COMPUTER SCIENCE

TOC

1. DFA

INTRODUCTION:

TOC: Any task that can be performed by calculator or Computer is called Computation.

SYMBOL: Symbol is the basic building block of TOC.

EX: - a, b, 0, 1, --- 🧍, 🧍

"Σ" **ALPHABET:** Some collection of Symbols

EX: - {a, b}
- {a, b, c}, {0, 1}, {0, 1, 2, 3 --- 9}

STRING: String is Sequence of Symbols.

$|\Sigma|^n$ EX: a, b, aa, ab, bc ------
1 1 2 2 2

LANGUAGE: Collection of Strings.

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

L_1 = Set of all strings of length 2.
= {aa, ab, ba, bb} → FINITE

L_2 = Set of all strings of length 3.
= {aaa, aab, aba, abb, baa, bab, bba, bbb} → FINITE

L_3 = Set of all strings where each string starts with 'a'.
= {a, aa, ab, aaa, aab, aba, abb --- } → INFINITE

NOTE: A language which can be formed over a 'Σ' (alphabet) can be Finite or INFinite.

POWER OF Σ: $\Sigma = \{a, b\}$

Σ^1 = Set of all strings over Σ of length '1'

$= \{a, b\}$

$\Sigma^2 = \Sigma \cdot \Sigma = \{a,b\}\{a,b\} = \{aa, ab, ba, bb\}$

Set of all strings of length '2'

$\Sigma^3 = \Sigma \cdot \Sigma \cdot \Sigma =$ -----

Set of all strings of length '3'

$|\Sigma^3| = 8$.

Σ^n = 'n' length string

NOTE: $\Sigma^0 = \{\epsilon\} \Rightarrow |\epsilon| = 0$

Σ^* : In place of '*' we can substitute any number starting from '0'

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2$ ----- INFINITE

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

Σ^*

L_1 L_2
L_3, L_4 -

$L_1 \subseteq \Sigma^*$
$L_2 \subseteq \Sigma^*$
$L_3 \subseteq \Sigma^*$
⋮

'C':

$\Sigma = \{a, b, \ldots z, A, B \ldots Z, 0, 1, \ldots 9, +, *, \ldots\}$

program in 'C' = String in ToC

```
valid main()
{
  int a,b
  :
}
```

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

L is finite

$L_1 = \{aa, ab, ba, bb\}$

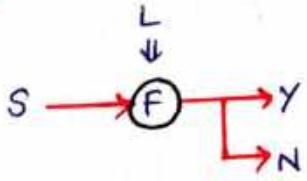
$S = aaa$ ✗

$\Sigma = \{a\}$

L is infinite

$L_1 = \{a, aa, aaa, \cdots\}$

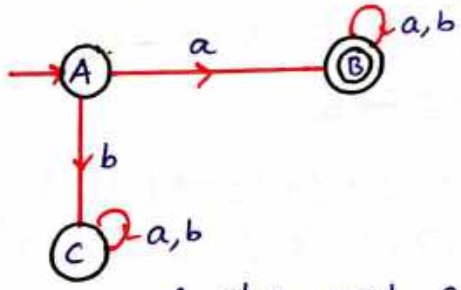
$S = baba$ ✗



EX: L_1 = Set of all strings which starts with 'a'

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

FA:



String : a a b

$A \xrightarrow{a} B \xrightarrow{a} B \xrightarrow{b} (B)$ FINAL STATE

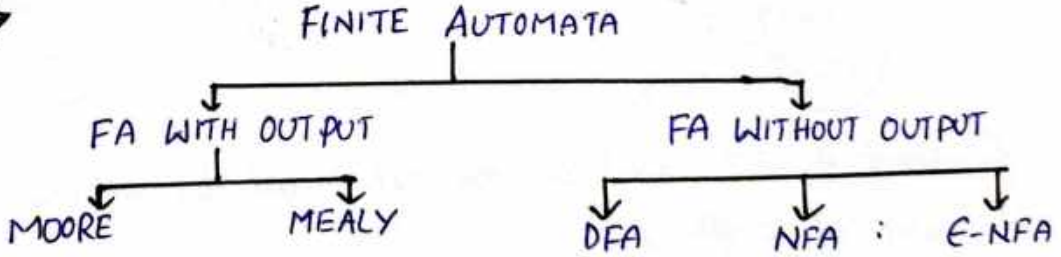
∴ String aab accepted by 'FA'

String : b b a

$A \xrightarrow{b} C \xrightarrow{b} C \xrightarrow{a} (C)$ NOT FINAL STATE

∴ String bba Not accepted by 'FA'

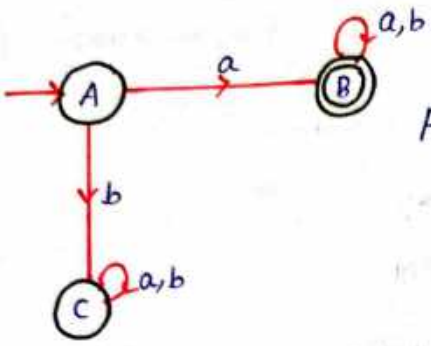
→



DFA : DETERMINISTIC FINITE AUTOMATION

NFA : NON-DETERMINISTIC FINITE AUTOMATION.

DFA: $(Q, \Sigma, \delta, q_0, F)$



$Q = \{A, B, C\}$
Final Set of all states

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

q_0 = start state - 'A'

F = Set of all final states - 'B'

$Q \supseteq F$

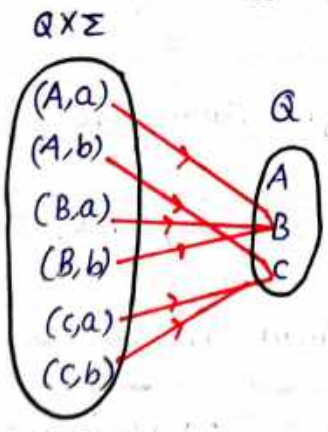
δ : is transaction function to $Q \times \Sigma \rightarrow Q$.

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

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

$$Q \times \Sigma = \{(A,a), (A,b), (B,a), (B,b), (C,a), (C,b)\}$$

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

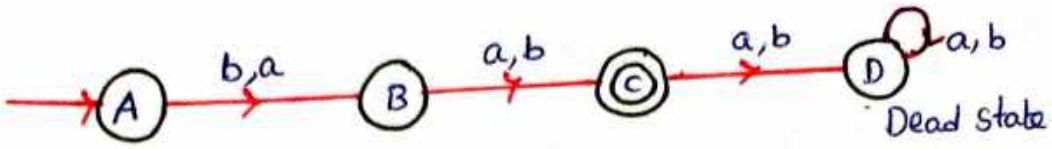


Ex: Construct a DFA, that accepts set of all strings over $\{a, b\}$ of length 2

→

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

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



'C' → accepting state.

A, B → NOT accepting states.

String accept: Scan the entire string, if we reach a final state from intial state.

Language accept: A finite automata is said to accept a language if all the strings in the language are accepted and all the strings not in the language are rejected.

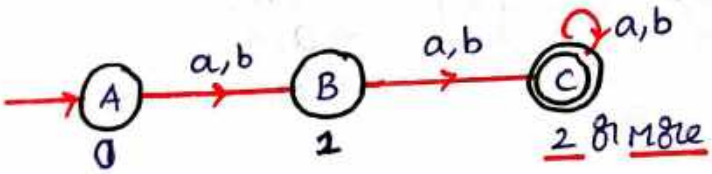
→ 

→ accepted by all strings

CONSTRUCTION OF MINIMAL DFA:

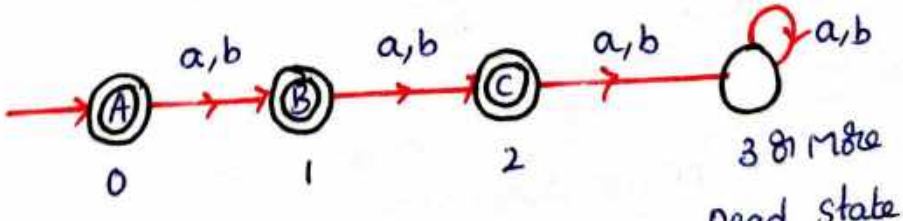
Ex: DFA $w \in \{a,b\}$ $|w| \geq 2$

→ $L = \{aa, ab, ba, bb, aaa \text{ --- } bbb \text{ --- }\}$ → FINITE



Ex: DFA with $w \in \{a,b\}$ $|w| \leq 2$

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



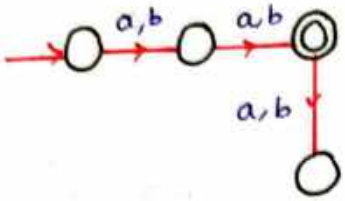
'ϵ' Intial state = Final state.

→ Many DFA's

→ '1' - Single Minimum 'DFA'

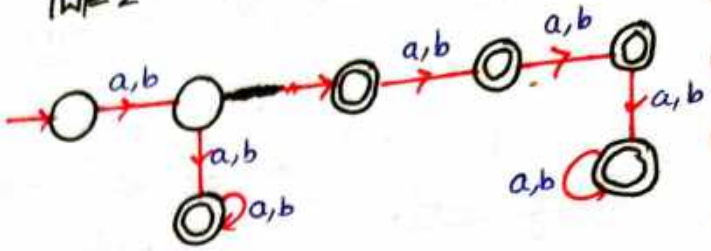
***

→ $|W| = 2$



$|W| = n$
Total No. of states
'n+2'

$|W| \geq 2$



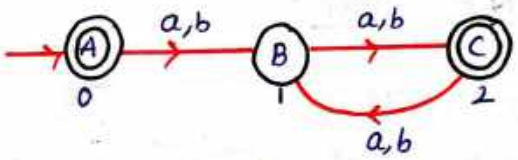
$|W| = n$
Total No. of states
'n+1'

$|W| \leq 2$

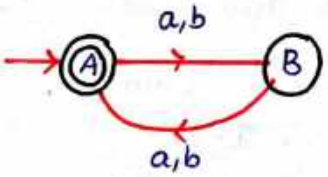
$|W| = n$
Total no of
state 'n+2'

Ex: Minimal DFA of $W \in \{a,b\}$ with $|W| \bmod 2 = '0'$

→ $L = \{ \otimes \rightarrow \epsilon, aa, ab, ba, bb, aaaa \text{----} bbbb \text{---} \}$



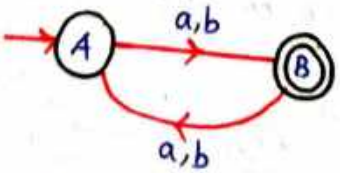
→ FA.
→ DFA.
→ NOT Minimal.



→ FA.
→ DFA.
→ Minimal DFA.

Ex: Minimal DFA of $W \in \{a,b\}$ with $|W| \bmod 2 = '1'$

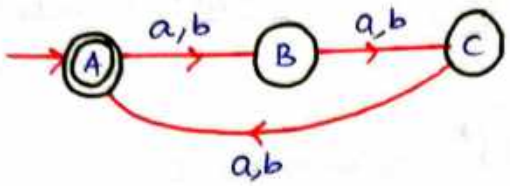
$L = \{ a, b, aaa, aba \text{---} bbb, aaaaa \text{-----} bbbbb \}$



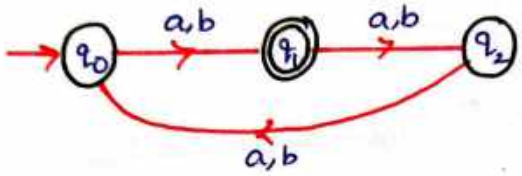
→ FINITE AUTOMATA.
→ DETERMINISTIC FINITE AUTOMATA.
→ MINIMAL DFA.

Ex: $w \in \{a,b\}^*$ (i) $|W| \bmod 3 = 0$

→ $L = \{ \epsilon, aaa \cdots b, bb, aaaaaa \cdots bbbbbb, \cdots \}$



(ii) $|W| \bmod 3 = 1$ (or) $|W| \cong \bmod 3$



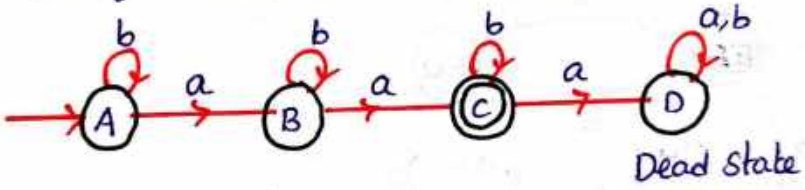
NOTE:

$|W| \bmod n = 0$

∴ There is 'n' no. of States.

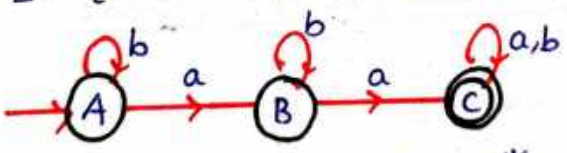
Ex: Minimal DFA $w \in \{a,b\}^*$ $n_a(w) = 2$

→ $L = \{aa, baa, aba, aab, bbaa, \cdots\}$



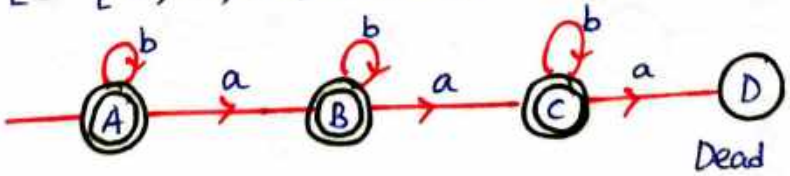
Ex: Minimal DFA $w \in \{a,b\}^*$ $n_a(w) \geq 2$

→ $L = \{aa, baa, aaab, aaaa \cdots\}$



Ex: Minimal DFA $w \in \{a,b\}^*$ $n_a(w) \leq 2$

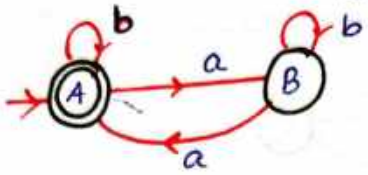
→ $L = \{ \epsilon, a, ab, aa, ba, \cdots \}$



Ex: Minimal DFA $\omega \in \{a,b\}^*$ $n_a(\omega) \bmod 2 = 0$

$n_a(\omega) \cong 0 \bmod 2$

→

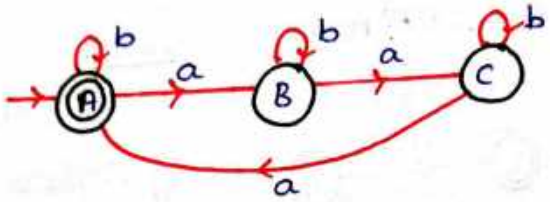


NOTE: $n_a(\omega) \cong K \bmod N$

'N' No. of States

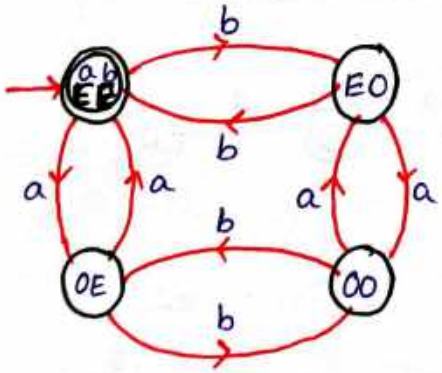
Ex: Minimal DFA $\omega \in \{a,b\}^*$ $n_a(\omega) \bmod 3 = 0$

→



EX: Minimal DFA $\omega \in \{a,b\}^*$ $n_a(\omega) \cong 0 \bmod 2$ && $n_b(\omega) \cong 0 \bmod 2$

→ $L = \{\epsilon, aa, bb, aabb, abab, baba, bbbb \text{----}\}$

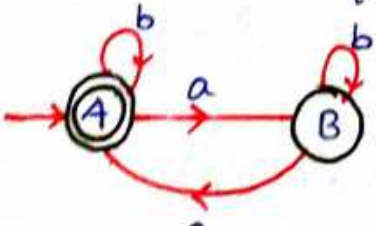


CONSTRUCTION OF DFA AND CROSS PRODUCT OF DFA:

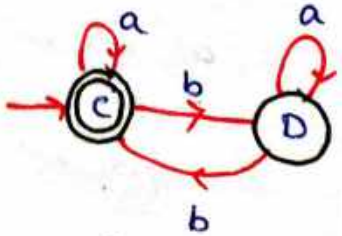
EX: $\omega \in (a,b)^*$ $n_a(\omega) \cong 0 \bmod 2$ && $n_b(\omega) \cong 0 \bmod 2$

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

'n' States

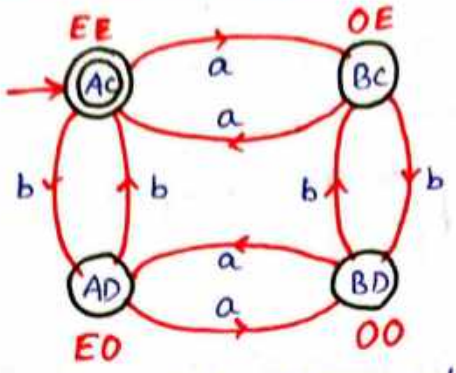


COUNTING a's



COUNTING b's

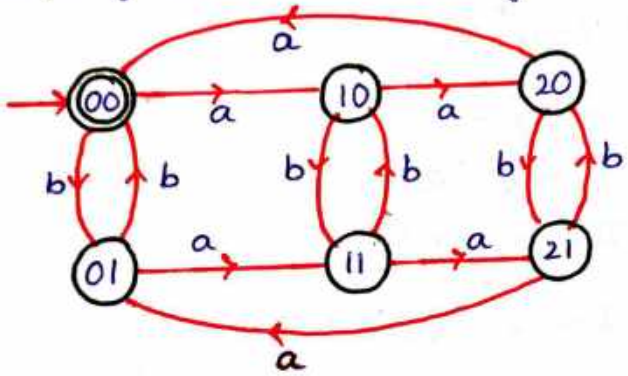
'm' States



Total
'n x m' States

Ex: Construct a minimal DFA which accepts set of all strings over {a, b} in which number of a's are divisible by 3 and number of b's are divisible by 2.

→



NOTE: → $n_a(\omega) \bmod 3 \neq n_b(\omega) \bmod 2$

{10, 20, 21, 11}

Final States.

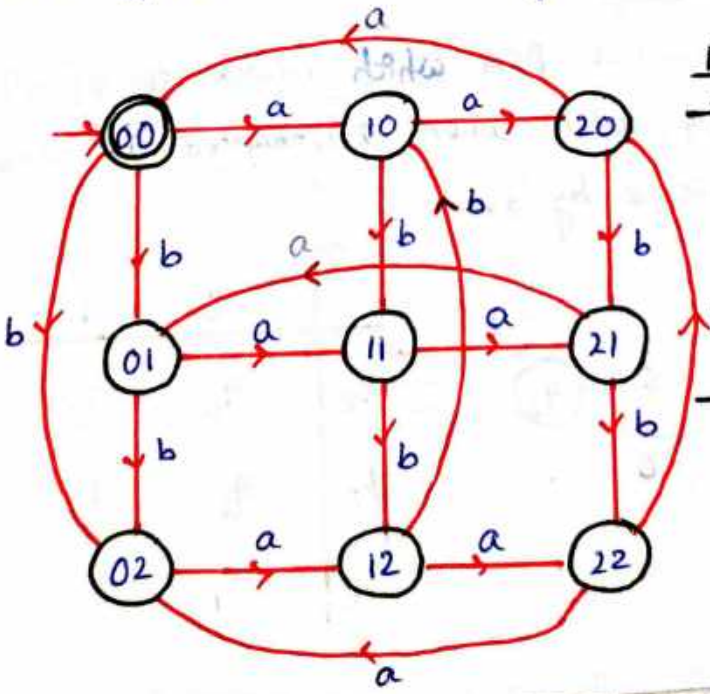
→ $n_a(\omega) \bmod 3 = n_b(\omega) \bmod 2$

{00, 11}

Final States.

Ex: Construct a Minimal DFA which accepts Set of all Strings over {a, b} in which number of a's are divisible by '3' and number of b's are divisible by '3'.

→



NOTE:

→ $n_a(\omega) \bmod 3 = 1$

&&

$n_b(\omega) \bmod 3 = 2$

{1, 2}

Final States

→ $n_a(\omega) \bmod 3 > n_b(\omega) \bmod 3$

{10, 20, 21}

Ex: Construct a Minimal DFA with which accepts set of all strings over {a,b} in which number of a's are divisible by 3 and number of b's are divisible by '3'.

→ Same as above Question.

Ex: Construct a Minimal DFA which accepts set of all strings over {0,1}, which when interpreted as binary number is divisible by '2'.

→ $\Sigma = \{0,1\}$ $w \in \{0,1\}^*$

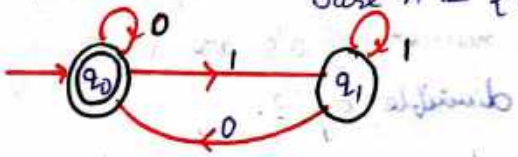
(10)
↙
1+2+0 = 2

(101)
↙
1+2+0 = 2
2×2+1 = 5

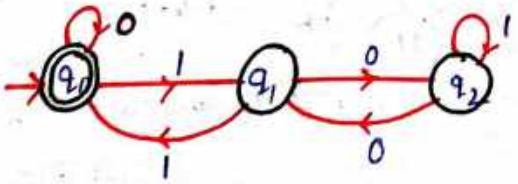
1 - unary - 0
2 - Binary - 0,1
3 - Ternary - 0,1,2
⋮
10 - Decimal - 0,1,2,3 -- 9

Base 3 - {0,1,2}

Base n - {0,1, ... n-1}



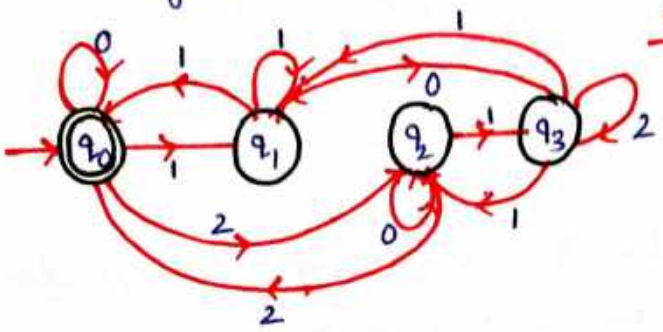
Ex: Construct a Minimal DFA which accepts set of all strings over {0,1}, which when interpreted as binary number is divisible by '3'.



	0	1
* → q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

Ex: Construct a minimal DFA, which accepts set of all strings over {0,1,2} which when interpreted as binary number is divisible by '4'.

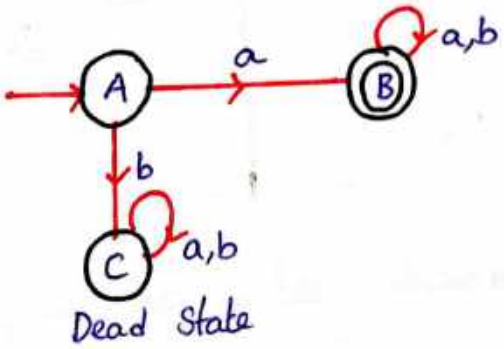
→



	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

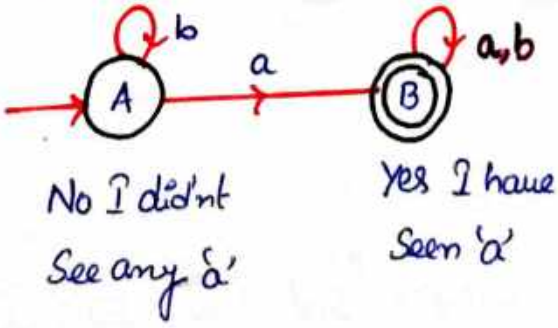
Ex: Construct a minimal DFA which accepts set of all strings over {a,b} where each string starts with an 'a'.

→ DFA $L = \{a, aa, ab, aaa, \text{-----}\}$



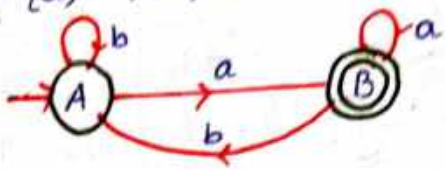
Ex: Construct a minimal DFA which accepts set of all strings over {a,b} where each string contains 'a'.

→ $L = \{a, aa, ab, ba, \text{-----}\}$



Ex: Construct a minimal DFA which accepts set of all strings over {a,b} where each string ends with an 'a'.

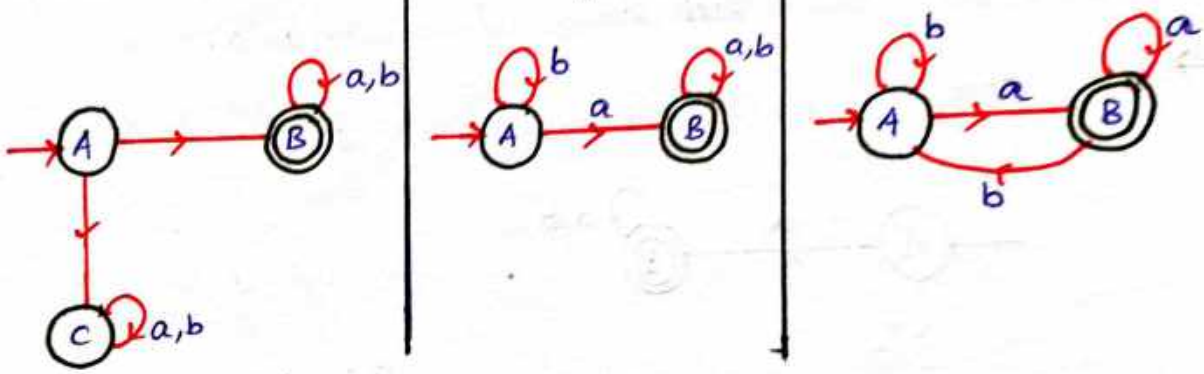
→ $L = \{a, aa, ba, aaa, aba, \text{----}\}$



Ex: Comparision between the DFAs that accept strings, starting with 'a' Containing 'a' and ending with 'a'.

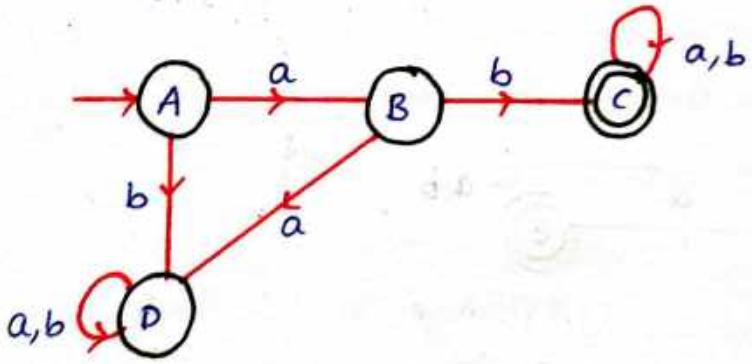
→

starts with 'a'	Containg 'a'	Ends with 'a'



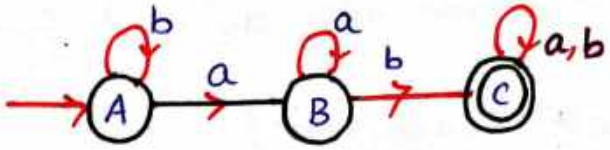
Ex: Construct a DFA which accepts set of all strings over {a,b} which start "ab".

→ $L = \{ab, aba, abb, \text{----}\}$



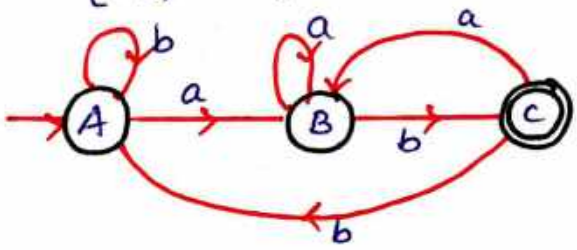
Ex: Construct a DFA which accepts set of all strings over {a,b} which Contains ab

→ $L = \{ab, aba, aba, bab, \text{----}\}$



Ex: Construct a DFA which accepts set of all strings over $\{a,b\}$ which ends with "ab".

→ $L = \{ab, aab, bab, abab, \text{----}\}$



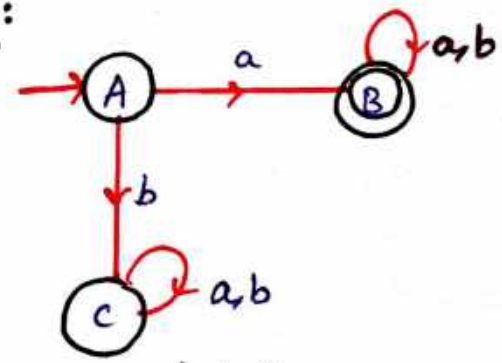
Ex: Construct a DFA which accepts set of all strings over $\{a,b\}$ which starts with 'a' and ends with 'b'.

→ $L_1 = \{a, ab, aa, aaa, \text{----}\}$

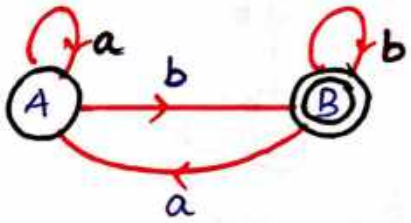
$L_2 = \{b, ab, bb, aab, \text{----}\}$

$L_1 \cdot L_2 = \{ab, aab, abb, \text{----}\}$

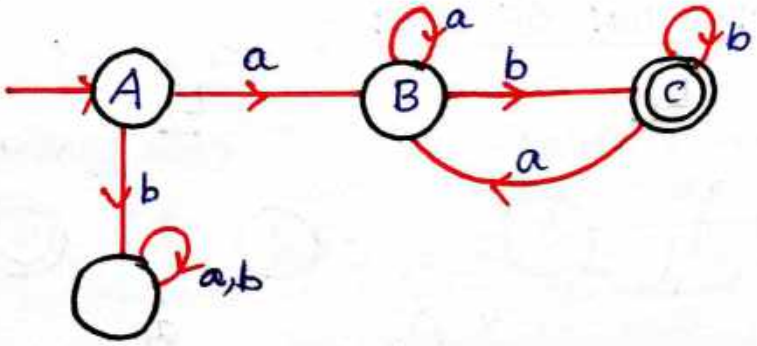
L_1:



L_2:

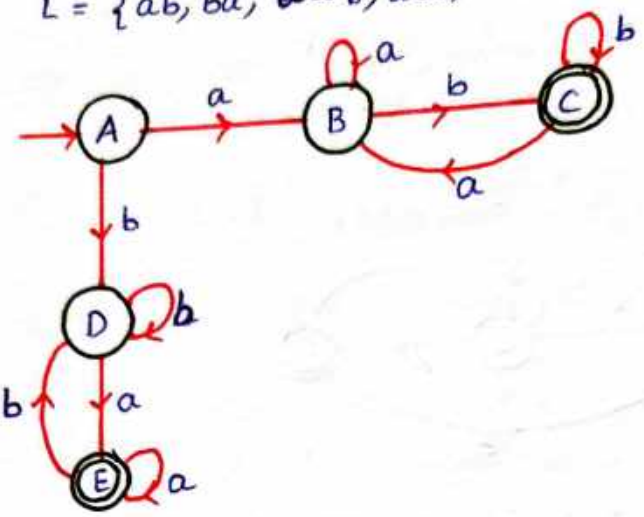


L_1L_2:



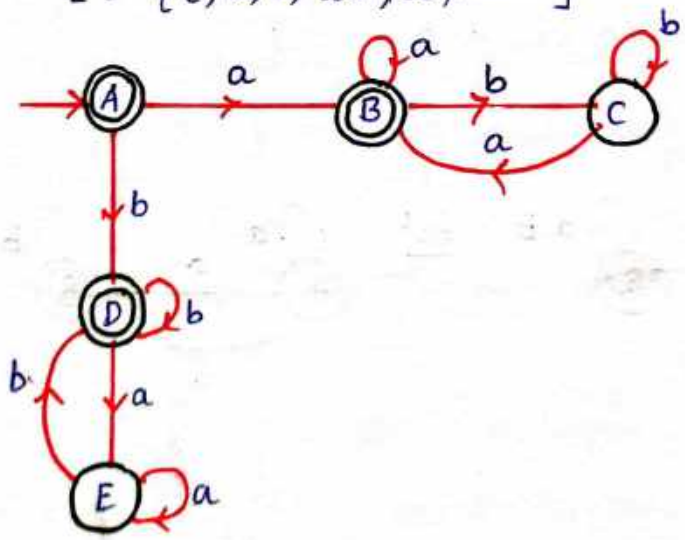
Ex: Construct a DFA which accepts set of all strings over {a,b} which starts and ends with different Symbol.

→ $L = \{ab, ba, aab, abb, \text{-----}\}$



Ex: Construct a DFA which accepts set of all strings over {a,b} which starts and ends with same Symbol.

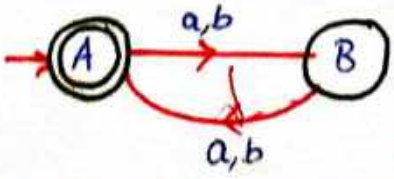
→ $L = \{\epsilon, a, b, aa, bb, \text{----}\}$



COMPLEMENTATION OF DFA:

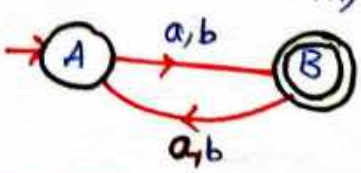
Ex:

L_1 (EVEN LENGTH)



$L_1 = \overline{L_2}$

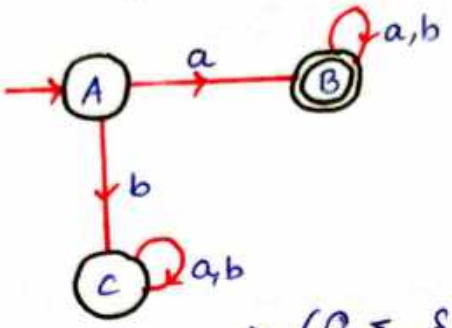
L_2 (ODD LENGTH)



Ex: Starting with 'a'

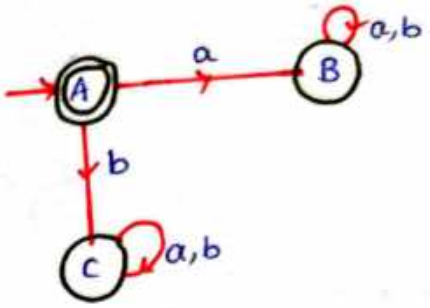
$L = \{a, ab, aa \cdots\}$

→



not starting with 'a'

$L = \{\epsilon, b, ba, \cdots\}$



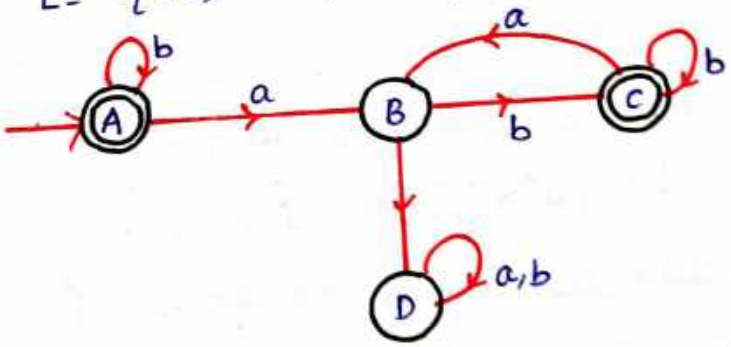
$\Rightarrow (Q, \Sigma, \delta, q_0, F)$

$\Downarrow$ COMPLEMENT

$(Q, \Sigma, \delta, q_0, Q-F)$

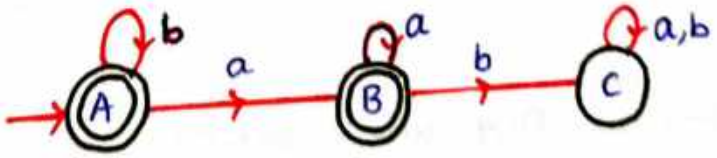
Ex: Construct a Minimal DFA which accepts set of all strings over $\{a,b\}$ in which every 'a' is followed by a 'b'.

→ $L = \{ab, abab, \cdots b, bb, \cdots\}$



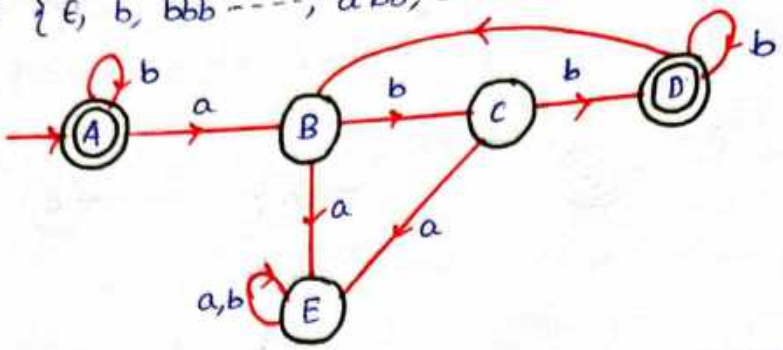
Ex: Construct a Minimal DFA which accepts set of all strings over $\{a,b\}$ in which every 'a' should never followed by a 'b'.

→ $L = \{\epsilon, a, aa, aaa, b, bb \cdots, bbba, ba \cdots\}$



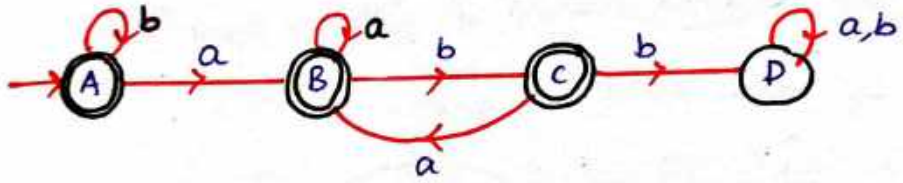
Ex: Construct a minimal DFA which accepts set of all strings over $\{a,b\}$ in which every 'a' should be followed by a 'bb'.

→ $L = \{\epsilon, b, bbb\text{---}, abb, bbabb, \text{-----}\}$



Ex: Construct a Minimal DFA which accepts set of all strings over {a,b} in which every 'a' should never be followed by a "bb".

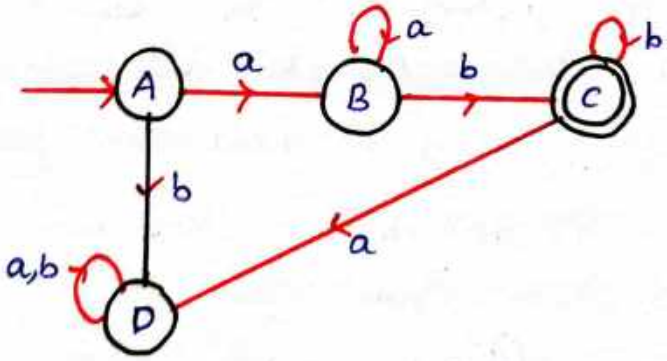
→ $L = \{\epsilon, b, bb, bbb, \text{---}, a, aa, aaa \text{---} ab, aab\}$



Ex: Construct a Minimal DFA which accepts

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

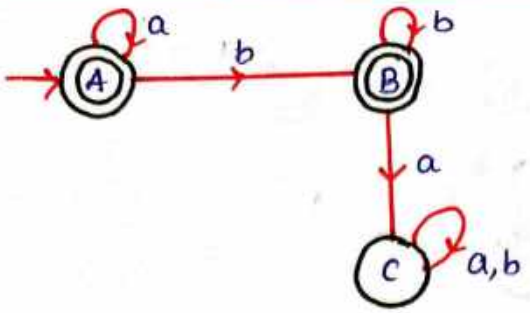
→ $L = \{ab, aab, abb, aabb, aaabb\text{----}\}$



Ex: Construct a Minimal DFA which accepts

$L = \{a^n b^m / n, m >= 0\}$

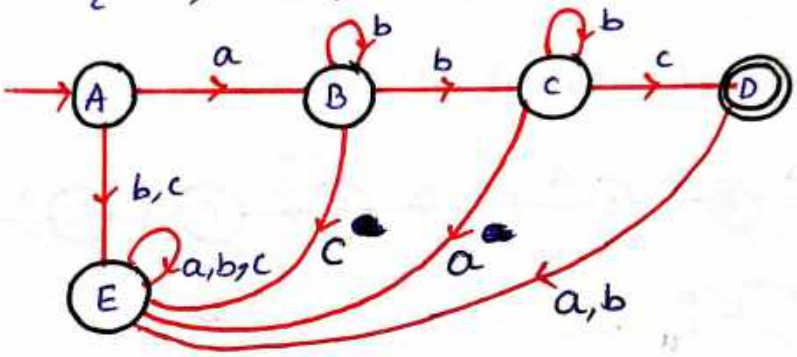
→ $L = \{\epsilon, a, aa, aaa \text{----}, b, bb, bbb\text{----}\}$



Ex: Construct a Minimal DFA which accepts

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

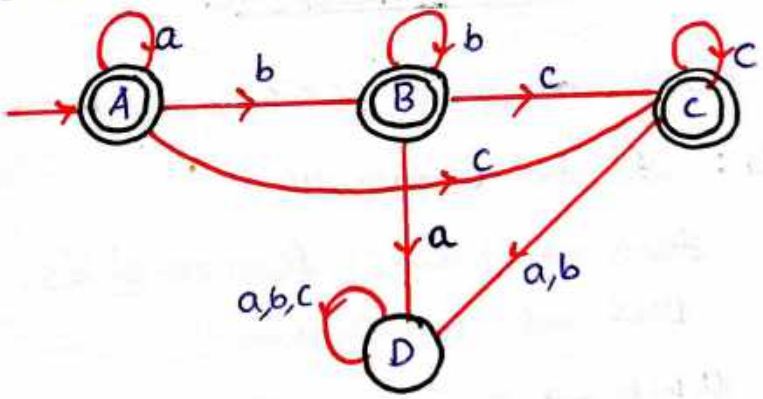
→ $L = \{abc, aabc, abbc, abcc \text{----} \}$



Ex: Construct a Minimal DFA which accepts

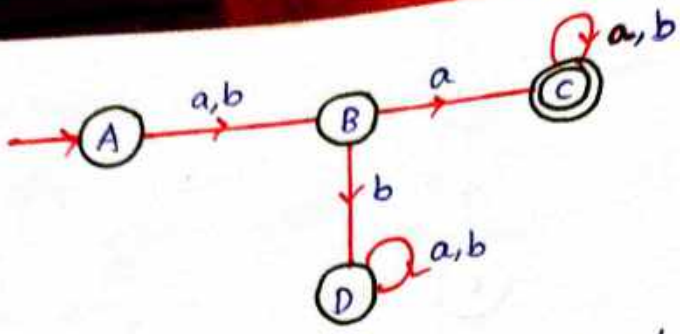
$$l = \{ a^n b^m c^l / n, m, l >= 0 \}$$

→ $L = \{ \epsilon, a, aaa \text{----} b, bb \text{----}, c, c, c \text{---} aabb, aacc \text{---} \}$



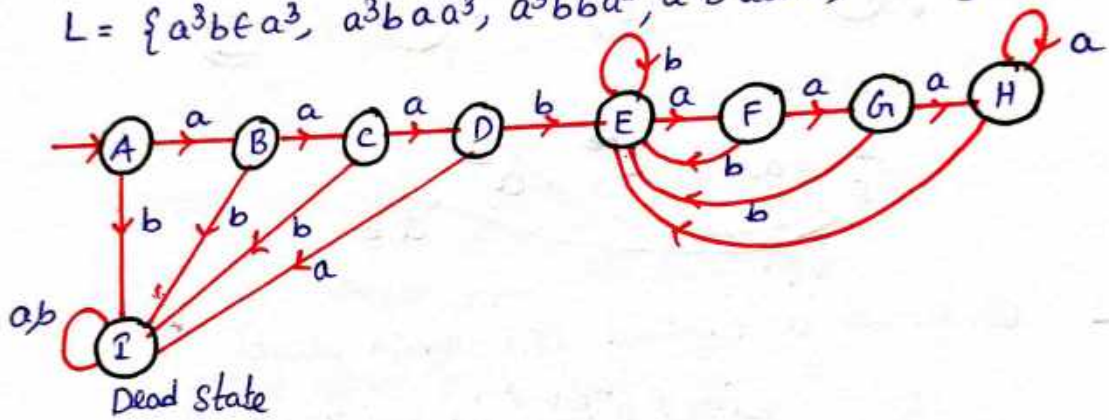
Ex: Construct a Minimal DFA which accepts set of all strings over $\{a, b\}$ such that second symbol from LHS is 'a'.

→ $L = \{a, aa, aa, ba, aa, \text{---} ba \text{---} \}$



Ex: Construct a DFA which accepts set of all strings over $\{a,b\}$ where strings are of the form $a^3 b\, w\, a^3$. where 'w' is any string over $\{a,b\}$

→ $L = \{a^3ba^3,\ a^3baa^3,\ a^3bba^3,\ a^3baaa^3, \dots\}$



OPERATIONS ON DFA LIKE UNION, CONCATENATION, CROSS PRODUCT, COMPLEMENTATION, REVERSAL:

~~UNION:~~ ~~Starts no of a's && Even no of b's~~

~~CONCATENATION:~~ ~~Does not Contain 'a'~~

CROSS PRODUCT: Even no. of a's && Even no. of b's.

COMPLEMENT: Does not Contain a.

REVERSAL: $aWb \rightarrow bwa$.

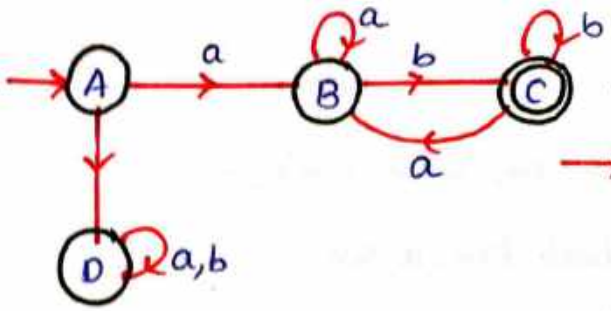
UNION: Starts and ends with different Symbols.

CONCATENATION: Starts with 'a' ends with 'b'.

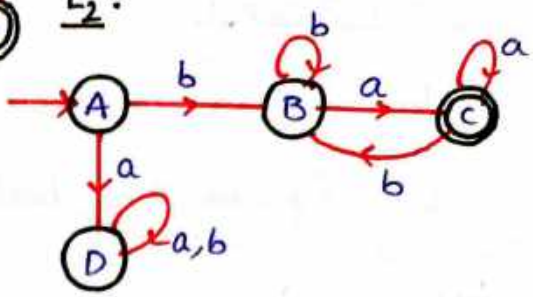
UNION: EX: $L_1 = \{ab, aab, abb, \dots\}$

$L_2 = \{ba, baa, bba \dots\}$

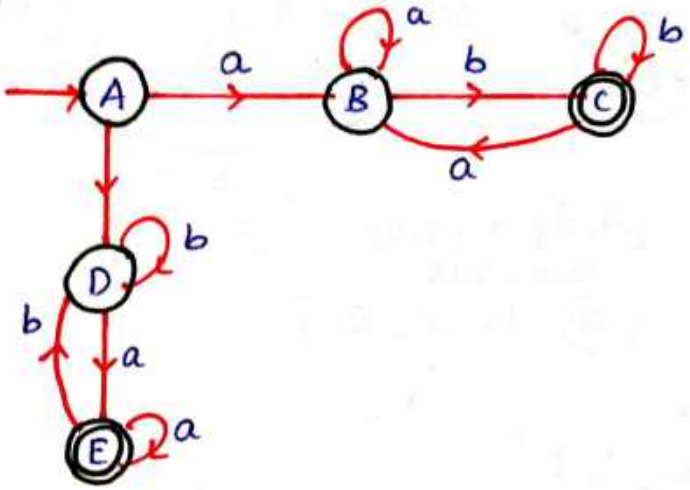
L_1:



L_2:



$L_1 \cup L_2$:



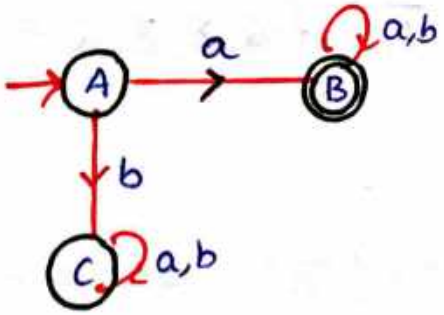
CONCATENATION:

L_1 = { Starting with 'a' } = {a, aa, ab, aaa, ---}

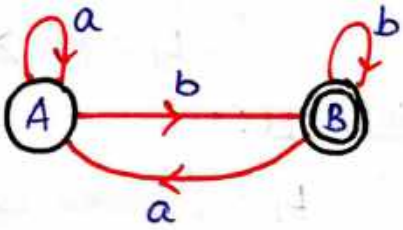
L_2 = { Ending with b} = {b, ab, bb, aab ---- }

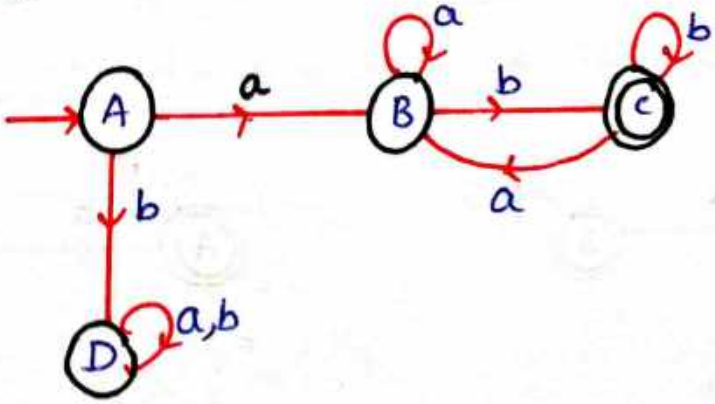
$L_1 \cdot L_2$ = {ab, aab, aab, ----- }

L_1:



L_2:



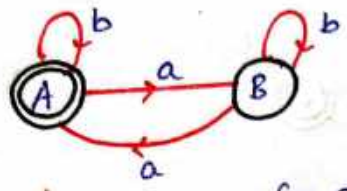


CROSS PRODUCT:

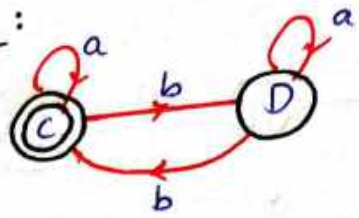
$L_1 = \{ \epsilon, b, aa, baa, bb, aba, aab, ----- \}$

$L_2 = \{ \epsilon, bb, abb, bab, bba, a, aa, ------ \}$

L_1:



L_2:



$\Rightarrow \{A, B\} \times \{C, D\}$

$\Rightarrow \{$ (AC) FINAL STATE, AD, BC, BD $\}$

COMPLEMENT:

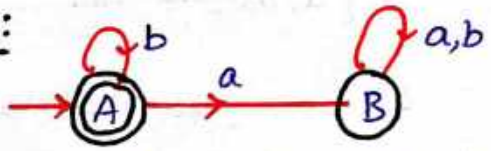
$L_1 = \{$ Containg 'a' $\}$

$L_1 = \{ a, aa, ba, aaa, bab, ----- \}$

$\overline{L_1} = \{ \epsilon, b, bb, bbbb, ------ \}$

L_1:

$\overline{L_1}$:



$L_1 = \{Q, \Sigma, \delta, F, q_0\}$ $\overline{L_1} = \{ Q, \Sigma, \delta, Q-F, q_0 \}$

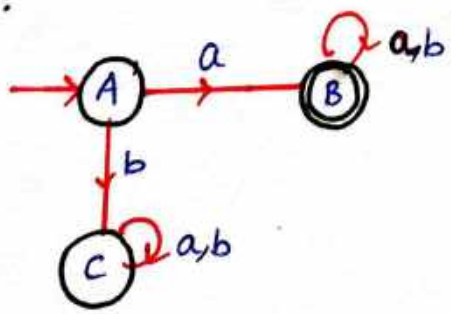
REVERSAL:

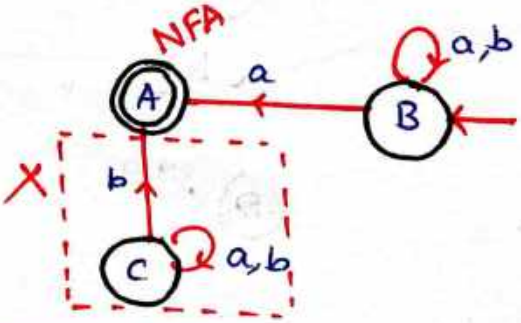
$L_1 = \{$ Starts with a $\}$

$= \{ a, aa, ab, aaa, aab, aba ---- \}$

$L_1^R = \{ a, aa, ba, aaa, baa, aba ---- \}$

L_1:





NOTE:

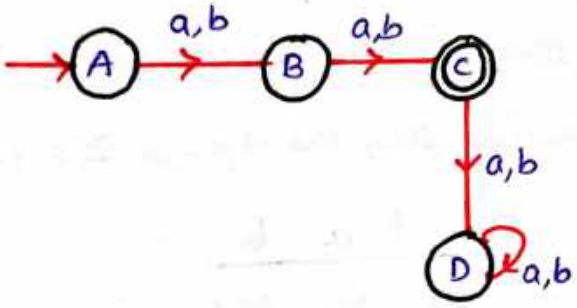
→ $L_1 - (DFA)^R - L_1^R -$ NFA & DFA

Ex: set of all strings of length '2'

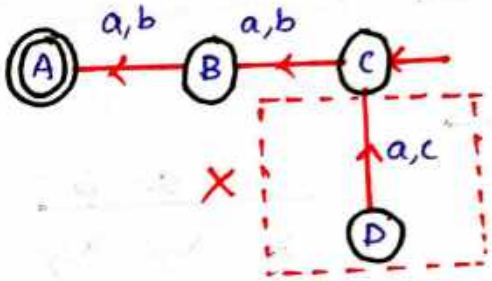
→ $L_1 = \{aa, ab, ba, bb\}$

$L_1^R = \{aa, ba, ab, bb\}$

L_1:



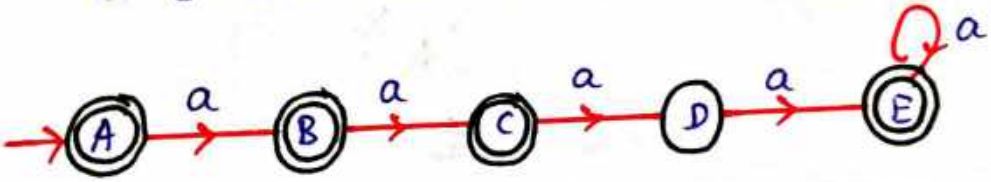
L_1^R:



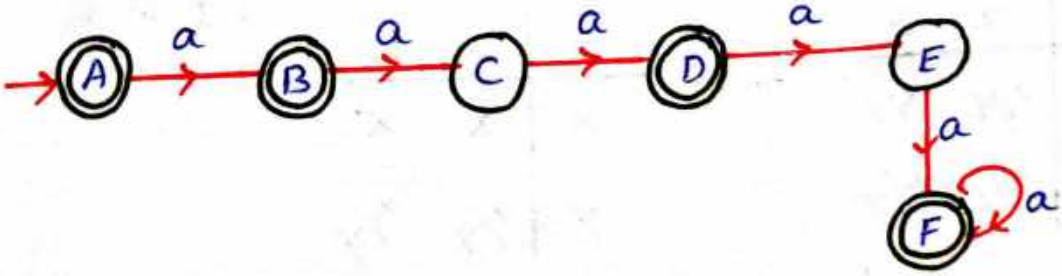
Ex: Construct a minimal DFA over {a}

1. For $\{a^n / n>=0, n! = 3\}$
2. For $\{a^n / n>=0, n! = 2, n! = 4\}$

→ 1. $L = \{\epsilon, a, aa, aaaa, aaaaaa \text{ -----}\}$



2. $L = \{\epsilon, a, aaa, aaaaa, \text{ ------}\}$



Ex: How many two state DFAs with a designated initial state and a designated final state can be constructed over the alphabet $\Sigma = \{a, b\}$?

→

	a	b
→ x	x/y	x/y
* y	x/y	x/y

$2 \times 2 \times 2 \times 2 = \underline{16}$

Ex: How many two states DFAs with a designated initial state and can be constructed over the alphabet $\Sigma = \{a, b\}$?

→

	a	b
x	x/y	x/y
y	x/y	x/y

16

	a	b
x	x/y	x/y
y	x/y	x/y

16

	a	b
x	x/y	x/y
y	x/y	x/y

16

	a	b
x	x/y	x/y
y	x/y	x/y

16

$4 \times 16 = \underline{64}$

Ex: How many two state DFA's can be constructed with a designated initial state that accept the empty language over the alphabet $\Sigma = \{a, b\}$

→

{} or ∅ ✓ ✓ {ϵ} ✗

	a	b
x	x/y	x/y
y	x/y	x/y

16

	a	b
x	x	x
y	x/y	x/y

4

$16 + 4 = \underline{20}$

Ex: How many two states DFAs can be Constructed with a designated initial state that accepts the universal Language over the alphabet $\Sigma = \{a, b\}$?

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

	a	b
*→ x	X/Y	X/Y
Y	X/Y	X/Y

} 16

	a	b
*→ x	x	x
Y	X/Y	X/Y

} 4

$16 + 4 = \underline{20}$

Ex: How many three state DFAs can be Constructed with a designated intial State over the alphabet $\Sigma = \{a, b\}$?

→

	a	b
x	X/Y/Z	X/Y/Z
Y	3	3 (X/Y/Z)
Z	3	3 (X/Y/Z)

$\underline{3^6}$

x Y Z } 1 (X) (Y) Z

(X) Y Z } 3 x (Y) (Z) } 3

x (Y) Z } 3 (X) x (Z)

x y (Z) } 3 (X) (Y) (Z) } 1

$3 + 3 + 1 + 1 = \underline{8}$

Total = $\underline{8 \times 3^6}$

Ex: How Many 'n' state DFAs Can be Constructed with a designated intial state over the alphabet Σ Containing 'm' Systems?

→ $|Q| = n \quad |\Sigma| = m$

	1	2	3	---	m
1	□	□	□	----	□
2	□				
3	⋮				
⋮					
n	□	□	□	---	□

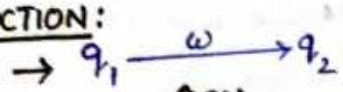
$m \times n$ ---- $(m \times n)$ times $\Rightarrow \underline{n^{m \times n}}$

$\Rightarrow ({}^nC_0 + {}^nC_1 + {}^nC_2 \cdots) \times n^{m \times n}$

$\Rightarrow \underline{2^n \times n^{m \times n}}$

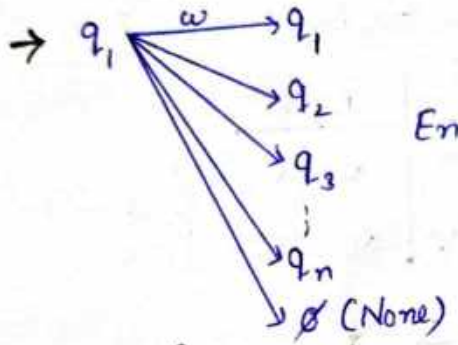
2. NFA

INTRODUCTION:



(DFA)

Ends in one state



(NFA)

Ends in Many states (or) one state (or) None state.

→ $(Q, \Sigma, \delta, q_0, F)$

Q : set of states (Finite)

Σ : i/p alphabet (F)

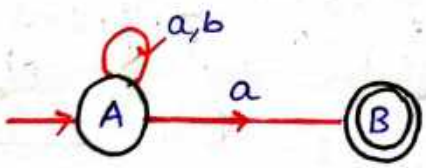
q_0 : start state $\quad \delta : Q \times \Sigma \rightarrow 2^Q$

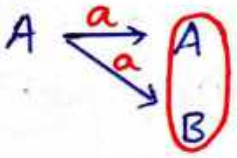
F : set of final states

Ex: L = {ends with 'a'}

Σ = {a, b}

→

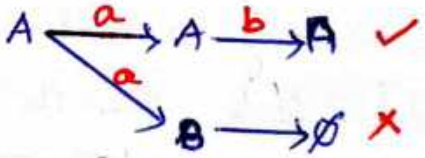




NFA

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

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



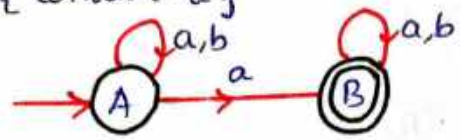
✓

✗

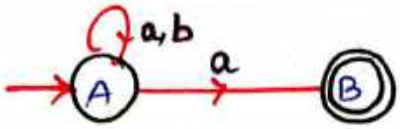
Ex: i) Σ = {a, b}

ii) L_1 = {starts with 'a'}

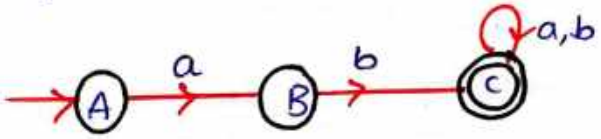
(ii) L_2 = { Contain 'a' }



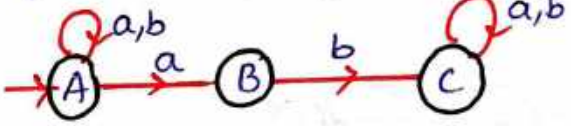
(iii) L_3 = { ends with 'a' }



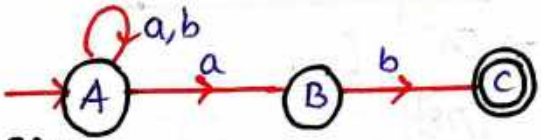
(iv) L_4 = { Starts with 'ab' }



(v) L_5 = { Contains 'ab' }



(vi) L_6 = { ends with 'ab' }



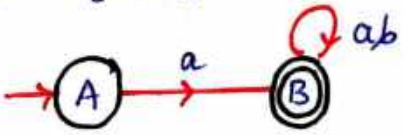
NFA → DFA:

Ex: **Converting of NFA to DFA:** for the example "all strings starting with a".

→ L_1 = { starts with 'a' }

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

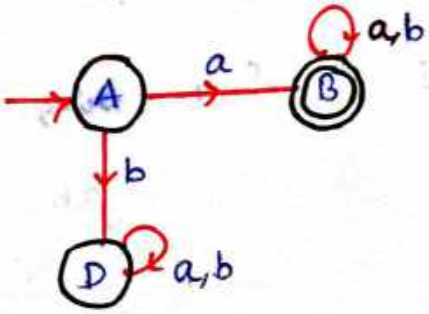
NFA:



	a	b
A	B	∅
*B	B	B

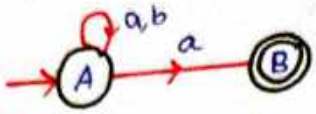
→

	a	b
A	B	D
*B	B	B
Dead-D	D	D



Ex: L = { Ends with an 'a' }

→ NFA:



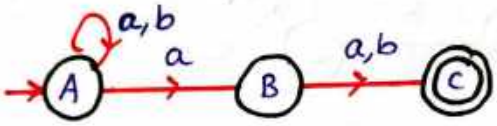
	a	b
→ A	{A,B}	{A}
* B	{ }	{ }

	a	b
→ [A]	[AB]	[A]
[AB]	[AB]	[A]

NOTE: [AB] is New state and Single State

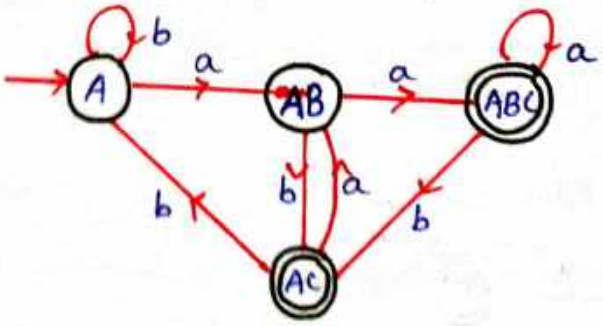
Ex: L={all strings in which Second Symbol from RHS is 'a'}

→ NFA: L = { aa, ab, aaa, aab, ---- }



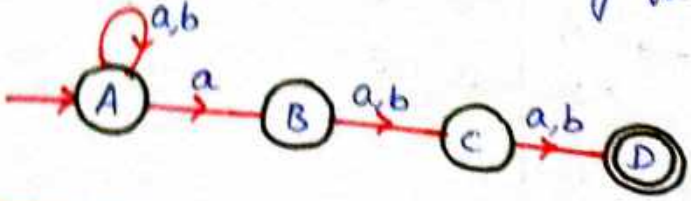
	a	b
→ A	{A,B}	{A}
B	{C}	{C}
* C	{ }	{ }

	a	b
→ [A]	[AB]	[A]
[AB]	[ABC]	[AC]
* [AC]	[AB]	[A]
* [ABC]	[ABC]	[AC]



Ex: L = {all strings in which third Symbol from RHS is 'a'}

→



	a	b
A	{A,B}	{A}
B	{C}	{C}
C	{D}	{D}
D	{ }	{ }

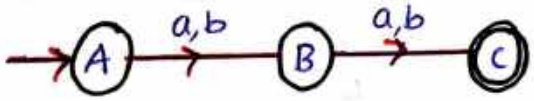
→ 4 states

	a	b
[A]	[AB]	[A]
[AB]	[ABC]	[AC]
[AC]	[ABD]	[AD]
[ABC]	[AB]	[A]
[ABC]	[ABCD]	[ACD]
[ABD]	[ABC]	[AC]
[ACD]	[ABD]	[AD]
[ABCD]	[ABD]	[ACD]

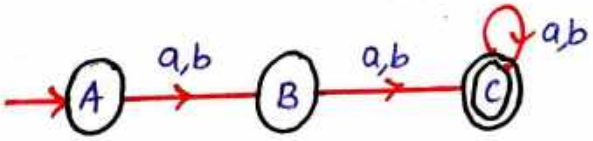
2^n States

EX: NFA for strings of length a. exactly 2
b. atmost 2
c. at least 2

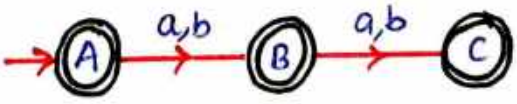
→ **A) EXACTLY '2':**



B) AT MOST '2':



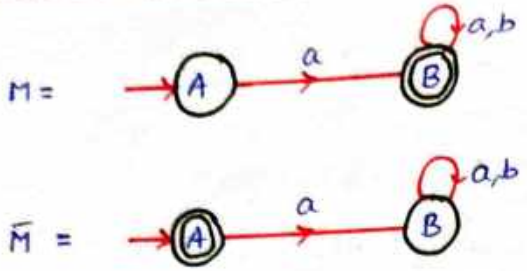
C) AT MOST '2':



COMPLEMENTATION OF NFA:

EX: NFA {a,b}

L = { starts with 'a' }



$L_1 = \{a, aa, ab, \dots\}$

$L_2 = \{\epsilon\}$

$\bar{L_1} \neq L_2$

NOTE:

$M \overset{C}{\Rightarrow} \bar{M}$

$\bar{L_1} \neq L_2$

Complementing NFA is different and Complementing the Language is Different.

3. MINIMISATION OF DFA

INTRODUCTION:

If DFA is given, we can apply some procedure using which we can decrease the No of states in DFA. This is also called as Minimisation of DFA.

PROPERTIES: (i) (P,Q) Equivalent

$\delta(P,w) \in F \Rightarrow \delta(Q,w) \in F$

$\delta(P,w) \notin F \Rightarrow \delta(Q,w) \notin F$

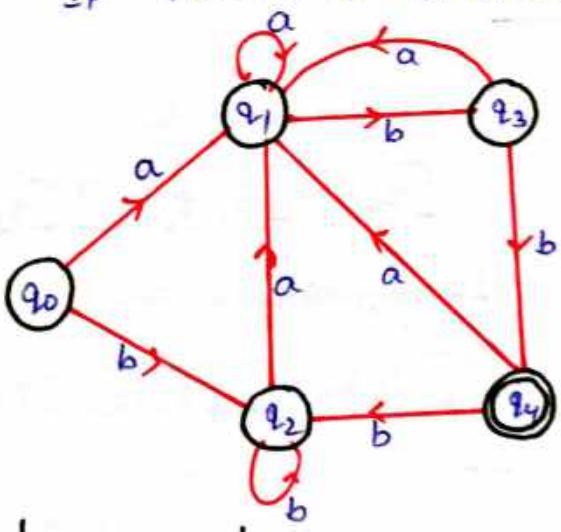
(ii) If $|W|=0$, Both (P,Q) are '0' equivalent.

If $|W|=1$, Both (P,Q) are '1' equivalent.

⋮

If $|W|=n$, Both (P,Q) are 'n' equivalent.

EX:



→

	a	b
→ 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

NON FINAL — FINAL

'0' equivalent $[q_0\ q_1\ q_2\ q_3]$ $[q_4]$

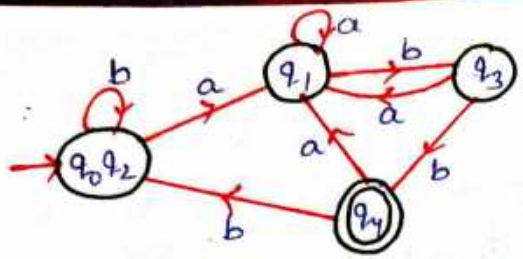
'1' equivalent $[q_0, q_1, q_2]$ $[q_3]$ $[q_4]$

'2' equivalent $[q_0, q_2]$ $[q_1]$ $[q_3]$ $[q_4]$

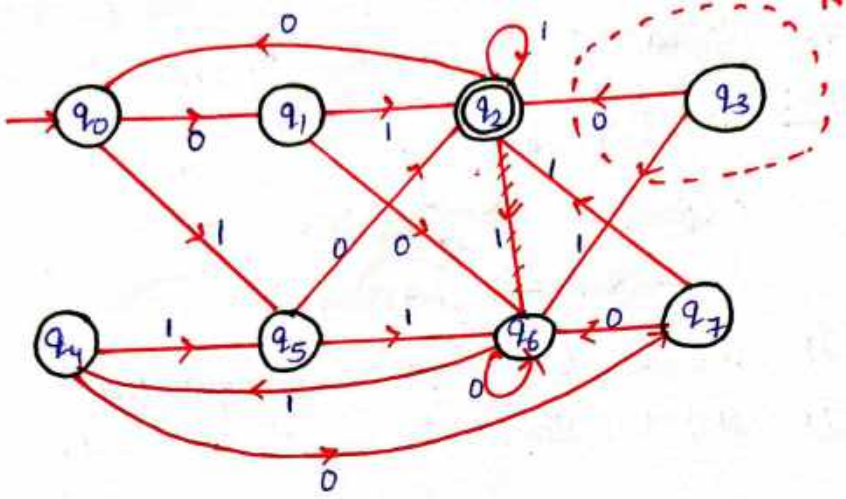
'3' equivalent $[q_0, q_2]$ $[q_1]$ $[q_3]$ $[q_4]$

⋮

(Same)



Ex:



→

		0	1
→	q_0	q_1	q_5
	q_1	q_6	q_2^*
*	q_2	q_0	q_2^*
NOT REACHABLE	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" equivalence states

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

"1" - equivalence states

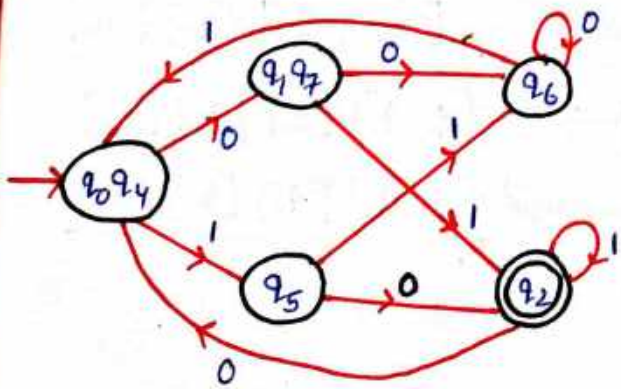
$[q_0 q_4 q_6]\ [q_1, q_7]\ [q_5]\ [q_4]\ [q_2]$

"2" - equivalent states

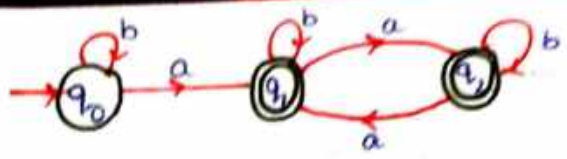
$[q_0 q_4]\ [q_6]\ [q_1 q_7]\ [q_5]\ [q_2]$

"3" - equivalent states

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



Ex:



→

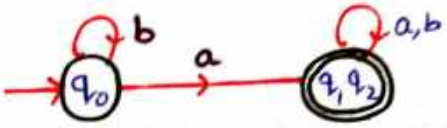
	a	b
→ q_0	q_1	q_0
* q_1	q_2	q_1
* q_2	q_1	q_2

"0"- equivalence $[q_0]\,[q_1, q_2]$

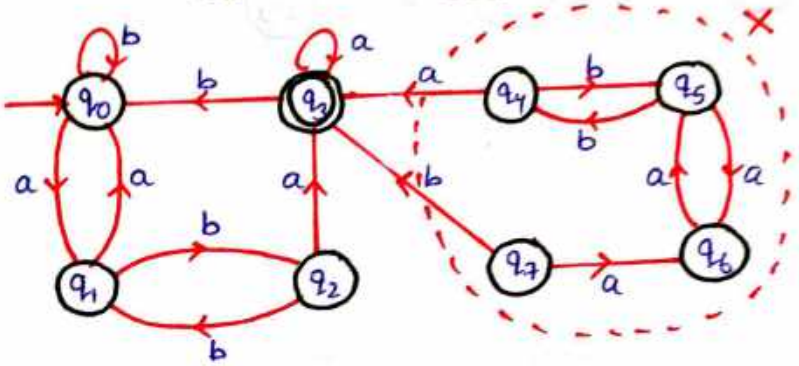
"1" - equivalence $[q_1]\,[q_1, q_2]$

⋮ ⋮

Same



Ex:



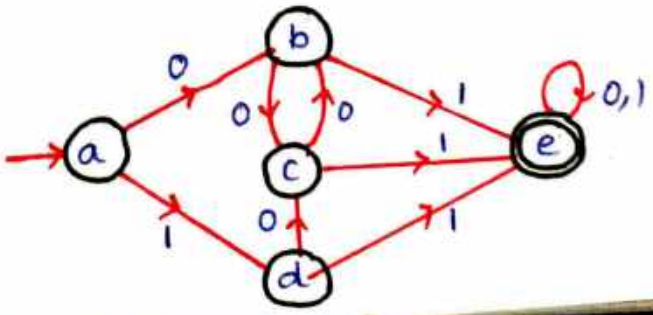
	a	b
→ q_0	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_1
* q_3	q_3	q_0
~~q_4~~	~~q_3~~	~~q_5~~ X
~~q_5~~	~~q_6~~	~~q_4~~ X
~~q_6~~	~~q_5~~	~~q_6~~ X
~~q_7~~	~~q_6~~	~~q_3~~ X

"0"- Equivalate $[q_0 q_1 q_2]\,[q_3]$

"1"- Equivalate $[q_0 q_1]\,[q_2]\,[q_3]$

"2" - Equivalate $[q_0]\,[q_1]\,[q_2]\,[q_3]$

EX:

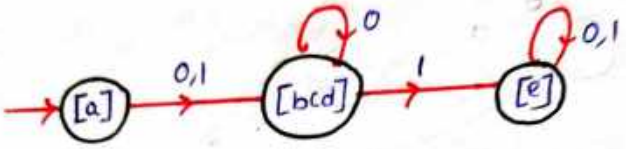


	0	1
→ a	b	d
b	c	e
c	b	e
d	c	e
* e	e	e

"0" - Equivalnce [a bc d] [e]

"1" - Equivalnce [a] [bc d] [e]

"2" - Equivalnce [a] [bcd] [e]



INTRO

a

Mo

Ex:

4. MOORE AND MELAY

INTRODUCTION:

FA with o/p

MOORE

MELAY

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

Q - Finite set of states

Σ - i/p alphabet

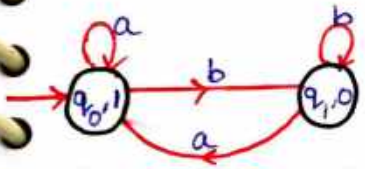
δ - Transition function

$Q \times \Sigma \to Q$

q_0 : intial state

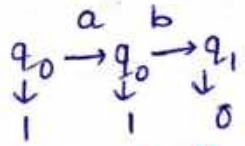
Δ : o/p alphabet

* λ : o/p function

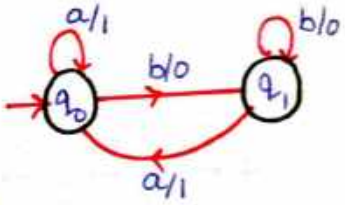


$\lambda : Q \to \Delta$

$q_0 \to 1 \quad q_1 \to 0$



'n' → 'n+1'



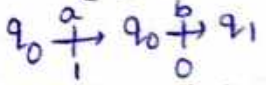
$\lambda : Q \times \Sigma \to \Delta$

$(q_0, a) \to 1$

$(q_0, b) \to 0$

$(q_0, b) \to 0$

$(q_0, a) \to 1$



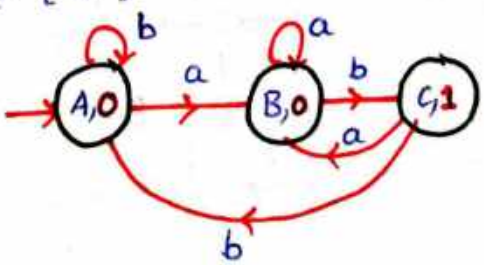
'n' → 'n'

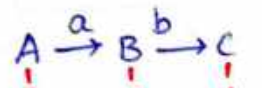
MOORE:

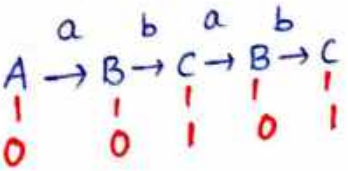
Ex: Construct a moore M/c that takes set of all strings over {a,b} as i/p and prints '1' as o/p for every occurance of 'ab' as a substring.

→ $\Sigma = \{a, b\}$

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



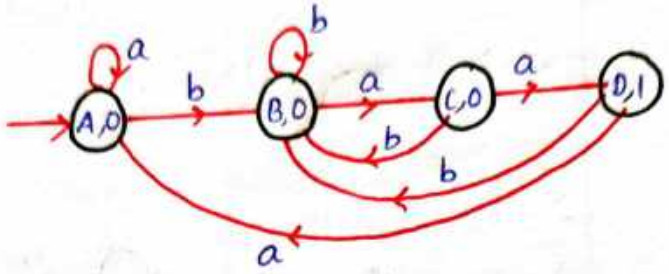


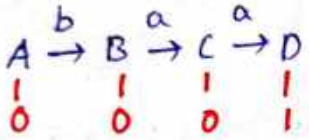


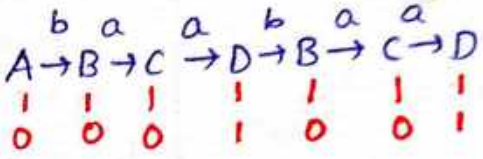
Ex: Construct a Moore M/c that takes set of all strings over {a,b} and counts no. of occurrences of substring 'baa'.

→ $\Sigma = \{a, b\}$

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





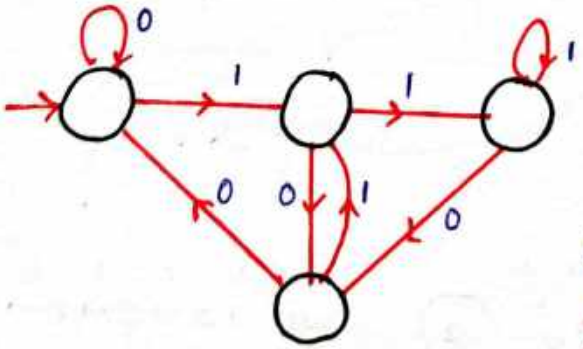


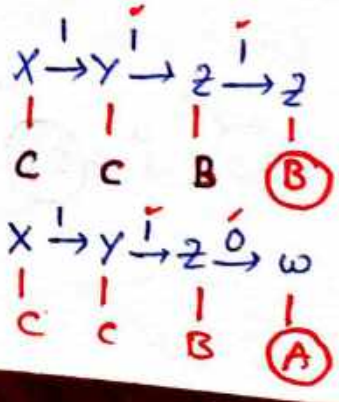
Ex: Construct a Moore M/c that takes set of all strings over {0,1} and produces 'A' as o/p if i/p ends with '10' & produces 'B' as o/p if i/p ends with '11' otherwise produces 'c'.

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

10 – A

11 – B

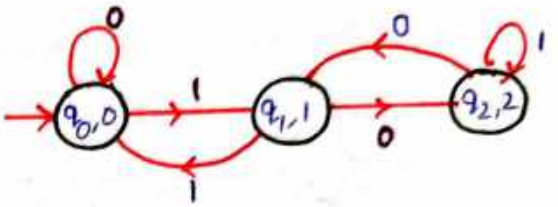




Ex: Construct a moore M/c that takes binary no's as i/p and produces residue modulo '3' as o/p.

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

	0	1	Δ
→ q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2



Ex: Construct a moore M/c that takes base 4 no's as i/p and produce residue modulo '5' as o/p.

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

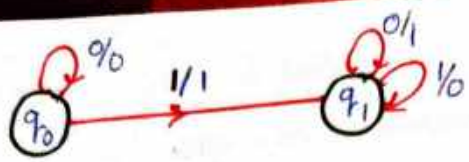
	0	1	2	3	Δ
→ q_0	q_0	q_1	q_2	q_3	0
q_1	q_4	q_0	q_1	q_2	1
q_2	q_3	q_4	q_0	q_1	2
q_3	q_2	q_3	q_4	q_0	3
q_4	q_1	q_2	q_3	q_4	4

MEALY:

Ex: Construct a Mealy M/c that takes binary numbers as i/p and produces 2's Complement of that number as o/p. Assume the string is read LSB to MSB and end carry is discarded.

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

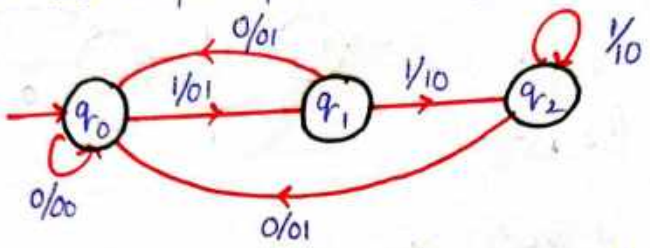
1 1 0 0
1's 0 0 1 1
2's 0 1 0 0



$q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_1$

0 0 1 0 ✓ 2'S COMPLEMENT

Ex: What is the output produced by the following state m/c



✗ a) 11 → 01

✓ c) Sum of present & previous bits

✗ b) 10 → 00

d) NOT

$q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_0 \xrightarrow{1} q_1$

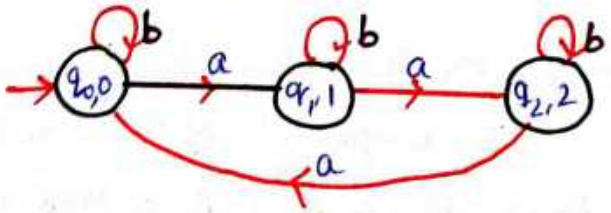
01 10 01 01

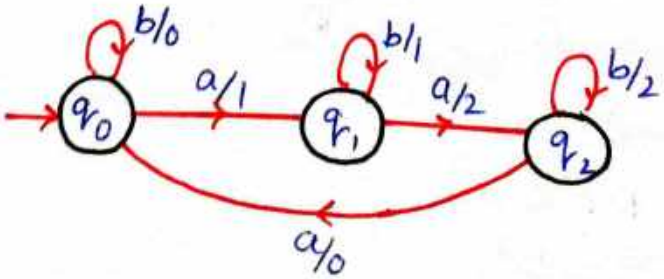
CONVERSION OF MOORE MACHINE TO MEALY MACHINE:

Moore ≅ Mealy

Having Same power

Ex: Count no. of a's %3





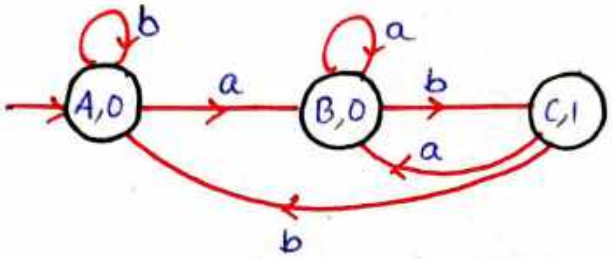
MOORE:

	a	b	Δ
q_0	q_1	q_0	0
q_1	q_2	q_1	1
q_2	q_0	q_2	2

MEALY:

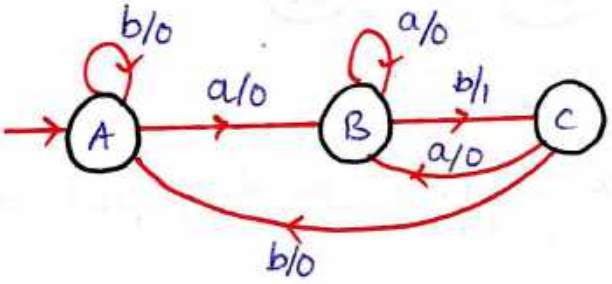
	a	b
q_0	$(q_1,1)$	$(q_0,0)$
→ q_1	$(q_2,2)$	$(q_1,1)$
q_2	$(q_0,0)$	$(q_2,2)$

EX:



Convert Moore to Mealy

→

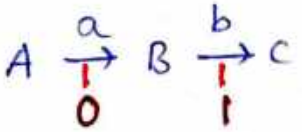


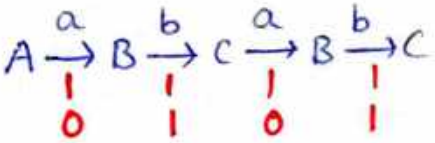
MOORE:

	a	b	Δ
A q_0	B	A	0
B q_1	B	C	0
C q_2	B	A	1

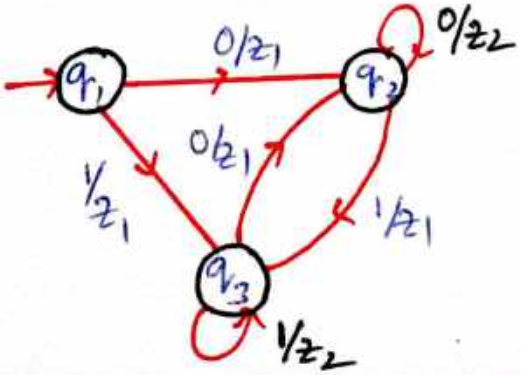
MEALY:

	a	b
A	(B,0)	(A,0)
→ B	(B,0)	(C,1)
C	(B,0)	(A,0)

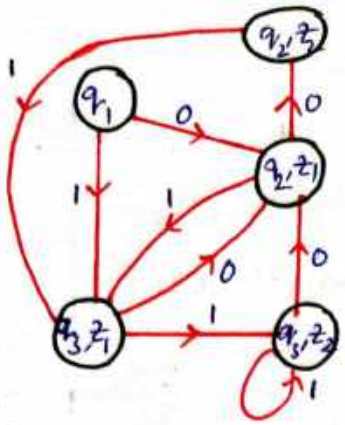




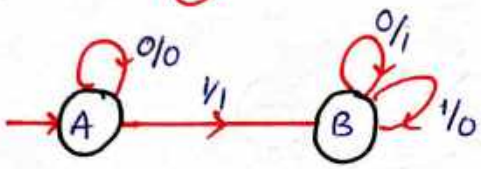
EX:



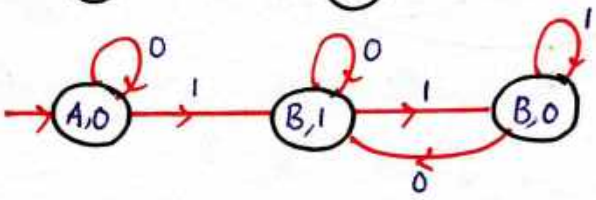
→



Ex:



→



NOTE:

→ Converting Moore to Mealy M/c the No. of States doesn't change.

→ Converting Mealy to Moore M/c the No. of States would Increased.

5. EPSILON NFA

ϵ-NFA:

$Q, \Sigma, \delta, q_0, F$

Q - Finite set of states.

Σ - i/p alphabet.

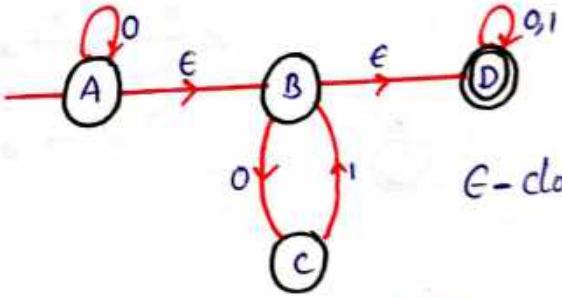
δ - Transition function

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

q_0 - Intial state

F - Final state

Ex:



ϵ-closure(A) = {A, B, D}

NOTE:

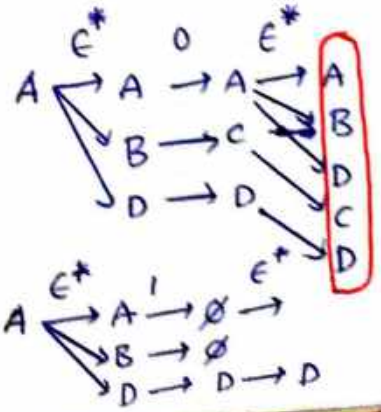
equal power equal power

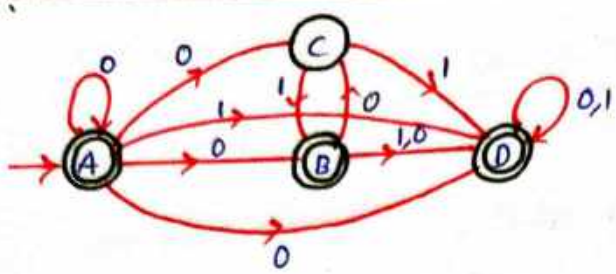
ϵ^* 0 ϵ^*

ϵ-closure (δ (ϵ closure (A), 0))

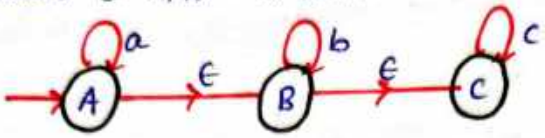
ϵ-NFA → NFA:

	0	1
A	{A,B,C,D}	{D}
B	{C,D}	{D}
C	{∅}	{B,D}
D	{D}	{D}





Ex: Convert ϵ-NFA $\longrightarrow$ NFA

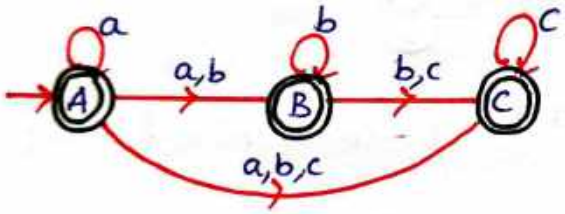


$\rightarrow$

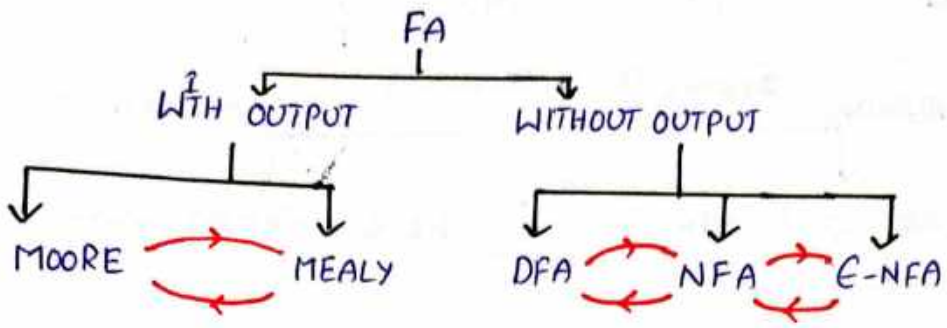
	a	b	c
A	{A,B,C}	{B,C}	{C}
B	{ }	{B,C}	{C}
C	{ }	{ }	{ }
●			

$A \xrightarrow{\epsilon^*} A \xrightarrow{a} A \xrightarrow{\epsilon^*} A$, B, C; $B \to \emptyset$; $C \to \emptyset$

$A \xrightarrow{\epsilon^*} A \xrightarrow{b} \emptyset \xrightarrow{\epsilon^*}$; $B \to B \to B$, C; $C \to \emptyset$

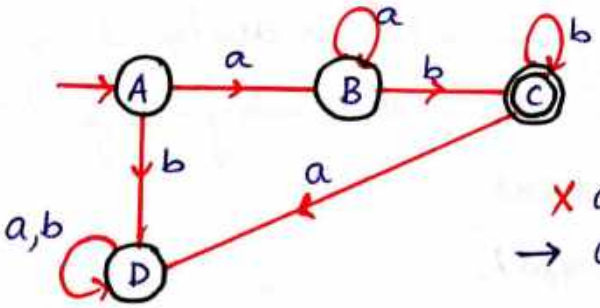


6. FAMILIES OF FORMAL LANGUAGE



Ex: $\{a^n b^n / n \geq 1\}$

→ $L = \{ab, aabb, aaabbb, \text{-----}\}$

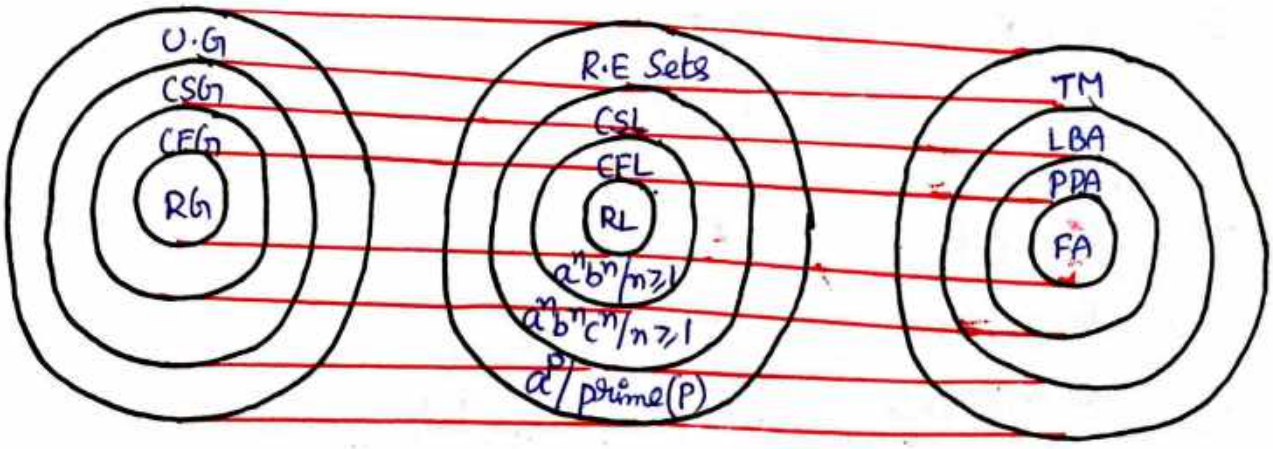


✗ aab ✓
→ accepted but Not in a Language

NOTE:

→ Finite automata + Memory = **PUSH DOWN AUTOMATA**

FAMILIES:

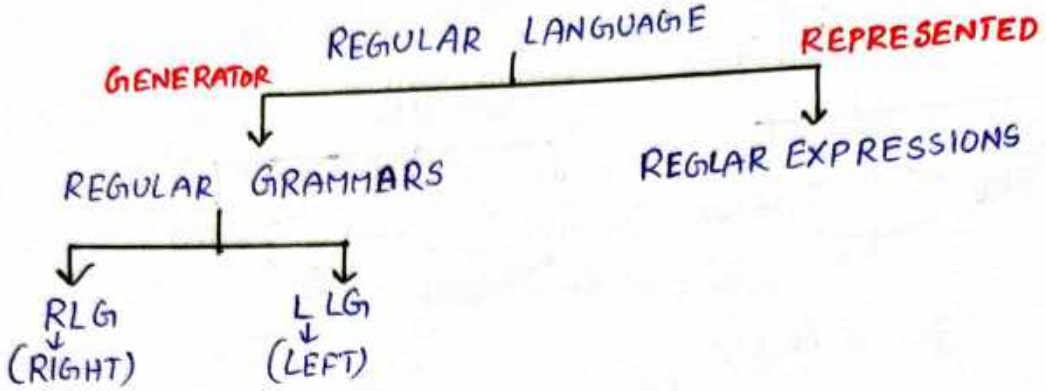


→ RL – $a^n / n \geq 1$ → CSL – $a^n b^n c^n / n \geq 1$

→ CFL – $a^n b^n / n \geq 1$ → R.E – a^p/ p is prime

7. REGULAR EXPRESSION AND CONVERSIONS

INTRODUCTION:



REGULAR EXPRESSION:

The Any language which is accetepted by Finite Automata. we can Iepresented using it by R.E.

(i) + (UNION)

(ii) • (Concat)

(iii) * (KLEEN closure)

a) $\emptyset, \epsilon, a \in \Sigma$ (primitive)

$\emptyset \to \{\}$, $\epsilon \to \{\epsilon\}$, $a \to \{a\}$, $b \to \{b\}$

b) $r_1 + r_2$, $r_1 \cdot r_2$, r_1^*

c) R.E

→ $\emptyset = \{\}$

$\epsilon = \{\epsilon\}$

$a = \{a\}$

$a^* = \{\epsilon, a, aa, aaa, ----\}$

$a^+ = a \cdot a^+ = \{a, aa, aaa, -----\}$

$(a+b)^* = \{\epsilon, a, b, aa, ab, bc, ----\}$

Ex: $\Sigma = \{a, b\}$ whose length is exactly '2'

→ $L_1 = \{aa, ab, ba, bb\}$ → FINITE

Regular Expression is union if language is finite

$\Rightarrow aa + ab + ba + bb$

$\Rightarrow a(a+b) + b(a+b)$

$\Rightarrow \underline{(a+b)(a+b)}$

Ex: $\Sigma = \{(a, b)\}$ whose length is atleast '2'

→ $L_1 = \{aa, ab, ba, bb, aaa, ----\}$ → INFINITE

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

↓ Exactly '2' ↓ More than 'two'

Ex: $\Sigma = \{a, b\}$ whose length is atmost '2'

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

$\Rightarrow \epsilon + a + b + aa + ab + ba + bb$

$\Rightarrow \underline{(a+b+\epsilon)(a+b+\epsilon)}$

Ex: $\Sigma = \{(a, b)\}$ even length string.

→ $L = \{\epsilon, aa, ab, ba, bb, -----\}$

$\Rightarrow \underline{((a+b)(a+b))^*} = (a+b)^{2n} / n \geq 0$

Ex: $\Sigma = \{a, b\}$ Set of all odd length strings

→ $\Rightarrow (a+b)^{2n+1} / n \geq 0$

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

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

Ex: $\Sigma = \{a,b\}$ Set of all strings divisible by '3'

→ $L = 0, 3, 6, 9, 12 \text{ ---}$

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

$\cong 2 \bmod 3$

$(a+b)^{3n+2} / n \geq 0$

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

Ex: $\Sigma = \{a,b\}$ No. of a's exactly '2'

→ $b^*a\,b^*a\,b^*$

Ex: $\Sigma = \{(a,b)\}$ atleast '2' a's

→ $b^*a\,b^*\,a\,(a+b)^*$

Ex: $\Sigma = \{ab\}$ No. of a's are atmost '2'

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

** Ex: $\Sigma = \{(a,b)\}$ No. of a's are even

→ $(b^*a\,b^*a\,b^*)^*$

$\{b, bb, bbb, \text{----}\}$ are also there

$\cong (b^*a\,b^*a)^*.\,b^*$

Ex: $\Sigma = \{a,b\}$ No. of strings starts with 'a'

→ $a\,(a+b)^*$

Ex: $\Sigma = \{a,b\}$ Set of strings Ends with 'a'.

→ $(a+b)^*a$

Ex: $\Sigma = \{a, b\}$ Set of all Strings Containg 'a'

→ $(a+b)^* a\ (a+b)^*$

Ex: $\Sigma = \{a, b\}$ Starting and ending with different Symbols.

→ $a\,(a+b)^* b + b(a+b)^* a$

Ex: $\Sigma = \{a, b\}$ starting and ending with Same Symbol.

→ $L = \{\epsilon, a, b, aa, bb, \text{-----}\}$

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

Ex: $\Sigma = \{a, b\}$ No 2 a's should Come together

→ $L = \{\epsilon, b, bb, bbb, a, ab, aba, abab \text{-----}\}$
ba, bab, baba---

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

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

(R)

$\Rightarrow a\,(b+ba)^* + (b+ba)^*$

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

Ex: $\Sigma = \{a, b\}$: No 2a's and No 2b's Come together.

$L = \{\epsilon, a, b, ab, ba, aba, bab \text{-----}\}$

	Starts with	Ends with
{a, aba, ababa---}	a	a — $(ab)^* a$ (R) $a(ba)^*$
{ab, abab, ababab---}	a	b — $(ab)^*$ (R) $a\,(ba)^* b$
{ba, baba, bababa, ---}	b	a — $(ba)^*$ (R) $b(ab)^* a$
{b, bab, babab, ----}	b	b — $(ba)^* b$ (R) $b(ab)^*$

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

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

$\Rightarrow \underbrace{(\epsilon+b)(ab)^*(a+\epsilon)}_{(R_1)}$

$\Rightarrow \underline{(\epsilon+a)(ba)^*(\epsilon+b)}$

IDENTITIES OF R.E:

→ $\emptyset + R = R + \emptyset = R$

→ $\emptyset \cdot R = R \cdot \emptyset = \emptyset$

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

→ $\epsilon^* = \epsilon$

** → $\emptyset^* = \epsilon$

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

** → $(a+b)^* = (a^*+b^*)^*$

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

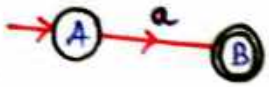
$(a^*+b^*)^*$

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

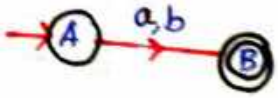
CONVERSION OF REGULAR EXPRESSION TO FA:

→ $\emptyset$ →

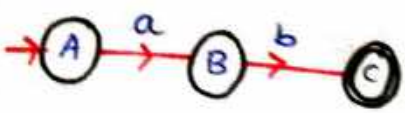
→ a →



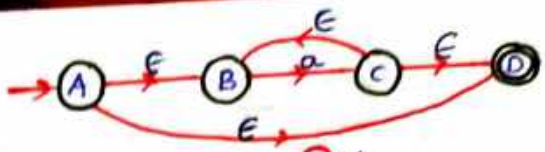
→ $a+b$ →



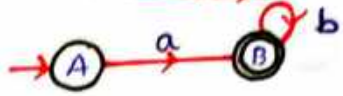
→ $a \cdot b$ →



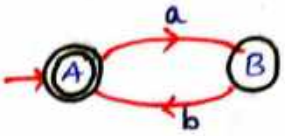
→ a^* →



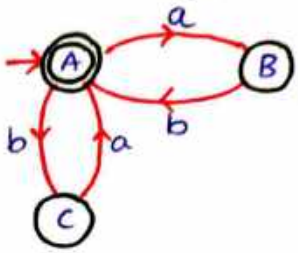
→ ab^* →



→ $(ab)^*$ →



→ $(ab+ba)^*$ →

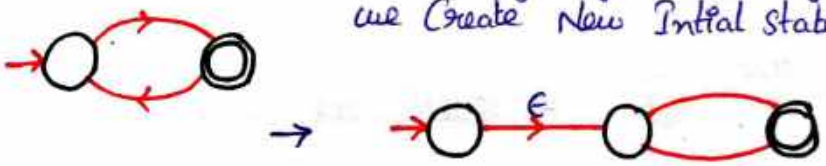


NOTE: Regular expression and Finite automata are both equivalent in power.

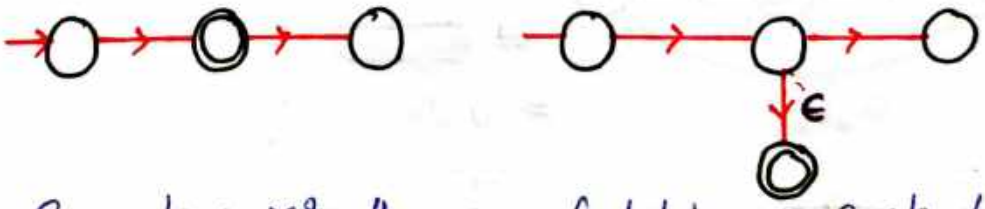
CONVERSION OF FA TO RE:

STATE ELIMINATION METHOD:

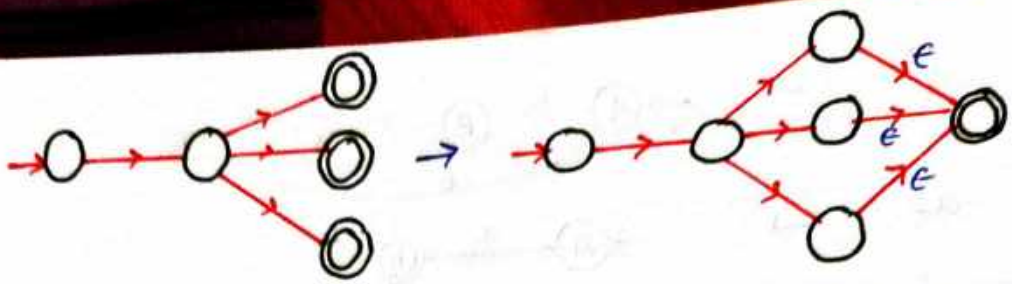
1) Intial State Have No Incoming Edges, If Having any Intial State, we Create New Intial State with 'ε' move.



2) Final State Have No outgoing Edges, If Having outgoing Edges then Create a New final state with 'ε' Move.

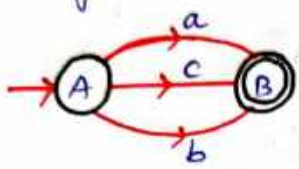


3) If we have More than one final state we Construct a New final state and with 'ε' Moves. Make all final States as Non-Final States.



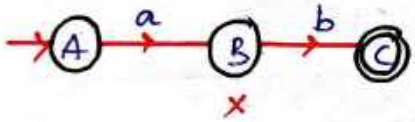
4) Other than the Initial State and final State eliminate the remaining states one by one.

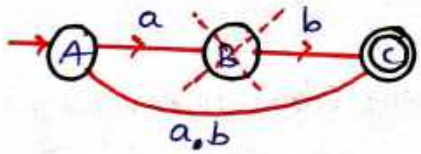
Ex:



$\Rightarrow (a+b+c)$

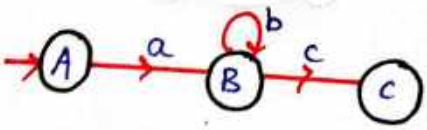
Ex:

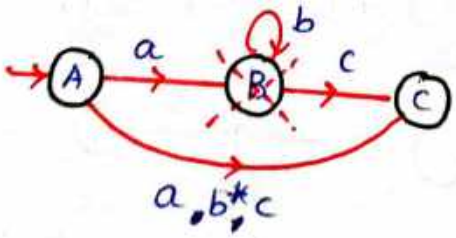




$\Rightarrow$ ab

Ex:

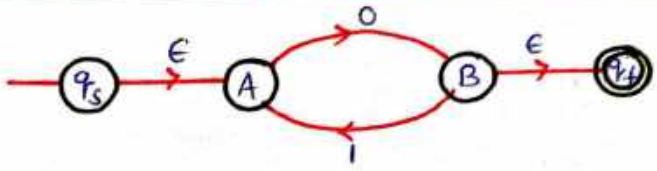


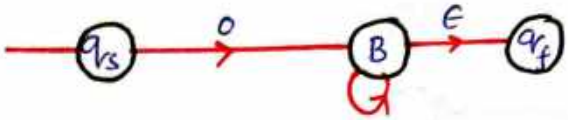


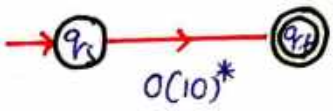
$\Rightarrow ab^*c$

Ex:

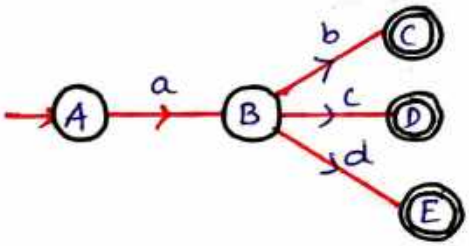
→


q_s
ϵ
A
0
1
B
ϵ
q_f

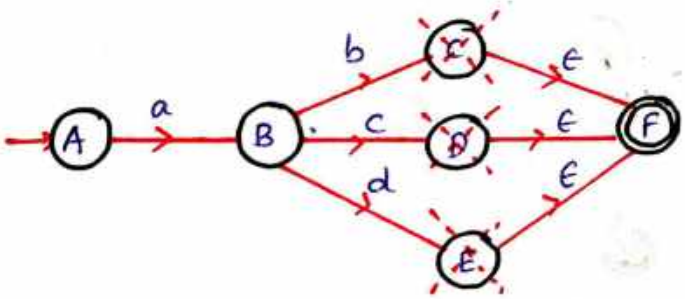

q_s
0
B
ϵ
q_f

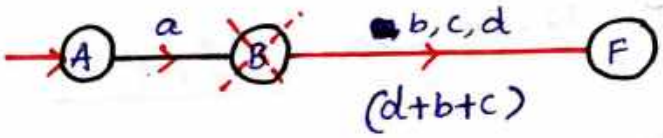

q_s
q_f
$0(10)^*$

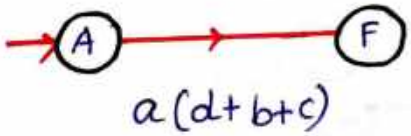
Ex:


A
a
B
b
C
c
D
d
E

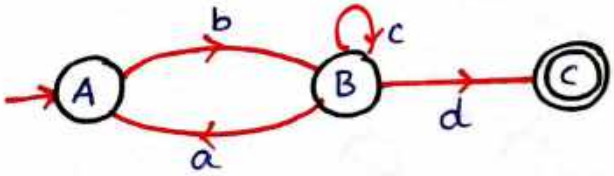
→


A
a
B
b
C
c
D
d
E
ϵ
ϵ
ϵ
F

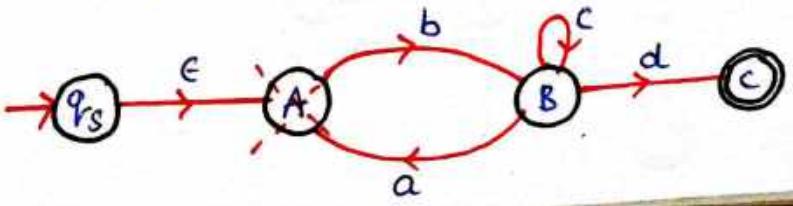

A
a
B
b, c, d
F
$(d+b+c)$

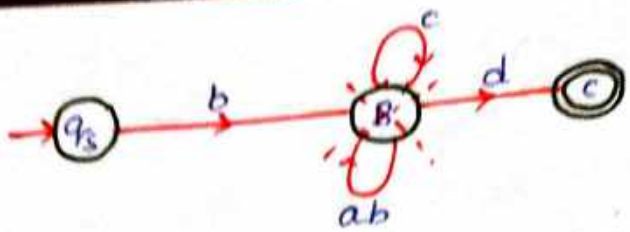

A
F
$a(d+b+c)$

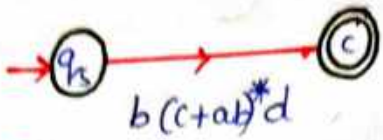
Ex:


A
b
B
c
C
d
a

→

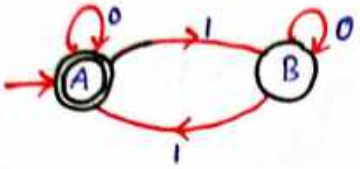

q_s
ϵ
A
b
B
c
d
C
a



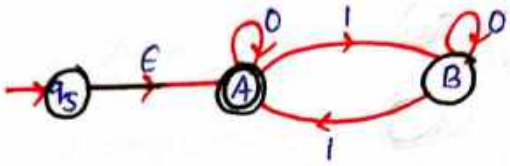


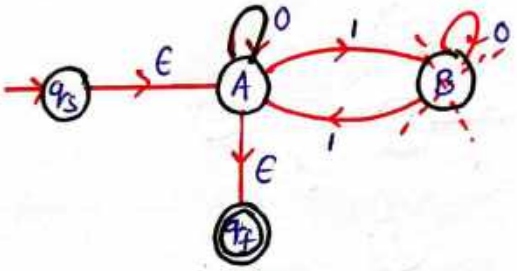
$b(c+ab)^*d$

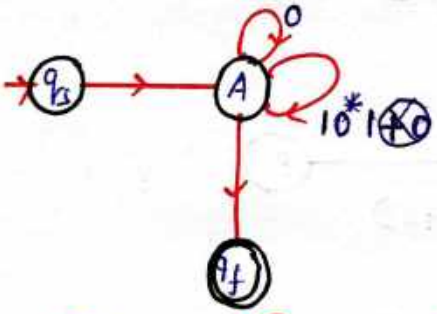
Ex:



→

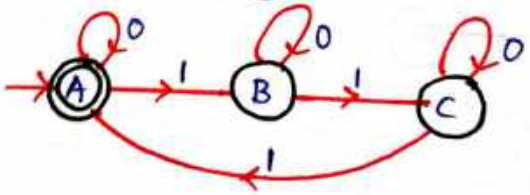




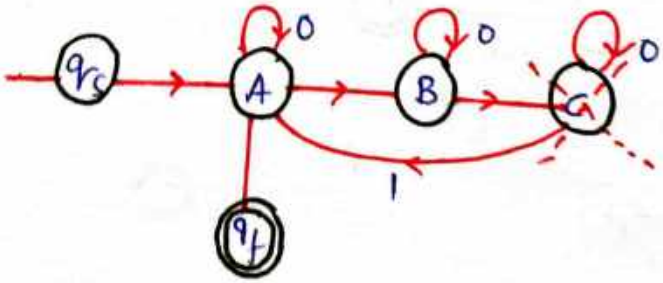


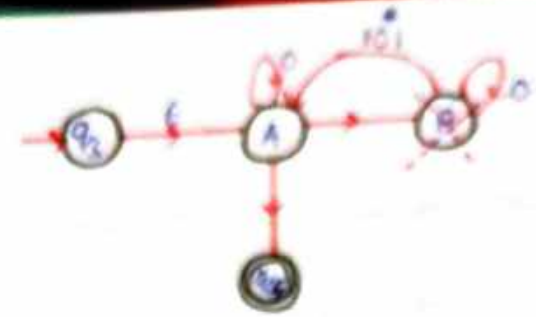
$\Rightarrow \underline{(10^*1+0^*)}$

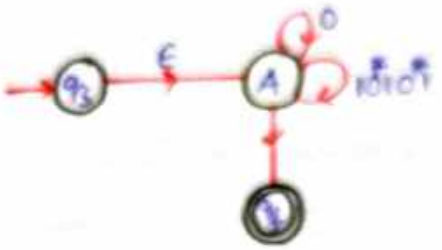
Ex:

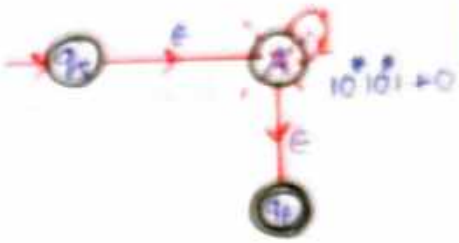


→









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

8. TESTING WHETHER A LANGUAGE IS REGULAR

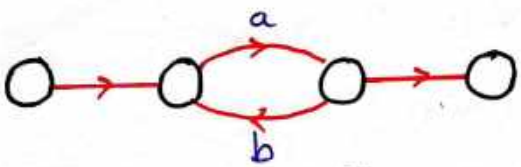
TESTING WHETHER A LANGUAGE IS REGLAR OR NOT:

Finite – Regular – R.E → +_ +_ +_ ---

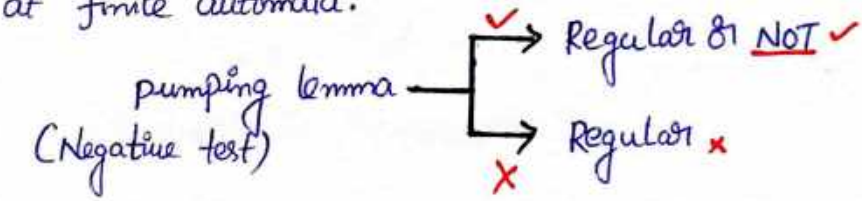
Infinite – L = {ab, abab, ababab ----}

→ If a Infinite language accepted by finite automata Definitly there is loop in the finite automata, and on the loop there should be a pattern.

→ If there is No Such pattern and No finite automata should possible.

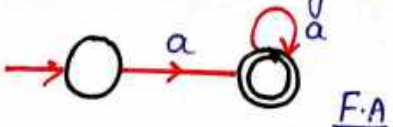


→ pumping lemma Says that if a infinite language accepted finite automata then there should be loop at finite automata.



Ex: $a^n / n \geq 1$ find Regular & Not?

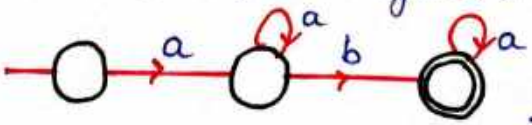
→



F.A

Regular

Ex: $a^n b^n / n, m \geq 1$ find Regular & not?



FA

REGULAR

Ex: $a^n b^n / n \leq 10^{10^{10^{10}}}$ Regular or Not?

→ n is bounded so finite language

∴ The language is Regular.

Ex: $a^n b^n / n \geq 1$ Regular or Not?

→ Counting in unbounded

Infinite automata ∴ The language is NOT Regular.

Ex: $WW^R / |W| = 2, \Sigma = \{a,b\}$

→ aaaa
abba
baab
bbbb
} Bounded ∴ The language is Regular

Ex: $WW^R / w \in (a,b)^*$

→ ....aba/aba... Infinite length

→ Unbounded

∴ The language is NOT Regular

Ex: $ww / w \in (a,b)^*$

→ ---abb/abb---- Infinite length

→ Unbounded

∴ The language is NOT Regular

Ex: $a^n b^m c^k / n,m,k \geq 1$

→ $a^n b^n c^k / n,m,k \geq 1$ → Generate Independently

'F.A'

Could not Save (or) any one Could not Remembered (or) Compared (or) Counted.

∴ The language is Regular

Ex: $a^n b^n c^n / n \geq 1$

→ ⇒ a, b, c are depends on 'n' and generate same No. of a's, b's, c's. Depends each other

∴ The language is NOT Regular

Ex: $a^i b^{2j} / i, j \geq 1$

→ generate any No. of a's and even No. of b's

$aa^* bb(bb)^*$ F.A

∴ The language is Regular

Ex: $a^i b^{4j} / i, j \geq 1$

→ generate any No. of a's and '4' multiples of b's

∴ The language is Regular and Finite Automata.

Ex: $\Sigma = \{a\}$

(i) a^n / n is even = $\{a^0, a^2, a^4, a^6, \ldots\}$ Loop (A.P) – REGULAR

(ii) a^n / n is odd = $\{a, a^3, a^5, a^7, \ldots\}$ Loop (A.P) – REGULAR

(iii) a^n / n is prime = Cannot find any loop – NOT REGULAR

(iv) $a^{n^2} / n \geq 1 = \{a, a^4, a^9, a^{16}, a^{25}\}$ Not (A.P) – NOT REGULAR

(v) $a^{2^n} / n \geq 1 = \{a, a^2, a^4, a^8, a^{16}, \ldots\}$ Not (A.P) – NOT REGULAR

Ex: $\Sigma = \{a, b\}$ $a^i b^{i^2} / i, j \geq 1$

$a^i b^{i^2}$ → b^{i^2} is Not in Arithmetic progression (A.P)

∴ b^{i^2} is Not Regular

∴ The language is NOT Regular.

Ex: $a^i b^{2^n} / i, n \geq 1$, $\Sigma = \{a, b\}$

→ $a^i b^{2^n}$ → b^{2^n} is not in arithmetic progression

∴ b^{2^n} is Not Regular

∴ The Language is NOT Regular.

Ex: $\Sigma = \{a, b\}$ where $w / n_a(w) = n_b(w)$, $w \in (a,b)^*$

→ No. of a's equal to No. of b's

∴ Storage is required

unbounded Counting

∴ The Language is NOT Regular.

Ex: $\Sigma = \{(a, b)\}$, where $w \in (a,b)^*$ where $w / n_a(w) \bmod 3 \leq n_b(w) \bmod 3$

→ $n_a(w) \bmod 3 \leq n_b(w) \bmod 3$

Mod is bounded Counting

∴ The Language is Regular.

Ex: $w w w^R / w \in (a,b)^*$

→ $w w w^R$ Comparision

Infinite

∴ The Language is NOT Regular

Ex: $a^n b^{n+m} c^m / n, m \geq 1$

→ $a^n b^{n+m} c^m$ Store 'a's and Compare b's and

Store b's and Compare c's

∴ The Language is NOT Regular.

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

→ $w = 101$

$x = 100$

$\underset{w}{101}\ \underset{x}{100}\ \underset{w^R}{101}$

1 0 1 1 0 0 1 0 1

← X →

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

∴ The language is Regular

Ex: $\omega x \omega^R / \omega \in (0,1)^+ \quad |x| = 5$

→ Restriction upon 'x'

'x' cannot go beyond some limit.

∴ The language Cannot Regular

Ex: $x\omega\omega^R y / x, y, \omega \in (0,1)^+$

→ $X = 10 \quad \omega = 100$

$Y = 11 \quad \omega^R = 001$

10 100 001 11

x ω ω^R y

1 0 1 0 0 0 1 1 1

x y

$(0+1)^*00(0+1)^* + (0+1)^*11(0+1)^*$

∴ The language is Regular

Ex: $x\omega\omega^R / x\bullet, \omega \in (0,1)^+$

→ $\omega = 101$

$\omega^R = 101$

$X = 11$

11 101 101

x ω ω^R

'X' is Not expand as like the above problem.

∴ The language is NOT Regular

Ex: $\omega\omega^R y / \omega, y \in (0,1)^+$

→ Like above problem 'y' is Not expanded

∴ The language is NOT Regular

PUMPING LEMMA FOR REGULAR LANGUAGE:

If L be an infinite regular language and $w \in L$ such that $|w| \geq n$ for some positive integer n, then w can be decomposed into three strings $w = xyz$ such that

(i) $y \neq \epsilon$ or $|y| > 0$

(ii) $|xy| \leq n$

(iii) For every $k \geq 0$, the string xy^kz is also in L.

NOTE:

1. Every regular language satisfies the pumping lemma property.
2. If a language 'L' does not satisfy the pumping lemma property then 'L' is non regular.
3. If a language 'L' satisfies the pumping lemma property, then 'L' need not be regular.
4. pumping lemma is used to prove that some of the languages are not regular.

9. GRAMMARS

INTRODUCTION: (V, T, P, S)

V – variables (vertices)

T – set of all terminals

P – production

S – start symbol

EX:

$$P\begin{cases} S \to aSB \\ S \to aB \\ B \to b \end{cases} \quad \begin{matrix} V = \{(S,B)\} \\ T = \{a,b\} \end{matrix} \Rightarrow a^n b^n / n \geq 1$$

DERIVATION:

Derivation of aabb from the above grammar.

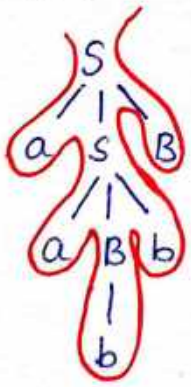
$S \Rightarrow aSB$ —

$\Rightarrow a\,a\,SB\,B$ — } Sentential form

$\Rightarrow a\,a\,bB$ —

$\Rightarrow$ aa bb

DERIVATION TREE:



aa bb

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

→ Language is finite

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

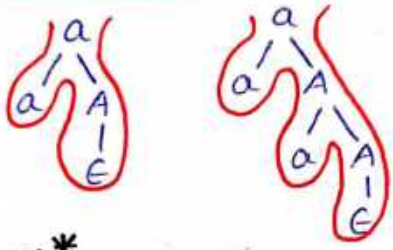
$\Rightarrow (a+b)(a+b)$

$S \rightarrow AA$
$A \rightarrow a/b$

Ex: $a^n / n \geq 0$

$\rightarrow$ L = a, aa, aaa, aaaa .....

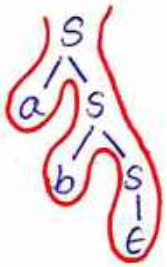
$A \rightarrow aA/\epsilon$ (or) $A \rightarrow Aa/\epsilon$



Ex: $(a+b)^*$

$\rightarrow$ L = ϵ, a, b, aa, ab, ba, bb .....

$S \rightarrow aS/bS/\epsilon$



ab

Ex: atleast '2' $\Sigma = \{a,b\}$

$\rightarrow$ $(a+b)(a+b)$ → two

$(a+b)(a+b)(a+b)^*$ → two or More

$S \rightarrow AAB$

$A \rightarrow a/b$

$B \rightarrow aB/bB/\epsilon$

Ex: atmost '2' $\Sigma = \{a,b\}$

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

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

Ex: all strings starting with 'a' and ending with 'b'.

→ $a(a+b)^*b$

$S \to aAb$

$A \to aA/bA/\epsilon$

Ex: all string starting and ending with diffrent Symbols

→ $a(a+b)^*b + b(a+b)^*a$

$S \to aAb/bAa$

$A \to aA/bA/\epsilon$

Ex: All strings starts and ends with Same Symbol.

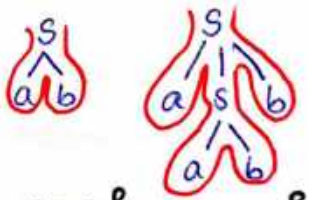
→ $a(a+b)^*a + b(a+b)^*b$

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

$A \to aA/bA\epsilon$

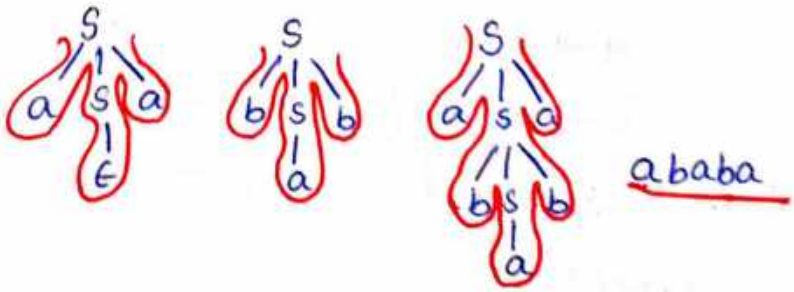
Ex: $a^n b^n / n \geq 1$

→ $S \to aSb/ab$



Ex: $w w^R \cup w a w^R \cup w b w^R$ where $w \in (a,b)^*$

→ $S \to aSa/bSb/a/b/\epsilon$



abaaba

Ex: Even length strings

→ $((a+b)(a+b))^*$

$S \to BS/\epsilon$

$B \to AA$

$A \to a/b$.

Ex: $a^n b^m / n, m \geq 1$

→

$S \to AB$

$A \to aA/a$

$B \to bB/b$

Ex: $a^n b^n c^m / n, m \geq 1$

→ $S \to AB$

$A \to aAb/ab$

$B \to cB/c$

Ex: $a^n c^m b^n / n, m \geq 1$

→ $S \to aSb / aAb$

$A \to cA/c$

Ex: $a^n b^n c^m d^m / n, m \geq 1$

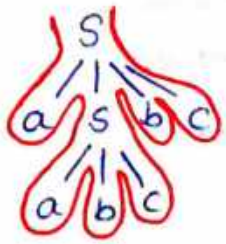
→ $S \to AB$

$A \to aAb/ab$

$B \to cBd/cd$

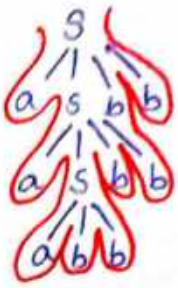
Ex: $a^n b^n c^n / n \geq 1$

→ $S \to aSbc / abc$



Ex: $a^n b^{2n} / n \geq 1$

→ $S \rightarrow aSbb / abb$



Ex: $a^n b^m c^m d^n / n, m \geq 1$

→ $S \rightarrow asd / aAd$

$A \rightarrow bAc / bc$

Ex: $a^n b^m c^n d^m / n, m \geq 1$

→ $a^n b^m c^n d^m$ ✗

No grammar

Ex: $a^{m+n} b^m c^n / m, n \geq 1$

→ $a^n a^m b^m c^n$

$S \rightarrow aSc / aAc$

$A \rightarrow aAb / ab$

Ex: $a^n b^{n+m} c^m / n, m \geq 1$

→ $a^n b^n b^m c^m$

$S \rightarrow AB$

$A \rightarrow aAb / ab$

$B \rightarrow bBc / bc$

TYPES OF GRAMMAR:

According to Chomsky

(i) Type 3

(ii) Type 2

(iii) Type 1

(iv) Type 0

TYPE-3:

Ex: $A \to \alpha B / \beta$ } RLG

$A, B \in V$

$\alpha, \beta \in T^*$

$A \to Bx / \beta$ } LLG

$A, B \in V$

$\alpha, \beta \in T^*$

Ex: $A \to Ba / a$ ✗ NOT TYPE-3

$B \to aB / a$

Because these grammar is Combination of Both Right Linear and left linear.

NOTE: Languages generated by Type-3 grammar are Regular languages.

CONVERSION OF FA TO RG:

Ex:

→ $A \to aB$
$B \to aB / bB / \epsilon$ } $L \xrightarrow{R}$ (REVERSE) $A \to Ba$
$B \to Ba / Bb / \epsilon$ } L^R

The above grammar is Right linear.

∴ The grammar is Type-3 grammar.

∴ The grammar also Regular grammar.

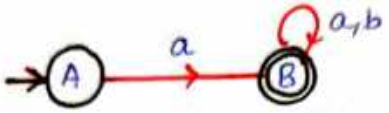
NOTE: → FA → RLG $\xrightarrow{R}$ LLG

(L) (L) (L^R)

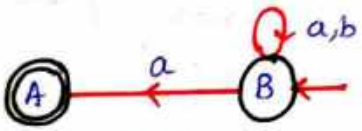
→ FA $\xrightarrow{R}$ FA → RLG $\xrightarrow{R}$ LLG

(L) (L^R) (L^R) $(L^R)^R$ = (L)

Ex:



→ Reversing the finite automata



(L^R)

⇒ $B \to aB/bB/aA$ $\xRightarrow{R}$ $B \to Ba/Bb/Aa$

$A \to \epsilon$ $A \to \epsilon$

(L^R) $(L^R)^R = L$

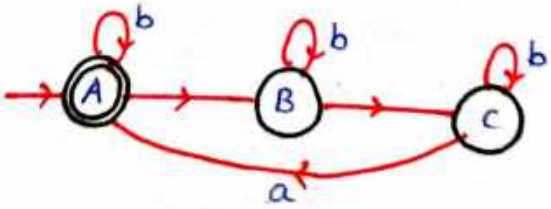
CONVERSION OF RG TO FA:
(RLG)

Ex: $A \to aB/bA/b$

$B \to ac/bB$

$c \to aA/bc/a$

→



The afoue Example is Conversion of RLG → FA

LLG → FA:

→ LLG $\xrightarrow{R}$ RLG → FA $\xrightarrow{R}$ FA
(L) (L^R) (L^R) $(L^R)^R = L$

→ REGULAR GRAMMAR $\cong$ FINITE AUTOMATA

Both are equal in power

TYPE-2:

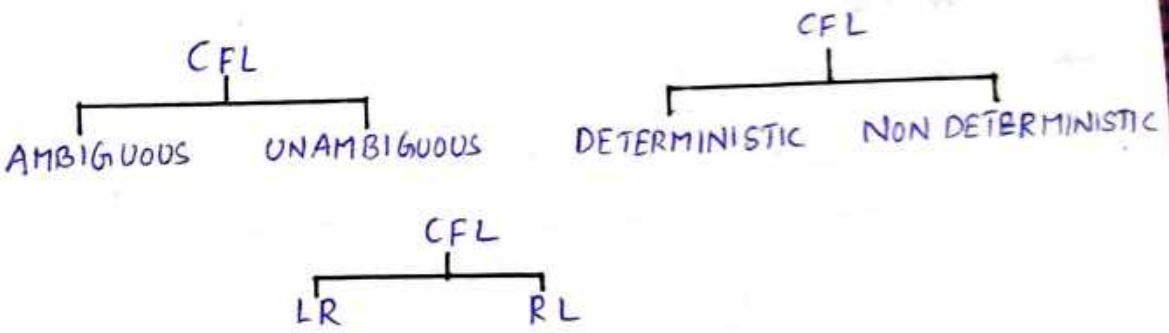
$A \to \alpha$

$A \in V$

$\alpha \in (V \cup T^*)$

→ Type-2 grammars are Context free grammars generated by Context-free languages.

→ The machine that accepted by CFG is push down automata.



$\omega \in L(G)$

1) Step 1 : $S \to \alpha$

2) Step 2 : $S \Rightarrow aAb$
abb

3) Step 3 : $S \Rightarrow aABb$
$aaBb$
$aabb$

(K) $\omega \in L(G)$

CFL → $A \to \epsilon$ × $A \to BC$

$A \to B$ $A \to a$

$|W|$

$\Rightarrow |W| + |W| = 2|W|$

$\Rightarrow P + P^2 + P^3 + \cdots + P^{2\omega}$

$O(P^{2\omega+1})$

NOTE: → Contex-free grammar should not Contain

ϵ- productions.

U-unit productions.

useless Symbols.

→ CYK algorithem time complexity $O(|W|^3)$

ELEMINATION OF ϵ-PRODUCTIONS:

Ex: $S \to aSb / aAb$

$A \to \epsilon$

→ Fining Nullable Variables

$A \Rightarrow \epsilon$ → Directly derived 'ϵ'

$A \Rightarrow () \Rightarrow () \Rightarrow \epsilon$ → Indirectly derived 'ϵ'

$\{A\}$

→ Replace Nullfle product with 'A' and without 'A'. And remove 'ϵ' product.

$S \to asb / aAb / ab$

~~$A \to \epsilon$~~

→ Also eleminate the Nullable products

$S \to asb / aAb / ab$

$\Rightarrow S \to asb / ab$

Ex: $S \rightarrow AB$
$A \rightarrow aAA/\epsilon$
$B \rightarrow bBB/\epsilon$

→ Nullable variables are

$\{A, B, S\}$ S' also Nullable while 'S' derived 'A' and 'B' replaced with 'ϵ' they also Nullable.

$\therefore \epsilon \in L(G)$

The S' is start symbol so we cannot remove 'ϵ' in entire grammar, we include 'ϵ' in any one of the productions.

$S \rightarrow AB/B/A/\epsilon$
$A \rightarrow aAA/aA/a$
$B \rightarrow bBB/bB/b$

} Replacing with and without on Right Hand side.

Ex: $S \rightarrow AbaC$
$A \rightarrow BC$
$B \rightarrow b/\epsilon$
$C \rightarrow D/\epsilon$
$D \rightarrow d$

→ Nullable Variable are

$\{C, B, A\}$

$S \rightarrow Abac/bac/Aba/ba$
$A \rightarrow Bc/B/c$
$B \rightarrow b$
$C \rightarrow D$
$D \rightarrow d$

ELIMINATION OF UNIT PRODUCTIONS:

Ex: $S \rightarrow Aa/B$
$B \rightarrow A/bb$
$A \rightarrow a/bc/B$

→ Writting gammer without using unit productions

$S \to Aa$

$B \to bb$

$A \to a/bc$

$S \to B \to bb$

$S \to B \to A \to a$ / bc

$S \to Aa/bb/a/bc$

$B \to bb/a/bc$

$A \to a/bc/bb$

$B \to A \to a$ / bc

$B \to A \to B$

$A \to B \to bb$

If we delete the unit production we have to add the chain products that the unit production produced.

Ex:

$S \to AB$

$A \to a$

$B \to C/b$

$C \to D$

$D \to E$

$E \to a$

→

$S \to AB$

$A \to a$

$B \to b/a$

$C \to a$

$D \to a$

$E \to a$

$B \to C \to D \to E \to a$

$C \to D \to E \to a$

$D \to E \to a$

ELIMINATION OF USELESS SYMBOLS:

Ex:

$S \to AB/a$

$A \to BC/b$

$B \to aB/C$

$C \to aC/B$

$T = \{a, b\}$

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

→ 'S' derived 'a' which is terminal

'A' derived 'b' which is terminal

{a, b, S, A} → useful

B → aB/c
C → ac/B
} useless Symbols Not derived anything

⇒ S → ~~AB~~/a
A → ~~Bc~~/b
B → aB/c ✗
C → ac/B ✗

⇒ S → a
A → b ✗
⇒ S → a

Ex: S → AB/AC
A → aAb/bAa/a
B → bbA/aaB/AB
C → abCA/aDb
D → bD/aC

T = {a, b}
V = {S, A, B, C, D}

→ {a, b, A, B, S} usefull Symbols

✗ C → abCA/aDb
✗ D → bD/aC
} useless So we eliminate

S → AB/~~Ac~~
A → aAb/bAa/a
B → bbA/aaB/AB

⇒

S → AB
A → aAb/bAa/a
B → bbA/aaB/AB

Ex: S → ABC/BaB
A → aA/BaC/aaa
B → bBa/a
C → CA/AC

→ {a, b, A, B, S} - usefull

✗ C → cA / Ac - useless

S → ~~ABC~~ / BaB
A → aA / ~~Bac~~ / aaa
B → bBa / a

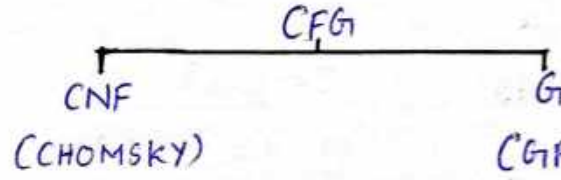
⇒

S → BaB
A → aA / aaa ✗
B → bBa / a

⇒ 'S' derived only 'B' Not reachable to 'A' So ~~delete~~ delete 'A'

⇒ S → BaB
B → bBa / a

CNF INTRODUCTION:



CNF:

If all the productions are of form A → BC & A → a where A, B, C are variables and 'a' is a terminal

ADV:

(i) length of each production is restricted.
(ii) Derivation tree & parse tree obtained from CNF is always binary tree.
(iii) The Number of steps required to derived a string of length (2|W|-1)
(iv) It is easy to apply CYK membership algorithm. $O(n^3)$

Ex: $S \to AB$
$A \to a$
$B \to b$

→ String ab

$S \Rightarrow AB$
$\Rightarrow aB$
$\Rightarrow ab$

3 steps

$|ab| = 2$ (Length of string)

$2(2)-1 = 3$

Ex: $S \to AB$
$A \to a$
$B \to b$

→ String abb

$S \Rightarrow AB$
$\Rightarrow ABB$
$\Rightarrow aBB$
$\Rightarrow abB$
$\Rightarrow abb$

5 steps

$|abb| = 3$

$2(3)-1 = 5$

Ex: Convert the following into CNF

$S \to bA/aB$
$A \to bAA/aS/a$
$B \to aBB/bS/b$

→ $N_a \to a$
$N_b \to b$

$S \to N_bA / N_aB$
$A \to N_bAA / N_aS / a$
$B \to N_aBB / N_bS / b$
$N_a \to a$
$N_b \to b$

$S \to N_b A / N_a B$

$A \to N_b S / N_a S / a$

$B \to N_a T / N_b S / b$

$N_a \to a$

$N_b \to b$

$S \to AA$

$T \to BB$

∴ The grammar is CNF.

Ex: Check weather the string "baaba" is a valid member of the following CFG

$S \to AB / BC$

$A \to BA / a$

$B \to CC / b$

$C \to AB / a$

→ 1 2 3 4 5
baaba

	5	4	3	2	1
1	A,S,C	∅	∅	A,S	B
2	S,A,C	B	B	A,C	
3	B	S,C	A,C		
4	A,S	B			
5	A,C				

→ $O(n^2)\ O(n)$

$O(n^3)$

$(1,2) = (11)(22)$

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

$= \{BA, BC\}$

BA generated 'A', BC generated S

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

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

$= AA, AC, CA, CC$

X X X ✓ B

$(3,4) = (3,3)(3,4)$

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

$= AB, CB$

↓ X

S,C

$(4,5) = (4,4)(5,5)$

$= \{(B), (A,C)\}$

$= \underset{\downarrow A}{BA}, \underset{\downarrow B}{BC}$

$(1,3) = (1,1)(2,3)(B)(1,2)(3,3)$

$\{B\}\ \{B\}$ $\quad$ $\{A,S\}\ \{A,C\}$

$\underset{}{BB}$ $\quad$ $\underset{\times}{AA}, \underset{\times}{AC}, \underset{\times}{SA}, \underset{\times}{SC}$

$(1,3) = \emptyset$

$(2,4) = (2,2)(3,4)\ (B)\ (2,3)(4,4)$

$= \{(A,C)(S,C)\}\ \{(B)(B)\}$

$= \underset{\times}{AS}, \underset{\times}{AC}, \underset{\times}{CS}, \underset{\downarrow B}{CC}$ $\quad$ $\underset{\times}{BB}$

$(3,5) = (3,3)(4,5)\ (B)\ (3,4)(5,5)$

$= \{(A,C)(A,S)\}\ \{(S,C)(A,C)\}$

$= \underset{\times}{AA}, \underset{\times}{AS}, \underset{\times}{CA}, \underset{\times}{CS}$ $\quad$ $\underset{\times}{SA}, \underset{\times}{SC}, \underset{\times}{CA}, \underset{\downarrow B}{CC}$

$(1,4) = (1,1)(2,4)\ (B)\ (1,2)(3,4)\ (B)\ (1,3)(4,4)$

$= \{(B),(B)\}$ $\quad$ $\{(A,S)(S,C)\}$ $\quad$ $\{\emptyset,\}$

$= \underset{\times}{BB}$ $\quad$ $\underset{\times}{AS}, \underset{\times}{AC}, \underset{\times}{SS}, \underset{\times}{SC}$ $\quad$ $\underset{\times}{\ }$

$(1,4) = \emptyset$

$(2,5) = (2,2)(3,5)\ (B)\ (2,3)(4,5)\ (B)\ (2,4)(5,5)$

$\{(A,C)(B)\}$ $\quad$ $\{(B),(A,S)\}$ $\quad$ $\{(B)(A,C)\}$

$\underset{S,C}{\underset{\downarrow}{AB}}, \underset{\times}{CB}$ $\quad$ $\underset{A}{\underset{\downarrow}{BA}}, \underset{\times}{BS}$ $\quad$ $\underset{\downarrow A}{BA}, \underset{\downarrow C}{BC}$

$(1,5) = (1,1)(2,5)\ (B)\ (1,2)(3,5)\ (B)\ (1,3)(4,5)\ (B)\ (1,4)(5,5)$

$\{(B)(S,A,C)\}$ $\quad$ $\{(A,S)(B)\}$ $\quad$ $\downarrow \emptyset$ $\quad$ $\downarrow \emptyset$

$\underset{\times}{BS}, \underset{\downarrow A}{BA}, \underset{\downarrow S}{BC}$ $\quad$ $\underset{S,C}{\underset{\downarrow}{AB}}, \underset{\times}{SB}$ $\quad$ $\emptyset$ $\quad$ $\emptyset$

Ex: Check weather the string "aabbb" is a valid member of following grammar or Not

$S \rightarrow AB$

$A \rightarrow BB/a$

$B \rightarrow AB/b$

→

	5	4	3	2	1
1	S,B	A	S,B	∅	A
2	S,B	A	S,B	B	
3	S,B	A	B		
4	A	B			
5	B				

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

A A = AA

$(2,3) = \{(2,2)(3,3)\}$

A B = AB

$(3,4) = \{(3,3)(4,4)\}$

B B = BB

$(4,5) = \{(4,4)(5,5)\}$

B B = BB

$(1,3) = \{(1,1)(2,3)\}$ (8) $\{(1,2)(3,3)\}$

(A) (S,B) ∅

AS, AB

X S,B

$(2,4) = \{(2,2)(3,4)$ (8) $(2,3)(4,4)\}$

(A)(A) (S,B)(B)

AA SB, BB

X X A

$(3,5) = \{(3,3)(4,5)\}$ (8) $\{(3,4)(5,5)\}$

B A A B

BA AB

X S,B

$(1,4) = \{(1,1)(2,4)\}$ 8) $\{(1,2)(3,4)\}$ 8) $\{(1,3)(4,4)\}$

(A,A) ∅ (S,B)(B)

AA SB, BB

X X A

$(2,5) = \{(2,2)(3,5)\}$ 8) $\{(2,3)(4,5)\}$ 8 $\{(2,4)(5,5)\}$

(A) (S,B) (S,B)(A) (2,4)(5,5)

AS AB SA, BA A B = AB

X S,B X X S,B

$(1,5) = \{(1,1)(2,5)\}$ (A) $\{(1,2)(3,5)\}$ (B) $\{(1,3)(4,5)\}$ (C) $\{(1,4)(5,5)\}$

A (S,B) | $\emptyset$ | (S,B)(A) | A B

$\frac{AS}{\times}$ $\frac{AB}{S,B}$ | $\frac{SA}{\times}$ $\frac{BA}{\times}$ | $\frac{AB}{S,B}$

GNF (GRIEBACH):

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

Ex: $A \to aBCDE$

$A \to a$

ADV:

1) The number of steps required to generate a string of length $|W|$ is $|W|$.
2) GNF is useful in order to convert a CFG to PDA

CONVERSION OF CFG TO PDA:

1) push the start symbol 'A' on to the stack

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

2) push RHS of A as follows

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

if $A \to a\alpha$ is in 'G'

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

if $A \to b\beta$ is in G

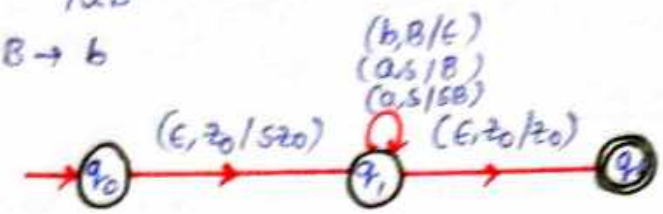
3) Add final state with

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

Ex: Convert the following into PDA:

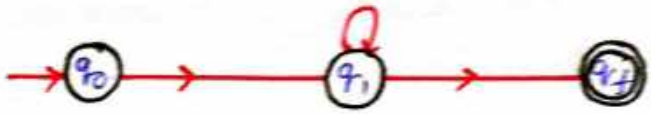
$S \rightarrow aSb \mid ab$

$\rightarrow$ $S \rightarrow aSB$
$\mid aB$
$B \rightarrow b$



Ex: $S \rightarrow aAA$
$A \rightarrow aS \mid bS \mid a$

$\rightarrow$ $S \rightarrow aAA$
$A \rightarrow aS \mid bS \mid a$



(a, S/AA)
(a, A/S)
(b, A/S)
(a, A/ϵ)

10. PDA

PDA: FA + Stack

$(Q, \Sigma, \delta, q_0, z_0, F, \Gamma)$

PDA
- DPDA
- NPDA

Q : Finite Set of States

Σ : Input Symbol

δ : Transition function

q_0 : Intial State

z_0 : Bottom of the Stack

F : Set of final States

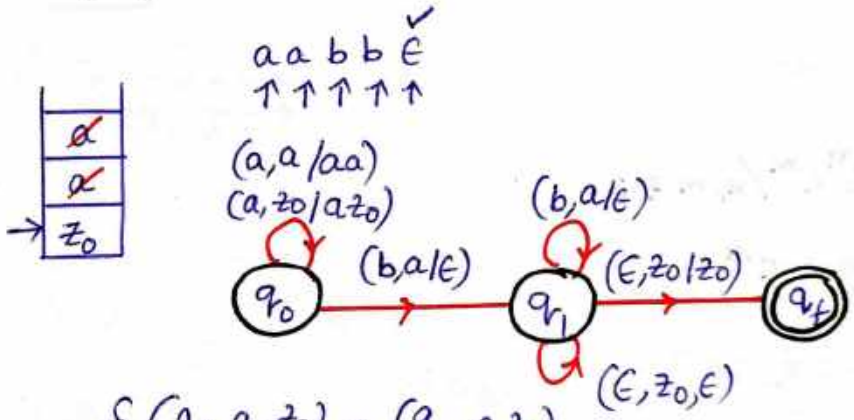
Γ : Stack alphabet

DPDA: $\delta: Q \times \{\Sigma \cup \epsilon\} \times \Gamma \rightarrow Q \times \Gamma^*$

NPDA: $\delta: Q : \{\Sigma \cup \epsilon\} \times \Gamma \rightarrow 2^{(Q \times \Gamma^*)}$

Ex: $a^n b^n / n \geq 1$

→



$\delta(q_0, a, z_0) = (q_0, a z_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_1, \epsilon)$

$\delta(q_1, b, a) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, z_0) = (q_f, z_0)$ → acceptance by final state.

(81) (q_1, ϵ) → acceptance by empty stack.

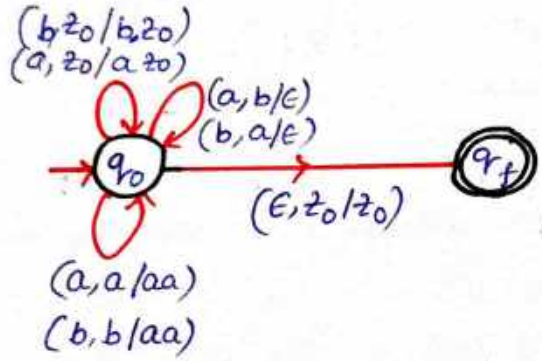
NOTE:

→ Both DPDA and NPDA are equivalent in power.

→ Both final state acceptance and empty stack acceptance are equivalent in power.

Ex: $W / n_a(w) = n_b(w) \; / a^n b^n / n \geq 1$

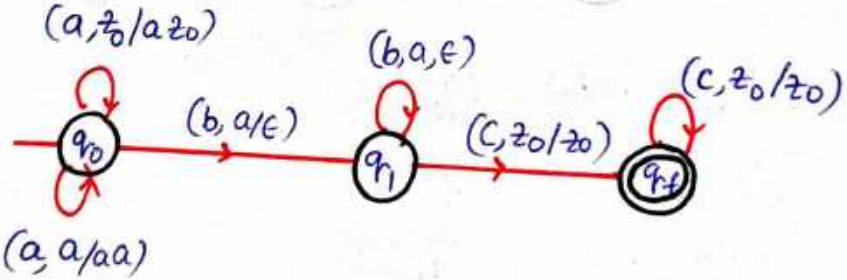
→



a b b b a a ∈
↑ ↑ ↑ ↑ ↑ ↑

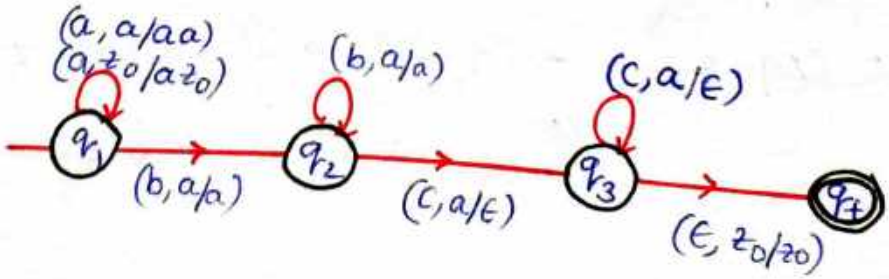
Ex: $a^n b^n c^m / n, m \geq 1$

→ $\underline{a^n b^n} / c^m$



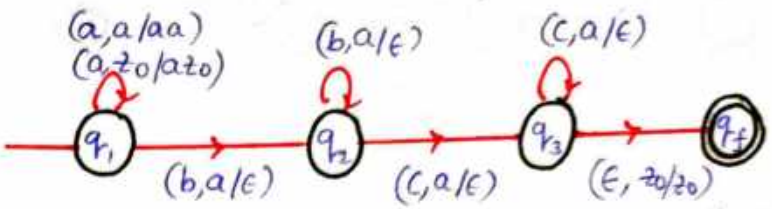
Ex: $a^n b^m c^n / n, m \geq 1$

→



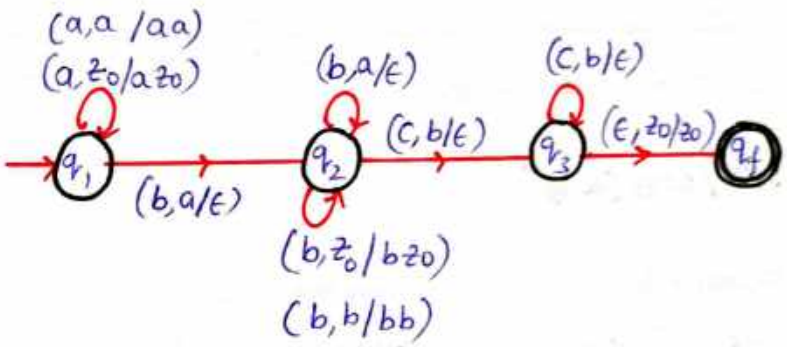
Ex: $a^{m+n}\, b^m\, c^n \;/\; m,n \geq 1$

→



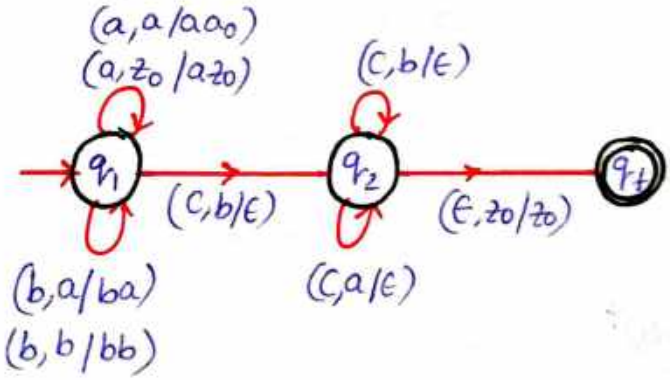
Ex: $a^n\, b^{m+n}\, c^m \;/\; n,m \geq 1$

→



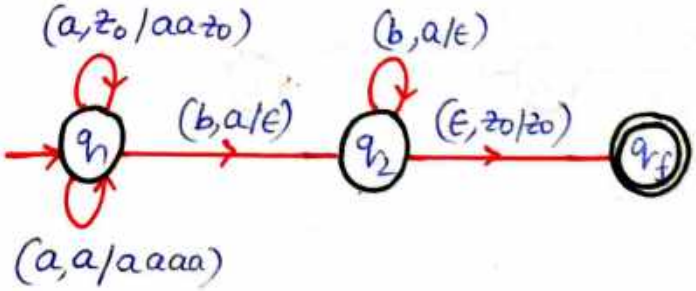
Ex: $a^n\, b^m\, c^{n+m} \;/\; n,m \geq 1$

→

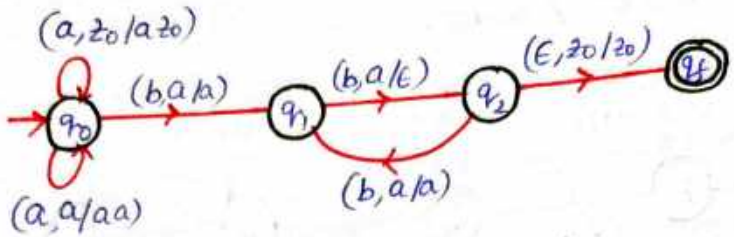


Ex: $a^n b^{2n} \;/\; n \geq 1$

→



(8)



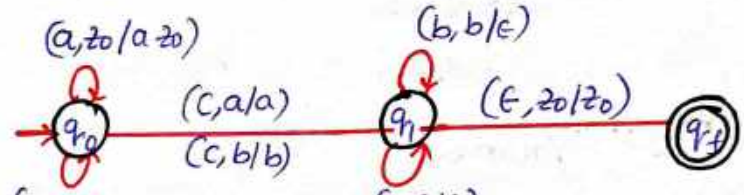
Ex: $a^n b^n c^n / n \geq 1$

→ The above language is NOT CFL.

∴ we cannot Give any pushdown automata to the above language.

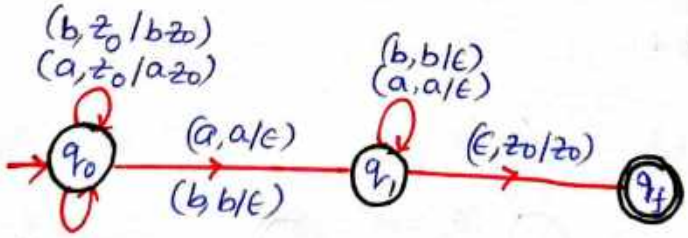
Ex: $WCW^R / w \in (a,b)^+$

→



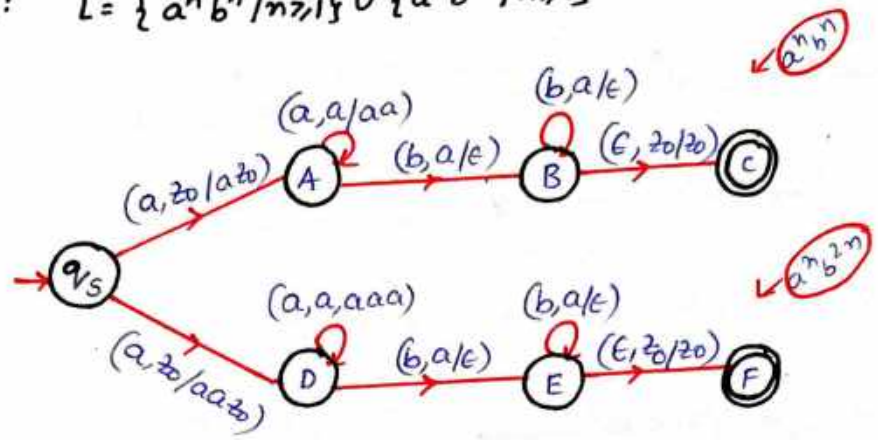
(b, z0/b z0)
(a,a/aa)
(b,a/ba)
(a,b/ab)
(b,b/bb)

Ex: $WW^R / w \in (a,b)^+$



(a, b/ab)
(b,a/ba)
(a, a/aa)
(b, b/bb)

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



EXAMPLES:

1) $a^{m+n} b^n c^m / n, m \geq 1$ ⟶ RL (✗), DCFL (✓), CFL (✓)

2) $a^m b^{m+n} c^n / n, m \geq 1$ ⟶ RL (✗), DCFL (✓), CFL (✓)

3) $a^m b^n c^{m+n} / n, m \geq 1$ ⟶ RL (✗), DCFL (✓), CFL (✓)

4) $a^m b^m c^n d^n / m, n \geq 1$ ⟶ RL (✗), DCFL (✓), CFL (✓)

5) $a^m b^n c^m d^n / m, n \geq 1$ ⟶ RL (✗), CFL (✗)

6) $a^m b^n c^n d^m / m, n \geq 1$ ⟶ RL (✗), DCFL (✓), CFL (✓)

7) $a^m b^i c^m d^k / m, n \geq 1$ ⟶ RL (✗), DCFL (✓), CFL (✓)

8) $a^m b^n / m \geq n$ ⟶ RL (✗), DCFL (✓), CFL (✓)

9) $a^n b^{2n} / n \geq 1$ ⟶ RL (✗), DCFL (✓), CFL (✓)

10) $a^n b^{n^2} / n \geq 1$ ⟶ RL (✗), CFL (✗)

11) $a^n b^{2^n} / n \geq 1$ ⟶ CFL ✗, RL ✗

12) $WW^R / w \in (a,b)^*$ ⟶ RL ✗, DCFL ✗, CFL ✓

13) $WW / w \in (a,b)^*$ ⟶ RL ✗, CFL ✗

14) $a^n b^n / c^m / n > m$ ⟶ RL ✗, CFL ✗

15) $a^n b^n c^n d^n / n \leq 10^{10}$ ⟶ RL ✓, DCFL ✓, CFL ✓

16) $a^n b^{2n} c^{3n} / n \geq 1$ ⟶ RL ✗, CFL ✗

17) $x \subset y / x, y \in (0,1)^*$ ⟶ RL ✓, DCFL ✓, CFL ✓

18) $xx^R / x \in (a,b)^*, |x| = l$ ⟶ RL ✓, DCFL ✓, CFL ✓

19) $WWW^R / w \in (a,b)^*$ ⟶ RL ✗, CFL ✗

20) $a^n b^{3^n} / n \geq 1$ ⟶ RL ✗, CFL ✗

21) $a^m b^n / m \neq n$ ⟶ RL ✗, CFL ✓, DCFL ✓

22) $a^m b^n / m = 2n+1$ ⟶ RL ✗, DCFL ✓, CFL ✓

23) $a^i b^j / i \neq 2j+1$ ⟶ RL ✗, DCFL ✓, CFL ✓

24) $a^{n^2} / n \geq 1$ ⟶ RL ✗, CFL ✗

25) $a^{2^n} / n \geq 1$ ⟶ RL ✗, CFL ✗

26) $a^{n!} / n \geq 1$ ⟶ RL ✗, CFL ✗

27) a^m / m is a prime ⟶ RL ✗, CFL ✗

28) a^k / k is even ⟶ RL ✓, CFL ✓

29) $a^i b^j c^k$ / i > j > k ⟶ RL ✗, CFL ✗

30) $a^i b^j c^k$ / j = i + k ⟶ RL ✗, DCFL ✓, CFL ✓

31) $a^i b^j c^k d^l$ / i = k & j = l ⟶ RL ✗, DCFL ✓, CFL ✓

32) $a^i b^j c^k d^l$ / i = k and j = l ⟶ RL ✗, CFL ✗

33) $a^m b^l c^k d^n$ / m, l, k, n ≥ 1. ⟶ RL ✓

34) $a^n b^{4m}$ / n, m ≥ 1 ⟶ RL ✓, CFL ✓

35) $\{a^3, a^8, a^{13}, \text{-----}\}$ ⟶ RL ✓, CFL ✓

36) $\{a^{2n+1} / n \geq 1\}$ ⟶ RL ✓, CFL ✓

37) $\{a^{n^n} / n \geq 1\}$ ⟶ RL ✗, CFL ✗

38) $\{\omega / \omega \in \{a,b\}^*, |\omega| \geq 100\}$ ⟶ RL ✓, CFL ✓

39) $\{\omega / \omega \in \{a,b,c\}^*, n_a(\omega) = n_b(\omega) = n_c(\omega)\}$ ⟶ RL ✗, CFL ✗

40) $\{\omega / \omega \in \{a,b\}, n_a(\omega) \geq n_b(\omega) + 1\}$ ⟶ RL ✗, CFL ✓

11. TESTING WHETHER LANGUAGE IS CONTEXT FREE LANGUAGE

PUMPING LEMMA FOR CONTEXT FREE LANGUAGES:

If L be a CFL and $w \in L$ such that $|w| \geq n$, for some positive Integer n, the w can be decomposed as $w = abcde$ such that

(i) $bd \neq \epsilon$

(ii) $|bcd| \leq n$

(iii) For all $i \geq 0$ the string $ab^i cd^i e$ is also in L

NOTE:

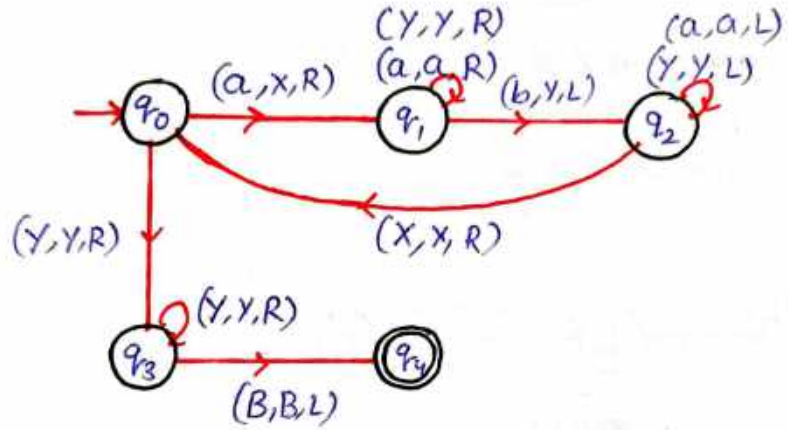
1. Every CFL ~~sta~~ satisfies pumping lemma property
2. If a language L does not satisfy the pumping lemma property then L is not CFL
3. If a language L Statisfies the pumping lemma property the L need not be CFL.
4. Every Context free language Satisfies the pumping lemma property.

12. TURING MACHINES

Ex: --- B B a a a b b b B B B ---- $a^n b^n / n \geq 1$

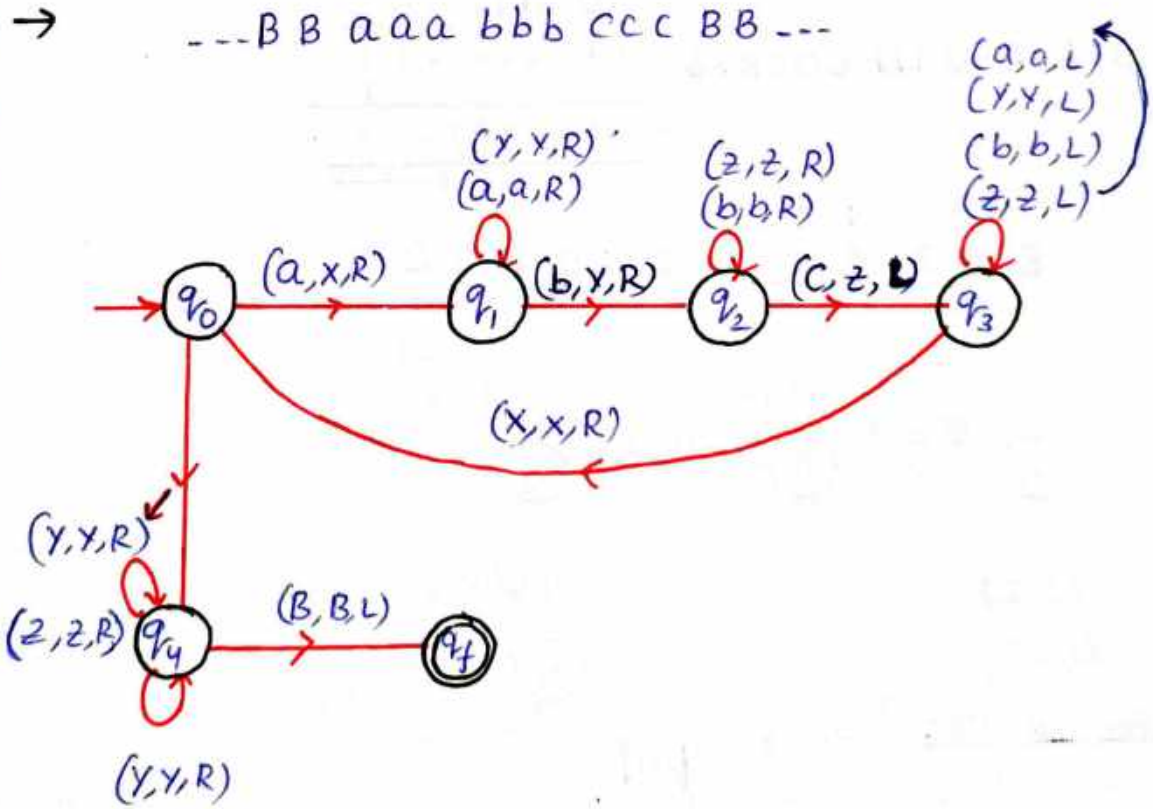
(B = BLANK)

→ ---- B B ~~a~~ ~~a~~ ~~a~~ ~~b~~ ~~b~~ ~~b~~ B B ---



Ex: $a^n b^n c^n / n \geq 1$

→ ---B B a a a b b b c c c B B ---



TRANSDUCER:

1'S COMPLEMENT: TM to find 1's complement of a binary number.

Ex: 0 0 1 1

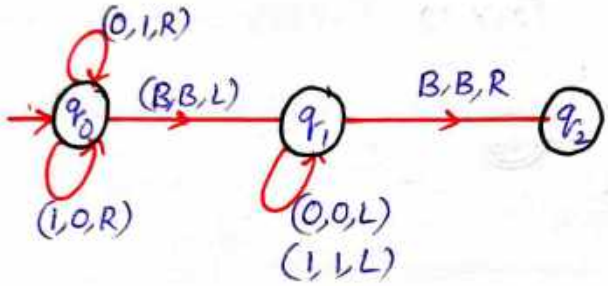
→ 1 1 0 0

Ex: B B B 0 0 1 1 B B B

→

1 1 0 0

B B B ~~0 0 1 1~~ B B B



2'S COMPLEMENT:

Ex: B B B 0 1 1 1 0 0 0 B B B

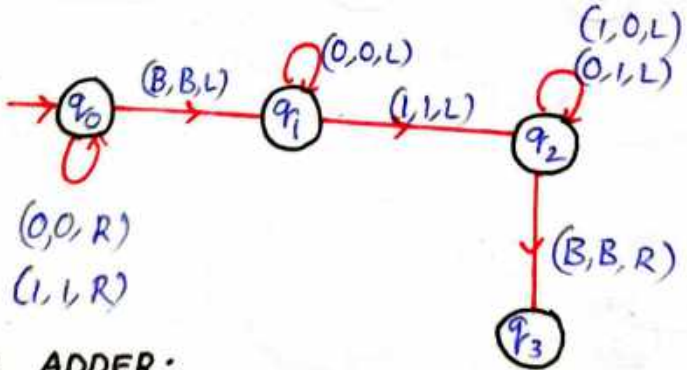
0 0 1 1 1 0 0 0

1 0 0 0 1 1 1

1 0 0 1 0 0 0

1 0 0

B B B ~~0 1 1~~ 1 0 0 0 B B B



TM AS ADDER:

3 = |1 1 1|

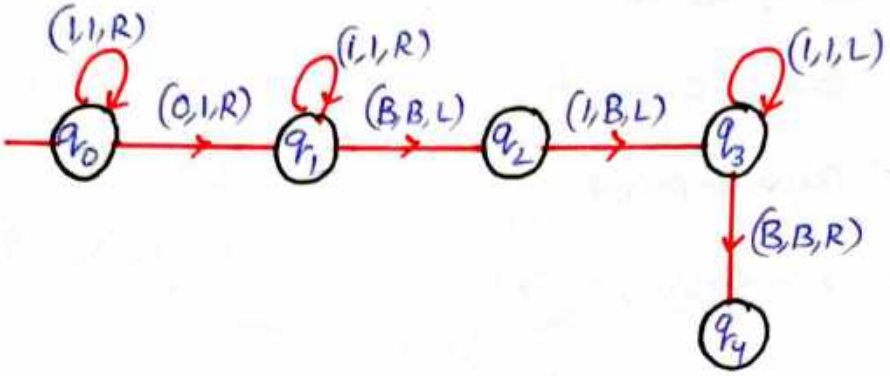
4 = |1 1 1 1|

7 = 1 1 1 1 1 1 1

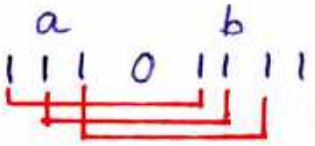
Ex: B B B 1 1 1 0 1 1 1 1 B B B

→ B B B 1 1 1 0 1 1 1 1 B B B

B B B 1 1 1 1 1 1 1 1 B B B



TM AS COMPARATOR:



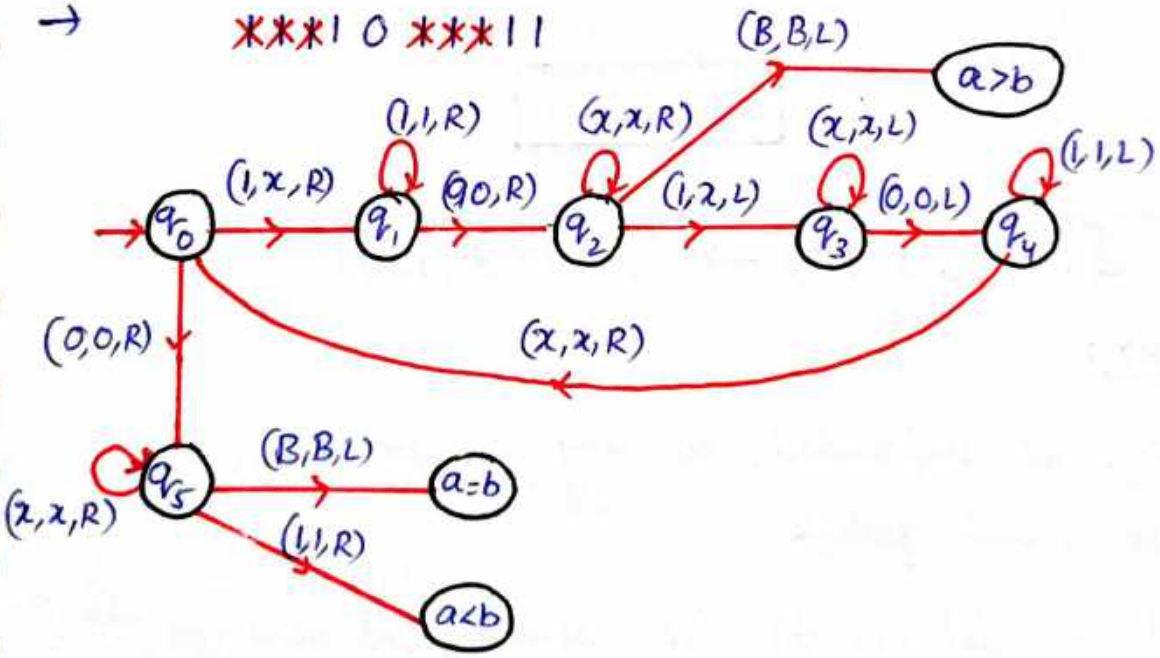
$a = b$

$a > b$

$a < b$

Ex: 1 1 1 1 0 1 1 1 1 1

→ * * * 1 0 * * * 1 1



NOTE: TM can perform any Mathematical fun

THE STANDARD TURING MACHINE:

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

Q - Set of States

Σ - Input alphabets

Γ - Tape alphabet

δ - Transition function

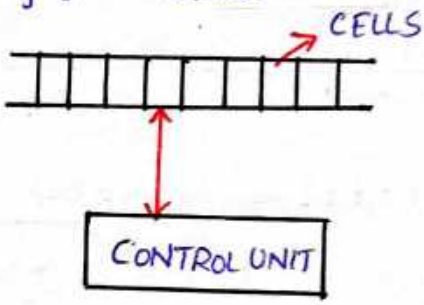
$B \in \Gamma$ - Is used to rep blank

$q_0 \in Q$ - Intial state

$F \subseteq Q$ - set of final states

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

$\Gamma = \{a, b, X, Y, B\}$



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

SUMMARY:

1) Tape is unfounded, so any number of left and right moves possible
2) It is deterministic, i.e atmost one move for each configuration.
3) No special i/p & o/p file.

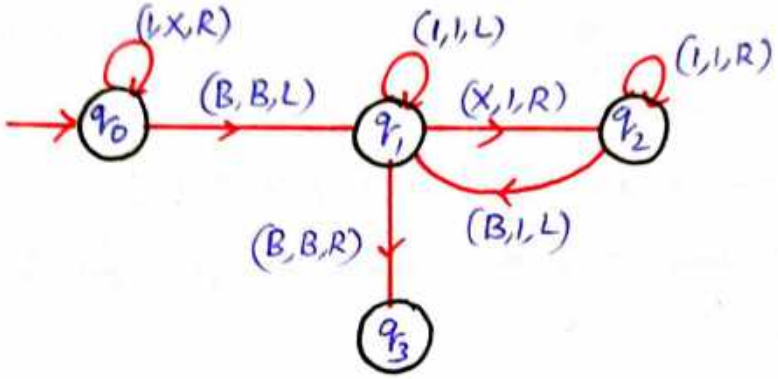
TM AS COPIER:

Ex: B B B 1 1 1 B B B

→ ...B B B 1 1 1 B B B...

⇒ ...B B B x x x B B B...

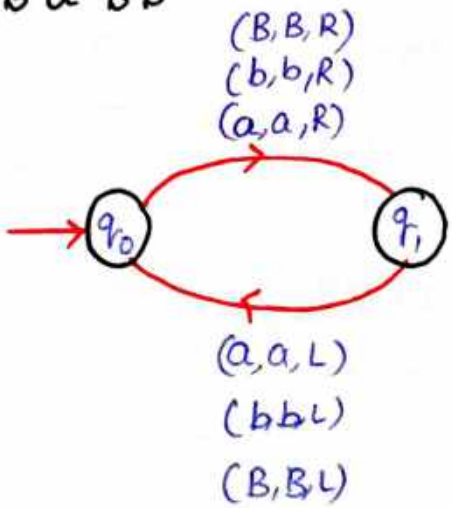
⇒ B B B 1 1 1 1 1 1 B B B



NON - HALTING TM:

Ex: B B a B B

→



TURING THESIS:

Any Computation that can be carried out by mechanical means can be performed by some turing Machine.

Some arguments why turing thesis is accepted

as definition of Mechanical Computation or Computer.

a) Anything that can be done by existing digital Computer can also be done by 'TM'.

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

c) Alternative models have been proposed for Mechanical Computation, but none be of them are powerful than the Turing Machine Model.

SOME MODIFICATIONS TO STANDARD TURING MACHINE:

1) TM with stay option.
2) TM with semi infinite tape
3) Offline TM
4) Multitape TM

$$\delta : Q \times \Gamma^n \rightarrow Q \times \Gamma^n \times \{L, R\}^n$$

5) Jumping TM
6) Non Earasing TM.
7) Always Writing TM.
8) Multidimensional TM

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, U, D\}$$

9) Multihead TM
10) Automata with a queue.

11) TM with only 3 states.

12) Multiple TM with stay option and atmost 2 states.

13) Non-deterministic TM.

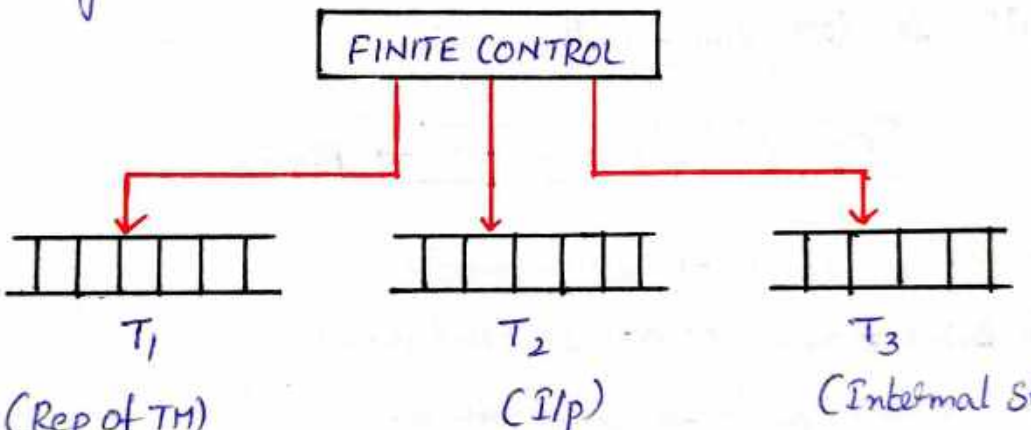
$$\delta: Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L,R\}}$$

14) A NPDA with two Independent stacks.

$$\delta: Q \times (\epsilon \cup (\epsilon)) \times \Gamma \times \Gamma \rightarrow 2^{Q \times \Gamma^* \times \Gamma^*}$$

UNIVERSAL TM AND ENCODING:

Every TM is as powerful as Computer, TM is as powerful as Computer is called Universal TM. It is Biggest Turing Machine.



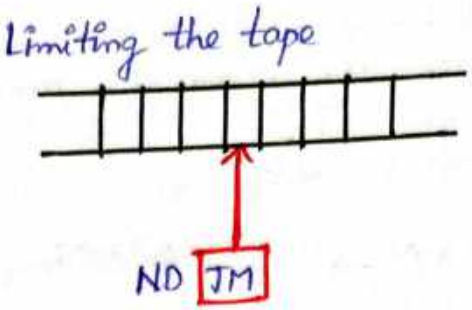
→ The entire Transition function for a Turing Machine can be represented as string of 0's and 1's

→ A Turing Machine can be used as Computer. And as powerful as a digital Computer.

→ Every Turing Machine can be represented as a string of 0's and 1's

→ Not every string of 0's and 1's might be a Turing Machine.

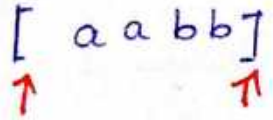
LINEAR BOUNDED AUTOMATA (LBA):



NOTE:

NTM + Tape (Stack) = PDA

TM + Tape (Finite) = FA



LBA = TM + Tape (input)

→ LBA is less powerful than LBA

→ FA < PDA < LBA < TM → POWER

→ (i) DFA = NFA → Equal power.

(ii) DPDA ≠ NPDA → Not In Equal power.

(iii) DLBA, NLBA → Undecidable.

(iv) DTM = NTM → Equal power.

EXAMPLES:

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) $L = \{a^n : n \text{ is a prime}\}$

5) $L = \{a^n : n \text{ is not a prime}\}$

6) $L = \{ww : w \in \{a,b\}^+\}$

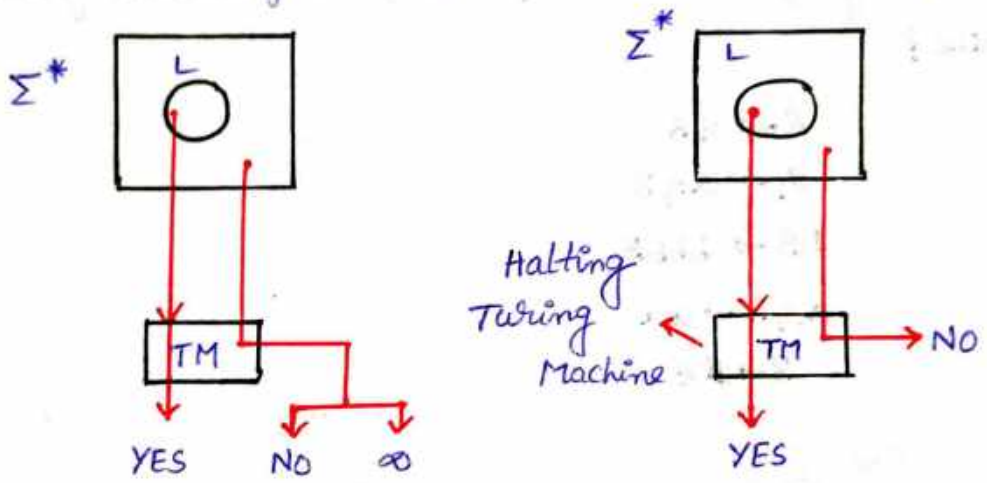
7) $L = \{\omega^n : \omega \in \{a,b\}^+, n>1\}$

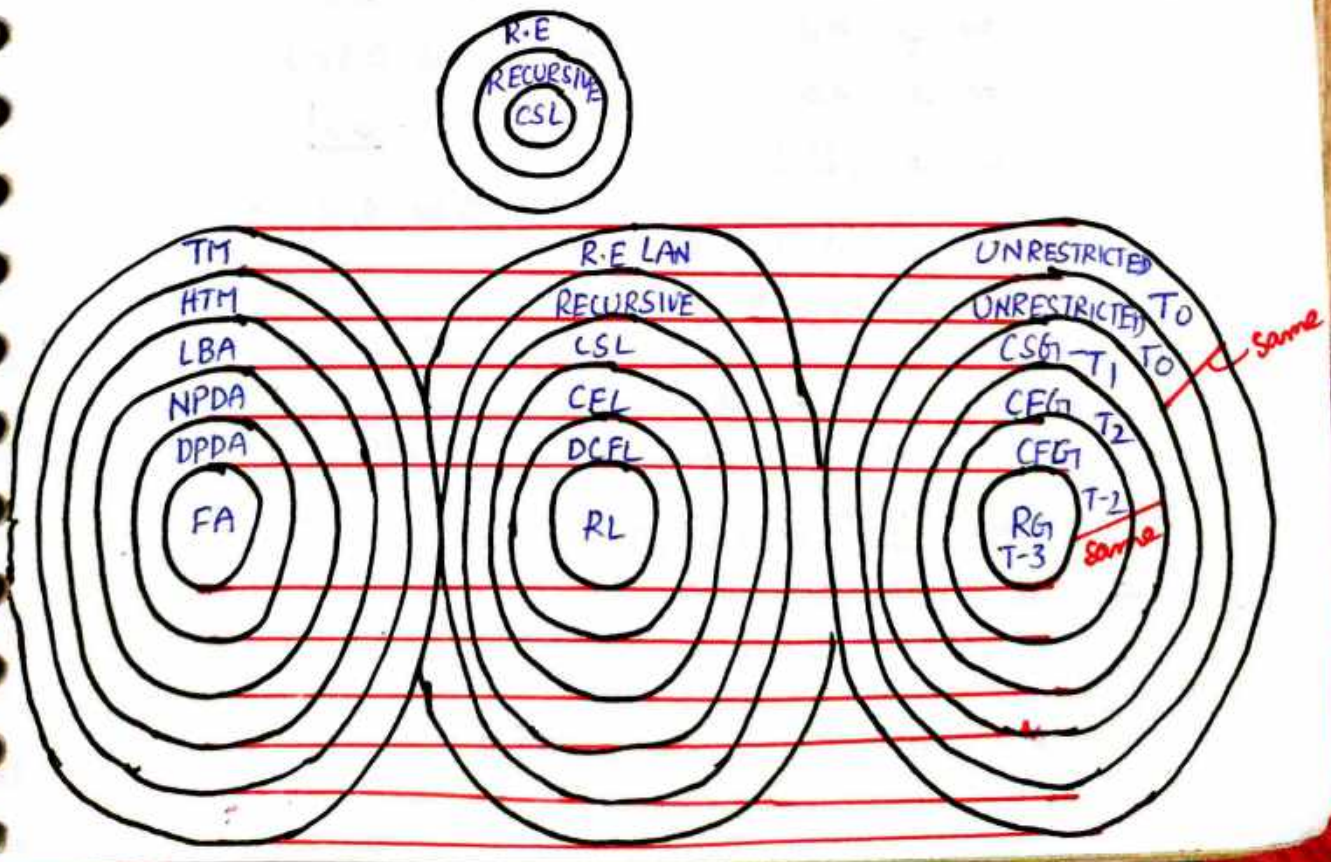
8) $L = \{ww w^R : w \in \{a,b\}^+\}$

RECURSIVELY ENUMERABLE AND RECURSIVE LANGUAGE:

Turing Machine
- Recursively Enumerable

Halting Turing Machine
- Recursive language.





UNRESTRICTED GRAMMARS:

A grammar is called unrestricted if all the productions are of the form $u \to v$

Where u is in $(V \cup T)^+$

v is in $(V \cup T)^*$

$TM \to R$

Ex: What language does the following unrestricted grammar derive?

$S \to S_1 B$

$S_1 \to a S_1 B$

$bB \to bbbB$

$aS_1 b \to aa$

$B \to \epsilon$

→

$S \Rightarrow S_1 B$

$\Rightarrow a S_1 bB$

$\Rightarrow a^2 S_1 b^2 B$

$\Rightarrow a^n S_1 b^n B$

$\Rightarrow a^{n-1} a S_1 b b^{n-1} B$

$\Rightarrow a^{n-1} aab^{n-1} B$

$\Rightarrow a^{n+1} b^{n-1} B$

$\Rightarrow a^{n+1} b^{n-2} bB$

$\Rightarrow a^{n+1} b^{n-2} b^2 bB$

$\Rightarrow a^{n+1} b^{n-2} b^k$ $\quad k = 1, 3, 5, \ldots$

$\Rightarrow a^{n+1} b^{n-2+k}$ $\quad n = 1, 2, 3, \ldots$

$\Rightarrow a^{n+1} b^{n+k-2}$

$\Rightarrow a^{n+1} b^{n+m}$ $\quad m = -1, 1, 3, \ldots$

$S \Rightarrow S_1 B$

$\Rightarrow aS_1 bB$

$\Rightarrow \boxed{aa}$

$S \Rightarrow S_1 B$

$\Rightarrow aS_1 bB$

$\Rightarrow aaS_1 bbB$

$\Rightarrow \boxed{aaab}$

CONTEXT SENSITIVE GRAMMARS:

A grammar is said to be Context Sensitive if all the productions are of form

$$x \to y$$

Where $x, y \in (V \cup T)^+$ and $|x| \leq |y|$

Ex: What is language generated by the following CSL:

$S \to abc \,/\, aAbc$

$Ab \to bA$

$Ac \to Bbcc$

$bB \to Bb$

→

$S \Rightarrow aAbc$

$\Rightarrow abAc$

$\Rightarrow abBbcc$

$\Rightarrow aBbbcc$

$\Rightarrow aabAbcc$

$\Rightarrow aabbAcc$

$\Rightarrow aabbBbccc$

$\Rightarrow aabBbbccc$

$\Rightarrow aaBbbbccc$

$\Rightarrow aaabbbccc$

Ex: $a^n b^m c^n d^m / n, m \geq 1$

→

$S \to aAcD \,/\, aBcD$

$A \to aAc \,/\, aBc$

$Bc \to cB$

$Bb \to bB$

$BD \to Ed$

$cE \to Ec$

$bE \to Eb$

$aE \to ab$

$BD \rightarrow FDd$

$CF \rightarrow Fc$

$bF \rightarrow Fb$

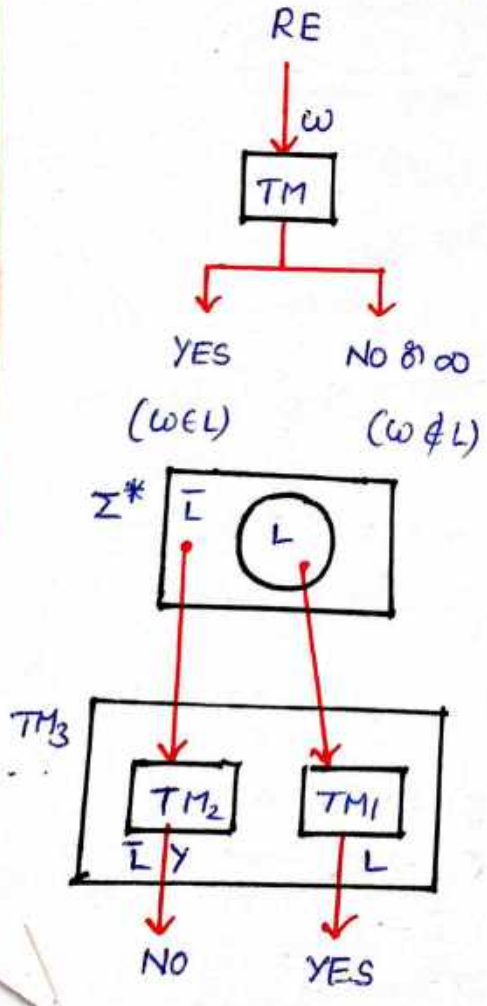
$aF \rightarrow abB$

THEOREM ON RECURSIVE AND RE LANGUAGES:

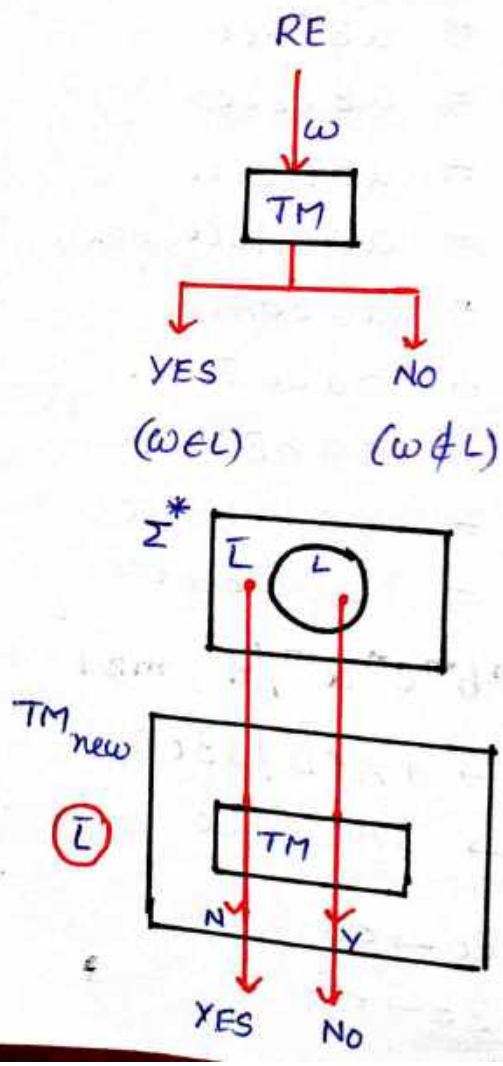
THEOREM:

If a Language L and its Complement $\bar{L}$ are both recursively enumerable, ~~them~~ then both Languages are recursive. If L is recursive, then $\bar{L}$ is also recursive and Consequently both are Recursively enumerable.

No Membership algo



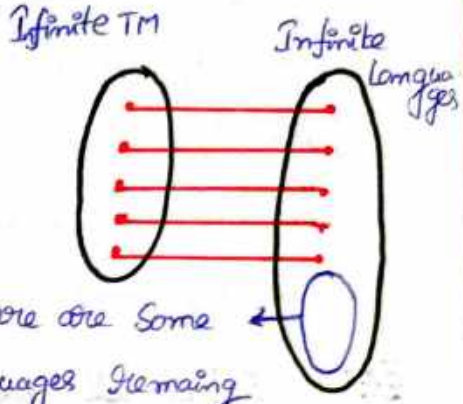
Membership algo



13. COUNTABILITY

INTRODUCTION:

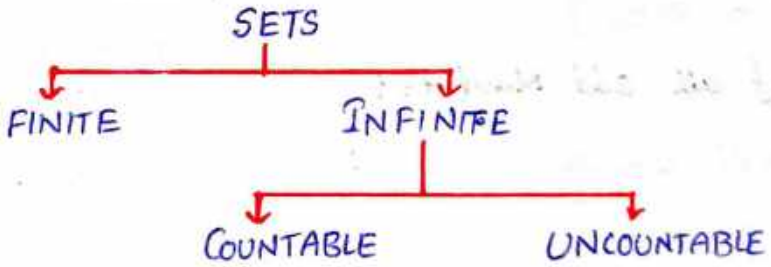
Q: For every language is there a TM accepting?



→ There are some languages remaing for which we will never be able to give a T.M

→ using this we can prove that T.M is not so powerful that it can accept everything.

→



COUNTABLE:

A Set 'S' is said to be Countable, if all 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.

$$F < C < U$$

ALTERNATIVE DEFINITION OF COUNTABILITY:

A Set is said to be Countable if there exists an Enumerable Method using which all the elements of the Set any can be generated and for any particular element, it takes only finite number of steps to generate it. The finite number of steps taken to generate an element can be used as its Index and hence a Mapping into natural numbers.

Ex: Set of all even Numbers

→

```
for (i = 0 to ∞)
  {
  generate (2i)
  }
```

0, 2, 4, 6, 8, ...
① ② ③ ④ ⑤ .....

COUNTABLE

Ex: Set of all odd Numbers?

→

```
for (i = 0 to ∞)
  {
  generate (2i+1)
  }
```

1, 3, 5, 7, 9, 11, ...
① ② ③ ④ ⑤ ⑥ .....

COUNTABLE

Ex: Set of all quotients

→ "P/q", P, q ∈ Z_+

$S = \{ 1/1, 1/2, 2/1, 1/3, 1/4, 3/4 \ldots\ldots \}$

② ③ ④ ⑤

$1/1$, $2/1$, $1/2$, $1/3$, $2/2$, $3/1$, $1/4$, ----

∴ Every element could be generated after fixed Number of steps, and the finite Number of steps going to be the Index for the element. And that Index could be used as the Mapping into the Natural Numbers.

→ The set of all Natural Numbers Mapped to the all Numbers of form P/q, therefore these Numbers are Countable.

Ex: **Set of all strings over any finite alphabet are Countable.**

→ $\Sigma = \{a, b\}$

$\Sigma^* = \{ \epsilon, a, b, aa, ab, ba, bb ---- \}$

$\{ \epsilon, a, b, aa, ab, ba, bb, aaa, aab, aba ------ \}$

① ② ③ ④ ⑤ ⑥ ------ proper order

∴ Every Number is Mapping into the Natural Number we can say that this set is Countable.

→ The set of all strings possible over any alphabet is always Countable.

NOTE:

→ Every Subset of Countable set is either finite or Countable.

Ex: **Set of all Turing Machines are Countable**

→ $\Sigma = \{0, 1\}$

(i) $\Sigma^* = \{ \epsilon, 0, 1, 00, 01, 10, 11, ------ \}$ → COUNTABLE

(iii) Set of all turing machines, $S \subseteq \Sigma^*$.

(iv) Every subset of countable set is either finite or countable.

$\therefore$ Using all these rules, we can prove that Set of all Turing Machines are countable.

IMPLICATIONS OF THE FACT THAT THE TURING MACHINS ARE COUNTABLE:

→ Turing Machines are Countable.
→ REL are Countable.
→ REL are Countable.
→ CSL are Countable.
→ CFL are Countable.
→ RL are Countable.
→ LBA are Countable.
→ PDA are Countable.
→ FA are Countable.

DIAGNOLIZATION METHOD TO PROVE THAT SET OF ALL LANGUAGES ARE COUNTABLE:

$\Sigma = \{a, b\}$, Σ^* is Countable, 2^{Σ^*} is uncountable

Σ^*	ϵ	a	b	aa	ab	ba	aaa----	
1)	0	1	1	01	0	0	0 ---	
2)	1	0	0	0	0	1	0 ----	00010---
3)	0	1	0	1	0	1	0 ----	11101---
4)	0	0	0	1	0	0	0 ----	Complement
5)	1	0	0	0	0	0	0 ---	
:								

→ 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 number of Countable sets is Countable.

→ The set of all Languages that are Recursively Enumerable is uncountable.

14. COMPUTABILITY AND DECIDABILITY

INTRODUCTION:

COMPUTABILITY:

If there is function which is defined on Domine. If our Turing Machine able to Solve that function, for all the Inputs Comming form the domine. It is definitly going to Halt then Such a function is Called Computability.

EX: function $f(n) = n^2+1$

algo & TM (Halts) ✓

Tape

DECIDABILITY:

PROBLEM: A Statement whose output will be either True & False.

EX: If 'n' a prime.

D = set of all natural Numbers.

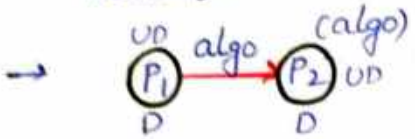
Whenever we Have Domain, Given any Input from the Domain to a Turing Machine. That Turing Machine Definitly Halts and Say Yes & No for Such a problem. If there exist Such a Turing Machine then we Can Say that the problem is decidable.

EX: A given grammar 'G' is ambiguous.

D = Set of all CFGs

Undecidable.

NOTE: → Given an instance of a problem, it is always decidable.



(i) If P_2 is decidable then P_1 is decidable.

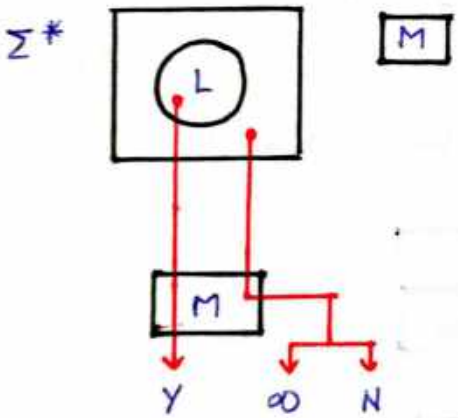
(ii) If P_1 is already proven to be undecidable, then definitely P_2 must be undecidable.

TM HALTING PROBLEM:

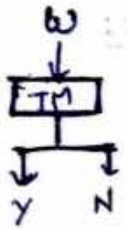
Given the description of a TM 'm' and an i/p 'w', does 'm' when started with 'w' as its i/p eventually halts?

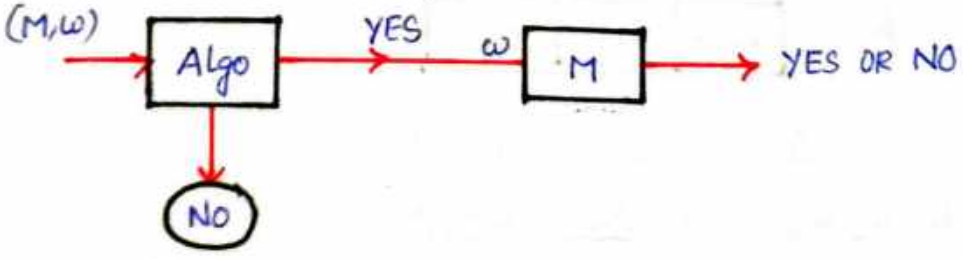
THEOREM:

If the Halting problem were decidable, then every RE language would be recursive. Consequently, the Halting problem is undecidable.



R.E	R
TM	TM which Halts





SOME UNDECIDABLE PROBLEMS BASED ON TM HALTING PROBLEM

→ 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.

→ Given a TM M, whether or not M halts if started with a blank tape this is undecidable.

→ Almost any problem related to RE language is undecidable.

POST CORRESPONDENCE PROBLEM:

Given two sequences of 'n' strings on some alphabet Σ, say $A = w_1, w_2 \cdots w_n$ and $B = v_1, v_2 \cdots v_n$, we say there exists a PC-solution for pair (A,B) if there is a non empty sequence of integers $i, j, \cdots k$ such that

$$w_i, w_j, \cdots w_k = v_i, v_j \cdots v_k$$

PC problem is to devise an algorithm that will tell us for any (A,B), whether or not there exist a PC-solution.

Ex:

$A =$

w_1	w_2	w_3
a	ab	bba

$B =$

v_1	v_2	v_3
baa	aa	bb

Solution is (3,2,3,1)

$w_3 w_2 w_3 w_1$ = bbaabbbaa

$v_3 v_2 v_3 v_1$ = bbaabbbaa

COMPLEXITY CLASSES:

P-CLASS: The set of all languages that are accepted by some deterministic TM in polynomial time. $O(n^k)$

NP-CLASS: The set of all languages accepted by non-deterministic TM in polynomial time is NP.

DECIDABILITY:

PROBLEM	RL (Regular)	DCFL	CFL	CSL	RL (Recursive)	REL
1) Does $\omega \in L$? (Membership problem)	D	D	D	D	D	UD
2) Is $L = \emptyset$? (Emptiness problem)	D	D	D	UD	UD	UD
3) Is $L = \Sigma^*$? (Completeness problem)	D	UD	UD	UD	UD	UD
4) Is: $L_1 \subseteq L_2$? (Subset problem)	D	UD	UD	UD	UD	UD
6) $L_1 \cap L_2 = \emptyset$	D	UD	UD	UD	UD	UD
7) Is L is finite or not? (finiteness)	D	D	D	UD	UD	UD
5) Is $L_1 = L_2$? (Equality problem)	D	UD	UD	UD	UD	UD
8) Is Complement of 'L' a language of same type or not?	D	D	UD	D	D	UD
9) Is intersection of two Languages of same type	D	UD	UD	UD	UD	UD
10) Is L Regular language	D	D	UD	UD	UD	UD

15. PROPERTIES OF CFLs

CLOSED UNDER UNION, CONCAT AND KLEENE CLOSURE:

UNION

L_1 L_2

$\downarrow$ $\downarrow$

CFG_1 CFG_2

$S_1 \rightarrow$ $S_2 \rightarrow$

$S \rightarrow S_1 / S_2$

S_1

S_2 ✓

CONCATENATION

$L_1 \cdot L_2$

$S \rightarrow S_1 \cdot S_2$

$S_1 \rightarrow$

$\rightarrow$

$S_2 \rightarrow$

$\rightarrow$ ✓

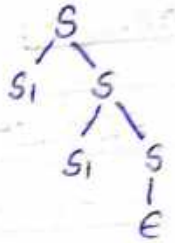
KLEENE CLOSURE

L_1^*

$\downarrow$

S_1

$S \rightarrow S_1 S / \epsilon$



✓

closed under → UNION
→ CONCATENATION
→ KLEENE CLOSURE

NOT CLOSED UNDER INTERSECTION AND COMPLEMENTATION:

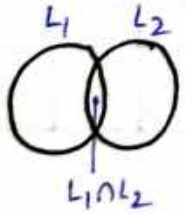
INTERSECTION

$L_1 \cap L_2$ ✗

$L_1 = \{a^n b^n c^m / n, m \geq 0\}$

$L_2 = \{a^m b^n c^n / n, m \geq 0\}$

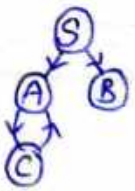
$L_1 \cap L_2 = \{a^n b^n c^n / n \geq 0\}$



COMPLEMENTATION

$L_1 \cap L_2 = \bar{L_1} \cup \bar{L_2}$

CFLs are Not closed under Complementation



Cycle

∴ The grammar is infinite

DECIDABLE PROBLEM ON CFL:

MEMBERSHIP

$w \in L$

EMPTINESS

CFG ✓

↓

SIMPLIFICATON ✓

✗ S (Start Symbol) (useless)

Then Grammar is EMPTY

FINITENESS

CFG

↓

SIMPLIFICATION

$S \rightarrow AB$

$A \rightarrow AC / a$

$C \rightarrow aA / b$

$B \rightarrow a$

S ⇒

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow b$

Grammar Having No cycle is Finite

16. PROPERTIES OF REGULAR LANGUAGES

PROPERTIES OF RL:

→ RL closed under - UNION
- INTERSECTION
- CONCATENATION
- COMPLEMENTATION
- KLEEN CLOSURE

→ RL languages are closed under Difference.

→ RL languages are closed under Reversal.

→ The function $h: \Sigma \to \Gamma^*$ is called Homomorphism

RL are closed under homomorphism

Ex: $\Sigma = \{a,b\}$ $\Gamma = \{0,1,2\}$

→ $h(a) = 01$
$h(b) = 112$

$h(aba) = 01112\underline{01}$

$L_1 = a^*b = (01)^*112$

→ RL are closed under Inverse Homomorphism

INVERSE HOMOMORPHISM:

The Inverse of Homomorphic Image of a language L is

$h^{-1}(L) = \{x / h(x) \text{ is in } L\}$

For a String ω, $h^{-1}(\omega) = \{x / h(x) \in L\}$

Ex: Let $\Sigma = \{0,1,2\}$ and $\Delta = \{a,b\}$ Define h by

$h(0) = a$, $h(1) = ab$ $h(2) = ba$

→ Let $L_1 = \{ababa\}$

$h^{-1}(L_1) = \{110, 022, 102\}$

$L_2 = a(ba)^*$

$= \{a, aba, ababa \text{ ----- }\}$

$h^{-1}(L_2) = \{0, 10, 110, \text{ ----- }\}$

02 022

102

EX: Let $\Sigma = \{0,1\}$ and $\Delta = \{a,b\}$ Define 'h' by

$h(0) = aa$, $h(1) = aba$ ~~Let L = (ab+ba)*a~~

→ Let $L = (ab+ba)^*a$

$L = \{a, aba, baa, ababa, abbaa \text{ ---}\}$

$h^{-1}(L) = \{1\}$

$h(h^{-1}(L)) = \{aba\} \neq L$

$\therefore h(h^{-1}(L)) \subseteq L$

RIGHT QUOTIENT:

Let L_1 and L_2 be Language on the same alphabet. Then the right Quotient of L_1 with L_2 is defined as

$L_1/L_2 = \{x : xy \in L_1 \text{ for some } y \in L_2\}$

Ex: $L_1 = \{01, 001, 101, 0001, 1101\}$

$L_2 = \{0,1\}$

→ $L_1/L_2 = \{\epsilon, 0, 1, 00, 11\}$

$L_1 = 1\Sigma^*1$, $L_2 = 0^*1$

Ex: $L_1 = \{11, 101, 1001, 10001 \text{ ---- }\}$

$L_2 = \{1, 01, 001, 0001, \text{ ----- }\}$

$L_1/L_2 = \{1, 10, 100, 1000 \text{-----}\}$

$= 10^*$

INIT OPERATION:

Regular sets are closed under INIT operation

Ex: $L = \{a, ab, aabb\}$

$Init(L) = \{\epsilon, a, \epsilon, a, ab, \epsilon, a, aa, aab, aabb\}$

SUBSTITUTION:

Regular sets are closed under Substitution

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

$f(a) = 0^*$

$f(b) = 01^*$

→ $L = a + b^*$

$f(L) = 0^* + (10)^*$

NOTE:

→ Regular languages are not closed under infinite union of regular Languages.

17. DECIDABLE PROBLEMS ON REGULAR LANGUAGES

EMPTINESS PROBLEM:

Emptiness problem of FA is decidable.

ALGORITHM:

1) Select the states which are not reachable from intial state and delete those states and corresponding transitions
2) In the remaining FA, if we find at least one final state, then the language accepted by given FA is Non empty otherwise empty.

INFINITENESS PROBLEM:

Infiniteness problem of RL is decidable.

ALGORITHM:

STEP1: Remove the states which are not reachable from Intial state and corresponding transitions.

STEP2: Delete the states and corresponding transitions from which we cannot reach final state.

STEP3: In the remaining FA, if we find at least one loop and one final state, then it is accepting infinite language.

EQUALITY:

Equality problem is decidable.

$L_1 = L_2$

↓ ↓

DFA, DFA2

↓ ↓

(m_1) DFA DFA (m_2)

18. GATE QUESTIONS ON TM AND UNDECIDABILITY

Q) Recursive languages are

✗ a) A proper subset of CFL

✗ b) Always recognizable by PDA

✗ c) Also called Type(0) languages

✓ d) Recognizable by TM.

T(0) – RE
T(1) – CSL
T(2) – CFL
T(3) – RL

Q) In which of the following cases every Non deterministic Machine has an equivalent deterministic machine?

a) FA ✗ b) PDA c) TM d) None

Q) Which of the following conversions is not possible (algorithmically)?

a) Regular grammar to CFG

b) NFA to DFA

✓ c) NPDA to DPDA

d) NTM to DTM

Q) Which one of the following is Not decidable?

a) Given a TM 'M', a string 'S' and an Integer 'k', 'M' accepts 'S' within 'k' steps.

✗ b) Equivalence of two 'TMs'

c) Language accepted by given Finite state Machine is Non empty

d) Language accepted by a CFG is Non-empty.

Q) Which of the following is True?

✓ a) The complement of recursive language is recursive.

b) The Complement of recursively enumerable language is recursively enumerable.

c) The Complement of Recursively enumerable language is either recursive or recursively enumerable.

d) The Complement of CFL is CFL

NOTE: (i) 'L' is CFL, $\bar{L}$ is CFL → **FALSE**

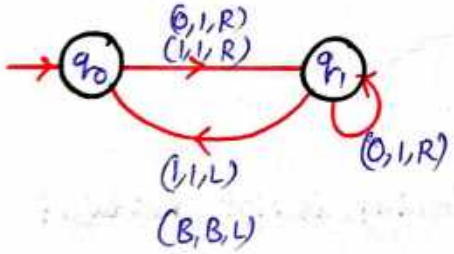
(ii) 'L' is CFL, $\bar{L}$ is Recursive → **TRUE**

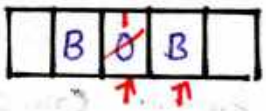
(iii) 'L' is CFL, $\bar{L}$ is Recursively Enumerable → **TRUE**

Q)

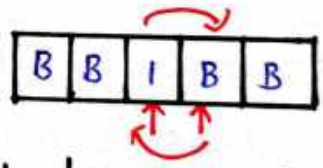
	0	1	B
q_0	$q_1, 1, R$	$q_1, 1, R$	Halt
q_1	$q_1, 1, R$	$q_0, 1, L$	q_0, B, L

→





Not going to Halt in any String '0' and '1'



Q) let 'L_1' be a recursive language, and L_2 be a recursively enumerable language but not R. which of the following is true?

a) $\bar{L}_1$ is recursive and $\bar{L}_2$ is RE

b) $\bar{L}_1$ is R and $\bar{L}_2$ is not RE

c) $\bar{L}_1$ and $\bar{L}_2$ is not RE.

d) $\bar{L}_1$ is RE and $\bar{L}_2$ is R.

Q) If L and $\bar{L}$ are RE then L is

a) Regular
b) CFL
c) CSL
d) Recursive

Q) let 'L' be a language and $\bar{L}$ be its Complement. which one of the following is NOT a viable possibility

a) Neither 'L' nor $\bar{L}$ is RE.
b) one of 'L' and $\bar{L}$ is RE but not recursive, the other is not RE.
c) Both 'L' and $\bar{L}$ are RE but not recursive.
d) Both 'L' and $\bar{L}$ are recursive.

Q) let <M> be the encoding of a Turing Machine as a string over $\Sigma = \{0,1\}$. let $L = \{ <M> / M$ is a turing Machine that accepts a string of length 2014$\}$. Then L is

a) Decidable and Recursively enumerable
b) un-decidable but RE
c) undecidable and not RE
d) Decidable but not RE

Q) Which of the following statement is False

a) The Halting problem for Turing Machine is undecidable
b) Determining whether a CFG is ambiguous is undecidable
c) Given two arbitary CFGs G_1 and G_2 whether $L(G_1) = L(G_2)$ is UD
d) Given two regular grammars G_1 and G_2 if is

undecidable whether $L_1(G_1) = L(G_2)$

Q) Which of the following are decidable?

- ~~a)~~ Whether the intersection of two regular languages is infinite. (D)
- b) Whether a given CFL is regular. (UD)
- c) Whether two PDAs accept the same language. (UD)
- ~~d)~~ Whether a given grammar is context free. (D)

Q) Which of the following problems are decidable?

a) Does a given ~~problem~~ program ever produce an output. (UD)

b) If L is a CFL, then $\bar{L}$ is also CFL. (UD)

~~c)~~ If L is regular, then $\bar{L}$ is also regular. (D)

~~d)~~ If L is recursive language, then is $\bar{L}$ also recursive? (D)

Q) Let Σ' be a finite non-empty alphabet and let 2^{Σ^*} be the power set of Σ^*. Which of the following is true?

a) Both 2^{Σ^*} and Σ^* are countable.

b) 2^{Σ^*} is countable and Σ^* is uncountable.

~~c)~~ 2^{Σ^*} is uncountable and Σ^* is countable.

d) Both are uncountable.

Q) Which of the following are undecidable?

a) Membership problem in CFL (D)

~~b)~~ Whether a given CFL is regular (UD)

c) whether a finite state automation Halts on all Inputs. (D)

~~d)~~ Membership problem for type '0' Language. (UD)

Q) It is decidable Whether

a) An arbitrary TM halts after 100 steps (D)

b) A Turing Machine prints a specific letter. (UD)

c) A Turing Machine Computes product of two numbers (D)

d) None of the above.

Q) Which of the following is strongest Correct statement about a finite language over some finite alphabet Σ?

a) It called be undecidable. ✗

b) It is turing-Machine recognizable. ✓

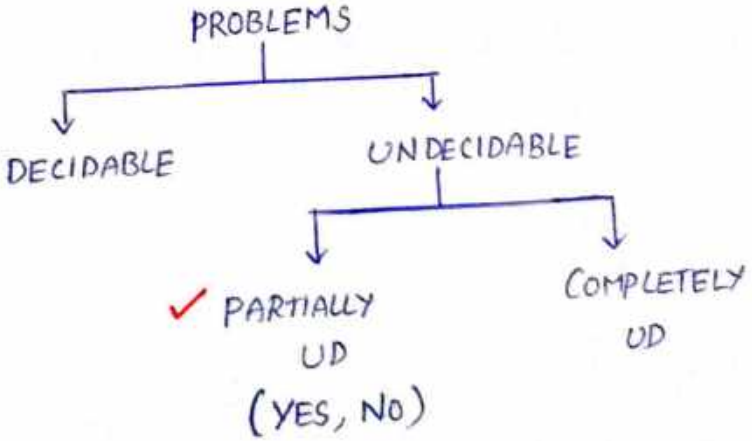
c) It is a regular Language. ✓

d) It is a CFL. ✓

Q) Consider the following problem 'x'. Given a TM 'm' over input alphabet Σ, any state 'q' of 'M' and a word $w \in \Sigma^*$, does the Computation of 'm' on 'w' visit the state of 'q'.

Which of the following statements about 'x' is Correct?

→



Q) Consider three decision problems P_1, P_2 and P_3. It is known that P_1 is decidable and P_2 is un-decidable. Which of the following TRUE?

a) P_3 is decidable if P_1 is reducible to P_3.

b) P_3 is undecidable if P_3 is reducible to P_2.

c) P_3 is undecidable if P_2 is reducible to P_3.

d) P_3 is decidable if P_3 is reducible to P_2's complement.

Q) Which of the following are undecidable.

1) 'G' is a CFG. Is $L(G) = \emptyset$ (D)

2) 'G' is a CFG. Is $L(G) = \Sigma^*$ (UD)

3) 'M' is a turing machine is L(M) regular? (UD)

4) 'A' is a DFA and 'N' is an NFA is $L(A) = L(N)$? (D)

Q) Which of the following is undecidable?

a) Deciding if a given CFG is ambiguous. (UD)

b) Deciding if a given string is generated by a CFG. (D)

c) Deciding if the Language generated by a given CFG is empty. (D)

d) Deciding if the Language generated by a given CFG is finite. (D)