rightarrow \# \text{Steps} = 3$



\* For n length String

'n' steps using GNF CFG

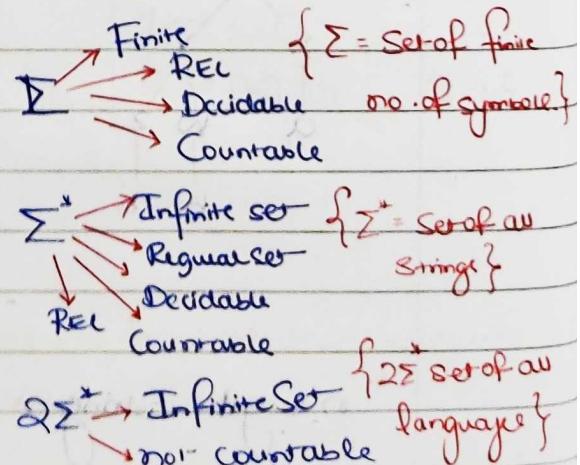


N = Natural Set

Z = Integer Set

2N = Even Set

2N+1 = Odd Set



	Regular Set.	DCFLs	CFLs	Recurs. Lang.	REls
1. $L_1 \cup L_2$	Regular	CFL	CFL	Recursive	REL
2. $L_1 \cap L_2$	Regular	ESL	CSL	Recursive set	REL
3. $\overline{L}$	Regular	DCFL	CSL	Recursive	either Recursive or not REL
4. $L_1 - L_2$	Regular	CSL	CSL	Recursive	Any.

Rice Theorem: Every non-trivial property is undecidable.

non-trivial property  $\leftarrow L = \{m \mid m \text{ is Tm, } m \text{ accepts Regular}\} \rightarrow \text{not REL}$

Trivial  $\leftarrow L = \{Tm \mid Tm \text{ accepts Rel}\}$   
 $= \text{Set of all Tms}$

$L = \{Tm \mid Tm \text{ accepts not REL}\}$   
 $= \emptyset$