

L-1

## FLAT

→ Alphabet - {a, b} {a, b, c} {0, 1} {0, 1, ..., 9} =  $\Sigma$

→ String - Ex: a, b, aa, ab, bc ...

→ If the Alphabet set  $\Sigma$  contains 'n' alphabets i.e. the no. of alphabets in the set are 'n' (denoted by  $|\Sigma|$ ) then the no. of possible strings of length 'n' possible over  $\Sigma$  is  $|\Sigma|^n$ .

→ Language = collection of strings

Let,  $\Sigma = \{a, b\}$  then Language

$L_1 = \{\text{set of all strings of length 2}\} \rightarrow \text{Finite Language}$

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

$L_2 = \{\text{set of all strings of length 3}\} \rightarrow \text{Finite Language}$

$$= \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$$

$L_3 = \{\text{set of all strings where each string start with 'a'}\}$

$$= \{a, aa, aaa, ab, abb, aba, \dots\} \rightarrow \text{Infinite Language}$$

## Powers of $\Sigma$

Let the Alphabet set be  $\Sigma = \{a, b\}$

$\Sigma^1 = \{\text{set of all strings of Length exactly one}\} = \{a, b\}$

$\Sigma^2 = \Sigma \cdot \Sigma = \{\text{set of all strings of Length exactly 2}\} = \{aa, ab, ba, bb\}$

$\Sigma^3 = \Sigma \cdot \Sigma \cdot \Sigma = \{\text{set of all strings of length exactly 3}\} = \{aaa, aab, aba, abb, baa, bab, bba, bbb\} = 8$

$\Sigma^0 = \{\text{set of all strings of length '0'}\} = \{\epsilon\}$  Epsilon = Null string [ $|\epsilon|=0$ ]

$\Sigma^* = \{\Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \dots\} = \{\{\epsilon\} \cup \{a, b\} \cup \{aa, ab, ba, bb\} \dots\}$

= {set of possible strings over (a, b) of all lengths}

= {Universal set}

### PRACTICAL SCENARIO

Now, consider the C language  $\Sigma = \{a, b, \dots, z, A, B, \dots, Z, 0, \dots, 9, +, -, *, /\}$

Now,

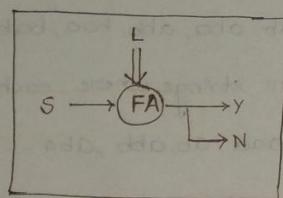
```
void main()
{
    int a,b;
}
```

}      program in 'C' but in TAC it is a string

Now, C-programming language = {set of all "valid" programs}

$$= \{P_1, P_2, P_3, \dots\} \quad (P_n)$$

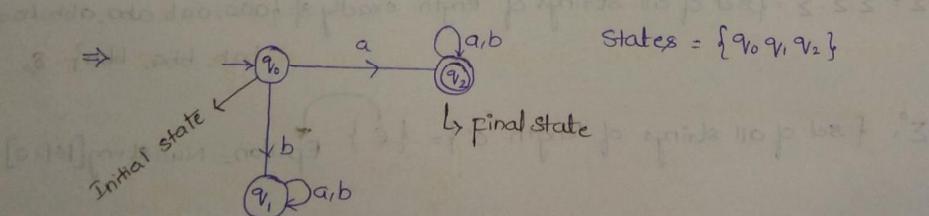
Now, Given any string and to find out whether the string belongs to the Language is a heavy task if the Language is Infinite.



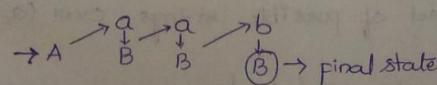
### CONSTRUCTION OF FINITE AUTOMATA

①  $L_1 = \{ \text{set of all strings which start with 'a'} \}$

$$= \{a, aa, ab, aaa, \dots\}$$



Now, Given string = a a b

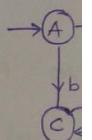


### Classification

FA with  
Moore

### DFA

DFA =  $(Q, \Sigma,$



② construct

Length 2?

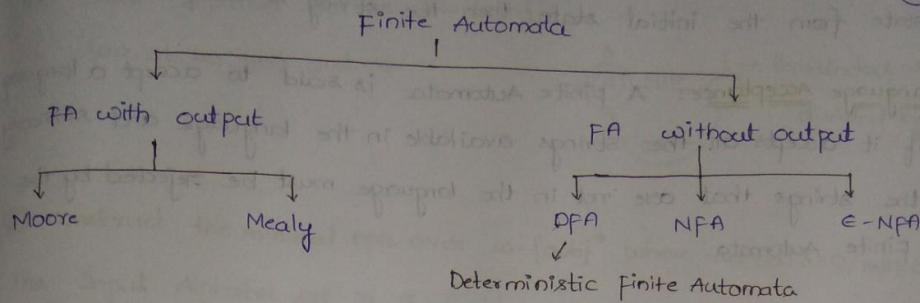
Sol  $\Sigma = \{a, b\}$

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

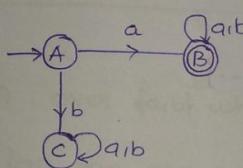
Given string

A

(2) Classification of the Finite Automata.



$$DFA = (Q, \Sigma, \delta, q_0, f)$$



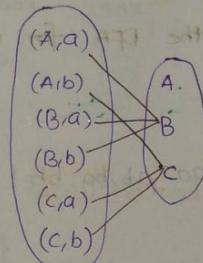
$Q$  = set of all states = {A, B, C}

$\Sigma$  = set of Alphabets = {a, b}

$q_0$  = {A},  $f$  = {B}

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

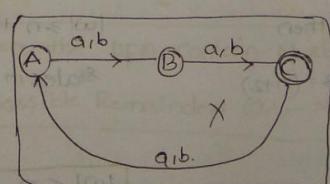
$\{A, B, C\} \times \{a, b\}$



(2) construct a DFA that accepts set of all strings over {a, b} of length 2?  $|w| = 2$ .

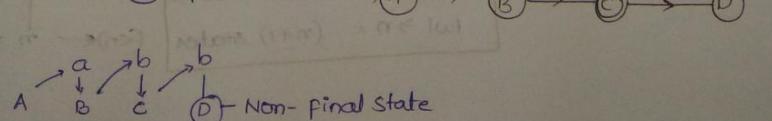
$$\text{sol } \Sigma = \{a, b\}$$

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



Given string

$$A \xrightarrow{a} B \xrightarrow{a} C \xrightarrow{\text{FS}}$$



⇒ String Acceptance: Scan the entire string, if we reach a final state from the initial state then the string is accepted. ④

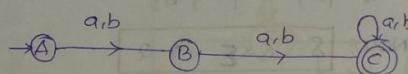
⇒ Language Acceptance: A Finite Automata is said to accept a language if it accepts all the strings available in the language and similarly the strings that are not in the language must be rejected by the Finite Automata.

### L-2 CONSTRUCTION OF MINIMAL DFA

construct a DFA for set of all strings over  $\{a, b\}$  such that length of the string is atleast 2?  $|w| \geq 2$

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

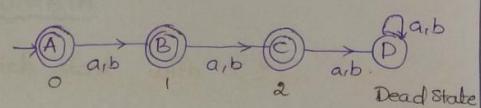
$$L = \{aa, bb, aab, aaa, bbb, baa, \dots\}$$



Now, construct the DFA for which  $|w| \leq 2$

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

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



Now, from the Above problems the conclusions that can be drawn are  
 $|w|=2$        $|w| \geq 2$        $|w| \leq 2$   
 $|w| \leq n$  then no of states in DFA is  $(n+1)$        $|w| \leq n$  the no. of states =  $(n+2)$

If  $|w|=n$  then

$$\text{no. of states} = (n+2)$$

$|w| \geq n$  then no of

states in DFA is  $(n+1)$

$|w| \leq n$  the no. of

states =  $(n+2)$

$ w  \leq n = (n+2) \text{ states}$
$ w  = n = (n+2) \text{ states}$
$ w  \geq n = (n+1) \text{ states}$

$(n+2)$	$(n+1)$
$(<n) \leftarrow n \rightarrow (>n)$	

⑤ Construct the

The Input Alph

$$L = \{aaa, ab, ba\}$$

⑥ Construct the

The Input Alph

$$L = \{aaa, bbb, \dots\}$$

If

⑦ CM DFA over

$$L = \{aaa, bbb\}$$

The Input Alph

For  $n_a(w) \geq 2$

⑧ CM DFA where

$$L = \{aaa, aab, \dots\}$$

The Input Alph

If  $a \in L$   
and each

a final

④

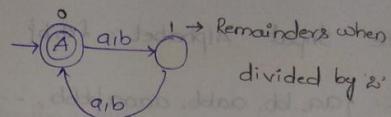
t a language  
d similarly  
ed by the

that length

⑤ construct the minimal DFA over  $\{a, b\}$  where  $|w| \bmod 2 = 0$ .

The Input Alphabet set is  $\Sigma = \{a, b\}$

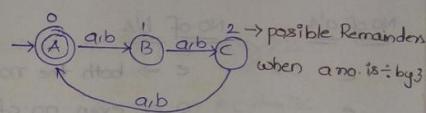
$L = \{aa, ab, ba, bb\}$  and ' $\epsilon$ '



⑥ construct the minimal DFA over  $w = \{a, b\}^*$  where  $|w| \bmod 3 = 0 / |w| \leq 0 \bmod 3$

The Input Alphabet set is  $\Sigma = \{a, b\}$

$L = \{aaa, bbb, abb, baa, aba, bab, \dots\}$

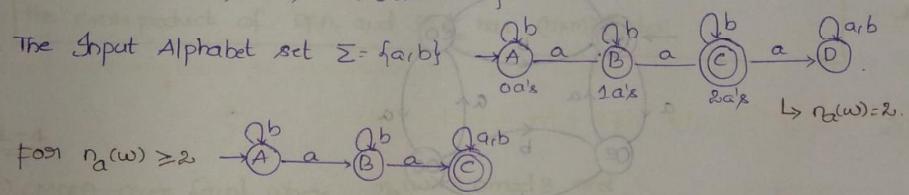


If  $|w| \bmod n = 0$  then the DFA has  $n$  states.

⑦ CM DFA over  $\{a, b\}$  where no. of 'a's in  $w = 2 \Rightarrow \{\eta_a(w) = 2\}$

$L = \{aa, baa, aba, aab, aabbb, \dots\}$

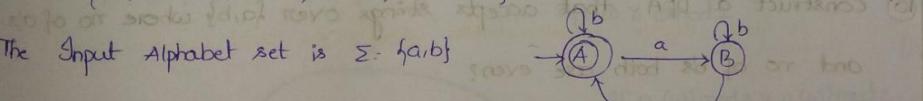
The Input Alphabet set  $\Sigma = \{a, b\}$



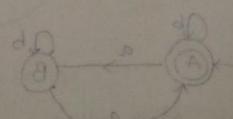
⑧ CM DFA where  $\eta_a(w) \bmod 2 = 0$  or  $\eta_a(w) \equiv 0 \pmod 2$ .

$L = \{aa, aaaab, baa, aab, aba, \dots\}$

The Input Alphabet set is  $\Sigma = \{a, b\}$



If  $|w| \bmod n \equiv K \pmod N$  then the DFA contain  $N$  states  
and each state represent the possible remainders ( $0, 1, \dots, N-1$ )



Q. CM DFA over  $\{a, b\}$  where  $n_a(w) \equiv 0 \pmod{2}$  and  $n_b(w) \equiv 0 \pmod{2}$

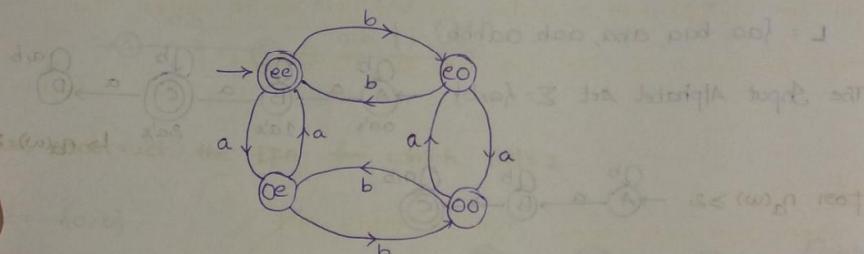
The Input Alphabet =  $\{a, b\}$

$$L = \{aa, bb, aabb, aaaa, bbbb, \dots \text{ and } \epsilon\}$$

The entire Group of strings for the above language can be classified into 4 categories. They are

<u>No. of a's</u>	<u>No. of b's</u>	Condition
e	e	$\Rightarrow$ both the no. of a's and no. of b's are even
e	o	even no. of a's and odd no. of b's
o	e	odd " " " even " "
o	o	odd " " " odd " "

∴ Therefore the DFA contains 4 states Representing each category



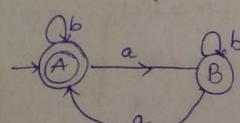
### L-3 CROSS PRODUCT METHOD

Q. construct a DFA that accepts strings over  $\{a, b\}$  where no. of a's and no. of b's both are even?

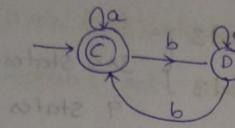
The Input Alphabet  $\Sigma = \{a, b\}$   $w = (a, b)^*$   $n_a(w) \equiv 0 \pmod{2}$

$n_b(w) \equiv 0 \pmod{2}$

Now, construct the DFA for counting a's

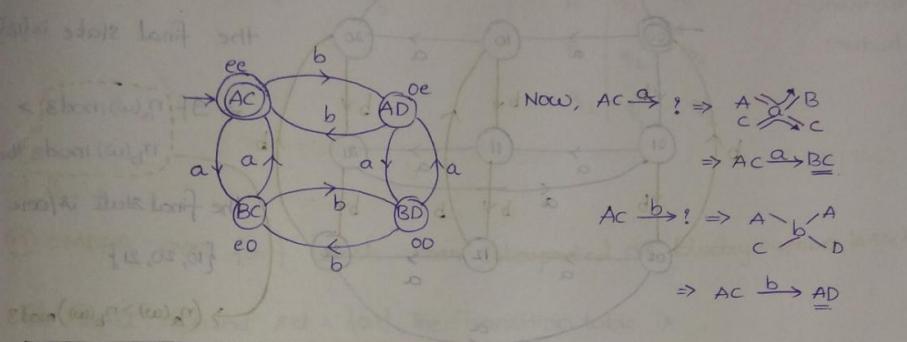


Similarly the DFA for counting the b's will be



Now, the cross product of the two Automaton will be

$$\{A, B\} \times \{C, D\} = \{AC, AD, BC, BD\}$$

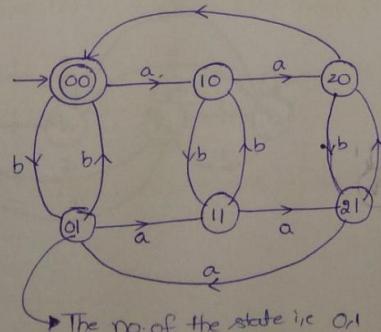


If the 1st DFA has 'n' states and 2nd DFA has 'm' states then  
the cross product of  $DFA_1$  and  $DFA_2$  has  $(nm)$  states.

#### L-4

① CMDFA over  $\{a, b\}$  where  $n_a(\omega) \equiv 0 \pmod{3}$  and  
 $n_b(\omega) \equiv 0 \pmod{2}$ .

The Input Alphabet  $\Sigma = \{a, b\}$ ,  $\omega = (a, b)^*$ ,  $n_a(\omega) \equiv 0 \pmod{3}$  and  
 $n_b(\omega) \equiv 0 \pmod{2}$

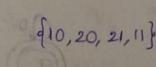


The no. of the state is 6.

Step represent that to reach that state we should cross 0'a's and 1'b's.

If  $n_a(\omega) \pmod{3} \geq n_b(\omega) \pmod{2}$

then final states are



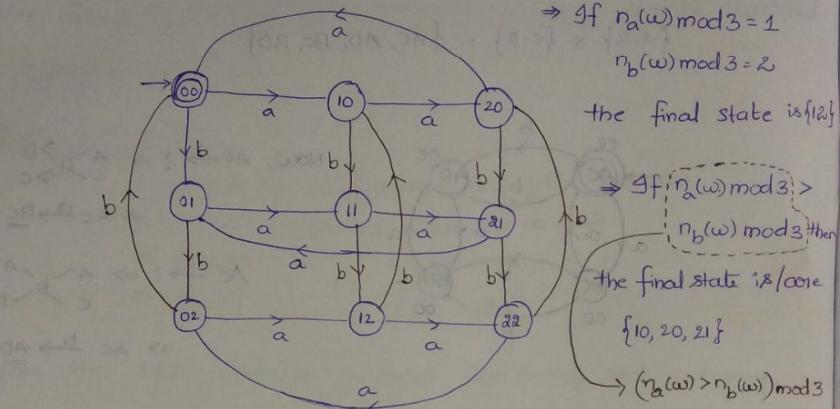
$\{10, 21\}$  = final states.



L-5

- (12) CM DFA over  $\{a, b\}$  where  $n_a(\omega) \equiv 0 \pmod{3}$  and  $n_b(\omega) \equiv 0 \pmod{3}$
- (8)
- $n_a(\omega) \equiv 0 \pmod{3}$  } 3x3 States  
 $n_b(\omega) \equiv 0 \pmod{3}$  } = 9 States

The Input Alphabet set =  $\{a, b\}$



L-6

- (13) CM DFA over  $\{0, 1\}$  which when Interpreted as Binary number is divisible by 3

The Input Alphabet set =  $\{0, 1\}$

1- Unary

Now,  $\omega(10)_2 = (1 \times 2) + 0 = 2$

2- Binary - 0, 1

3- Ternary - 0, 1, 2

4- Quaternary - 0, 1, 2, 3

10- Decimal - 0, 1, ..., 9

16- Hexadecimal - 0, 1, ..., 15

32- Octal - 0, 1, ..., 7

64- Duodecimal - 0, 1, ..., 11

128- Sexagesimal - 0, 1, ..., 11

256- Septenary - 0, 1, ..., 24

512- Hexagesimal - 0, 1, ..., 503

1024- Octonary - 0, 1, ..., 1023

2048- Nonary - 0, 1, ..., 2047

4096- Undenary - 0, 1, ..., 4095

8192- Duotrigesimal - 0, 1, ..., 8191

16384- Trivitrigesimal - 0, 1, ..., 16383

32768- Tetrvitrigesimal - 0, 1, ..., 32767

65536- Pentrvitrigesimal - 0, 1, ..., 65535

131072- Hexrvitrigesimal - 0, 1, ..., 131071

262144- Septrvitrigesimal - 0, 1, ..., 262143

524288- Octrvitrigesimal - 0, 1, ..., 524287

1048576- Nonrvitrigesimal - 0, 1, ..., 1048575

2097152- Undrvitrigesimal - 0, 1, ..., 2097151

4194304- Trivitrigesimal - 0, 1, ..., 4194303

8388608- Tetrvitrigesimal - 0, 1, ..., 8388607

16777216- Pentrvitrigesimal - 0, 1, ..., 16777215

33554432- Septrvitrigesimal - 0, 1, ..., 33554431

67108864- Octrvitrigesimal - 0, 1, ..., 67108863

134217728- Nonrvitrigesimal - 0, 1, ..., 134217727

268435456- Undrvitrigesimal - 0, 1, ..., 268435455

536870912- Trivitrigesimal - 0, 1, ..., 536870911

1073741824- Tetrvitrigesimal - 0, 1, ..., 1073741823

2147483648- Pentrvitrigesimal - 0, 1, ..., 2147483647

4294967296- Septrvitrigesimal - 0, 1, ..., 4294967295

8589934592- Octrvitrigesimal - 0, 1, ..., 8589934591

17179869184- Nonrvitrigesimal - 0, 1, ..., 17179869183

34359738368- Undrvitrigesimal - 0, 1, ..., 34359738367

68719476736- Trivitrigesimal - 0, 1, ..., 68719476735

137438953472- Tetrvitrigesimal - 0, 1, ..., 137438953471

274877906944- Pentrvitrigesimal - 0, 1, ..., 274877906943

549755813888- Septrvitrigesimal - 0, 1, ..., 549755813887

1099511627776- Octrvitrigesimal - 0, 1, ..., 1099511627775

2199023255520- Nonrvitrigesimal - 0, 1, ..., 2199023255519

4398046511040- Undrvitrigesimal - 0, 1, ..., 4398046511039

8796093022080- Trivitrigesimal - 0, 1, ..., 8796093022079

17592186044160- Tetrvitrigesimal - 0, 1, ..., 17592186044159

35184372088320- Pentrvitrigesimal - 0, 1, ..., 35184372088319

70368744176640- Septrvitrigesimal - 0, 1, ..., 70368744176639

140737488353280- Octrvitrigesimal - 0, 1, ..., 140737488353279

281474976706560- Nonrvitrigesimal - 0, 1, ..., 281474976706559

562949953413120- Undrvitrigesimal - 0, 1, ..., 562949953413119

1125899906826240- Trivitrigesimal - 0, 1, ..., 1125899906826239

2251799813652480- Tetrvitrigesimal - 0, 1, ..., 2251799813652479

4503599627304960- Pentrvitrigesimal - 0, 1, ..., 4503599627304959

9007199254609920- Septrvitrigesimal - 0, 1, ..., 9007199254609919

18014398509219840- Octrvitrigesimal - 0, 1, ..., 18014398509219839

36028797018439680- Nonrvitrigesimal - 0, 1, ..., 36028797018439679

72057594036879360- Undrvitrigesimal - 0, 1, ..., 72057594036879359

144115188073758720- Trivitrigesimal - 0, 1, ..., 144115188073758719

288230376147517440- Tetrvitrigesimal - 0, 1, ..., 288230376147517439

576460752295034880- Pentrvitrigesimal - 0, 1, ..., 576460752295034879

1152921504590069760- Septrvitrigesimal - 0, 1, ..., 1152921504590069759

2305843009180139520- Octrvitrigesimal - 0, 1, ..., 2305843009180139519

4611686018360279040- Nonrvitrigesimal - 0, 1, ..., 4611686018360279039

9223372036720558080- Undrvitrigesimal - 0, 1, ..., 9223372036720558079

18446744073441116160- Trivitrigesimal - 0, 1, ..., 18446744073441116159

36893488146882232320- Tetrvitrigesimal - 0, 1, ..., 36893488146882232319

73786976293764464640- Pentrvitrigesimal - 0, 1, ..., 73786976293764464639

147573952587528929280- Septrvitrigesimal - 0, 1, ..., 147573952587528929279

295147905175057858560- Octrvitrigesimal - 0, 1, ..., 295147905175057858559

590295810350115717120- Nonrvitrigesimal - 0, 1, ..., 590295810350115717119

1180591620700231434240- Undrvitrigesimal - 0, 1, ..., 1180591620700231434239

2361183241400462868480- Trivitrigesimal - 0, 1, ..., 2361183241400462868479

4722366482800925736960- Tetrvitrigesimal - 0, 1, ..., 4722366482800925736959

9444732965601851473920- Pentrvitrigesimal - 0, 1, ..., 9444732965601851473919

18889465931203702947840- Septrvitrigesimal - 0, 1, ..., 18889465931203702947839

37778931862407405895680- Octrvitrigesimal - 0, 1, ..., 37778931862407405895679

75557863724814811791360- Nonrvitrigesimal - 0, 1, ..., 75557863724814811791359

151115727449629623582720- Undrvitrigesimal - 0, 1, ..., 151115727449629623582719

302231454899259247165440- Trivitrigesimal - 0, 1, ..., 302231454899259247165439

604462909798518494330880- Tetrvitrigesimal - 0, 1, ..., 604462909798518494330879

1208925819597036988661760- Pentrvitrigesimal - 0, 1, ..., 1208925819597036988661759

2417851639194073977323520- Septrvitrigesimal - 0, 1, ..., 2417851639194073977323519

4835703278388147954647040- Octrvitrigesimal - 0, 1, ..., 4835703278388147954647039

9671406556776295909294080- Nonrvitrigesimal - 0, 1, ..., 9671406556776295909294079

19342813113552591818588160- Undrvitrigesimal - 0, 1, ..., 19342813113552591818588159

38685626227105183637176320- Trivitrigesimal - 0, 1, ..., 38685626227105183637176319

77371252454210367274352640- Tetrvitrigesimal - 0, 1, ..., 77371252454210367274352639

154742504908420734548705280- Pentrvitrigesimal - 0, 1, ..., 154742504908420734548705279

309485009816841469097410560- Septrvitrigesimal - 0, 1, ..., 309485009816841469097410559

618970019633682938194821120- Octrvitrigesimal - 0, 1, ..., 618970019633682938194821119

1237940039267365876389642240- Nonrvitrigesimal - 0, 1, ..., 1237940039267365876389642239

2475880078534731752779284480- Undrvitrigesimal - 0, 1, ..., 2475880078534731752779284479

4951760157069463505558568960- Trivitrigesimal - 0, 1, ..., 4951760157069463505558568959

9903520314138927011117137920- Tetrvitrigesimal - 0, 1, ..., 9903520314138927011117137919

19807040628277854022234275840- Pentrvitrigesimal - 0, 1, ..., 19807040628277854022234275839

39614081256555708044468551680- Septrvitrigesimal - 0, 1, ..., 39614081256555708044468551679

79228162513111416088937103360- Octrvitrigesimal - 0, 1, ..., 79228162513111416088937103359

158456325226222832177874066720- Nonrvitrigesimal - 0, 1, ..., 158456325226222832177874066719

316912650452445664355748133440- Undrvitrigesimal - 0, 1, ..., 316912650452445664355748133439

633825300904891328711496266880- Trivitrigesimal - 0, 1, ..., 633825300904891328711496266879

1267650601809782657422992533760- Tetrvitrigesimal - 0, 1, ..., 1267650601809782657422992533759

2535301203619565314845985067520- Pentrvitrigesimal - 0, 1, ..., 2535301203619565314845985067519

5070602407238130629691970135040- Septrvitrigesimal - 0, 1, ..., 5070602407238130629691970135039

10141204814476261259383940270080- Octrvitrigesimal - 0, 1, ..., 10141204814476261259383940270079

20282409628952522518767880540160- Nonrvitrigesimal - 0, 1, ..., 20282409628952522518767880540159

40564819257905045037535761080320- Undrvitrigesimal - 0, 1, ..., 40564819257905045037535761080319

81129638515810080075071522160640- Trivitrigesimal - 0, 1, ..., 81129638515810080075071522160639

16225927703162016015014304432120- Tetrvitrigesimal - 0, 1, ..., 16225927703162016015014304432119

32451855406324032030028608864240- Pentrvitrigesimal - 0, 1, ..., 32451855406324032030028608864239

64903710812648064060057217728480- Septrvitrigesimal - 0, 1, ..., 64903710812648064060057217728479

129807421625296128120114435456960- Octrvitrigesimal - 0, 1, ..., 129807421625296128120114435456959

259614843250592256240228870913920- Nonrvitrigesimal - 0, 1, ..., 259614843250592256240228870913919

519229686501184512480457741827840- Undrvitrigesimal - 0, 1, ..., 519229686501184512480457741827839

1038459373002369024960915483655680- Trivitrigesimal - 0, 1, ..., 103845937300236902496091548365559

2076918746004738049921830967311360- Tetrvitrigesimal - 0, 1, ..., 207691874600473804992183096731129

4153837492009476099843661934622720- Pentrvitrigesimal - 0, 1, ..., 415383749200947609984366193462269

8307674984018952199687323869245440- Septrvitrigesimal - 0, 1, ..., 830767498401895219968732386924539

1661534976803790439937464773849080- Octrvitrigesimal - 0, 1, ..., 1661534976803790439937464773849039

3323069953607580879874929547698160- Nonrvitrigesimal - 0, 1, ..., 332306995360758087987492954769809

6646139907215161759749859095396320- Undrvitrigesimal - 0, 1, ..., 664613990721516175974985909539629

13292279814430323519497718190792640- Trivitrigesimal - 0, 1, ..., 1329227981443032351949771819079259

2658455

(8)

L-7

(4) CM DFA over  $\{0,1\}$  which interpreted as Binary number is divisible by 3.

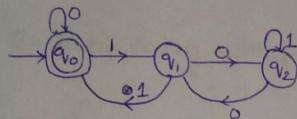
The Input Alphabet set =  $\{0,1\}$

state set

 $\{3=1\}$  $\{3=2\}$ state is  $\{1,2\}$  $\omega \bmod 3 >$  $(\omega) \bmod 3 \text{ then}$ 

state is 8/one

21



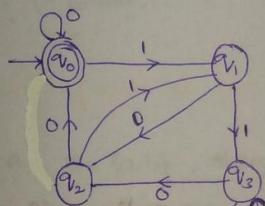
The Transition table is

	0	1
$q_0$	$q_0$	$q_1$
$q_1$	$q_2$	$q_0$
$q_2$	$q_1$	$q_2$

Shortcut  
method

 $>n_b(\omega) \bmod 3$ 

The Input Alphabet set =  $\{0,1\}$ , The Transition table is

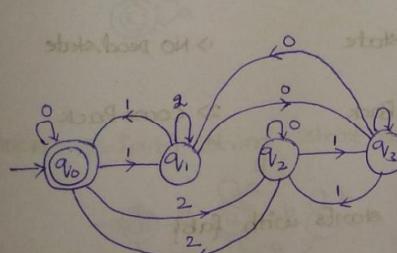


$\Rightarrow$  Shortcut

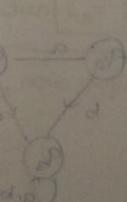
*	0	1
$q_0$	$q_0$	$q_1$
$q_1$	$q_2$	$q_3$
$q_2$	$q_0$	$q_1$
$q_3$	$q_2$	$q_3$

$q_0$  final state

Now if  $\Sigma = \{0,1,2\}$  (Ternary numbers) then, the Transition table is.



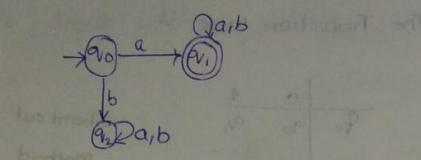
*	0	1	2
$q_0$	$q_0$	$q_1$	$q_2$
$q_1$	$q_3$	$q_0$	$q_1$
$q_2$	$q_2$	$q_3$	$q_0$
$q_3$	$q_1$	$q_2$	$q_3$



L-9

- ⑯ CMDFAs over  $\Sigma = \{a, b\}$  and all the strings start with 'a'?

$$L = \{aa, aab, aaa, \dots\}$$



(10) for Intuition logic see  
AAMS

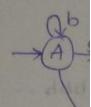
L-14

- ⑰ CMDFAs



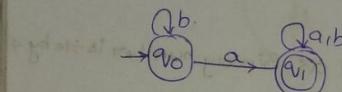
L-15

- ⑱ CMDFAs



L-10

- ⑯ CMDFAs,  $\Sigma = \{a, b\}$  where each string contains 'a'?



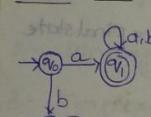
L-11

- ⑰ CMDFAs  $\Sigma = \{a, b\}$  each string ending with 'a'?

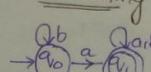


Now, comparing the above 3 DFAs we get

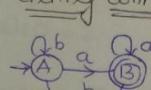
starts with 'a'



containing 'a'



ending with 'a'

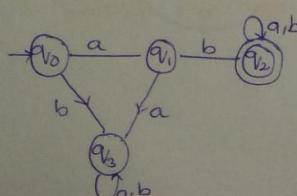


▷ Dead state is present

▷ No come back

L-13

- ⑯ CMDFAs,  $\Sigma = \{a, b\}$  and each string starts with 'ab'?



▷ NO Dead state

▷ No come Back

▷ NO Dead state

▷ Come Back

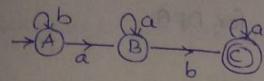
L-18

- ⑰ CMDFAs

$\Sigma =$

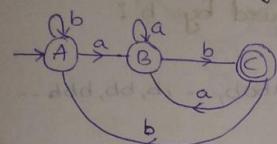
L-14

- ④ CMDFAs  $\Sigma = \{a, b\}$ , Each string containing {ab}



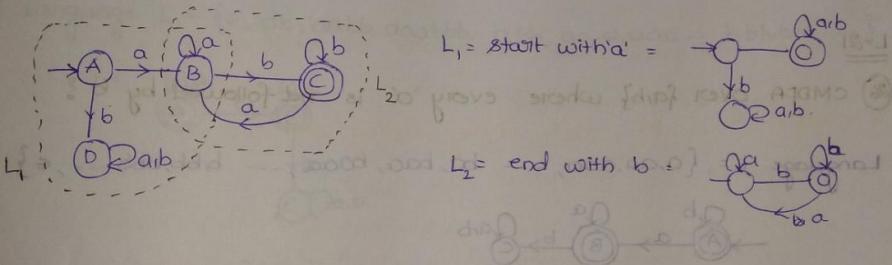
L-15

- ⑤ CMDFAs  $\Sigma = \{a, b\}$  ending with fab



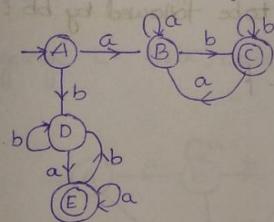
L-16

- ⑥ CMDFAs  $\Sigma = \{a, b\}$  start with 'a' and end with 'b'?



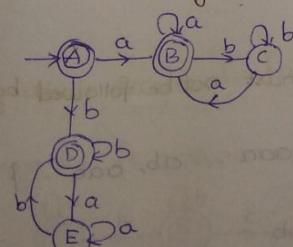
L-17

- ⑦ CMDFAs  $\Sigma = \{a, b\}$ , string start and end with different symbol?



L-18

- ⑧ CMDFAs  $\Sigma = \{a, b\}$  string start and end with same symbol?



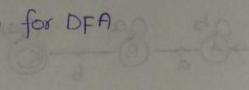
Two Languages are said to be complement to each other when  $L_2 = \Sigma^* - L_1$

where  $L_1, L_2$  are defined on same  $\Sigma$ .

## L-19 COMPLEMENTATION OF DFA

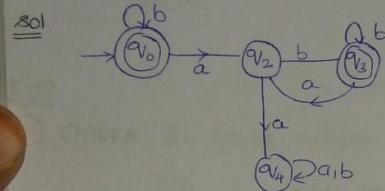
1) The complementation method is Applicable only for DFA.

$$2) (Q, \Sigma, S_0, q_b, f) \xrightarrow{\text{comp}} (Q, \Sigma, S_0, q_b, Q - F)$$



## L-20

25) CMDFA over  $\{a, b\}$  where every 'a' is followed by 'b'?



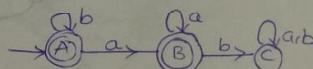
$$L = \{\epsilon, ab, abab, \dots, b, bb, bbb, \dots\}$$



## L-21

26) CMDFA over  $\{a, b\}$  where every 'a' is not followed by 'b'?

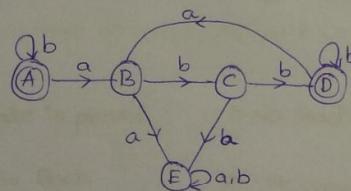
$$\text{Language } L = \{a, aa, aaa, \dots, ba, baa, baaa, \dots, bbb, bbbb, \dots, \epsilon\}$$



## L-22

27) CMDFA over  $\{a, b\}$  where every 'a' has to be followed by 'bb'?

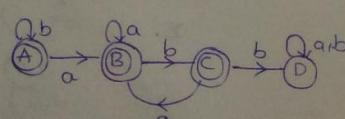
$$\text{Language } L = \{\epsilon, b, bbb, \dots, abb, bbabb, \dots\}$$



## L-23

28) CMDFA over  $\{a, b\}$  where every 'a' must not be followed by 'bb'?

$$\text{Language } L = \{\epsilon, b, bbb, bbb\dots, a, aa, aaa\dots, ab, aab, \dots\}$$



## L-24

29) CMDFA which

The Language  $L =$



## L-25

30) CMDFA which a

The Language  $L =$



## L-26

31) CMDFA which

The Language  $L =$



## L-27

32) CMDFA which

The Language  $L =$

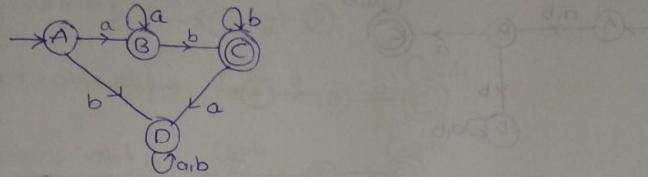


(n)

L-24

⑩ CMDFAs which accept the Language  $L = \{a^n b^m / n, m \geq 1\}$

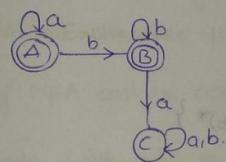
The Language  $L = \{ab, aabb, aaabbb, aaaabbbb, \dots, abb, aabb, aaabb\}$



L-25

⑪ CMDFAs which accept  $L = \{a^n b^m / n, m \geq 0\}$

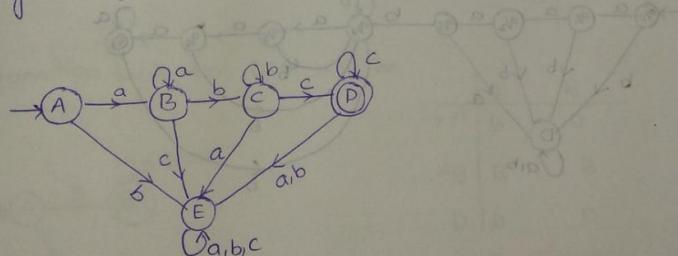
The Language  $L = \{\epsilon, ab, aabb, aaabbb, abb, aa, a, aaa, \dots, bbbb, bbbb, \dots\}$



L-26

⑫ CMDFAs which accept  $L = \{a^n b^m c^l / n, m, l \geq 1\}$

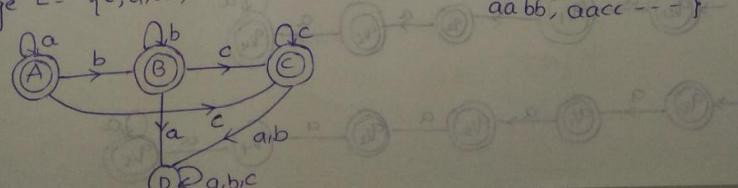
The Language  $L = \{abc, aabc, abbc, abcc, \dots\}$



L-27

⑬ CMDFAs which accept  $L = \{a^n b^m c^l / n, m, l \geq 0\}$

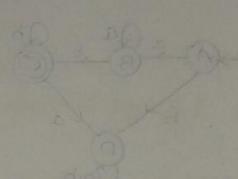
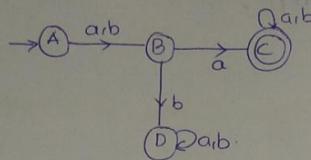
The Language  $L = \{\epsilon, a, aa, aaa, \dots, bbbb, bbbb, \dots, c, cc, ccc, \dots, abc, aabbcc, aabb, aacc, \dots\}$



L-89

- ③ CM DFA over  $\{a, b\}$  such that the 2nd symbol from the LHS is 'a'?

The Language  $L = \{aaa, aa, ba, aab, ba\ldots\}$



If the  $n$ th symbol from the LHS is 'a' then the DFA contains  $(n+2)$  states

L-30

- ④ CM DFA over  $\{a, b\}$  where strings are of the form  $a^3bwa^3$  where  $w$  is any string over  $\{a, b\}$

The Language  $L = \{a^3bwa^3 / w \in \{a, b\}^*\}$

$$L = \{a^3b \in a^3, a^3b \underline{a}^3, a^3b \underline{b} a^3, a^3b \underline{aa} a^3 \ldots\}$$

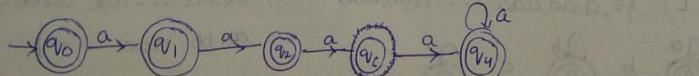
The min. length string =  $aaa\text{baaa}$

L-32

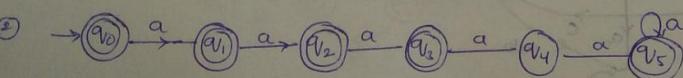
- ⑤ CM DFA over  $\{a\}$  such that  $1) \{a^n / n > 0, n \neq 3\}$

$$2) \{a^n / n > 0, n \neq 4\}$$

①



②



L-34 NFA

$L_1 = \{\text{starts with } a\}$

$L_2 = \{a \text{ containing } b\}$

$L_3 = \{a \text{ ends with } b\}$

$L_4 = \{a \text{ starts with } b\}$

$L_5 = \{a \text{ contains } b\}$

$L_6 = \{a \text{ ends with } b\}$

$L_7 = \{a \text{ starts with } ab\}$

$L_8 = \{a \text{ containing } ab\}$

$L_9 = \{a \text{ ends with } ab\}$

$L_{10} = \{a \text{ starts with } ba\}$

$L_{11} = \{a \text{ containing } ba\}$

$L_{12} = \{a \text{ ends with } ba\}$

$L_{13} = \{a \text{ starts with } aab\}$

$L_{14} = \{a \text{ containing } aab\}$

$L_{15} = \{a \text{ ends with } aab\}$

$L_{16} = \{a \text{ starts with } baa\}$

$L_{17} = \{a \text{ containing } baa\}$

$L_{18} = \{a \text{ ends with } baa\}$

### L-34 NFA

$L_1 = \{ \text{starts with } a \} \Rightarrow \xrightarrow{\quad} A \xrightarrow{a} B$

$L_2 = \{ \text{containing } a \} \Rightarrow \xrightarrow{\quad} A \xrightarrow{a} B$

$L_3 = \{ \text{ends with } a \} \Rightarrow A \xrightarrow{a} B$

$L_4 = \{ \text{start with } ab \} \Rightarrow \xrightarrow{\quad} A \xrightarrow{a} B \xrightarrow{b} C$

$L_5 = \{ \text{contains } ab \} \Rightarrow A \xrightarrow{a} B \xrightarrow{b} C$

$L_6 = \{ \text{End with } ab \} \Rightarrow A \xrightarrow{a} B \xrightarrow{b} C$

### L-35 NFA - DFA CONVERSION

where

The powerlessness (Expressive power) of both NFA and DFA are same because every NFA can be converted to DFA. ( $NFA \cong DFA$ )

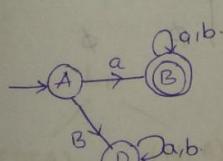
$L_1 = \{ \text{starts with } a \} \quad \Sigma = \{a, b\}$

NFA for  $L_1$  is:  $\xrightarrow{\quad} A \xrightarrow{a} B$

The state Transition table for the above NFA is

	a	b
A	B	$\emptyset$
B	B	B

The state Transition table for DFA will be

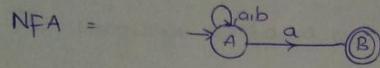


	a	b
A	B	D
B	B	B

The transition function for NFA is

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

Language  $L = \{ \text{ending with } a \}$   $\Sigma = \{a, b\}$

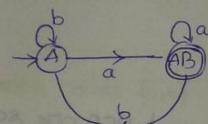


The TT will be

	a	b
A	$\{AB\}, \{A\}$	$\emptyset$
B	$\{\emptyset\}$	$\{B\}$

DFA Transition table will be

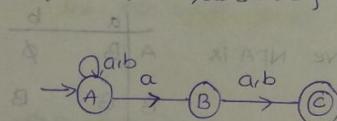
	a	b
$\rightarrow A$	$[AB]$	$[A]$
$[AB]$	$[AB]$	$[A]$
$[B]$	$\emptyset$	$\emptyset$



L-37

$L = \{ \text{all strings where the 2nd symbol form RHS is } a \}$   $\Sigma = \{a, b\}$

NFA =  $L = \{aa, ab, aaa, aab, \dots\}$

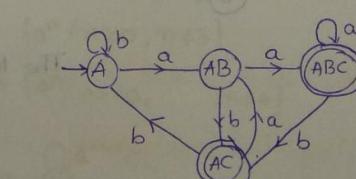


Transition table

	a	b
$\rightarrow A$	$\{AB\}, \{A\}$	$\emptyset$
B	$\{C\}$	$\{C\}$
C	$\emptyset$	$\emptyset$

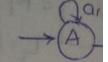
The state Transition table for DFA will be

	a	b
$\rightarrow [A]$	$[AB], [A]$	$\emptyset$
$[AB]$	$[ABC], [AC]$	$\emptyset$
$*[ABC]$	$[ABC], [AC]$	$\emptyset$
$*[AC]$	$[AB], [A]$	$\emptyset$
$[B]$	$[C], [C]$	$\emptyset$



L-38

$L = \{ \text{All strings} \}$

NFA : 

The transition

$\rightarrow [A]$	$[A]$
$[AB]$	$[B]$
$[ABC]$	$[C]$
$[AC]$	$[C]$
$*[ABCD]$	$[D]$
$*[ACD]$	$[D]$
$*[ABD]$	$[D]$
$*[AD]$	$[D]$

L-39

NFA for strings

- a) exactly 2 L
- b) Atleast 2 L
- c) Atmost 2 L

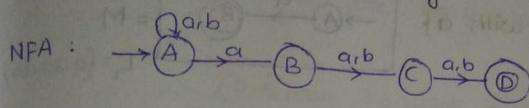
(DFA)

$n(DFA)$

No of states  
States

L-38

$\{ \text{All strings from which 3rd symbol from RHS is } a \}$



	a	b
$\rightarrow A$	{A,B}	{A}
B	{C}	{C}
C	{D}	{D}
*D	{Φ}	{Φ}

The transition table for DFA will be

	a	b
$\rightarrow [A]$	[AB]	[A]
[AB]	[ABC] ✓	[AC] ✓
[ABC]	[ABCD] ✓	[AC] ✓
[AC]	[ABD] ✓	[AD] ✓
*[ABCD]	[ABCD]	[ACD]
*[ACD]	[ABD]	[AD]
*[ABD]	[ABC]	[AC]
*[AD]	[AB]	[A]

Note : If the minimal NFA contains 'n' states then the minimal DFA contains " $2^n$ " states (maximum).

L-39

NFA for strings of length a. exactly 2    b> Atmost 2    c> Atleast 2

a) exactly 2    L = {aa, ab, ba, bb}     $\xrightarrow{\text{a}, \text{b}} \text{A} \xrightarrow{\text{a}, \text{b}} \text{B} \xrightarrow{\text{a}, \text{b}} \text{C}$

b) Atleast 2    L = {aaa, bbb, aa, bb, ba, bb, ...}     $\xrightarrow{\text{a}, \text{b}} \text{A} \xrightarrow{\text{a}, \text{b}} \text{B} \xrightarrow{\text{a}, \text{b}} \text{C} \xrightarrow{\text{q}, \text{a}, \text{b}}$

c) Atmost 2    L = {ε, a, b, ab, ba, bb, aaa}     $\xrightarrow{\text{a}, \text{b}} \text{A} \xrightarrow{\text{a}, \text{b}} \text{B} \xrightarrow{\text{a}, \text{b}} \text{C}$

$L_1$   
 (DFA)      (NFA)

$$n(\text{(DFA)}) \geq n(\text{(NFA)})$$

$$\boxed{\text{No of states in DFA} \geq \text{No of states in NFA}}$$

Note : The minimum no of states present in the finite Automata(NFA) which accepts set of all strings of length 'n' is ' $n$ ' (Only for NFA)

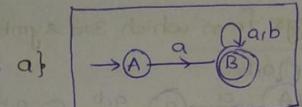
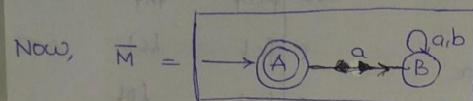
L-40

COMPLEMENTATION OF NFA

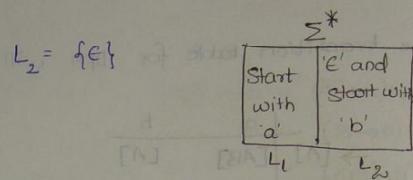
NFA

OVoI  $\{a, b\}$  L = {Starts with a}↓ from start state A to B via a  
↓ from B to final state B via a, b

Now,



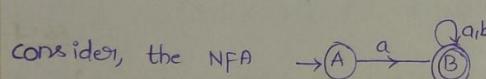
L = {a, aa, ab}

 $L_2 = \{\epsilon\}$ 

Note :

- 1> When we compliment the DFA the language will also get complimented.
- 2> when we compliment the NFA the language may/maynot get complimented.

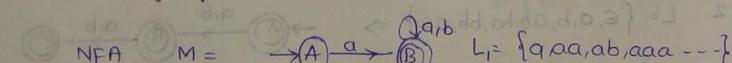
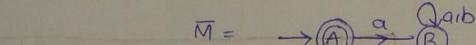
consider, the NFA



① what is the compliment of language accepted by this NFA?

 $L_1 = \{a, aa, ab, aaa, \dots\}$  $L_2 = \{\epsilon, b, bb, ba, bbb, \dots\}$ 

② what is the language accepted by compliment of the NFA given above?

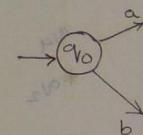
 $M = \text{NFA} \rightarrow \text{DFA}$  $L_1 = \{a, aa, ab, aaa, \dots\}$  $L_2 = \{\epsilon\}$  $(AQN) \cap (AQG)$  $((ANQ)) \cap ((AG)) = \emptyset$  $(AG) \cap (ANQ) = \emptyset$  $[ANQ] \leq [AG] \text{ in states to add}$  $[ANQ] \text{ in states}$ 

L-41

MINIMISATION

Two states 'p' and 'q' in state

PARTITION!



'0' Equivalents

1 Equivalents

2 Equivalents

3 Equivalents

1&gt; check w/

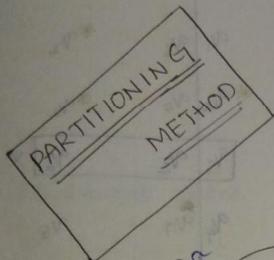
2&gt; If they a

## L-44 MINIMISATION OF DFA

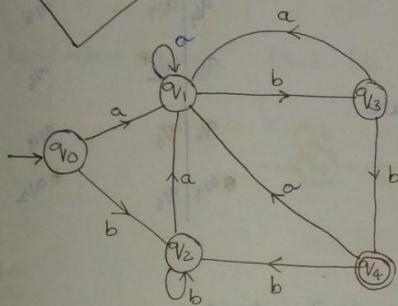
Two states 'P' and 'Q' are Equivalent when

$(P, Q)$  are Equivlant if

$$\begin{aligned} \xrightarrow{\text{Input Alphabet}} \\ \delta(p, \omega) \in F \Rightarrow \delta(q, \omega) \in F \\ \delta(p, \omega) \notin F \Rightarrow \delta(q, \omega) \notin F \end{aligned}$$



If  
 $|w|=0 \Rightarrow '0'$  Equivalent  
 $|w|=1 \Rightarrow '1'$  Equivalent  
 $|w|=n = 'n'$  Equivalent



The Transition table is

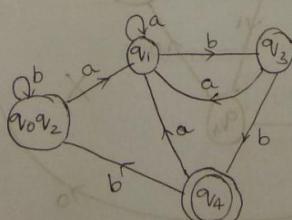
	a	b
$\rightarrow q_0$	$q_1$	$q_2$
$q_1$	$q_1$	$q_3$
$q_2$	$q_1$	$q_2$
$q_3$	$q_1$	$*q_4$
$*q_4$	$q_1$	$q_2$

'0' Equivalent =  $[q_0 \ q_1 \ q_2 \ q_3] \ [q_4]$

'1' Equivalent =  $[q_0 \ q_1 \ q_2] \ [q_3] \ [q_4]$

'2' Equivalent =  $[q_0 \ q_2] \ [q_1] \ [q_3] \ [q_4]$  } Same, so stop here

'3' Equivalent =  $[q_0 \ q_2] \ [q_1] \ [q_3] \ [q_4]$



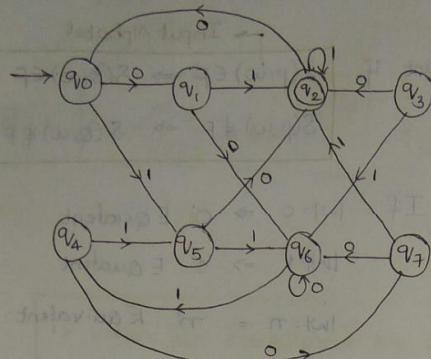
Note: The states that are not reachable from the initial state must be Removed.

1) Check whether they are in same group in previous Equivalence

2) If they are not equal, now check if they are going to same states or not

L-42

Minimise the following DFA



$\Rightarrow q_3$  is not Reachable from Initial state

0 Equivalent states

$[q_0 \ q_1 \ q_4 \ q_5 \ q_6 \ q_7] \ [q_2]$

1 Equivalent states

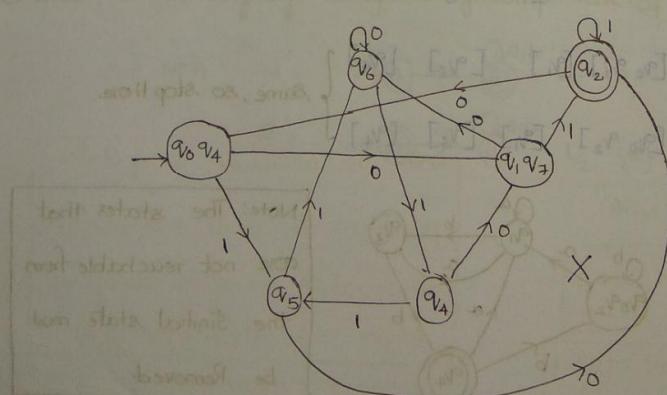
$[q_0 \ q_4 \ q_6] \ [q_1 \ q_7] \ [q_5] \ [q_2]$

2 Equivalent states

$[q_0 \ q_4] \ [q_6] \ [q_1 \ q_7] \ [q_5] \ [q_2]$

3 Equivalent states

$[q_0 \ q_4] \ [q_6] \ [q_1 \ q_7] \ [q_5] \ [q_4] \ [q_2]$



L-43 (GAT) MINIMISATION OF DFA

(26)

Mimise the

The Transition table will be

	0	1
$\rightarrow q_0$	$q_1$	$q_5$
$q_1$	$q_6$	$*q_2$
$*q_2$	$q_0$	$*q_2$
$q_3$	$q_2$	$q_6$
$q_4$	$q_7$	$q_5$
$q_5$	$q_2$	$q_6$
$q_6$	$q_6$	$q_4$
$q_7$	$q_6$	$*q_2$

0 Equivale

1 Equivale

L-44

Minimize the

The Transition

a	
$q_0$	$q_1$
$q_1$	$q_0$
$q_2$	$q_3$
$*q_3$	$*q_3$
$X$	
Not Reachable	

(20)

L-43 (GATE)

Minimise the DFA

table will be

$$\begin{array}{c} 1 \\ q_5 \end{array}$$

$$*q_2$$

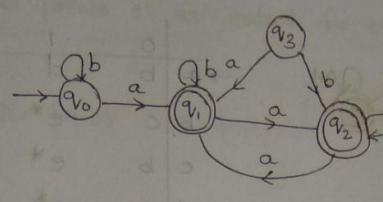
$$*q_2$$

$$q_6 \quad X$$

$$q_5$$

$$q_6$$

$$q_4$$

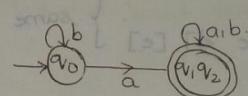
$$*q_2$$


The Transition table is

	a	b
$\rightarrow q_0$	$q_1^*$	$q_0$
$*q_1$	$q_2^*$	$q_1^*$
$*q_2$	$q_1^*$	$q_2^*$

0 Equivalent states

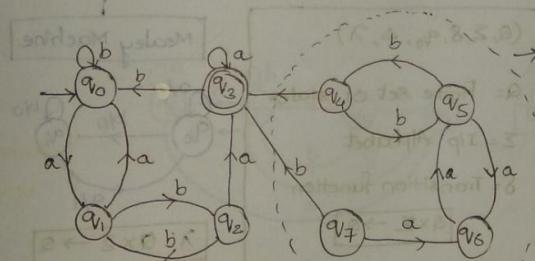
1 Equivalent states

$$\begin{array}{c} [q_0] \quad [q_1, q_2] \\ [q_0] \quad [q_1, q_2] \end{array} \quad \left\{ \begin{array}{l} \text{same, stop here} \\ \text{state instances} \end{array} \right.$$
L-44

Minimise the DFA

same so stop

Here



Not REACHABLE FROM

INITIAL STATE

$$\Delta \leftarrow P : R$$

The Transition table is

	a	b	(a,b)
$q_0$	$q_1$	$q_0$	$(q_1, P)$
$q_1$	$q_0$	$q_2$	
$q_2$	$q_3^*$	$q_1^*$	
$q_3$	$q_3^*$	$q_0$	

0 Equivalent  $[q_0, q_1, q_2]$   $[q_3]$ 1 Equivalent  $[q_0, q_1]$   $[q_2]$   $[q_3]$ 2 Equivalent  $[q_0]$   $[q_1]$   $[q_2]$   $[q_3]$ 

	$q_4$	$q_3$	$q_5$
$q_4$	$q_5$	$q_4$	$q_4$
$q_5$	$q_5$	$q_4$	$q_6$
$q_6$	$q_5$	$q_3$	$q_3$

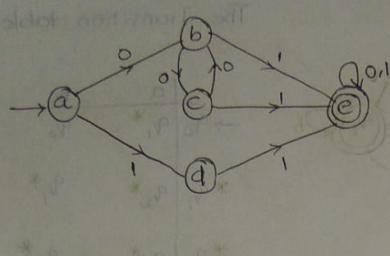
X

Not Reachable

 $q_7$  $q_6$  $q_3$

L-45

Minimize the DFA



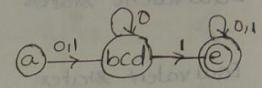
The Transition table

	0	1
0	a b c d	d
b	c	e*
c	b	e*
d	c	e*
e*	e*	e*

0 Equivalent states [a b c d] [e]

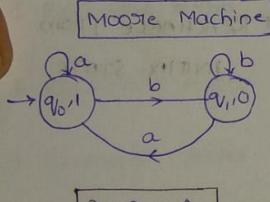
1 Equivalent states [a] [b c d] [c]

2 Equivalent states [a] [b c d] [e] } same

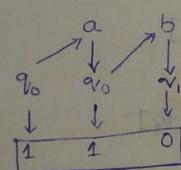


### L-46 MOORE AND MEALEY MACHINES

Finite Automata with output



$$\lambda: Q \rightarrow \Delta$$



If we give 'n' length Input then  
the output will be  $(n+1)$  length  
string

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

$Q$  = Finite set of states

$\Sigma$  = I/p Alphabet

$\delta$  = Transition function

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

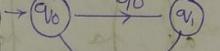
$q_0$  = Initial state

$\Delta$  = O/p Alphabet

$\lambda$  = O/p function.

Mealey Machine

Mealey Machine



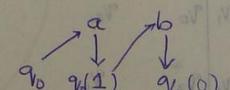
$$\lambda: Q \times \Sigma \rightarrow \Delta$$

$(q_0, a) \rightarrow 1$

$(q_0, b) \rightarrow 0$

$(q_1, a) \rightarrow 1$

$(q_1, b) \rightarrow 0$



$$[1 0]$$

If we give 'n' length Input then  
output will be 'm' length string  
in Mealey Machine

L-47

Construct a moore  
as input and pos.  
substring.

$$\Sigma = \{a\}$$

L-48

Counting the occ.

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

L-49

Construct a moore

and produces a

output if i/p

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

10 - A

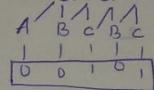
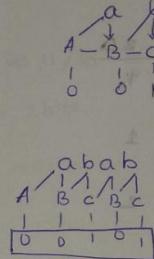
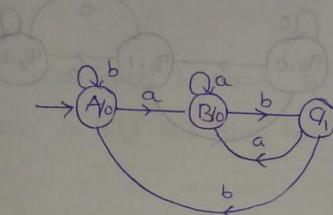
11 - B.

18.

(22)

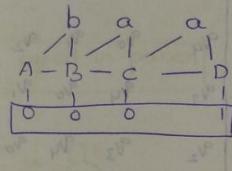
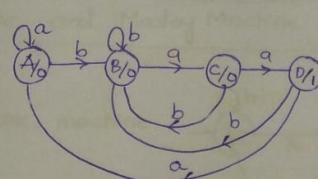
L-47  
 construct a moore machine that takes set of all strings over  $\{a, b\}$  as input and prints '1' as output for every occurrence of 'ab' as substring.

$$\Sigma = \{a, b\} \quad \Delta = \{0, 1\} \quad [\because \text{construct DFA ending with } ab]$$

L-48

counting the occurrence of substring 'bac', construct Moore Machine?

$$\Sigma = \{a, b\} \quad \Delta = \{0, 1\} \quad [\because \text{construct DFA ending with } bac]$$

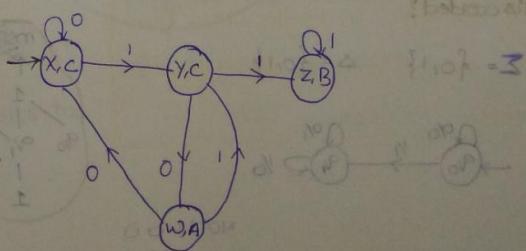
L-49

Construct a moore machine that takes set of all strings over  $\{0, 1\}$  and produces 'A' as output if input ends with '10' or produces 'B' as output if i/p ends with '11' otherwise produces 'C'?

$$\Sigma = \{0, 1\} \quad \Delta = \{A, B, C\}$$

10 - A  
11 - B.

In Input then  
n length string



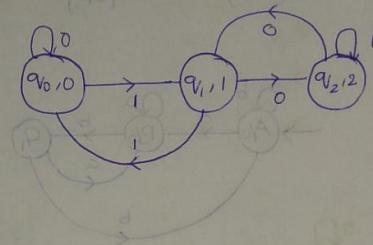
L-50

Moore Machine Residue modulo 'n'

construct a moore machine that takes binary nos as input and produces residue modulo '3' as output.

$$\Sigma = \{0,1\} \quad \Delta = \{0,1,2\}$$

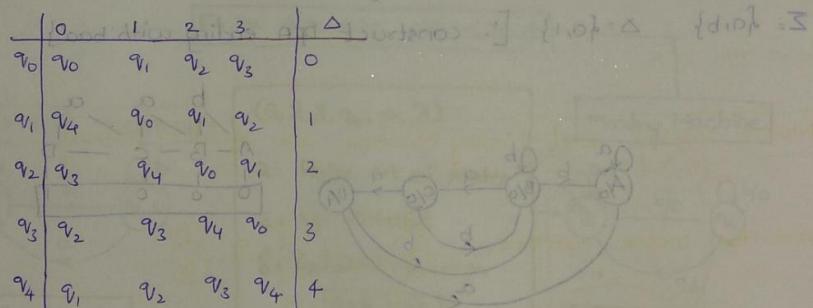
	0	1	$\Delta$
$q_0$	$q_0$	$q_1$	0
$q_1$	$q_2$	$q_0$	1
$q_2$	$q_1$	$q_2$	2



construct a moore machine that takes base 4 nos as Input and produces residue modulo '5' as output?

$$\Sigma = \{0,1\} \quad \Delta = \{0,1,2,3,4\}$$

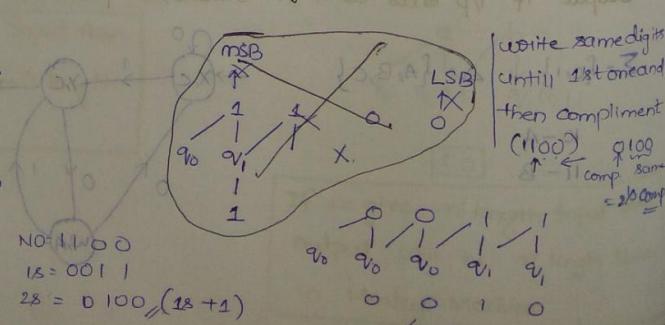
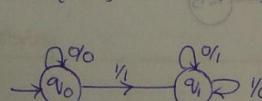
	0	1	2	3	$\Delta$
$q_0$	$q_0$	$q_1$	$q_2$	$q_3$	0
$q_1$	$q_4$	$q_0$	$q_1$	$q_2$	1
$q_2$	$q_3$	$q_4$	$q_0$	$q_1$	2
$q_3$	$q_2$	$q_3$	$q_4$	$q_0$	3
$q_4$	$q_1$	$q_2$	$q_3$	$q_4$	4



### L-51 MEALY MACHINE

construct a mealy machine that takes Binary number as i/p and produces 2's complement of that no. as o/p. Assume the string is Read from Least Significant bit to most significant bit and end coding is discarded?

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



L-52  
What is the

option a, b  
must cont

option c:

L-53 CON

Moore

convert th

Now, in t

(q0, b) is  
output a

Now, q0  
q1, and q1

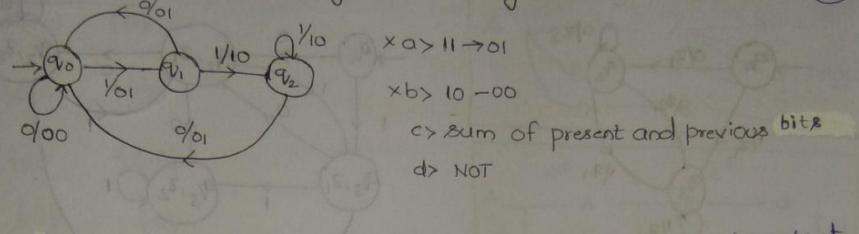
is 1 ∵

→ Similarity

L-52

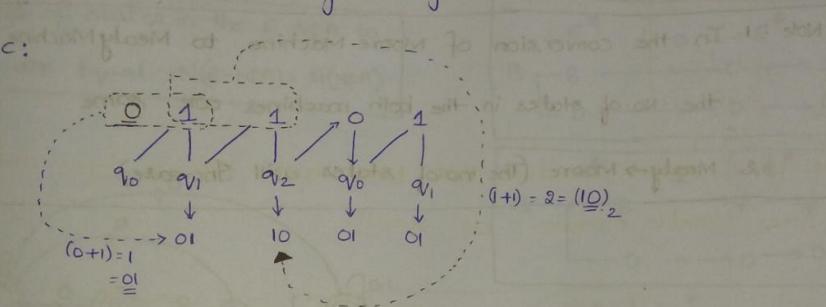
what is the output produced by the following state machine

25



c> sum of present and previous bits  
d> NOT

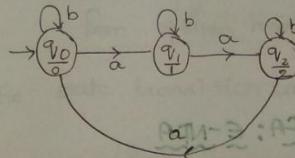
option c:



### L-53 CONVERSION OF MOORE MACHINE TO MEALY MACHINE

Moore Machine and Mealy Machine are equal in power

Convert the moore machine

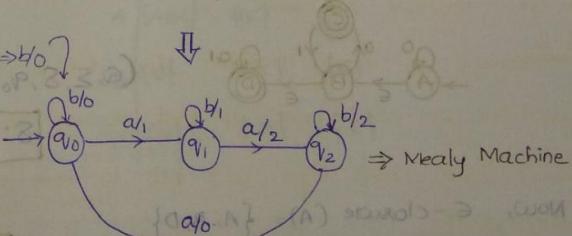


Now, in the moore machine

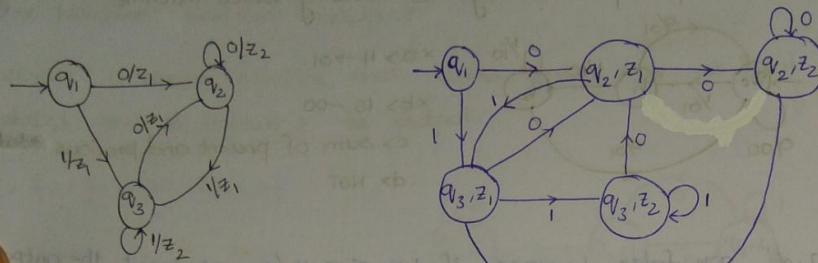
$(q_0, b)$  is going to  $q_0$  and the output associated with  $q_0$  is  $b/0$

Now,  $q_0$  on 'a' is going to  $q_1$ , and o/p associated at  $q_1$  is 1  $\therefore a/1$  (in Mealy machine).

Similarly continue for other states too.



### L-55 CONVERSION OF MEALY MACHINE TO MOORE MACHINE



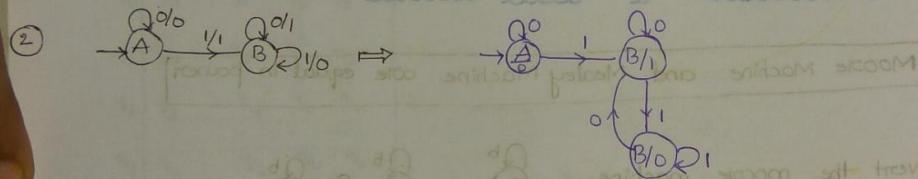
Note : 1. In the conversion of Moore Machine to Mealy Machine

the No. of states in the both machines are same

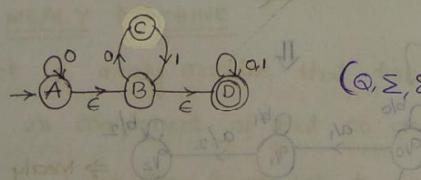
2. Mealy  $\rightarrow$  Moore (The no. of states will Increase)

L-56

### L-56 CONVERSION OF MOORE MACHINE TO MEALY MACHINE



### L-57 EPSILON NFA: $\epsilon$ -NFA



Now,  $\epsilon$ -closure (A) = {A, B, D}

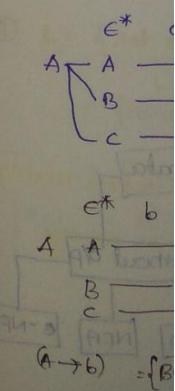
$\epsilon$ -closure (B) = {B, D}

$\epsilon$ -closure (C) = {C}

$\epsilon$ -closure (D) = {D}

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

$\epsilon$ -closure (A) = The states that are  
Reachable from 'A' on seeing  
 $\epsilon$ psilon( $\epsilon$ )



The state tr

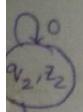
	0	{ABCD}
B.	1	{CD}
C	0	{Φ}
*D	1	{D}

The No. of  
NFA are E

LINE

86

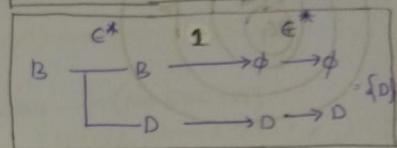
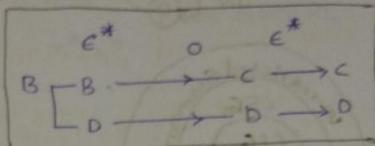
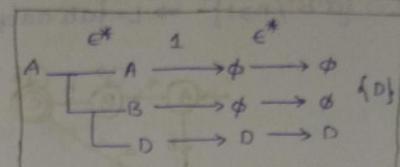
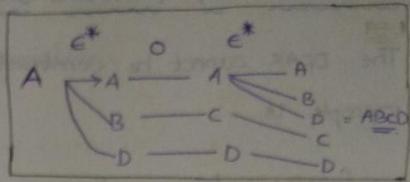
The state transition table



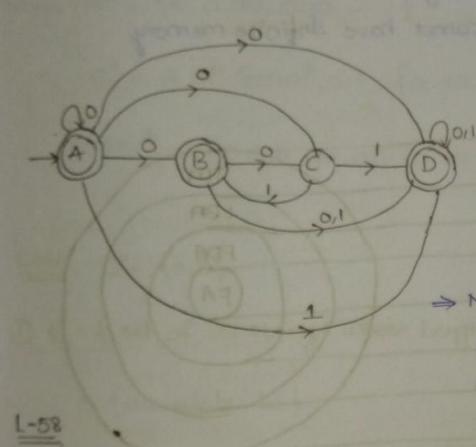
	0	1
A	{ABCD}	{D}

B	{CD}	{D}
C	{Φ}	{B,D}

D	{D}	{D}
*		



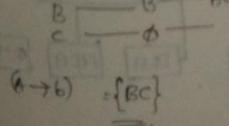
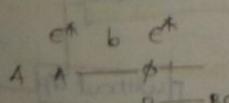
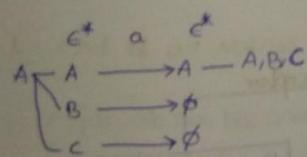
The No. of states in the  $\epsilon$ -NFA to NFA are Equal.  $N(\epsilon\text{-NFA}) = N(\text{NFA})$



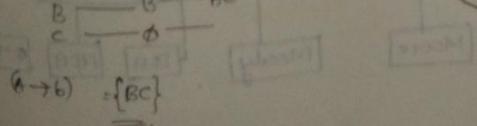
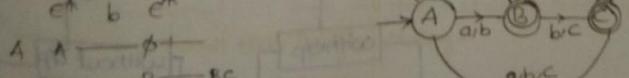
→ No. of final states will increase from  $\epsilon$ -NFA to NFA conversion. The states on which you can reach final states from  $\epsilon$ -transition are also final states.

The state transition table is

	a	b	c
A	{ABC}	{BC}	{C}
B	{Φ}	{BC}	{C}
C	{Φ}	{Φ}	{C}



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

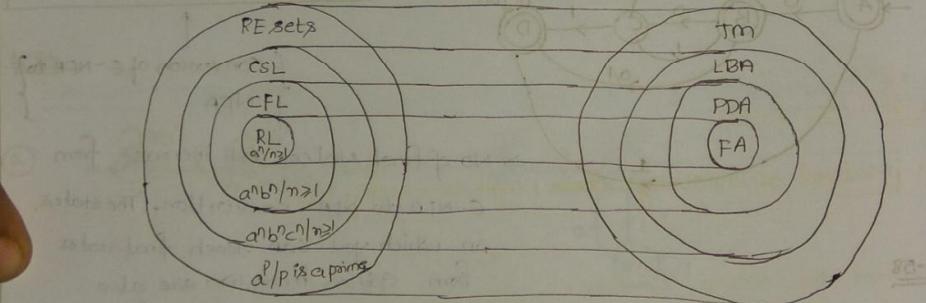
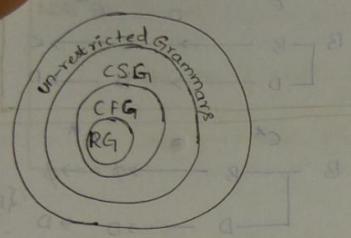
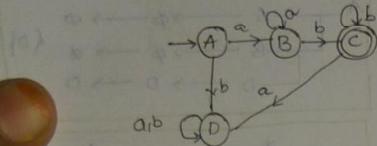


## FAMILIES OF FORMAL LANGUAGES

L-59

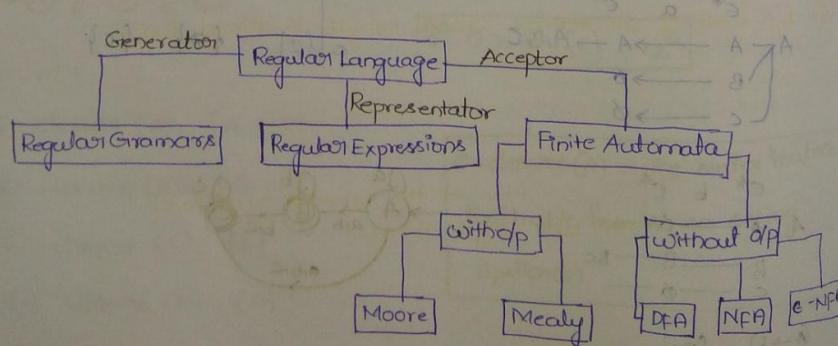
The DFA's cannot be constructed for all the languages and one such example is

$$L = \{a^n b^n / n \geq 1\} \Rightarrow L = \{ab, aabb, aaabbb, \dots\}$$



## REGULAR EXPRESSIONS AND CONVERSIONS

L-60



## Regular Expressions

The Regular Expressions

1) Union (+)

2) Concatenation (.)

3) Kleen closure (\*)

a)  $\emptyset = \{\}$

$\epsilon = \{\epsilon\}$

$a = \{a\}$

Now,  $a^* = \{\epsilon,$

$a\}$

$(a+b)^* =$

$(a+b)^*$

L-61  $\Sigma = \{a,$

①  $L_1 = \{ \text{set of}$

$s + s (d/t)$

$= \{aa, a$

$ab, ba, bb\}$

$= \{aa,$

$ab, ba, bb\}$

$= \{aa,$

$ab, ba, bb\}$

$= \{aa,$

$ab, ba, bb\}$

②  $L_1 = \{ \text{set of}$

$= \{aa, a$

$= \{ (a+$

$*d(a+d)) *d\}$

d one such

at the string  
ed, so we  
nols of 'a' arrived  
f required at  
site. But FA  
nony

### Regular Expressions

The Regular Expression contains three operations

1) Union (+)

2) Concatenation (.)

3) Kleen closure (\*)

$$\begin{aligned} a) \quad \emptyset &= \{ \} \\ \epsilon &= \{ \epsilon \} \\ a &= \{ a \} \end{aligned}$$

→ primitive RE's.

$$\begin{aligned} (b) \quad \gamma_1 + \gamma_2 &= \{ \gamma_1 \cup \gamma_2 \} = \{ \} \\ \gamma_1 \cdot \gamma_2 &= \{ \gamma_1 \cdot \gamma_2 \} = \{ \} \\ \gamma_1^* &= \{ \gamma_1^* \} = \{ \} \end{aligned}$$

are also RE

$$\text{Now, } a^* = \{ \epsilon, a, aa, aaa, \dots \}$$

$$a^+ = a \cdot a^* \quad (or) \quad a^+ \cdot a = \{ a, aa, aaa, \dots \} \quad (\text{No } \epsilon)$$

$$(a+b)^* = \{ \epsilon, a, b, aa, ab, ba, bb, \dots \} = \{ \text{set of all strings possible over } \{a, b\} \}$$

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

$$① L_i = \{ \text{set of all strings whose Length is exactly two} \}$$

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

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

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

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

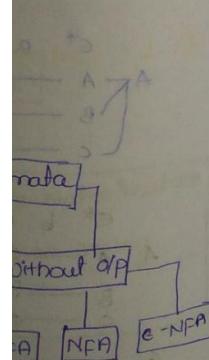
$$② L_i = \{ \text{set of all strings whose Length is atleast two} \}$$

$$= \{ aa, ab, ba, bb, aaa, bbbb, \dots \}$$

$$= \{ (a+b)(a+b)(a+b)^* \} \quad (\text{or}) \quad \{ (a+b)^* (a+b)(a+b) \} \quad \{ (a+b)(a+b)^* (a+b) \}$$

All the strings over {a, b}.

All two Length strings



③  $L_1 = \{ \text{set of all strings whose Length is almost 2} \}$

$$L_1 = \{ \epsilon, a, b, ab, ba, bb, aa \}$$

$$L_1 = \{ \epsilon + a+b+ab+ba+bb+aa \}$$

$$L_1 = \{ (a+b+\epsilon) (a+b+\epsilon) \}$$

④  $L_2 = \{ \text{even Length strings} \}$

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

$$= \underbrace{(a+b)(a+b)}_{\text{Even Length strings repeated any no. of times gives strings of even length.}}^*$$

$$\text{Analysis: } ((a+b)(a+b))^*$$

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

$$= (a+b)^{2*}$$

= even.

⑤  $L_3 = \{ \text{odd Length strings} \}$

$$= \{ \epsilon, aaa, aab, aab, baa, bbb \dots \}$$

$$= \{ ((a+b)(a+b))^* (a+b) \}$$

$$\text{Analysis: } (a+b)^{2n+1}$$

$$= (a+b)^{2n} (a+b)$$

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

⑥  $L_4 = \{ \text{Length is divisible by 3} \}$

$$L_4 = \{ \text{length strings, } 3 \text{ LS, } 6 \text{ LS, } 9 \text{ LS \dots} \}$$

$$= \{ ((a+b)(a+b)(a+b))^* \}$$

$$\text{Analysis: } (a+b)^{3n+2}$$

$$= (a+b)^{3n} (a+b)^2$$

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

⑦  $L_5 = \{ \text{Length } \equiv 2 \pmod{3} \}$

$$= \{ ((a+b)(a+b)(a+b))^* (a+b)(a+b) \}$$

⑧  $L_6 = \{ \text{exactly 2 als} \}$

{atleast 2}

$$= \{ b^* a b^* a b^* \}$$

{atmost 2}

$$= \{ b^* a b^* a (a+b)^* \}$$

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

⑨  $L_7 = \{ \text{at least one even} \}$

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

⑩  $L_8 = \{ \text{starts with 2} \}$

$$= \{ a(a+b)^* \}$$

⑪  $L_9 = \{ \text{start and end} \}$

$$= \{ a(a+b)^* b \}$$

L-62

\*\*\*  $L_1 = \{ \text{no 2 als} \}$

\*\*\*  $= \{ \epsilon, b, bb, bbb, ba, bab, \dots \}$

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

$$= (a+\epsilon)(b+ab)$$

⑬  $L_2 = \{ \text{NO 2 als a} \}$

$$= \{ \epsilon, a, b, ab, \dots \}$$

Starts

fab, aba, ababa \dots \} - a

fab, abab, ababab - fa

{ba, baba \dots \} - b

{b, bab, bbabab \dots \} b

⑨  $L = \{ \text{all code even} \} = \{ aa, aab, bbb, bba, \dots \}$

$= (b^* a b^* a b^*)^* + (b^*)$  The Language/RE with not generate the strings  
 $\{ b, bbb, bba, \dots \}$  so we add "b\*" to generate the strings

⑩  $L = \{ \text{starts with } a \} = \{ a(a+b)^* \}$

$\{ \text{Ends with } a \} = \{ (a+b)^* a \}$

$\{ \text{contains } a \} = \{ (a+b)^* a (a+b)^* \}$

even.

nes gives strings

$$\begin{aligned} & n+1 \\ & (a+b) \\ & (a+b)^* (a+b) \end{aligned}$$

$$\begin{aligned} & (a+b)^{3n+2} \\ & (a+b)^2 \\ & (a+b)(a+b)^*(a+b)(a+b) \end{aligned}$$

$$= \{ a(a+b)^* b \}$$

$$\{ b(a+b)^* a \}$$

$$L = \{ \epsilon, a, b, aa, bbb, aaa, aba, \dots \}$$

$$= \{ a(a+b)^* a + b(a+b)^* b + ab + ba + \dots \}$$

L-62

\*\*\* ⑪  $L = \{ \text{no 2a's should come together} \}$

$$= \{ \epsilon, b, bb, bbb, \dots, a, ab, aba, abab, ababa, \dots \}$$

$$ba, bab, baba, babab, \dots \} = \underbrace{(b+ab)^*}_{\text{set of all strings}} + \underbrace{(bab)^* a}_{\text{ending with } a}$$

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

$$\text{or} \quad \{ \text{set of all strings ending with } b \}$$

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

⑫  $L = \{ \text{NO 2a's and no 2b's come together} \}$

$$= \{ \epsilon, a, b, ab, ba, aba, bab, \dots \}$$

Starts with  
 $\{ a, \underline{ab}, \underline{bab}, \dots \} - a$

Ends with

$$a - (ab)^* a$$

$$b - (ab)^* b$$

$$a - (ba)^* b$$

$$b - (ba)^* b$$

fab, abab, ababab, fa

{ba, baba, \dots} - b

{b, b\underline{ab}, b\underline{bab}, \dots} - b

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

$$\Rightarrow (ab)^* a + (ab)^* + b(ab)^* a + b(ab)^*$$

$$= \underbrace{(ab)^*}_{(ab)^*} (\epsilon+a) + \underbrace{b(ab)^*}_{b(ab)^*} (b+\epsilon)$$

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

## Identities of Regular Expressions V.V.V Imp.

$$① \phi + R = R + \phi = R$$

$$⑥ \epsilon + RR^* = RR^* + \epsilon = R^*$$

$$② \phi \cdot R = R \cdot \phi = \phi$$

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

$$③ \epsilon R = R\epsilon = R$$

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

$$④ \epsilon^* = \epsilon$$

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

$$⑤ \phi^* = \epsilon$$

L-63

## Regular Expression to Finite Automata

$$1) \phi : \rightarrow \circlearrowleft$$

$$2) a : \rightarrow \circlearrowleft \xrightarrow{a} \circlearrowleft$$

$$3) a+b : \rightarrow \circlearrowleft \xrightarrow{a,b} \circlearrowleft$$

$$4) ab : \rightarrow \circlearrowleft \xrightarrow{a} \circlearrowleft \xrightarrow{b} \circlearrowleft$$

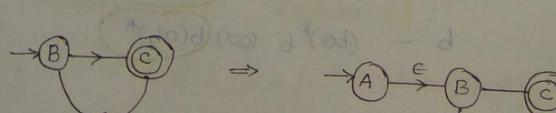
$$1) ab^* : \rightarrow \circlearrowleft \xrightarrow{b} \circlearrowleft^b$$

$$2) (ab)^* : \rightarrow \circlearrowleft \xrightarrow{a} \circlearrowleft \xrightarrow{b} \circlearrowleft$$

## Finite Automata to Regular Expression conversion

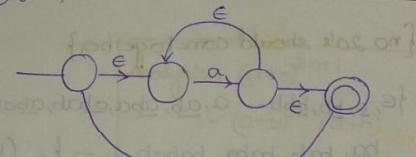
### Rule 1

The initial state should not have any incoming edge

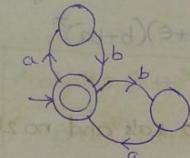


### STATE CREATION METHOD

$$⑤ a^* :$$

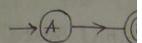


$$③ (ab+ba)^* :$$



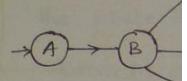
### Rule 2

The final state

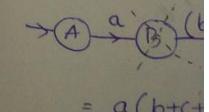
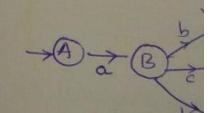
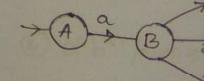
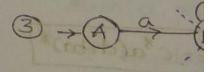
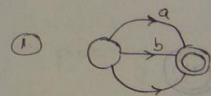


### Rule 3

There must be  
except the



### Examples



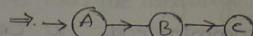
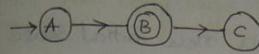
### STATE ELIMINATION METHOD

$$(a+d)^*(cd)d + (a+d)^*(dt) =$$

$$(a+d)(a+d)^*(cd) =$$

Rule - 2

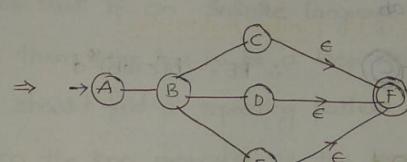
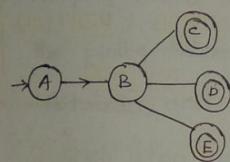
The final state should not have any outgoing edge and there should be only one final state



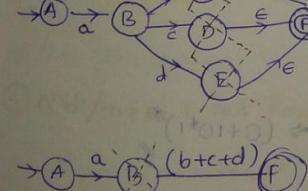
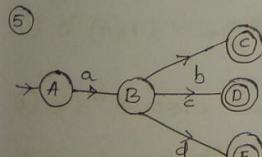
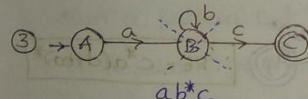
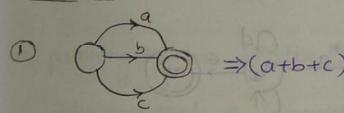
33

Rule - 3

There must be only one final state and now eliminate the other states except the initial and final states in any order you wish.



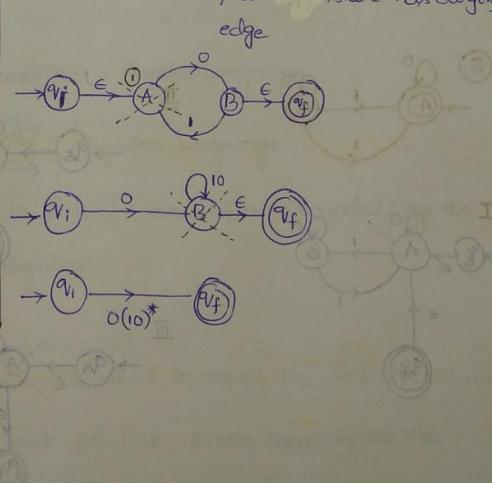
Examples



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

②  $\rightarrow ab$

④ Here the initial state is having incoming edge and final state has outgoing edge



## TESTING

L-64  
If the

$\Rightarrow$  Infinite L

that is c

not is a

$\Rightarrow$  Pumping L

the Finite

Automata

which w

the Lang

be cons

for testing pattern

$\Rightarrow$  pumping L

will say th

such a p

is Regular

sub. whic

I  $a^n / n \geq 1$ :

II  $a^n b^m / n, m$

III  $a^n b^n / n \leq 1$

IV  $a^n b^n / n \geq 1$

V  $a^n b^n / n \geq 1$

VI  $a^n b^n / n \geq 1$

VII  $a^n b^n / n \geq 1$

VIII  $a^n b^n / n \geq 1$

IX  $a^n b^n / n \geq 1$

X  $a^n b^n / n \geq 1$

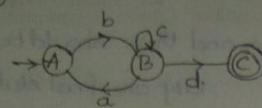
XI  $a^n b^n / n \geq 1$

XII  $a^n b^n / n \geq 1$

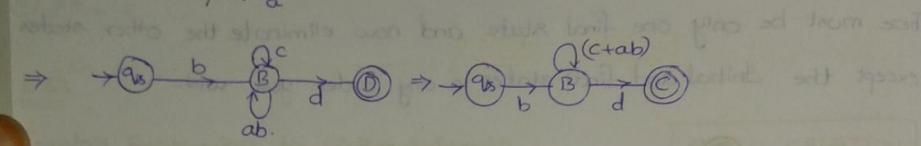
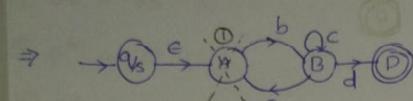
XIII  $a^n b^n / n \geq 1$

XIV  $a^n b^n / n \geq 1$

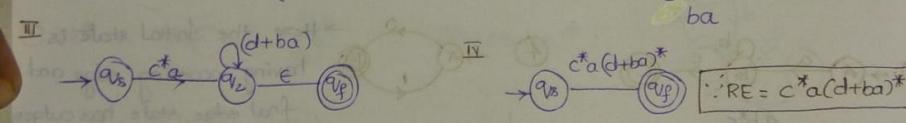
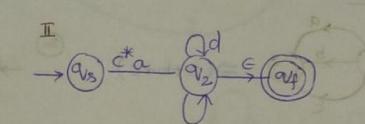
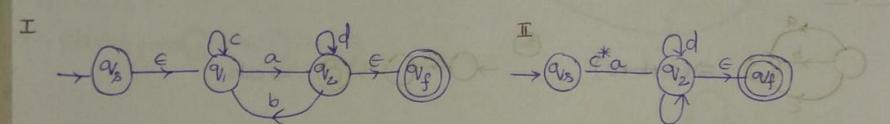
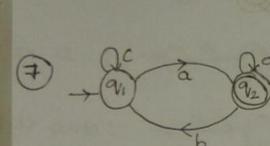
⑥



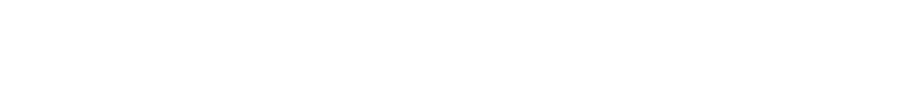
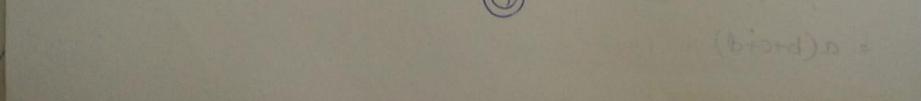
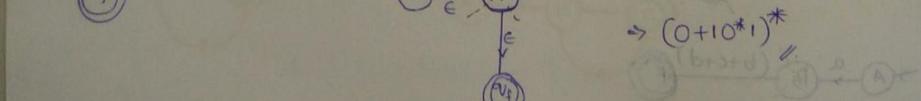
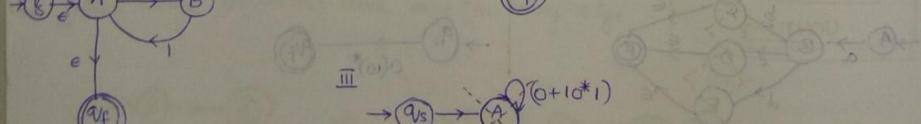
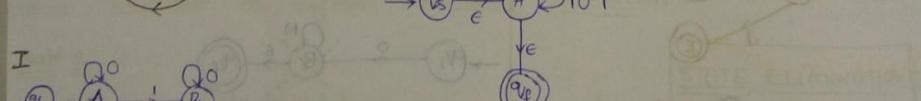
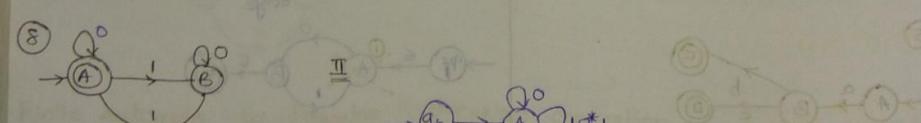
The initial state has incomming edge so create a new initial state



$$\therefore RE = b(c+ab)^*d$$



$$\therefore RE = c^*a(d+ba)^*$$



## TESTING WHETHER A "LANG" IS "REGULAR" OR NOT

If the Language is finite then the language is Regular

⇒ Infinite Language may or may not be Regular. And the algorithm that is used to find whether the given language is Regular or not is called "PUMPING LEMMA".

⇒ Pumping Lemma states that if an Infinite language has to be accepted by the Finite Automata then there has to be a loop inside the finite Automata, and we should find a repeating pattern in the Language which when kept on the loop generates all the remaining strings in the Language, if we do not find any such pattern then NO DFA can be constructed for that language and the language is not Regular.

⇒ pumping Lemma is the Negative test which means pumping Lemma will say that a language is not Regular if we are not able to find such a pattern but pumping lemma doesn't say whether the language is Regular if you find such a pattern.

①  $a^n / n \geq 1$ :  $\rightarrow Q_1 \xrightarrow{aa} Q_2 \rightarrow$  DFA possible = Regular Language

②  $a^m b^m / m, m \geq 1$ :  $\rightarrow Q_1 \xrightarrow{aa} Q_2 \xrightarrow{bb} Q_3 =$  Regular Language

③  $a^n b^n / n \leq 10^{10}$ : The Language is finite ( $\because$  value of  $n$  is bounded) so the Language is Regular

④  $a^n b^n / n \geq 1$ : Not Regular because we need to count the infinite occurrences of 'a's before 'b' but FA has finite memory so the Language is not Regular.

⑤  $w w^R / |w| = 2$ , where  $(a, b)$ :  $\left\{ \begin{array}{l} aaaa \\ abba \\ baab \\ bbbb \end{array} \right\}$  - Finite Language  $\Rightarrow$  Regular Language

⑥  $ww^R / w \in \{a,b\}^*$  = {set of all palindromes} = The Language is not Regular,  
because the string 'w' can be of any length and  
that must be compared Against  $w^R$  and the word/string  
'w' can be infinite Length, Hence FA has no capacity to  
store Infinite Length strings  $\Rightarrow$  Not- Regular Language

⑦  $a^n b^m c^k / n,m,k \geq 1$  = Regular Language [No compositions, a,b,c generated independently]  
 $\Sigma = \{a,b,c\}$

⑧  $a^i b^j / i,j \geq 1$  = Regular Language =  $aa^* bb(bb)^*$

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

⑨  $a^n / n$  is even =  $\{a^0, a^2, a^4, a^6, a^8, \dots\}$  difference can be kept in loop of  
 $1 - 3 - 5 - 7$  DFA

$a^n / n$  is odd =  $\{a^1, a^3, a^5, a^7, \dots\}$  Regular Language

⑩  $a^n / n$  is prime =  $\{a^1, a^3, a^5, a^7, a^9, \dots\}$  = {Not in Ap  $\Rightarrow$  No Repeating pattern}

$a^{n^2} / n \geq 1$  =  $\{a^4, a^9, a^{16}, a^{25}, \dots\}$  = {Not in Ap  $\Rightarrow$  Not RL}

$a^{2^n} / n \geq 1$  =  $\{a^2, a^4, a^8, a^{16}, \dots\}$  = {Not in Ap  $\Rightarrow$  Not RL}

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

⑪  $a^i b^{j^2} / i, j \geq 1$  = Even the a,b are generated independently, due

to  $j^2$ , there is not Repeating pattern in the

Language and hence Not Regular

⑫  $a^i b^{2^n} / i, n \geq 1$  = a can be independently generated and  $2^n (2^i)$   
the power n) has no Repeating pattern.

⑬  $a^i b^p / i \geq 1$  =  $b^p$  cannot be generated and NO Repeating pattern.

⑭  $f(w) / n_a(w) = n_b(w)$  string and At least one character is a

$$n_a(w) \leq n_b(w)$$

$$n_a(w) \geq n_b(w)$$

= Not Regular Language ( $\because$  counting a's need  
Infinite Memory)

⑮  $n_a(w) \bmod 3 \leq$

⑯  $a^n / n \geq 10 \Rightarrow$

⑰  $www^R / w \in \{a\}$

⑱  $a^n b^n c^n / n \geq 1$

⑲  $a^n b^{n+m} c^m / n,$

Imp v.v Imp

⑳  $wxw^R / w, x \in$

㉑  $wxw^R / w \in \{0,1\}$

$$1 \times 1 = 5$$

㉒  $xwxy / x, y \in$

㉓  $xww^R / x, w \in$

㉔  $wwy / w, y \in$

㉕ Set of all s

MT2 36  
 not Regular  
 lengths and  
 be word/string  
 capacity to  
 be Language  
 generated independently  
 $a^* b^* c^*$

⑥  $n_a(w) \bmod 3 \leq n_b(w) \bmod 3 \Rightarrow$  Regular Language (DFA can be constructed for mod 3 counters)

⑦  $a^n / n \geq 10^3 \Rightarrow$  Regular Language (DFA can be constructed)

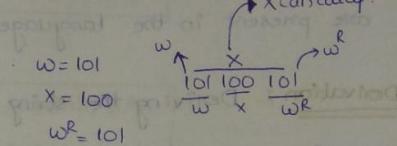
⑧  $w w w^R / w \in (a, b)^*$  = Not Regular Language

⑨  $a^n b^n c^n / n \geq 1$  = Not Regular Language

⑩  $a^n b^{n+m} c^m / n, m \geq 1$  = Not Regular Language

Imp v.v. Imp

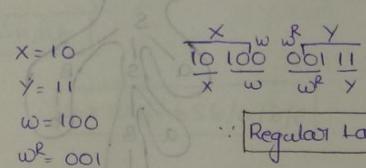
⑪  $w x w^R / w, x \in (0, 1)^+$  : Regular Language :  $w = 101$



$$RE = 1 (0+1)^* 1 + 0 (0+1)^* 0$$

⑫  $w x w^R / w \in (0, 1)^+$  : Not Regular Language ( $\because x$  cannot go beyond '5' length)

⑬  $x w w^R y / x, y, w \in (0, 1)^+$  :



$$RE: (0+1)^* 00 (0+1)^* \\ (or) \\ (0+1)^* 11 (0+1)^*$$

⑭  $x w w^R / x, w \in (0, 1)^+$  : Not Regular Language

⑮  $w w^R y / w, y \in (0, 1)^+$  : Not Regular Language

⑯ Set of all strings having equal number of '01' and '10' is Regular.

as need

## GRAMMARS

L-65

A Grammar is generally represented by  $(V, T, P, S)$

$V = \text{Vertices}$   
 $T = \text{Terminals}$   
 $P = \text{productions}$

$$\begin{array}{ll} S \rightarrow aSB & V = \{S, B\} \\ S \rightarrow aB & T = \{a, b\} \\ B \rightarrow b & P = \left\{ \begin{array}{l} S \rightarrow aSB \\ S \rightarrow aB \\ B \rightarrow b \end{array} \right\} \\ & S = \{S\} \end{array}$$

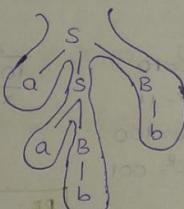
⇒ The main intention of the grammar is to generate all the strings that are present in the language

Derivation: Deriving the string from the grammar using the start symbol

$$\begin{array}{l} S \rightarrow aSB \\ S \rightarrow aB \\ B \rightarrow b \end{array}$$

Derivation of aabb

$$\begin{array}{l} S \rightarrow aSB \\ \Rightarrow a \underline{a} \underline{B} B \\ \Rightarrow a a \underline{\underline{B}} B \\ \Rightarrow a a b \underline{B} \\ \Rightarrow a a b b \end{array} \quad \begin{array}{l} \xrightarrow{\text{Sentential forms (or) Sequential forms.}} \\ \left\{ \begin{array}{l} \text{Left most derivation} \\ \text{Right most derivation} \end{array} \right. \end{array}$$



### CONSTRUCTION OF GRAMMARS

$$① L = \{aa, ab, ba, bb\}$$

$$S \rightarrow aa/ab/ba/bb$$

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

$$\begin{array}{l} S \rightarrow AA \\ A \rightarrow a/b \end{array}$$

$$② L = \{a^n / n \geq 1\} - L = \{a, aa, aaa, aaaa, \dots\}$$

$$A \rightarrow aA/e$$

$$\begin{array}{l} S \rightarrow A \\ A \rightarrow aA/a \end{array}$$

$$③ L = (a+b)^* \quad L = \{aab, ab, ba, aa, bb, \dots\}$$

$$S \rightarrow aS/bS/\epsilon$$

$$\begin{array}{l} ④ L = \{ \text{set of all strings over } (a+b) \} \\ = (a+b)(a+b) \end{array}$$

$$\begin{array}{l} ⑤ L = \{ \text{Length of } a \} \\ = (a+b+\epsilon) \cdot A \end{array}$$

$$\begin{array}{l} ⑥ L = \{ \text{start with } a } \\ = a(a+b) \end{array}$$

$$\begin{array}{l} ⑦ L = \{ \text{contain and end with } a } \\ = a(a+b)^* \end{array}$$

$$L = \{ \text{same as } ⑦ \}$$

$$⑧ L = \{a^n b^n / n \geq 1\}$$

$$\begin{array}{l} ⑨ L = \{a^m b^k\} \cup \{b^m a^k\} \\ \text{where } m, k \in \{0, 1, 2, \dots\} \end{array}$$

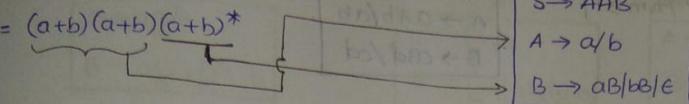
$$⑩ L = \{ \text{set of all strings over } (a+b) \}$$

$$⑪ L = \{a^n b^m / n \leq m\}$$

$$⑫ L = \{a^n c^m b^n\}$$

④  $L = \{ \text{set of all strings of length at least 2} \}$

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



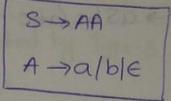
$$S \rightarrow AAB$$

$$A \rightarrow a/b$$

$$B \rightarrow aB/bB/\epsilon$$

⑤  $L = \{ \text{length at most 2} \}$

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

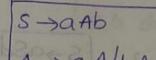


$$S \rightarrow AA$$

$$A \rightarrow a/b/\epsilon$$

⑥  $L = \{ \text{start with 'a' end with 'b'} \}$

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

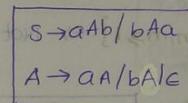


$$S \rightarrow aAb$$

$$A \rightarrow aA/bA/\epsilon$$

⑦  $L = \{ \text{start and end with diff symbols} \}$

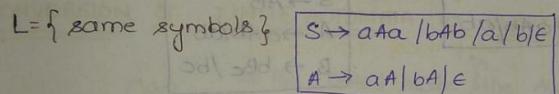
$$= a(a+b)^*b + b(a+b)^*a$$



$$S \rightarrow aAb/bAa$$

$$A \rightarrow aA/bA/\epsilon$$

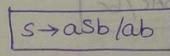
$L = \{ \text{same symbols} \}$



$$S \rightarrow aAa/bAb/a/b/\epsilon$$

$$A \rightarrow aA/bA/\epsilon$$

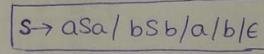
⑧  $L = \{ a^n b^n / n \geq 1 \}$



$$S \rightarrow aAb/ab$$

⑨  $L = \{ waw^R \cup waw^L \cup wbw^R \}$

$$w \in (a+b)^*$$

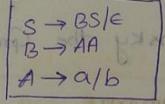


$$S \rightarrow awa/bwb/a/b/\epsilon$$



⑩  $L = \{ \text{set of all strings of even length} \}$

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

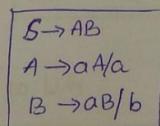


$$S \rightarrow BS/\epsilon$$

$$B \rightarrow AA$$

$$A \rightarrow a/b$$

⑪  $L = \{ a^n b^m / n, m \geq 1 \}$

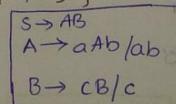


$$S \rightarrow AB$$

$$A \rightarrow aA/a$$

$$B \rightarrow aB/b$$

$L = \{ a^n b^n c^m / n, m \geq 1 \}$

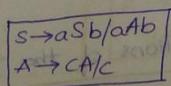


$$S \rightarrow AB$$

$$A \rightarrow aAb/ab$$

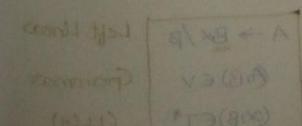
$$B \rightarrow CB/c$$

⑫  $L = \{ a^n c^m b^n \} / n, m \geq 1$



$$S \rightarrow aSb/aAb$$

$$A \rightarrow CA/c$$



$$S \rightarrow aBc/aBc$$

$$A \rightarrow CB/c$$

$$B \rightarrow Tc(Ac)$$

## PUSH DOWN AUTOMATA

L-68

Push Down Automata = Finite Automata + stack

Push Down Automata =  $(Q, \Sigma, \delta, q_0, Z_0, F, \gamma)$

$Q$  = finite set of states

$Z_0$  = Bottom of the Stack

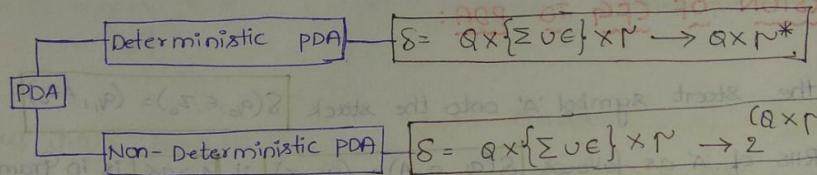
$\Sigma$  = Input symbol

$F$  = set of final states

$\delta$  = Transition function

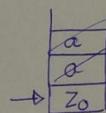
$\gamma$  = Stack Alphabet

$q_0$  = Initial state



$$\textcircled{1} \quad L = \{a^n b^n / n \geq 1\}$$

Let the string be  $aabb$



$(a, a/aa)$

$(a, Z_0/aZ_0)$

$(b, a/\epsilon)$

$(b, a/\epsilon)$

$(a, Z_0/aZ_0)</math$

$$⑬ L = \{a^n b^n c^m d^m \mid n, m \geq 1\}$$

$$\begin{array}{l} S \rightarrow AB \\ A \rightarrow aAb/ab \\ B \rightarrow cBd/cd \end{array}$$

$$⑭ L = \{a^n b^n c^n \mid n \geq 1\}$$

$$S \rightarrow aSbc/abc$$

$$⑮ L = \{a^n b^{2n} \mid n \geq 1\}$$

$$S \rightarrow aSbb/abb$$

$$⑯ L = \{a^n b^m c^m d^n \mid n, m \geq 1\}$$

$$\begin{array}{l} S \rightarrow aSd/aAd \\ A \rightarrow bAc/bc \end{array}$$

$$⑰ L = \{a^n b^m c^n d^m \mid n, m \geq 1\} \Rightarrow \text{Not possible to give CFG.}$$

$$⑱ L = \{a^n a^n b^m c^m\}$$

$$\begin{array}{l} S \rightarrow aSc/aAc \\ A \rightarrow aAb/ab \end{array}$$



For this language  
we cannot give such  
grammar & we give LSG  
for this language

$$\begin{array}{l} ⑲ L = \{a^n b^{n+m} c^m \mid n, m \geq 1\} \\ S \rightarrow AB \\ A \rightarrow aAb/ab \\ B \rightarrow bBc/bc \end{array}$$

$$⑳ L = \{a^n b^m c^{n+m} \mid n, m \geq 1\}$$

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

$$\begin{array}{l} S \rightarrow aSc/aAc \\ A \rightarrow bAc/bc \end{array}$$

L-66

### CLASSIFICATION OF GRAMMARS

According to Chomsky the Grammars are classified into 4 types.

1) Type 3 / Regular Grammars.

2) Type 2

3) Type 1

4) Type 0

Type - 3:

If the Grammar has all the productions of the form

Right Linear  
Grammar  
(RLG)

$$\begin{array}{l} A \rightarrow \alpha B/B \\ (\alpha, B) \in V \\ (\alpha, B) \in T^* \end{array}$$

$$\begin{array}{l} A \rightarrow Bx/B \\ (A, B) \in V \\ (A, B) \in T^* \end{array}$$

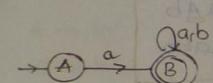
Left Linear  
Grammar  
(LLG)

Ex:

$$\begin{array}{l} \text{RLG} \\ A \rightarrow aB/B \\ B \rightarrow aB/B \end{array}$$

The Languages  
are always

### CONVERSION OF

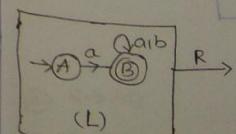


→  $\rightarrow A \xrightarrow{a} B$

→  $\rightarrow A \xrightarrow{a} B$

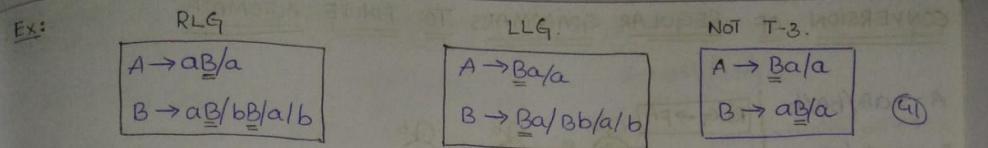
### Finite Automata

1) Convert the ab



→  $\rightarrow A \xrightarrow{a} B$

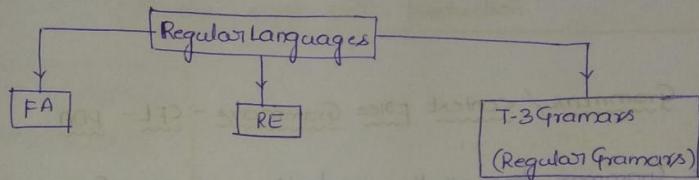
→  $\rightarrow A \xrightarrow{a} B$



bc bc  
 $\eta(bc)^n$  X  
 this language  
 give such  
 we give LSG.  
 language

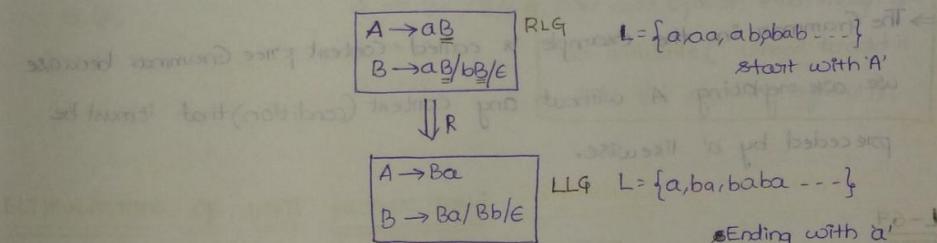
The Languages that are generated by Type-3 Grammar  
 are always Regular Languages.

T-3 Grammars should be either  
 LLG or RLG but not both.

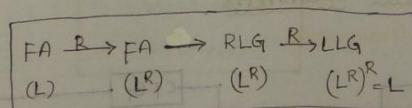


### CONVERSION OF FINITE AUTOMATA TO REGULAR GRAMMAR

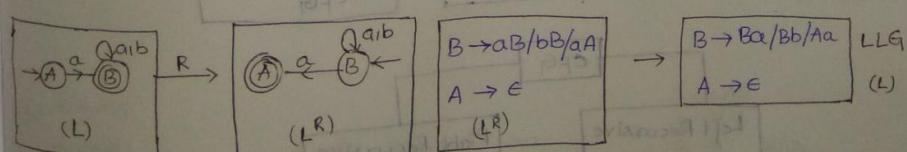
$\Rightarrow$  The Initial state in the Automata is same as the Start State in the Grammar



Finite Automata to LLG:



Convert the above FA to LLG.

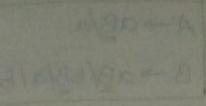
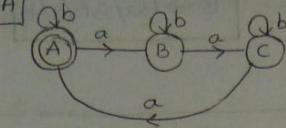
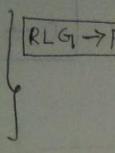


Left Linear  
Grammar  
(LLG)

## CONVERSION OF REGULAR GRAMMARS TO FINITE AUTOMATA

42

$$\begin{aligned} A &\rightarrow ab/bA/b \\ B &\rightarrow aC/bB \\ C &\rightarrow aA/bC/a \end{aligned}$$



LLG<sub>1</sub> → FA CONVERSION :

$$\begin{array}{c} \text{LLG}_1 \xrightarrow{R} \text{RLG} \xrightarrow{R} \text{FA} \xrightarrow{R} \text{FA} \\ (\text{L}) \quad (\text{LR}) \quad (\text{LR})^R = \text{L} \end{array}$$

Type-2 Grammars / context free Grammars - CFL - PDA

If the Grammar has all the productions of the form

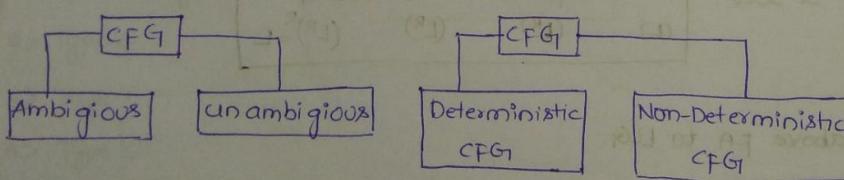
$A \rightarrow \alpha$  AEV  
 $\alpha \in (VUT)^*$

Ex:  $A \rightarrow aAb/bb$   
 $A \Rightarrow a\underline{A}b$   
 $\Rightarrow aabb$

→ The Grammar in the example is called Context Free Grammar because we are replacing 'A' without any content (condition) that it must be preceded by 'a' likewise.

L-67

Type 2: context free Grammars



Left Recursive Grammar

Right Recursive Grammar

## ELIMINATION

$$\begin{array}{l} ① S \rightarrow aSb \\ A \rightarrow a \\ \vdots \\ B \rightarrow bBB \end{array}$$

$$\begin{array}{l} ② S \rightarrow AB \\ A \rightarrow aAa \\ B \rightarrow bBB \end{array}$$

$$\begin{array}{l} ③ S \rightarrow Abac \\ A \rightarrow BC \\ B \rightarrow B/E \\ C \rightarrow D/E \\ D \rightarrow d \end{array}$$

## ELIMINATION

$$\begin{array}{l} ① S \rightarrow Aa/B \\ B \rightarrow A/bb \\ A \rightarrow abc \end{array}$$

$$\begin{array}{l} ② S \rightarrow AB \\ A \rightarrow a \\ B \rightarrow C/b \\ C \rightarrow D \\ D \rightarrow E \end{array}$$

ELIMINATION OF  $\epsilon$ -PRODUCTIONS

$$\textcircled{1} \quad S \rightarrow aSb / aAb$$

$A \rightarrow \epsilon$

$$\left. \begin{array}{l} S \rightarrow asb / ab \\ S \rightarrow aSb / ab \end{array} \right\} \Rightarrow$$

$$S \rightarrow asb / (aAb) / ab$$

↓  
No production with  $\epsilon$

$$\Rightarrow S \rightarrow asb / ab$$

$$\textcircled{2} \quad S \rightarrow AB$$

$A \rightarrow aAA / \epsilon$

$$B \rightarrow bBB / \epsilon$$

Nullable states:  $\{A, B, S\}$

⇒ Whenever the start state is nullable then the language contains  $\epsilon$  and it must be present in the start state production.

$$S \rightarrow AB / B / A / \epsilon$$

$$A \rightarrow aAA / aA / a$$

$$B \rightarrow bBB / bB / b$$

$$\textcircled{3} \quad S \rightarrow AbAC$$

$$A \rightarrow BC$$

$$B \rightarrow B / \epsilon$$

$$C \rightarrow D / \epsilon$$

$$D \rightarrow d$$

Nullable symbols:  $\{B, C, A\}$

$$S \rightarrow AbAC / AbA / ba / baC$$

$A \rightarrow Bc / B / C$  (Don't add epsilon even though B/C are nullable ( $\because$  we need to add it only for starting symbol)).

ELIMINATION OF UNIT PRODUCTIONS

$$\textcircled{1} \quad S \rightarrow Aa / B$$

$$B \rightarrow A / bb$$

$$A \rightarrow a / bc / B$$

Step-1:

$$\begin{array}{l} S \rightarrow Aa / bb / a / bc \\ B \rightarrow bb / a / bc \\ A \rightarrow a / bc / bb \end{array}$$

$$\Rightarrow S \rightarrow B \rightarrow bb$$

$$\Rightarrow S \rightarrow \cancel{A} \rightarrow a \rightarrow \cancel{bc}$$

$$\Rightarrow B \rightarrow \cancel{A} \rightarrow a$$

$$\Rightarrow A \rightarrow \cancel{B} \rightarrow bb$$

$$A \rightarrow bb$$

$$\textcircled{2} \quad S \rightarrow AB$$

$$A \rightarrow a$$

$$B \rightarrow C / b$$

$$C \rightarrow D$$

$$D \rightarrow E$$

$$E \rightarrow a$$

$$B \rightarrow \cancel{C} \rightarrow D \rightarrow E \rightarrow a$$

$$C \rightarrow \cancel{D} \rightarrow E \rightarrow a$$

$$D \rightarrow \cancel{E} \rightarrow a$$

$$S \rightarrow AB$$

$$A \rightarrow a$$

$$B \rightarrow \cancel{C} / b$$

$$C \rightarrow a$$

$$D \rightarrow a$$

$$E \rightarrow a$$

useless symbols  
Not Reachable  
from Start State

## ELIMINATION OF USELESS SYMBOLS

$$① S \rightarrow A/a \quad T = \{a, b\}$$

$$A \rightarrow B/c/b$$

$$V = \{S, A, B, C\}$$

$$B \rightarrow AB/C$$

$$C \rightarrow ac/B$$

Now, the useful symbols (which will generate some terminal) are {a, b, S, A}

$\Rightarrow B, C$  are not useful symbols ( $\because$  I could not generate any string from B, C.)

[Chor]

$$\begin{cases} S \rightarrow a \\ A \rightarrow b \end{cases}$$

But here 'A' is not reachable, therefore the mini mixed Grammar will be  $S \rightarrow a$

$$② S \rightarrow AB/A$$

$$A \rightarrow aAb/bAa/a$$

$$B \rightarrow bbA/aaB/AB$$

$$\begin{cases} C \rightarrow abCA/aDb \\ D \rightarrow bD/ac \end{cases} \times$$

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

$$V = \{S, A, B, C, D\}$$

$$\text{Useful symbols} = \{a, b, A\}$$

$$\{B, S\}$$

$$\begin{cases} S \rightarrow AB \\ A \rightarrow aAb/bAa/a \\ B \rightarrow bbA/aaB/AB \end{cases}$$

$$③ S \rightarrow ABC/BaB$$

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

$$\text{Useful symbols} = \{a, b, B, A, S\}$$

$$A \rightarrow aA / BaC/aaa$$

$$V = \{S, A, B, C\}$$

$$B \rightarrow bBb/a$$

$$C \rightarrow CA/AC \times$$

$$S \rightarrow BaB$$

$$A \rightarrow aA/aaa \times$$

$$B \rightarrow bBb/a$$

$\Rightarrow$  Not Reachable  $\therefore A \leftarrow S$

$$\Rightarrow S \rightarrow BaB \quad \text{and } A \leftarrow S$$

$$B \rightarrow bBb/a \quad \text{and } A \leftarrow S$$

① convert

$$S \rightarrow bA/a$$

$$A \rightarrow bAA$$

$$B \rightarrow aBb$$

Advantages

1) Length c

2) Derivation

3) The no. o

4) It is ec

② convert

$$BA \leftarrow S$$

$$A \leftarrow A$$

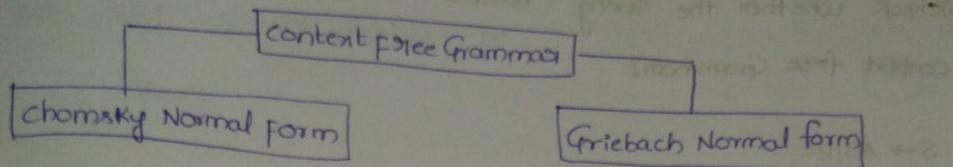
$$B \leftarrow A$$

$$BA \leftarrow S$$

$$A \leftarrow A$$

$$B \leftarrow A$$

## CNF - INTRODUCTION [CHOMSKY-NORMAL FORM]



CNF:

If all the productions are of the form  $A \rightarrow BC$  or  $A \rightarrow a$  where  $A, B$  are variables and ' $a$ ' is a terminal.

Advantages:

- 1) Length of each production is Restricted
- 2) Derivation tree (or) parse tree obtained from CNF is always Binary tree
- 3) The no. of steps required to derive a string of length  $|w|$  is  $(2|w|-1)$
- 4) It is easy to apply CYK membership Algorithm.  $O(n^3)$  time

① Convert the following into CNF?

$$S \rightarrow bA / aB$$

$$A \rightarrow bAA / aS / a$$

$$B \rightarrow aBB / bS / b$$

Let  $\{ N_a \rightarrow a \}$  then  
 $N_b \rightarrow b$

$$S \rightarrow N_b A / N_a B$$

$$A \rightarrow N_b (AA) / N_a S / a$$

$$B \rightarrow N_a (BB) / N_b S / b$$

$$N_a \rightarrow a$$

$$N_b \rightarrow b$$

$$\Rightarrow \begin{array}{l} S \rightarrow N_b A / N_a B \\ A \rightarrow N_b T / N_a S / a \\ B \rightarrow N_a P / N_b S / b \\ N_a \rightarrow a \\ N_b \rightarrow b \\ T \rightarrow AA \\ P \rightarrow BB \end{array}$$

## CYK ALGORITHM (MEMBERSHIP ALGORITHM)

46

cell no  $\underline{\underline{1}}$   
The value  
Now see

i) check whether the string "baaba" is a valid member of the following context free Grammar?

$$S \rightarrow AB/BC$$

$$A \rightarrow BA/a$$

$$B \rightarrow cc/b$$

$$C \rightarrow AB/a$$

Sol: The CYK Algorithm is used to check the string belong to the language generated by the Grammar or not.

⇒ The CYK Algorithm can be applicable for only CNF form Grammars

Given Grammar

$$\begin{aligned} S &\rightarrow AB/BC \\ A &\rightarrow BA/a \\ B &\rightarrow cc/b \\ C &\rightarrow AB/a \end{aligned}$$

CNF form of the grammar (1)				
5	4	3	2	1
A(S)C	∅	∅	A, S	B
S, A, C	B	B	A, C	
B	S, C	A, C		
A, S	B			
A, C				

Cell no  $(1,1)$ :

Now the  $(1,1)$  represent the variable 'b' in the required string  
 $\downarrow$   
 $[1 \text{ to } 1]$

⇒ Now check what are the variables that are required to generate 'b'

$$\Rightarrow B \checkmark$$

so the value of cell  $(1,1) = B$

Cell no  $(3,3)$ :

The Alphab

cell no :  $\underline{\underline{4,4}}$

$(4,4) = b$

Cell no  $(1,2)$

$(1,2)$  mean:

Cell no  $(2,3)$

$(2,3) = (2,2)$

$\{A\} = \{b\}$

$\{2, A\} = \{b\}$

Cell no  $(3,4)$

$(3,3) \times (4,4)$

cell no (2,2) :-

The value of (2,2) in the given string is 'a' (2 to 2)  $\rightarrow$  start with '2' and end with '2'.

Now see which variable derives 'a'

$$= \{A, C\}$$

$$\{2+ \} \times \{2A\} = \{a\} \times \{2\} = (47)$$

cell no (3,3) :-

The Alphabet in the (3,3) is 'a' so  $\{A, C\}$  will generate 'a'.

$$= \{A, C\}$$

cell no (4,4) :-

$(4,4) = b \Rightarrow$  Generated by 'B'

$(5,5) = a' = \{A, C\}$  will generate 'a'.

cell no (1,2) :-  $\begin{bmatrix} 1 & 2 \\ b & a \end{bmatrix} \times \begin{bmatrix} 3 & 4 & 5 \\ a & b & a \end{bmatrix} = (4)$

(1,2) means start with '1' and end with '2'  $\Rightarrow$  "ba"

$$(12) = (11) \times (22) \quad [\text{cross product}]$$

$$= \{B\} \times \{A, C\} = \{BA, BC\} = \{A, S\}$$

Generated by  
Generated by  
 $\{A\} = (22) \quad \{A\} = (4)$   
 $\{A, 2\}$

cell no (2,3) :-

$$(23) = (22) \times (33)$$

$$= \{A, C\} \times \{A, C\}$$

$$= \{AA, AC, CA, CC\} = \{B\}$$

$\xrightarrow{\quad \times \quad \times \quad \times \quad B \quad}$  Not derived by any of the symbols.

cell no (3,4) :-

$$(33) \times (44) = \{A, C\} \times \{B\}$$

$$= \{AB, CB\} \quad \text{and left generated} = \{BA, BC\} \quad \text{right generated} = \{A, S\} = \{A, S\}$$

$$(1,3) = \boxed{baa} \underset{\substack{| \\ 1 \\ 2 \\ 3}}{b} \underset{\substack{| \\ 4 \\ 5}}{a} = \underline{(baa)}$$

$$(1) \quad (2 \ 3)$$

② GRIE BACH

If all the  
 $x \in V^*$  then it

Ex:  $A \rightarrow aB$   
 $A \rightarrow a$

Advantages

1> The no. of

2> GNF is

CONVERSION

1> Push the

2> push RHS

3> Add final

SC

$$\text{NOW, } (1,5) = baa \underset{\substack{| \\ 1 \\ 2 \\ 3 \\ 4 \\ 5}}{a}$$

$$(1,4) = baab$$

$$(1,5) = \begin{cases} (11) (2,5) = \{A, S\} \\ (12) (3,5) = \{S, C\} \\ (13) (4,5) = \{\emptyset\} \\ (14) (5,5) = \{\emptyset\} \end{cases} \quad (14) = \begin{cases} (11) (2,4) = BBX \\ (12) (3,4) = AS, AC, SS, SC \\ (13) (4,4) = \{\emptyset\} \end{cases}$$

$$(2,5) = \underset{\substack{| \\ 2 \\ 3 \\ 4 \\ 5}}{2 \ 3 \ 4 \ 5}$$

$$(2,5) = \begin{cases} (22) (3,5) = AB, CB \rightarrow \{S, C\} \\ (23) (4,5) = BA, BS = \{A\} \\ (24) (4,5) = BA, BC = \{A, S\} \end{cases}$$

If the cell (last cell) (1,5) contains the starting symbol 'S': (we can generate the string from the given Grammar  $\{S\} \times \{> A\} = \{AB\} \times \{SS\}$ )

Now, what are the various substrings that can be generated from the Grammar?

Now check where the starting symbols are present in the table i.e. it is

present at cells (1,2) (3,4) (4,5) (2,5) (1,5)

$$\begin{array}{l} \downarrow \downarrow \downarrow \downarrow \\ \text{baaba} \quad \{ba\} \{ab\} \{ba\} \{aab\} = \{ba, ab, aaba\} \end{array}$$

Given string

If all the productions of the Grammar are of the form  $A \rightarrow a\alpha$  where  $\alpha \in V^*$  then it is called GNF.

Ex:  $A \rightarrow aBCDE$

$A \rightarrow a$

### Advantages

1) The no. of steps required to generate a string of length  $|w|$  is  $|w|$ .

2) GNF is useful in order to convert a CFG to "push down Automata".

### CONVERSION OF CFG TO PDA:

1) Push the start symbol 'A' onto the stack

$$\delta(q_0, \epsilon, z_0) = (q_1, Az_0)$$

2) push RHS of 'A' as follows

$$\delta(q_1, a, A) = (q_1, \alpha)$$

if  $A \rightarrow a\alpha$  is in Grammer

$$\delta(q_1, b, A) = (q_1, \beta)$$

if  $A \rightarrow b\beta$  is in Grammer

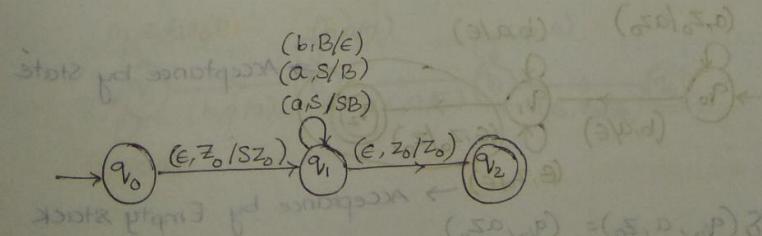
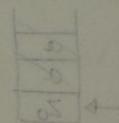
3) Add final state with

$$\delta(q_1, \epsilon, z_0) = (q_f, \epsilon)$$

①  $S \rightarrow aSB/ab \rightarrow$  Not in GNF

$$\begin{array}{l} S \rightarrow aSB/AB \\ B \rightarrow b \end{array}$$

In GNF

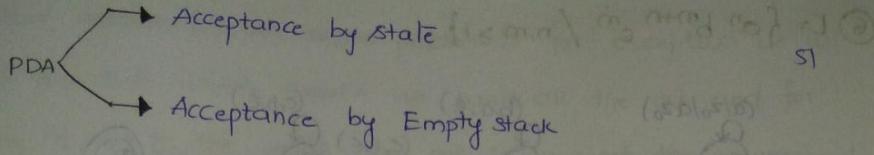


$$(5, 5, 5, 5) = (5, 5, 5, 5) \beta$$

$$(6, 6, 6, 6) = (6, 6, 6, 6) \beta$$

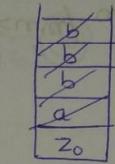
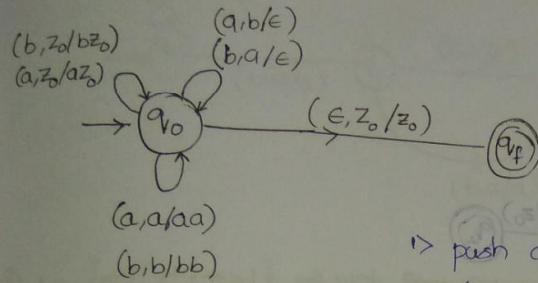
$$(7, 7, 7, 7) = (7, 7, 7, 7) \beta$$

$$(5, 5, 5, 5) = (5, 5, 5, 5) \beta$$



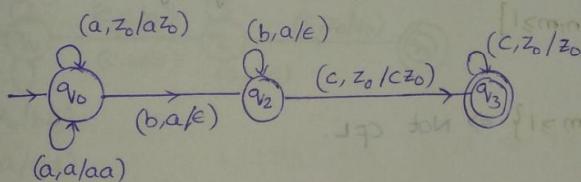
$$② \omega \in (a,b) / n_a(\omega) = n_b(\omega)$$

Let the string be  $a b b b a a$



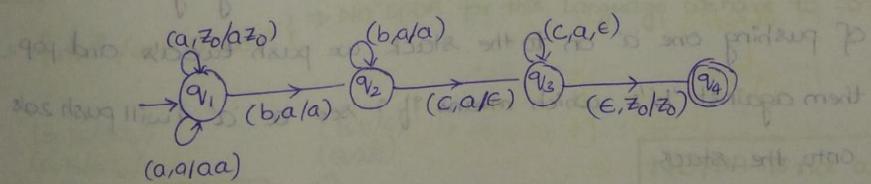
- 1> push 'a's onto the stack.
- 2> If you see 'b', pop one 'a' & vice versa.
- 3> If your Alphabet is 'b' & Bottom of Stack =  $z_0$   
→ push 'b' on stack.

$$③ L = \{a^n b^n c^m / n, m \geq 1\}$$

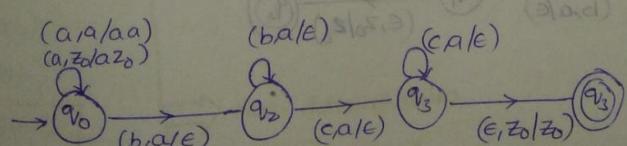


L-69

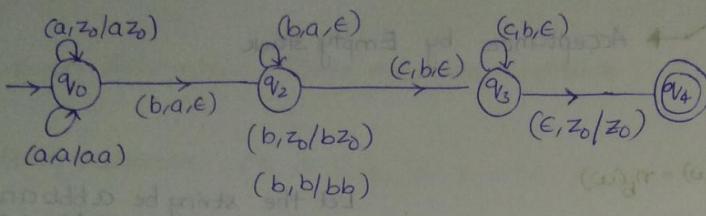
$$④ L = \{a^n b^m c^n / n, m \geq 1\}$$



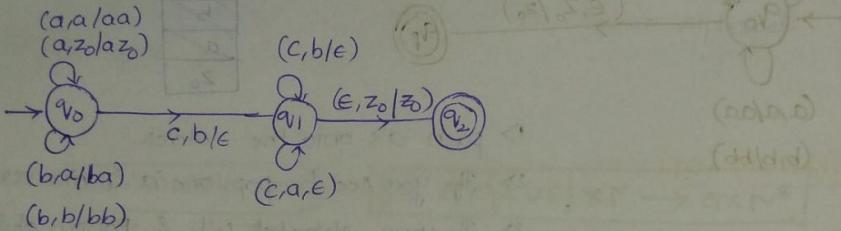
$$⑤ L = \{a^{m+n} b^m c^n / m, n \geq 1\}$$



⑥  $L = \{a^n b^{m+n} c^m / n, m \geq 1\}$



⑦  $L = \{a^n b^m c^{n+m} / n, m \geq 1\}$



⑧  $L = \{a^n b^n c^m d^m / n, m \geq 1\}$

⇒ context free languages

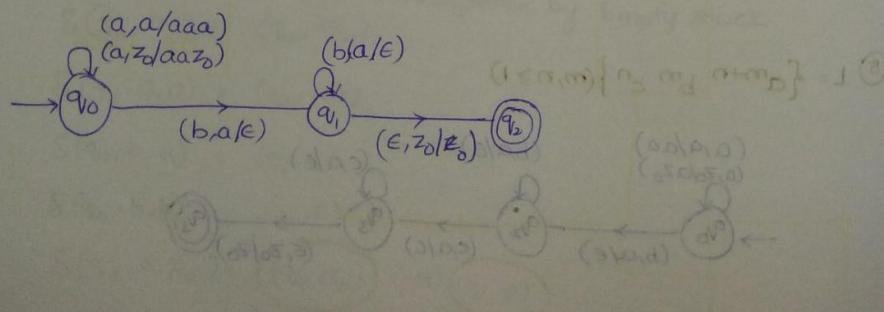
$L = \{a^n b^m c^m d^n / n, m \geq 1\}$

⑨  $L = \{a^n b^m c^n d^m / n, m \geq 1\}$  ⇒ Not CFL.

⑩  $L = \{a^n b^{2n} / n \geq 1\}$

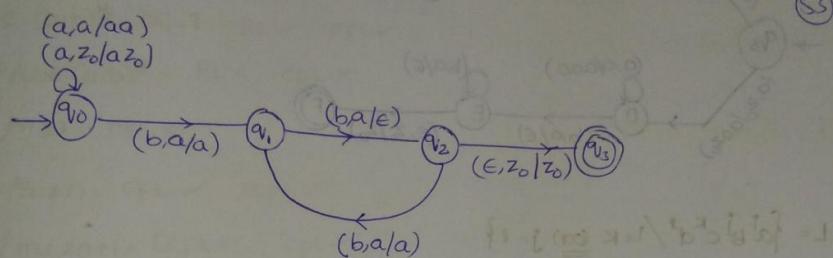
The language accepted is  $L = \{abb, aabbbb, aaabbbbbbb, \dots\}$

⇒ The solution to construct PDA for the above language is instead of pushing one 'a' on to the stack we push two 'a's and pop them against 'b's which means if I see a 'a' I will push 2a's onto the stack.



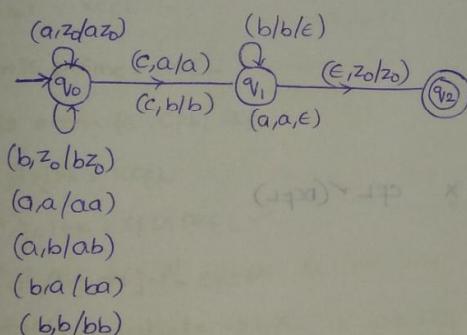
52

The other solution for the above problem is for every two 'b's we should pop 1 'a'. which means we push all the 'a's and for pushing 'b' on to the stack we pop 1 'a' against the occurrence of 2 'b's.



$$\text{① } L = \{a^n b^n c^n | n \geq 1\} \Rightarrow \text{Not Rego CFL}$$

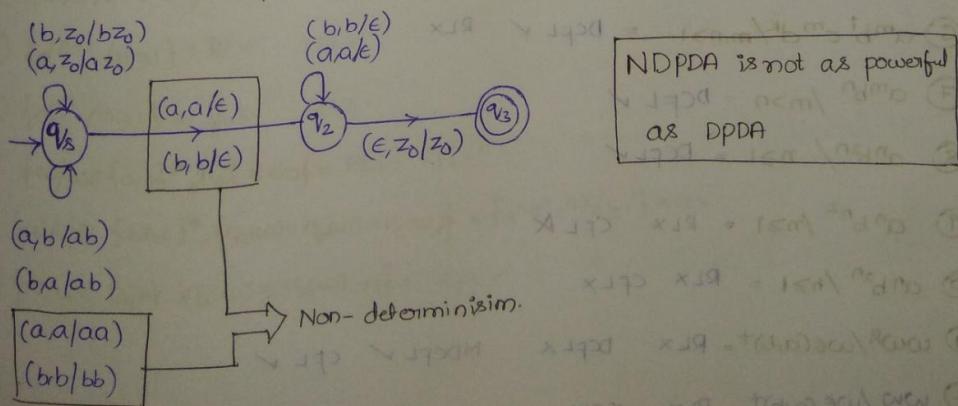
$$\text{② } L = \{wcbw^R | w \in (a,b)^+\}$$



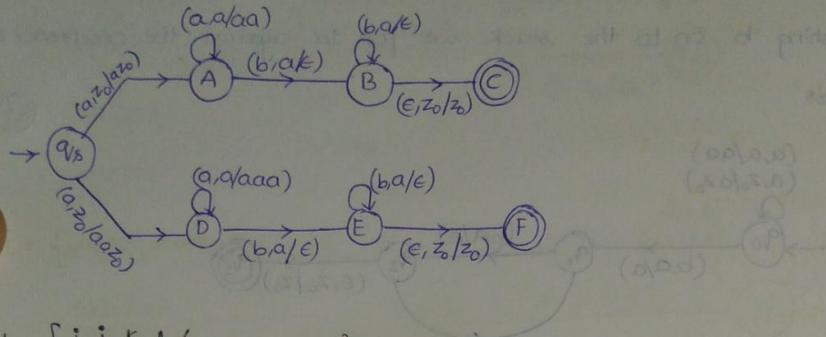
instead

L-70

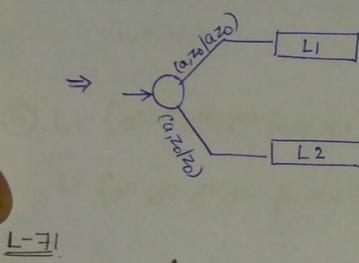
$$L = \{ww^R | w \in (a+b)^+\} \Rightarrow \text{No "DPDA" for this language we have to construct "NDPDA"}$$



$$L = \{a^n b^n / n \geq 1\} \cup \{a^n b^{2n} / n \geq 1\}$$



$$\begin{aligned} ③ L &= \{a^i b^j c^k d^l / i=k \text{ } j=l\} \\ &= \{a^m b^j c^m d^l\} \cup \{a^i b^m c^k d^m\} \end{aligned}$$



$$\begin{aligned} ① a^{m+n} b^n c^m / n, m \geq 1 &\Rightarrow RL \times CFL \checkmark (DCFL) \\ a^m b^{m+n} c^n / n, m \geq 1 \end{aligned}$$

$$② amb^n c^{m+n} = RL \times CFL \checkmark$$

$$③ am b^m c^n d^n = RL \times CFL \checkmark$$

$$④ a^m b^n c^m d^n / m, n \geq 1 = RL \times CFL \times DCFL$$

$$⑤ am b^n c^n d^m / m, n \geq 1 = CFL \checkmark \quad RL \times$$

$$⑥ amb^i c^m d^k / m, n \geq 1 = DCFL \checkmark \quad RL \times$$

$$⑦ amb^n / m > n = DCFL \checkmark$$

$$⑧ a^n b^{2n} / n \geq 1 = DCFL \checkmark$$

$$⑨ a^n b^{n^2} / n \geq 1 = RL \times CFL \times$$

$$⑩ a^n b^{2n} / n \geq 1 = RL \times CFL \times$$

$$⑪ \omega \omega^R / \omega \in (a, b)^+ = RL \times DCFL \times NDCFL \checkmark \quad CFL \checkmark$$

$$⑫ \omega \omega / \omega \in (a, b)^+ = RL \times \times CFL$$

$$⑬ anb^n c^m / m$$

$$⑭ anb^n c^{ndn}$$

$$⑮ anb^{2n} c^{3n} /$$

$$⑯ xcy / x, y, \epsilon$$

$$⑰ xz^R / x \in C$$

$$⑱ \omega \omega \omega^R / \omega$$

$$⑲ a^n b^{3n} / n \geq 1$$

$$⑳ amb^n / m \geq 1$$

$$㉑ amb^n / m \geq 1$$

$$㉒ ab^j / i \neq j$$

$$㉓ a^{n^2} / n \geq 1$$

$$㉔ a^{2n} / n \geq 1$$

$$㉕ a^n / m \geq 1$$

$$㉖ am / mis$$

$$㉗ ak / k \leq e$$

$$㉘ a^i b^j c^k / i > j$$

$$㉙ a^i b^j c^k / j = i$$

$$㉚ a^i b^j c^k d^l /$$

$$㉛ amb^i c^k d^n$$

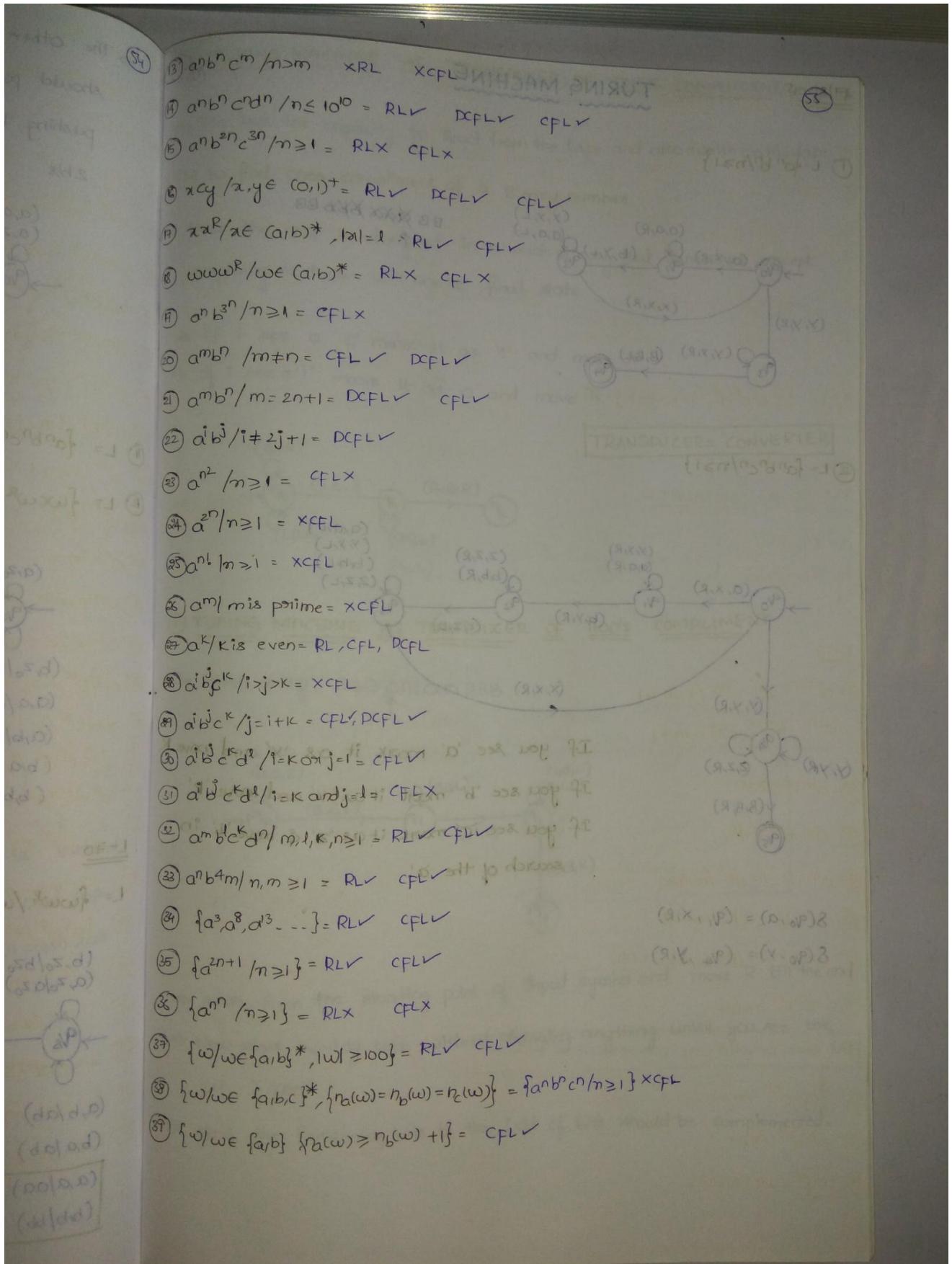
$$㉜ a^n b^{4m} / n,$$

$$㉝ \{a^3, a^8, a^1\}$$

$$㉞ \{a^{2n+1} / n\}$$

$$㉟ \{a^{2n} / n\}$$

$$㉟ \{\omega / \omega \in f\}$$



### ③ TURING MACHINE AS TRANSDUCER OF ONE'S COMPLEMENT

⇒ TM has the capacity to Read from the tape and also write on the tape

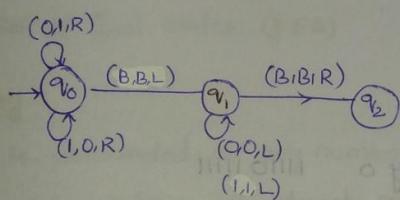
⇒ TM to find ones compliment of a Binary number

⇒ Here the TM is designed in a such a way that it need not accept anything so there will not be final state

⇒ If i see a '0' mark it as '1' and move 'R'

⇒ If i see a '1' mark it as '0' and move 'R'

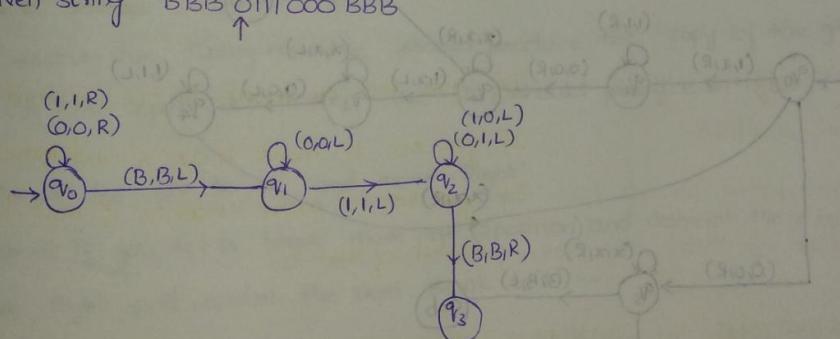
**TRANSDUCER = CONVERTER**



### ④ TURING MACHINE AS TRANSDUCER OF TWO'S COMPLEMENT

The Given string BBB 0111000 BBB

and move R  
move R  
move 'L' in



⇒ start from the starting point of input symbol and move 'R' till the end

⇒ Now start moving Left without changing anything until you see the 1st '1' (one)

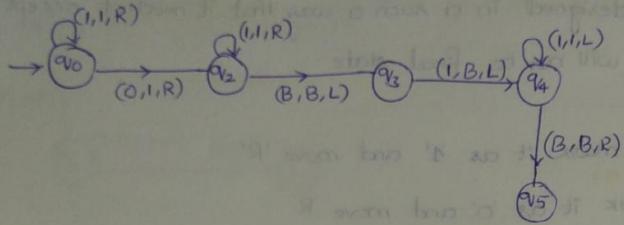
⇒ Now, the Remaining bits to the Left of bits should be complemented.

### TM AS AN ADDER (OF UNARY NUMBERS)

$$③ = (III), \Rightarrow |x|=3$$

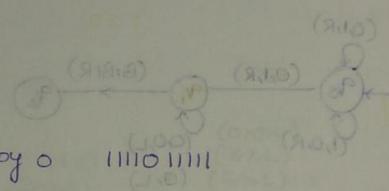
$$④ = (IIII), \Rightarrow |y|=4$$

and both  $x, y$  are separated by '0'

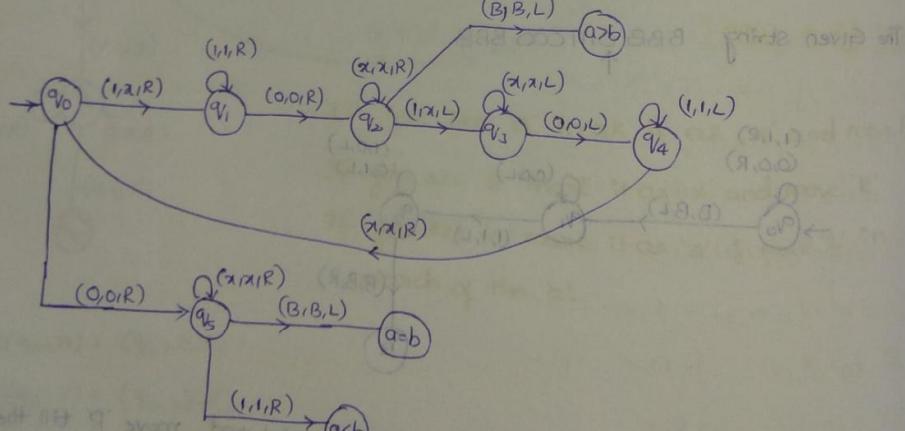


### TM AS A COMPARATOR

$$\begin{aligned} a = 4 &= (III), \\ b = 5 &= (IIII), \end{aligned} \quad \left. \begin{aligned} a, b \text{ are separated by } 0 \\ \{ \end{aligned} \right.$$



### TM AS A COMPARATOR



TM can perform any mathematical function

### THE STANDARD

$$M = (Q, \Sigma, \Gamma, \delta, q_0, F)$$

$Q$  = Set of states

$\Sigma$  = Input Alphabet

$\Gamma$  = Tape Alphabet

$\delta$  = Transition Function

$q_0$  = Initial state

$B$  = is used to

$F$  = Set of final states

### Summary

→ Tape is unbounded

⇒ It is deterministic

⇒ No special rules

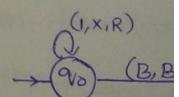
### TM AS A COPIER

Copier means that it copies the input. If the input is

→ Make all the

→ And now if you

→ Move Right



## THE STANDARD TURING MACHINE

M =  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Q = Set of states

$\Sigma$  = Input Alphabet

$\Gamma$  = Tape Alphabet

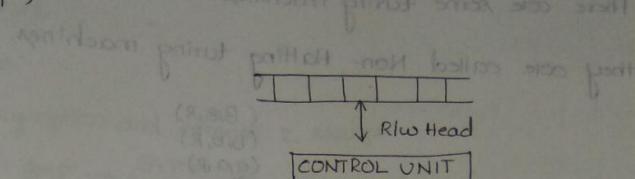
$\delta$  = Transition function

$q_0$  = Initial state ( $q_0 \in Q$ )

B = is used to Represent Blank ( $B \in \Gamma$ )

F = set of final states. ( $F \subseteq Q$ )

$$\delta : (Q \times \Gamma) \rightarrow (Q \times \Gamma \times \{L, R\})$$



### Summary

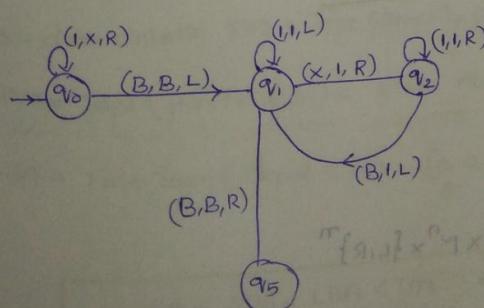
- 1) Tape is unbounded, so any number of Left and Right moves are possible
- 2) It is deterministic, i.e. at most one move for each configuration.
- 3) NO special i/p or o/p files

### 8) TM AS A COPIER

Copier means the Turing machine should produce the copy of the given input. If the input is B11B then the TM should produce BB111BB

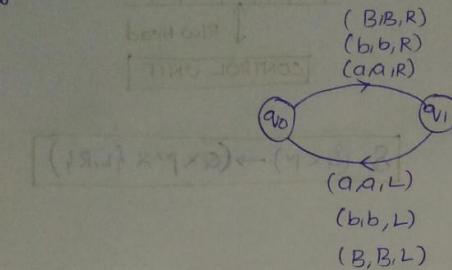
- Make all the 1's to 'x' and move 'Right'
- And now if you see a blank move left (1 position) and convert the x to 1
- Move Right and convert the next Blank to 1.

### IMPLEMENTATION OF STANDARD TURING MACHINE



### NON-HALTING TURING MACHINE

There are some turing machines that will never halt for some inputs they are called Non-Halting turing machines.



### TURING THESIS

Any computation that can be carried out by mechanical means can be performed by some Turing machine.

⇒ Anything that can be done by existing digital computer can also be done by TM.

⇒ No one has yet been able to suggest a problem, solvable by what we intuitively consider an algorithm, for which TM program cannot be written.

⇒ Alternative models have been proposed for mechanical computation, but none of them are more powerful than the Turing machine model.

i)

### MODIFICATIONS TO STANDARD TURING MACHINE

1) TM with stay option

2) TM with semi infinite tape

3) offline TM

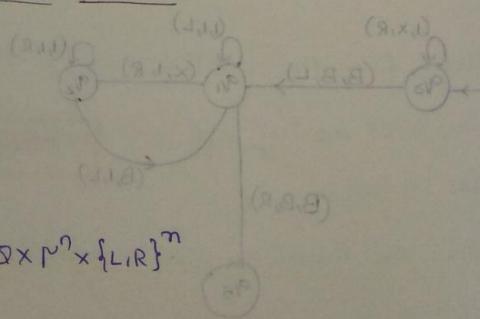
4) Multitape TM

$$\delta: Q \times \Sigma^m \rightarrow Q \times \Sigma^m \times \{L, R\}^m$$

5) Jumping TM

6) Non-Erasing TM

7) Always writing TM



8) Multidimensional

9) Multihead TM

10) Automata with

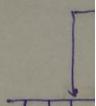
11) TM with only

12) Multitape TM

13) Non-determinist

14) A NPDA with

15) UNIVERSAL TM = A



(Rep of TM)

$$\delta(q_1, i) =$$

$$\delta(q_2, i) =$$

8) on

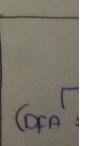
→ Every TM

### LINEAR BOUND

⇒ Non-determin

⇒ NTM + Tape

⇒ LBA = TM +



8) Multidimensional TM  $\delta: Q \times R \rightarrow Q \times R \times \{L, R, U, D\}$

9) Multicell TM

10) Automata with a Queue

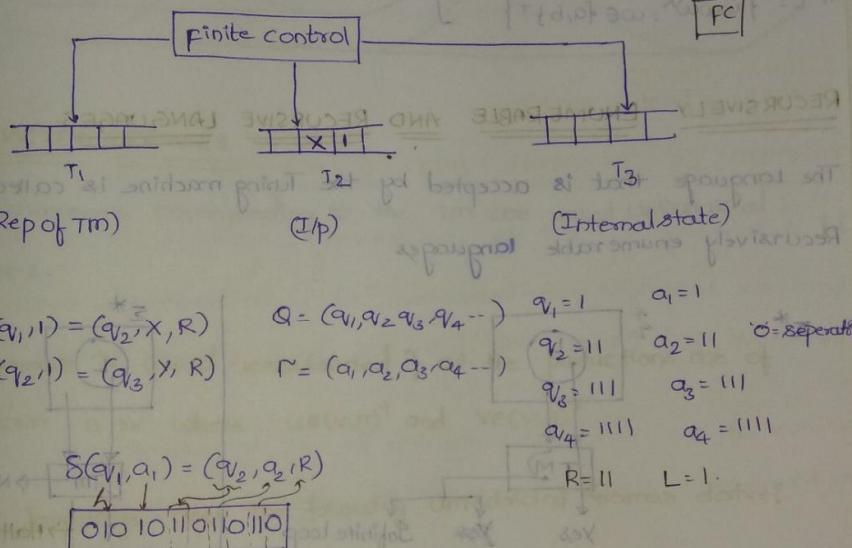
11) TM with only 3 states

12) Multitape TM with stay option and atmost 2 states

13) Non-deterministic TM  $\delta: Q \times N \rightarrow {}_2^{Q \times R \times \{L, R\}}$

14) A NPDA with 2 independent stacks  $\delta: Q \times (\Sigma \cup \{\epsilon\}) \times N \times N \rightarrow {}_2^{Q \times N^* \times N^*}$

### UNIVERSAL TM AND ENCODING TM



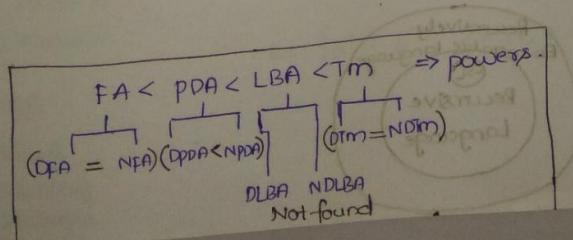
→ Every "TM" can be represented as the string of 0's and 1's.

### LINEAR BOUN DEP AUTOMATA

⇒ Non-deterministic TM + Tape (stack) = PDA

⇒ NTM + Tape (Finite size) = Finite Automata

⇒ LBA = TM + Tape (Input size)  $[a \ a \ b \ b]$



The some of the Languages that are accepted by the LBA are

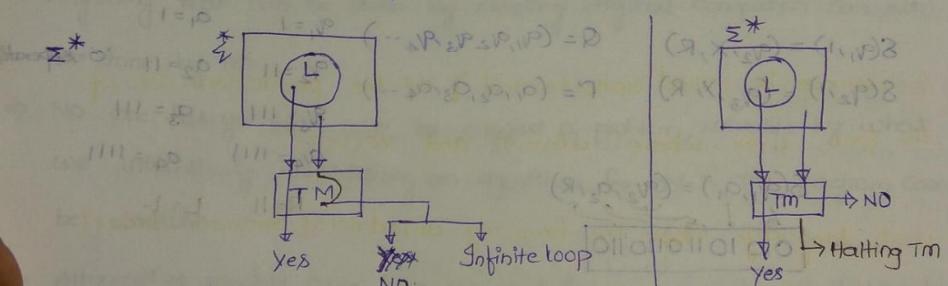
- 1)  $L = \{a^n b^n c^n : n \geq 1\}$
  - 2)  $L = \{a^n : n \geq 0\}$
  - 3)  $L = \{a^n : n = m^2, m \geq 1\}$
  - 4)  $L = \{a^n : n \text{ is prime}$
  - 5)  $L = \{a^n / n \text{ is not prime}\}$
  - 6)  $L = \{ww : w \in (a+b)^*\}$
  - 7)  $L = \{w^n : w \in (a+b)^*, n \geq 1\}$
  - 8)  $L = \{www^R : w \in \{a,b\}^*\}$

## Context Sensitive Languages

# THE BIG PIC

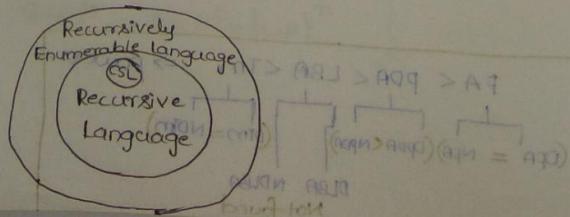
## RECURSIVELY ENUMERABLE AND RECURSIVE LANGUAGES.

The language that is accepted by the Turing machine is called  
(State transition) (qD) (or fog*s*)  
Recursively enumerable languages



⇒ If there exist a turing machine for a language then the language is called Recursively Enumerable language.

⇒ If there exists a halting TM for a language then the language is called Recursive language



① what lar

$$S \rightarrow S, \beta$$

$s_1 \rightarrow as, b$

$$b\beta \rightarrow b\bar{b}b\beta$$

$$as, b \rightarrow a_0$$

B → 2

so:  $\zeta \rightarrow \zeta$

20

9

$\Rightarrow$

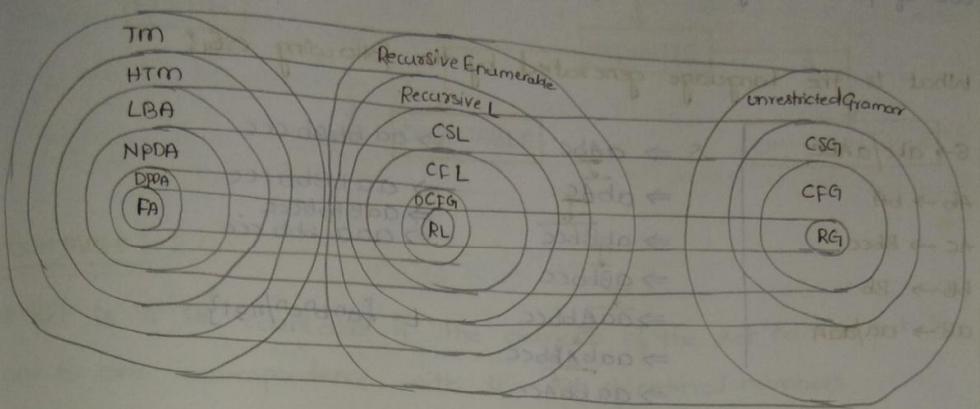
1

513

$\Rightarrow \text{as, bB}$

$\Rightarrow aa\ bbb[$

58 5.13



## UNRESTRICTED GRAMARS

⇒ The Grammars corresponding to the TM are called Unrestricted Grammars.

⇒ A Grammar is called Unrestricted if all the productions are of the form  $u \rightarrow v$  where  $u \in (VUT)^*$  and  $v \in (VUT)^*$

① what language does the following unrestricted Grammar derive?

$$S \rightarrow S_1 B$$

$$S_1 \rightarrow aS_1 b$$

$$bB \rightarrow bbbB$$

$aS, b \rightarrow aa$

$$B \rightarrow \lambda$$

S0) :  $S \Rightarrow S_1 B$

$\Rightarrow \alpha s, b \beta$

$\Rightarrow aaB$

$\Rightarrow \alpha\alpha$

5-115

→ as, BB  
→ as, LH

$$\Rightarrow aa\ bbbB$$

— 11 —

$$L = \{a^{n+1} b^{n+k} \mid n \geq 1, k = -1, 1, 3, 5, \dots\}$$

## CONTEXT SENSITIVE GRAMMAR - EXAMPLE 1

A Grammar is said to be context sensitive if all the productions are of form  $x \rightarrow y$  where  $(x,y) \in (VUT)^*$  and  $|x| \leq |y|$

What is the language generated by the following CSG?

$$S \rightarrow abc/aAbc$$

$$Ab \rightarrow bA$$

$$AC \rightarrow Bbcc$$

$$BB \rightarrow Bb$$

$$AB \rightarrow aa/aaa$$

$$S \Rightarrow a\overline{Abc}$$

$$\Rightarrow ab\overline{Ac}$$

$$\Rightarrow ab\overline{Bbcc}$$

$$\Rightarrow ab\overline{Bbcc}$$

$$\Rightarrow a\overline{a}abbcc$$

$$\Rightarrow aab\overline{Abbcc}$$

$$\Rightarrow aa\overline{bbAcc}$$

$$\Rightarrow aa\overline{bbBbcc}$$

$$\Rightarrow aa\overline{bBbcc}$$

$$\Rightarrow aa\overline{Bbcc}$$

$$\Rightarrow aaa\overline{bbbccc}$$

$$L = \{a^n b^n c^n / n \geq 1\}$$

## EXAMPLE-2 [Not in GATE]

### THEOREM ON RECURSIVE AND RE LANGUAGES

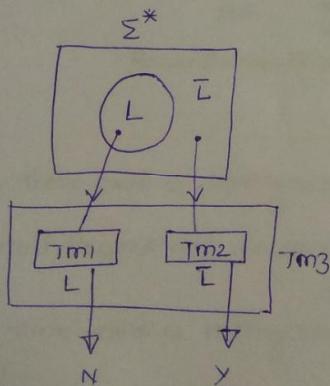
Recursively Enumerable

If a language 'L' and its complement  $\bar{L}$  are both RE, then

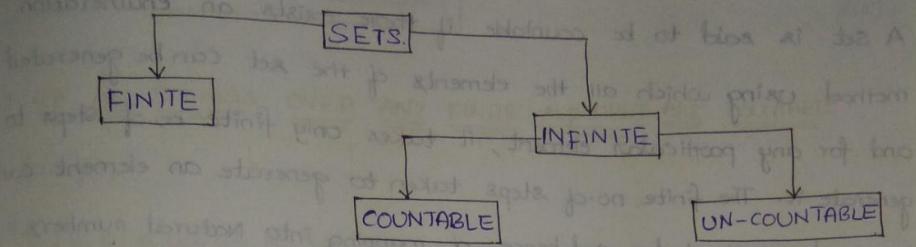
both the languages are Recursive. If 'L' is recursive, then  $\bar{L}$  is also recursive and consequently both are Recursively enumerable.

⇒ There is no membership algorithm for RE Languages.

⇒ Membership Algorithm exists for the Recursive Languages.



## COUNTABILITY



## COUNTABLE

A set is 'S' is countable if the elements of the set can be put in one to one correspondence with the set of natural numbers.

## UNCOUNTABLE

A set is uncountable if it is infinite and not countable.

one to one correspondence is there  
so countable

$$O = \{1, 3, 5, 7, 9, 11, \dots\} \quad N = \{1, 2, 3, 4, 5, 6, \dots\}$$

one-one correspondence is there so

countable

Real No's { 0 — 1 }

→ Infinite No's are present and could eat up all the Natural

Nos set so set of Real numbers is uncountable

→ There are some languages for which Turing machines cannot be constructed. ⇒ There are some languages which are not Recursively Constructed.

## Enumerable

## ALTERNATIVE DEFINITION OF COUNTABILITY

66

A set is said to be countable if there exists an enumeration method using which all the elements of the set can be generated and for any particular element, it takes only finite no. of steps to generate it. The finite no. of steps taken to generate an element can be used as its index and hence a mapping into Natural numbers.

Set of all Even numbers = for  $i=0$  to  $\infty$ )

{ starting at nos 2 & add 2 to all elements till infinite  
generate ( $2i$ ) }  
as per cue consecutive numbers with  
increasing position for the set above

0, 2, 4, 6, 8, ...

↓ ↓ ↓ ↓  
1 2 3 4 5 → index  
step steps

Set of all odd numbers = for  $i=0$  to  $\infty$ )

{ generate ( $2i+1$ ) }  
↓ ↓ ↓ ↓ ↓  
1 2 3 4 5 6 → index

Set of all Real numbers = No Enumeration method so NOT countable

SET OF ALL QUOTIENTS ARE COUNTABLE

Quotients of  $\frac{p}{q}$  where  $(p, q) \in \mathbb{Z}^+$

Location sit 1/2 gives blues line having 2/1 etc similar

$S = \left\{ \frac{1}{1}, \frac{1}{2}, \frac{2}{1}, \frac{1}{3}, \frac{1}{4}, \frac{3}{4}, \dots \right\}$

$= \left\{ \frac{1}{1}, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}, \dots \right\} \times \frac{1}{2}, \text{ will not be generated so this}$

Enumeration method is wrong.

so follow this enumeration method of  $(\frac{p}{q})$

> find the sum of  $(p+q)$  and go in increasing order of the sum

sum = ②      ③      ④      ⑤

$\left\{ \frac{1}{1}, \frac{2}{1} \right\} \left\{ \frac{1}{2}, \frac{2}{1} \right\} \left\{ \frac{1}{3}, \frac{2}{2}, \frac{3}{1} \right\} \left\{ \frac{1}{4}, \frac{2}{3}, \frac{3}{2}, \frac{4}{1} \right\} \dots \dots \dots$

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪

so Every No.  
set of all que

SET OF ALL

$S = \{a, b\}$

$S^* = \{\epsilon, a, b\}$

NOW the Er

> First Gene

all the st

the Lexico

$S = \{a, b,$

↓ ↓  
Step = {1 2

: SET OF  
ARE

Now, Another

Eve

represents  
for each i, j  
10000

base 10111

Now we

"Every L

66

so Every ' $\frac{p}{q}$ ' can be generated after finite no. of steps and therefore the

"Set of all Quotients are countable"

67

SET OF ALL STRINGS OVER ANY FINITE ALPHABET ARE COUNTABLE

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

$\Sigma^* = \{e, a, b, aa, ab, ba, bb, \dots\}$

Now the Enumeration method that must be followed is

1) First Generate all the strings of length 0( $e$ ) and now generate

all the strings of length 1  $\{a, b\}$  and in the set of length 1 follow the Lexicographic order.

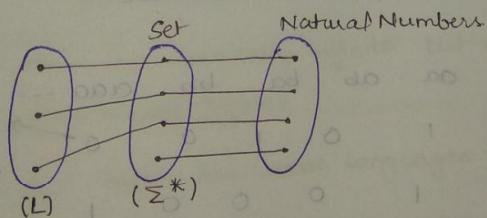
$S = \{e, a, b, aa, ab, ba, bb, aaa, aab, \dots\}$

Step = {1 2 3 4 ...}

$\therefore$  SET OF ALL POSSIBLE STRINGS OVER ANY FINITE ALPHABET SET ARE COUNTABLE, i.e.  $\Sigma^*$  IS COUNTABLE

Now, Another important property is.

Every subset of countable set is either finite or countable



Now we know that  $\Sigma^*$  is countable and  $L \subseteq \Sigma^*$  therefore

"Every Language is countable" → NOT ALL LANGUAGES

GIVEN A LANGUAGE ITS STRING CAN BE PUT IN ONE-ONE CORRESPONDANCE OF NATURAL NO'S.

## SET OF ALL TURING MACHINES ARE COUNTABLE

$\Sigma = \{0, 1\}$  (1)  $\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, \dots\}$  = countable

(2) Every TM can be encoded as a strings of 0s and 1s

<3> Set of all TMs ( $S$ )  $\{S \subseteq \Sigma^*\}$

$\Leftrightarrow$  Every subset of countable set is finite or countable

$\therefore$  SET OF ALL TM'S ARE COUNTABLE

## IMPLICATIONS OF THE FACT THAT THE TURING MACHINES ARE COUNTABLE

TM's are countable

$\Rightarrow$  Recursively Enumerable Languages are Countable

$\Rightarrow$  Recursive Languages are countable

$\Rightarrow$  CSL are countable  $\Rightarrow$  LBA are countable

$\Rightarrow$  CFL are countable  $\Rightarrow$  PDA are countable

$\Rightarrow$  RL are countable  $\Rightarrow$  Finite Automata are countable

## DIAGNOLIZATION METHOD TO PROVE SET OF ALL LANGUAGES ARE UNCOUNTABLE

### UNCOUNTABLE

$\Sigma = \{a, b\}$   $\Sigma^*$  is countable,  $\mathcal{L}$  is uncountable

$\Sigma^*$	$\epsilon$	a	b	aa	ab	ba	bb	aaa	...
1)	0	1	1	1	0	0	1	0	0
2)	0	1	0	1	0	0	0	0	1
3)	1	0	1	1	1	0	1	0	0
4)	1	0	0	0	0	0	0	0	1
5)	0	1	1	0	0	0	1	1	0
6)	1	0	1	0	0	1	0	1	0
7)	1	1	1	1	0	0	0	0	1
8)	0	1	1	1	1	0	1	1	1

Represent ele in power set :: binary

0 1 1 0 0 0 0 1

↓ Comp. ment

1 0 0 1 1 1 0

↓

Not in the language

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

Set of Languages possible are uncountable

69

All TMs are countable but all the Languages possible are uncountable

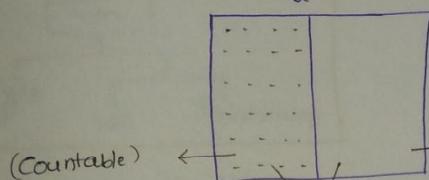
There exist some languages which a Turing machine cannot accept.

### SOME PROBLEMS ON COUNTABILITY

⇒ If  $S_1$  and  $S_2$  are countable sets, then  $S_1 \cup S_2$  is countable  
and  $S_1 \times S_2$  is countable

⇒ The cartesian product of finite no of countable sets is countable

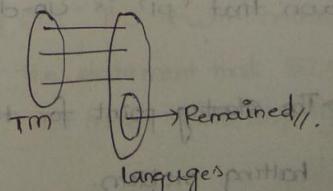
⇒ The set of all Languages that are not recursively Enumerable is uncountable.



→ If 'S' is countably infinite then  $\Sigma^*$  is uncountable  
put  $S = \Sigma^*$  → If  $\Sigma^*$  is countably infinite then  $\Sigma^*$  (set of all languages)  
are uncountable (uncountably infinite)

Now, TM - are countably infinite but all languages are uncountably infinite

⇒ There are some languages for which TM does not exist.



## COMPUTABILITY AND DECIDABILITY

Computability      and      Decidability

function

$$f(n) = (n^2 + 1)$$

Algo (on TM (halts) ✓  
Tape)

Decidability  
Problem

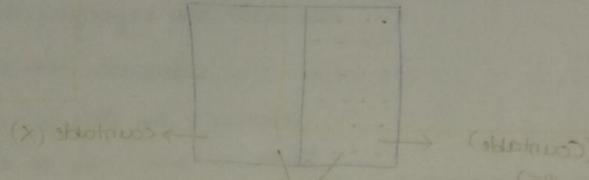
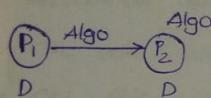
A statement whose answer is either True or False then it is decidable.

Ex: If  $f(n)$  is a prime

D = set of all Natural nos

Ex: A given Grammar  $G'$  is Ambiguous  
Domain = {set of all CFGs}

Given an instance of a problem, it is always decidable



Given a problem 'p1' and we can convert into the problem 'p2' using an Algorithm and the problem 'p2' can be solved using another Algorithm, which means that there exists an Algorithm for problem 'p1'. The conversion should be in such a way that the solution to 'p2' should also be a solution to p1. which means if the problem p2 is True then the solution to p1 should also be True.

And the procedure of conversion is called "Reducibility". If you find out that if 'p2' is decidable then 'p1' is also decidable. But before it is known that 'p1' is un-decidable then the problem 'p2' is undecidable

→ The starting point for the problem of decidability is the halting problem.

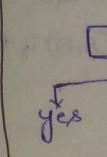
TM - HALTING

Given the d  
started with  
halts.

Theorem  
if the halting  
be Recursiv

Recursively

TM

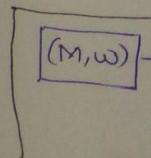


⇒ No member

Consider a

$\Sigma^*$

L = RE  
TM = m



## TM - HALTING PROBLEM

Given the description of a TM ' $m$ ' and an input ' $w$ ', does ' $m$ ' when started with ' $w$ ', does ' $m$ ' when started with ' $w$ ' as its input eventually halts.

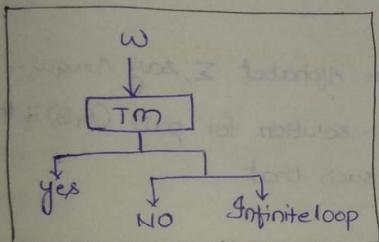
### Theorem

If the halting problem were decidable, then every RE language would be Recursive. Consequently, the halting problem is undecidable.

### Recursively Enumerable

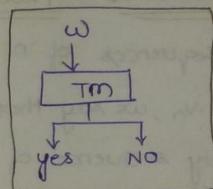
### Recursive language

TM



$\Rightarrow$  No membership Algo

TM which halts



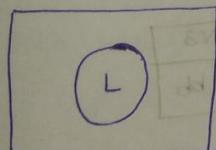
$\Rightarrow$  Membership Algo exists

Consider a Language ' $L$ ' which is Recursively Enumerable.

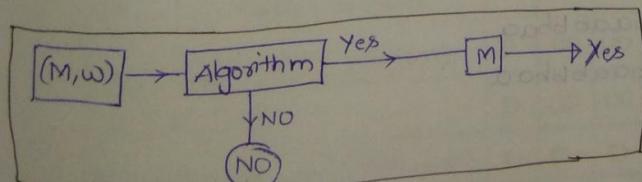
(No halting TM exists)

$\Sigma^*$

$L = \text{RE}$   
 $TM = m$



$\therefore$  This is the membership Algo we got for the "REL" but it will contradict the statement that "REL do not have Membership Algo".



## SOME UNDECIDABLE PROBLEMS BASED ON TM HALTING PROBLEM

1) The state entry problem is, given a TM, a state  $q \in Q$  and  $w \in \Sigma^*$ , decide whether or not the state  $q'$  is ever entered when 'M' is applied to 'w'. This is undecidable.

2) Given a TM M, whether or not M halts if started with a blank space. This is undecidable.

3) Almost any problem related to RE languages is undecidable.

## POST CORRESPONDENCE PROBLEM

Given two sequences of 'n' strings on some Alphabet  $\Sigma$ , say  $A = w_1 w_2 \dots w_n$  and  $B = v_1 v_2 \dots v_n$ , we say there exists a pc-solution for pair  $(A, B)$  if there is a non empty sequence of integers  $i, j, k$  such that

$$w_i w_{i+1} \dots w_k = v_j v_{j+1} \dots v_k$$

pc-problem is to device an algorithm that will tell us for any  $(A, B)$ , whether or not there exist a pc-solution.

Ex:

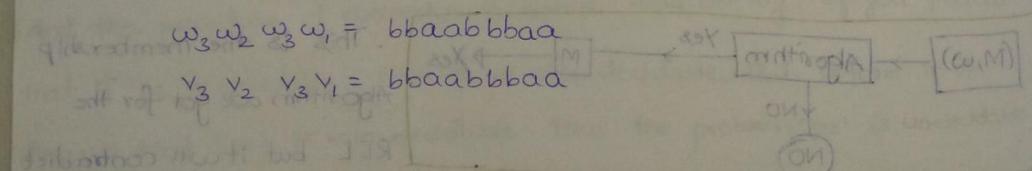
$w_1$	$w_2$	$w_3$
a	ab	bba

$v_1$	$v_2$	$v_3$
baa	aa	bb

The post correspondence solution is (3, 2, 3, 1)

$$w_3 w_2 w_3 w_1 = bbaabbbbaa$$

$$v_3 v_2 v_3 v_1 = bbaabbbbaa$$



## COMPLEXITY

### P-CLASS

The set of TM in polytime

### NP-CLASS

The set of problems whose solutions can be verified in polytime

$$(a \cup b \cup c)$$

$$2 \times 2 \times 2$$

$$= (2^n)$$

$$= O(2^n)$$

## DECIDABILITY

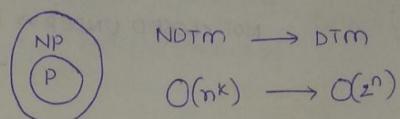
## COMPLEXITY CLASSES

### P-CLASS

The set of all languages that are accepted by some deterministic TM in polynomial time.

### NP-CLASS

The set of all languages accepted by non-deterministic TM in polynomial time is called NP-class.

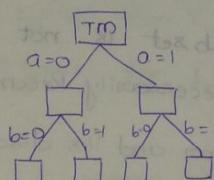


(avbvc)

$2 \times 2 \times 2$

$= \Omega(2^n)$

$= O(2^n)$



## DECIDABILITY TABLE

PROBLEM	RL	DCFL	CFL	CSL	Recursive Language	REL
Does $w \in L$ ? (Membership Algo)	D	D	D	D	D	UD
Is $L = \emptyset$ ? (Emptiness problem)	D	D	D	UD	UD	UD
$L = \Sigma^*$ (Completeness problem)	D	UD	UD	UD	UD	UD
Is $L_1 = L_2$ (Equality problem)	D	UD	UD	UD	UD	UD
Is $L_1 \subseteq L_2$ (Subset problem)	D	UD	UD	UD	UD	UD
Is $L_1 \cap L_2 = \emptyset$	D	UD	UD	UD	UD	UD
Is $L$ finite or not? (Finiteness)	D	D	D	UD	UD	UD
Is complement of $L$ a language of same type or not?	D	D	D	UD	UD	UD
Is intersection of 2 languages of same type	D	UD	UD	UD	UD	UD
Is $L$ Regular Language	D	D	UD	UD	UD	UD

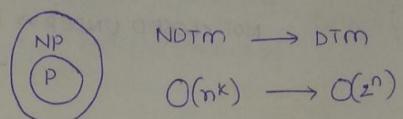
## COMPLEXITY CLASSES

### P-CLASS

The set of all languages that are accepted by some deterministic TM in polynomial time.

### NP-CLASS

The set of all languages accepted by non-deterministic TM in polynomial time is called NP-class.

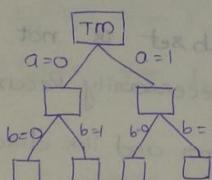


(avbvc)

$2 \times 2 \times 2$

$= \Omega(2^n)$

$= O(2^n)$



## DECIDABILITY TABLE

PROBLEM	RL	DCFL	CFL	CSL	Recursive Language	REL
Does $w \in L$ ? (Membership Algo)	D	D	D	D	D	UD
Is $L = \emptyset$ ? (Emptiness problem)	D	D	D	UD	UD	UD
$L = \Sigma^*$ (Completeness problem)	D	UD	UD	UD	UD	UD
Is $L_1 = L_2$ (Equality problem)	D	UD	UD	UD	UD	UD
Is $L_1 \subseteq L_2$ (Subset problem)	D	UD	UD	UD	UD	UD
Is $L_1 \cap L_2 = \emptyset$	D	UD	UD	UD	UD	UD
Is $L$ finite or not? (Finiteness)	D	D	D	UD	UD	UD
Is complement of $L$ a language of same type or not?	D	D	D	UD	UD	UD
Is intersection of 2 languages of same type	D	UD	UD	UD	UD	UD
Is $L$ Regular Language	D	D	UD	UD	UD	UD

74)  $L \Rightarrow$  closed under union, concatenation, closure

KLEEN  
PHISIM,  
SUBSTITUTION

	$L_1$	$L_2$	Concatenation	Closure
	$s_1 \rightarrow$	$s_2 \rightarrow$	$s_1 \rightarrow$ , $s_2 \rightarrow$	$L_1 \rightarrow$ , $L_1^*$
CFL	$\left\{ \begin{array}{l} S \rightarrow s_1 s_2 \\ s_1 \rightarrow \\ s_2 \rightarrow \end{array} \right.$		$S \rightarrow s_1 s_2$ , $s_1 \rightarrow$ , $s_2 \rightarrow$	$s_1 \rightarrow$ , $S \rightarrow s_1 s_1   \epsilon$
				CFL

Not closed under intersection  $\Rightarrow L_1 \cap L_2$

PLEMENTA-

$$L_1 = \{a^n b^n c^m / m, n \geq 0\}$$

$$L_2 = \{a^m b^n c^n / n, m \geq 0\}$$

$$L_1 \cap L_2 = \{a^n b^n c^n / n \geq 0\}$$

NOT CFL.

Not closed under complementation  $= \overline{L_1 \cap L_2} = \overline{L_1} \cup \overline{L_2}$

If we assume  $L_1 = \text{CFL}$  then  $\overline{L_1} = \text{CFL}$   
 If we assume  $L_2 = \text{CFL}$  then  $\overline{L_2} = \text{CFL}$

should be contentfree which  
 means  $L_1 \cap L_2$  should be CFL because but  
 we have seen that  $L_1 \cap L_2$  is not CFL. so our assumption is wrong  
 That the complementation of CFL is  $\therefore$  CFL is not closed under  
 complementation

15.

$\frac{L_1}{L_2}$

$\Rightarrow L_1 / L_2$  means right quotient of  $L_1$  with  $L_2$

$$L_1 / L_2 = \{x : xy \in L_1 \text{ for some } y \in L_2\}$$

$$L_1 = \{01, 001, 101, 0001, 1101\}$$

$$L_2 = \{01\}$$

$$L_1 / L_2 = \{\epsilon, 0, 1, 00, 11\} \Rightarrow \{\epsilon, 0, 1, 00, 11\}$$

$$\text{Now, } L_1 = 10^*$$

$$L_1 / L_2 = \{11, 101, 1001, 10001, \dots\}$$

$$L_2 = 0^*$$

$$L_1 / L_2 = \{1, 10, 100, 1000, \dots\}$$

$$L_1 / L_2 = 10^*$$

$$\frac{11}{0} = \emptyset, \frac{101}{0} = \emptyset$$

$\Rightarrow 11$  don't match with 10

→ Some of the decidable problems of the context free languages are  
Emptiness, finiteness, Membership

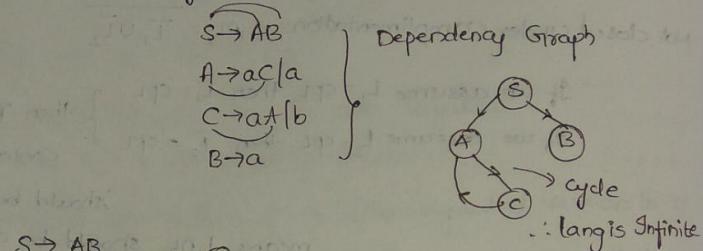
$$4. L_1^* \Rightarrow R_1 \text{ is } R_1^*$$

1> Membership- CYK Algorithm

2> Emptiness — If you find starting symbol is - useless  
then the lang is Empty.

3> Finiteness— Take Grammar 'G' and apply simplification  
and draw dependency Graph, If Graph has  
cycle then the lang is Infinite.

say, the simplified Grammar is



6. closed under

$$L_1 - L_2 \Rightarrow L_1$$

$L_1$  is regular language

↳ No cycle  $\Rightarrow$  the language is finite

### PROPERTIES OF REGULAR LANGUAGES

CLOSED UNDER UNION, INTERSECTION, CONCATENATION, COMPLEMENTATION

KLEN CLOSURE

1>  $L_1 \cup L_2 \Rightarrow L_1$  has Regular expression  $R_1$

$L_2$  has Regular expression  $R_2$

$$\therefore L_1 \cup L_2 = [R_1 + R_2] \in RE \text{ (Regular language)}$$

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

$h(a) =$

$h(b) =$

$$2> L_1 \cap L_2 \Rightarrow L_1 \rightarrow R_1 \quad L_1 \cap L_2 = [R_1 \cdot R_2] = \text{Regular Language}$$

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

$$3. L_1 \cdot L_2 =$$

→ Some of the decidable problems of the context free languages are  
Emptiness, finiteness, Membership

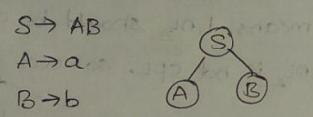
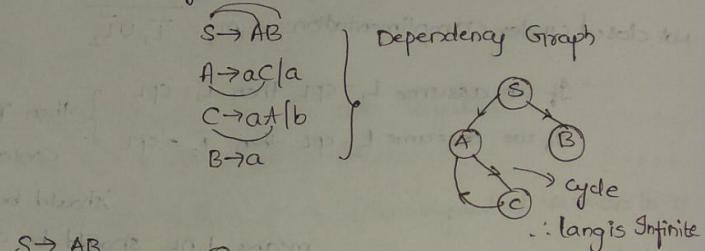
$$4. L_1^* \Rightarrow R_1 \text{ is } R_1^*$$

1> Membership- CYK Algorithm

2> Emptiness — If you find starting symbol is - useless  
then the lang is Empty.

3> Finiteness— Take Grammar 'G' and apply simplification  
and draw dependency Graph, If Graph has  
cycle then the lang is Infinite.

say, the simplified Grammar is



### PROPERTIES OF REGULAR LANGUAGES

CLOSED UNDER UNION, INTERSECTION, CONCATENATION, COMPLEMENTATION

KLEN CLOSURE

1>  $L_1 \cup L_2 \Rightarrow L_1$  has Regular expression  $R_1$

$L_2$  has Regular expression  $R_2$

$$\therefore L_1 \cup L_2 = [R_1 + R_2] \in RE \text{ (Regular language)}$$

2>  $L_1 \cap L_2 \Rightarrow L_1 \rightarrow R_1$        $L_1 \cap L_2 = [R_1 \cdot R_2] = \text{Regular Language}$

$$3. L_1 \cdot L_2 =$$

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

$$h(a) =$$

$$h(b) =$$

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

$L_i^* \Rightarrow R_i$  is the RE for  $L_i$

(76)

$R_i^*$  is the RE of  $L_i^*$   $\therefore R_i^*$  is Regular Expression

(77)

$\therefore$  Regular languages are closed under closure

5.  $\bar{L}_i = \Sigma^* - L_i \Rightarrow$  Language  $L_i$  has DFA [Given  $L_i$  is Regular]

$\Rightarrow$  Complement the DFA then we get DFA accepting  $\bar{L}_i$

$\Rightarrow$  Since  $L_i$  has DFA it is Regular

6. closed under difference and Reversal

$$L_1 - L_2 \Rightarrow L_1 - L_2 \text{ can be written as } L_1 - L_2 = L_1 \cap \bar{L}_2$$

$\downarrow$        $\downarrow$        $\downarrow$   
 $= RL \cap RL$   
 $= RL$



$L_i$  is regular language  $\Rightarrow L_i^R$  should be Regular

$\therefore L_i \rightarrow$  has DFA

$\Rightarrow$  Reverse the DFA then we get DFA accepting Reverse of the language

7. closed under Homomorphism and Inverse Homomorphism

$$\Sigma = \{a, b\} \quad N = \{0, 1, 2\}$$

$$h(a) = 01, \quad h(b) = 112$$

$$\text{Let } L = a^*b$$

$$= (01)^* 112 \Rightarrow \text{Regular Expression}$$

$\therefore$  closed homomorphism

$$\Sigma = \{a, b\} \quad N = \{0, 1, 2\}$$

$$\text{Let } h = \{ababa\}$$

$$\frac{\underline{ababab}}{1 \quad 1 \quad 0}$$

$$p(0) = a$$

$$p(1) = ab$$

$$p(2) = ba$$

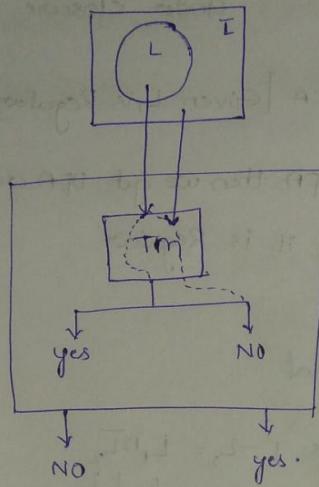
$$h^{-1}(h) = \{110022, 102\}$$

$$\frac{\underline{ababab}}{1}$$

8. NOT CLOSED UNDER INFINITE UNION.

→ The complement of Recursive language is Recursive

Say there is a Recursive Language then it means there is a TM



This TM just works in opposite way, which means if the inner TM says NO (the present in L) string is not then the outer TM says Yes the string is present in L and viceversa.

∴ we got a halting TM that halts for L also. ∴ THE COMPLEMENT OF  
RECURSIVE LANGUAGE IS RECURSIVE.