

L-1

FLAT

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

→ String - Ex: $a, b, aa, ab, bc, \dots$

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

→ Language = collection of strings

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

$L_1 = \{\text{set of all strings of length 2}\} \rightarrow \text{Finite Language}$
 $= \{aa, ab, ba, bb\}$

$L_2 = \{\text{set of all strings of length 3}\} \rightarrow \text{Finite Language}$
 $= \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$

$L_3 = \{\text{set of all strings where each string starts with 'a'}\}$
 $= \{a, aa, aaa, ab, abb, aba, \dots\} \rightarrow \text{Infinite Language}$

Powers of Σ

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

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

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

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

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

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

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

$= \{\text{Universal set}\}$

PRACTICAL SCENARIO

(2)

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

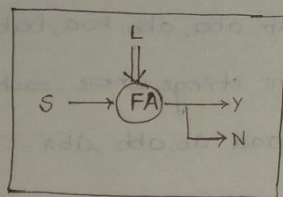
```
Now, void main()
{
    int a, b;
    ;
}
```

program in 'C' but in TMC it is a string

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

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

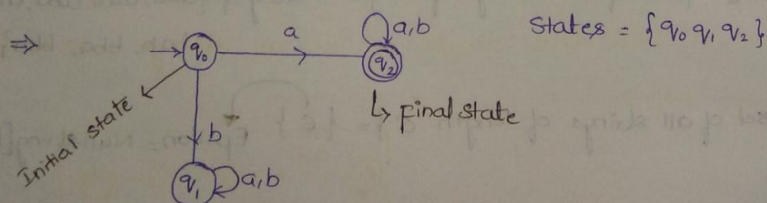
Now, Given any string and to find out whether the string belongs to the language is a heavy task if the language is infinite.



CONSTRUCTION OF FINITE AUTOMATA

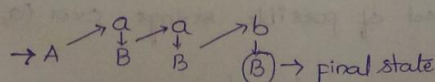
① $L_1 = \{\text{set of all strings which start with 'a'}\}$

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



States = $\{q_0, q_1, q_2\}$

Now, Given string, = a a b

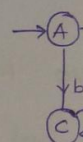


Classification

FA with output
Moore

DFA

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

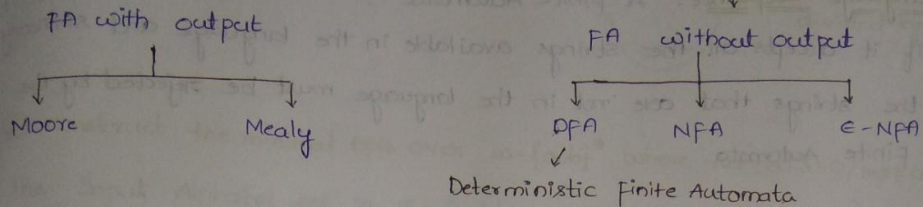


②

Classification of the finite Automata.

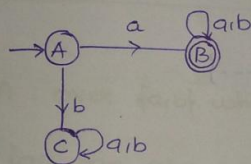
③

Finite Automata



DFA

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



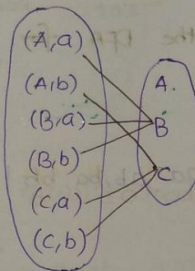
$Q = \{\text{set of all states}\} = \{A, B, C\}$

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

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

$$\text{Now, } \delta: Q \times \Sigma \rightarrow Q$$

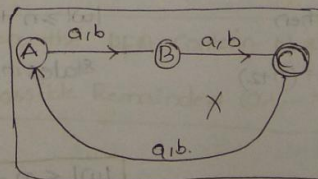
$\{A, B, C\} \times \{a, b\}$



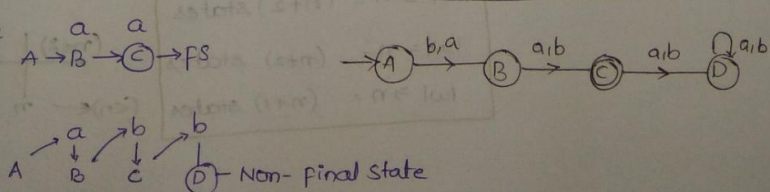
② construct a DFA that accepts set of all strings over $\{a, b\}$ of Length 2? $|w| = 2$.

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

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



Given string



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

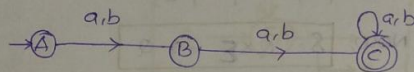
⇒ Language Acceptance: A finite Automata is said to accept a language if it accepts all the strings available in the language and similarly the strings that are not in the language must be rejected by the finite Automata.

L-2 CONSTRUCTION OF MINIMAL DFA

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

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

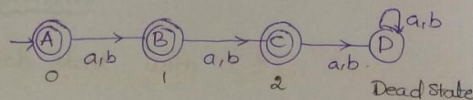
$L = \{aa, bb, aab, aaa, bbb, baa, \dots\}$



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

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

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



Now, from the Above problems the conclusions that can be drawn are

$|w| = 2$

$|w| \geq 2$

$|w| \leq 2$

If $|w| = n$ then

$|w| \geq n$ then no of

$|w| \leq n$ the no of

No. of states = $(n+2)$

States in DFA is $(n+1)$

States = $(n+2)$

$|w| \leq n = (n+2)$ states

$|w| = n = (n+2)$ states

$|w| \geq n = (n+1)$ states

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

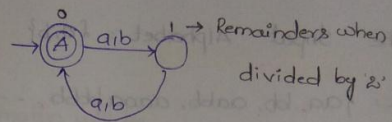
a final

(4)

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

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

$L = \{aa, ab, ba, bb\}$ and ϵ

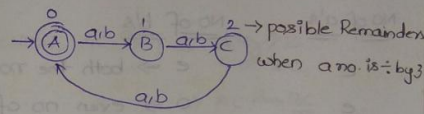


Remainders when divided by 2

(6) Construct the minimal DFA over $w = \{a,b\}^*$ where $|w| \bmod 3 = 0$ or $|w| \bmod 3 = 1$

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

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



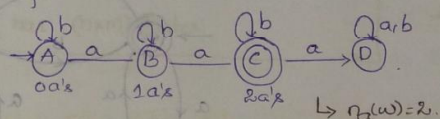
possible Remainders when a no. is div by 3

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

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

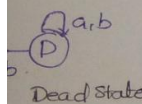
$L = \{aa, baa, aba, aab, aabbb, \dots\}$

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



$\Rightarrow n_a(w) = 2$

for $n_a(w) \geq 2$

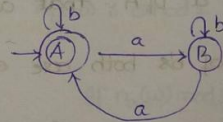


Dead state

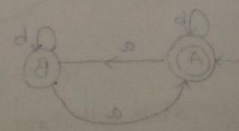
(8) CMDFA where $n_a(w) \bmod 2 = 0$ or $n_a(w) \bmod 2 = 1$

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

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



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



⑨ CMDFA over $\{a,b\}$ where $n_a(w) \equiv 0 \pmod 2$ and $n_b(w) \equiv 0 \pmod 2$

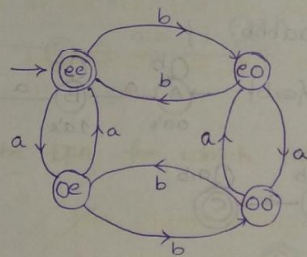
The Input Alphabet = $\{a,b\}$

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

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

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

$\therefore$ Therefore the DFA contains 4 states representing each category



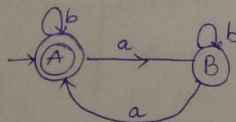
L-3 CROSS PRODUCT METHOD

⑩ construct a DFA that accepts strings over $\{a,b\}$ where no. of a's and no. of b's both are even?

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

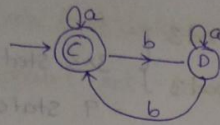
$n_b(w) \equiv 0 \pmod 2$

Now, construct the DFA for counting a's



③

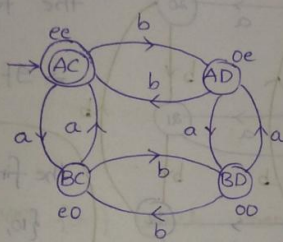
similarly the DFA for counting the b's will be



⑦

Now, the cross product of the two Automaton will be

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



Now, $AC \xrightarrow{a} ? \Rightarrow$

$\Rightarrow AC \xrightarrow{a} BC$

$AC \xrightarrow{b} ? \Rightarrow$

$\Rightarrow AC \xrightarrow{b} AD$

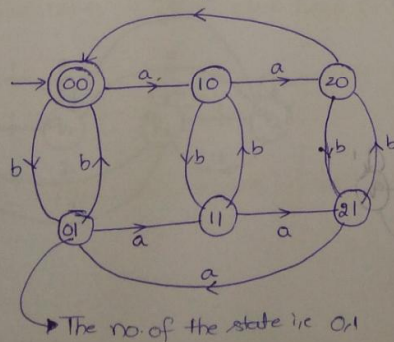
If the 1st DFA has 'n' states and 2nd DFA has 'm' states then the cross product of DFA₁ and DFA₂ has (n*m) states.

L-4

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

The Input Alphabet $\Sigma = \{a, b\}$, $w = (a, b)^*$

$n_a(w) \equiv 0 \pmod 3$ and $n_b(w) \equiv 0 \pmod 2$



If $n_a(w) \pmod 3 \geq n_b(w) \pmod 2$

then final states are

$\{10, 20, 21, 11\}$

If $n_a(w) \pmod 3 = n_b(w) \pmod 2$

$\{00, 11\}$ = final states.

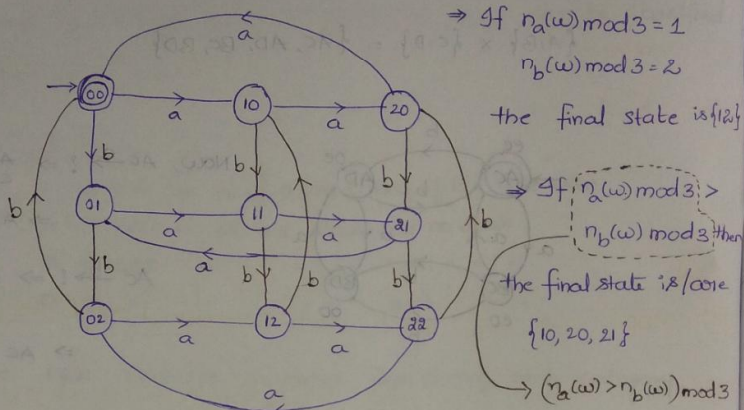
→ The no. of the state is 01

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

L-5

- (12) CMDFA over $\{a, b\}$ where $n_a(w) \equiv 0 \pmod 3$
 $n_b(w) \equiv 0 \pmod 3$ } 3×3 States
 $= 9$ States

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



L-6

- (13) CMDFA over $\{0, 1\}$ which when Interpreted as Binary number is ≥ 10

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

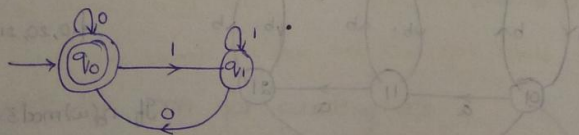
1- Unary

2- Binary - 0, 1

3- Ternary - 0, 1, 2

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

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



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

Base $(101)_2 = 1 \times 2 + 0 = 2$
 $2 \times 2 + 1 = 5$

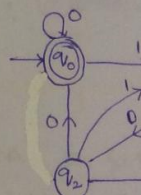
L-7

- (14) CMDFA over $\{a, b\}$
The Input Alphabet

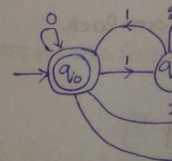


L-8

- (15) CMDFA over $\{a, b\}$
The Input Alphabet



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



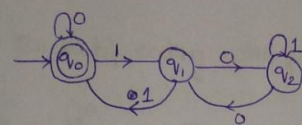
8

L-7

14) CMDFA over $\{0,1\}$ which interpreted as Binary number is $\div 1$ by 3.

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

The Transition table is



	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

Shortcut method

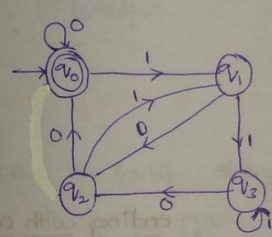
$13 = 1$
 $13 = 2$
 state is $\{1,2\}$

$w \bmod 3 >$
 $w \bmod 3$ then
 state is $\{0,1,2\}$

L-8

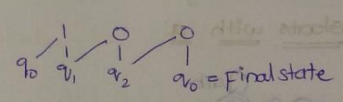
15) CMDFA over $\{0,1\}$ which when interpreted as Binary number is $\div 1$ by 4

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



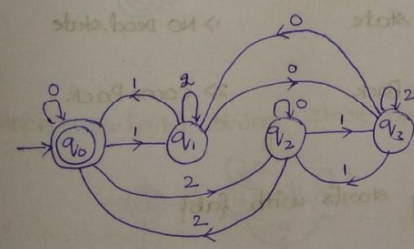
	0	1
q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_3	q_0
q_3	q_0	q_1

Shortcut



number is $\div 1$ by 2

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



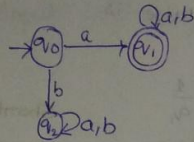
	0	1	2
q_0	q_0	q_1	q_2
q_1	q_2	q_3	q_0
q_2	q_3	q_0	q_1
q_3	q_0	q_1	q_2

Short cut

L-9

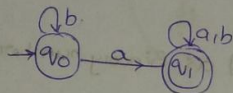
(16) CMDFA over $\Sigma = \{a, b\}$ and all the strings start with 'a'?

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



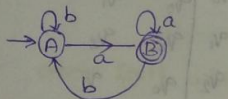
L-10

(17) CMDFA, $\Sigma = \{a, b\}$ where each string contains 'a'?



L-11

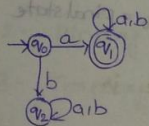
(18) CMDFA $\Sigma = \{a, b\}$ each string ending with 'a'?



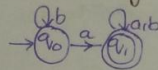
L-12

Now, comparing the above 3 DFAs we get

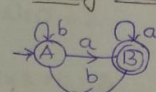
starts with 'a'



containing 'a'



ending with 'a'



1> Dead state is present

1> NO Dead state

1> NO Dead state

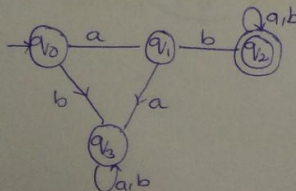
2> No come Back

2> No come Back

2> Come Back

L-13

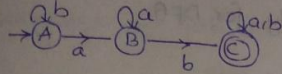
(19) CMDFA, $\Sigma = \{a, b\}$ and each string starts with {ab}



10

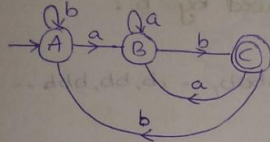
L-14

20) CMDFA $\Sigma = \{a, b\}$, Each string containing $\{ab\}$



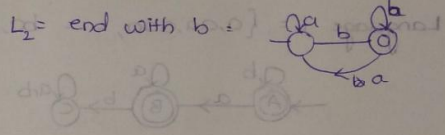
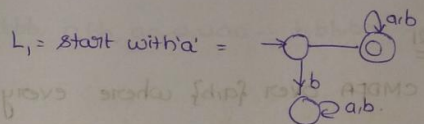
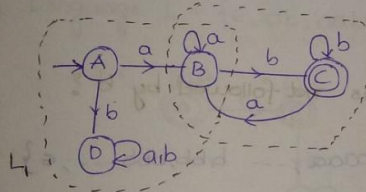
L-15

21) CMDFA $\Sigma = \{a, b\}$ ending with $\{ab\}$



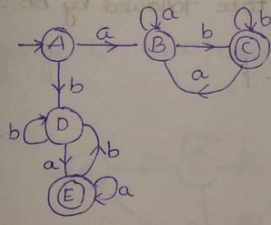
L-16

22) CMDFA $\Sigma = \{a, b\}$ start with 'a' and end with 'b'?



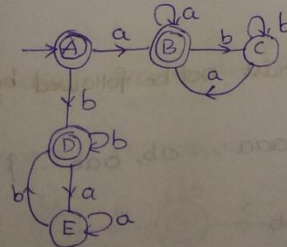
L-17

23) CMDFA $\Sigma = \{a, b\}$, string start and end with different symbol?



L-18

24) CMDFA $\Sigma = \{a, b\}$ string start and end with same symbol?



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

where L_1, L_2 are defined on same Σ

19 COMPLEMENTATION OF DFA

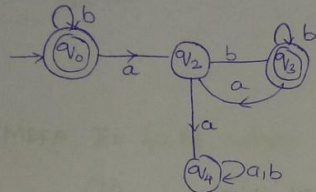
1) The complementation method is applicable only for DFA.

2) $(Q, \Sigma, \delta, q_0, f) \xrightarrow{\text{comp}} (Q, \Sigma, \delta, q_0, Q-F)$

L-20

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

Sol

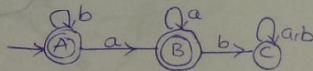


$L = \{\epsilon, ab, abab, \dots, b, bb, bbb, \dots\}$

L-21

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

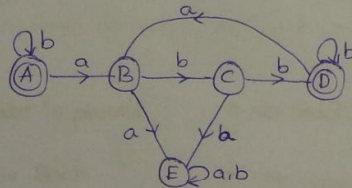
Language $L = \{a, aa, aaa, \dots, ba, baa, baaa, \dots, bbb, bbb, \dots, \epsilon\}$



L-22

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

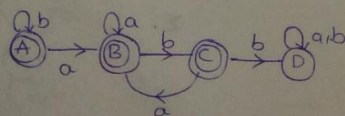
Language $L = \{\epsilon, b, bbb, \dots, abb, bbabb, \dots\}$



L-23

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

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



L-24

29) CMDFA which

The Language $L =$

L-25

30) CMDFA which a

The Language $L =$

L-26

31) CMDFA which

The Language $L =$

L-27

32) CMDFA which

The Language $L =$

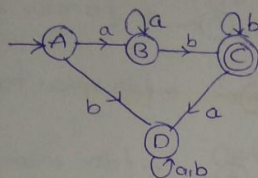
(12)

L-24

(27) CMDFA which accepts the Language $L = \{a^n b^m / n, m \geq 1\}$

(13)

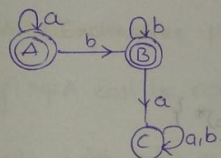
The Language $L = \{ab, aabb, aaabbb, aaaaabbbb, \dots\}$
 $abb, aabb, aaabb\}$



L-25

(28) CMDFA which accepts $L = \{a^n b^m / n, m \geq 0\}$

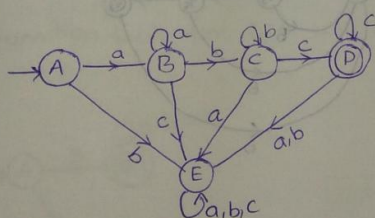
The Language $L = \{\epsilon, ab, aabb, aaabbb, abb, aa, a, aaa, \dots, b, bb, bbb, \dots\}$



L-26

(31) CMDFA which accepts $L = \{a^n b^m c^l / n, m, l \geq 1\}$

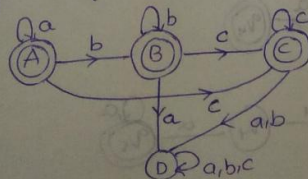
The Language $L = \{abc, aabc, abbc, abcc, \dots\}$



L-27

(32) CMDFA which accepts $L = \{a^n b^m c^l / n, m, l \geq 0\}$

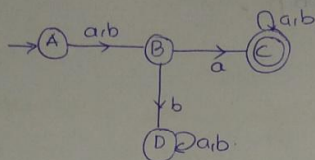
The Language $L = \{\epsilon, a, aa, aaa, \dots, b, bb, bbb, \dots, c, cc, ccc, \dots, abc, aabccc, \dots, aabb, aacc, \dots\}$



L-29

23) CMDFFA over $\{a, b\}$ such that the 2nd symbol from the LHS is 'a'?

The Language $L = \{aaaa, aa, ba, aab, ba---\}$



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

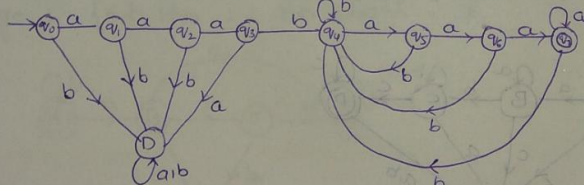
L-30

24) CMDFFA over $\{a, b\}$ where strings are of the form $a^3 b w a^3$ where w is any string over $\{a, b\}$

The Language $L = \{a^3 b w a^3 \mid w \in (a, b)^*\}$

$L = \{a^3 b \in a^3, a^3 b a a^3, a^3 b b a^3, a^3 b a a a^3, \dots\}$

The min. length string = $aaabadaa$

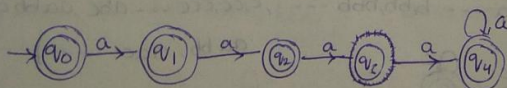


L-32

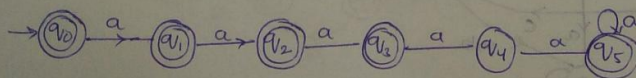
25) CMDFFA over $\{a\}$ such that 1) $\{a^n \mid n \geq 0, n \neq 3\}$

2) $\{a^n \mid n \geq 0, n \neq 4\}$

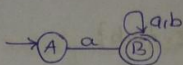
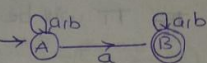
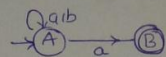
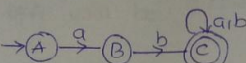
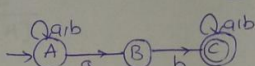
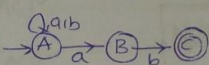
①



②

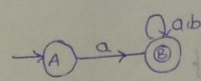


(14)

L-34 NFA $L_1 = \{\text{starts with 'a'}\} \Rightarrow$  $L_2 = \{\text{containing a}\} \Rightarrow$  $L_3 = \{\text{ends with a}\} \Rightarrow$  $L_4 = \{\text{start with ab}\} \Rightarrow$  $L_5 = \{\text{contains ab}\} \Rightarrow$  $L_6 = \{\text{End with ab}\} \Rightarrow$ L-35 NFA - DFA CONVERSION

where

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

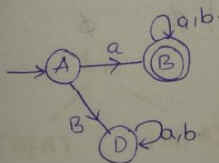
 $L_1 = \{\text{starts with a}\} \Sigma = \{a, b\}$ NFA for L_1 is. 

The state Transition table for the above NFA is

	a	b
A	B	$\emptyset$
B	B	B

The state Transition table for DFA will be

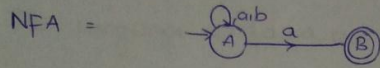
	a	b
$\rightarrow A$	B	D
*B	B	B
D	D	D



The transition function for NFA is

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

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

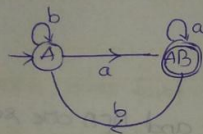


The TT will be

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

DFA Transition table will be

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

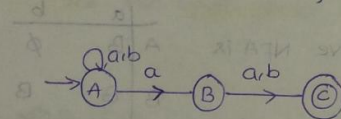


$$\begin{aligned}
 [AB] \rightarrow a &= (A \rightarrow a) \cup (B \rightarrow a) \\
 &= \{A, B\} \cup \{\emptyset\} \\
 &= \{A, B\} \\
 [AB] \rightarrow b &= (A \rightarrow b) \cup (B \rightarrow b) \\
 &= [A] \cup [\emptyset] = [A]
 \end{aligned}$$

L-37

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

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

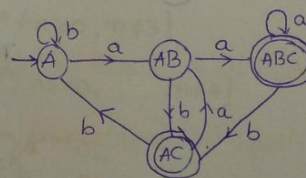


Transition table

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

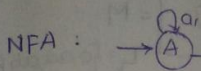
The state Transition table for DFA will be

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



L-38

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



The transition

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

L-39

NFA for string

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

(DFA)

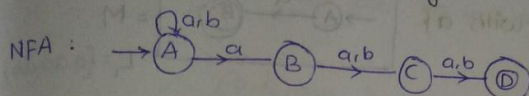
$n((DFA))$

No. of states
state

(16)

L-38

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



The transition table for DFA will be

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

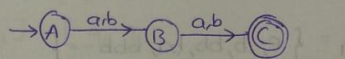
	a	b
→ A	{AB}	{A}
B	{C}	{C}
C	{D}	{D}
*D	{∅}	{∅}

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

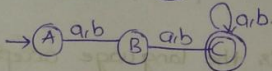
L-39

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

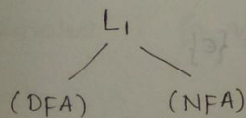
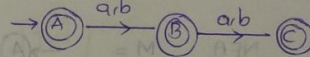
a) exactly 2 $L = \{aa, ab, ba, bb\}$



b) Atleast 2 $L = \{aaa, bbb, aa, bb, ba, bb, \dots\}$



c) Atmost 2 $L = \{\epsilon, a, b, ab, ba, bb, aa\}$



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

$$[\text{No. of states in DFA}] \geq [\text{No. of states in NFA}]$$

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

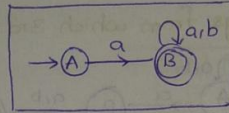
L-40

COMPLEMENTATION OF NFA

NFA

over $\{a, b\}$

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

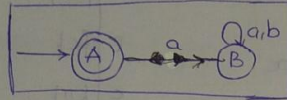


$= M$

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

Now,

$\bar{M} =$



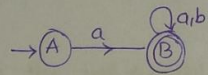
$L_2 = \{\epsilon\}$

Σ^*	
Start with 'a'	'a' and 'b' start with 'b'
L_1	L_2

Note:

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

consider, the NFA

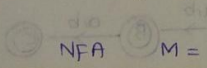


1> what is the complement of language accepted by this NFA?

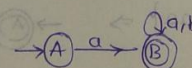
$L_1 = \{a, aa, ab, aaa, \dots\}$

$\bar{L}_1 = \{\epsilon, b, bb, ba, bbb, \dots\}$

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

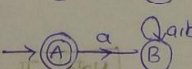


$M =$



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

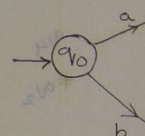
$\bar{M} =$



$L_2 = \{\epsilon\}$

L-41. MINIMISATION

Two states 'p'



'0' Equivalen

1 Equivalen

2 Equivalen

3 Equivalen

1> check wh

2> If they a

Ex

L-41 MINIMISATION OF DFA

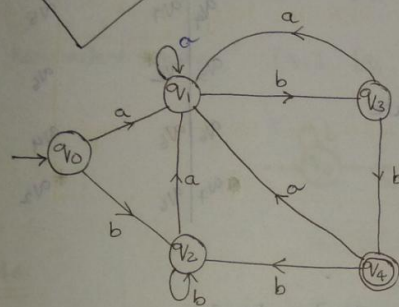
Two states 'p' and 'q' are Equivalent when

(p, q) are Equivalent if

$$\begin{aligned} \delta(p, w) \in F &\Rightarrow \delta(q, w) \in F \\ \delta(p, w) \notin F &\Rightarrow \delta(q, w) \notin F \end{aligned}$$

PARTITIONING METHOD

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



The Transition table is

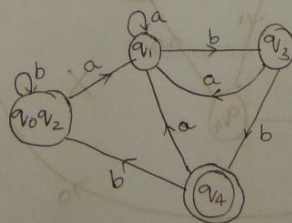
	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	$*q_4$
$*q_4$	q_1	q_2

0 Equivalent = $[q_0, q_1, q_2, q_3]$ $[q_4]$

1 Equivalent = $[q_0, q_1, q_2]$ $[q_3]$ $[q_4]$

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

3 Equivalent = $[q_0, q_2]$ $[q_1]$ $[q_3]$ $[q_4]$

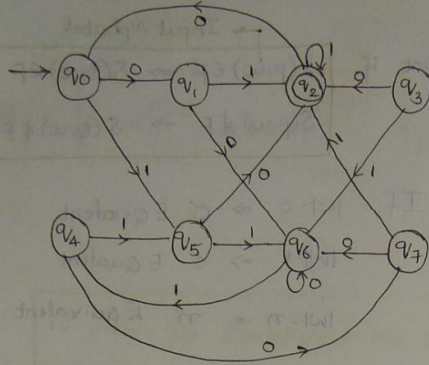


Note: The states that are not reachable from the initial state must be Removed.

- 1) Check whether they are in same group in previous Equivalence
- 2) If they are not Equal, now check if they are going to same states or not

L-42

Minimise the following DFA



$\Rightarrow q_3$ is not Reachable from Initial state

The Transition table will be

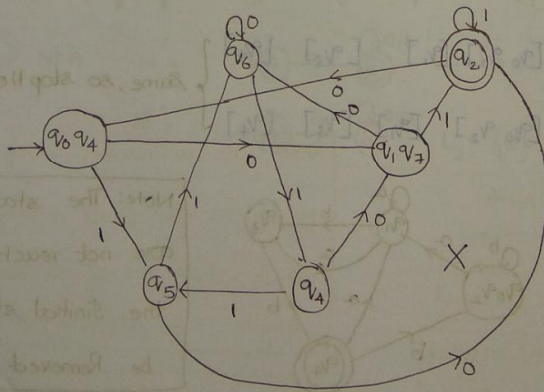
	0	1
$\rightarrow q_0$	q_1	q_5
q_1	q_6	$*q_2$
$*q_2$	q_0	$*q_2$
q_3	q_2	q_6 X
q_4	q_7	q_5
q_5	q_2 *	q_6
q_6	q_6	q_4
q_7	q_6	$*q_2$

0 Equivalent states $[q_0, q_1, q_4, q_5, q_6, q_7]$ $[q_2]$

1 Equivalent states $[q_0, q_4, q_6]$ $[q_1, q_7]$ $[q_5]$ ~~$[q_2]$~~ $[q_2]$

2 Equivalent states $[q_0, q_4]$ $[q_6]$ $[q_1, q_7]$ $[q_5]$ ~~$[q_2]$~~ $[q_2]$

3 Equivalent states $[q_0, q_4]$ $[q_6]$ $[q_1, q_7]$ $[q_5]$ ~~$[q_2]$~~ $[q_2]$ } same so stop here



L-43 (GAT)

Minimise the

0 Equival

1 Equival

L-44

Minimise the

The Transition

	a
q_0	q_1
q_1	q_0
q_2	$*q_3$
$*q_3$	$*q_3$
q_4	q_3
q_5	q_6
q_6	q_5
q_7	q_6

X
Not
Reachable

(20)

L-43 (GATE)

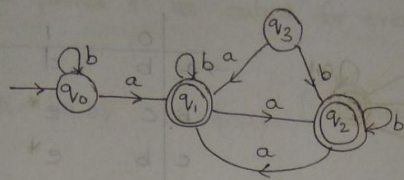
Minimise the DFA

table will be

1	
q_5	

* q_2 * q_2

q_6	X
-------	---

 q_5 q_6 q_4 * q_2 

The Transition table is

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

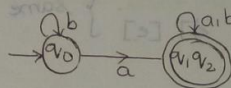
0 Equivalent states

 $[q_0]$ $[q_1, q_2]$

1 Equivalent states

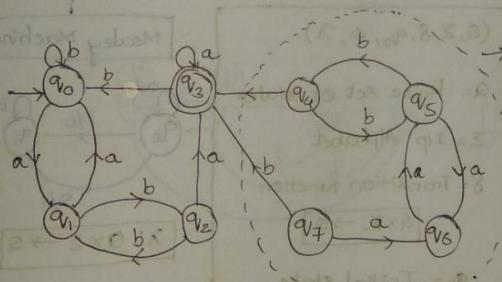
 $[q_0]$ $[q_1, q_2]$

same, stop here



L-44

Minimise the DFA

NOT REACHABLE FROM
INITIAL STATE

The Transition table is

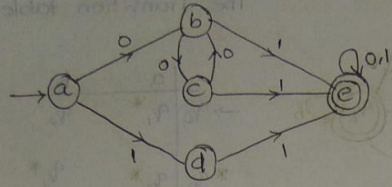
	a	b
q_0	q_1	q_0
q_1	q_0	q_2
q_2	* q_3	q_1
* q_3	* q_3	q_0

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

X	q_4	q_5	q_6
Not Reachable	q_4	q_5	q_6

L-45

Minimise the DFA



The Transition table is.

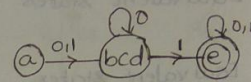
	0	1
a	b	d
b	c	e*
c	b	e*
d	c	e*
e	e	e*

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

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

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

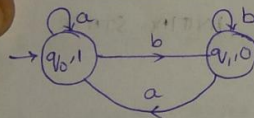
} same



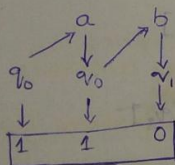
L-46 MOORE AND MEALEY MACHINES

Finite Automata with output

MOORE Machine



$\lambda: Q \rightarrow \Delta$



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

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

Q = Finite set of states

$\Sigma = I/P$ Alphabet

$\delta =$ Transition function

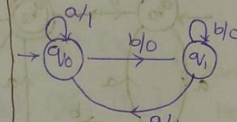
$Q \times \Sigma \rightarrow Q$

$q_0 =$ Initial state

$\Delta =$ o/p Alphabet

$\lambda =$ o/p function.

Mealey Machine



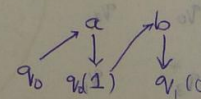
$\lambda: Q \times \Sigma \rightarrow Q$

$(q_0, a) \rightarrow 1$

$(q_0, b) \rightarrow 0$

$(q_1, a) \rightarrow 1$

$(q_1, b) \rightarrow 0$



10

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

L-47

Construct a moore as input and print substring.

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

L-48

counting the occ

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

L-49

Construct a moore

and produces

output if i/p

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

10 - A

11 - B.

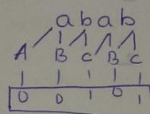
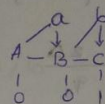
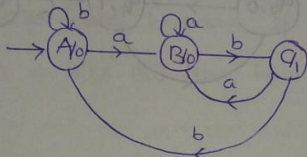
22

is.

L-47

construct a moore machine that takes set of all strings over $\{a,b\}$ as input and prints '1' as output for every occurrence of 'ab' as substring. (23)

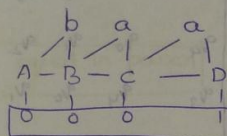
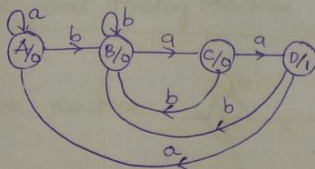
$\Sigma = \{a,b\}$ $\Delta = \{0,1\}$ $\therefore$ construct DFA ending with 'ab'.



L-48

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

$\Sigma = \{a,b\}$ $\Delta = \{0,1\}$ $\therefore$ construct DFA ending with 'baa'.

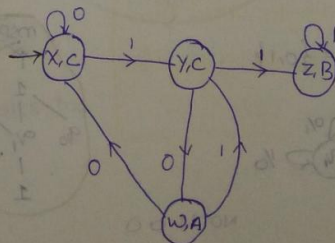


L-49

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

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

10 - A
11 - B



h Input then
n length string
e

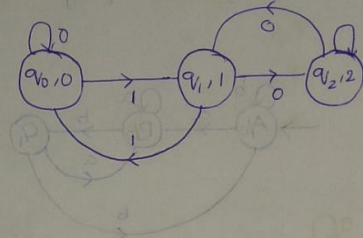
L-50

Moose Machine Residue modulo 'n'

construct a moose machine that takes binary no's as input and produces residue modulo '3' as output.

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

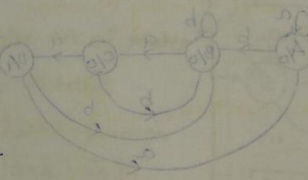
	0	1		Δ
q_0	q_0	q_1		0
q_1	q_2	q_0		1
q_2	q_1	q_2		2



construct a moose machine that takes base 4 no's as input and produces residue modulo '5' as output?

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

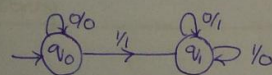
	0	1	2	3		Δ
q_0	q_0	q_1	q_2	q_3		0
q_1	q_4	q_0	q_1	q_2		1
q_2	q_3	q_4	q_0	q_1		2
q_3	q_2	q_3	q_4	q_0		3
q_4	q_1	q_2	q_3	q_4		4



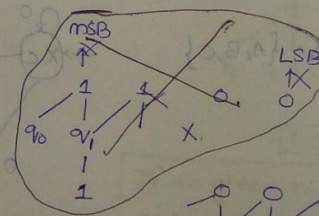
L-51 MEALY MACHINE

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

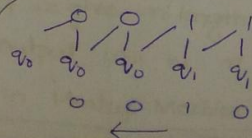
$$\Sigma = \{0, 1\} \quad \Delta = \{0, 1\}$$



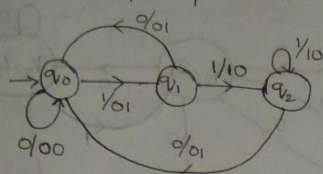
NO: 1100
1's = 0011
2's = 0100 (1's + 1)



write same digits until 1st one and then complement (1100) → 0100
← 1's comp
← 2's comp



What is the output produced by the following state machine

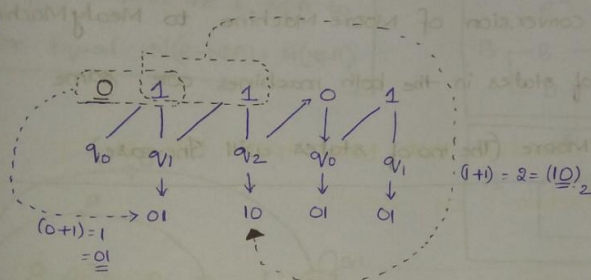
 $\times b \geq 10 - 00$

c) sum of present and previous bits

Q > NOT

option a, b are false because if we give 11/10 as input the output must contain 4 bits. But they have given 2 bits.

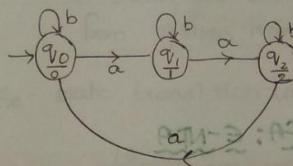
option c:



L-53 CONVERSION OF MOORE MACHINE TO MEALY MACHINE

Moore Machine and Mealy Machine are equal in power

Convert the moose machine



⇒ Now, in the moore machine

(q_0, b) is going to q_0 and the

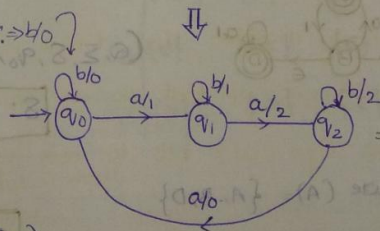
output associated with q_0 is 0. $\Rightarrow b/c$

$\Rightarrow$ Now, q_p on 'a' is going to

q_i and o/p associated at q_i

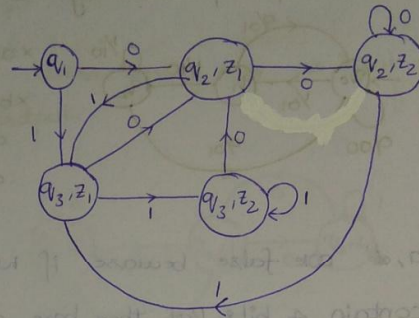
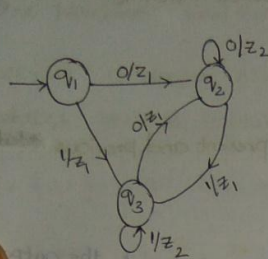
is 1 $\therefore a/1$ (in Mealy machine).

→ Similarly continue for other states too.



$\Rightarrow$ Mealy Machine

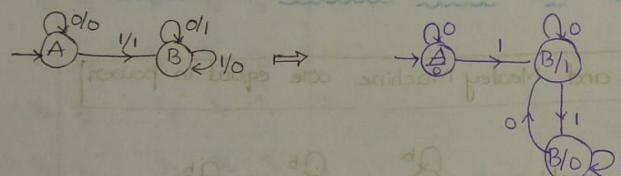
L-55 CONVERSION OF MEALY MACHINE TO MOORE MACHINE



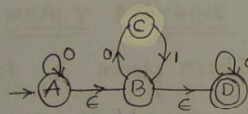
Note: 1. In the conversion of Moore machine to Mealy machine the No. of states in the both machines are same.
2. Mealy $\rightarrow$ Moore (The no. of states will increase)

L-56

②



L-57 EPSILON NFA: ϵ -NFA



(Q, Σ, S, q_0, F)

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

Now, ϵ -closure(A) = {A, B, D}

ϵ -closure(B) = {B, D}

ϵ -closure(C) = {C}

ϵ -closure(D) = {D}

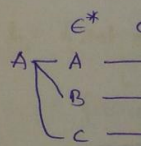
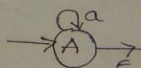
ϵ -closure(A) = The states that are Reachable from 'A' on seeing Epsilon(ϵ)

The state transition

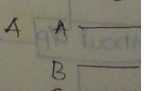
	0
$\rightarrow A$	{ABCD}
B.	{CD}
C	{ $\emptyset$ }
*D	{D}

The No. of NFA are

L-58



ϵ^* b



$A \rightarrow B = \{B\}$

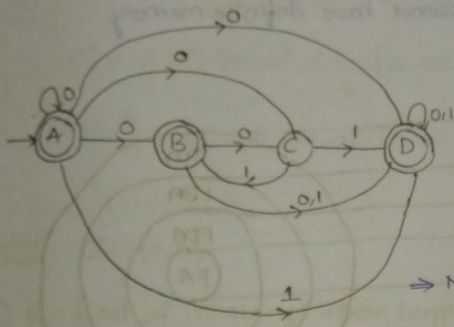
HINE

(20)

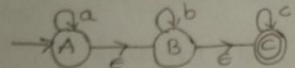
The state transition table

	0	1
$\rightarrow A$	$\{ABCD\}$	$\{D\}$
B	$\{CD\}$	$\{D\}$
C	$\{\phi\}$	$\{B, D\}$
*D	$\{D\}$	$\{D\}$

The No. of states in the E-NFA to NFA are Equal $N(E-NFA) = N(NFA)$

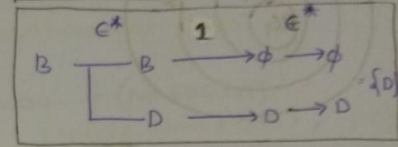
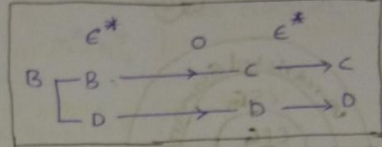
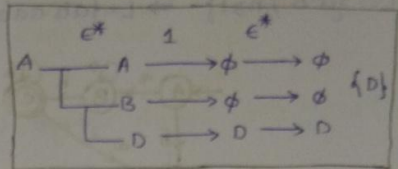
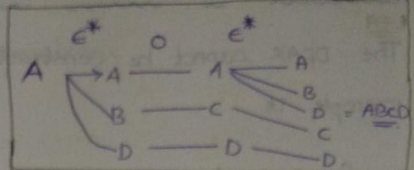
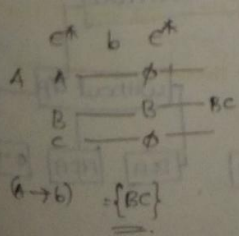
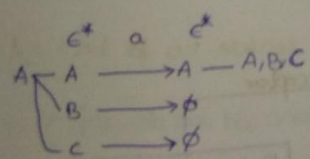


L-58



The state transition table is

	a	b	c
A	$\{ABC\}$	$\{BC\}$	$\{C\}$
B	$\{\phi\}$	$\{BC\}$	$\{C\}$
C	$\{\phi\}$	$\{\phi\}$	$\{C\}$



Conversion of E-NFA to NFA

⇒ No. of final states will increase from E-NFA to NFA conversion. (The states on which you can reach final states from epsilon transition are also final states)

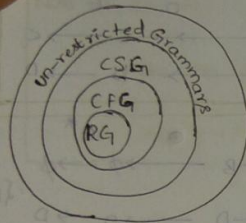
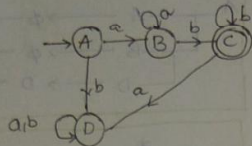
states that are on seeing

FAMILIES OF FORMAL LANGUAGES

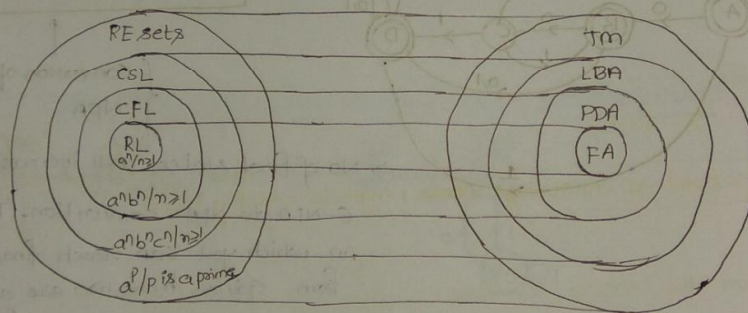
L-59

The DPFA cannot be constructed for all the languages and one such example is

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

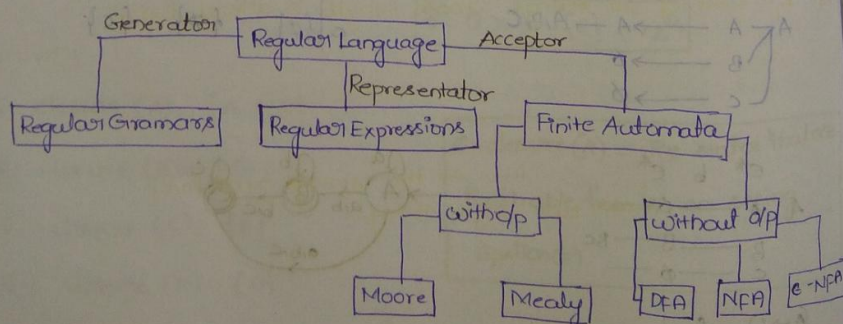


The Trap in this DFA is that the string "aab" will also be accepted, so we are unable to count the no. of 'a' arrived before 'b' so the memory required at every state will be infinite. But FA cannot have infinite memory.



REGULAR EXPRESSIONS AND CONVERSIONS

L-60



Regular Expression

The Regular Expression

- 1> Union (+)
- 2> Concatenation (.)
- 3> Kleen closure (*)

$$a) \emptyset = \{ \}$$

$$\epsilon = \{ \epsilon \}$$

$$a = \{ a \}$$

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

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

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

$$L-61 \quad \Sigma = \{ a, b \}$$

$$1) L_1 = \{ \text{set of } a^n b^n \}$$

$$= \{ aa, aaaa, \dots \}$$

$$= \{ aa, aaaa, \dots \}$$

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

$$= \{ a^{2n} \}$$

$$2) L_2 = \{ \text{set of } a^n b^n \}$$

$$= \{ aa, aaaa, \dots \}$$

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

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

Regular Expressions

The Regular Expression contains three operations

- 1) Union (+)
- 2) Concatenation (.)
- 3) Kleen closure (*)

$$a) \begin{cases} \phi = \{\} \\ \epsilon = \{\epsilon\} \\ a = \{a\} \end{cases} \rightarrow \text{primitive RE's.}$$

$$(b) \begin{cases} r_1 + r_2 \\ r_1 \cdot r_2 \\ r_1^*, r_2^* \end{cases} \rightarrow \text{are also RE}$$

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

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

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

$$\underline{L-61} \quad \Sigma = \{a, b\}$$

$$\textcircled{1} L_1 = \{\text{set of all strings whose Length is exactly two}\}$$

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

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

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

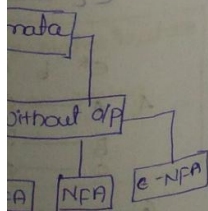
$$= \{(a+b)(a+b)\}$$

$$\textcircled{2} L_1 = \{\text{set of all strings whose Length is atleast two}\}$$

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

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

$\hookrightarrow$ All the strings over $\{a, b\}$.
 $\hookrightarrow$ All two Length strings



③ $L_1 = \{ \text{set of all strings whose length is at most 2} \}$

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

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

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

④ $L_1 = \{ \text{even length strings} \}$

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

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

→ Even length strings repeated any no. of times gives strings of even length.

$$\begin{aligned} \text{Analysis: } & ((a+b)(a+b))^* \\ & = ((a+b)^2)^* \\ & = (a+b)^{2*} = \text{even.} \end{aligned}$$

⑤ $L_1 = \{ \text{odd length strings} \}$

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

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

$$\begin{aligned} \text{Analysis: } & (a+b)^{2n+1} \\ & = (a+b)^{2n} (a+b) \\ & = ((a+b)(a+b))^* (a+b) \end{aligned}$$

⑥ $L_1 = \{ \text{Length is divisible by 3} \}$

$$L = \{ 0 \text{ length strings, 3 Ls, 6 Ls, 9 Ls, } \dots \}$$

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

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

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

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

⑧ $L = \{ \text{exactly 2 a's} \}$ {atleast 2}

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

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

{atmost 2}

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

⑨ $L = \{ \text{a's come even} \}$

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

⑩ $L = \{ \text{starts with } \epsilon \}$

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

⑪ $L = \{ \text{start and end} \}$

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

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

L-62

*** (12) $L = \{ \text{no 2 a's shd} \}$ ***

$$= \{ \epsilon, b, bb, bbb, \dots \}$$

$$\{ ba, bab, \dots \}$$

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

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

⑬ $L = \{ \text{NO 2 a's a} \}$

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

Starts

$$\{ aab, aabab, \dots \} - a$$

$$\{ ab, abab, ababab, \dots \} - a$$

$$\{ ba, baba, \dots \} - b$$

$$\{ b, bab, babab, \dots \} - b$$

① $L = \{a's \text{ come even}\} = \{aa, aab, bbb, bbb, \dots\}$
 $= (b^*ab^*ab^*)^* + (b^*)$ $\Rightarrow$ The Language/RE will not generate the strings $\{b, bb, bbb, \dots\}$ so we add b^* to generate the strings. (31)

② $L = \{\text{starts with } a\} \quad \{\text{Ends with } a\} \quad \{\text{contains } a\}$
 $= \{a(a+b)^*\} \quad \{(a+b)^*a\} \quad \{(a+b)^*a(a+b)^*\}$

③ $L = \{\text{start and end with diff symbols}\} \quad \{\text{start and end with same symbols}\}$
 $= \{a(a+b)^*b\} \quad L = \{\epsilon, a, b, aa, bbb, aaa, aba, \dots\}$
 $= \{a(a+b)^*a + b(a+b)^*b + a + b + \epsilon\}$

L-62

④ $L = \{\text{no 2 a's should come together}\}$
 $= \{\epsilon, b, bb, bbb, \dots, a, ab, aba, abab, ababa, \dots\}$
 $= (b+ab)^*(\epsilon+a) \quad \text{or} \quad (a+\epsilon)(b+ba)^*$
 $\downarrow$
 $\{ \text{set of all strings ending with } a \}$
 $\{ \text{set of all strings ending with } b \}$

⑤ $L = \{\text{NO 2 a's and no 2 b's come together}\}$
 $= \{\epsilon, a, b, ab, ba, aba, bab, \dots\}$

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

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

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

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

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

- ① $\phi + R = R + \phi = R$
- ② $\phi \cdot R = R \cdot \phi = \phi$
- ③ $\epsilon R = R \epsilon = R$
- ④ $\epsilon^* = \epsilon$
- ⑤ $\phi^* = \epsilon$
- ⑥ $\epsilon + RR^* = RR^* + \epsilon = R^*$
- ⑦ $(a+b)^* = (a^* + b^*)^* = (a^*b^*)^*$
 $= (a^* + b)^*$
 $= (a + b^*)^* = a^*(ba^*)^* = b^*(ab^*)^*$

L-63

Regular Expression to finite Automata

1) ϕ :

2) a :

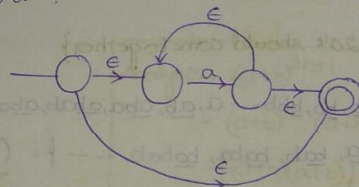
3) $a+b$:

4) ab :

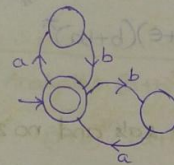
① ab^* :

② $(ab)^*$:

⑤ a^* :



③ $(ab+ba)^*$:



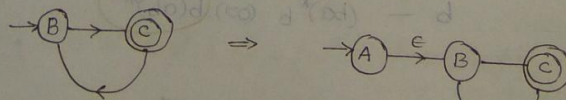
STATE CREATION

METHOD

Finite Automata to Regular Expression conversion

Rule 1

The initial stage should not have any incoming edge

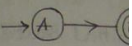


STATE ELIMINATION

METHOD

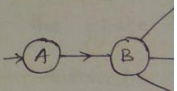
Rule 2

The final state

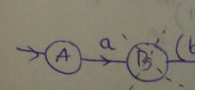
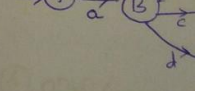
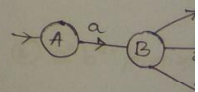
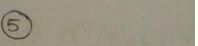
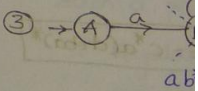
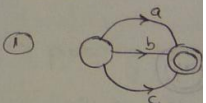


Rule 3

There must be except the

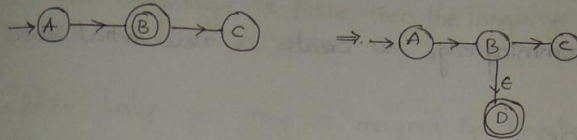


Examples



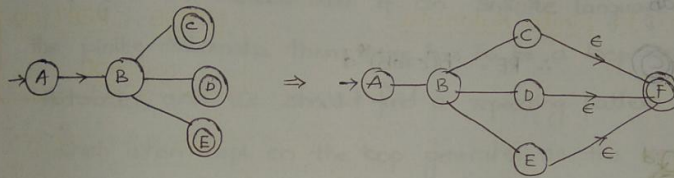
$$= a(b+c)$$

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

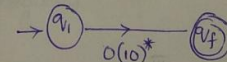
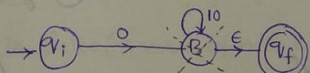
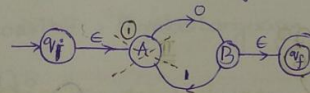
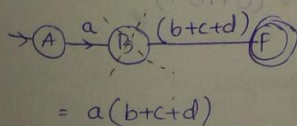
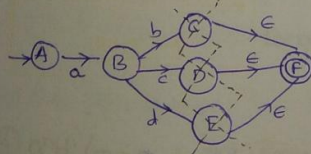
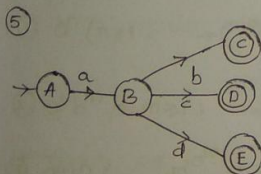
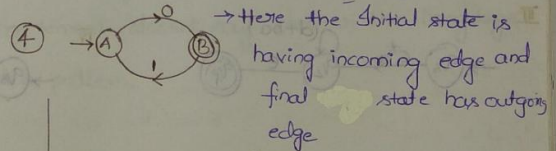
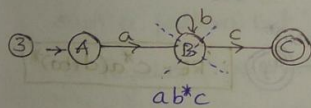
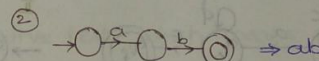
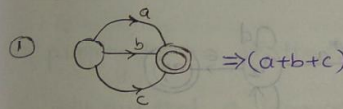


Rule-3

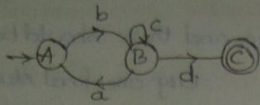
There must be only one final state and now eliminate the other states except the initial and final states in any order you wish.



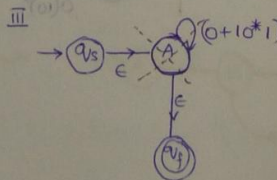
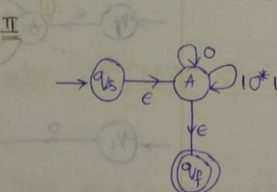
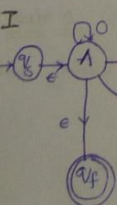
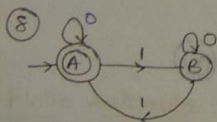
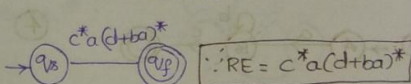
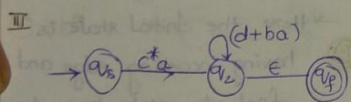
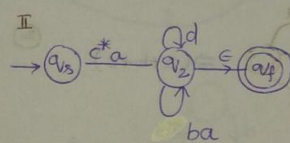
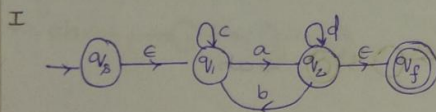
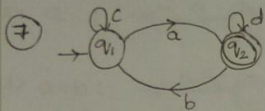
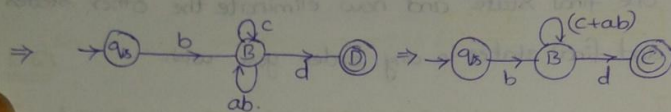
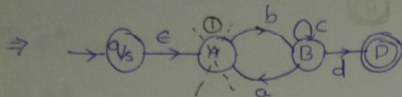
Examples



⑥



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



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

TESTING

L-64

If the

$\Rightarrow$ Infinite L

that is c

not is a

$\Rightarrow$ Pumping Lemma

the finite

Automata

which w

the language

be cons

$\Rightarrow$ pumping Lemma

will say t

such a p

is Regular

① $a^n/n \geq 1$

② $a^n b^m/n, m$

③ $a^n b^n/n \leq 1$

④ $a^n b^n/n \geq 1$

⑤ $w w^R/|w| = 2$

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

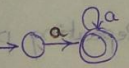
L-64

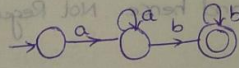
If the language is finite then the language is Regular

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

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

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

① $a^n/n \geq 1$:  ⇒ DFA possible = Regular Language

② $a^n b^m/n, m \geq 1$:  = Regular Language

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

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

⑤ $w \in R/|w| = 2, w \in \{a, b\}$: $\left\{ \begin{array}{l} aaaa \\ abba \\ baab \\ bbbb \end{array} \right\}$ - finite language $\Rightarrow$ Regular language

⑥ $w w^R / w \in \{a, b\}^* = \{ \text{set of all palindromes} \} = \text{The Language is not Regular}$

because the string 'w' can be of any length and that must be compared against w^R and the word/string

'w' can be infinite length, hence FA has no capacity to store infinite length strings $\Rightarrow$ Not- Regular Language

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

$\hookrightarrow a a^* b b^* c c^*$

⑧ $a^i b^j / i, j \geq 1 = \text{Regular Language} = a a^* b b (b b)^*$

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

⑨ $a^n / n \text{ is even}$

$= \{a^2, a^4, a^6, a^8, \dots\}$ difference can be kept in loop of DFA

$a^n / n \text{ is odd}$

$= \{a^1, a^3, a^5, a^7, \dots\} = \text{Regular Language}$

⑩ $a^n / n \text{ is prime}$

$= \{a^1, a^2, a^3, a^5, a^7, a^{11}, \dots\} = \{ \text{Not in AP} \Rightarrow \text{No Repeating pattern} \}$

$a^{n^2} / n \geq 1$

$= \{a^1, a^4, a^9, a^{16}, a^{25}, \dots\} = \{ \text{Not in AP} \Rightarrow \text{Not RL} \}$

$a^{n^2} a^{n^2} / n \geq 1$

$= \{a^1, a^4, a^9, a^{16}, \dots\} = \{ \text{Not in AP} \Rightarrow \text{Not RL} \}$

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

⑪ $a^i b^j / i, j \geq 1$

$= \text{Even the a, b are generated independently, due}$

to j^2 there is not Repeating pattern in the Language and hence Not Regular

⑫ $a^i b^{2n} / i, n \geq 1$

$= a^i \text{ can be independently generated and } 2n / (2^{\text{th}} \text{ power } n) \text{ has no Repeating pattern.}$

⑬ $a^i b^j / i \geq 1$

$= b^j \text{ cannot be generated and No Repeating pattern.}$

⑭ $\{w / n_a(w) = n_b(w)\}$

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

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

$= \text{Not Regular Language (counting a's need Infinite Memory)}$

⑮ $n_a(w) \bmod 3 \leq$

⑯ $a^n / n \geq 10 \Rightarrow$

⑰ $w w w^R / w \in \{a\}$

⑱ $a^n b^n c^n / n \geq 1$

⑲ $a^n b^{n+m} c^m / n,$

Imp V.V. Imp

⑳ $w x w^R / w, x \in$

㉑ $w x w^R / w \in \{0, 1\}$
 $|x| = 5$

㉒ $x w w^R y / x, y$

㉓ $x w w^R / x, w \in$

㉔ $w w^R y / w, y \in$

㉕ Set of all s

15) $n_a(w) \bmod 3 \leq n_b(w) \bmod 3 \Rightarrow$ Regular Language (DFA can be constructed for mod 3 counters) (37)

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

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

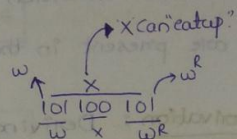
18) $a^n b^n c^n / n \geq 1 =$ Not Regular Language

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

Imp V.V. Imp

20) $wxw^R / w, x \in (0,1)^+ : \text{Regular Language}$

$w = 101$
 $x = 100$
 $w^R = 101$



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

21) $wxw^R / w \in (0,1)^+ : \text{Not Regular Language}$ ($\because$ 'x' cannot go beyond '5' length)
 $1 \times 1 = 5$

22) $xww^Ry / x, y, w \in (0,1)^+ :$
 $x = 10$
 $y = 11$
 $w = 100$
 $w^R = 001$

$$\frac{x}{x} \frac{w}{w} \frac{w^R}{w^R} \frac{y}{y}$$

$\therefore$ Regular Language

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

23) $xww^R / x, w \in (0,1)^+ : \text{Not Regular Language}$

24) $ww^Ry / w, y \in (0,1)^+ : \text{Not Regular Language}$

25) Set of all strings having equal number of '01' and '10' is Regular.
 $0(0+1)^*0 + 1(0+1)^*1 + 1+0$

GRAMMARS

L-65

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

V = vertices
 T = Terminals
 P = productions
 S = start symbol

$$S \rightarrow aSB$$

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

$$P = \left\{ \begin{array}{l} S \rightarrow aSB \\ S \rightarrow aB \\ B \rightarrow b \end{array} \right\}$$

$$S = \{S\}$$

$$S \rightarrow aB$$

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

$$B \rightarrow b$$

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

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

$$S \rightarrow aSB$$

$$S \rightarrow aB$$

$$B \rightarrow b$$

Derivation of aabb

$$S \rightarrow aSB$$

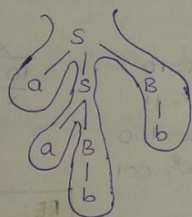
$$\Rightarrow a a B B$$

$$\Rightarrow a a b B$$

$$\Rightarrow a a b b$$

Sentential forms (or) Sequential forms.

Left most derivation



CONSTRUCTION OF GRAMMARS

① $L = \{aa, ab, ba, bb\}$

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

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

$$A \quad A$$

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

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

$$A \rightarrow aA/\epsilon$$

(or)

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

③ $L = (a+b)^*$ $L = \{\epsilon, a, b, aa, bb, \dots\}$

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

④ $L = \{\text{set of all strings of length } n\}$
 $= (a+b)^n$

⑤ $L = \{\text{Length of string}\}$
 $= (a+b)^n$

⑥ $L = \{\text{start symbol}\}$
 $= a(a+b)^n$

⑦ $L = \{\text{start and end symbol}\}$
 $= a(a+b)^n$

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

⑧ $L = \{a^n b^n/n \geq 1\}$

⑨ $L = \{w w^R\} \cup \{w a w^R\}$
 $w \in (a+b)^*$

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

⑪ $L = \{a^n b^m/n, m \geq 1\}$

⑫ $L = \{a^n c^m b^n\}$

38

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

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

$$\begin{aligned} S &\rightarrow AAB \\ A &\rightarrow a/b \\ B &\rightarrow aB/bB/\epsilon \end{aligned}$$

39

⑤ $L = \{\text{Length at most 2}\}$

$$= \underbrace{(a+b+\epsilon)}_A \underbrace{(a+b+\epsilon)}_A$$

$$\begin{aligned} S &\rightarrow AA \\ A &\rightarrow a/b/\epsilon \end{aligned}$$

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

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

$$\begin{aligned} S &\rightarrow aAb \\ A &\rightarrow aA/bA/\epsilon \end{aligned}$$

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

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

$$\begin{aligned} S &\rightarrow aAb/bAa \\ A &\rightarrow aA/bA/\epsilon \end{aligned}$$

⑧ $L = \{\text{same symbols}\}$

$$\begin{aligned} S &\rightarrow aAa/bAb/a/b/\epsilon \\ A &\rightarrow aA/bA/\epsilon \end{aligned}$$

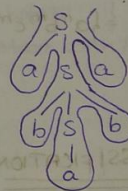
⑨ $L = \{a^n b^m / n \geq 1\}$

$$S \rightarrow aSb/ab$$

⑩ $L = \{wuw^k\} \cup \{waw^k\} \cup \{wbw^k\}$

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

$$S \rightarrow aSa/bSb/a/b/\epsilon$$

⑪ $L = \{\text{set of all strings of even length}\}$

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

$$\begin{aligned} S &\rightarrow BS/\epsilon \\ B &\rightarrow AA \\ A &\rightarrow a/b \end{aligned}$$

⑫ $L = \{a^n b^m / n, m \geq 1\}$

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow aA/a \\ B &\rightarrow aB/b \end{aligned}$$

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

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow aAb/ab \\ B &\rightarrow cB/c \end{aligned}$$

⑬ $L = \{a^n c^m b^n / n, m \geq 1\}$

$$\begin{aligned} S &\rightarrow aSb/aAb \\ A &\rightarrow cA/c \end{aligned}$$

PUSH DOWN AUTOMATA

L-68

Push Down Automata = Finite Automata + stack

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

Q = finite set of states

Z_0 = Bottom of the stack

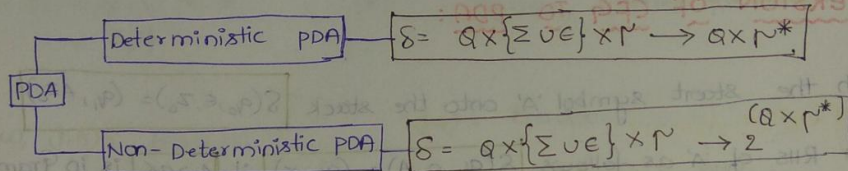
Σ = Input symbol

F = set of final states

δ = Transition function

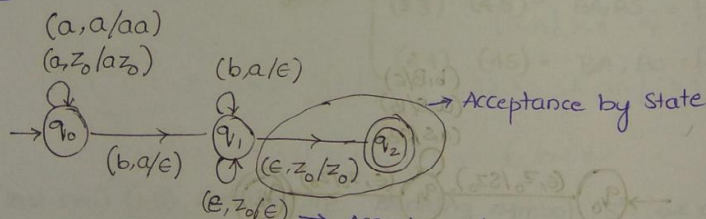
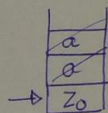
Γ = Stack Alphabet.

q_0 = Initial state



① $L = \{a^n b^n / n \geq 1\}$

Let the string be $aabb$
 $\uparrow \uparrow \uparrow \uparrow$



$\delta(q_0, a, Z_0) = (q_0, aZ_0)$ → Acceptance by Empty stack

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

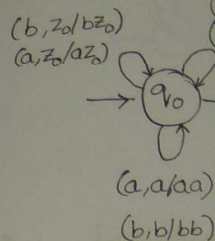
$\delta(q_0, b, a) = (q_1, \epsilon)$

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

$\delta(q_1, \epsilon, Z_0) = (q_1, Z_0) \text{ (or) } (q_1, \epsilon)$

Acceptance by state → Acceptance by Empty stack

② $w \in (arb)^n / n \geq 1$

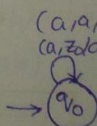


③ $L = \{a^n b^n c^m\}$

L-69

④ $L = \{a^n b^m c^n\}$

⑤ $L = \{a^m b^n\}$

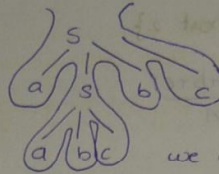


13) $L = \{a^n b^n c^m d^n / n, m \geq 1\}$

$S \rightarrow AB$
 $A \rightarrow aAb/ab$
 $B \rightarrow cbd/cd$

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

$S \rightarrow aSbc/abc$



$aabcbbc$

$= a^n (bc)^n$

For this language

we cannot give such grammar & we give 'CSG' for this language

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

$S \rightarrow aSbb/abb$

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

$S \rightarrow aSd/aAd$
 $A \rightarrow bAc/bc$

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

18) $L = \{a^n a^m b^m c^n\}$

$S \rightarrow aSc/aAc$
 $A \rightarrow aAb/ab$

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

$S \rightarrow AB$
 $A \rightarrow aAb/ab$
 $B \rightarrow bBc/bc$

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

$S \rightarrow aSc/aAc$
 $A \rightarrow bAc/bc$

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

L-66

CLASSIFICATION OF GRAMMARS

According to Chomsky the Grammars are classified into 4 types.

1) Type 3 / Regular Grammars

2) Type 2

3) Type 1

4) Type 0

Type-3:

If the Grammar has all the productions of the form

Right Linear
Grammar
(RLG)

$A \rightarrow \alpha B / \beta$

$(A, B) \in V$
 $(\alpha / \beta) \in T^*$

$A \rightarrow B\alpha / \beta$

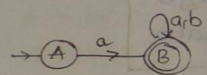
Left Linear
Grammar
(LLG)

Ex:

RLG
 $A \rightarrow aB/c$
 $B \rightarrow aB/$

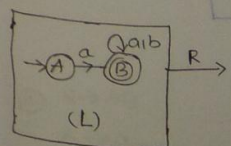
The Languages are always

CONVERSION OF



Finite Automata

Convert the



Ex:

RLG

$A \rightarrow aB/a$
 $B \rightarrow aB/bB/a/b$

LLG

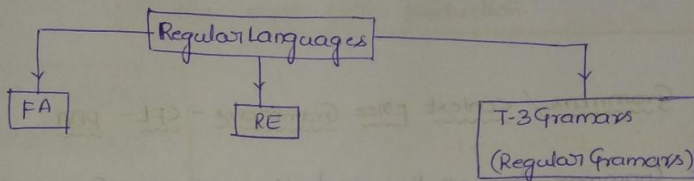
$A \rightarrow Ba/a$
 $B \rightarrow Ba/Bb/a/b$

Not T-3.

$A \rightarrow Ba/a$
 $B \rightarrow aB/a$

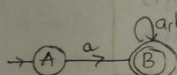
The Languages that are generated by Type-3 Grammar are always Regular Languages.

{ T-3 Grammar should be either LLG or RLG but not both. }



CONVERSION OF FINITE AUTOMATA TO REGULAR GRAMMAR

⇒ The Initial state in the Automata is same as the start state in the Grammar



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

RLG

$L = \{a, aa, ababab \dots\}$
start with 'A'

↓ R

$A \rightarrow Ba$
 $B \rightarrow Ba/Bb/\epsilon$

LLG

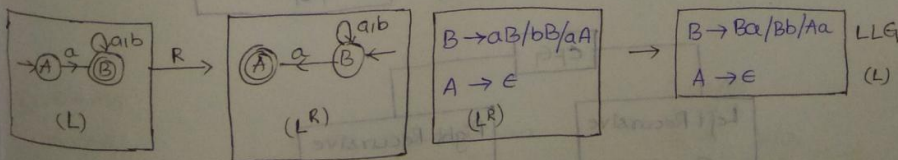
$L = \{a, ba, baba \dots\}$

Ending with 'a'

Finite Automata to LLG:

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

⇒ Convert the above FA to LLG.



Left Linear Grammar (LLG)

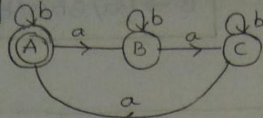
CONVERSION OF REGULAR GRAMMARS TO FINITE AUTOMATA

$A \rightarrow ab/bA/b$

$B \rightarrow aC/bB$

$C \rightarrow aA/bC/a$

$RLG \rightarrow FA$



LLG $\rightarrow$ FA CONVERSION :

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

Type-2 Grammars / Context Free Grammars - CFL - PDA

If the Grammar has all the productions of the form

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

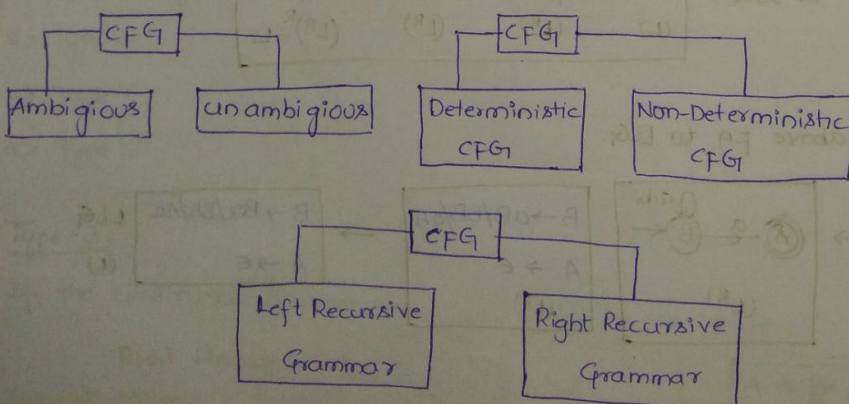
Ex: $A \rightarrow aAb/b$

$A \Rightarrow aAb$
 $\Rightarrow aabb$

$\Rightarrow$ The Grammar in the example is called Context free Grammar because we are replacing 'A' without any context (condition) that it must be preceded by 'a' likewise.

L-67

Type 2: Context Free Grammars



ELIMINATION

① $S \rightarrow aSb$

$A \rightarrow \epsilon$

② $S \rightarrow AB$

$A \rightarrow aA$

$B \rightarrow bB$

③ $S \rightarrow Aba$

$A \rightarrow BC$

$B \rightarrow B/\epsilon$

$C \rightarrow D/\epsilon$

$D \rightarrow d$

ELIMINATION

① $S \rightarrow Aa/b$

$B \rightarrow A/bb$

$A \rightarrow a/bc$

② $S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow C/b$

$C \rightarrow D$

$D \rightarrow \epsilon$

42

ELIMINATION OF ϵ -PRODUCTIONS

$$\textcircled{1} \begin{cases} S \rightarrow aSb / aAb \\ A \rightarrow \epsilon \end{cases} \Rightarrow$$

$$S \rightarrow asb / ab$$

$$S \rightarrow asb / \cancel{aAb} / ab$$

No production with ϵ

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

$$\textcircled{2} \begin{cases} S \rightarrow AB \\ A \rightarrow aAA / \epsilon \\ B \rightarrow bBB / \epsilon \end{cases}$$

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

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

$$\begin{cases} S \rightarrow AB / B / A / \epsilon \\ A \rightarrow aAA / aA / a \\ B \rightarrow bBB / bB / b \end{cases}$$

$$\textcircled{3} \begin{cases} S \rightarrow AbaC \\ A \rightarrow BC \\ B \rightarrow B / \epsilon \\ C \rightarrow D / \epsilon \\ D \rightarrow d \end{cases}$$

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

$$S \rightarrow AbaC / Aba / ba / baC$$

$$A \rightarrow Bc / B / C$$

$$C \rightarrow D$$

$$D \rightarrow d$$

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

ELIMINATION OF UNIT PRODUCTIONS

$$\textcircled{1} \begin{cases} S \rightarrow Aa / B \\ B \rightarrow A / bb \\ A \rightarrow a / bc / B \end{cases}$$

Step-1:

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

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

$$\Rightarrow S \rightarrow \cancel{A} \rightarrow A \rightarrow \begin{matrix} a \\ bc \end{matrix}$$

$$\Rightarrow B \rightarrow \cancel{A} \rightarrow \begin{matrix} a \\ bc \end{matrix}$$

$$\Rightarrow A \rightarrow \cancel{B} \rightarrow \begin{matrix} a \\ bb \end{matrix}$$

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

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

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

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

$$S \rightarrow AB$$

$$A \rightarrow a$$

$$B \rightarrow b / a$$

$$C \rightarrow a$$

$$D \rightarrow a$$

$$E \rightarrow a$$

Useless symbols
Not Reachable
from start state

ELIMINATION OF USELESS SYMBOLS

① $S \rightarrow A\cancel{B}/a$

$T = \{a, b\}$

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

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

$B \rightarrow AB/C$
 $C \rightarrow aC/B$

X

Now, the useful symbols (which will generate some terminals) are $\{a, b, S, A\}$
 $\Rightarrow B, C$ are not useful symbols ($\because$ I could not generate any string from B, C).

$S \rightarrow a$
 $A \rightarrow b$

But here 'A' is not Reachable, therefore the minimised Grammar will be $S \rightarrow a$

② $S \rightarrow AB/\cancel{AC}$

$T = \{a, b\}$

$A \rightarrow aAb/bAa/a$

$B \rightarrow bA/aAB/AB$

$C \rightarrow abCA/aDb$
 $D \rightarrow bD/aC$

X

$S \rightarrow AB$

$A \rightarrow aAb/bAa/a$

$B \rightarrow bA/aAB/AB$

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

③ $S \rightarrow \cancel{ABC}/BaB$

$T = \{a, b\}$

$A \rightarrow aA/\cancel{B}aC/aaa$

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

$B \rightarrow bBb/a$

$C \rightarrow CA/\cancel{AC}$

X

$S \rightarrow BaB$

$A \rightarrow aA/aaa$

$B \rightarrow bBb/a$

Not Reachable

$\Rightarrow S \rightarrow BaB$

$B \rightarrow bBb/a$

CNF - INTR

CNF:

If all the A, B are V

Advantages

- 1) Length
- 2) Derivation
- 3) The no.
- 4) It is ec

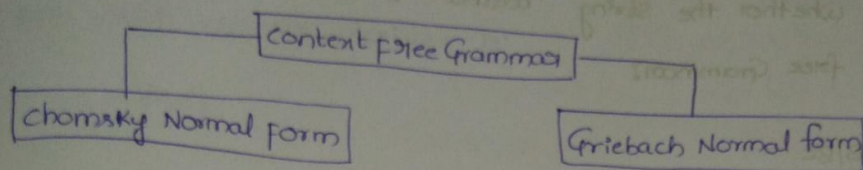
① Convert

$S \rightarrow bA/a$

$A \rightarrow bAA$

$B \rightarrow aAB$

CNF - INTRODUCTION [CHOMSKY - NORMAL FORM] 45



CNF:

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

Advantages:

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

① Convert the following into CNF?

$$S \rightarrow bA/aB$$

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

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

Let $\begin{cases} N_a \rightarrow a \\ N_b \rightarrow b \end{cases}$ then

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

$$A \rightarrow N_b \textcircled{AA} / N_a S / a$$

$$B \rightarrow N_a \textcircled{BB} / N_b S / b$$

$$N_a \rightarrow a$$

$$N_b \rightarrow b$$

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

CYK ALGORITHM (MEMBERSHIP ALGORITHM)

Check whether the string "baaba" is a valid member of the following context free Grammar?

$S \rightarrow AB/BC$

$A \rightarrow BA/a$

$B \rightarrow cc/b$

$C \rightarrow AB/a$

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

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

Given Grammar

$S \rightarrow AB/BC$

$A \rightarrow BA/a$

$B \rightarrow cc/b$

$C \rightarrow AB/a$

Given string

$b/a/a/b/a$
1/2/3/4/5

	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				

Cell no (1,1):

Now the (1,1) represent the variable 'b' in the required string
↓
[1 to 1]

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

⇒ B ✓

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

The Value
Now see

Cell no (3,3):
The Alphabet

Cell no (4,4)

(4,4) = b

Cell no (1,2)

(1,2) means

(12)

Cell no (2,3)

(2,3) = (2

Cell no (3,4)

(33) x (44)

following

cell no (2,2):

The value of (2,2) in the given string is 'a' (2 to 2) end with '2'.

Now see which variable derives 'a'

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

cell no (3,3):

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

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

cell no (4,4):

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

cell no (5,5):

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

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

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

$$\begin{aligned} (12) &= (11) \times (22) \quad [\text{cross product}] \\ &= \{B\} \times \{A, C\} = \{BA, BC\} = \{A, S\} \end{aligned}$$

Generated by

cell no (2,3):

$$\begin{aligned} (2,3) &= (22) \times (33) \\ &= \{A, C\} \times \{A, C\} \end{aligned}$$

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

$\hookrightarrow$ Not derived by any of the symbols.

cell no (3,4):

$$(4,5) = (44) \times (55)$$

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

$$= \{AB, CB\}$$

$$= \{BA, BC\} = \{A, S\}$$

generate

$$(1,3) = \begin{bmatrix} b & a & a \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} b & a \\ 4 & 5 \end{bmatrix} = (baa)$$

$$(1) \mid (2 \ 3) \quad (12) \mid (3)$$

$$(1,3) = (11)(23) \quad \text{or} \quad (12)(33)$$

$$\begin{aligned} &= \{B\} \times \{B\} \quad \{A,S\} \times \{A,C\} \\ &= \{BB\} \quad \{AA, AC, SA, SC\} \\ &= \{\emptyset\} \quad \{x \times x \times x\} = \{\emptyset\} \end{aligned}$$

$$(2,4) : (22) \times (3,4) \quad \text{or} \quad (23) \times (44)$$

$$= \{A,S\} \times \{S,C\} \quad \text{or} \quad \{B\} \times \{B\}$$

$$\begin{aligned} &= \{AS, AC, SS, SC\} \\ &\quad \times \times \times \times \\ &= \{\emptyset\} \end{aligned} \quad = \{BB\}$$

similarly continue for remaining cells

$$\text{Now, } (1,5) = baa ba$$

$$(14) = baab$$

$$(1,5) = \begin{cases} (11) (2 \ 5) = \{A,S\} \\ (12) (3 \ 5) = \{S,C\} \\ (13) (4 \ 5) = \{\emptyset\} \\ (14) (55) = \{\emptyset\} \end{cases} \quad (14) = \begin{cases} (11) (2 \ 4) = BB \times \\ (12) (3 \ 4) = AS, AC, SS, SC \times \\ (13) (44) = \{\emptyset\} \end{cases}$$

$$(2,5) = \begin{matrix} 2 & 3 & 4 & 5 \end{matrix}$$

$$(2,5) = \begin{cases} (22) (3 \ 5) = AB, CB \rightarrow \{S,C\} \\ (23) (4 \ 5) = BA, BS = \{A\} \\ (24) (45) = BA, BC = \{A,S\} \end{cases}$$

If the cell (last cell) (1,5) contains the starting symbol 'S' we can generate the string from the given Grammar

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

Now check where the starting symbols are present in the table, i.e. it is present at cells (1,2) (3,4) (4,5) (2,5) (1,5)

baaba
1 2 3 4 5

$$\begin{matrix} \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ \{ba\} & \{ab\} & \{ba\} & \{aaba\} & \{ba, ab, aaba\} \end{matrix} \quad \text{Given string}$$

② GRIE BACH

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

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

Advantages

- 1> The no. of
- 2> GNF is

CONVERSION

- 1> Push the
- 2> push RHS
- 3> Add final

$$① S \rightarrow aSb,$$

$$\begin{aligned} S &\rightarrow aSB \\ B &\rightarrow b \end{aligned}$$

GRIEBACH

NORMAL FORM (GNF)

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

Ex: $A \rightarrow aBCDE$

$A \rightarrow a$

Advantages

- 1) The no. of steps required to generate a string of length $|w|$ is $|w|$.
- 2) GNF is useful in order to convert a CFG to "push down Automata".

CONVERSION OF CFG TO PDA:

1) Push the start symbol 'A' onto the stack $\delta(q_0, \epsilon, Z_0) = (q_1, AZ_0)$

2) push RHS of 'A' as follows $\delta(q_1, a, A) = (q_1, \alpha)$ if $A \rightarrow \alpha$ is in Grammar

3) Add final state with

$\delta(q_1, \epsilon, Z_0) = (q_f, \epsilon)$

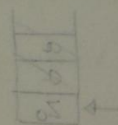
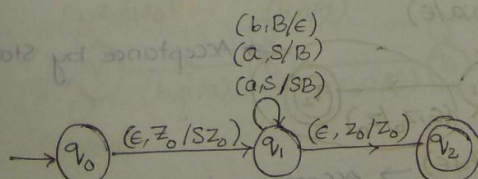
$\delta(q_1, b, A) = (q_1, B)$
if $A \rightarrow bB$ is in Grammar

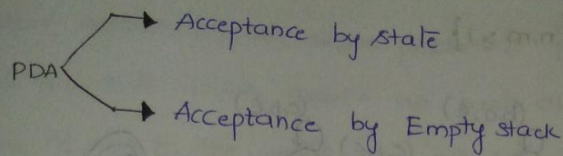
① $S \rightarrow aSb/ab \rightarrow$ Not in GNF

$S \rightarrow aSB/ab$

$B \rightarrow b$

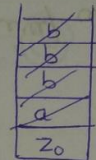
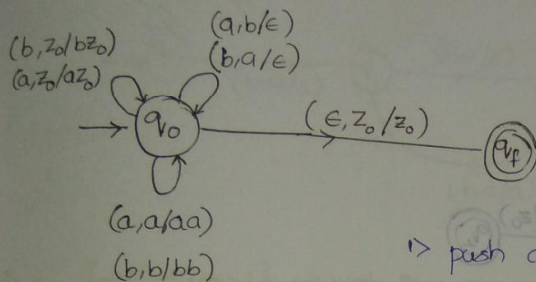
In GNF





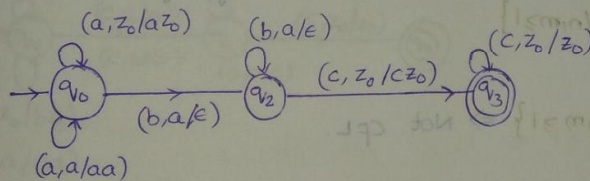
② $w \in (a^*b^*) / \eta_a(w) = \eta_b(w)$

Let the string be $abbbba$
 $\uparrow \uparrow \uparrow \uparrow \uparrow$



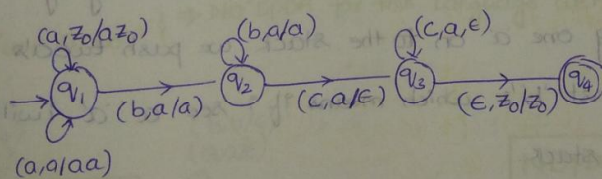
- 1> push a's onto the stack.
- 2> If you see b, pop one 'a' & vice versa.
- 3> If your Alphabet is 'b' & Bottom of Stack = z0
 => push 'b' on stack.

③ $L = \{a^n b^n c^m / n, m \geq 1\}$

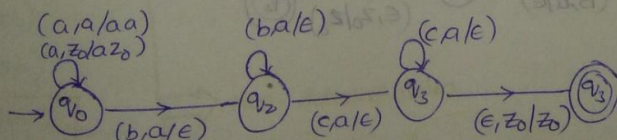


L-69

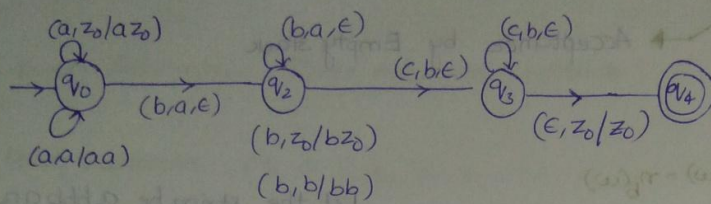
④ $L = \{a^n b^m c^n / n, m \geq 1\}$



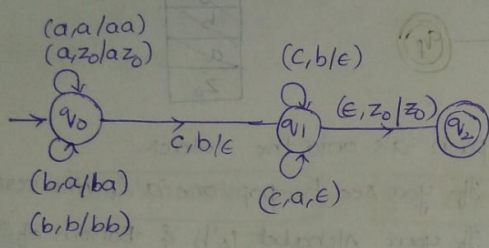
⑤ $L = \{a^{m+n} b^m c^n\} (m, n \geq 1)$



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



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



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

⇒ context free languages

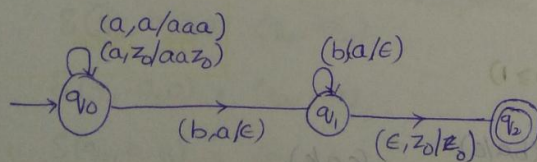
$$L = \{a^n b^m c^m d^n / n, m \geq 1\}$$

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

$$⑩ L = \{a^n b^{2n} / n \geq 1\}$$

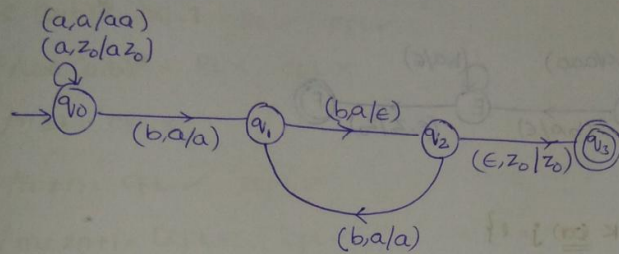
The language accepted is $L = \{abb, aabbbb, aaabbbbbbb, \dots\}$

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



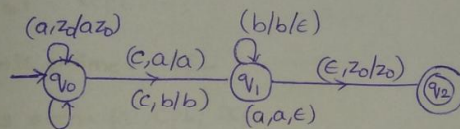
52

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



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

② $L = \{wcw^R / w \in (a,b)^+\}$



(b, z0/bz0)
(a, a/aa)
(a, b/ab)
(b, a/ba)
(b, b/bb)

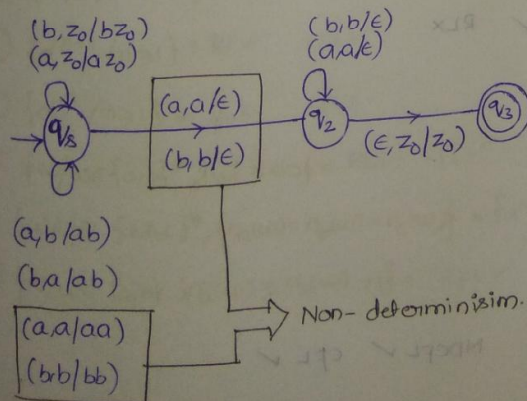
instead

d pop

sh z0's

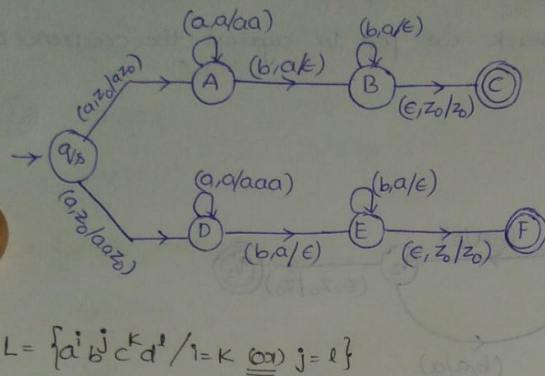
L-70

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

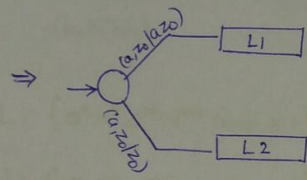


NDPDA is not as powerful as DPDA

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

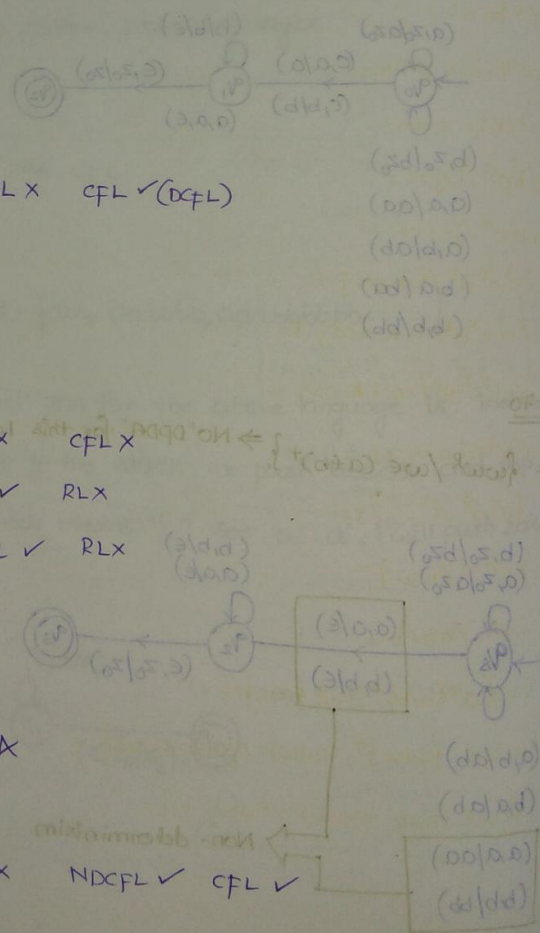


$$\begin{aligned} \textcircled{3} L &= \{a^i b^j c^k d^l / i=k \text{ or } j=l\} \\ &= \{a^m b^j c^m d^l\} \cup \{a^i b^m c^k d^m\} \end{aligned}$$



L-71

- ① $a^{m+n} b^n c^m / n, m \geq 1 \Rightarrow RL \times \quad CFL \checkmark (DCFL)$
 $a^m b^{m+n} c^n / n, m \geq 1$
- ② $a^m b^n c^{m+n} = RL \times \quad CFL \checkmark$
- ③ $a^m b^m c^n d^n = RL \times \quad CFL \checkmark$
- ④ $a^m b^n c^m d^n / m, n \geq 1 = RL \times \quad CFL \times$
- ⑤ $a^m b^n c^n d^m / m, n \geq 1 = CFL \checkmark \quad RL \times$
- ⑥ $a^m b^i c^m d^k / m, n \geq 1 = DCFL \checkmark \quad RL \times$
- ⑦ $a^m b^n / m > n = DCFL \checkmark$
- ⑧ $a^n b^{2n} / n \geq 1 = DCFL \checkmark$
- ⑨ $a^n b^{n^2} / n \geq 1 = RL \times \quad CFL \times$
- ⑩ $a^n b^{2^n} / n \geq 1 = RL \times \quad CFL \times$
- ⑪ $ww^R / w \in (a|b)^+ = RL \times \quad DCFL \times \quad NDCFL \checkmark \quad CFL \checkmark$
- ⑫ $ww / w \in (a|b)^+ = RL \times \quad \times CFL$



- ⑬ $a^n b^n c^m / n$
- ⑭ $a^n b^n c^n d^n$
- ⑮ $a^n b^{2^n} c^{3^n} / n$
- ⑯ $xcy / x, y \in \Sigma^*$
- ⑰ $xx^R / x \in \Sigma^*$
- ⑱ $ww^R / w \in \Sigma^*$
- ⑲ $a^n b^{3^n} / n \geq 1$
- ⑳ $a^m b^n / m \neq n$
- ㉑ $a^m b^n / m = n$
- ㉒ $a^i b^j / i \neq j$
- ㉓ $a^{n^2} / n \geq 1$
- ㉔ $a^{2^n} / n \geq 1$
- ㉕ $a^{n!} / n \geq 1$
- ㉖ $a^m / m \text{ is even}$
- ㉗ $a^i b^j c^k / i \geq j$
- ㉘ $a^i b^j c^k / j = i$
- ㉙ $a^i b^j c^k d^l / i = j$
- ㉚ $a^i b^j c^k d^l / i = j = k$
- ㉛ $a^m b^l c^k d^n$
- ㉜ $a^n b^{4m} / n, m$
- ㉝ $\{a^3 a^8, a^{11}\}$
- ㉞ $\{a^{2n+1} / n \geq 1\}$
- ㉟ $\{a^n / n \geq 1\}$
- ㊱ $\{w / w \in \Sigma^*\}$
- ㊲ $\{w / w \in \Sigma^*\}$
- ㊳ $\{w / w \in \Sigma^*\}$

33) $a^n b^n c^m / n > m$ XRL XCFL

34) $a^n b^n c^n / n \leq 10^{10} = RL \checkmark$ DCFL $\checkmark$ CFL $\checkmark$

35) $a^n b^{2n} c^{3n} / n \geq 1 = RL \times$ CFL $\times$

36) $xy / x, y \in (0,1)^+ = RL \checkmark$ DCFL $\checkmark$ CFL $\checkmark$

37) $xx^R / x \in (a,b)^*, |x|=1 = RL \checkmark$ CFL $\checkmark$

38) $www^R / w \in (a,b)^* = RL \times$ CFL $\times$

39) $a^n b^{3n} / n \geq 1 = CFL \times$

40) $a^m b^n / m \neq n = CFL \checkmark$ DCFL $\checkmark$

41) $a^m b^n / m = 2n+1 = DCFL \checkmark$ CFL $\checkmark$

42) $a^i b^j / i \neq 2j+1 = DCFL \checkmark$

43) $a^{n^2} / n \geq 1 = CFL \times$

44) $a^{2^n} / n \geq 1 = XCFL$

45) $a^{n!} / n \geq 1 = XCFL$

46) a^m / m is prime = XCFL

47) a^k / k is even = RL, CFL, DCFL

48) $a^i b^j c^k / i > j > k = XCFL$

49) $a^i b^j c^k / j = i+k = CFL, DCFL \checkmark$

50) $a^i b^j c^k d^l / i=k \text{ or } j=l = CFL \checkmark$

51) $a^i b^j c^k d^l / i=k \text{ and } j=l = CFL \times$

52) $a^m b^l c^k d^n / m, l, k, n \geq 1 = RL \checkmark$ CFL $\checkmark$

53) $a^n b^4 m / n, m \geq 1 = RL \checkmark$ CFL $\checkmark$

54) $\{a^3, a^8, a^{13}, \dots\} = RL \checkmark$ CFL $\checkmark$

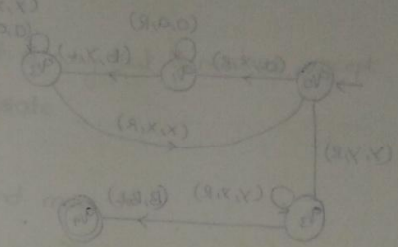
55) $\{a^{2n+1} / n \geq 1\} = RL \checkmark$ CFL $\checkmark$

56) $\{a^n / n \geq 1\} = RL \times$ CFL $\times$

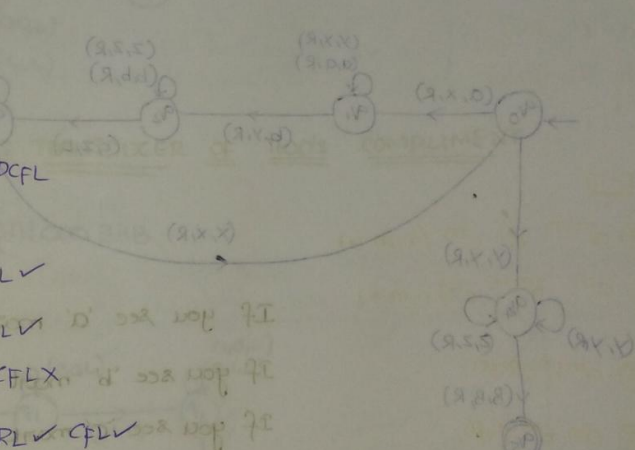
57) $\{w / w \in \{a,b\}^*, |w| \geq 100\} = RL \checkmark$ CFL $\checkmark$

58) $\{w / w \in \{a,b,c\}^*, \{n_a(w) = n_b(w) = n_c(w)\} = \{a^n b^n c^n / n \geq 1\} \times CFL$

59) $\{w / w \in \{a,b\}^* \{n_a(w) \geq n_b(w) + 1\} = CFL \checkmark$

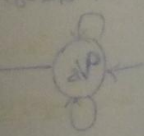


TRANSUCER CONVERSION



$(a, x, (p)) = (a, (p))$

$(a, x, (p)) = (x, (p))$



(d, a)	(d, a)
(d, a)	(d, a)
(d, a)	(d, a)
(d, a)	(d, a)

(36)

③ TURING MACHINE AS TRANSDUCER OF ONE'S COMPLIMENT

(57)

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

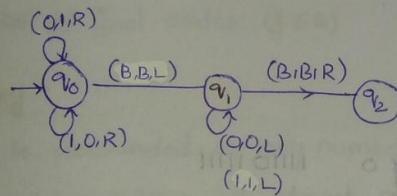
⇒ TM to find ones complement of a Binary number

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

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

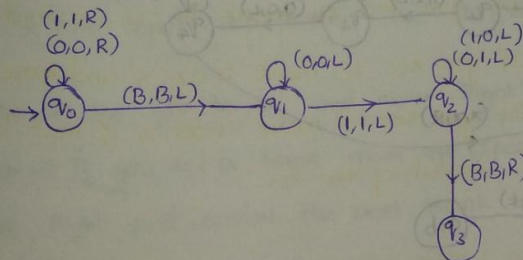
⇒ If i see a '1' mark it as '0' and move 'R'

TRANSDUCER = CONVERTER



④ TURING MACHINE AS TRANSDUCER OF TWO'S COMPLIMENT

The Given string BBB 0111000 BBB



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

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

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

THE STANDARD TURING MACHINE

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

Q = Set of states

Σ = Input Alphabet

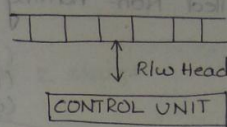
Γ = Tape Alphabet

δ = Transition function

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

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

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



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

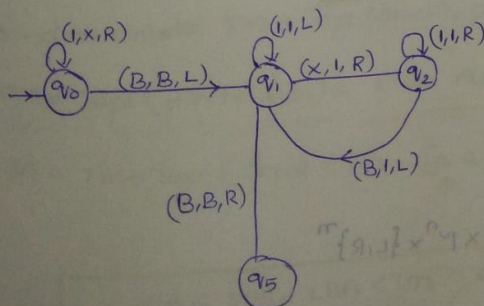
Summary

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

8) TM AS A COPIER

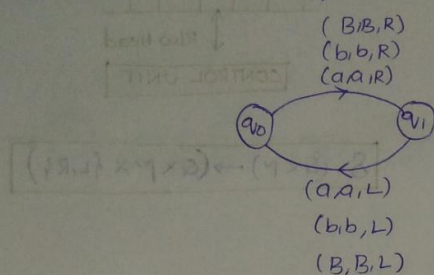
copier means the Turing machine should produce the copy of the given input. If the input is B11B then the TM should produce B1111BB

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



9) NON-HALTING TURING MACHINE

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



10) TURING THESIS

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

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

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

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

11) MODIFICATIONS TO STANDARD TURING MACHINE

1) TM with stay option

2) TM with semi infinite tape

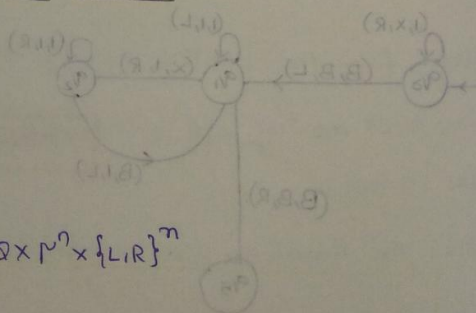
3) Offline TM

4) Multitape TM $\delta: Q \times \Sigma^n \rightarrow Q \times \Sigma^n \times \{L, R\}^n$

5) Jumping TM

6) Non-Erasing TM

7) Always writing TM



8) Multidimensional

9) Multitape TM

10) Automata with

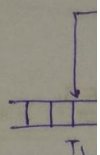
11) TM with only

12) Multitape TM

13) Non-determinist

14) A NPDA with

15) UNIVERSAL TM



(Rep of TM)

$\delta(q_1, 1) =$

$\delta(q_2, 1) =$

$\delta(q_3, 1) =$

→ Every "TM"

13) LINEAR BOUN

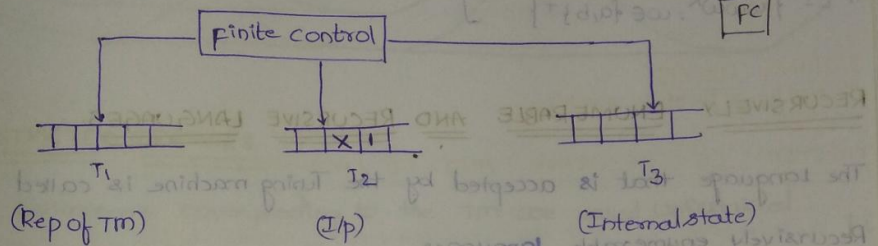
⇒ Non-determinist

⇒ NTM + Tape

⇒ LBA = TM +

- 66
- 8) Multidimensional TM $\delta: Q \times \Sigma^n \rightarrow Q \times \Sigma^n \times \{L, R, U, D\}$
 - 9) Multithead TM
 - 10) Automata with a Queue
 - 11) TM with only 3 states
 - 12) Multitape TM with stay option and atmost 2 states
 - 13) Non-deterministic TM $\delta: Q \times \Sigma \rightarrow 2^{Q \times \Sigma \times \{L, R\}}$
 - 14) A NPDA with 2 independent stacks $\delta: Q \times (\Sigma \cup \epsilon)^* \times \Sigma^* \times \Sigma^* \rightarrow 2^{Q \times \Sigma^* \times \Sigma^*}$

12) UNIVERSAL TM AND ENCODING TM



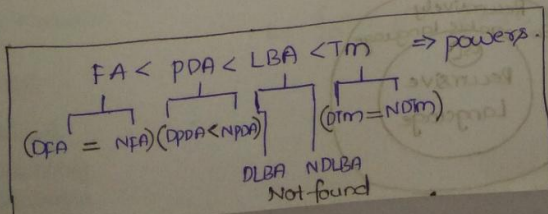
$$\begin{aligned}
 \delta(q_1, 1) &= (q_2, X, R) & Q &= (q_1, q_2, q_3, q_4, \dots) & q_1 &= 1 & a_1 &= 1 \\
 \delta(q_2, 1) &= (q_3, X, R) & \Gamma &= (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \dots) & q_2 &= 11 & a_2 &= 11 \\
 & & & & q_3 &= 111 & a_3 &= 111 \\
 & & & & q_4 &= 1111 & a_4 &= 1111 \\
 & & & & R &= 11 & L &= 1
 \end{aligned}$$

01010110110110

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

13) LINEAR BOUNDED AUTOMATA

- ⇒ Non-deterministic TM + Tape (stack) = PDA
- ⇒ NTM + Tape (Finite size) = Finite Automata
- ⇒ LBA = TM + Tape (Input size) $[a a b b]$



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

1) $L = \{a^n b^n c^n : n \geq 1\}$

2) $L = \{a^n : n \geq 0\}$

3) $L = \{a^n : n = m^2, m \geq 1\}$

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

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

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

7) $L = \{w^n : w \in (a+b)^+, n \geq 1\}$

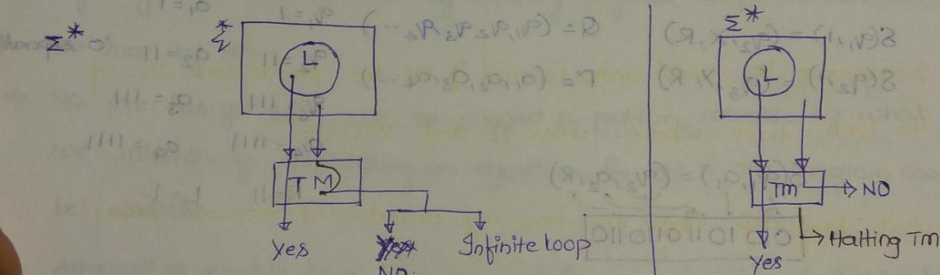
8) $L = \{ww^R : w \in \{a,b\}^+\}$

Context Sensitive Languages

"LBA" is a "Halting TM"

RECURSIVELY ENUMERABLE AND RECURSIVE LANGUAGES

The Language that is accepted by the Turing machine is called Recursively enumerable languages

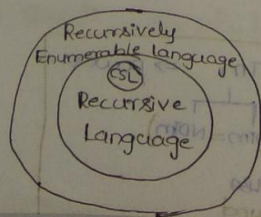


Recursively Enumerable Languages

Recursive Language

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

⇒ If there exists a Halting TM for a language then the language is called Recursive language



THE BIG PICTURE

UNRESTRICTED

⇒ The Grammar

Grammars

⇒ A Grammar

the form

① what language

$$S \rightarrow S_1 B$$

$$S_1 \rightarrow a S_1 b$$

$$b B \rightarrow b b b B$$

$$a S_1 b \rightarrow a a$$

$$B \rightarrow \lambda$$

so: $S \Rightarrow S_1$

$$\Rightarrow a S_1$$

$$\Rightarrow a a$$

$$\Rightarrow a a$$

$$S \Rightarrow S_1 B$$

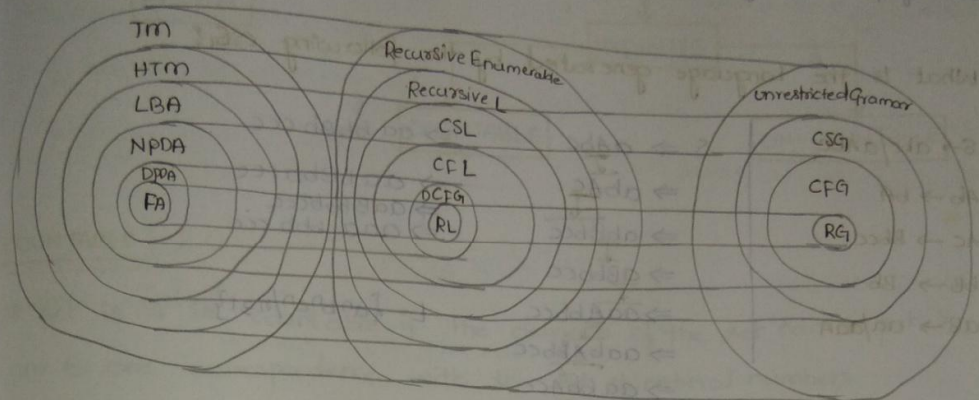
$$\Rightarrow a S_1 b B$$

$$\Rightarrow a S_1 b b b B$$

$$\Rightarrow a a b b b B$$

$$\Rightarrow a a b b b$$

THE HIERARCHY



UNRESTRICTED GRAMMARS

⇒ The Grammars corresponding to the TM are called Unrestricted Grammars.

⇒ A Grammar is called unrestricted if all the productions are of the form $u \rightarrow v$ where $u \in (V \cup \Sigma)^+$ and $v \in (V \cup \Sigma)^*$

① what language does the following unrestricted Grammar derive?

$$S \rightarrow S_1 B$$

$$S_1 \rightarrow a S, b$$

$$b B \rightarrow b b b B$$

$$a S, b \rightarrow a a$$

$$B \rightarrow \lambda$$

sol: $S \Rightarrow S_1 B$

$$\Rightarrow a S, b B$$

$$\Rightarrow a a B$$

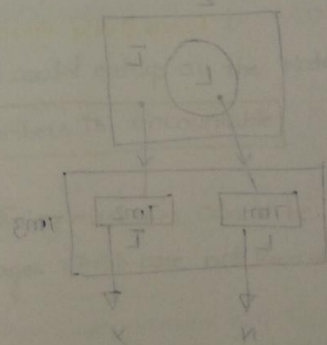
$$\Rightarrow a a$$

$$S \Rightarrow S_1 B$$

$$\Rightarrow a S, b B$$

$$\Rightarrow a S, b b b B$$

$$\Rightarrow a a b b b B$$



$$L = \{a^{n+1} b^{n+K} / n \geq 1, K = -1, 1, 3, 5, \dots\}$$

CONTEXT SENSITIVE GRAMMAR - EXAMPLE 4

64

A Grammar is said to be Context sensitive if all the productions are of form $x \rightarrow y$ where $(x,y) \in (V \cup T)^+$ and $|x| \leq |y|$

What is the Language generated by the following CSG?

$S \rightarrow abc/aAbc$	$S \Rightarrow aAbc$	$\Rightarrow aa bbb bccc$
$Ab \rightarrow bA$	$\Rightarrow abAc$	$\Rightarrow aa bbb bccc$
$Ac \rightarrow Bbcc$	$\Rightarrow abBcc$	$\Rightarrow aa bbb bccc$
$bB \rightarrow Bb$	$\Rightarrow aBbbcc$	$\Rightarrow aaa bbb ccc$
$aB \rightarrow aa/aAa$	$\Rightarrow aaAbbcc$	
	$\Rightarrow aabAbbcc$	$L = \{a^n b^n c^n / n \geq 1\}$
	$\Rightarrow aabbbAcc$	

EXAMPLE-3 [Not in GATE]

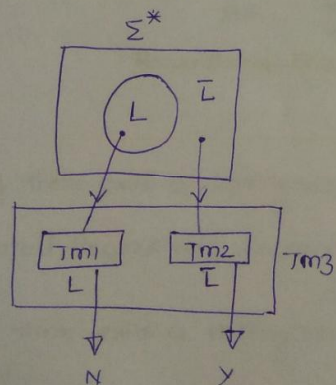
THEOREM ON RECURSIVE AND RE LANGUAGES

Recursively Enumerable

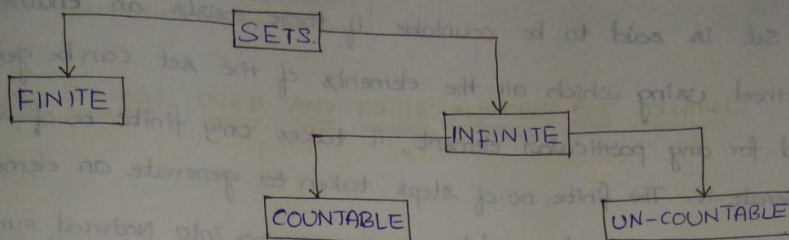
If a Language 'L' and its Complement $\bar{L}$ are both RE, then both the Languages are Recursive. If 'L' is Recursive, then $\bar{L}$ is also Recursive and consequently both are Recursively enumerable.

$\Rightarrow$ There is no membership Algorithm for RE Languages.

$\Rightarrow$ Membership Algorithm exists for the Recursive Languages



COUNTABILITY



COUNTABLE

A set 'S' is countable if the elements of the set can be put in one to one correspondence with the set of natural numbers

UNCOUNTABLE

A set is uncountable if it is infinite and not countable.

$$\begin{array}{l}
 E = \{0, 2, 4, 6, 8, 10, 12, \dots\} \\
 N = \{1, 2, 3, 4, 5, 6, \dots\}
 \end{array}
 \begin{array}{c}
 \boxed{\begin{array}{c} 2i \\ \downarrow \\ i+1 \end{array}}
 \end{array}
 \begin{array}{l}
 \text{one to one correspondence is there} \\
 \text{so countable}
 \end{array}$$

$$\begin{array}{l}
 O = \{1, 3, 5, 7, 9, 11, \dots\} \\
 N = \{1, 2, 3, 4, 5, 6, \dots\}
 \end{array}
 \begin{array}{c}
 \boxed{\begin{array}{c} 2i+1 \\ \downarrow \\ i+1 \end{array}}
 \end{array}
 \begin{array}{l}
 \text{one-one correspondence is there so} \\
 \text{countable}
 \end{array}$$

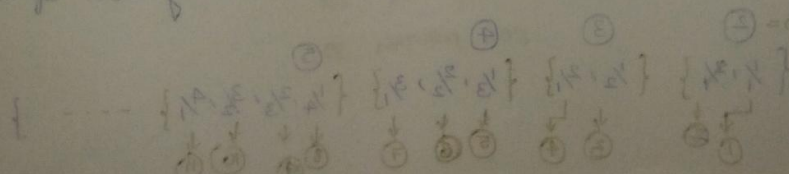
Real No's $\{0 \text{ --- } 1\}$

→ Infinite No's are present and could eat up all the Natural

No's set so set of Real numbers is uncountable

→ There are some languages for which Turing machines cannot be constructed. ⇒ There are some languages which are not Recursively

Enumerable



ALTERNATIVE DEFINITION OF COUNTABILITY

66

A Set is said to be countable if there exists an enumeration method using which all the elements of the set can be generated and for any particular element, it takes only finite no. of steps to generate it. The finite no. of steps taken to generate an element can be used as its index and hence a mapping into Natural numbers.

Set of all Even numbers = for $(i=0 \text{ to } \infty)$

0, 2, 4, 6, 8, ...
↓ ↓ ↓ ↓ ↓
1 2 3 4 5 → index
step steps

generate $(2i)$

Set of all odd numbers = for $(i=0 \text{ to } \infty)$

1, 3, 5, 7, 9, 11, 13, ...
↓ ↓ ↓ ↓ ↓
1 2 3 4 5 6 → index

generate $(2i+1)$

Set of all Real numbers = No Enumeration method so Not countable

SET OF ALL QUOTIENTS ARE COUNTABLE

Quotients of P/Q where $(P, Q) \in \mathbb{Z}^+$

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

$= \{1/1, 1/2, 1/3, 1/4, 1/5, 1/6, \dots\} \times 2/1$ will not be generated so this

Enumeration method is wrong.

so follow this enumeration method, of (P/Q)

1) find the sum of $(P+Q)$ and go in increasing order of the sum

sum = ② ③ ④ ⑤

$\{1/1, 2/1\}$ $\{1/2, 2/2\}$ $\{1/3, 2/3, 3/1\}$ $\{1/4, 2/4, 3/4, 4/1\}$...

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪

66

So Every P_n can be generated after finite no. of steps and therefore the

Set of all Quotients are countable

(67)

SET OF ALL STRINGS OVER ANY FINITE ALPHABET ARE COUNTABLE

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

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

Now the Enumeration method that must be followed is

1) first Generate all the strings of length 0 (ϵ) and now generate all the strings of length 1 $\{a, b\}$ and in the set of length $\{1, 2, \dots\}$ follow the Lexicographic order.

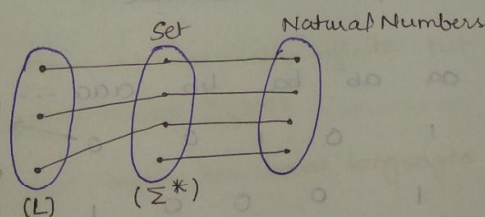
$$S = \{\epsilon, a, b, aa, ab, ba, bb, aaa, aab, \dots\}$$

$$\text{Steps} = \begin{matrix} \downarrow & \downarrow & \downarrow & \downarrow \\ 1 & 2 & 3 & 4 \end{matrix} \dots$$

$\therefore$ SET OF ALL POSSIBLE STRINGS OVER ANY FINITE ALPHABET SET ARE COUNTABLE, i.e. Σ^* IS COUNTABLE

Now, Another important property is.

Every subset of countable set is either finite or countable



Now we know that Σ^* is countable and $L \subseteq \Sigma^*$ therefore

"Every Language is countable" $\rightarrow$ NOT ALL LANGUAGES

$\rightarrow$ GIVEN A LANGUAGE ITS STRING CAN BE PUT IN ONE-ONE CORRESPONDANCE OF NATURAL NOS.

SET OF ALL TURING MACHINES ARE COUNTABLE

$\Sigma = \{0,1\}$ (1) $\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, \dots\}$ = countable

(2) Every TM can be encoded as a strings of 0s and 1s

(3) Set of all TM's (S) $\{s \in \Sigma^*\}$

(4) Every subset of countable set is finite or countable

$\therefore$ SET OF ALL TM'S ARE COUNTABLE

IMPLICATIONS OF THE FACT THAT THE TURING MACHINES ARE COUNTABLE

TM's are countable

$\Rightarrow$ Recursively Enumerable languages are Countable

$\Rightarrow$ Recursive Languages are countable

$\Rightarrow$ CSL are countable $\Rightarrow$ LBA are Countable

$\Rightarrow$ CFL are countable $\Rightarrow$ PDA are countable

$\Rightarrow$ RL are countable $\Rightarrow$ Finite Automata are countable

DIAGONLIZATION METHOD TO PROVE SET OF ALL LANGUAGES ARE

UNCOUNTABLE

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

Σ^*	ϵ	a	b	aa	ab	ba	bb	aaa	...
1)	0	1	1	1	0	1	0	0	...
2)	0	1	0	1	0	0	0	1	...
3)	1	0	1	1	1	0	0	0	...
4)	1	0	0	0	0	0	1	0	...
5)	0	1	1	0	0	1	1	0	...
6)	1	0	1	0	1	0	1	0	...
7)	1	1	1	1	0	0	0	0	...
8)	0	1	1	1	0	1	1	1	...

Represent elem power set $\therefore \{0,1\}^{\Sigma^*}$

10011110

Not in the language

69

There exist some languages which a Turing machine cannot accept.

COUNTABLE

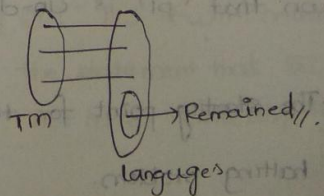
⇒ The cartesian product of finite no. of countable sets is countable

$$2^{\Sigma^*}$$

ARE

• in power
• $\frac{1}{2}$ of power

○ 1 } Compl
○ } -ment



COMPUTABILITY AND DECIDABILITY

Computability

and

Decidability

function

$$f(n) = (n^2 + 1)$$

Algo (on TM (Halts) ✓

Tape)

Problem

A stmt whose True or false then it is decidable and is either

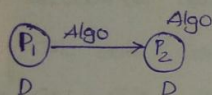
Ex: 2^n is a prime

D = set of all Natural no's

Ex: A given Grammar G_1 is Ambiguous
Domain = {Set of all CFGs}

→ un
decidable

Given an instance of a problem, it is always decidable



Given an problem 'p1' and we can convert into the problem p2 using an Algorithm and the problem 'p2' can be solved using another Algorithm, which means that there exists an Algorithm for problem 'p1'. The conversion should be in such a way that the solution to 'p2' should also be a solution to p1. which means if the ^{sol to} problem p2 is True then the solution to p1 should also be True.

And the procedure of conversion is called "Reducability". If you find out that if 'p2' is decidable then 'p1' is also decidable. But before it is known that 'p1' is un-decidable then the problem 'p2' is undecidable

⇒ The starting point for the problem of decidability is the halting problem.

TM - HALTING

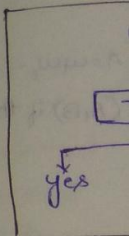
Given the d
started with
halts.

Theorem

If the halting
be Recursive

Recursively

TM



⇒ No member

Consider a

Σ^*

$L = RE$

$TM = m$

(M, w)

TM- HALTING PROBLEM

Given the description of a TM 'M' and an input 'w' does 'M' when started with 'w', does 'M' when started with 'w' as its input eventually halts.

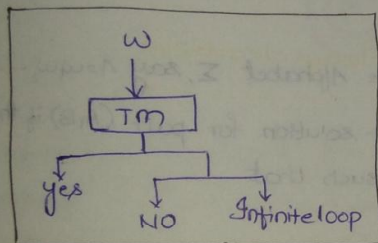
Theorem

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

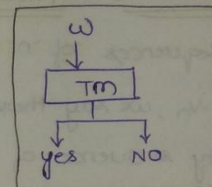
Recursively Enumerable

Recursive language

TM



TM which halts.

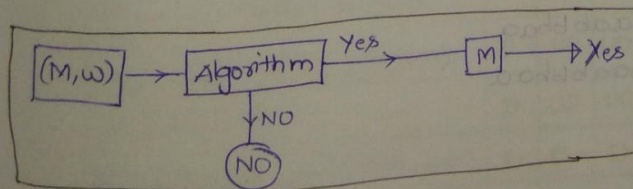
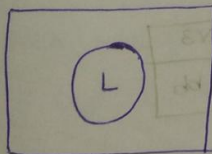


⇒ No membership Algorithm

⇒ Membership Algo exists

Consider a Language 'L' which is Recursively Enumerable
(No halting TM exists)

Σ^*
L = RE
TM = M



∴ This is the membership Algorithm we got for the 'REL' but it will contradict the statement that 'REL do not have Membership Algo'.

SOME UNDECIDABLE PROBLEMS BASED ON TM HALTING PROBLEM

- 1) The state entry problem is, given a TM, a state $(q \in Q)$ and $w \in \Sigma^+$, decide whether or not the state 'q' is ever entered when 'M' is applied to 'w'. This is undecidable.
- 2) Given a TM M, whether or not M halts if started with a blank space. This is undecidable.
- 3) Almost any problem related to RE languages is undecidable.

POST CORRESPONDENCE PROBLEM

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

$$w_i w_j \dots w_k = v_i v_j \dots v_k$$

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

Ex:

$$A = \begin{array}{|c|c|c|} \hline w_1 & w_2 & w_3 \\ \hline a & ab & bba \\ \hline \end{array}$$

$$B = \begin{array}{|c|c|c|} \hline v_1 & v_2 & v_3 \\ \hline baa & aa & bb \\ \hline \end{array}$$

The post correspondence solution is $(3, 2, 3, 1)$

$$w_3 w_2 w_3 w_1 = bbaabbbbaa$$

$$v_3 v_2 v_3 v_1 = bbaabbbbaa$$

COMPLEXITY

P-CLASS

The set of TM in polyn

NP-CLASS

The set of time is cal

$$(a \vee b \vee c)$$

$$2 \times 2 \times 2$$

$$= 2^n$$

$$= O(2^n)$$

DECIDABILITY

BLEM (72)

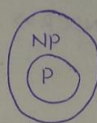
COMPLEXITY CLASSES

P-CLASS

The set of all languages that are accepted by some deterministic TM in polynomial time

NP-CLASS

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



NDTM $\rightarrow$ DTM

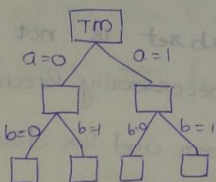
$O(n^k) \rightarrow O(2^n)$

(avbvc)

$2 \times 2 \times 2$

$= 2^n$

$= O(2^n)$



DECIDABILITY TABLE

PROBLEM	RL	DFL	CFL	CSL	Recurs- ive Lan- guage	REL
Does $w \in L$? (Membership Algo)	D	D	D	D	D	UD
Is $L = \emptyset$? (Emptiness problem)	D	D	D	UD	UD	UD
$L = \Sigma^*$ (Completeness problem)	D	UD	UD	UD	UD	UD
Is $L_1 = L_2$ (Equality problem)	D	UD	UD	UD	UD	UD
Is $L_1 \subseteq L_2$ (Subset problem)	D	UD	UD	UD	UD	UD
Is $L_1 \cap L_2 = \emptyset$	D	UD	UD	UD	UD	UD
Is L finite or not? (Finiteness)	D	D	D	UD	UD	UD
Is complement of L a language of same type or not?	D	D	D	UD	UD	UD
Is intersection of 2 languages of same type	D	UD	UD	UD	UD	UD
Is L Regular Language	D	D	UD	UD	UD	UD

BLEM (72)

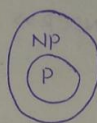
COMPLEXITY CLASSES

P-CLASS

The set of all languages that are accepted by some deterministic TM in polynomial time

NP-CLASS

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



NDTM $\rightarrow$ DTM

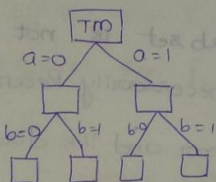
$O(n^k) \rightarrow O(2^n)$

(avbvc)

$2 \times 2 \times 2$

$= 2^n$

$= O(2^n)$



DECIDABILITY TABLE

PROBLEM	RL	DFL	CFL	CSL	Recurs- ive Lan- guage	REL
Does $w \in L$? (Membership Algo)	D	D	D	D	D	UD
Is $L = \emptyset$? (Emptiness problem)	D	D	D	UD	UD	UD
$L = \Sigma^*$ (Completeness problem)	D	UD	UD	UD	UD	UD
Is $L_1 = L_2$ (Equality problem)	D	UD	UD	UD	UD	UD
Is $L_1 \subseteq L_2$ (Subset problem)	D	UD	UD	UD	UD	UD
Is $L_1 \cap L_2 = \emptyset$	D	UD	UD	UD	UD	UD
Is L finite or not? (Finiteness)	D	D	D	UD	UD	UD
Is complement of L a language of same type or not?	D	D	D	UD	UD	UD
Is intersection of 2 languages of same type	D	UD	UD	UD	UD	UD
Is L Regular Language	D	D	UD	UD	UD	UD

(74)

CFL $\Rightarrow$ closed under

union

Concatenation

closure

(75)

$$\begin{array}{cc} L_1 & L_2 \\ S_1 \rightarrow & S_2 \rightarrow \end{array}$$

$$\begin{array}{cc} L_1 & L_2 \\ S_1 \rightarrow & S_2 \rightarrow \end{array}$$

$$\begin{array}{cc} L_1 & L_1^* \\ S_1 \rightarrow & \downarrow \\ & S \rightarrow S_1 S | \epsilon \end{array}$$

$$\text{CFL} \left\{ \begin{array}{l} S \rightarrow S_1 / S_2 \\ S_1 \rightarrow \\ S_2 \rightarrow \end{array} \right.$$

$$\begin{array}{l} S \rightarrow S_1 \cdot S_2 \\ S_1 \rightarrow \\ S_2 \rightarrow \end{array} \text{CFL}$$

CFL

Not closed under intersection $\Rightarrow L_1 \quad L_2$

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

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

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

NOT CFL

Not closed under complementation $= L_1 \cap L_2 = \overline{L_1 \cup L_2}$ If we assume $L_1 = \text{CFL}$ then $\overline{L_1} = \text{CFL}$ If we assume $L_2 = \text{CFL}$ then $\overline{L_2} = \text{CFL}$ then $\overline{L_1 \cup L_2}$ should be context-free and $\overline{L_1 \cup L_2}$

should be context-free which

means $L_1 \cap L_2$ should be CFL because butwe have seen that $L_1 \cap L_2$ is not CFL. so our assumption is wrong

That the complementation of CFL is Context-free

CFL is not closed under complementation

15.

15.

 $\Rightarrow L_1 / L_2$ means right quotient of L_1 with L_2

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

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

$$L_2 = \{01\}$$

$$L_1 / L_2 = \{\epsilon, 0, 1, 00, 11\} \Rightarrow \{\epsilon, 1, 0, 10, 100, 1000, \dots\}$$

$$\text{Now, } L_1 = 10^*1$$

$$L_2 = 0^*1$$

$$L_1 / L_2 = \left\{ \begin{array}{l} 11 \quad 101 \quad 1001 \quad 10001 \quad \dots \\ \swarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ 1 \quad 01 \quad 001 \quad 0001 \quad \dots \end{array} \right\}$$

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

$$L_1 / L_2 = 10^*$$

$$\frac{11}{01} = \phi, \quad \frac{101}{11} = \phi$$

 $\Rightarrow 11$ does not match with 101

→ Some of the decidable problems of the context free languages are
Emptiness, finiteness, Membership

1) Membership - CYK Algorithm

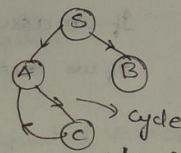
2) Emptiness — If you find starting symbol is useless then the lang is Empty.

3) Finiteness — Take Grammar 'G' and apply simplification and draw dependency Graph, If Graph has cycle then the lang is Infinite.

Say, the simplified Grammar is

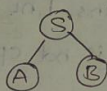
$S \rightarrow AB$
 $A \rightarrow aC|a$
 $C \rightarrow aA|b$
 $B \rightarrow a$

Dependency Graph



∴ lang is Infinite

$S \rightarrow AB$
 $A \rightarrow a$
 $B \rightarrow b$



∴ No cycle ⇒ the language is finite

PROPERTIES OF REGULAR LANGUAGES

CLOSED UNDER UNION, INTERSECTION, CONCATENATION, COMPLEMENTATION

KLEEN CLOSURE

1) $L_1 \cup L_2 \Rightarrow L_1$ has Regular expression R_1

L_2 has Regular expression R_2

∴ $L_1 \cup L_2 = [R_1 + R_2] = \text{RE (Regular language)}$

2) $L_1 \cap L_2 \Rightarrow L_1 \rightarrow R_1$
 $L_2 \rightarrow R_2$
 $L_1 \cap L_2 = [R_1 \cdot R_2] = \text{Regular language}$

3. $L_1 \cdot L_2 =$

→ Some of the decidable problems of the context free languages are
Emptiness, finiteness, Membership

1) Membership - CYK Algorithm

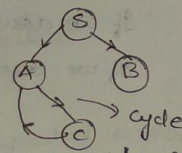
2) Emptiness - If you find starting symbol is useless then the lang is Empty.

3) Finiteness - Take Grammar 'G' and apply simplification and draw dependency Graph, If Graph has cycle then the lang is Infinite.

say, the simplified Grammar is

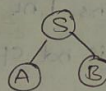
$S \rightarrow AB$
 $A \rightarrow aC|a$
 $C \rightarrow aA|b$
 $B \rightarrow a$

Dependency Graph



∴ lang is Infinite

$S \rightarrow AB$
 $A \rightarrow a$
 $B \rightarrow b$



∴ No cycle ⇒ the language is finite

PROPERTIES OF REGULAR LANGUAGES

CLOSED UNDER UNION, INTERSECTION, CONCATENATION, COMPLEMENTATION

KLEIN CLOSURE

1) $L_1 \cup L_2 \Rightarrow L_1$ has Regular expression R_1

L_2 has Regular expression R_2

∴ $L_1 \cup L_2 = [R_1 + R_2] = \text{RE (Regular language)}$

2) $L_1 \cap L_2 \Rightarrow L_1 \rightarrow R_1$
 $L_2 \rightarrow R_2$
 $L_1 \cap L_2 = [R_1 \cdot R_2] = \text{Regular language}$

3. $L_1 \cdot L_2 =$

4. $L_1^* \Rightarrow R_1$ is the RE for L_1

(76)

R_1^* is the RE of L_1^* $\therefore R_1^*$ is Regular Expression

$\therefore$ Regular languages are closed under closure

5. $\bar{L}_1 = \Sigma^* - L_1 \Rightarrow$ Language L_1 has DFA [Given L_1 is Regular]

$\Rightarrow$ Complement the DFA then we get DFA accepting $\bar{L}_1$

$\Rightarrow$ Since L_1 has DFA it is Regular

6. closed under difference and Reversal

$$\begin{aligned} L_1 - L_2 &\Rightarrow L_1 - L_2 \text{ can be written as } L_1 - L_2 = L_1 \cap \bar{L}_2 \\ &\downarrow \quad \downarrow \\ &= RL \cap RL \\ &= \underline{RL} \end{aligned}$$



infinite

L_1 is regular language $\Rightarrow L_1^R$ should be Regular

$\therefore \Rightarrow L_1 \rightarrow$ has DFA

$\Rightarrow$ Reverse the DFA then we get DFA accepting Reverse of the language

NOTATION

7. closed under Homomorphism and Inverse Homomorphism
 $\Sigma = \{a, b\}$ $\Gamma = \{0, 1, 2\}$ $\hookrightarrow$ function $h: \Sigma \rightarrow \Gamma^*$ $\hookrightarrow h^{-1}(w) = \{x/h(x) \in w\}$

$h(a) = 01$, $h(b) = 112$ Let $L = a^*b$
 $= (01)^*112 \Rightarrow$ Regular Expression
 $\therefore$ closed homomorphism

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

Let $L_1 = \{ababa\}$

$\frac{ababa}{1 \quad 1 \quad 0}$

$\Gamma(0) = a$

$\Gamma(1) = ab$

$\Gamma(2) = ba$

$h^{-1}(L_1) = \{110022, 102\}$

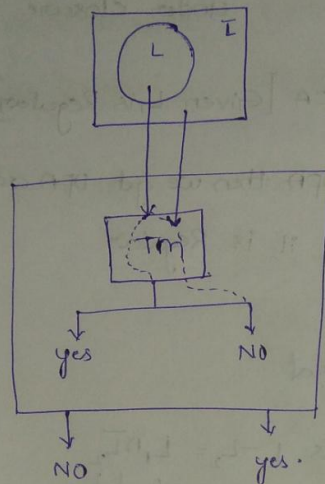
$\frac{ababa}{1 \quad 1 \quad 0}$

8. NOT CLOSED UNDER INFINITE UNION.

→ The complement of Recursive language is Recursive

⑧

Say there is a Recursive Language then it means there is a HTM



This TM just works in opposite way. which means if the inner TM says NO (the present in L) string is not then the outer TM says yes the string is present in $\bar{L}$ and viceversa.

∴ we got a halting TM that halts for $\bar{L}$ also. ∴ THE COMPLEMENT OF RECURSIVE LANGUAGE IS RECURSIVE.