9.07.18

"THEORY OF COMPUTATION"

* Categorized into 3 types:-

i) Automata Theory → There are various mathematical models:-
   → final automata automation
   → Language
   → Grammar

ii) Compatibility → Two types:-
   * Solvable
   * Not solvable

iii) Complexity → Two types:-
   * hard
   * easy

* Basis of Theory Of Computation:-

1) Symbol → a-z, A-Z, 0 1 2 ..... 9, 12 ....

2) Alphabet → Collection of symbols (Σ)
   ↳ { Σ(a, b) Σ(0, 1, 2 ...) }
     { Σ(a, b, c) Σ(a, b, c ...) }

3) String → Sequence of symbols over an alphabet

Example:-

Given ∈ (a, b)

String [a - a → String of length over Σ(a,b)]

3) Language – Collection of strings over an alphabet $\Sigma(a,b)$

Example → L = {a, b, aa, ab, bb, ba, aaa}
infinite

L_1 = Set of strings of length 2 over $\Sigma(a,b)$
= {ab, ba, aa, bb}
finite = $2^2 = 4$ (no of alphabet)

L_2 = Set of strings of length 3 over $\Sigma(a,b)$
= {aaa, bbb, aab, bba, aba, abb, bab, baa}
= $2^3 = 8$ = finite

Example of infinite no.

L = Set of strings start with 'b' symbol over an alphabet $\Sigma(a,b)$
= {ba, bb, bba ...}
= infinite.

Q Whether a string is present in a language or not?

L_1 = Set of string of length 3 over an alphabet $\Sigma(a,b)$
= {aaa, bbb, aab, bba, aba, bab}

* Linear Search :-

Grammar
↓
Language
↓
Mathematical Calculation
or Automata or finite model

```
int a[], b[];
{
printf("Enter a and b:");
scanf("%d %d", &a, &b);
for(i=a; i<=a; length()++);
if(a[i]==b)
{
print(
```

* Finite Automata → finite representation of the language of finite sets.

11.07.18 Mathematical Proof :-

1. Proof by Contradiction :-

Q Prove that $\sqrt{2}$ is an irrational number.

Let us assume $\sqrt{2}$ is a rational no.

$\Rightarrow \sqrt{2} = \frac{p}{q}$ } p and q are co-prime, $q \neq 0$.

$\Rightarrow (\sqrt{2})^2 = \frac{p^2}{q^2} \Rightarrow 2 = \frac{p^2}{q^2} \Rightarrow q^2 = \frac{p^2}{2}$

Theorem (imp)

If a prime number divides p^2, that prime no. divide p

Let $p = 2x$] Now squaring on both sides

$\Rightarrow p^2 = 2^2 x^2$

$\Rightarrow 2q^2 = 4x^2$

$\Rightarrow \frac{2q^2}{4} = x^2 \Rightarrow x^2 = \frac{2q^2}{4}$] $\rightarrow$ 2 divides q^2 then 2 divides q

Q Prove that the squareroot of any prime no. is an irrational no.

Soln. Let prime no. be x

Let us assume that $\sqrt{x}$ is a rational no.

$\Rightarrow (\sqrt{x})^2 \Rightarrow \sqrt{x} = \frac{p}{q}$ } p and q are co-primes, $q \neq 0$.

$\Rightarrow (\sqrt{x})^2 = \frac{p^2}{q^2}$ } Squaring on both sides.

$\Rightarrow x = \frac{p^2}{q^2}$]

$\Rightarrow q^2 = \frac{p^2}{x}$]

From theorem we see that,

2 divides p^2 then

2 divides p.

Let $p = 2x$] from theorem.

Now squaring on both sides:-

$\Rightarrow p^2 = x^2 x^2$

$\Rightarrow 2q^2 = x^2 x^2$

$\Rightarrow q^2 = x^2 x^2$

$\Rightarrow x^2 = \frac{q^2}{x^2}$

$\Rightarrow$ x divides q^2 so x divides q

Hence proved.

2) Proof by induction :-

Q Prove that :-

$$1^3 + 2^3 + 3^3 + \dots + n^3 = \frac{n^2(n+1)^2}{4} \quad \text{for } n \geq 1$$

Soln. Basis $\rightarrow$ for $n = 1$

then $\frac{1^2(1+1)^2}{4} = \frac{1 \cdot (2)^2}{4} = \frac{4}{4} = 1$.

So it is true for $n = 1$.

for $n = n$.

then $\frac{n^2(n+1)^2}{4} = \frac{n^2(n^2+1+2n)}{4}$

$= \frac{n^4 + n^2 + 2n^3}{4}$

Induction → Assume that $S(n)$ is true for $n=k$.

$S(k) = 1^3+2^3+\dots+k^3 = \frac{k^2(1+k)^2}{4}$ is true

Now assume $S(n)$ is true for $n=(k+1)$

T.P→ $S(k+1) = 1^3+2^3+3^3+\dots+(k)^3+(k+1)^3 = \frac{(k+1)^2(k+2)^2}{4}$

LHS → $1^3+2^3+3^3+\dots+k^3+(k+1)^3$

$= \frac{k^2(1+k)^2}{4} + (k+1)^3$

$= (k+1)^2\left[\frac{k^2}{4}+(k+1)\right]$

$= (k+1)^2\left[\frac{k^2+4(k+1)}{4}\right]$

$= (k+1)^2\left[\frac{k^2+4k+4}{4}\right]$

$= \frac{(k+1)^2(k+2)^2}{4}$ RHS

So LHS=RHS.

proved//.

Q Prove that $7^n - 3^n$ is divisible by 4.

Soln Basis → Assume that let $n=1$

So $7^1-3^1 = 4$ } which is divisible by 4.

Induction → Let us assume that $n=k$

So $7^k - 3^k = 4x$ } which is divisible by 4.

Now we have to prove for $n=(k+1)$

TP→ $7^{(k+1)} - 3^{(k+1)}$ is divisible by 4.

$= 7^k \cdot 7 - 3^k \cdot 3$

$= 7(4x+3^k) - 3^k \cdot 3$

$= 28x + 7\cdot 3^k$

OR

TP→ $7^{(k+1)} - 3^{(k+1)}$ is divisible by 4.

LHS → $7^{(k+1)} - 3^{(k+1)}$

$= 7^k \cdot 7 - 3^k \cdot 3 + 7\cdot 3^k - 7\cdot 3^k$

$= 7(7^k - 3^k) - 3^k \cdot 3 + 7\cdot 3^k$

$= 7(4x) + 4\cdot 3^k$

So this is proved.

Q Prove that $2(7)^n + 3(5)^n - 5$ is divisible by 24

Soln Basis → Let $n=1$

$= 2(7)^1 + 3(5) - 5$

$= 14+15-5$

$= 29-5 = 24$ which is divisible

So $n=1$ is true.

Now let us assume that $n=k$.
So $2(7)^k+3(5)^k-5$ is divisible by 24x.

Now we have to prove for $n=(k+1)$

LHS → $2(7)^{k+1}+3(5)^{k+1}-5$

$2[7^k \cdot 7^1]+3[5^k 5]-5$

$7\left[2[7^k]\right]+\left[3\cdot 5^k\right]5-5$

$=7\left(2\cdot 7^k+3\cdot 5^k-5-3\cdot 5^k+5\right)+3(5^k)5-5$

$=7(2\cdot 7^k+3\cdot 5^k-5-3(5)^k+5)+3(5)^k 5-5$

$=7(24x)-3(5)^k 7+5\cdot 7+3(5)^k 5-5$

$=7(24x)-21(5^k)+35+35(5)^k-5$

$=7(24x)-6\cdot 5^k+30$ $\quad x=21\cdot 6$

$=24(7x)-6(4y)$

$=24(7x)-24y$

$=24(7x-y)$

Q Proof that $2^n > n$ for all +ve integer $n \geq 1$ or $n \geq 0$

Soln.. Basis → for $n=1$
So $2^1>1$
It is true for $n=1$.

Induction → Now let us assume that it is true for $n=k_0$.
$2^k > k_0$ → true.

We have to prove that $2^{k+1} > (k+1)$

LHS → 2^{k+1}
$2^k \cdot 2^1$

$2^k > k$
$2^k \cdot 2 > k \cdot 2$
$2^{k+1} > 2k$
$2^{k+1} > k+k_0 \to k$.

So $(k+1)$ is true.

12.07.18

Finite Automata

- Deterministic
- Non-deterministic

i) Deterministic Finite Automata :-

Ex-1) Construct a DFA that accepts a set of strings starting with symbol 'a' over an alphabet $\Sigma = \{a, b\}$.

$L = \{a, aa, ab, abb, aba \ldots\}$ → infinite

Ex-2 Construct a DFA that accepts a set of strings ending with symbol 'a' over an alphabet $\Sigma = \{a, b\}$.

$L = \{a, ba, bba, baa, aba, \ldots\}$ infinite.

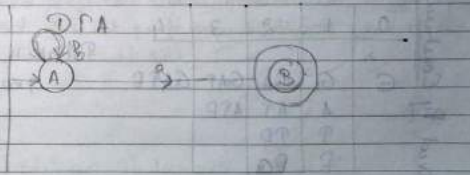
Strings

a

ba

A | B | B | B

DFA



16.07.18

* Language And Strings:-

The formal language consist of a finite no of symbols which is the building block of formal language.

→ Alphabet - It is finite set of symbols. Denoted by 'Σ'.

Ex - Σ = {a, b}

Σ1 = {0, 1} → binary alphabet

Σ2 = {Δ, 0, *, □, $}

→ String - It is a sequence of symbols derived from an alphabet. Denoted as 'ω'

Ex - 010010 derived from Σ = {0, 1}

Ex - abbcab derived from Σ1 = {a, b, c}

Ex - ΔΔ00□□ derived from Σ2 = {Δ0□}

Empty string (ω) = ∈, |ω| = 0

→ Substring - If ω is a string and z is a substring of ω then the symbols of z must be occuring consecutively in ω

Ex → ω = GATE

n = |ω| = 4

$\frac{n(n+1)}{2} + 1$

$\frac{4(5)}{2} + 1 = \frac{20}{2} + 1 = 10 + 1 = 11$

Proved By Construction:-

0	1	2	3	4
∈	G	GA	GAT	GATE
	A	AT	ATE	
	T	TE		
	E			

Total = 11

Proved.

→ Prefix - ω = abbc , Σ = {a, b, c}; find out the prefix

{a, ab, abb, abbc, ∈}

|ω| = 4

No of prefixes = |ω| + 1 = 4 + 1 = 5

→ Sufix - {c, bc, bbc, abbc, ∈}

No of sufixes = |ω| + 1 = 4 + 1 = 5

Lexographical Order Of Strings:-

→ Language - It is a set of strings defined over an alphabet

Ex → A is a language

A = {set of strings of length 2} over the alphabet Σ = {0, 1}

*) A = {00, 01, 10, 11}. Is it in lexographical order or dictionary order.

Yes, this is in lexographical order of string.

** As we have proved that language is a set which means that it should perform Union, Intersection, Difference, and Compliment. Yes, it performs all those operations. It also has Universal set.

Σ = {a, b}

Σ° = {set of strings of length 0} = ∈

Σ^1 = { set of strings of length 1 } = {a, b}.

Σ^2 = set of strings of length 2 } = {ab, aa, bb, ba}

Σ^k = set of strings of length k } = {w | |w| = k}

$|\Sigma^0| = 1$, $|\Sigma^1| = 2$, $|\Sigma^2| = 4$

2^0 2^1 2^2.

$|\Sigma^k| = 2^k$ ..... $|\Sigma^\infty| = 2^\infty$

Σ^* → of any length.

$\boxed{\Sigma^*} = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots \Sigma^\infty$

↓

Universal set.

$\bigcup_{k=0}^{\infty} \Sigma^*$ → kneale closure.

19

18.07.18

* Formal Defination of DFA :-

DFA is represented by :-

$(Q, \Sigma, \delta, q_0, F)$ → 5 tuple

Q – a finite set of states

Σ – a finite set of alphabets

δ – Transition function

q_0 – Initial state

F – A set of final state

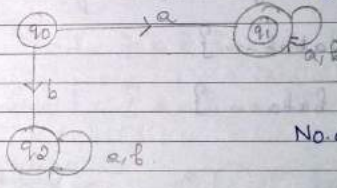
Q Construct a DFA which accepts a set of strings over $\Sigma = \{a, b\}$ starting with symbol 'a'.

Soln. $Q = \{q_0, q_1, q_2\}$ $\Sigma = \{a, b\}$ $q_0 = q_0/A$

$F = q_1$ $\delta = Q \times \Sigma$ $q_0 = a$

$q_1 = b$

$q_2 =$



a, aa, ab, abb, aba, aaa,

No. of state No. of alphabet

	a	b
q_0	q_1	q_2
q_1	q_1	q_1
q_2	q_2	q_2

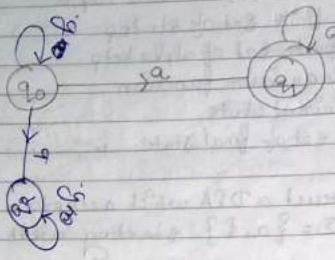
Q Construct a DFA which accepts a set of strings over $\Sigma = \{a, b\}$ containing the symbol a.

Soln. $L = \{a, aa, ab, ba, aba, bab \ldots\}$

(Not possible) $L = \{b, bb, bbb \ldots\}$

Strings

a
aa
ab
abb
b
ba
bb
bbabb



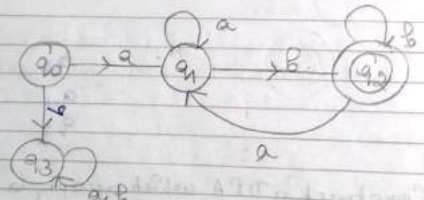
Q Construct a DFA which accepts a set of strings over $\Sigma = \{a, b\}$ starting with symbol 'a' entering with symbol of 'b'.

Soln. $L = \{ab, abb, aab, abab \ldots\}$

(Not possible) $L = \{ba, aba, baba \ldots\}$

Strings

ab
aba
abab
b
bb
ba



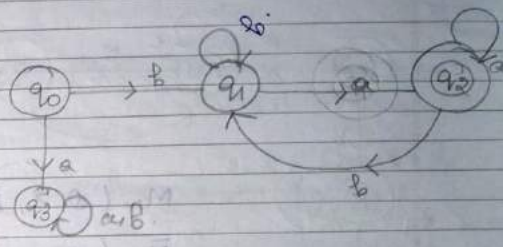
Q Starting with symbol 'b' and ending with 'a'

Soln. $L = \{ba, bba, baa \ldots\}$

(Not possible) $= \{ab, abb, aab \ldots\}$.

Strings

✓ ba
✓ baa
✗ baab
✓ baaba
✓ bba



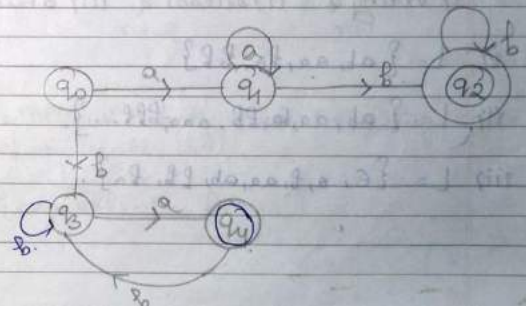
Q $\Sigma = \{a, b\}$ Starting with and ending with different symbols.

Soln. $L = \{ab, ba, aab, baba \ldots\}$

(Not possible) $L = \{a, b, aa, bb, aaa, bbb \ldots\}$

Strings

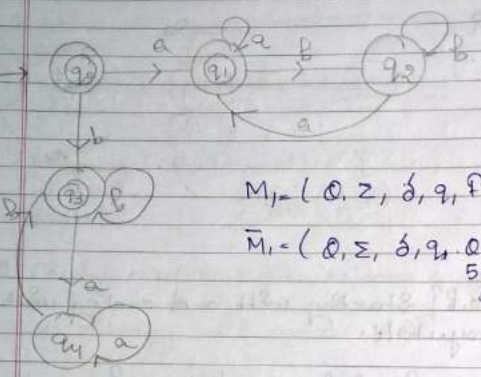
ab
ba
aab
baba
abab
abb



19.07.18

Q Construct a DFA which accepts set of strings starting and ending with same symbol over $\Sigma\{a,b\}$.
(This is compliment of last example of DFA)

Soln. $L = \{\epsilon, a, b, aa, bb, aaa, aba, bbb, bab \ldots\}$.



$M_1 = (Q, \Sigma, \delta, q_1, F)$

$\bar{M}_1 = (Q, \Sigma, \delta, q_1, Q-F)$

5-2

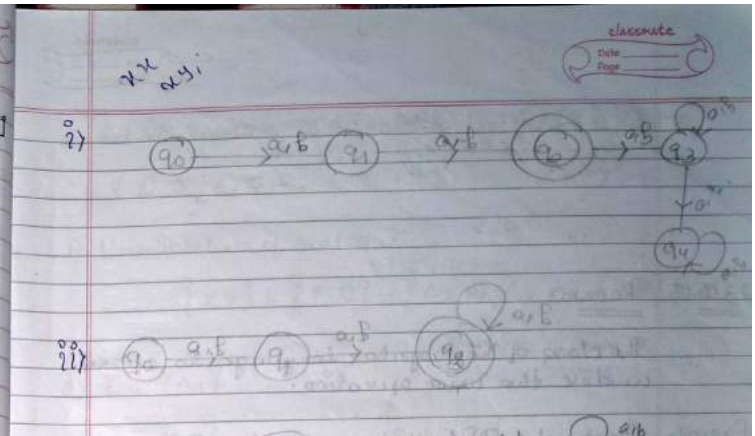
3.

Q Construct a DFA which accepts a set of strings over $\Sigma\{a,b\}$
i) exactly 2 ii) atleast 2 iii) atmost 2.

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

ii) $L = \{ab, aa, ba, bb, aaa, bbb \ldots\}$.

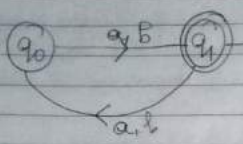
iii) $L = \{\epsilon, a, b, aa, ab, bb, ba\}$.



length (n)	Total no. of states	Final state
exactly n	n+2	1
atleast 2	n+1	1
atmost 2	n+2	n+1

Q Construct a DFA which accepts a set of strings over $\Sigma\{a,b\}$ of odd length.

Soln. $L = \{a, b, aaa, bbb, aba, bab \ldots\}$.

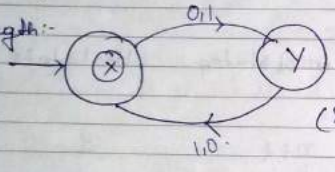


23.04.18 **Theorem:-**

The class of the regular languages is closed under the union operation.

Example - Construct a DFA which constructs a DFA of even length and ends with 1.

1. even length:-

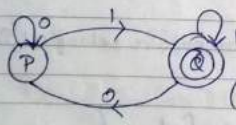


$(\{X,Y\}, \{0,1\}, \delta_1, X, X)$

δ_1-

	0	1
X	Y	Y
Y	X	X

2. ends with 1



$(\{P,Q\}, \{0,1\}, \delta_2, P, Q\}$

δ_2

	0	1
P	P	Q
Q	P	Q

Let the resulting DFA be $DFA_1 \cap DFA_2$

$(Q, \Sigma, \delta, q, F)$

1) $Q \rightarrow$ Total no of states.

$\{X,Y\} \times \{P,Q\} = \{(X,P), (X,Q), (Y,P), (Y,Q)\}$

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

3) $\delta = \{0,1\}$

$\delta((X,P),0) = \delta((X,0),(P,0))$
$= \delta(Y,P)$

$\delta((X,P),1) = \delta((X,1),(P,1))$
$= \delta(Y,Q)$

$\delta((X,Q),0) = \delta((X,0)(Q,0))$
$= \delta(Y,P)$

$\delta((X,Q),1) = \delta((X,1),(Q,1))$
$= \delta(Y,Q)$

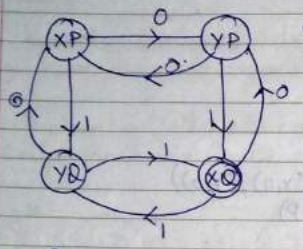
$\delta((Y,P),0) = \delta((Y,0),(P,0))$
$= \delta(X,P)$

$\delta((Y,P),1) = \delta((Y,1),(P,1))$
$= \delta(X,Q)$

$\delta((Y,Q),0) = \delta((Y,0)(Q,0)) = \delta(X,P)$

$\delta((Y,Q),1) = \delta((Y,1),(Q,1)) = \delta(X,Q)$

Now drawing the DFA



By seeing both DFA we will see that X and Q combining we will get QXQ as final state in final DFA.

$L = \{11, 1111, 1011, 0011 \ldots\}$ → being satisfied

(Not Poss) $L = \{1, 0, 00, 00110 \ldots\}$.

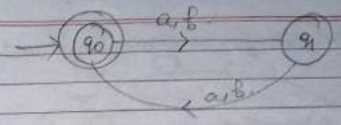
Pg-83 1.5 to 1.6

Q Construct a DFA which accepts a string $\{w\}$ w has even length and an odd no of a's? $\Sigma \{a, b\}$.

Sol. I. Even length.

$L = \{aa, bb, ab, ba, aaaa, bbbb \ldots\}$.

(X) $L = \{a, b, aaa, bbb, aba, \ldots\}$.



$(\{q_0, q_1\}, \{a, b\}, \delta, q_0, q_1\}$.

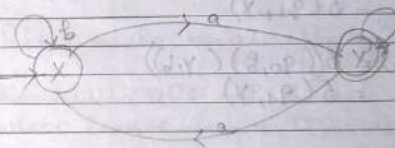
$\delta_1 =$

	a	b
q_0	q_1	q_1
q_1	q_0	q_0

II. An odd no of a's.

$L = \{a, ab, ba, aaa, abb \ldots\}$

$xL = \{aa, bb, aba \ldots\}$.



$(\{X, Y\}, \{a, b\}, \delta_2, X, Y\}$.

$\delta_2 =$

	a	b
X	Y	X
Y	X	Y

Let the resulting DFA be DFA_1 and DFA_2

$(Q, \Sigma, \delta, q, F)$

1) $Q \rightarrow$ Final states of states.

$\{q_0 q_1\} \times \{X, Y\} = \{\{q_0, X\}\ \{q_0, Y\}\ \{q_1, X\}\ \{q_1, Y\}\}$

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

3. $\delta = \{a, b\}$

$\delta((q_0, X), a) = \delta((q_0, a)(q_1, 0))$

$= \delta(q_1, Y)$

$\delta((q_0, Y), b) = \delta((q_0, b)(X, b))$

$= \delta(q_1, X)$

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

$= \delta(q_1, X)$

$\delta((q_0, Y), b) = \delta((q_0, b)(Y, b))$

$= \delta(q_1, Y)$

$\delta((q_1, X), a) = \delta((q_1, a)(X, a))$

$= \delta(q_0, Y)$

$\delta((q_1, X), b) = \delta((q_1, b)(X, b))$

$= \delta(q_0, X)$

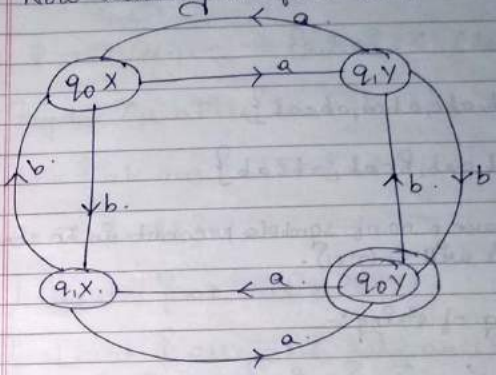
$\delta((q_1, Y), b) = \delta((q_1, b)(Y, b))$

$= \delta(q_0, X)$

$\delta((q_1, Y), b) = \delta((q_1, b)(Y, b))$

$= \delta(q_0, Y)$

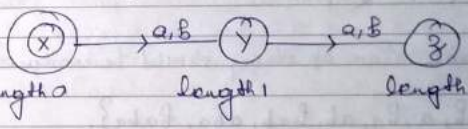
Now drawing the final DFA:-



FL = {ab, ba, aaab, abbb ....}. → Matching.

Q Construct a DFA which accepts a set of strings whose length is a multiple of 3?

Soln



25.07.18

* Strings:-

Ex- 'abcab' $\Sigma = \{a, b, c\}$

Prefix → {ϵ, a, ab, abc, abca, abcab}

Sufx → {ϵ, b, ab, cab, bcab, abcab}

If there are n no. of symbols present in the string the prefix will be (n+1).

* Substring of string:-

Ex- 'abc' $\Sigma = \{a, b, c\}$

Substring → {ϵ, a, b, c, ab, bc, abc}.

$$n \rightarrow \frac{n(n+1)}{2} + 1$$

Ex- baba (no. of string should be unique)

{ϵ, b, a, ba, ab, bab, aba, baba}.

* Two types of substring:-

i) Trival → string itself – {ϵ, abc}

ii) Non-trival – $\left(\frac{n(n+1)}{2} + 1\right) - 2$

'abc' → $\left(\frac{3(3+1)}{2} + 1\right) - 2 = 5$.

* ϵ → string of length 0.

* ϕ → empty.

* Regular Operations :-

1. $L_1 = \{ab, aa\}$ $\Sigma = \{a, b\}$

2. $L_2 = \{ba, bb\}$

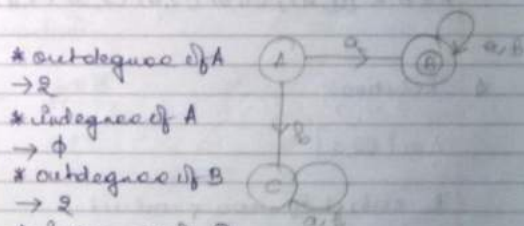
$L_1 \cup L_2$ → {ab, aa, ba, bb}.

$L_1 \circ L_2$ → {abba, abbb, aaba, aabb}
(concatination)

$L_1 \times L_2$ → {(ab, ba), (ab, bb), (aa, ba), (aa, bb)}

L_1^* → {ab, aa, ϵ, abab, aaaa, ababab, aaaa ... } infinite.
(star)

* Concatination is not applicable in DFA, only in NFA



* outdegree of A → 2
* indegree of A → ϕ
* outdegree of B → 2
* indegree of B → 3
* outdegree of C → 2
* indegree of C → 3

* Outdegree will always be 2

* Relational function :-

n	f(n) / n²
1	1
2	4
3	9

Domain = {1, 2, 3} → input given.
Range = {1, 4, 9}

* Relation :-

A = {1, 2, 3} C = {1, 2, 3} D = {2, 3}
B = {4, 5} C will be subset not proper subset of A.
C ⊆ A
D ⊂ A proper subset.

* Relation of A to B → Subset of given.

A × B = {(1, 4), (1, 5) (2, 4) (2, 5) (3, 4) (3, 5)}

* Relation :-

A = {1, 2, 3}

R subset of cross product.

i) Reflexive = R{(1, 1) (2, 2) (3, 3)}

ii) Symmetric = R { (1, 2) (2, 1) (1, 3) (3, 1) (2, 3) (3, 2) }.

iii) Transitive = R { (1, 2), (2, 3), (1, 3) }
(if (1, 3) is not present then transitive)

* Boolean Algebra :-

1) Not / Negation

0	1
1	0

2) AND (Conjunction)

A	B	A ∧ B
0	0	0
0	1	0
1	0	0
1	1	1

3) OR (Disjunction)

A	B	A ∨ B
0	0	0
0	1	1
1	0	1
1	1	1

4) Exclusive OR (A ⊕ B)

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

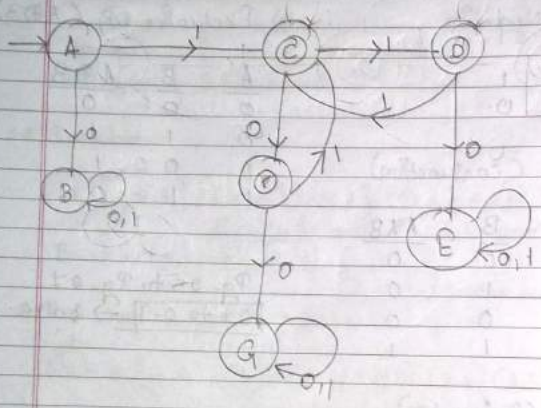
Pg- 25 to Pg- 27
0.1 to 0.9 → Solve.

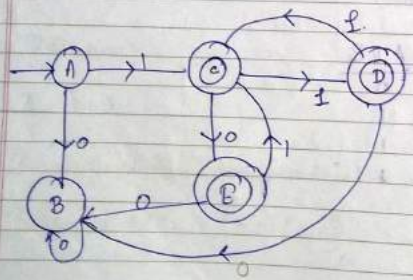
(1.6) Q Construct a DFA for $\Sigma\{0,1\}$.

i) where every odd position of w is 1

Soln: $L = \{1, 000, 101, 111, 1111, 11, 1010 \ldots\}$

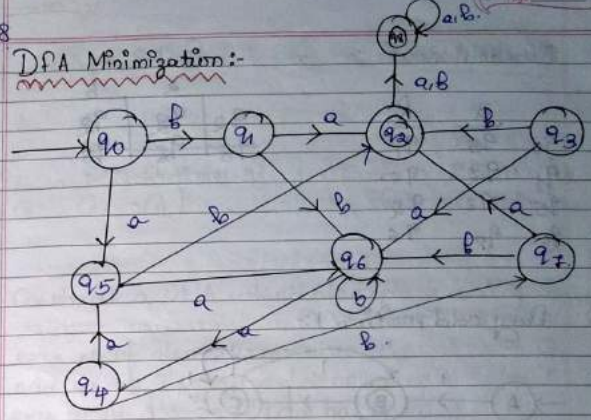
(x) $L = \{0, 00, 000, 0000, \epsilon, 010, 0101, 01010 \ldots\}$.





26.07.18

* DFA Minimization:-



* Transition table:-

Strings	a	b			
$\rightarrow q_0$	q_5	q_1	q_0	q_5	q_1
$*q_1$	q_2	q_6	q_1	q_2	q_6
$*q_2$	q_8	q_8	q_5	q_6	q_2
q_3	q_6	q_2	q_6	q_0	q_6
$\rightarrow q_4$	q_5	q_1			
q_5	q_6	q_2			
q_6	q_4 q_0	q_6			
$*q_7$	q_2	q_6			
$*q_8$	q_8	q_8			

Dead state is the non-final state from which we can never reach final state using any kind of symbol. An initial state can never be a dead state

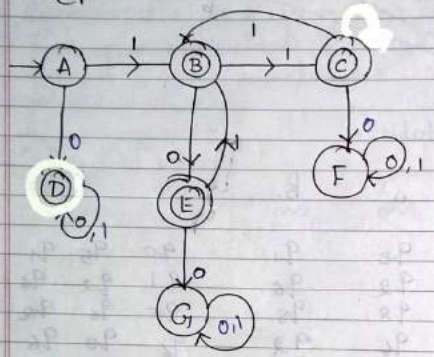
Non-final State :-

	a	b
q_0	q_5	q_1
q_1	q_2	q_6
q_5	q_6	q_2
q_6	q_0	q_6

final Set :-

	a	b
q_2	q_2	q_2
q_8	q_2	q_8

Q Every odd position 1 :-



* Transition table :-

	0	1
A	D	B
B	E	C
C	F	B
D	D	D
E	G	B
F	F	F
G	G	G

Non-final State

A	D	B
D	D	D
F	F	F
G	G	G

delete

final State

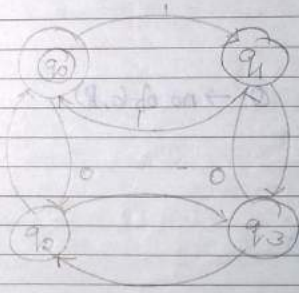
	C
B	E C
C	D B
E	D B

Q Construct a DFA which accepts a string $\Sigma\{0,1\}$

i) even no. of 0's and even no of 1's
ii) odd no. of 0's and odd no. of 1's
iii) even no. of 0's and odd no. of 1's
iv) odd no. of 0's and even no. of 1's.

i) Even no of 0's and even no of 1's.

$L = \{\epsilon, 00, 11, 0011, 1100, 0101, 0000111, \ldots\}$. (x) $L = \{0000, 001, 0000 11110 \ldots\}$.



28/04/18

Non-deterministic finite automata (NFA)

* Formal description of a NFA:-

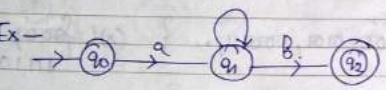
A non-deterministic finite automation is a 5 tuple $(Q, \Sigma, \delta, q_0, F)$, where:-

$Q \rightarrow$ A finite set of states.
$\Sigma \rightarrow$ A finite alphabet.
$\delta \rightarrow Q \times \Sigma_\epsilon \rightarrow P(Q)$ is transition function
$q_0 \rightarrow \in Q$ is the start state.
$F \rightarrow \subseteq Q$ is the set of accept states.

* In DFA :-

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

Ex-



* In NFA:-

$\delta : Q \times \Sigma \rightarrow 2^Q$ $\quad Q \rightarrow$ no of (a, b)

$2^2 = 4 \Rightarrow \phi$, q_0, q_1, $\{q_0, q_1\}$

Ex- $Q = \{q_0, q_1, q_2\}$
$\Sigma = \{a, b\}$
$q = \{q_0\}$
$F = \{q_2\}$

$\phi \rightarrow$ empty state

* $2^3 = 8$
$\{\phi, q_0, q_1, q_2, (q_0, q_1), (q_1, q_2), (q_0, q_2), (q_0, q_1, q_2)\}$

Q Construct a NFA for a set of strings starting with symbol 'a' over $\Sigma = \{a, b\}$

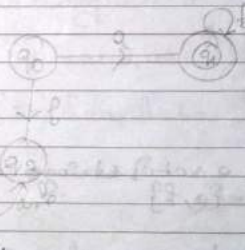
* DFA → It considers both accepted and rejected string.

* NFA → It considers only accepted string.

* DFA:-

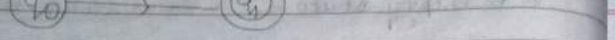
$L = \{a, ab, abb, aba, aab, abba, abab \ldots\}$

x $L = \{b, ba, bab, baa \ldots\}$.



* NFA :-

$L = \{a, ab, abb, aba, aab, abab, abbb, abba \ldots\}$



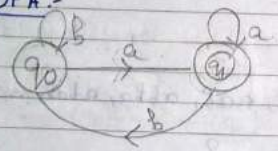
	a	b
q_0	q_1	ϕ
q_1	q_1	q_1

Q Construct a NFA for a set of strings ending with symbol 'a'.

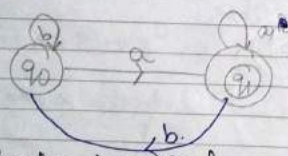
L = {ba, bba, aba, aaa, baa, baba, ...}

✗ L = {ab, abb, abab, abbb, ababb ...}

DFA :-



NFA :-

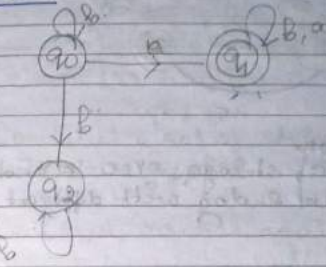


Q Construct a NFA for a set of strings contains 'a' over symbol $\Sigma = \{a, b\}$

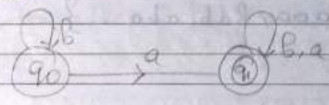
L = {a, ab, ba, abb, aba, aaa, abaa, aaaa ....}

(✗) L = {b, bb, bbb ...}.

DFA :-



NFA :-



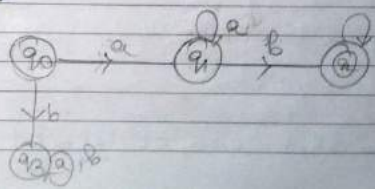
* Every NFA is not DFA but Every DFA is not NFA

Q Construct a NFA for a set of strings starting with 'a' and ending with 'b'

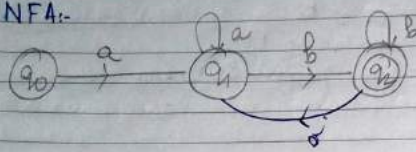
L = {ab, aab, abb, abbb, abab ...}

✗ L = {ba, baa, baba.., bbba ...}

DFA :-



NFA:-

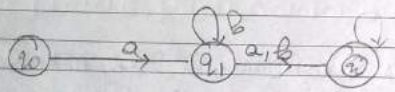


Q Construct a DFA for set of strings over 'a' and 'b' starting and ending with different symbol.

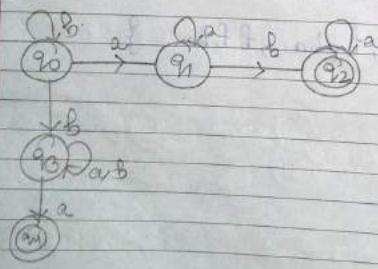
L = { ab, ba, abb, baa, aab, bba ... }

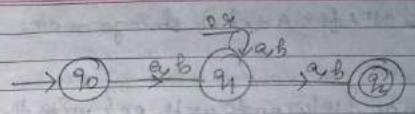
x L = { aa, bb, aaa, bab, aba ... }.

DFA:-



NFA:-

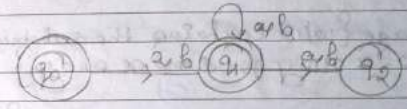


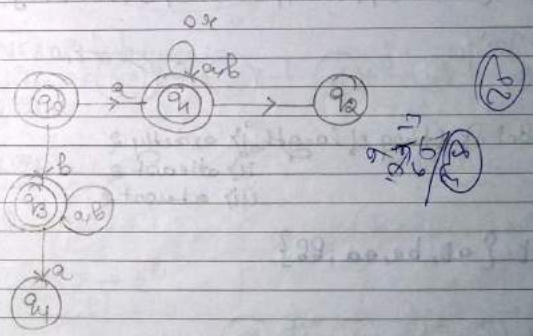


Q Construct a DFA for set of strings starting and ending with same symbol.

L = { ∈, a, aa, bab, aaa, ... }.

x L = { ba, ab, abb, baa, baaa, abbb ... }.

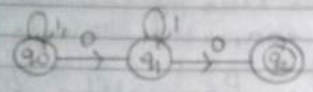




Q 1.7 Construct a NFA for a set of strings over $\Sigma = \{0,1\}$.

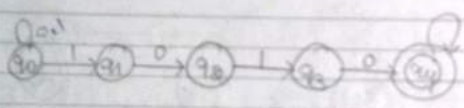
a) The language $\{w|w$ ends with $00\}$ with three states.

$L = \{00, 100, 1100, 0100 \ldots\}$.



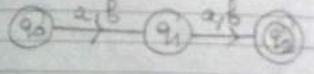
b) The language $\{w|w$ contains the substring 0101 (i.e. $w = x0101y$ for some x and $y\}$

$L = \{10101, 1010, 0101, 00101, 01010 \ldots\}$



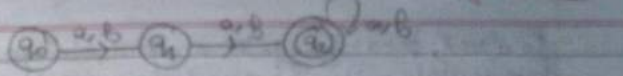
c) Set of string of length i) exactly 2
ii) atleast 2
iii) atmost 2.

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



ii) atleast 2 → ≥ 2.

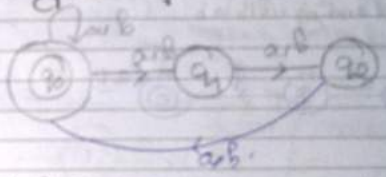
$L = \{ab, aa, bb, ba, aaa, bbb, aba, bab \ldots\}$



iii) atmost 2 ≤ 2

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

Q String of length multiple of 3 $\Sigma \{a,b\}$



30.07.18

NFA is of 2 types:-
i) Without ∈
ii) With ∈

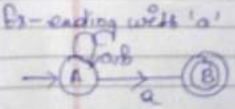
Without ∈

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

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

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

Ex- ending with 'a'



With ∈

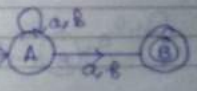
$(Q, \Sigma, \delta, q_0, F)$

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

$\delta: Q \times \Sigma_\epsilon \to 2^Q$ or

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

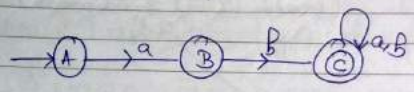
Ex -



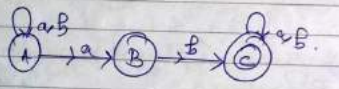
In a NFA, for a language the accepted strings can have multiple paths. The accepted strings can go to the non-final state (single circle) and final state (double circle) as the accepted strings should go to final states so we consider the path which lead to the double circle and ignore the path which lead to single circle. For rejected strings of a particular language in a NFA, string should go to the non-final state (single circle), then it should'll have the option to go to the final circle in any move.

Example

Q Starting and ending with ab.

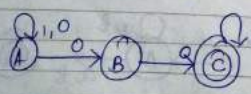


Q Containing ab

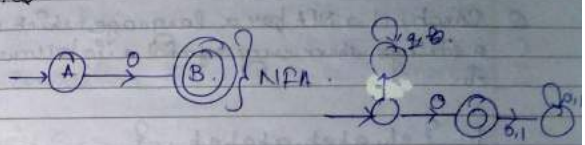


Q The language which contains the string ends with 00 with 3 states.

L = { 00, 100, 000, 1100, 1000, 0100 }

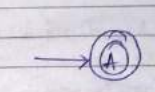


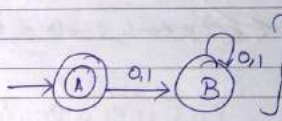
Q Language which contains {0} :- Σ {0,1} which has 2 states



Q Draw a NFA for a language which contains a string {ε} with only one state.

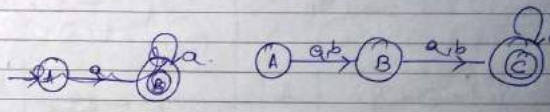
Soln.





Q Draw a NFA for language which contains the strings which consist of consecutive 'a' consecutive 'b' over Σ {a,b}.

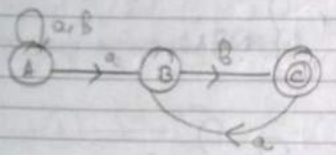
L = { aa, bb, aaa, bbb ... }.



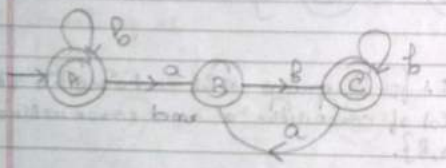
Q Construct a NFA for a language which contains a strings where every symbol a is followed by b.

L = { ab, abab, ababab... }

L = { ∈, a, b, aa, bb, aba... }



NFA



01.02.19

NFA with ∈ :-

(Q, Σ, δ, q, F)

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

NFA with ∈ :-

(Q, Σ, δ, q, F)

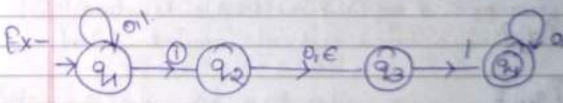
Q → A finite set of state
Σ → input alphabet → A finite set of symbols
δ → Transaction function

$\delta : Q \times \Sigma_\epsilon \to 2^Q$

or

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

Ex-



$Q \times \Sigma \to 2^Q$

A, B	a, b	ϕ, A, B, AB

$Q \times \Sigma_\epsilon \to 2^Q$

A, B	a, b → symbol; c → string	ϕ, A, B, AB

Q1) Formal Description:-

$(Q, \Sigma, \delta, q, F)$

$Q \rightarrow \{q_1, q_2, q_3, q_4\}$

$\Sigma \rightarrow \{0,1\}$ → should not be considered

δ:	0	1	ϵ
q_1	q_1	$\{q_1, q_2\}$	ϕ
q_2	q_2	ϕ	q_3
q_3	ϕ	q_4	ϕ
q_4	q_4	q_4	ϕ

Q2) Find out the ϵ-closure of each state of NFA

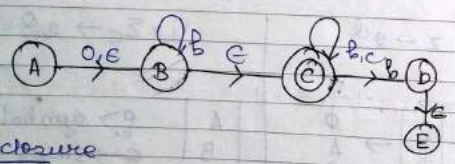
Soln. $q_1 = \{q_1\}$ One will be the state itself

$q_2 = \{q_2, q_3\}$

$q_3 = \{q_3\}$

$q_4 = \{q_4\}$

Ex-



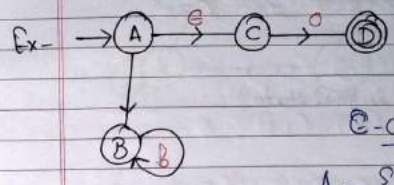
* ϵ-closure

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

$B = \{B, C\}$

$C = \{C\}$

Ex-



ϵ-Closure of

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

$B = \{B\}$

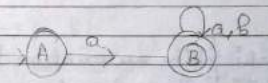
$C = \{C\}$

$D = \{D\}$.

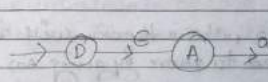
Q Construct a NFA with ϵ for a language which contains strings starting with symbol 'a'.

Soln. $L = \{a, aa, ab, aab, aba, aaa \ldots\}$

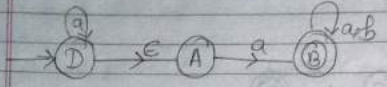
Case-1 When initial state is not the final state



Case-2

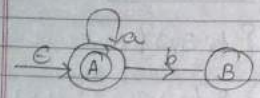


OR

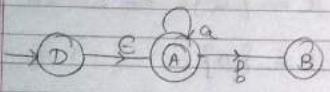


Case II When initial state is final state

Soln.



OR



* Conversion of NFA to DFA OR.

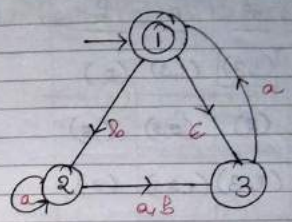
Equivalence of NFA and DFA :-

Theorm :-

Every non-deterministic finite automation has an equivalent deterministic finite automata.

(Question will be given, where theorm will be given and will be asked to prove by giving an example).

Ex- Q 1.41)



δ	a	b	ϵ
1	ϕ	2	3
2	(2,3)	3	ϕ
3	1	ϕ	ϕ

ϵ-closure

1 = {1,3}
2 = {2}
3 = {3}.

let the NFA $(Q, \Sigma, \delta, q, F)$
let the DFA $(Q_1, \Sigma, \delta_1, q_1, F)$

⇒ $Q_1 \rightarrow 2^Q$ [Q → no. of state present in DFA]
$= 2^3$
$= 8 = \{\phi, 1, 2, 3, (1,2), (2,3), (3,1), (1,2,3)\}$

⇒ $\Sigma \rightarrow \{a, b\}$

→ If the initial state of NFA does not contain ϵ in outdegree then initial state of DFA will be the same.

→ Containing ϵ in outdegree then the i/s of DFA will be ϵ-closure of initial state of NFA.

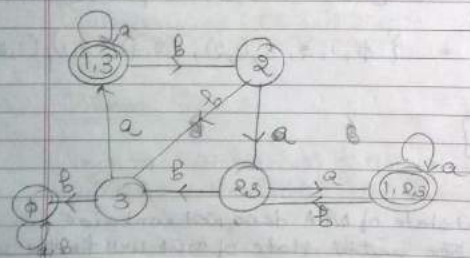
Normal

δ	a	b
ϕ	ϕ	ϕ
1	ϕ	2
2	(2,3)	3
3	(1,3)	ϕ
(1,2)	(2,3)	(2,3)
(2,3)	(1,2,3)	(3)
(1,3)	(1,3)	(2)
(1,2,3)	(1,2,3)	(2,3)

Minimization

δ	a	b
[1,3]	(1,3)	(2)
(2)	(2,3)	(3)
(2,3)	(1,2,3)	(3)
(1,3)	(1,3)	ϕ
[1,2,3]	(1,2,3)	(2,3)
ϕ	ϕ	ϕ

Now by seeing minimization table, we can draw the DFA

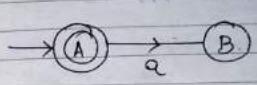


This is the DFA

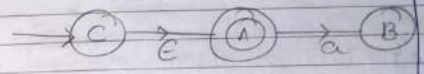
02.08.18

Q Add Epsilon (ϵ) in the NFA

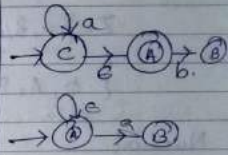
1)



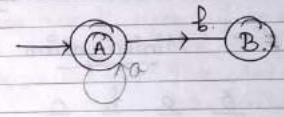
Soln



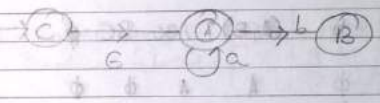
This is wrong



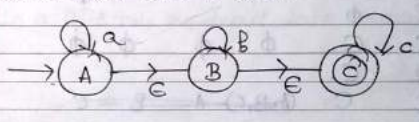
2)



Soln.



Q2) Convert the NFA to DFA:-



δ	a	b	c	ϵ
A	A	ϕ	ϕ	B
B	ϕ	B	ϕ	C
C	ϕ	ϕ	C	ϕ

ϵ-closure

A = {A, B, C}

B = {B, C}

C = {C}.

δ	a
X	XX
XX	XX

Let the NFA $(Q, \Sigma, \delta, q, F)$
Let the DFA $(Q_1, \Sigma, \delta, q_1, F)$

$Q_1 = 2^Q$
$= 2^3$
$= 8$

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

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

Normal

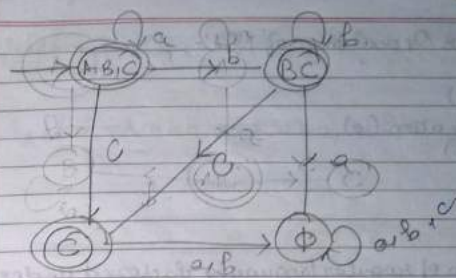
δ	a	b	c
ϕ	ϕ	ϕ	ϕ
A	ABC	ϕ	ϕ
B	ϕ	BC	ϕ
C	ϕ	ϕ	C
(A,B)	ABC	BC	ϕ
(B,C)*	ϕ	BC	C
(C,A)*	ABC	ϕ	C
(A,B,C)*	ABC	BC	C

Minimization

δ	a	b	c
ϕ	ϕ	ϕ	ϕ
A	ABC	BC	C
B	ϕ	BC	ϕ
C	ϕ	ϕ	C
(A,B)	ABC	BC	C
(B,C)*	ϕ	BC	C
(C,A)*	ABC	ϕ	C
(A,B,C)*	ABC	BC	C

Now by seeing minimization table, we can draw

Let $A, B, C = q_0$
$BC = q_1$
$C \to q_2$
$\phi \to q_3$.



* There are 3 types of unrequired states in DFA :

1. Unreachable – The state to which we cannot reach from initial state
2. Duplicate State –

δ	a	b
X	B	C
X	B	C

Duplicate state ←

3. Dead State – Always ends in self loop

* In a DFA only one dead state required.

04.03.18

Regular Operations of NFA:-

1) Union (U)
2) Concatenation (o)
3) Star (*)

Theorem:-

The class of regular languages is closed under union operation.

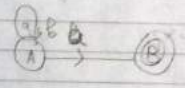
* Union of 2 NFA :-

Ex- Construct a NFA for a language which contains string
$\Sigma = \{a, b\}$

i) {w/w has ending with a or starting with a}

$NFA_1 \rightarrow (Q_1, \Sigma, \delta_1, q, F)$

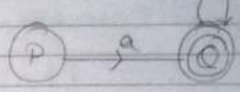
$L_1 = \{a, ba, baa, bba, aaa, aba \ldots\}$



$(\{A, B\}, \{a, b\}, \delta, A, B)$

$NFA_2 \quad (Q_2, \Sigma, \delta_2, q, F)$

$L_2 = \{a, ab, aaa, aab, aba, abb, \ldots\}$

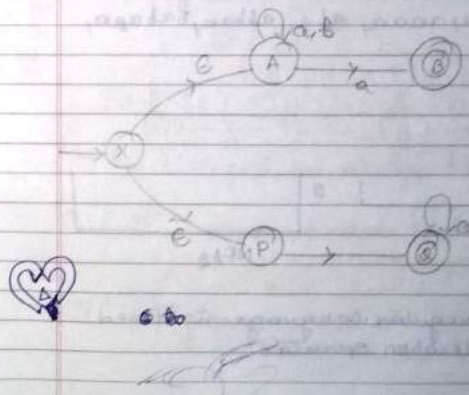


$(\{P, Q\}, \{a, b\}, \delta_2, P, Q)$

$L = L_1 \cup L_2$

$\{a, aa, aaa, ba, ab, bba, aba, abb, aab \ldots\}$

$NFA = NFA_1 \cup NFA_2$



Formal Description :

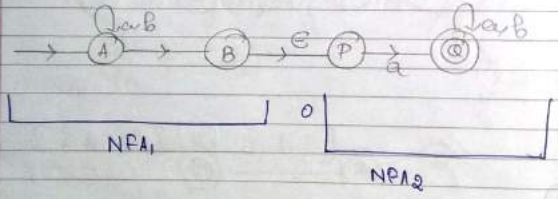
* Concatination of 2 NFA :-

$L_1 \circ L_2$

$NFA = NFA_1 \circ NFA_2$

$L = \{aa, aaa, aaaa, aba, abbaa, babaaa, \}$

NFA →



The class of regular languages is closed under concatenation operation

* Star operation :-

— x —

☆ Kleene closure (kleene star / kleene operation) $[\Sigma^*]$

It is an unary operation on a set of symbols (Σ) that gives the infinite set of all possible strings of all possible lengths over (Σ) sigma including (ϵ) epsilon.

Ex → $\Sigma = \{a, b\}$.

$* = 0, 1, 2, 3, \ldots, n$

length → $\Sigma^* \Rightarrow \Sigma^0 = \{\epsilon\}$ | $a^0 b^0$

$\Sigma^1 = \{a, b\}$ | $a^1 b^1$

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

$\Sigma^3 = \{aaa, bbb, aba, aab, abb, bab, bba, baa\}$

⋮

$\Sigma^* = \{\Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \ldots \cup \Sigma\}$.

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

↓

This is in proper order, shortest order, string order

$\Sigma^* = \{\epsilon, a, b, aa, ab, ba, bb, aaa, bbb, \ldots\}$ → lexographic order

* Lexographic order – It is same as the dictionary order, but the

* Shortlex order – It is modified lexographic order is identical to lexographic order except that shorter strings proceed longer strings.

* Kleene Plus (Σ^+)

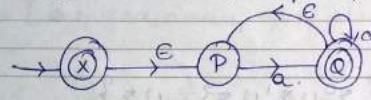
$\Sigma^+ = \Sigma^* - \{\epsilon\}$

The set of kleene plus is the infinite set of all possible strings of all possible lengths over Σ excluding the string $\{\epsilon\}$ epsilon.

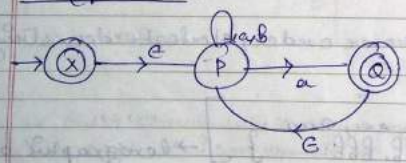
—x—

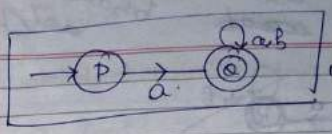
Star → Let the NFA starting with a

$L_2^* = \{\epsilon, a, aa, aaa, aba, abaaba, \cdots\}$



ending with a





without state, starting with a.

Q Construct a NFA for a language.

{w | w has an a's multiple of 3 (or) each a is followed by atleast one b}.

2) Apply star operation on NFA for a language which contains strings starting and ending with different symbol.

Q3)

δ	a	b	c
A*	E	F	E
B	B	B	B
C	B	A	C
D	A	A	G
E*	I	B	H
F	F	F	F
G	E	F	G
H	I	B	H
I	E	F	G

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

Q1) What are the states required for a minimal DFA?

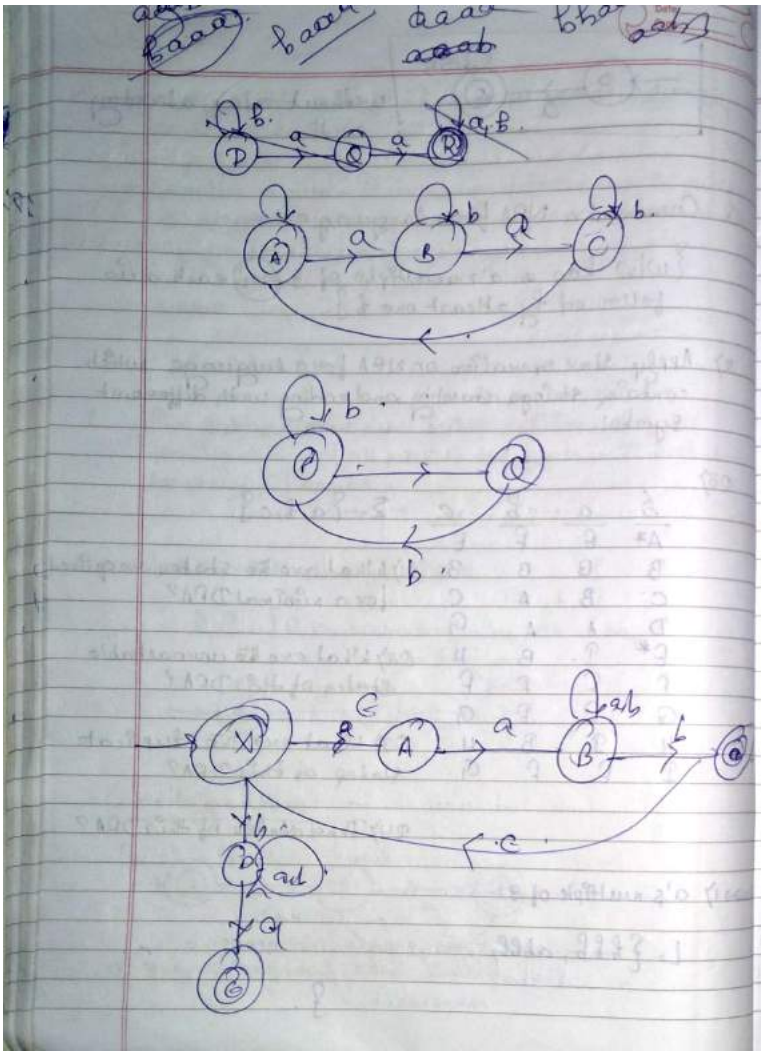
Q2) What are the unreachable states of this DFA?

Q3) What are the duplicate states of this DFA?

Q4) Dead states of this DFA?

Ans → 1) a's multiple of 3 :-

L = {bbb, abbb, aaa, aaaaaa, aaaaaaaaa, aaab, baaa, baaaaaa, aaaaaab, aaaaaaaaab, b}.



06.08.18

Introduction To Regular Expression :-

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

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

$\Sigma^* = a^* b^* = \{\epsilon, a, aa, aaa \ldots\ b, bb, bbb \ldots\}$

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

$b^* = \{\epsilon, b, bb, bbb, bbbb \ldots\}$

Regular (Grammar) → Regular (Language) → Finite (Automata) (NFA or DFA)

(Regular expression)

Regular expression is used to express a regular language

Rule 1: Any symbol belonging to the alphabet (Σ) is a regular expression.

If $\Sigma = \{a, b\}$, then a is a regular
b is a regular

Rule 2: The null string (ϵ) which is a string of length 0 is a regular

Rule 3: The empty string ϕ is a regular expression

Rule 4: If R is a regular expression then (R) is also a regular expression

Rule 5: If R_1 and R_2 are regular expression then $R_1 \cup R_2$ is also a regular expression.

Ex - If $\Sigma = \{a, b\}$ if a is a regular expression then b is a regular expression acc to rule 1 then
a ∪ b is a RE
a * b is a RE

Rule 6: If R_1 and R_2 are regular expressions then $R_1 \circ R_2$ is also a

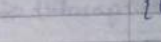
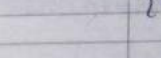
a → RE ab → RE.
b → RE
a ∘ b → RE

Rule 7: If R_1 is a regular expression then R_1^* is also a RE.

Ex → $\Sigma = \{a, b\}$
a → RE
a* → RE.

* In RE * (star operation) is done first by default followed by concatenation and then finally Union unless parenthesis change the usual order.

** Regular Expression | Regular lang. & Corr. Regular set

Regular Expression	Regular lang. & Corr. Regular set
0	{0} lang. contains string only 0 
1	{1} Lang. contain string only 1 
Union: 0+1 or; 0∪1 or; (0∪1) or; ({0} ∪ {1}) or; Σ	{0,1} → lang. containing string either 0 or 1 over the alphabet 0,1
star operation: 0* or; {0}*	{ϵ, 0, 00, 000 ...} Lang containing any no. of zero over Σ = {0,1}.
1* or; {1}*	{ϵ, 1, 11, 111 ...} Lang containing any no of one over Σ = {0,1}
(0∪1)*; Σ*; (Kleene closure) or; (0+1)*	{ϵ, 0, 1, 00, 01, 10, 11, 000, 111 ...} Lang containing all possible strings of any length over Σ = {0,1}.
Concatenation: Σ*1	{1, 01, 11, 001, 011, 101 ...} Lang containing all possible strings which ends with 1 over Σ = {0,1}.

$1\Sigma^*$	{10, 11, 100, 101, 110 ...} Lang. containing all possible strings which starts with 1.
$0\Sigma^*$	{0, 00, 01, 000, 001 ...}
Σ^*0	{0, 00, 10, 000, 010 ...}
$0\Sigma^* \cup \Sigma^*1 \rightarrow$	{01, 001, 010, 000, 011 ... ∪ 1, 01, 11, 001, 021 ...} Lang containing all possible string either starts with 0 or ends with 1
10^*	{1, 10, 100, 1000 ...} (Lang. containing starting with 1 and followed by any no of zeros)
0^*1	{01, 001, 0001, 00001, ...} Lang. containing any no of zeros but must end with one
1^*0	lang containing any no of 1's but must end with a single zero
01^*	

$(0 \cup 1)0^*$	{0, 1}. Lang containing that starts with either 0 or 1 but must follow by any no of zeros. (OR) {0, 00, 000, 0000 ... 1, 10, 100, 1000 ...}

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

i) $a + b + c$
ii) $ab + ba$
iii) $ab(d + c)$
iv) $(aa)^*$
v) $(a + b)c^*$
vi) $(abc)^*$
vii) $(a + b)c$

i) $a + b + c \rightarrow$ Lang. containing either a or b or c over the alphabet $\Sigma = \{a, b, c, d\}$.

{a, aa, aaa, aaaa or
b, bb, bbb, bbbb or
c, cc, ccc, cccc ...}

ii) $ab + ba \rightarrow$ Lang. containing starts with a and ends with b

and or

Lang containing starts with b and ends with a.

ab rahega or
(ba) rahega

{ab, abb, aab, abbb.. }

or

{ba, baa, bba, baaa.. }.

{ab, ba, abb, baa, aab, bba, abbb, baaa.. }

iii) ab(d+c)

d+c — Lang containing either d or c.

08.08.18

R	L(R)
$(a+b)c(d+e)$	$\rightarrow$ {acd, bce, ace, bcd}
$(a+b)(c+d+e)x$	$\rightarrow$ {acx, adx, aex, bcx, bdx, bex}
abc^*de	$\rightarrow$ {abcde, abccde, abcccde...}
$(0+11)2(33+1)$	$\rightarrow$ {0233, 11233, 021, 1121}
$(ab(c+d)e^*$	$\rightarrow$ {abce, abcee, abceee... abde, abdee, abdeee..}

Examples of Books:-

$0^* 1 0^*$ → lang. contains strings which contains a single one (1) {1, 010, 01, 10..}

$\Sigma^* 1 \Sigma^*$ → lang contain strings which consist of a single one atleast. {01101, 10101}

$\Sigma^* 001 \Sigma^*$ → lang contain string which consist of substrings as (001). {01, 00110, 10001...

$1^*(01^*)^*$ → lang contain strings where every zero is followed by a single 1.

$(\Sigma\Sigma)^*$ → lang contains strings having even length.
{ϵ, 00, 11, 01, 1000, 111, 000111..}

$(\Sigma\Sigma\Sigma)^*$ → lang contains strings having length multiple of 3.
{ϵ, 000, 111, 011, 000111, 000111010, 000111000010}

$(01 \cup 10)$ → {01, 10} Lang. contains strings either 01 or 10.
(or)

$0\Sigma^*1 \cup 1\Sigma^*1 \cup 0 \cup 1$ → Lang. contains strings starting and ending with diff. symbol

$(0 \cup \epsilon)1^*$ → {01^*, $\epsilon 1^*$}
{01, 011, 0111, 01111...
ϵ1, ϵ11, ϵ111, ϵ1111...}.

$(0 \cup \epsilon)(1 \cup \epsilon)$ →

= {01, 0ϵ, ϵ1, $\epsilon\epsilon$}.

= {01, 0, 1, ϵ} Lang. contains strings either 01 or 0 or 1 or ϵ.

Scanned by CamScanner

* Identities of Regular Expression

1. Adding empty lang. to any other lang. will not change the original lang.

$R \cup \phi = \phi \cup R = R + \phi = \phi + R = R$.

2. $R \circ \epsilon = R\epsilon = \epsilon R = R$

Joining or concatenating the empty string (ϵ) with any string will not change the lang.

3. $R \cup \epsilon \neq R$.

Ex - $R = 0$ or 1 $\quad \Sigma = \{0,1\}$.

$R \cup \epsilon = \{0\} \cup \epsilon$

$= \{0, \epsilon\}$.

4. $R \circ \phi$ or $\phi \circ R = R\phi = \phi R = \phi$

Concatenating the empty set / empty lang (ϕ) results the empty set.

5. $(\phi)^* = (\phi^*) = \epsilon$

If the lang. is empty the star operation on that empty language only gives empty string (ϵ)

6. $\epsilon + RR^*$

$= \epsilon + R^+$

$= R^*$.

Ex - $RR^* \quad R = 0 \quad \Sigma = \{0,1\}$.

$R^* = \{\epsilon, 0, 00, 000, 0000, 00000 \ldots\}$

$RR^* = 0R^*$.

$= \{0\epsilon, 00, 000, 0000 \ldots\}$

$= \{0, 00, 000, 0000 \ldots\}$

$= R^+$. (Kleene Plus) (without ϵ)

7. $R + \epsilon = \epsilon + R$.

8. $(P + Q)R = PR + QR$.

9. $(PQ)^* P = P(QP)^*$

10. $RR^* = R^* R$.

* Application of Regular Expression:-

1) They are useful tools to design a compiler for programming language.

Q Draw the Regular Expression for the language

1. $L = \{aa, aaaa, aaaaaa \dots\}$

$= (aa)^+$ (then it will progress in even no)

$\Rightarrow L = aa\{\epsilon, aa, aaaa \dots\}$

$= aa\,(aa)^*$ [R R*]

$= (R^+)\,(aa)^+$.

2. $L = \{00, 0011, 001111 \dots\}$

$= 00\{\epsilon, 11, 1111 \dots\}$

$= 00(11)^*$.

3. which contain strings over $\Sigma = \{a, b\}$ starting with a.

Ans → $a\Sigma^*$.

4. contains all strings.

Ans → Σ^*

5. All strings ending with a.

Ans → $\Sigma^* a$. or $(a \cup b)^* a$ or $(a+b)^* a$.

6. Strings containing a. or containing a aaa substring.

Ans → $\boxed{\Sigma^* a \Sigma^*}$ ~~or~~ ~~or~~ ~~or~~

7. Lang. contain strings having bba as substring.

Ans → $\Sigma^* bba \Sigma^*$

8. Lang. contain strings where it begins with 00 and ends with 11

Ans → $00\,\Sigma^*\,11$

9. Lang contain strings exactly length 2.

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

Ans → → $aa + bb + ab + ba$ or
→ $aa \cup bb \cup ab \cup ba$.

further solve it, we get

$= aa + ab + bb + ba$
$= a(a+b) + b(a+b)$
$= (a+b)(a+b)$
$= \Sigma\Sigma$
$= (a \cup b)(a \cup b)$

10) Exactly length 3.

Ans → $\Sigma\Sigma\Sigma$

$(aa)^*$ aa aaaa

11) Lang contain string of length atleast 2, $\Sigma = \{a, b\}$

Ans → i) exactly 2 → $(a+b)(a+b)$
ii) more than 2 → $(a+b)(a+b)(a+b)^*$.

So finally we get,

$\Sigma\Sigma\Sigma^*$

12) atleast 8 length.

Ans → $\Sigma\Sigma\Sigma\Sigma^*$.

13) Atmost length 2.

Ans → i) exactly 2 → $\Sigma\Sigma$ $(a+b)(a+b)$
ii) less than 2 → $(a+b)(a+b) \cup a \cup b \cup \epsilon$.

$\Sigma\Sigma \cup \Sigma \cup \epsilon$

14) Atmost length 3.

Ans → $\Sigma\Sigma\Sigma \cup \Sigma \cup \epsilon \Sigma\Sigma$.

15) Lang. contain strings where every string contains alternate 0 and 1.

Ans → $L = \{1010, 0, 1, 01, 10, 101010, 010101 \ldots\}$

$0 \cup 1 \cup (01)^+ \cup (10)^+$

Using * operation

$0 \cup 1 \cup (01)^*0 \cup 0(10)^*1$

11.08.18 Proofs of Theorem:- (with examples) (DFA)

Pg-45
1.25) ✓

Q1)

a) Proof of the theorem "the class of Regular" language is closed under Union Operation. (Proof of Const.)

Proof idea - (In other words if A_1 and A_2 are Regular Expression, show $A_1 \cup A_2$)

We have regular languages A_1 and A_2 want on to show that $A_1 \cup A_2$ is also regular.

Let say some finite Automation M_1 recognises A_1 and some finite Automation M_2 recognises A_2. The to proof that $A_1 \cup A_2$ is regular we construct a finite Automation called it M which recognize $A_1 \cup A_2$. This is a proof by construction.

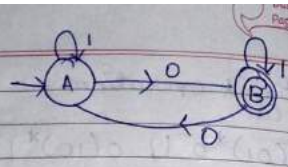
Proof - Let M_1 recognize A_1 where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and M_1 recognize A_2 where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F)$

Construct a finite automation M to recognize $A_1 \cup A_2$ where $(Q, \Sigma, \delta, q, F)$

Eg - $M_1 = \{(A, B), (0, 1), \delta, A, B)$

δ_1 =

	0	1
A	B	A
B	A	B



$M_2 = (\{P, Q\}, \{0,1\}, \delta_2, P, Q)$

δ_2 =

	0	1
P	P	Q
Q	Q	P

1. $Q = \{(M_1, M_2) \mid M_1 \in Q_1 \text{ & } M_2 \in Q_2\}$
This set Q is the cartesian product of sets Q_1 and Q_2 and is written as $Q_1 \times Q_2$

Eg- $Q = \{(A,B), (B,Q)\}$

2. Σ, the alphabet is the same the M_1 and M_2 if both M_1 and M_2 have different alphabet which is Σ_1 and Σ_2 then the input alphabet of resulting automata will be $\Sigma_1 \cup \Sigma_2$

3. δ, the transition function is defined for each other $(M_1, M_2) \in Q$, and each $a \in \Sigma$.

let $\delta((M_1, M_2), a) = (\delta_1(M_1, a), \delta_2(M_2, a))$

Eg→ $\delta((A,P), Q) = (\delta_1(A,0), \delta_2(P,0))$

$\delta(AQ, 0) = AQ \quad QQ$

$\delta(AP, 1) \quad A1 \quad Q1$

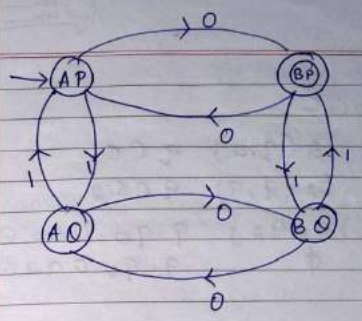
$\delta(AQ, 1) \quad A1 \quad Q1$

$\delta(BP, 0) \quad B0 \quad P0$

$\delta(BQ, 1) \quad B1 \quad Q1$

$\delta(BP, 1) \quad B1 \quad P1$

For proofing through eg- we have to also all of them.



4. q_1, the initial state q pair $(q_1, q_2) \rightarrow (AP)$

5. F, the final state F is the set of pairs in which either number is an accept state of M_1 or M_2.

$F = \{(M_1, M_2) \mid M_1 \in F_1 \text{ or } M_2 \in F_2\}$

$F / AQ, BP, BQ\}$.

Define δ.

$q \in Q$ and $a \in \Sigma \in$

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \\ \delta_2(q,a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ & } a = \epsilon \\ \phi & q = q_0 \text{ & } a \neq \epsilon \end{cases}$$

$$\delta(q,a) \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta(q,a) & q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q,a) \cup \{q_2\} & q = F_1 \text{ and } a = \epsilon \\ \delta_2(q,a) & q \in Q_2 \end{cases}$$

$$\delta(q,a) \begin{cases} \delta_1(q,a) & q \in Q_1 \\ \delta_2(q,a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ & } a = e \\ \phi & q = q_0 \text{ & } a \neq \epsilon \end{cases}$$

$0 \cup 1 \cup (01)^+ \cup (10)^+$

~~$0 \cup 1 \cup (01)^* 0 \cup 0$~~

$0 \cup 1 \cup (01)^* 0 \cup 0(10)^*]$.

Monday
13.08.18

* Equivalence with finite Automata:-

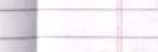
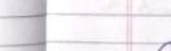
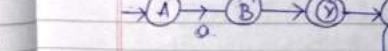
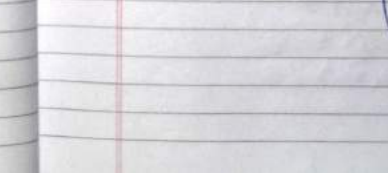
⇒ Theorem 1.54 :-

A lang is regular iff some regular expression describes it.

To prove this theorem, we need to proof 2 lemmas

→ Lemma 1.55 (1st lemma)
if a lang. is described by a regular expression, then it is regular.

Proof - Let, we have a regular expression R which describes a lang A. We have to show how to convert the regular expression R into a finite automata which recognize the lang A.

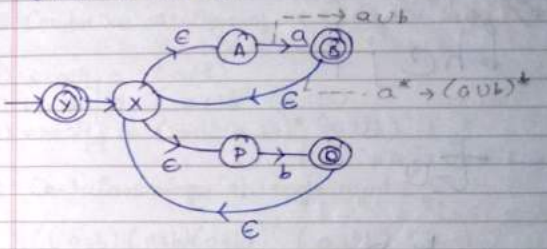
	Regular Expression		Finite Automata
1)	ϕ		
2)	a	$\Sigma = \{a, b\}$	
3)	b	$\Sigma = \{a, b\}$	
4)	ab / a o b	$\Sigma = \{a, b\}$	
5)	a∪b / a+b	$\Sigma = \{a, b\}$	

6) a^* 

7) ϵ 

Q $a(a \cup b)^*$ $\Sigma = \{a, b\}$

1. $a \cup b$
2. $(a \cup b)^*$
3. $a \circ (a \cup b)^*$



Pg-68 and 69

Q $(a \cup b)^* a (a \cup b)^* \rightarrow \Sigma^* a \Sigma^*$.

i) a
ii) b
iii) $a \cup b$.
iv) $(a \cup b)^* \rightarrow x$
v) $xa \rightarrow y$
vi) yx.

bhejo to

bhejo

bhejo

tu

usko

Q Draw a regular expression for a language which contains strings of even length.

a) exactly $0 \rightarrow \epsilon$
exactly $2 \rightarrow \epsilon \cup (a+b)(a+b)$
exactly $4 \rightarrow \epsilon \cup (a+b)(a+b)(a+b)(a+b)$

$(\Sigma\Sigma)^*$.

Q Multiple of 3 / length mod $\cong 0$.

$(\Sigma\Sigma\Sigma)^*$

Q Contains strings of odd length / length mod $2 \cong 1$.

i) exactly 0 (a/b)
ii) exactly 2 $(a+b)^* (a \cup b)$
iii) exactly 4 $((a+b)(a+b))^* (a \cup b)$

Q Contain strings of length mod $3 \cong 1$.

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

Q Strings of length mod $3 \cong 2$.

$(\Sigma\Sigma\Sigma)^* (a \cup b)(a \cup b)$ R = Remainder.

R = length 1, R = length 2

16.08.18

Q Draw a RE for a lang. which contains strings where no. of a's are exactly 2. $\Sigma = \{a, b\}$

Soln. $L = \{ aa, baa, aab, aba, baab, aabb, \ldots \}$.

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

Q exactly one a.

$b^* a b^*$

Q Lang. where strings have atleast 2 no. of a's.

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

i) exactly 2
ii) more than 2

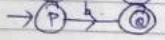
Q atmost 2 no. of a's. $\Sigma = \{a, b\}$

i) exactly 2 $\rightarrow b^* a b^* a b^*$
ii) exactly 1 $\rightarrow b^* a b^*$
iii) exactly 0 $\rightarrow b^*$

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

Q Convert following expression to finite automata:-

i) Draw FA of string a 

ii) Draw FA of string b 

iii) Draw FA of $(a \cup b)$

$a \cup b \rightarrow$ 

iv) Apply * operation now $(a \cup b)^*$ (on y with x).

v) Apply concatenation operation at last and generate a single finite automata.

$x \rightarrow (a \cup b)^*$.

a	b	x	ab.
y	x	ab	
z	a		
p	b		

vi) Apply (o) with z and a.

vii) Apply (o) with ? and b.

↑

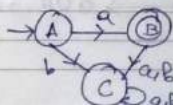
Lemma - 1.58 (RE → FA)

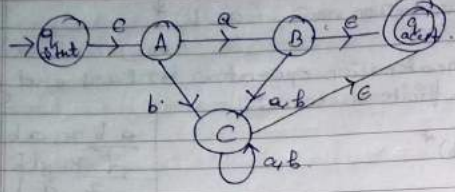
* Lemma - 1.60 (FA → RE) :-

→ If a lang. is regular then it is described by a RE.

Prove idea → If a lang. A is regular then a RE must describe this language. Because the lang. A is regular, it is accepted by finite automata. Here we describe the procedure for converting FA into an equivalent RE. (Sipser - DFA)

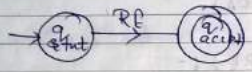
→ Let a DFA be k no. of steps.
→ Generate a GNFA (Generalized Non-deterministic FA) with $(k+2)$ no. of steps.
→ fx → Let construct a DFA to accept a string a $\Sigma = \{a, b$





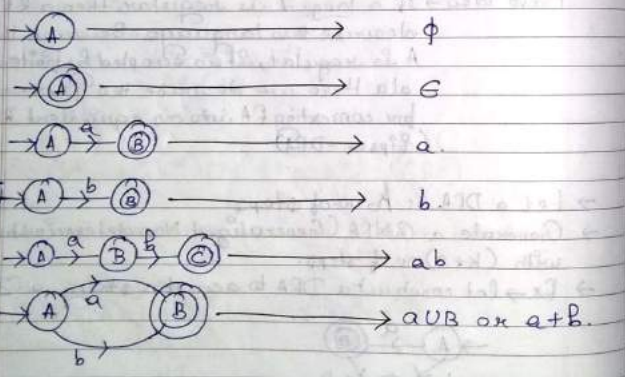
GNFA →

→ Allast we have to find out :-

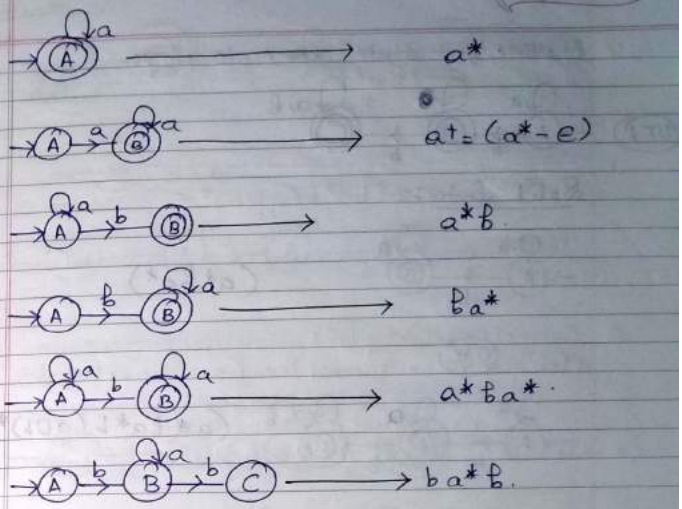


Finite Automata | Regular Expression

Single finite set with no looping.



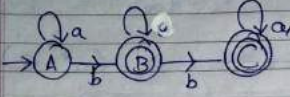
Single finite state with no looping



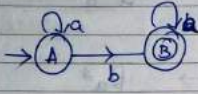
—x—

Multiple finite state with no looping:-

DFA

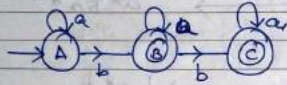


Split into:-



(a^*ba^*)

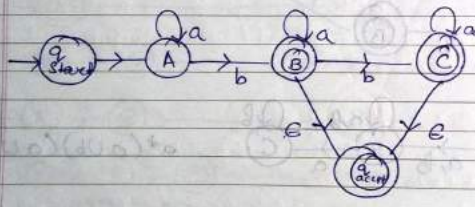
OR



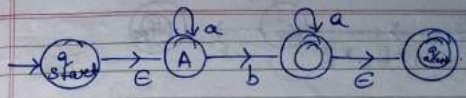
$(a^*ba^*b(a \cup b)^*)$

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

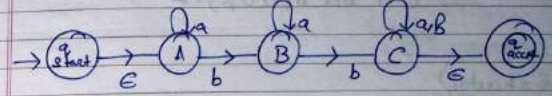
Q How to draw GNFA?



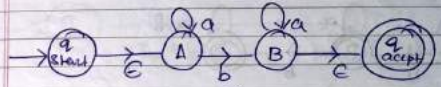
Now split this:- To reach the q accept state 2 steps is possible.



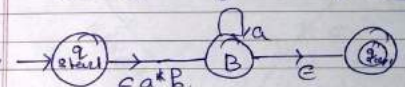
OR.



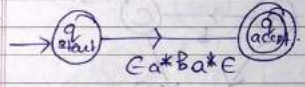
Steps to achieve final state:- (stepwise)

i) 

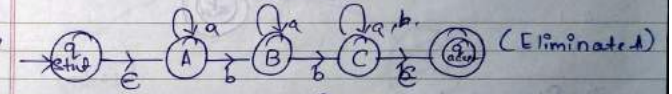
(Eliminate A)

ii) 

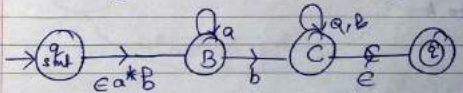
(Eliminate B)

iii) 

OR

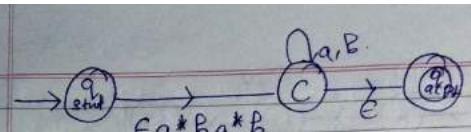
i) 

(Eliminate A)

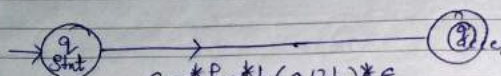
ii) 

(Eliminate B)

iii)

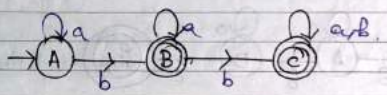


iv)

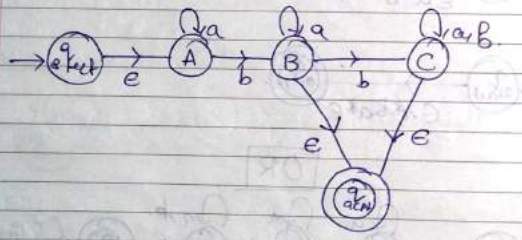


18.08.18 (Saturday)

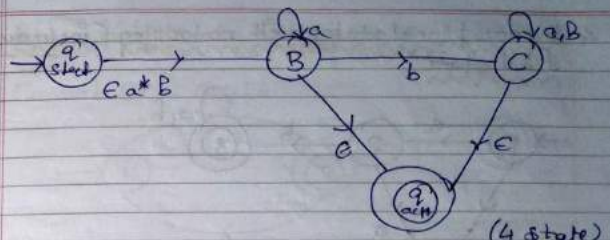
Q Convert the following DFA into regular expression using state elimination procedure. (without splitting)



i) To GNFA

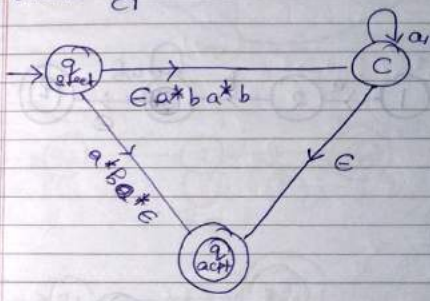


ii) Eliminate A



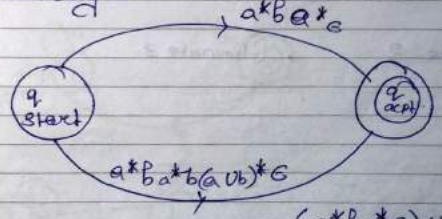
(4 state)

iii) Eliminating B.



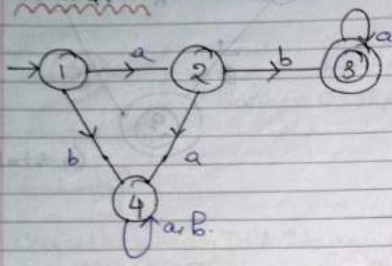
(3 state)

iii) Eliminating C

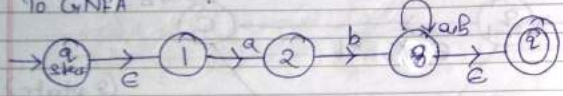


$(a^*ba^*\epsilon) \cup (a^*ba^*b(a \cup b)^*)$

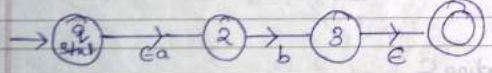
* Any no. of final states with no looping (including dead state)



i) To GNFA



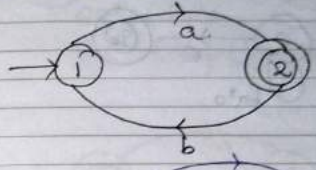
ii) Eliminate A

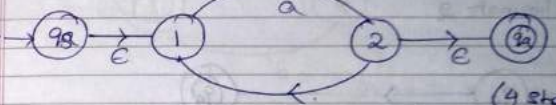


iii) Eliminate 2 , Eliminate 3.

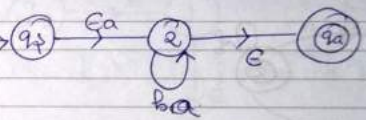
* Single finite state with no looping:-

Ex→ i) Convert FA to GNFA.

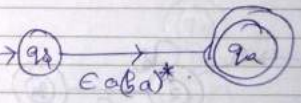


ii) 

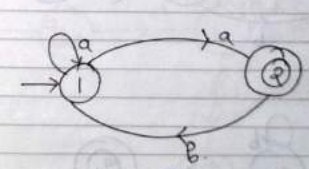
(4 states)

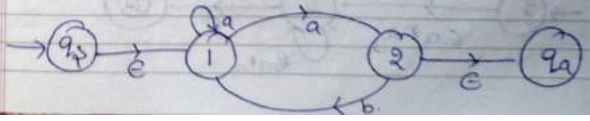
iii) 

(Eliminate state 1)

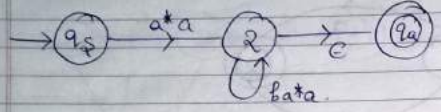
iv) 

(Eliminate state 2)

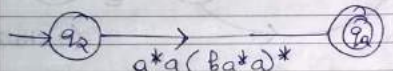
v) 

GNFA 

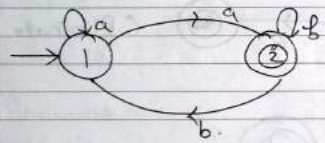
iy Eliminate (1)



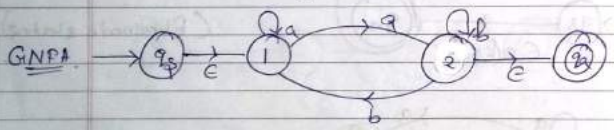
iiy Eliminate 2



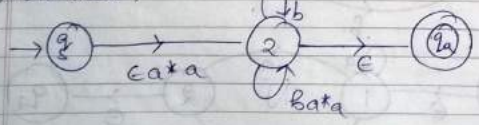
Q2



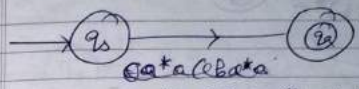
GNFA



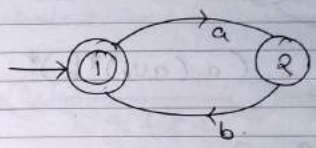
iy Eliminate 1



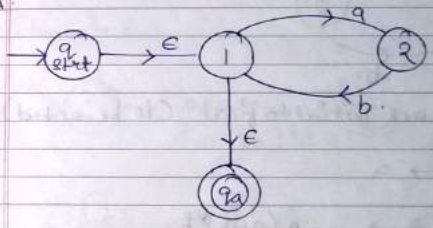
iiy Eliminate 2

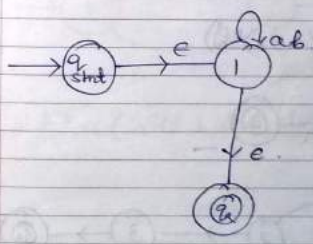


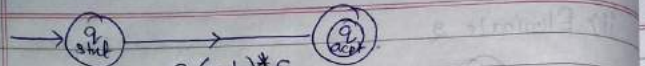
Q3



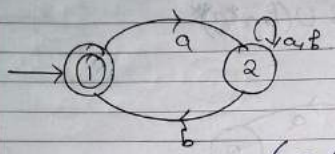
GNFA





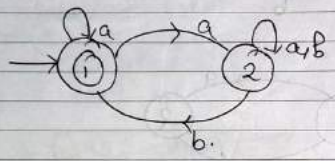


Q4)



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

Q5)



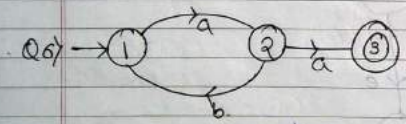
Soln. How to reach initial to final? (to be noted)

i) direct → a

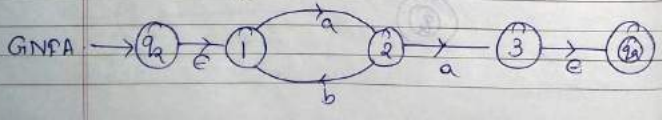
ii) through 2 → $a(a \cup b)^* b$

final → $(a \cup (a(a \cup b)^* b))$

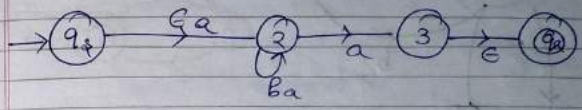
Q6)



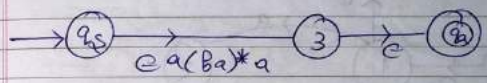
GNFA



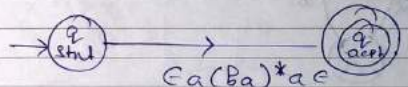
i) Eliminate 1.



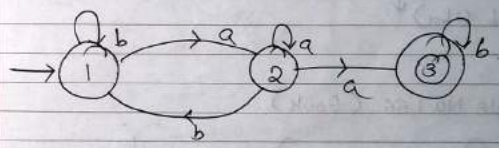
ii) Eliminate 2.



iii) Eliminate 3.

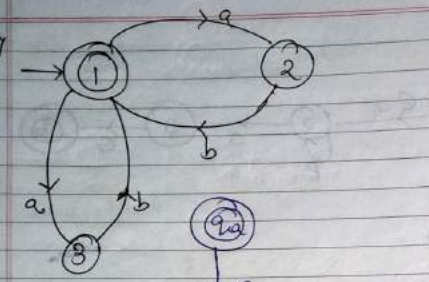


Q7)

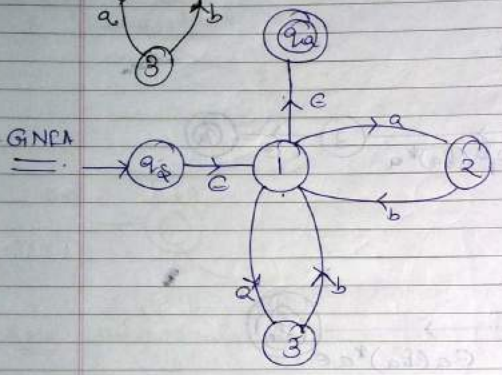


$b^* a (a \cup bb^* a)^* a b^*$.

Q8)

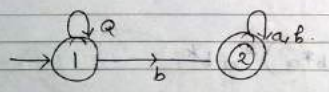


GNFA



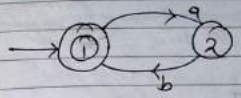
$(ab \cup ba)^*$

Q Example No. 1.66 (Book).



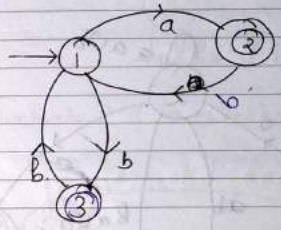
$a^* b (a \cup b)^*$

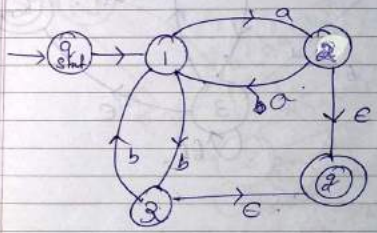
* Single final state with looping (including dead state):-

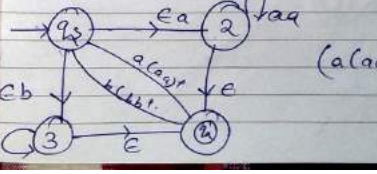


$(ab)^*$

* Multiple final state with looping :-



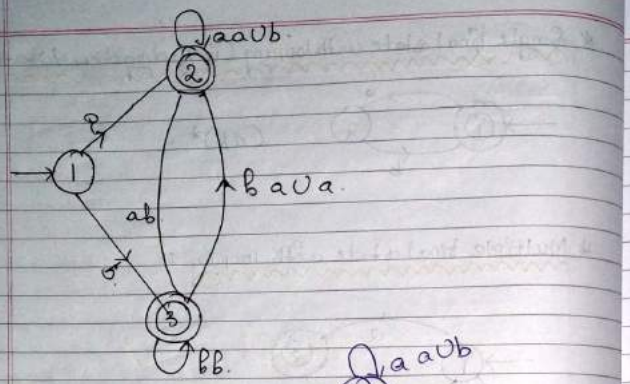




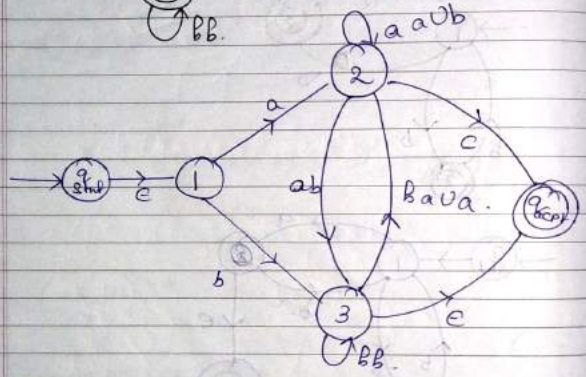
$(a(aa)^+ \cup b(BB^*))$

1.68)

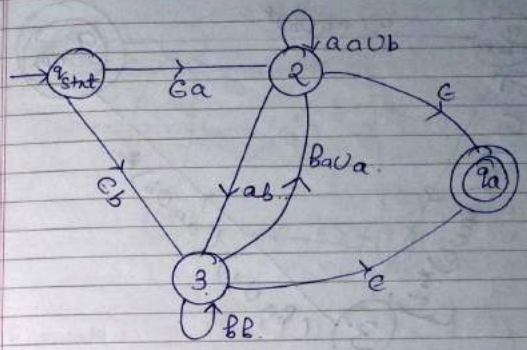
a)



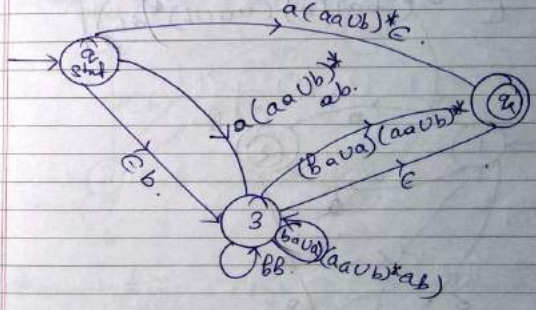
GNFA

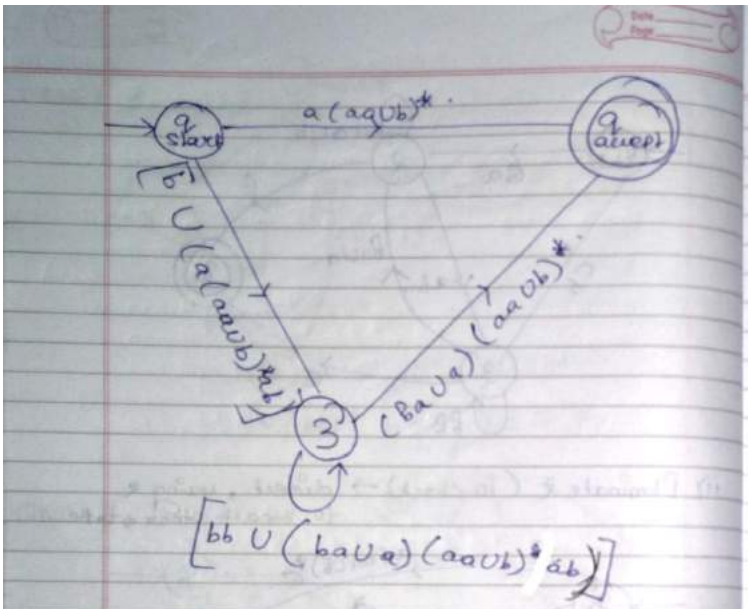


i) Eliminate 1

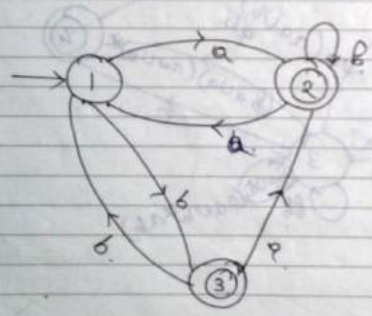


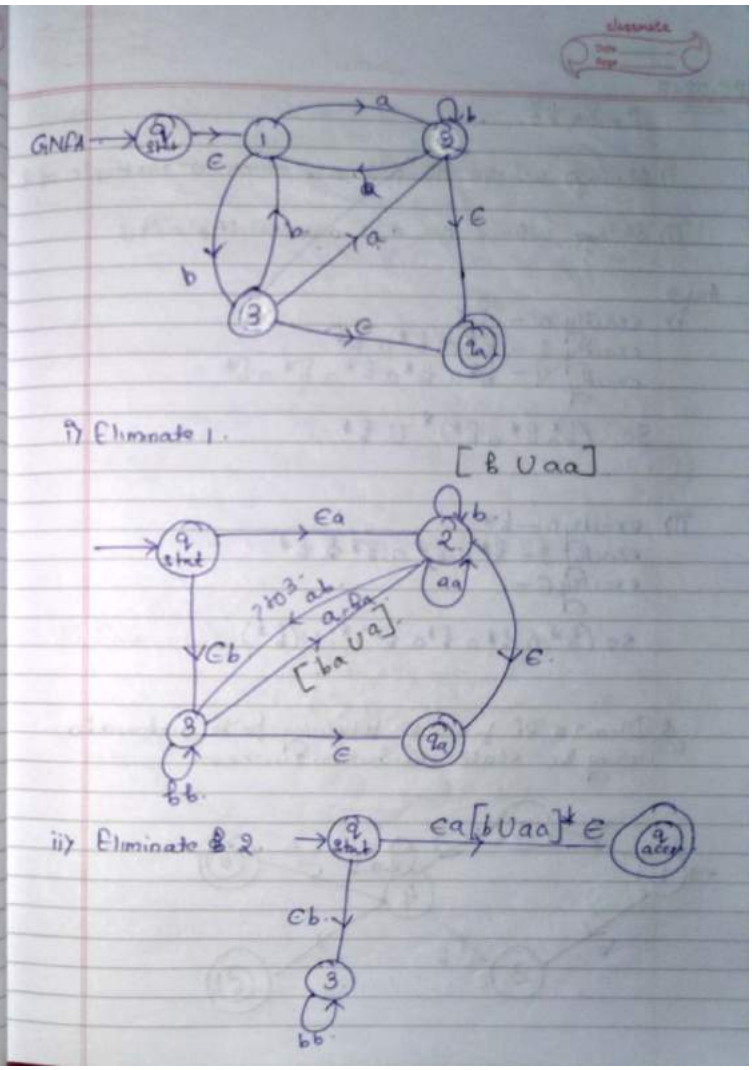
ii) Eliminate 2 (To check) → direct, using 2 to reach other state.





b)





20.08.18

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

i) Strings where no. a's are even or multiple of

ii) Strings where no. a's are multiple of 3.

Ans→

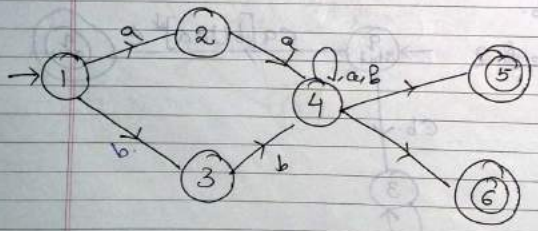
i) exactly 0 – b^*
exactly 2 – $b^* a b^* a b^*$
exactly 4 – $b^* a b^* a b^* a b^* a b^*$.

So $(b^* a b^* a b^*)^* \cup b^*$.

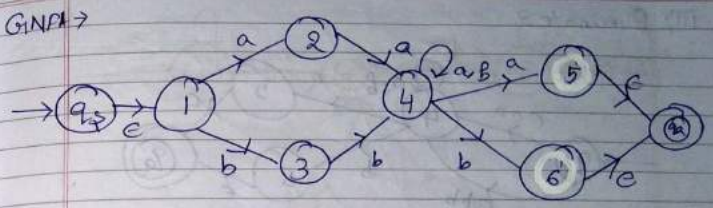
ii) exactly 0 – b^*
exactly 3 – $b^* a b^* a b^* a b^*$.
exactly 6 –

So $(b^* a b^* a b^* a b^*)^* \cup (b^*)$.

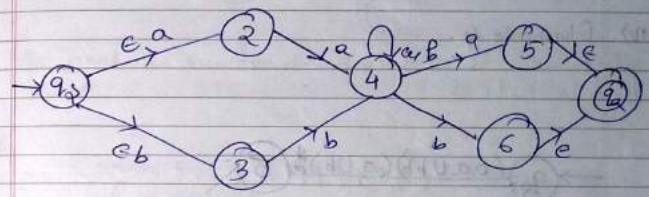
Q Draw a RE from the following finite automata using the state elimination process.



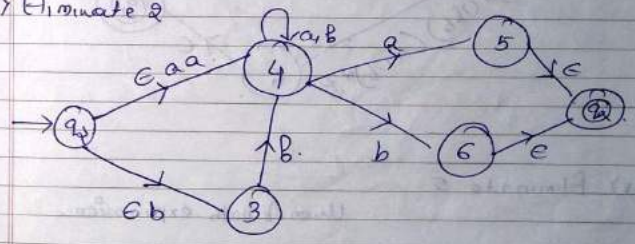
GNFA →



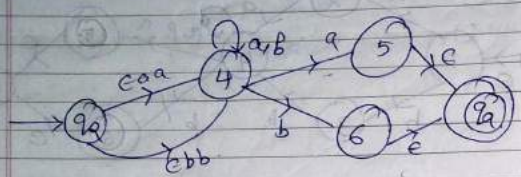
i) Eliminate 1



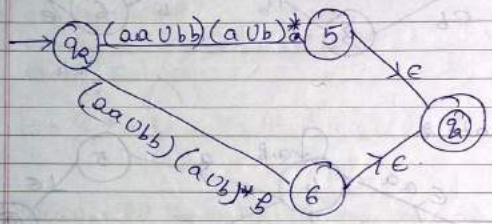
ii) Eliminate 2



iii) Eliminate 3.



iv) Eliminate 4.

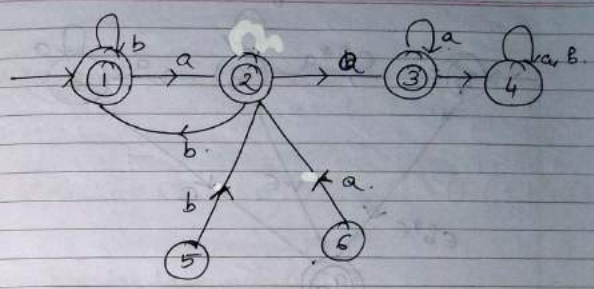


v) Eliminate 5.

Union of both expression.

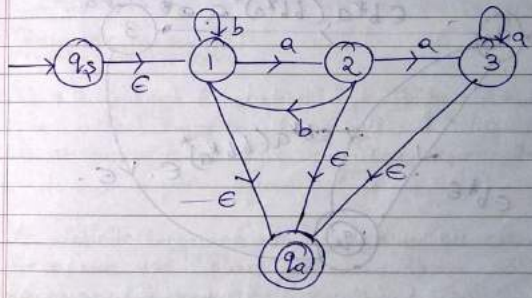
aa∪bb

Q2)

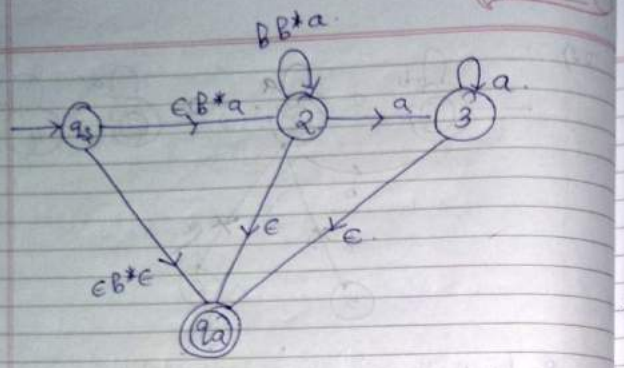


So first we remove dead state and unreachable state.

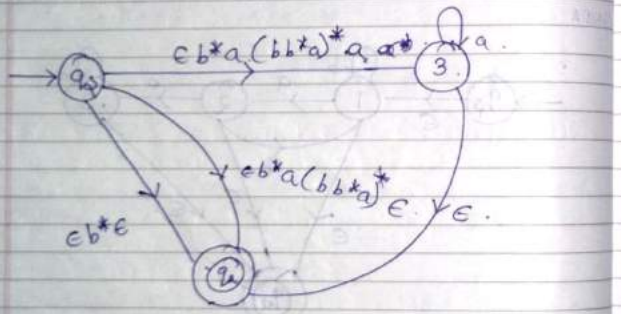
GNFA



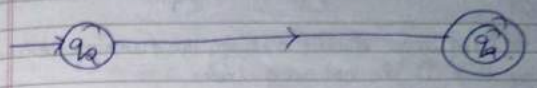
i) Eliminate 1



ii) Eliminate 2



iii) Eliminate 3.



(ϵb*ϵ) ∪ (ϵb*a (bb*a)* a a*ϵ) ∪ (ϵb*a(bb*a)

✓

220818

* Non-Regular language :-

Theorem :-
A language which contains a set of strings can be finite or infinite. A finite language is always regular.

* Regular language :-

If a language is accepted by a finite automata then it is called Regular language. A Regular language is always accepted by a regular expression.

→ An infinite language can be regular or cannot be.
→ Here, we use 'pumping lemma' theory to check a language is regular or not.

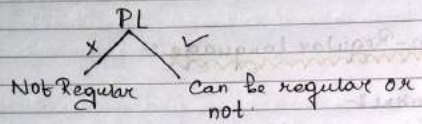
* Note :-

→ Every regular language satisfy the pumping lemma theory
→ But if a language (infinite) satisfy the pumping lemma

theory, we cannot say it is regular or not.

→ If a language doesn't satisfy the pumping lemma theory then it is not regular.

→ The pumping lemma theory is used to prove that the certain language are not regular.



Theorem → Pumping Lemma :- Regular.

Let L is an infinite regular language accepted by finite automata consist of 'p' no. of states.

Proof → Let a string $s \in L$ with $|s| \geq p$ (atleast p) then s can be written as in 3 pieces xyz

where :- i) $xyz \in$ language, $xy^iz \in L$ [$i \geq 0$]

ii) $|y| > 0$ $(y \neq \epsilon)$

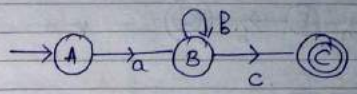
iii) substring $|xy| \leq p$ (atmost p)

Example → $L = \{ a^n b c, n \geq 0 \}$ → regular lot

$L = \{ ac, abc, abbc \ldots \}$

Draw NFA or DFA :-

FA →



$p = 3$ (A, B, C)

String → $|s| \geq 3$.] So can't take ac, you can choose abc and greater than that.

$s = \underset{x}{a}\underset{y}{b}\underset{z}{c}$ $\quad s = \underset{x}{a}\underset{y}{bb}\underset{z}{c}$

i) $ab^ic \in L$, $i \geq 0$.
$i = 0$, $ac \in L$
$i = 1$, $abc \in L$
$i = 2$, $abbc \in L$, so it satisfies condition 1

ii) $|y| \geq 1$

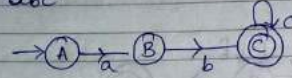
iii) $|xy| = |ab|$
$= 2 \leq 3$.

$s = abc \rightarrow n$.

A B B C → (n+1) sequence.
[pigeonhole principle]. for a

According to "pigeonhole principle" for a substring b, y is repeated, that's why it cannot be zero.

abc^n



$p = 3$

Min: $\underbrace{a}_{x}\underbrace{b}_{y}\underbrace{c}_{z}$

Pg-79 1. Proof the Pumping Lemma theorm and justify this with an example.

2. Using Pumping Lemma theorm prove that particular language is not regular.

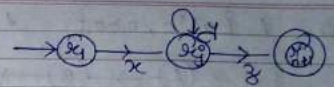
Proof:- Let $M = \{Q, \Sigma, \delta, q_1, F\}$ be a finite automata recognizing the language A and p be the no of states of M.

Let string $s = s_1, s_2 \ldots\ldots s_n$ is the language A of the length n, where $n \geq p$ (atleast p). Let $R_1, R_2 \ldots R_{n+1}$ is the sequence of states the FA goes through while processing the string s.

The sequence has length $(n+1)$ which is greater (p) or atleast $(p+1)$. Among the 1st $(p+1)$ states present in the sequence it must be same state by the pigeonhole principle. We call the 1st appearance of the repeated states i.e r_j and appearance of the repeated state is r_n.

→ Now let string $s = s_1, s_2, s_3 \ldots s_n$

Let $x = s_1 \ldots s_{j-1}$
$y = s_j \ldots s_{j-1}$
$z = s_j \ldots s_n$



As x taken FA from the initial state r_1 to r_j
As y taken FA from the initial state r_j to r_j.
As z taken FA from the initial state r_j to r_{n+1}.

Cond.1 If the language 'L' is regular and the string $s = xyz$ goes to final (r_{n+1}) states then the $xy^iz, i \geq 0$ must reach to the final state.

Cond.2 We know that the position of the sequence of the repeated state j and l is not the same, it means $|y| > 0$.

Cond.3 As $l \leq (p+1)$, so $|xy| \leq p$.
Thus we have satisfied all the three condition of pumping lemma for regular language.

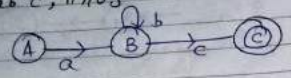
Cond.3)

→ If a regular language, then is a p known as "pumping length". If s is any string, s is the length of atleast p. Then s may be divided into 3 pieces xyz & at

1) Let A be a regular language accepted a FA with 'p' no of states. Let s be a string $\in A$. $|s| \geq p$. (length of string = atleast p).

Then divide s into to 3 pieces xyz.

Ex- $A = \{ab^n c, n \geq 0\}$ $L = \{ac, abc, abbc, \ldots\}$



$p = 3$ $\underset{x}{a}\ \underset{y}{b}\ \underset{z}{c}$

* Pigeonhole Principle:-

→ This principle states that if p pigeons are placed into fewer than p holes, then some hole has to have more than one pigeon in it.

→ According to pigeonhole principle, the substring y in the string xyz goes through sequence of states where atmost 2 states must be same.

→ That's why the length of substring y can't never be zero(0).

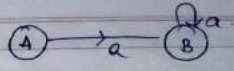
$\underset{a}{x}\ \underset{b}{y}\ \underset{c}{z} = 3$

$\underset{x}{A}\ \underset{y}{B}\ \underset{z}{B}\ C$ $|y| = 1$

Q $L = \{a^n, n \geq 1\}$ (regular language)

Soln: $n=1$ a
$n=2$ aa
$n=3$ aaa

$L = \{a, aa, aaa \ldots\}$.



$p = 2$ (A, B)

$|s| \geq 2$.

So can't take a, you can take aa or more than that

$s = \underset{x}{a}\ \underset{y}{a}\ \underset{z}{\epsilon}$

i) a^i:

$i = 0$
$i = 1, a \in L$
$i = 2, aa \in L$
$i = 3, aaa \in L$

ii) $|y| = 1$

iii) $|xy| = |aa| = 2 \leq 2$.

25.08.18

Ex-1.73

Let Φ be a language.

Step-1 → A lang. L

Step-2 → FA with 'p' no of states.

Step-3 → Put a string ϕ from the lang. of length atleast p

Step-4 → Divide ϕ into xyz

Step-5 → Check 3 conditions

$L = \{ a^n b^n \mid n \geq 0 \}$ if $n = 1, 2 \ldots$

$L = \{ \epsilon, ab, aabb, aaabbb \ldots \}$.

* Using method of contradiction :-

Assume to the contrary that lang. B is regular. Let p be the complete length given by pumping lemma. Choose the string ϕ from lang. L which is $a^p b^p$. Now string ϕ is a member of lang. L and string has length $\geq p$. Here pumping lemma guarantees that ϕ can be split into 3 pieces $\phi = xyz$, where any $i \geq 0$ string $(xy)^i z$ is in the language.

Consider 3 cases to show that particular condition is not possible

1. String y consists of only "a's". In this case $xyyz$ has more no of a's than "b" so the string $xyyz$ is not in the lang. by violating condition 1 of Pumping lemma. Represents contradiction.

2. String y consists of only b's. In this case xy^0z has more no of b's than a's. So that why xy^2z is not in the language by violating condition 1 of pumping lemma. Represents contradiction.

3. String y consists of both a's and b's. In this case string yz, xy^2z or $xyyz$ can have equal no of a's and equal no of b's but they will out of order. Some a's will come after b. This means xy^2z is not in the language by violating the length of complete lemma. Also represents contradiction.

. This lang. is not regular using contradiction

Q2) $L = \{a^n b a^n \mid n > 0\}$ is not regular.

$S = xyz$

1) $y = b \rightarrow aba$
2) $y = a \rightarrow \underset{xyz}{caba} \rightarrow \underset{x}{\epsilon} \underset{y}{aba} \underset{z}{a}$
3) $y = ab \rightarrow \underset{xyz}{abaa} \rightarrow \underset{x}{aba} \underset{y}{\epsilon} \underset{z}{a}$
4) $y = ba \rightarrow \underset{xyz}{abaa} \rightarrow \underset{x}{\epsilon} \underset{y}{aba} \underset{z}{a}$
5) $y = aba \rightarrow \underset{x}{\epsilon} \underset{y}{aba} \underset{z}{\epsilon}$

xy^2z

1. abba.
2. ϵaaba
3. ϵabaa
4. abaaa
5. ϵabaabaϵ

1. If y consists of only b's. The resulting string xy^2z contains more than one b thats why xy^2z is not in the language. This case represents a contradiction.

2. String y consist of only a's. The resulting string xy^2z contains ~~---~~ one a thats why xy^2z is not in the lang. Represents a contradiction.

3. Resulting Substring y consist of is ~~and~~ then b's. Result string xy^2z contains more than one a and b. thats why xy^2z is not in the lang. Represents contradiction.

4. Contains more than one b then a.

5. Multiple of aba.

Prove that :-

1. $L = \{a^n \mid n$ is a prime no.$\}$ is not regular.

2. $L = \{a^n \mid n = 2, 3, 5, 7 \ldots\}$ is not regular.
   or
   $L = \{$string of length $\rightarrow$ prime no.$\}$.

Let L be the regular lang and p is the pumping length given by pumping lemma (ie there is a finite automata which accepts the lang. L with p no. of strings). Choose s to be the string from lang. L where $|s| \geq p$ (atleast p). Let s be the string of length = p. Divide the string s, into 3 parts x, y, z where $|y| > 0$ and length of $|xy| \leq p$. Since $xyz \in L$ then xy^iz should also belong to z for $\forall i \geq 0$.

Lang. contain strings of length (which is prime no) We have to show that length of xy^iz is also prime no.

Let $|xy^iz| = |xyz| + |(i-1)y|$

$= p + |y(i-1)|$
$= p + k(i-1)$ $(|y| = k)$ $(|y| > 0)$
$= p + k(p+1-1)$ $(i = p+1)$
$= p + kp$
$= p(k+1)$

Here the length of xy^iz is a composite no. That's why this resulting string xy^iz is not in the lang. L. That's why given question is not regular using contradiction.

Exercise 1.29

Q $L = \{0^n 1^n 2^n \text{ where } n \geq 0\}$ is not regular

$s = xyz$.

1) $y = 1 \rightarrow$ 0 1 2 $\rightarrow xy^2z$
$x\ y\ z$ $= 0112$

2) $y = 0 \rightarrow$ ϵ 0 12 $\rightarrow xy^2z$
$x\ y\ z$ $= \epsilon 0012$

3) $y = 2 \rightarrow$ 0 1 2 $\epsilon \rightarrow xy^2z$
$x\ y\ z = 0122\epsilon$

4) $y = 01 \rightarrow$ ϵ 01 2 $\rightarrow xy^2z$
$x\ y\ z$ $= \epsilon 01012$

5) $y = 12 \rightarrow$ 0 12 $\epsilon \rightarrow xy^2z$
$x\ y\ z = 012126$

6) $y = 012 \rightarrow$ ϵ 012 $\epsilon \rightarrow xy^2z$
$x\ y\ z = \epsilon 012012\epsilon$

0 1 2

$0^2 1^2 2^2$

0112

089

29.08.18 Pg - 97

Assume that particular lang L is regular. Let 'p' be the pumping length given by pumping lemma. Choose s to be the string from the language because the string s is a member of lang. L and the length of the string is atleast p, the pumping lemma guarantees that strings can be split into xyz $\boxed{i \geq 0}$, string xy^iz is in the lang. L.

Here we consider 2 possibilities:-

Cond: i) String y consists only of zeros, ones or two's (0,1,2). In this case, string $xyyz$ ($i = 2$) will not have equal no of 0's, 1's and 2's that's why $xyyz$ is not a member of lang. which is contradiction.

Cond: ii) Substring y consists of more than one kind of symbol. In these cases string $xyyz$ ($i = 2$) will have zeros (0's), ones (1's) and twos (2's) out of pattern, that's why $xyyz$ is not a member of lang which represents contradiction.

That's why L is not regular using method of contradiction.

Q $L = \{a^n \mid n \text{ is a prime no}\}$ is not regular.
(OR Method)

$n = 2, 3, 5, 7, 11 \ldots$

$L = \{aa, aaa, aaaaa, aaaaaaa \ldots\}$.

Let L is a regular language → FA (P = no. of states = 3)

$|s| \geq P = 3$ (Always take ≥ 3)

$|aaaaaaa| = 7 \geq 3$.

i) $xyz \in L$, $xy^iz \in L$.

$\underbrace{aa}_{x}\ \underbrace{a}_{y}\ \underbrace{aaaa}_{z}$

$|aaa| \leq 3 \quad 3 \leq 3$ (✓)

$xy^iz \in L$. $(i = 2)$

$\underbrace{aa}_{x}\ \underbrace{aa}_{y}\ \underbrace{aaaa}_{z}$ $|aa\,aa\,aaaa| = 8 \geq 3$.

$8 \notin L$.

As it is composite so it will not belong to L.

- ✓ i) $xyz \in L$, $xy^iz \in L$, $i \geq 0$
- ✓ ii) $|y| > 0$
- ✓ iii) $|xy| \leq p$

Example No. 1.16 :-

$L = \{1^{n^2} \mid n > 0\}$ is not regular.

$L = \{n^2 \text{ no. of } 1's\}$.

$n = 0, 1, 2, 3, 4 \ldots$

$n^2 \in 0, 1, 4, 9, 16, 25 \ldots$

$L = \{\varepsilon, 1, 1111, 111111111, \ldots\}$.

Let L is a regular language → FA (P = no. of states = 4)

$|s| \geq P = 4$

$|1111111111| = 9 \geq 4$.

$xy \leq P$.
$P < P^2$.
$|y| < P$.

i) $\underbrace{1111}_{x}\ \underbrace{1}_{y}\ \underbrace{111111}_{z}$

$|1111| \leq 4 \quad 4 \leq 4$ (✓)

$xy^iz \in L$. $(i = 2) \rightarrow xyyz$.

$\underbrace{111}_{x}\ \underbrace{11}_{y}\ \underbrace{11111}_{z}$ $|111\,11\,11111| = 10 \geq 3$.

$10 \notin L$

As 10 cannot be n^2.

∴ OR Method :- (Pg - 82).

$|xyyz| = |xyz| + |y|$

$< p^2 + p$

$< p^2 + p < (p+1)^2 = p^2 + 1^2 + 2p$.

$|xyz| < |xyyz| < (p+1)^2$

$p^2 < |xyyz| < (p+1)^2$

Q $L = \{0^i 1^j \mid i > j\}$ is not regular

Soln $L = \{001, 0001, 00011 \ldots\}$.

Let L is a regular language ($p = 4$)

$S = 00011$

$|s| = 5 > 4$.

$\underset{x}{\epsilon}\ \underset{y}{00011}\ \underset{z}{\epsilon}$

$|xy| \leq p$

$1 \leq 4$

$xy^i z$ $i = 0$

$0011 \notin L$

$S = 00011$

$\underset{x}{000}\ \underset{y}{1}\ \underset{z}{1}$

$xy^i z$ $i = 2$

$000111 \notin L$

Example 1.75

Q Prove that

$L = \{ww \mid w = \{0,1\}^*\}$ is not regular

~~$L = \{01, 10, 0101, 1010, 100, 0, 1, 11, 00 \ldots\}$.~~

$L = \{00, 11, 0101, 1010, 100100 \ldots 0011\,0011, 1100\,1100 \ldots\}$

Let L is a regular language ($p = 5$)

$S = 100100$

$\underset{x}{1}\ \underset{y}{00}\ \underset{z}{100}$

$|xy| \leq p$

$3 \leq 5$

$S = \underset{x}{100}\ \underset{y}{100}\ \underset{z}{}$

$|xy| \leq p$

$3 \leq 5$

$xy^i z$ $i = 2$

$xyyz$

$100100 \notin L$

$|xy| \leq 5$

$xy^i z$ $i = 2$

$xyyz$

$1000100 \notin L$

$y = 0/00$ $|y| \leq p$

$y = 1/11$

$xy = 01$

$xy = 10$

Q Example 1.74

$L = \{w \mid w$ has equal no. of 0's and 1's$\}$.

$y = 0,1$ $L = \{01, 10, 0011, 1100, 111000, 000111 \ldots\}$.

$y = 11, 111, 10$.

Let L is a regular language → FA ($p = 4$)

$y = 10$ ✓ $S = 111000$.

$\underset{x}{111}\ \underset{y}{000}\ \underset{z}{}$

$|xy| \leq p$

$4 \leq 4$

$xy^i z$ $i = 2$ $i = 0$

$111010\,00 \in L$ 111000

$y=1$: $S = \underset{x}{\varepsilon}\,\underset{y}{111}\,\underset{z}{000}$ — xy^iz $(i=3)$ — $\in 111\,111000 \notin L$ — $|xy| \leq p$, $1 \leq 4$ — ×

$y=0$: $S = \underset{x}{111}\,\underset{y}{000}\,\underset{z}{\varepsilon}$ — xy^iz $(i=2)$ — $111\,000\,00 \notin L$ — $|xy| \leq p$, $4 \leq 4$ — ×

$y=11$: $S = \varepsilon\,\underset{xy}{111000}\,\underset{z}{\varepsilon}$ — xy^iz $(i=3)$ — $\in 111111\,1000 \notin L$ — $|xy| \leq p$, $3 \leq 4$ — ×

Ex-1.29

c) $L = \{a^{2^n} \mid n \geq 0\}$ is not regular.

$L = \{2^n$ nb. of $a's\}$, $n \geq 0$.

04.10.18

$L_1 = \{a^n b^n c^m d^m, n \geq 1, m \geq 1\}$.

$L_2 = \{a^n b^m c^m d^n, n \geq 1, m \geq 1\}$.

are CFL. Design CFG from following langua

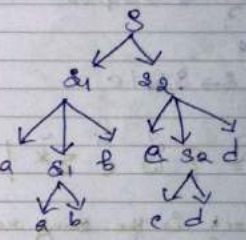
i) $L_1 \cup L_2$
ii) $L_2 . L_1$
iii) L_1^*

$L_1 = \{abcd, abccdd, aabbccdd, \ldots\}$

$L_2 = \{abcd, aabcdd, aabbccdd, aabbbbccdddd \ldots\}$

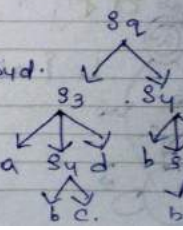
$L_1 =$

$S_p \to S_1 S_2$
$S_1 \to aS_1b \mid ab$.
$S_2 \to aS_2d \mid cd$.



$L_2 =$

$S_q \to S_3 S_4$.
$S_3 \to aS_3d \mid ad \mid aS_4d$.
$S_4 \to bS_4c \mid bc$



$L_1 =$ $S \to aS_1 bS_2$.
$S_1 \to aS_1 b/\epsilon$
$S_2 \to cS_2 d/cd$.

or $S \to AB$
$A \to aAb/ab$
$B \to cBd/cd$.

$L_2 =$ $S_q \to aS_q d / aPd$.
$P \to bPc/bc$

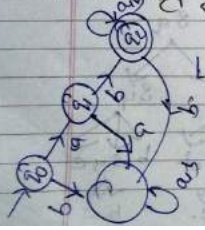
i) $L_1 \cup L_2 =$
$S \to S_p / S_q$

ii) $L_2 . L_1$
$S \to S_q . S_p$

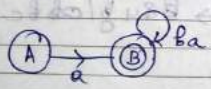
iii) L_1^*
$S \to S . S_p / \epsilon$.

Q $L_1 = \{ \omega \in \Sigma^* / \omega = (ab)^n, n \geq 1 \}$.

i) Check is the lang. regular or not
ii) Justify your answer.

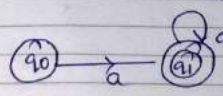


$L = \{ abababab... \}$.

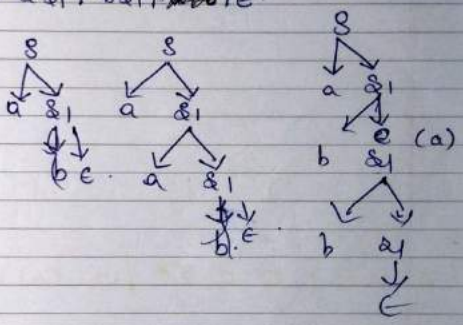


starting with symbol a $\Sigma = \{a, b\}$.
draw a CFG which contains.

$L = \{ a, ab, aab, aba, aabb, aaaa, aaa, abb... \}$.



$S \to aS_1$
$S_1 \to aS_1 / bS_1 / \epsilon$.



$i \geq 0$
$|vy| > 0$
$|vxy| \leq p$
$uv^i xy^i z \in L$.

30.08.18

Chapter-2 :- Context Free Language :-

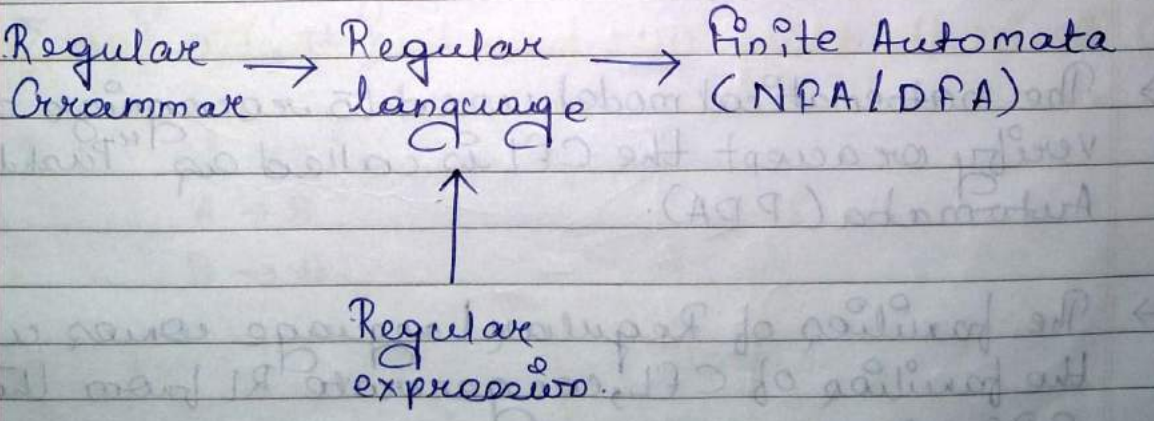
→ Most compilers are interpreters contain a component called a 'parser' that extracts the meaning of a program prior to generating compiled code or performing the interpreted execution.

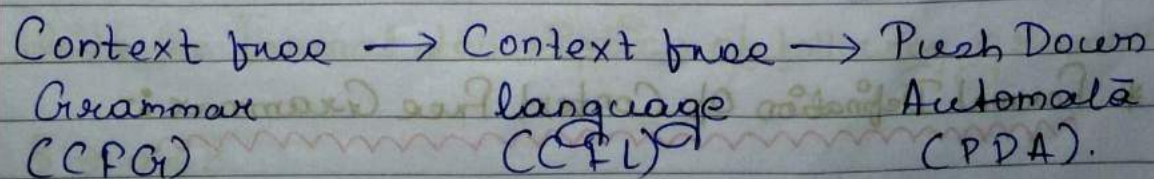
→ A no. of methodologies facilitate the construction of a parser once a context-free grammar is available.

→ The collection of languages associated with context-free grammars are called "context-free languages"

→ "Pushdown Automata" - Are useful because they allow us to gain additional insight into the power of context-free grammars.

2.1 → Context-Free Grammars :-





In this chapter we present the context free grammar which is a powerful grammar for describing languages.

Application :-

1. The most important application of CFG occurs in the specification and compilation of programming languages.
2. The design of compilers and interpreters for any programming language often starts with obtaining a grammar for the language.

* Every Regular language is a context free language

RL ⊂ CFL

→ The mathematical model used to recognize or verify or accept the CFL is called as "Pushdown Automata (PDA).

→ The families of Regular Language comes under the families of CFL, can generate RL from the CFG also.

* Formal Definition Of Context Free Grammar :-

Let's formalize our notion of a context-free grammar (CFG) :-

A CFG is represented as a 4 tuple :-

where (V, Σ, R, S)

$V \rightarrow$ finite set called the variables.
$\Sigma \rightarrow$ finite set, disjoint from V, called "terminals"
$R \rightarrow$ R is a finite set of rules with each rule being a variable and a string of variables and terminals.
$S \rightarrow S \in V$ is the start variable.

Example →
$A \rightarrow 0A1$ → capital letters do not contain in the language.
$A \rightarrow B$ → a, B can be possible.
$B \rightarrow \#$

$V \rightarrow \{A, B\}$

$\Sigma \rightarrow \{0, 1, \#\}$

R :
$A \rightarrow 0A1$
$A \rightarrow B$
$B \rightarrow \#$

$S \rightarrow A$

Q Write the formal description of following grammar :-

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

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

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

R : $S \to aaa\$B$
$B \to aB$
$B \to b$

$S : \{S\}$ (starting symbol).

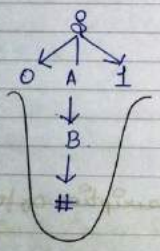
* Context free grammar to Context free language (CFG → CFL) :-

Q $S \to 0A1$
$A \to B$
$B \to \#$

all capital letters and the symbols that never comes in string.

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

i) Derive the string 0#1 from this particular grammar.



(Parse tree / derivation tree)

Q Derive the string 000#111 from the following grammar.

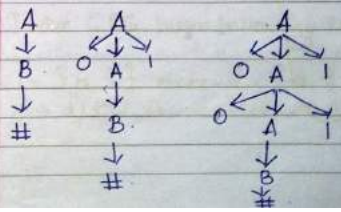
$A \to 0A1$
$A \to B$
$B \to \#$

(Parse tree / Derivation tree)

Q Draw the language generated by the above grammar.

$A \to 0A1$
$A \to B$
$B \to \#$
} Grammar

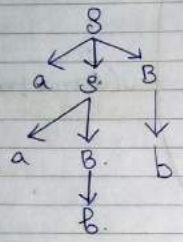
$L = \{\#, 0\#1, 00\#11, \ldots\}$.



$L = \{ 0^n 1^n , n \geq 0 \}$

Q $S \to aSB$
$S \to aB$
$B \to b$

Generate the string aabb from the above grammar.



left most derivation:-

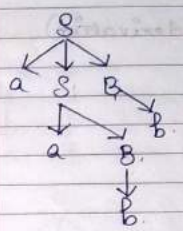
$S \to a \underline{S} B$
$\to a\ aB\ \underline{B}$
$\to a\ ab\ \underline{B}$

right most derivation :-

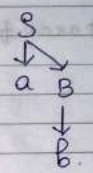
$S \to a\ S\ \underline{B}$
$\to a\ \underline{S}\ b$
$\to a\ aB\ b$
$\to a\ ab\ b.$

Q Derive the language from the above grammar:-

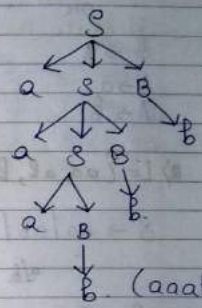
$S \to aSB$
$S \to aB$
$B \to b$



(aabb)



(ab)



(aaabbb)

$L = \{ ab, aabb, aaabbb \ldots \}$

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

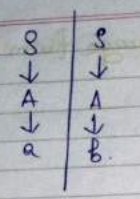
01.09.18

* Context Free Language to Context Free Grammar:-

Q Derive CFG from following language:-

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

$S \to a$
$S \to b$
or
$S \to A$
$A \to a/b$
or
$S \to A$
$A \to a$
$A \to b$



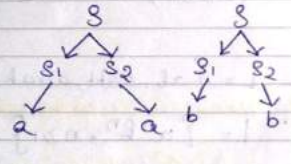
(parse tree derivation)

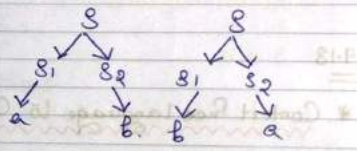
2) $L = \{aa, ab, ba, bb\}$

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

a/b a/b

$S \to S_1 S_2$
$S_1 \to a/b$
$S_2 \to a/b$

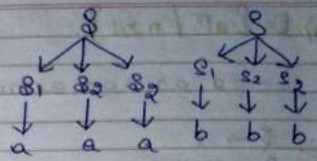


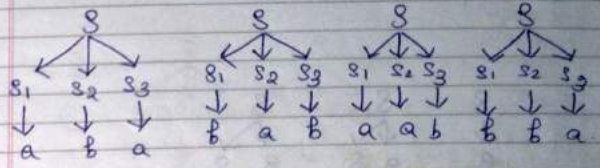


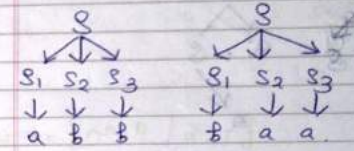
(parse tree derivation)

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

$S \to S_1, S_2 S_3$
$S_1 \to a/b$
$S_2 \to a/b$
$S_3 \to a/b$





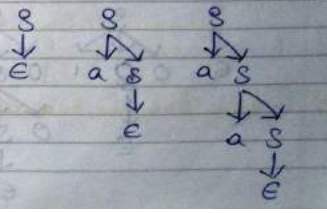


Q Draw CFG of the following language :-

$L = \{a^n / n \geq 0\}$

$L = \{\epsilon, a, aaa, aa, aaaa \ldots\}$.

$S \to \epsilon$
$S \to aS$
} (OR) $S \to \epsilon / aS$.



4) $L = \{ a^n \mid n \geq 1 \}$

$L = \{ a, aa, aaa, aaaa \ldots \}$

$S \to a$

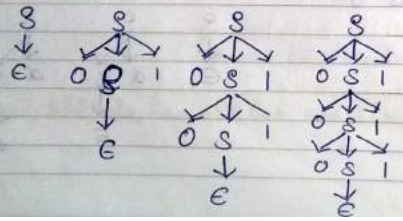
$S \to aS$

5) $L = \{ 0^n 1^n \mid n \geq 0 \}$

$L = \{ \epsilon, 01, 0011, 000111 \ldots \}$

$S \to \epsilon$

$S \to 0S1$

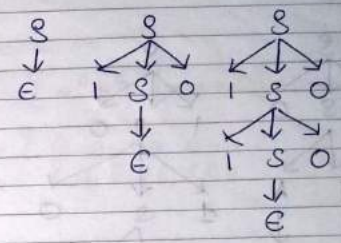


6) $L = \{ 1^n 0^n \mid n \geq 0 \}$

$L = \{ \epsilon, 10, 1100, 111000 \ldots \}$

$S \to \epsilon$

$S \to 1S0$



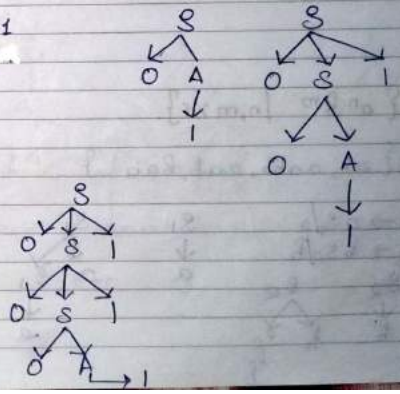
7) $L = \{ 0^n 1^n \mid n \geq 1 \}$

$L = \{ 01, 0011, 000111, 00001111 \ldots \}$

$S \to 0S1$

$S \to 0A1$

$A \to 1$



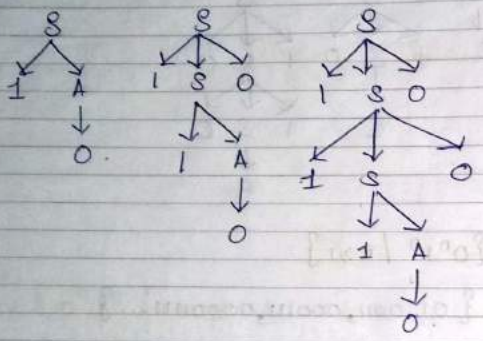
8) $L = \{ 1^n 0^n / n \geq 1 \}$

$L = \{ 10, 1100, 111000 \ldots \}$

$S \to 1S0$

$S \to 1A0$

$A \to 0$



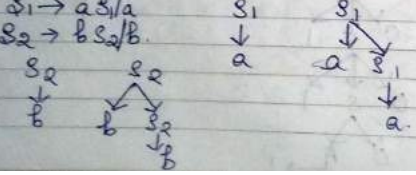
03/09/18

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

$L = \{ ab, aabb, aab, baa \ldots \}$

$S_1 \to aS_1/a$

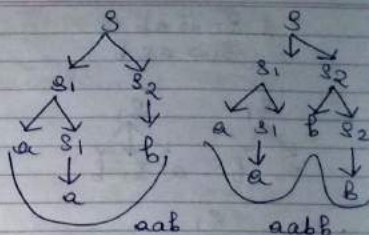
$S_2 \to bS_2/b$



$S \to S_1 S_2$

$S_1 \to aS_1/a$

$S_2 \to bS_2/b$



Q $L = \{ a^n b^m / n, m \geq 0 \}$.

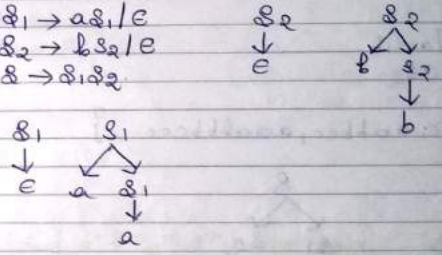
$L_1 = \{ \epsilon, a, aa, aaa \ldots \}$

$L_2 = \{ \epsilon, b, bb, bbb \ldots \}$

$S_1 \to aS_1/\epsilon$

$S_2 \to bS_2/\epsilon$

$S \to S_1 S_2$

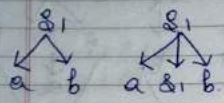


Q $L = \{ a^n b^n c^m / n, m \geq 1 \}$

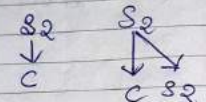
$S_1 = L_1 = \{ ab, aabb, aaabbb \ldots \}$

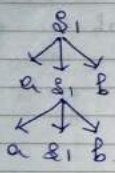
$S_2 = L_2 = \{ c, cc, ccc \ldots \}$

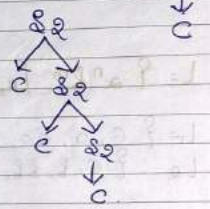
$S_1 \to ab$
$S_1 \to aS_1b$



$S_2 \to c$
$S_2 \to cS_2$

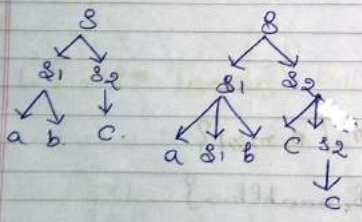






$S \to S_1S_2$
$S_1 \to ab \mid aS_1b$
$S_2 \to c \mid cS_2$

$L = \{abc, aabbcc, aaabbbccc \ldots\}$

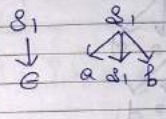


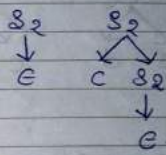
Q2) $L = \{a^n b^n c^m \mid n, m \geq 0\}$

$L_1 = \{\epsilon, ab, aabb, aaabbb \ldots\}$
$L_2 = \{\epsilon, c, cc, ccc \ldots\}$

$S_1 \to \epsilon$
$S_1 \to aS_1b$

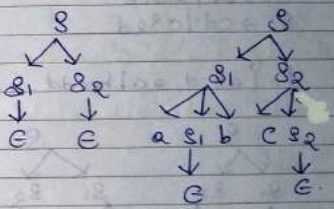
$S_2 \to \epsilon$
$S_2 \to cS_2$





$L = \{\epsilon, abc, aabbcc \ldots\}$

$S \to S_1S_2$
$S_1 \to \epsilon \mid aS_1b$
$S_2 \to \epsilon \mid cS_2$

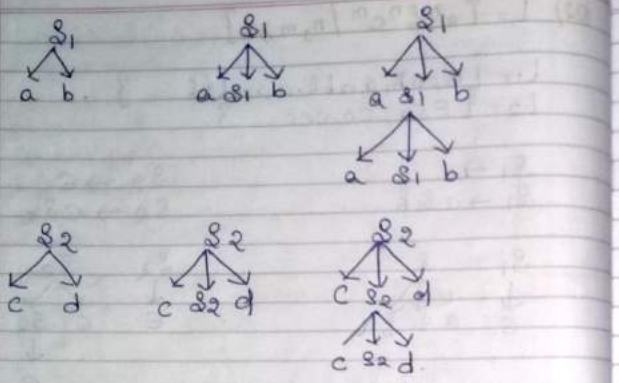


Q3) $L = \{a^n b^n c^m d^m \mid n, m \geq 1\}$

$L_1 = \{ab, aabb, aaabbb \ldots\}$
$L_2 = \{cd, ccdd, cccddd \ldots\}$

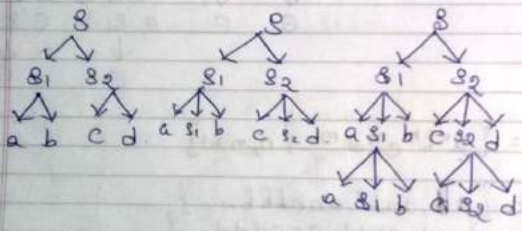
$S_1 \to ab$
$S_1 \to aS_1b$

$S_2 \to cd$
$S_2 \to cS_2d$



$S \to S_1 S_2$
$S_1 \to ab \mid a S_1 b$
$S_2 \to cd \mid c S_2 d$

$L = \{abcd, aabbccdd \ldots.\}$



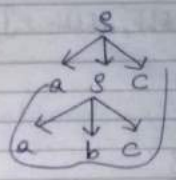
Q4) $L = \{a^n b c^n ; n \geq 1\}$

Q5) $L = \{a^n b c^n , n \geq 0\}$

05.09.18

4) $L = \{abc, aabcc \ldots.\}$

$S \to abc$
$S \to aSc$



or

$S \to aS_1c$
$S_1 \to b$
$S \to aSc$

Wrong method
$S \to b$
$S \to aSc$

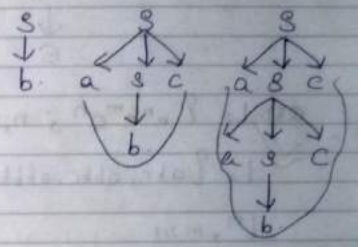
As extra b will be created Hence it will be wrong.

5) $L = \{b, abc, aabcc \ldots.\}$

$S \to b \mid aSc$

or

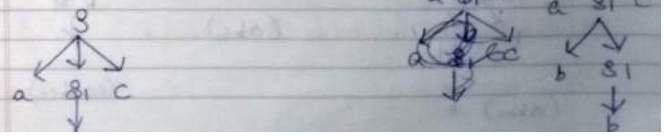
$S \to b$
$S \to aSc$



Q6) $L = \{ab^n c ; n \geq 1\}$

$L = \{abc, abbc, abbbc, abbbbc \ldots.\}$

$S \to abc$ $S \to aS_1c$
$S \to \cancel{aSb}$ $S \to bS_1 \mid b$

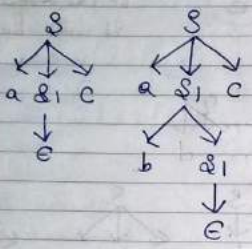


Q7) $L = \{ ab^n c \; ; n \geq 0 \}$

$L = \{ ac, abc, abbc, abbbc \ldots \}$

$S_1 \rightarrow bS_1 / \epsilon$

$S \rightarrow aS_1 c$



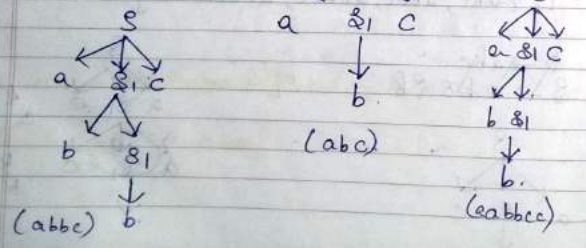
Q8) $L = \{ a^n b^m c^n \; ; n, m \geq 1 \}$

$L = \{ abc, abbc, abbbc \ldots \}$

$b^m, m \geq 1$

$S_1 \rightarrow bS_1 / b$

$S \rightarrow aSc / aS_1 C$

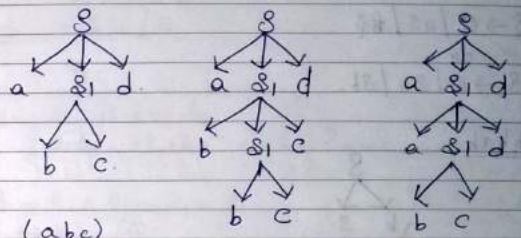


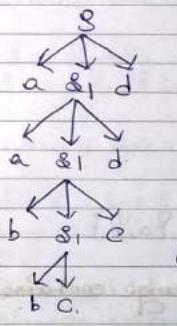
Q9) $L = \{ a^n b^m c^m d^n \; ; n, m \geq 1 \}$

$S \rightarrow aSd / aS_1 d$ (final drawn for $b^m c^m$)

$S_1 \rightarrow bS_1 c / bc$

$L = \{ abcd, abbccd, aabcdd, aabbccdd \ldots \}$





6.09.12

Q Derive CFG from.

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

$(a+b)^*$ or $(a \cup b)^*$

$= \Sigma^*$

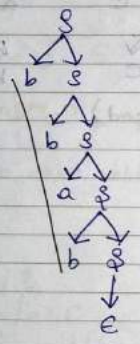
$L = \{ \epsilon, a, b, aa, ab, ba, bb, aaa \ldots \}$

$S \to \epsilon / aS / bS$

or

$S \to \epsilon / Sa / Sb$.

bbab



$\Sigma^+ = S \to aS/bS \quad \Sigma = \{a, b\}$.

Q Derive CFG from the language containing strings of

i) exactly one a, over $\Sigma = \{a, b\}$

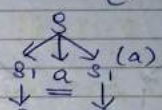
ii) atleast three a's

iii) atleast one a.

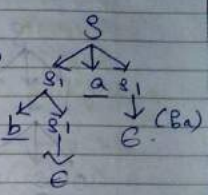
i) $L = \{ a, ab, ba \ldots \}$ (exactly one a)

$S \to S_1 a S_1$

$S_1 \to \epsilon / b S_1$



exactly 2 a's



$S \to S_1 a S_1 a S_1$

$S_1 \to b S_1 / \epsilon$

S

ii) exactly three a's.

$L = \{ aaa, ababa, aaba, baaa, aaab \ldots \}$.

$S \to S_1 a S_1 a S_1 a S_1$

$S_1 \to b S_1 / \epsilon$

iii) atleast one a

$S \to S_1 a S_1$

$S_1 \to a S_1 / b S_1 / \epsilon$

iv) atleast three a's.

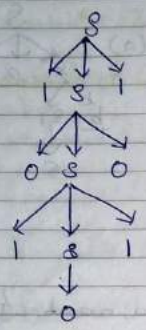
$S \to S_1 a S_1 a S_1 a S_1$

$S_1 \to a S_1 / b S_1 / \epsilon$

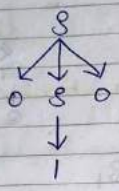
2.4) Q Derive CFG from the language which contains e) strings where every string is palindrome.

Soln. $L = \{ \epsilon, 0, 1, 00, 11, 101, 010 \ldots \}$

1010101



010



101

$S \rightarrow 1S1 / 0S0 / 1 / 0 / \epsilon$

6.09.18

Chapter-3 :-

Ambiguity

Q Show that the particular CFG is ambiguous.

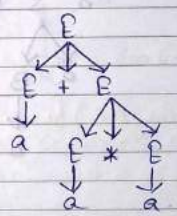
Ambiguous - If a grammar generate the same thing in several different ways (means - a string will have more than one parse tree we can say that the string is derived ambiguously from that grammar. If a grammar generated some string ambiguously. We can say grammar is ambiguous.

Example →

$E \rightarrow E+E / E*E / a$

$2+3*5$

$\Rightarrow 2+15 = 17$



$a+a*a$

→ If every generated string from the grammar has a unique parse tree then we can say the grammar is unambiguous.

→ If a grammar is ambiguous then the string has 2 different phase trees, not 2 different derivation

Example →

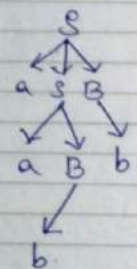
$S \to aSB$

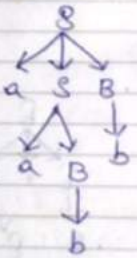
$S \to aB$

$B \to b$

aabb

left most derivation	right most derivation
$S \to a\underline{S}B$	$S \to aS\underline{B}$
$\to aa\underline{B}B$	$\to aS\underline{b}$
$\to aab\underline{B}$	$\to a\underline{a}Bb$
$\to aabb$	$\to aabb$





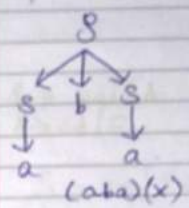
(aabb)

Inherently Ambiguous → A string w is derived ambiguously in context free grammar G, if it has 2 or more diff left most derivation / right most derivation. Sometimes some CFL can be generated only by ambiguous grammars.

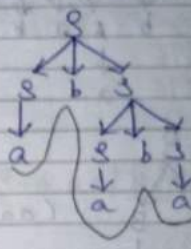
Q Show that the particular CFG is ambiguous

$S \to SbS/a$

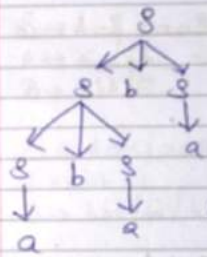
$L = \{ a, aba, ababa \ldots \}$



(aba)(x)



(ababa) ✓



(ababa) ✓

→ This particular grammar shows 2 parse tree that's why the CFG is ambiguous.

8.09.18

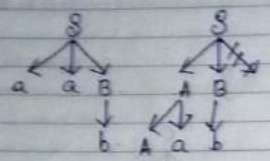
Q Show that the following CFG is ambiguous or not

$S \to AB / aaB$

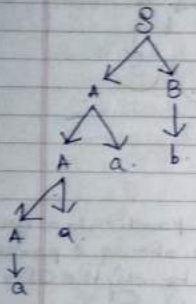
$A \to a / Aa$

$B \to b$

Soln. L = { multiple a and one b }



(aab) → ambiguous



(aaab) ✗

* Elimination of null productions (ϵ) from CFG:-

→ If the start symbol contains ϵ then it cannot be eliminated.

S → aA/ϵ
A → b

S → aA/B
B → ϵ

Cannot be deleted.

1) S → aA
A → b/ϵ

L = { a, b, ab }

↳ S → aA
A → b

From where we remove other place where A is present add ϵ and have merged aA ϵ → a

2) S → as/bA
A → aA/ϵ

S → as/bA/ϵ ⇒ S → as/b/bA
A → aA

3) S → AaB/aaB
A → ϵ/bb
B → bbA/ϵ

To Remove: A → ϵ (i)
B → ϵ (ii)

S → AaB/aaB/aB
A → bb
B → bbA/ϵ/bb

Removing ϵ from A (all A present anywhere)

S → AaB/aaB/Aa/aa/aB/a
A → bb
B → bbA/bb/

4) S → ABAC
A → aA/ϵ
B → bB/ϵ
C → c

S → ABAC/BAC/ABC/AC
A → aA
B → bB/ϵ
C → c

from A

$S \to ABAC / BC / AAC / C / BAC / ABC / AC /$
$A \to aA / a$
$B \to bB / b$
$C \to c$

5) $S \to AbaC$
$A \to BC$
$B \to b / \epsilon$
$C \to D / \epsilon$
$D \to d$

From B

$S \to AbaC$
$A \to BC / C$
$B \to b$
$C \to D / \epsilon$
$D \to d$

From C

$S \to AbaC / Aba$
$A \to BC / C / B / \epsilon$
$B \to b$
$C \to D$
$D \to d$

Now again from A

$S \to AbaC / Aba / baC / ba$
$A \to BC / C / B$
$B \to b$
$C \to D$
$D \to d$

* Elimination of unit production from CFG :-

1) $S \to Aa / B$ (i) $S \to B$
$B \to A / bb$ (ii) $B \to A$
$A \to a / bc / B$ (iii) $A \to B$
} To Remove

i) To remove B ($S \to B$)

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

S S S
↓ ↓ ↓
B B B
↓ ↓ ↓
A A bb.
↓ ↓
a bc

ii) To remove A ($B \to A$)

$S \to Aa / a / bc / bb$.
$B \to bb / a / bc$.
$A \to a / bc / B$.

B B
↓ ↓
A A
↓ ↓
a bc.

iii) To remove B ($A \to B$)

$S \to Aa / a / bc / bb$
$B \to bb / a / bc$
$A \to a / bc / bb$.

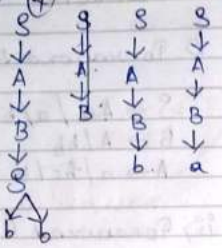
A A A
↓ ↓ ↓
B B B
↓ ↓ ↓
bb a bc

Repeated so will not be written

2) $S \to A/bb$ $S \to A$
$A \to B/b$ $A \to B$
$B \to S/a$ $B \to S$

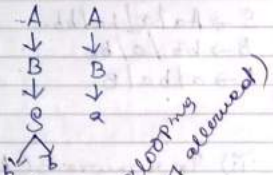
i) To remove A ($S \to A$)

$S \to bb/S/B/b/a/$
$A \to B/b$
$B \to S/a$



ii) To remove B ($A \to B$).

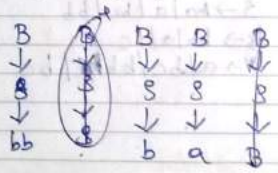
$S \to bb/S/A/b/a$
$A \to b/S/a/bb$
$B \to S/a$



iii) To remove S ($B \to S$)

$S \to bb/S/A/b/a$
$A \to b/S/a/bb$
$B \to a/bb/S/b/a/A$

(looping not allowed)



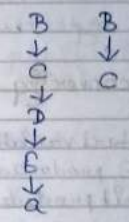
3) $S \to AB$
$A \to a$
$B \to C/E$
$C \to D$
$D \to E$
$E \to a$

To remove

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

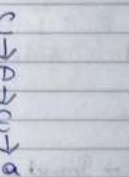
i) To remove C ($B \to C$)

$S \to AB$
$A \to a$
$B \to b/a/$
$C \to D$
$D \to E$
$E \to a$



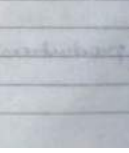
ii) To remove D ($C \to D$).

$S \to AB$
$A \to a$
$B \to b/a$
$C \to a$
$D \to E$
$E \to a$



iii) To remove E ($D \to E$)

$S \to AB$
$A \to a$
$B \to b/a$
$C \to a$
$D \to a$



12/9/18

* Chomsky Normal Form:-

The CFG is in CNF, every production of grammar is in the form of

$A \to BC$

$A \to a$

Where a is the terminal & ABC are the variables

(single variable) → (single variable) (single variable).

a single v → a single terminal.

Procedure of converting CFG to CNF:-

1. Insert a new start variable into the CFG
2. Remove all the ϵ productions from the grammar
3. Remove all the unit productions from the grammar
4. Convert the remaining productions of the grammar into a proper form by adding additional variables & prod.

Ex-2.10

$S \to ASA \mid aB$

$A \to B \mid S$

$B \to b \mid \epsilon$

1. $S_0 \to S$ } → Insert a new start variable.
   $S \to ASA \mid aB$
   $A \to B \mid S$
   $B \to b \mid \epsilon$

2. Remove all ϵ productions.

($B \to \epsilon$)

$S_0 \to S$

$S \to ASA \mid aB \mid a$

$A \to B \mid S \mid \epsilon$

$B \to b$

($A \to \epsilon$)

$S_0 \to S$

$S \to ASA \mid aB \mid a \mid S \mid AS \mid SA$

$A \to B \mid S$

$B \to b$

3. Remove all the unit productions.

$S_0 \to S$

$S \to ASA \mid aB \mid a \mid AS \mid SA \mid S$

$A \to B \mid S$

$B \to b$

$S_0 \to S$, $S \to S$, $A \to B$, $A \to S$

i) $S \to S$ (remove).

$S_0 \to S$

$S \to ASA \mid aB \mid a \mid AS \mid SA$

$A \to B \mid S$

$B \to b$

ii) $S_0 \to S$ (remove)

$S_0 \to ASA \mid aB \mid a \mid AS \mid SA$

$S \to ASA \mid aB \mid a \mid AS \mid SA$

$A \to B \mid S$

$B \to b$

iii) A → S (remove)

S₀ → ASA/aB/a/SA/AS
S → ASA/aB/a/SA/AS
A → S/ε
B → b

A → S
S → B
B → b

iv) A → S (remove)

S₀ → ASA/aB/a/SA/AS
S → ASA/aB/a/SA/AS
A → b/ASA/aB/a/SA/AS
B → b

A → S
S → ε
ASA/aB/a/AS/SA

4. S₀ → AS/SA/a [V → V.V]
S → AS/SA/a [V → V.V and V → T]
A → SA/AS/a [V → V.V and V → P]
B → b [V → P]

And for remaining one we can add certain things for making them all into single format

X → a

S₀ → ~~AS~~/XB/AS/SA
S → ~~PSA~~/XB/SA/AS
A → SA/AS/a
B → b

P → AS

S₀ → AS/SA/a/XB/PA
S → AS/SA/a/XB/PA
A → b/AS/SA/a/PA/XB
B → b

Q2) S → bA/aB
A → bAA/aS/a
B → aBB/bS/a

1) Add new start variable

S₀ → S
S → bA/aB
A → bAA/aS/a
B → aBB/bS/a

2. No ε production

3. Remove unit productions

S₀ → S (remove)

S₀ → bA/aB
S → bA/aB
A → bAA/aS/a
B → aBB/bS/a

4. Making into proper form,

X → a

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

let $X \to a$

$S_0 \to XB$
$S \to XB$
$A \to a / XS$
$B \to a$

let $P \to b$

$S_0 \to PA / XB$
$S \to XB / PA$
$A \to XS / a$
$B \to PS / a$

let $A \to Y \to AA$ $Z \to BB$

$S_0 \to PA / XB$
$S \to XB / PA$
$A \to PY / XS / a$
$B \to PS / a / XZ$
$X \to a$
$P \to b$
$Y \to AA$
$Z \to BB$

Q3) $S \to a$
$S \to b$
$S \to CSS$

Q4) $S \to 1A / 0B$
$A \to 1AA / 0S / 0$
$B \to 0BB / 1$

Soln 3)

1. Insert a new start

$S_0 \to S$
$S \to a$
$S \to b$
$S \to cSS$

2. No ϵ production

3. Remove unit productions.

($S_0 \to S$) remove

$S_0 \to a / b / cSS$
$S \to a$
$S \to b$
$S \to cSS$

4. Making into proper format.

$S_0 \to a / b / XY$
$S \to a$
$S \to b$
$S \to XY$
$X \to a$ ~~$\to$~~ c.
$Y \to SS$

Soln 4)

1. Unreal non variable

$S_0 \to S$
$S \to 1A/0B$
$A \to 1AA/0S/0$
$B \to 0BB/1$

2. No ϵ productions

3. Remove unit productions.

($S_0 \to S$) remove.

$S_0 \to 1A/0B$
$S \to 1A/0B$
$A \to 1AA/0S/0$
$B \to 0BB/1$

4. Making into proper format

$S_0 \to YA/XB$
$S \to YA/XB$
$A \to YP/XS/0$
$B \to XQ/1$
$X \to 0$
$Y \to 1$
$P \to AA$
$Q \to BB$

15.09.18 Saturday

Push Down Automata :-

Formal Definition :-

A PDA is a 6 tuple :- $(Q, \Sigma, \Gamma, \delta, q_0, F)$

Q = set of total no of states
Σ = i/p alphabet
Γ = stack alphabet
$\delta = Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to P(Q \times \Gamma_\epsilon)$ is the transition funtion
$q_0 \in Q$ is the initial state
$F \subseteq Q$ is the set of final states.

RL → FA (NFA/DFA)
CFL → PDA (FA + stack)

RL ⊂ CFL

FA (NFA/DFA)	PDA
i) To accept a regular lang. we use finite automata.	To accept a context free lang we use PDA. (FA + stack)
ii) ~~By giving an input symbol~~	Our aim is to reach double circle & to make sure the stack is empty.
iii) In FA our goal is to reach the double circle	

iii) $q_1 \xrightarrow{a} q_2$

By giving input symbol a we reach from q_1 to q_2

$q_1 \xrightarrow{a, b \to c} q_2$ (replaced ↳ top of stack symbol)

By giving input symbol a we reach from state q_1 to q_2 & top of stack which is b is replaced with symbol c

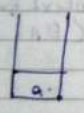
* Push Down Automata (PDA) :-

1. $q_1 \xrightarrow{a, \epsilon \to \$} q_2$

(Push $) [C is empty stack]

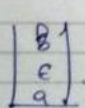
Stack is changed from q_1 to q_2 & top of stack is empty (ϵ) is replaced with symbol $.

2. $q_1 \xrightarrow{a, \epsilon \to a} q_2$



a is pushed by giving symbol a.

3. $q_1 \xrightarrow{a, \epsilon \to a} q_2 \xrightarrow{b, \epsilon \to b} q_3$

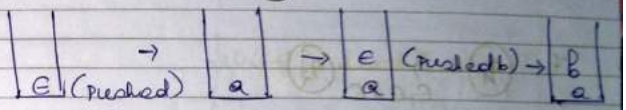


→ b is pushed

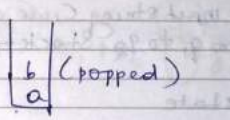
4. $q_1 \xrightarrow{a, \epsilon \to a} q_2$, $q_2 \xrightarrow{b, \epsilon \to b}$

a, b, c

All will be scanned q_3, $q_3 \xrightarrow{c, b \to \epsilon}$

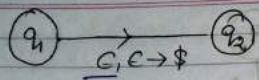


ϵ (pushed) → a → ϵ a (pushed b) → b a

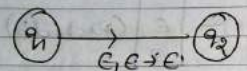


b a (popped)

5. $q_1 \xrightarrow{a, \epsilon \to \$} q_2 \xrightarrow{b, \$ \to \epsilon} q_3$

6. 

By giving input string ϵ we can skip from q_1 to q_2 without any symbol but the top of stack i.e dirty so replaced with $.

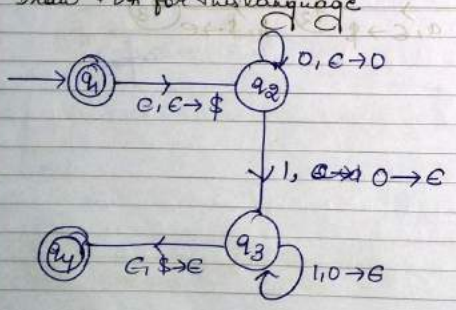
7. 

Empty stack. By giving input string ϵ, we can skip from q_1 to q_2. Stack → unchanged

Used for making dead state

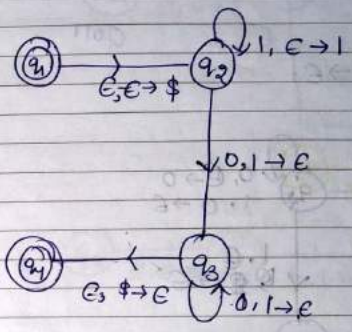
Ex 2.14 $L = \{ 0^n 1^n, n \geq 0 \}$

Draw PDA for this language

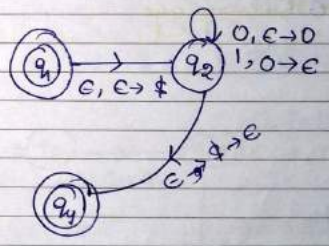


Q2) $L = \{ 1^n 0^n, n \geq 0 \}$.

$L = \{ \epsilon, 10, 1100, 111000 \ldots \}$.



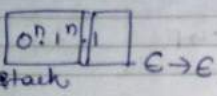
Q) $0^n 1^n$ · $0^n 1^n 0^n$



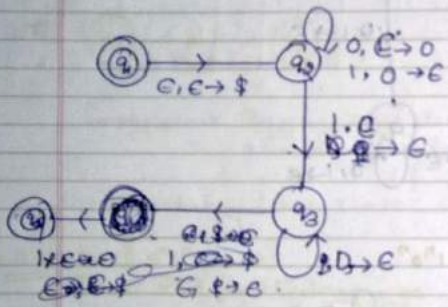
15.09.18

Q4) $L = \{0^n 1^{n+1}, n \geq 1\}$

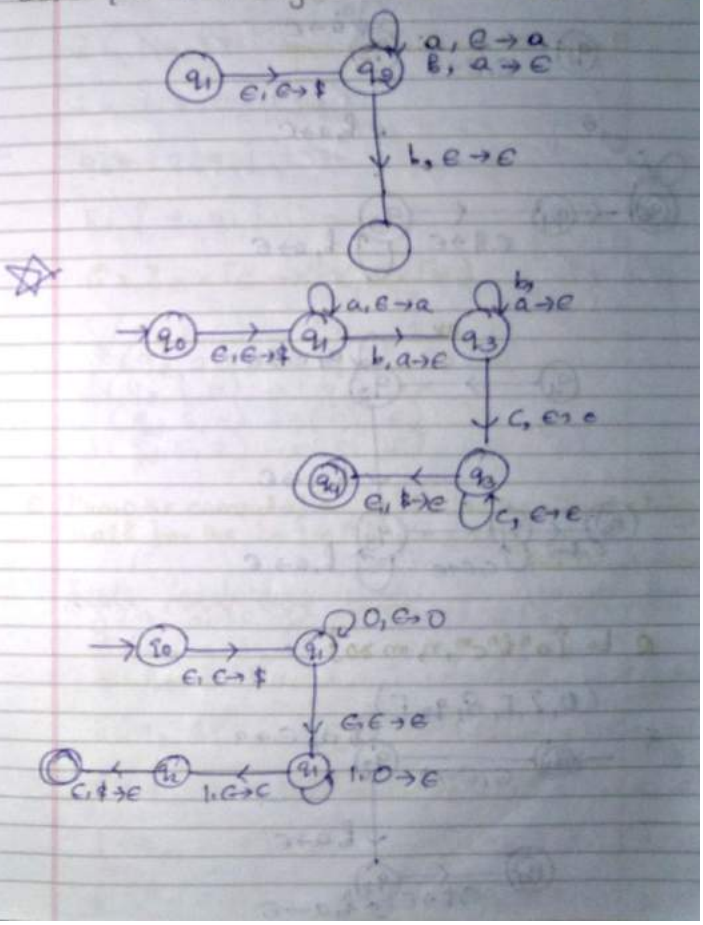
Draw PDA for this language.



0011

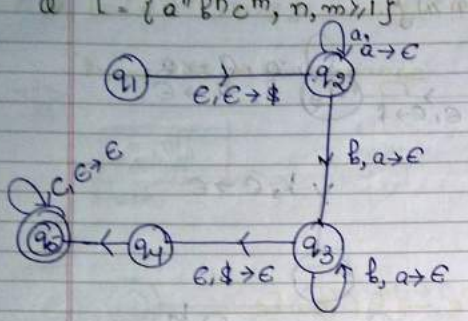


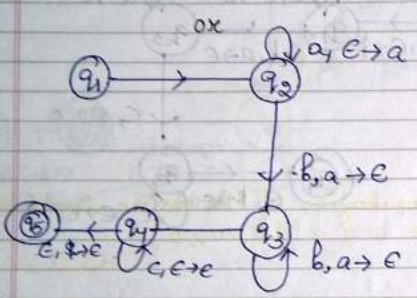
Q5) $L = \{a^n b^n c^m, n \geq 1\}$



Monday
18.09.18

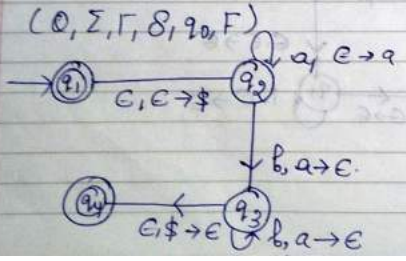
Q $L = \{a^n b^n c^m, n, m \geq 1\}$





Q $L = \{a^n b^n c^m, n, m \geq 0\}$

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



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

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

$\Gamma = \{\$, a\}$ (push)

$q_0 = \{q_1, q_4\}$

$F = \{q_1, q_4\}$

$Q \times \Sigma_\epsilon \times \Gamma_\epsilon \rightarrow P(Q \times \Gamma_\epsilon)$

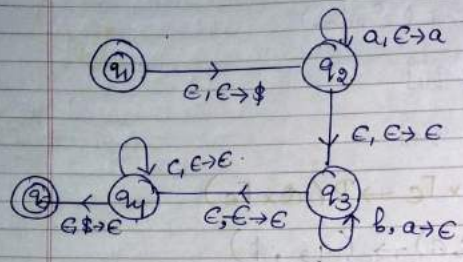
$\delta(q_1, \epsilon, \epsilon) \rightarrow (q_2, \$)$
$\delta(q_2, a, \epsilon) \rightarrow (q_2, a$ (push))
$\delta(q_2, b, a) \rightarrow (q_3, \epsilon)$
$\delta(q_3, b, a) \rightarrow (q_3, \epsilon)$
$\delta(q_3, \epsilon, \$) \rightarrow (q_4, \epsilon)$

Q Draw the computation of PDA for the input string aabb for the $L = \{a^n b^n, n \geq 0\}$ (this is not req)

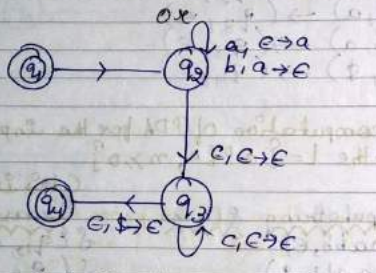
State	Input string	Stack	Transition func.
$\delta(q_1, aabb, \epsilon)$			$\delta(q_1, \epsilon, \epsilon)$
$\delta(q_2, aabb, \$)$			$\delta(q_2, \$)$
$\delta(q_2, abb, a\$)$			$\delta(q_2, a, \epsilon)$ (q_2, a)
$\delta(q_2, bb, aa\$)$			$\delta(q_2, a, \epsilon) \rightarrow (q_2, a)$
$\delta(q_3, b, a\$)$			$\delta(q_2, b, a) \rightarrow (q_3, \epsilon)$
$\delta(q_3, \epsilon, \$)$			$\delta(q_3, b, a) \rightarrow (q_3, \epsilon)$
$\delta(q_4, \epsilon, \epsilon)$			$\delta(q_3, \epsilon, \$) \rightarrow (q_4, \epsilon)$

Q $L = \{a^n b^n c^m, n, m \geq 0\}$

$L = \{\epsilon, ab, c, abc, abcc, aabbcc \ldots\}$



or



Ex-2.16 $L = \{a^i b^j c^k / i, j, k \geq 0\}$ $i = j$ or $i = k$

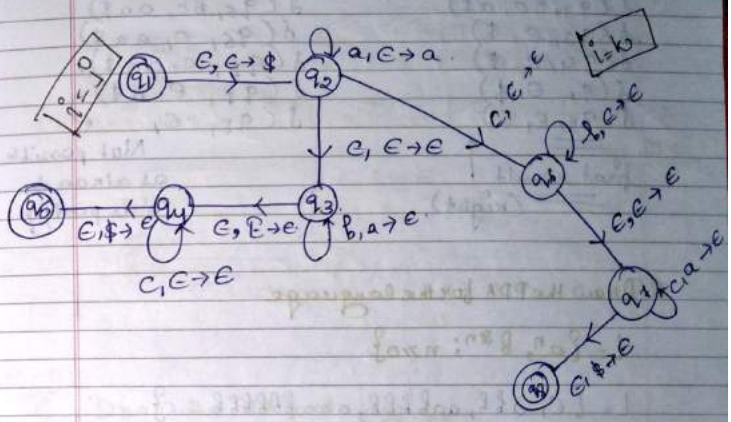
$L = \{a^i, b^j, c^k, i = j\}$

or $L = \{a^i b^j c^k, i = k\}$.

$L = \{a^i b^j c^k, i = k\}$.

$L = \{\epsilon, ab, c, abc, aabbc \ldots\}$

$L = \{\epsilon, ac, b, abc, aabcc, aaabccc \ldots\}$.



(Wed)
19.09.18

Show the computation of PDA from the language $L = \{a^i b^j c^k / i, j, k \geq 0 \text{ & } i = j \text{ or } i = k\}$

aabbc

Soln.

$\delta(q_1, aabbc, \epsilon)$
$\delta(q_2, aabbc, \$)$
$\delta(q_2, abbc, a\$)$
$\delta(q_2, bbc, aa\$)$

δ(q₃, bbc, aa$)

$\delta(q_3, bbc, aa\$)$
$\delta(q_3, bc, a\$)$
$\delta(q_3, c, \$)$
$\delta(q_4, c, \$)$
$\delta(q_4, \epsilon, \$)$
$\delta(q_5, \epsilon, \epsilon)$.

final result ↑ (right)

$\delta(q_6, bbc, aa\$)$
$\delta(q_6, bc, aa\$)$
$\delta(q_6, c, aa\$)$
$\delta(q_7, c, aa\$)$.
$\delta(q_7, \epsilon, a\$)$
$\delta(q_8, \epsilon,$

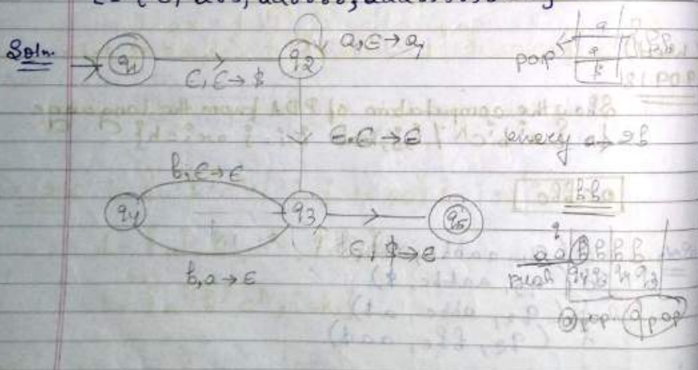
Not possible as already 'a' is pending

Q Draw the PDA for the language

$L = \{a^n, b^{2n};\ n \geq 0\}$.

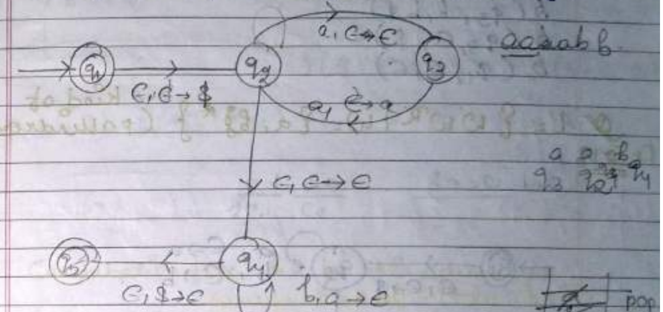
$L = \{\epsilon, abb, aabbbb, aaabbbbbb \ldots\}$

Soln.



Q $L = \{a^{2n} b^n \mid n \geq 0\}$.

$L = \{\epsilon, aab, aaaabb, aaaaaabbb \ldots\}$.

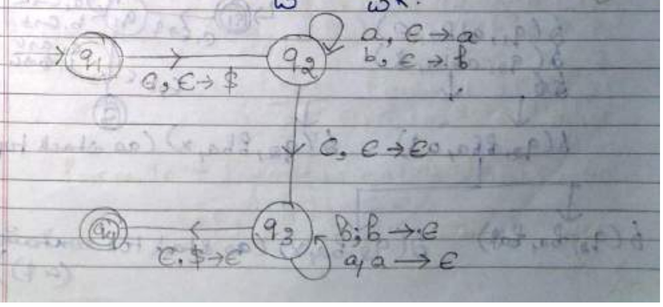


20.09.18

Q Draw the PDA for the language

$L = \{wcw^R \mid w = \{a, b\}^*\}$. (Kind of palindrome)
$L = \{abcba\}$.

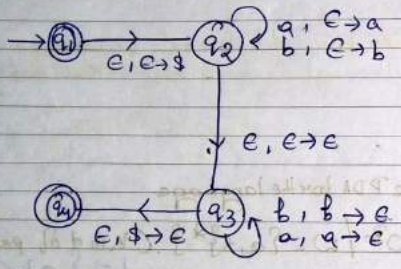
let $w = ab$, so, $\underset{w}{ab}\ c\ \underset{w^R}{ba}$.



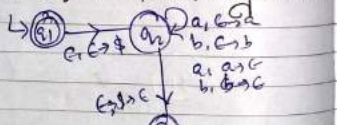
bcb → $\delta(q_1, bcb, \epsilon)$
$\delta(q_2, bcb, \$)$
$\delta(q_2, cb, b\$)$
$\delta(q_3, b, b\$)$
$\delta(q_3, \epsilon, \$)$
$\delta(q_4, \epsilon, \epsilon)$

Ex-2 Q $L = \{ w w^R \mid w = \{a, b\}^* \}$ (palindrome) kind of

let $w = ab$ $\quad \frac{ab}{w} \; \frac{ba}{w^R}$



Q Show the computation of PDA for input string abba



$\delta(q_1, abba, \epsilon)$
$\delta(q_2, abba, \$)$
$\delta(q_2, bba, a\$)$ $\quad \delta(q_2, bba, x)$ (as stack top → $)
$\delta(q_2, ba, ba\$)$ $\quad \delta(q_2, ba, x)$ (as stack top contains (a $))

$\delta(q_2, a, bba\$)$ $\quad \delta(q_2, a, a\$)$
$\delta(q_2, \epsilon, abba\$)$ ✗ $\quad \delta(q_2, \epsilon, x)$ ✗
$\delta(q_2, \epsilon, aa\$)$ ✗ $\quad \delta(q_2, \epsilon, \$)$
$\delta(q_4, \epsilon, \epsilon)$

← Q Computation for bb for this language.

bb → $\delta(q_1, bb, \epsilon)$
$\delta(q_2, bb, \$)$
$\delta(q_2, b, b\$)$ $\quad \delta(q_2, bb, \$)$
$\delta(q_2, \epsilon, bb\$)$ ✗ $\quad \delta(q_2, b, b\$)$ $\quad \delta(q_3, b, x)$ ✗
$\delta(q_3, \epsilon, \$)$
$\delta(q_4, \epsilon, \epsilon)$.

classmate

Date
Page

0 L = {strings of palindrome nature}

$\delta(q_1, aba, \epsilon)$
$\delta(q_2, aba, \$)$

$\delta(q_2, ba, a\$)$ $\quad$ $\delta(q_3, ba, \$)$

$\delta(q_2, a, ba\$)$ $q_3\{a, a\$)$ $\quad$ $\delta(q_3, a, x)$
(x)

22.09.18 (Saturday)

1) $\{a^n b^n, n>0, m>1\}$

2) $\{a^n b^n c, n>0\}$
3) $\{a^n b^n c^m \mid n, m>0\}$
4) $\{a^n c b^n \mid n>0\}$
5) $\{a^n b^{n+1}, n>0\}$
6) $\{a^n c^m b^n \mid n, m>0\}$
7) $\{a^n b^{n+m} \mid n, m>0\}$
8) $\{a^{n+1} b^n, n>0\}$
9) $\{a^{n+m} b^n \mid n, m>0\}$
10) $\{a^{n+1} b^{n+2}, n>0\}$

11) $L = \{a^n b^n c^m \mid n, m>0\}$ or $\{b^n a^n, n>0\}$

12) $L = \{w c w^R, w = \{a, b\}^*\}$

last year

(All done)

13) $L = \{w w^R, w = \{a, b\}^*\}$

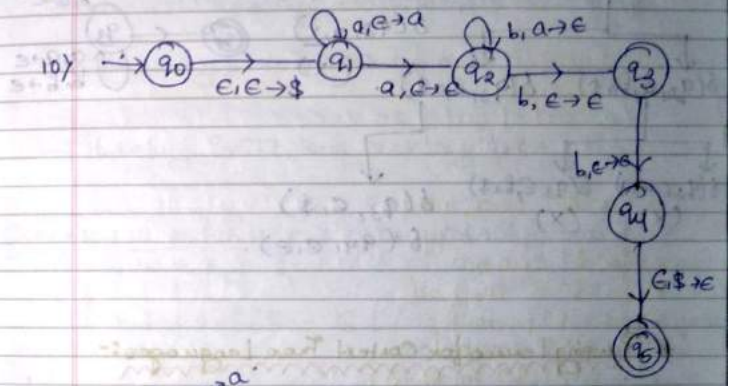
14) L = {palindrome strings}

15) $L = \{a^{2n} b^n, n>0\}$

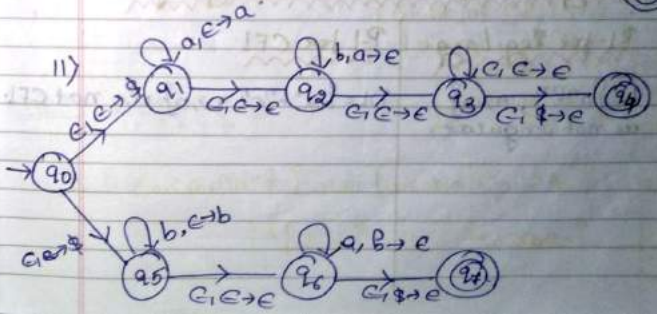
16) $L = \{a^n b^{2n}, n>0\}$

17) $L = \{a^{2n+1} b^{n+3}, n>0\}$.

10)



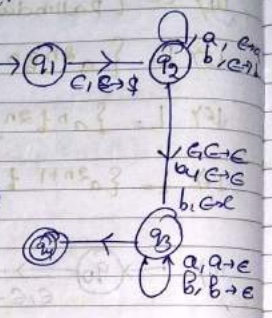
11)



Q Show computation for string aba for lang containing palindrome string.

Soln.

$\delta(q_1, aba, \epsilon)$
$\delta(q_2, aba, \$)$



$\delta(q_2, ba, a\$)$ $\delta(q_3, ba, \$)$
$\delta(q_3, a, X)$
(X)

$\delta(q_2, a, ba\$)$ $\delta(q_3, a, a\$)$

$\delta(q_2, \epsilon, aba\$)$ (X) $\delta(q_3, \epsilon, ba\$)$ (X)

$\delta(q_3, \epsilon, \$)$
$\delta(q_4, \epsilon, \epsilon)$.

* Pumping Lemma for Context Free Languages:-

P.L for Reg. Lang.	P.L for CFL
1) $L = \{a^n b^n, n>0\}$ is not regular.	$DL = \{a^n b^n c^n, n>0\}$ is not CFL.

Lets assume $a^n b^n c^n$ is a CFL, so this lang will be accepted by a PDA with p no. of steps (Pumping length (P)=3).

lets assume p=3.
Choose the string from the lang whose length is atleast p. $L = \{\epsilon, abc, aabbcc..\}$

→ lets choose the string "aabbcc". $|s| = 6 > 3$
→ Now split the string into 5 pieces (u v x y z)

→ Split uvxyz in such a way that i) $|vy| > 0$
ii) $|vxy| < p$

→ If the lang. is CFL, then $uv^i xy^i z \in L$; $i \geq 0$

Case 1: $\underline{a}\ \underline{a}\ \underline{b}\ \underline{b}\ \underline{c}\ \underline{c}$ } should satisfy conditions (in red)
$u\ v\ x\ y\ z$

$i=2$; aaaabbbcc ∉ L (as there should be equal no. of a, b & c, but here c is only 2 whereas a & b are three)

Case 2: $\underline{aa}\ \underline{bb}\ \underline{cc}$
$u\ v\ x\ y\ z$

$i=2$ aaabbbccc ∉ L (No equal no of abc).

Case 3: a a b b c c here p=4.

u v x y z

|vxy| ≤ 4

aaa bb ccc ∉ L

Case 4: a a b b c c here p=4.

i=2 |vxy| ≤ 4

aa b a bb ccc ∉ L

(as out of order so does not belong to this language)

For every case of v & y uv^ixy^iz does not belong to the lang. using the method of contradiction so, we can assume that lang. is not accepted.

Theorem :-

If 'd' is a cfl there is a no. p (pl) where, if 's' is any string in 'z' of length atleast p, then s may be divided into 5 parts s = uvxyz satisfying the following condition :-

i) $i \geq 0$
ii) $uv^ixy^iz \in a$
iii) $|vy| > 0$
iv) $|vxy| \leq p$ (pumping length)

Ex-2.36 Use the pumping lemma to show that the lang. $L = \{a^n b^n c^n ; n \geq 0\}$ is not context free.

We assume that the lang. L is a CFL (using the method of contradiction)

Let p be the pumping length for lang. L that is guaranteed to exist by the pumping length & let the string $s = a^p b^p c^p$. Clearly the string s is member of language L & length of string s is atleast p. Now, divide the string s into (u v x y z) to show that one of the 3 conditions of lemma is violated.

Now consider 2 cases for string s

Case-1 When both v & y contain one type of symbol, or v does not contain the combination of a & b and b & c. In this case the string uv^2xy^2z (i=2) does not contain equal no of a's, b's & c's. Therefore the string s cannot be a member of the lang. L. This violates the condition of lemma & represents contradiction.

Case 2 When v or y contain more than one kind of symbol. In this case the string uv^2xy^2z (i=2) but it was out of order. Hence, the string s cannot be a member of L. And this violates the condition 1 of PL (Pumping lemma) (Represents contradiction)

So, the assumption that lang. L is a CFL is false. In this lang. this is not CFL.

(Monday)
24.09.18

Ex-237) Using Pumping Lemma show that the language $L = \{a^i b^j c^k / 0 \le i \le j \le k\}$ is not CFL

Soln. $L = \{\epsilon, abc, bc, c, aabbcc, abbcc, abcc \ldots\}$.

Let $p = 5$

Choose the string from the lang whose length is atleast p

→ Lets choose the string "aabbcc" $|s| \ge 5 (p)$

→ Split uvxyz in such a way that $|vy| > 0$ and $|vxy| \le p$

→ If the lang is CFL, then $uv^i xy^i z \in L$; $i \ge 0$.

Case 1: a a b b c c ($i=2$).
u v x y z

$aaabbbcc \notin L$ (as less no of c as compared to a & b).

Case-2 a a b b c c
u v x y z

$i=0$ (pumping down condition) aabbbcc

aabc $\notin L$

Case 3 a a b b c c (condition satisfied)
u v x y z

$uv^i xy^i z$.

$i=2$ aaabbbccc

aababbccc $\notin L$.

Case-4) a a b b c c (condition satisfied)
u v x y z

$i=2$ $aaaabbbccc \notin L$.

For every case of v & y $uv^i xy^i z$ does not belong to the lang by the method of contradiction. So we can assume lang is not accepted.

Q 2.38 $L = \{ww, w \in \{0,1\}^*\}$ is not CFL

$L = \{\epsilon, 00, 11, 0101, 1010, 1001, 1101, 1100 \ldots\}$.
1011101.

Let $p = 5$

Choose the string from the language whose length is atleast p.

lets choose the string "10101" $|s| \ge 5$

Split uvxyz in such a way that it

1010 or 1001 w 1010.

Should satisfy the conditions i) |vy| > 0
ii) |vxy| ≤ p.

→ If the lang. is CFL, then $uv^ixy^iz \in L$; $i \geq 0$

Case 1:- $\frac{1}{u}\ \frac{01}{v}\ \frac{1}{x}\ \frac{0}{y}\ \frac{1}{z}$

i=2 1 0101 1001 ∉ L

Case-2

$\frac{1}{u}\ \frac{0}{v}\ \frac{11}{x}\ \frac{0}{y}\ \frac{1}{z}$

i=2 1 0011 1010 1 ∉ L.

Case-3 $\frac{\epsilon}{u}\ \frac{10}{vxy}\ |\ \frac{10}{z}$. v & y is 1st w.

i=2 0110010 ∉ L.

Case-4. 10 10 v & y is 2nd w.

$\frac{10}{u}\ |\ \frac{1}{v}\ \frac{\epsilon}{x}\ \frac{0}{y}\ \frac{\epsilon}{z}$

i=2 101100 ∉ L

Case-5) v & y is both w.

10 | 10

$\frac{1}{u}\ \frac{0}{v}\ \frac{\epsilon}{x}\ \frac{1}{y}\ \frac{0}{z}$

i=2 100110. ∉ L.

$a^nb^nc^n$, $a^ib^jc^k$ — possible questions

26.09.18

Theoretical Proof

Q) $\{ a^n b^n c^n \mid n \geq 0$
$a^i b^j c^k \mid i,j,k \geq 0$ & $i=j=k$

Cases

V	Y	
a	b	→ Case1
b	c	
c	a	
a	bc	→ Case2
ab	c	

Q) $\{ ww \mid w = \{a,b\}^*$
$\{a,b,c\}^*$
$\{a,b\}^*$

Cases

1) V & Y - in wtw means 1st half of the string.

2) V & Y - in 2nd w means 2nd half of the string

3) V & Y - in both w.

26.09.18

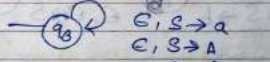
Equivalence with CFG :-

A lang is context free iff some pushdown automation recognizes it.

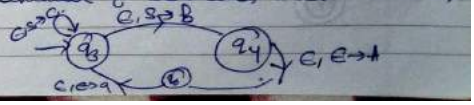
To prove this theorm we need to prove 2 lemma.

Lemma no. 2.21 :- If a lang is context free then some push down automation recognizes it

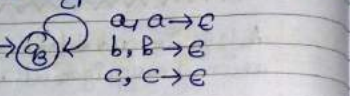
1. Go from initial state q_1 to q_2 with ϵ by pushing a $ to the stack.
2. Go from the stage q_2 to q_3 with ϵ by pushing the start symbol of the grammar into the stack
3. In the state q_3 make every production of grammar a self loop.
4. If the production of the grammar contain or single terminal then. ($S \to A$; $S \to a$, $A \to B$)



5. If the production of a grammar contains combination of grammar & terminal then. $S \to aAb/b$



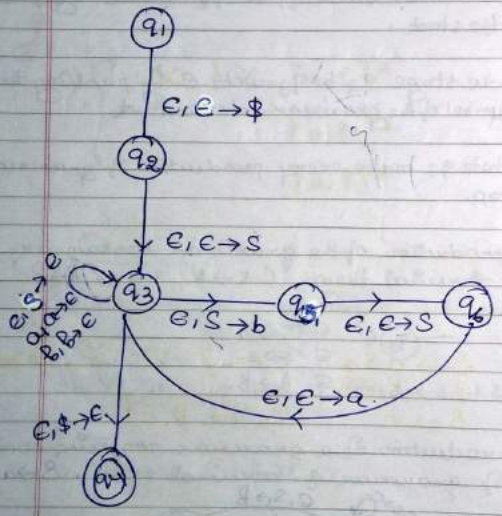
6. Make all the terminals of the grammar in the form of $S \to aAb|c$



Terminals = {a, b, c}
Variables in capital

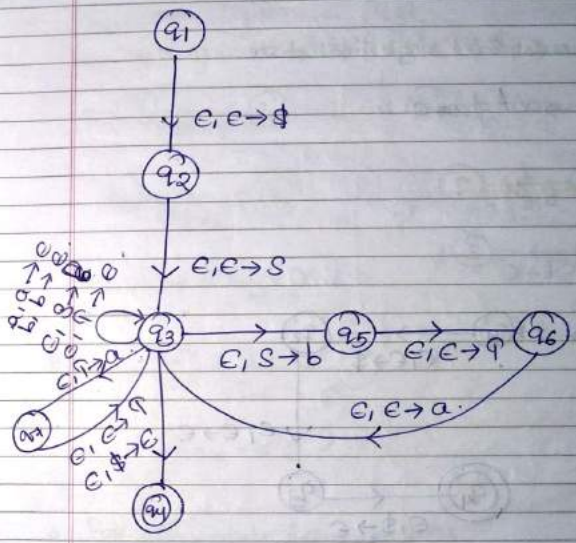
7. Go from q_3 to q_4 (final state) with ϵ by removing the $ from the stack

Ex→ $S \to aSb|\epsilon$ $L = \{a^n b^n ; n \geq 0\}$.



~~Send ed book~~ :-

Ex-2.25 $S \to aTb|b$
$T \to Ta|\epsilon$



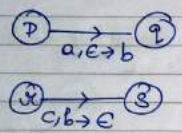
Example :- $S \to aSa$ $L = \{wcw^R ; w \in (a+b)^*\}$
$S \to bSb$.
$S \to c$ $T = \{a, b, c\}$.

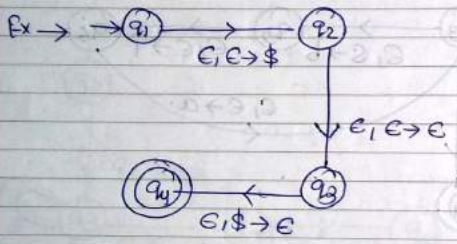
26.09.18

Lemma 2.27 (PDA to CFG)

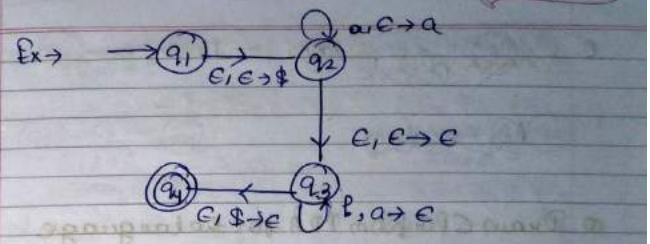
If a pushdown automation recognizes some lang then it is context free

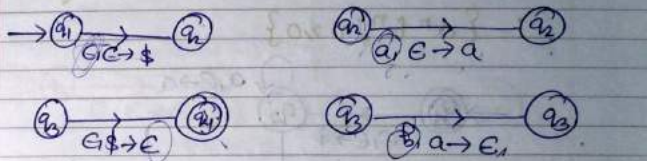
1. PDA must have single final state
2. $A_{ps} \to a A_{qr} c$





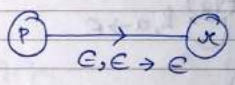
$A_{q_1 q_{2_1}} \to \epsilon A_{q_2 q_3} \epsilon$





$A_{q_1 q_4} \to \epsilon A_{q_2 q_3} \epsilon$

$A_{q_2 q_3} \to a A_{q_2} A_{q_3} b.$

3. 

$A_{px} \to \epsilon$

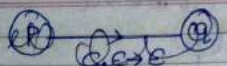
4. For every state belongs to PDA,

$p, q \in Q$

$A_{pp} \to \epsilon$

$A_{qq} \to \epsilon$

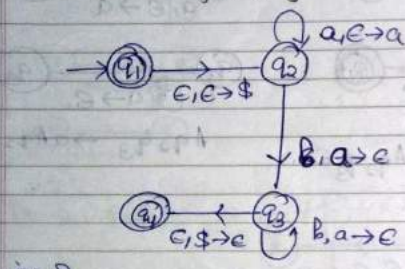
$A_{q_1 q_1}$
$A_{q_2 q_2}$
$A_{q_3 q_3}$
$A_{q_4 q_4}$.

8. 

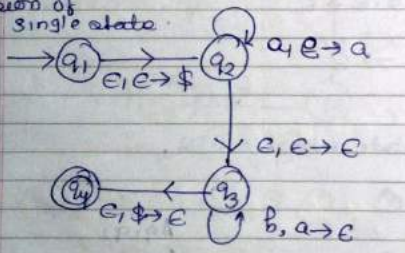
$A_{pq} \rightarrow c A_{rq}$

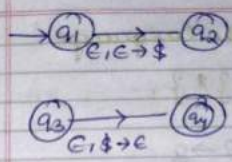
Q Draw CFG from PDA for the language

$L = \{a^n b^n ; n \geq 0\}$



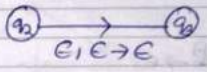
Conversion of q_1 to single state





$A_{q_1 q_4} \rightarrow \varepsilon A_{q_2 q_3} \varepsilon$

$A_{q_2 q_3} \rightarrow a A_{q_2 q_3} b$

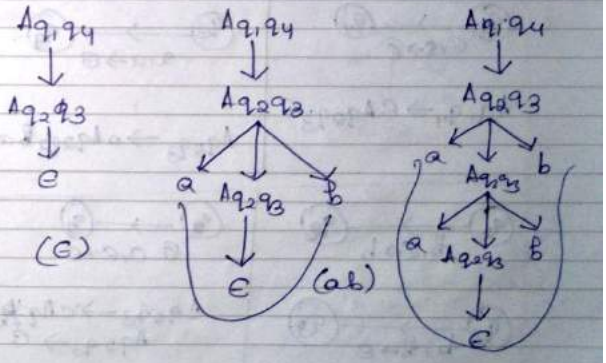


$A_{q_2 q_3} \rightarrow \varepsilon$

$A_{q_1 q_4} \rightarrow A_{q_2 q_3}$

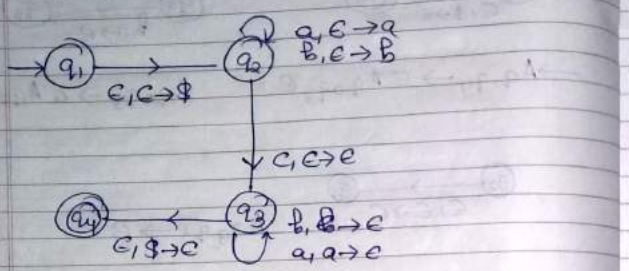
$A_{q_2 q_3} \rightarrow a A_{q_2 q_3} b \mid \varepsilon$

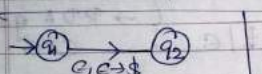
} → PDA got from CFG.

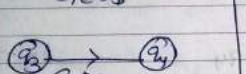


Q. ADA CFG from PDA for the language

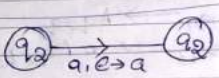
$L = \{ w c w^R \mid w = \{a, b\}^* \}$.



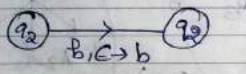


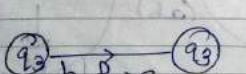


$Aq_1q_4 \rightarrow \varepsilon Aq_2q_3 \varepsilon$

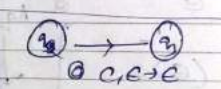


$Aq_2q_3 \rightarrow a Aq_2q_3 a$





$Aq_2q_3 \rightarrow b Aq_2q_3 b$



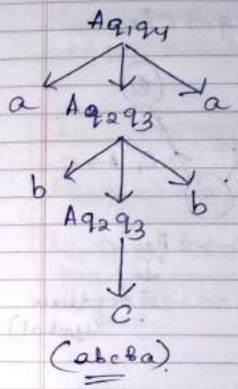
$Aq_2q_3 \rightarrow c$

$Aq_2q_3 \rightarrow C$

aba. abcba.

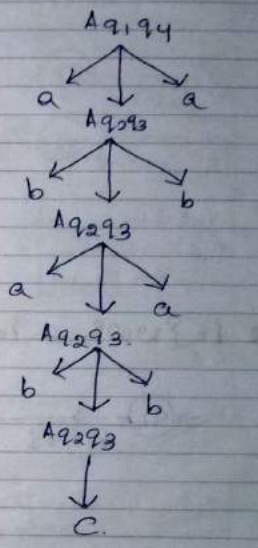
$Aq_1q_4 \rightarrow Aq_2q_3$.

$Aq_2q_3 \rightarrow a Aq_2q_3 a / b Aq_2q_3 b / c$



(abcba)

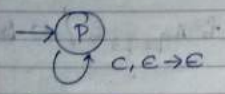
abiaboboaba



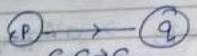
(ababcbaba)

P.T.O

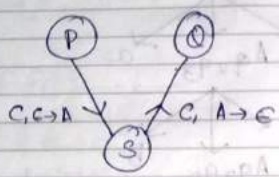
5. Two cases:-



$A_{PP} \rightarrow c\, A_{PP}$



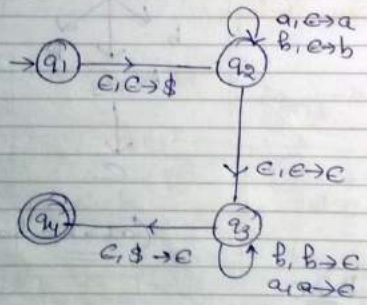
$A_{pq} \rightarrow c\, A_{pq}$

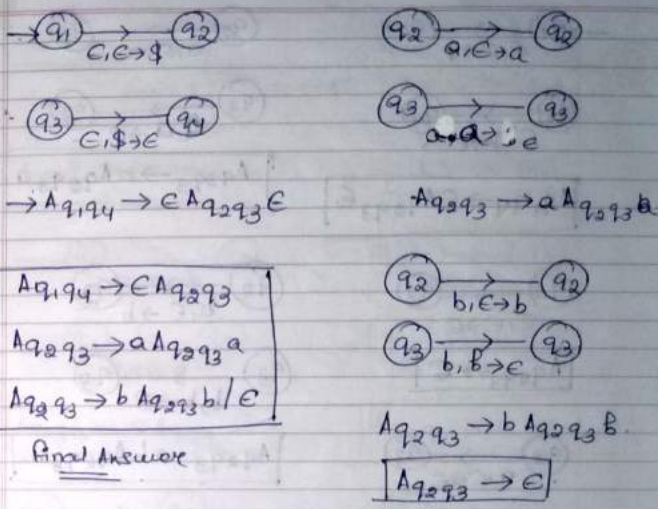


$A_{PS} \rightarrow c\, A_{SS}\, \varepsilon$

ε (to save symbol)

Q $L = \{w w^R, w = \{a, b\}^*\}$





$A_{q_1 q_4} \rightarrow \varepsilon\, A_{q_2 q_3}\, \varepsilon$

$A_{q_2 q_3} \rightarrow a\, A_{q_2 q_3}\, a$

$A_{q_1 q_4} \rightarrow \varepsilon A_{q_2 q_3}$

$A_{q_2 q_3} \rightarrow a A_{q_2 q_3} a$

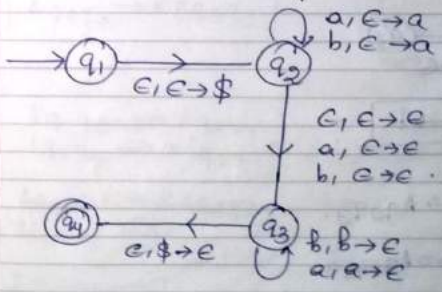
$A_{q_2 q_3} \rightarrow b A_{q_2 q_3} b \mid \varepsilon$

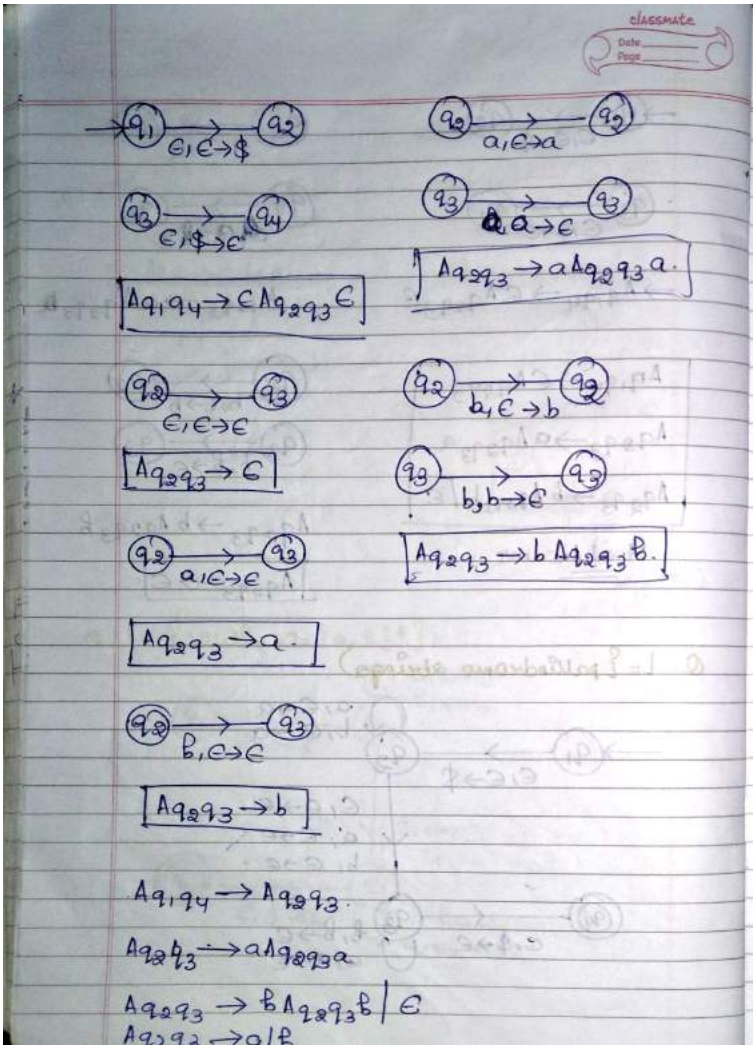
Final Answer

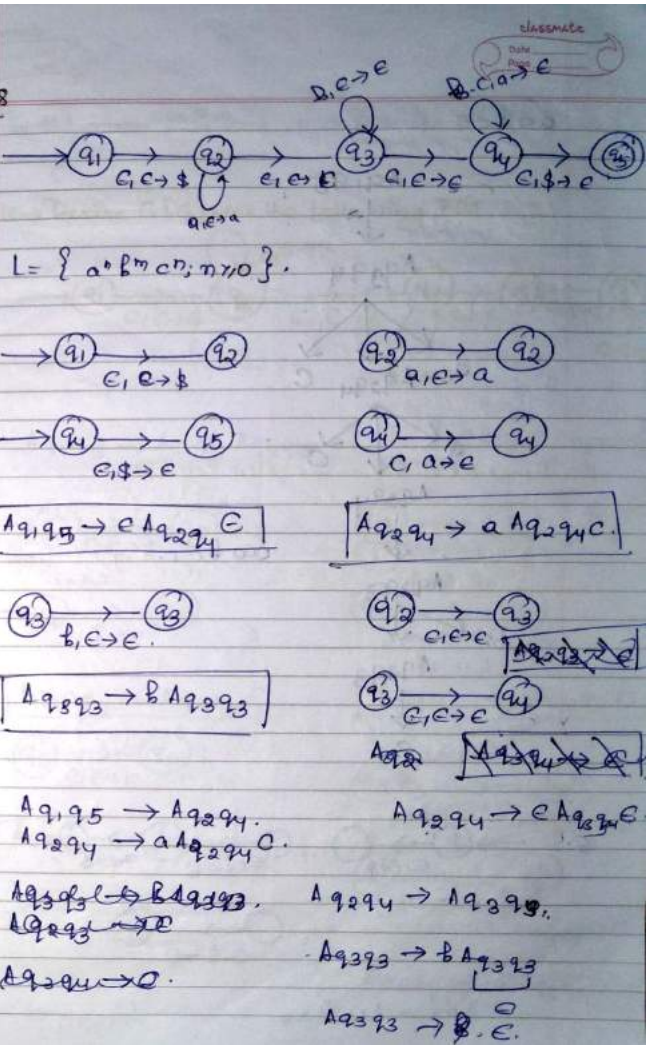
$A_{q_2 q_3} \rightarrow b\, A_{q_2 q_3}\, b$

$A_{q_2 q_3} \rightarrow \varepsilon$

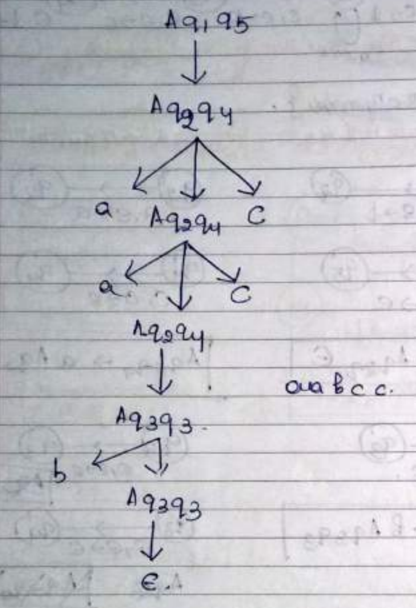
Q L = {palindrome strings}



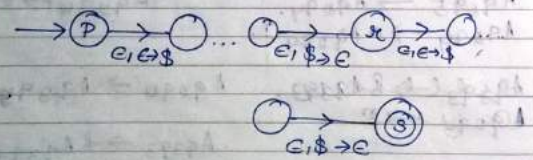




aabcc



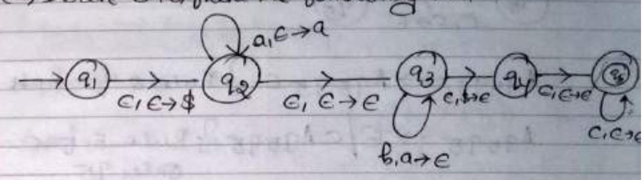
aabcc.

6. 

$A_{ps} = A_{pr} A_{rq}$.

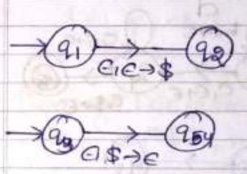
In the three positions p, r & s the stack must be empty

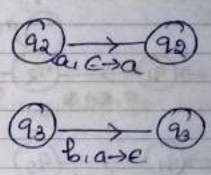
Example → Derive CFG from the following PDA:



$A_{q_1q_5} \rightarrow A_{q_1q_4} A_{q_4q_5}$

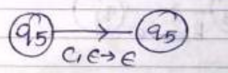
$L = \{a^n b^n c^m ; n, m \geq 0\}$.





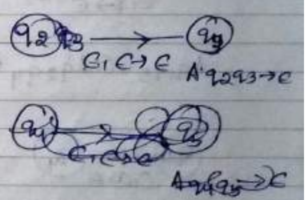
$A_{q_1q_4} \rightarrow \epsilon A_{q_2q_3} \epsilon$

$A_{q_2q_3} \rightarrow a A_{q_2q_3} b$.



$A_{q_5q_5} \rightarrow c A_{q_5q_5}$

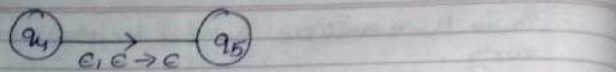
$A_{q_5q_5} \rightarrow \epsilon$

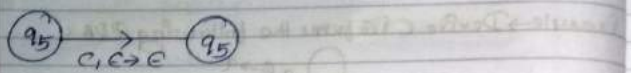


$A_{q_2q_3} \rightarrow \epsilon$

$A_{q_4q_5} \rightarrow \epsilon$

$A_{q_2q_5} \rightarrow \epsilon A_{q_3q_4} \epsilon$

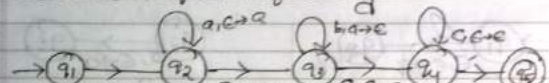


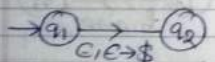


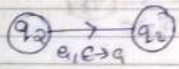
$A_{q_4 q_5} \to A_{q_5 q_5} c$ } Rule a for both

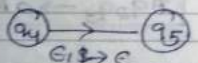
$A_{q_5 q_5} \to \epsilon / c A_{q_5 q_5}$ } Rule b for only q_5

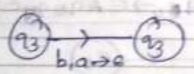
Q Derive the CFG from the following PDA





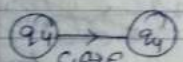


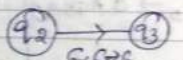


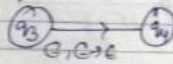


$A_{q_1 q_5} \to \epsilon A_{q_2 q_4} \epsilon$

$A_{q_2 q_3} \to a A_{q_2 q_3} b.$







$A_{q_4 q_4} \to c A_{q_4 q_4}$

$A_{q_4 q_4} \to \epsilon$

$A_{q_2 q_4} \to \epsilon A_{q_2 q_3} \epsilon$

$a^n b^n c^m$ abc aabbcc

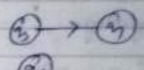
$A_{q_0 q_5}$

$A_{q_0 q_3} \to \epsilon$

$A_{q_0 q_4} \to A_{q_1 q_3} A_{q_3 q_4}$.

$A_{q_2 q_3} \to \epsilon$

$A_{q_3 q_4} \to \epsilon A_{q_4 q_4} c$ }



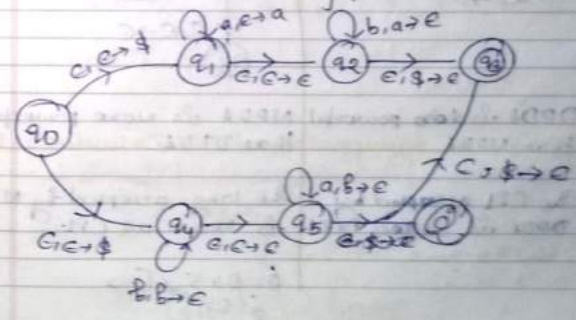
$A_{q_4 q_4} \to c A_{q_4 q_4}$.

$A_{q_4 q_4} \to \epsilon$.

Q Derive CFG from the following PDA for the lang $L = \{ a^n b^n, n \geq 0 \}$

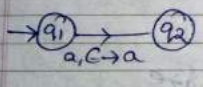
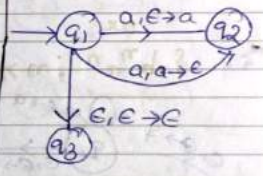
or

$\{ b^n a^n, n \geq 0 \}$.



29.09.18

Diff between Deterministic PDA and Non-deterministic PDA?

DPDA	NPDA (PDA)
1. In this PDA, we know the centre symbol of language. Ex- $L=\{wcw^R; w=\{a,b\}^*\}$ $L=\{a^n b^n; n>0\}$	We do not know the centre symbol of the lang. Ex- $L=\{ww^R; w=\{a,b\}^*\}$
2. We can convert every DPDA into NPDA.	We cannot convert every NPDA to DPDA.
3. There has to be only one move for a state and a symbol 	There can be multiple moves in any situation. 
4. DPDA is less powerful than NPDA	NPDA is more powerful than DPDA
5. The CFL accepted by DPDA is called DCFL	The lang. accepted by NPDA is known as CFL.
	$\delta = Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to$ $2^{(Q \times \Gamma_\epsilon)}$

Note → These languages not accepted by DPDA requires NPDA to accept it

→ All CFL are accepted by NPDA.

DCFL ⊂ CFL.

* Formal defination of DPDA :-

$(Q, \Sigma, \Gamma, \delta, q_0, P)$

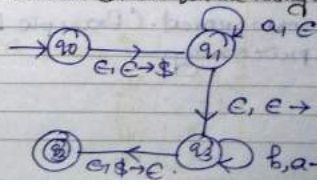
$Q \to$ total no. of states.

$\delta \to Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to (Q \times \Gamma_\epsilon) \cup \phi$

* The transition function (δ) must satisfy the following condition.

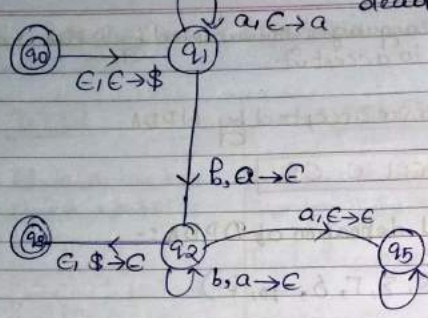
→ For every $q \in Q$, $a \in \Sigma$, then every $b \in \Gamma$, exactly one of the values $\delta(q,a,x)$, $\delta(q,a,\epsilon)$, $\delta(q,\epsilon,x)$ & $\delta(q,\epsilon,\epsilon)$ is not ϕ.

Q Construct a DPDA for the language $L=\{a^n b^n; n>0\}$



From every state a, b should come out so from q2 we will give a to a dead state.

DPDA →



→ q_5 is dead state & in dead state everything comes in self loop.

* Syllabus for midsem :-

Chapter-1

1. Theorem proofs based on proof techniques.
   - two DFA's (Union/Intersection)
   - two NFAs (Union/concatenation)
   - one NFA (Star)
   - NFA to DFA
2. Application of theorem proved. (Example Pake proof)
3. String & lang. processing.
4. NFA to DFA
5. RE to NFA.
6. NFA to RE.
7. Pumping lemma (Regular xy^iz →.
8. Derivations (lang → CFG) (Chapter-2)
9. Parse tree
10. Ambiguity checking.
11. CFG to
    * i) CFL
      ii) FA.
12. CFL to
    * i) CFG
      ii) FA.
13. FA to
    * i) CFG
      ii) CFL.
14. Simplification of CFG (Elimination of null & unit)
15. Chomsky Normal Form.
16. RE to
    i) RL to RE
    ii) RE to RL.

8. Derivations

Derive CFG from $L_1 = \{ a^n b^n c^m ; n, m > 0 \}$.

$L_2 = \{ a^n c^m b^n ; n, m > 0 \}$.

CFG from i) $L_1 \cup L_2$
ii) $L_1 \circ L_2$
iii) $L_2 \circ L_1$
iv) L_1^*
v) L_2^*

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

$L_1 = \{ abc, aabbcc, aaabbbcc \ldots \}$

$S_1 \to \epsilon$ $\quad$ $S_2 \to \epsilon$

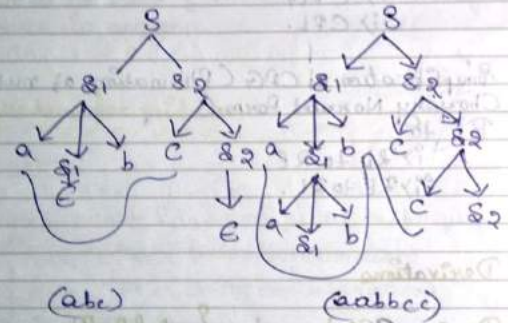
$S_1 \to aS_1b$ $\quad$ $S_2 \to cS_2$.

$L_1 =$ $S \to S_1 S_2$

$S_1 \to \epsilon / aS_1b$

$S_2 \to \epsilon / cS_2$.

$S \to \epsilon / aSb$



(abc) $\quad$ (aabbcc)

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

$L = \{ acb, accb, aaccbb \ldots \}$.

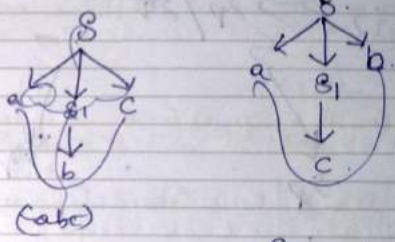
(start symbol)

$S \to S_3$.

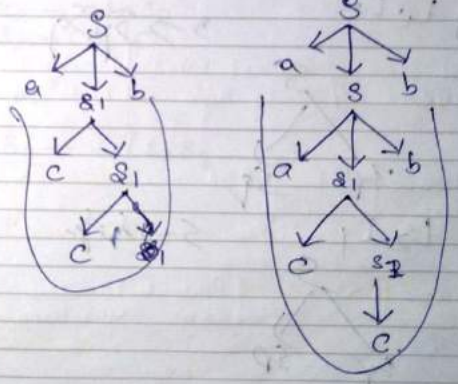
$S_3 \to aS_3b / S_3 / \epsilon$

$S_3 \to B / BS_3 / \epsilon$

$S_2 \to aSb / aS_3b / \epsilon$



(abc) $\quad$ (acb)

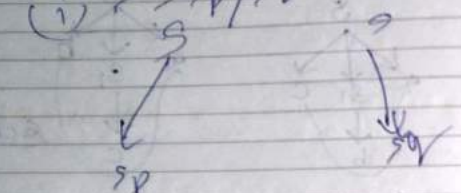


aaccbb.

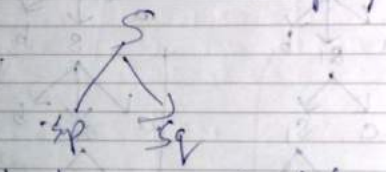
$S \rightarrow$

L_1 $S_p \rightarrow$ L_2 $S_q \rightarrow$

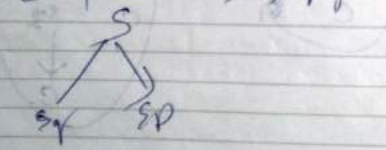
(i) $S \Rightarrow S_p / S_q$



(ii) $L_1 . L_2$ $S \Rightarrow S_p . S_q$



(iii) $L_2 . L_1$ $S \rightarrow S_q . S_p$



1) $(L_1 \cup L_2) =$

$S \rightarrow S_p / S_q$
$S_p \rightarrow S_1 S_2$
$S_q \rightarrow a S_q b / a S_3 b / \epsilon$
$S_1 \rightarrow \epsilon / a S_1 b$
$S_2 \rightarrow \epsilon / c S_2$
$S_3 \rightarrow c S_3 / \epsilon$

2) $(L_1 \circ L_2) =$

$S \rightarrow S_p S_q$
$S_p \rightarrow S_1 S_2$
$S_q \rightarrow a S_q b / a S_3 b / \epsilon$
$S_1 \rightarrow \epsilon / a S_1 b$
$S_2 \rightarrow \epsilon / c S_2$
$S_3 \rightarrow c S_3 / \epsilon$

3) $(L_2 \circ L_1) =$

$S \rightarrow S_q S_p$
$S_p \rightarrow S_1 . S_2$
$S_q \rightarrow a S_q b / a S_3 b / \epsilon$
$S_1 \rightarrow \epsilon / a S_1 b$
$S_2 \rightarrow \epsilon / c S_2$
$S_3 \rightarrow c S_3 / \epsilon$

4) $(L_1^*) =$

$L_1 = q$

$S_p \rightarrow \epsilon / S_1 S / S_2 S_p$

5) $L_2^* =$

$S_q \rightarrow \epsilon / S_3 S_q$

* Closure properties of CFL :-

Theorm 1 The class of CFL is closed under union operation

Theorm 2 The class of CFL is closed under concatenation operation

Theorm 3 The class of CFL is closed under star operation

1.01.10.18

* Chomsky Hierarchy of Grammar / Languages :-

Type-0	Unrestricted grammar	Recursively Enumerable lang.	Turing Machine.
Type-1	Context Sensitive Grammar	Context Sensitive language	Linear Bounded Automation (LBA)
Type-2	Context free grammar	CFL	PDA.
Type-3	Regular Grammar	RL	Finite Automata (FA) (NFA / DFA)

Regular Grammar :-

It can be of 2 types :-

1) Right linear grammar
2) left linear grammar.

Right linear G	left linear Grammar
$A \to xB / y$. (variables always right side) $\to B, A$. $x, y \to$ terminal always single terminal.	$A \to Bx / y$ $A \to Bx$ $A \to y$
$S \to aA$ $S \to b$	$S \to Aa$ $S \to b$.
$S \to B / \epsilon$ $B \to b / cB$	$S \to B / \epsilon$ $\therefore S \to Bc / \epsilon$
$S \to abA$ $A \to bA$	$S \to AaB$ $A \to Ab$

* CFG :-

$A \in \alpha$ where $\alpha \in (V \cup T)^*$ — Any combination of variable & terminal

Ex → $a^n b^n c^m$

$S \rightarrow aSBS_1$
$S_1 \rightarrow cS_1 / \epsilon$

** A regular grammar cannot be both right & left linear

× $S \rightarrow aA$, $A \rightarrow B/B$, $B \rightarrow \epsilon$ — This is not regular grammar but it is CFG.

× $S \rightarrow aAb$, $S \rightarrow \epsilon$ — both side terminal, so it is not RG but CFG.

* CFG to CFL

$S \rightarrow aS_1b$
$S_1 \rightarrow cS_1d / \epsilon$

$L = \{ a c^m d^m b ; m > 0 \}$

* CFL to CFG :-

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

$S \rightarrow aSd / ad / aS_1d$
$S_1 \rightarrow bS_1c / bc / \epsilon$

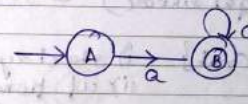
* CFL to FA :-

(RL)

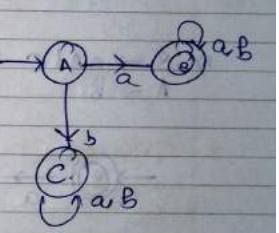
Construct a FA from the following CFL where lang. containing strings starting with a. $\Sigma = \{a, b\}$

$L = \{ a, ab, aab, aba, aaaa, aaa \ldots \}$.

NFA



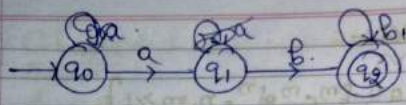
DFA



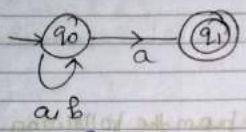
$L = \{ a^n b^m , n, m \geq 1 \}$.

$L = \{ ab, aabb, aabbb, \ldots \}$

CFL	RL
stack $(a^n b^n, wcw^R)$ $(a^n b^n)$	$a^n b^m$. $(ab)^n$.



* FA to CFL :-



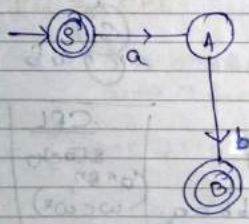
RL = L = { a, aa, aab, aba, abba... }.

* CFG to FA :- (Direct Conversion)
(Without converting to CFL).
(Right linear)

1) S → aA | ∈ L = {a, ab}.
A → b.

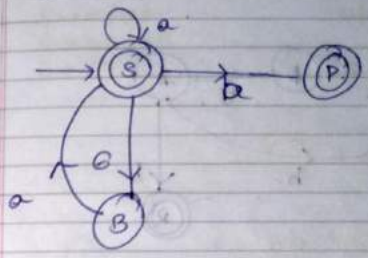
i) start symbol → S
ii) if from any symbol ∈ comes that becomes final

iii) V → single terminal then it will go to double circle ← (Any name)



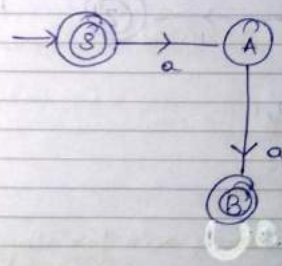
2) S → aS | b | ∈
S → B. or S → ∈B
B → aS

self loop as aS.

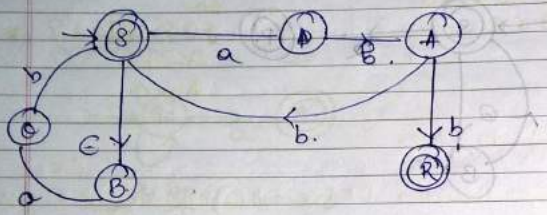


L = { ∈, b, ab, .. }.

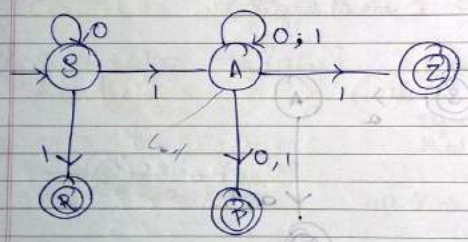
3) S → ∈ | aA
A → aB | bB.
B → ∈



4) $S \to abA/B/baB/\epsilon$
$A \to bS/b$
$B \to aS$.



5) $S \to 0S/1A/1$
$A \to 0A/1A/0/1$



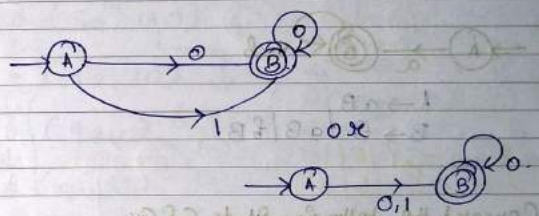
0 & 1 & 2

1) $A \to 0B/1B$
$B \to \epsilon/0B$

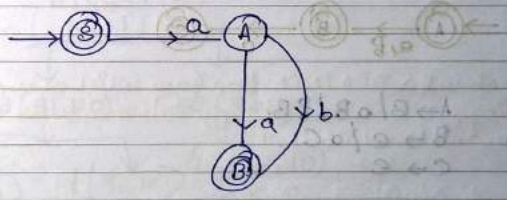
2) $S \to \epsilon/aA$
$A \to aB/bB$
$B \to \epsilon$

3) $S \to AB/\epsilon$
$B \to bB$
$A \to c$
We can convert it to
$S \to cB/\epsilon$
$B \to bB$
$A \to c$.

1)



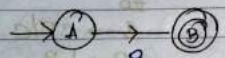
2)



* PA to CFG:- (Direct Conversion)
Always should be right linear

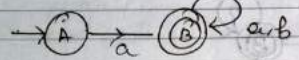
1) 

$A \to \epsilon$

2) 

$A \to aB$

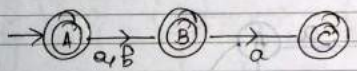
$B \to \epsilon$

3) 

$A \to aB$

$B \to \epsilon / aB / bB$

Q Convert the following FA to CFG:-

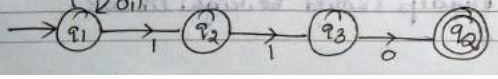


$A \to \epsilon / aB / bB$

$B \to \epsilon / aC$

$C \to \epsilon$

Q Show the computation of NFA for input string 10101110



$\delta(q_1, 10101110)$ $\quad$ $\delta(q_2, 0101110)$

$\delta(q_1, 0101110)$ $\quad$ $\times$

$\delta(q_1, 101110)$ $\quad$ $\delta(q_2, 01110)$

$\delta(q_1, 01110)$ $\quad$ $\times$

$\delta(q_1, 1110)$ $\quad$ $\delta(q_2, 1110)$

$\delta(q_2, 110)$ $\quad$ $\delta(q_3, 110)$

$\delta(q_2, 10)$ $\quad$ $\delta(q_3, 10)$

$\delta(q_2, 0)$ $\quad$ $\delta(q_3, 10)$

$\delta(q_4, \epsilon)$ $\quad$ 0

$\delta(q_1, 1110)$

$\delta(q_1, 110)$ $\quad$ $\delta(q_2, 110)$

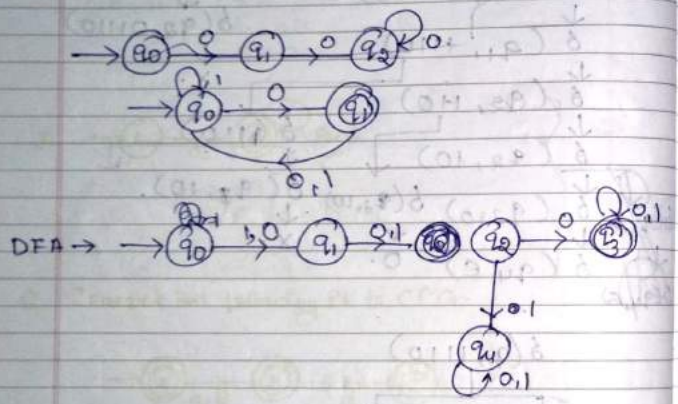
$\delta(q_1, 10)$ $\quad$ $\delta(q_2, 10)$ $\quad$ $\delta(q_3, 10)$

$\delta(q_1, 0)$ $\quad$ $\delta(q_2, 0)$ $\quad$ $\delta(q_3, 0)$

$\delta(q_1, \epsilon)$ $\quad$ $\delta(q_2, \epsilon)$ $\quad$ $\delta(q_4, G)$

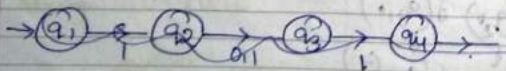
Q Construct a DFA with min no. of states to recognize strings over $\Sigma = \{0,1\}$ where the third symbol from left hand side of the string is 0.

$L = \{000, 0001, 100, 110, 1100, \ldots\}$

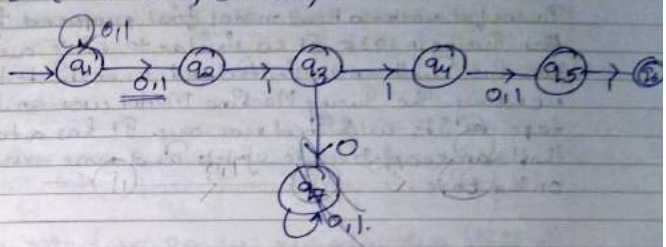


Q Construct a DFA/NFA for a lang which obtain strings where the 1st, 2nd & 4th symbol, 3rd & 5th symbol from left hand side is 1

01 01 01 1 01 1



$L = \{01101, 0111101, 11111\}$



Alfex Nicolson Dusenka.

22.10.18

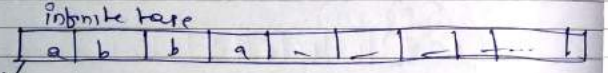
Turing Machine.

Powerful mathematical model first proposed by Alan Turing in 1936. It is similar to finite automation but with an unlimited and unrestricted memory. The Turing Machine Model uses an infinite tape as it's unlimited memory. It has a tape head that can read & write symbols and move around on the tape.

In finite automata we can go only further, cannot go back but in Turing Machine we can go in any direction. And Turing Machine can even read and also write.

Initial the tape contain only the input string.

Ex- L={ abba }



tape head.

Initially the tape contains only input string & is blank everywhere. following summarizes the difference:-

i) A FA can only read the symbol from the string but a Turing machine can both read & write on the tape

ii) The read write head (tape head) of Turing Machine can move both to the left or to the right but FA can read only from left to right.

iii) The tape used in Turing Mathematical Model is infinite.

iv) The special states is infinite for rejecting & accepting take effect immediately.

* Formal Defination :-

It consist of 7 tuples :-

$(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

$Q \rightarrow$ total no. of states used in Turing M/c.
$\Sigma \rightarrow$ input alphabet not containing blank symbols.
$\Gamma \rightarrow$ tape alphabet where $\Sigma \subseteq \Gamma$
(any replaced alphabet or

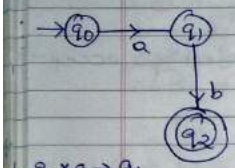
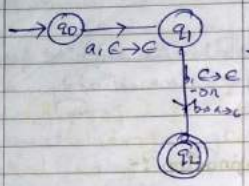
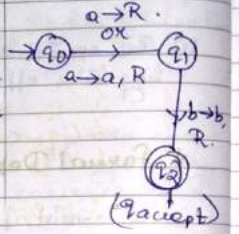
$\delta \rightarrow Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$

L → left, R → right } which direction to move.

$q_0 \rightarrow$ initial state of turing machine.
$q_{accept} \rightarrow$ accept state of the string.
$q_{reject} \rightarrow$ rejected state for the strings.

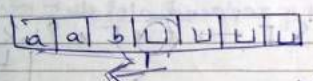
where; $q_{accept} \neq q_{reject}$.

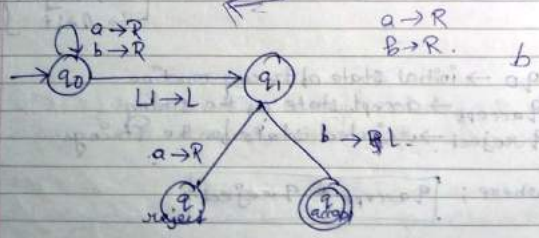
* Difference between transition functions

FA	PDA	TM
$L = \{a, b\}$	$L = \{a, b\}$	$L = \{ab\}$
		
$q_0 \times a \to q_1$ $q_1 \times b \to q_2$		

Q Construct a Turing Machine that accepts the lang. containing string ending with symbol b. over $\Sigma = \{a, b\}$

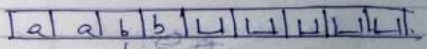
$L = \{ b, bb, bbb, ab, aab, abb, \ldots \}$.

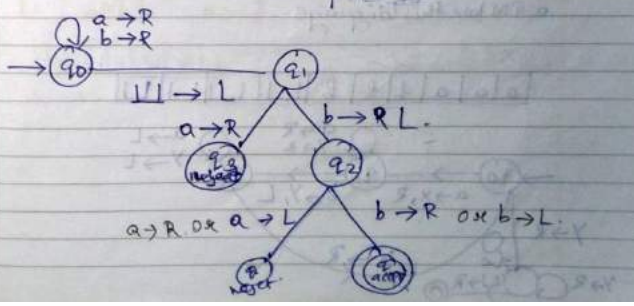
let 

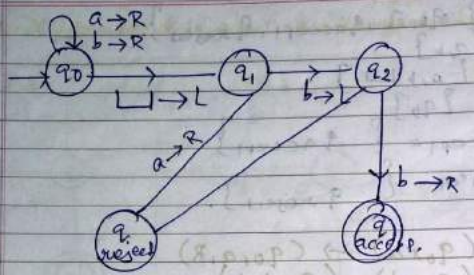


Q Construct a TM for a lang. L which contains strings ending with symbol bb.

$L = \{ bb, abb, bbb, aabb, abbb \ldots \}$.

aabb → 





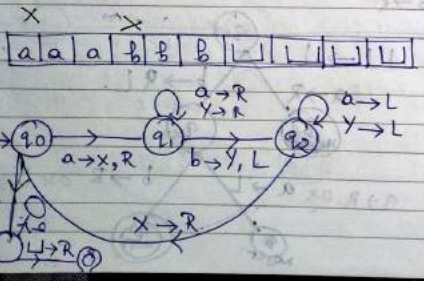
24-10-18

Q Construct a TM for a language $L = \{a^n b^n ; n \geq 1\}$ over $\Sigma = \{a, b\}$

→ The TM is a mathematical model used for recursively enumerable language.

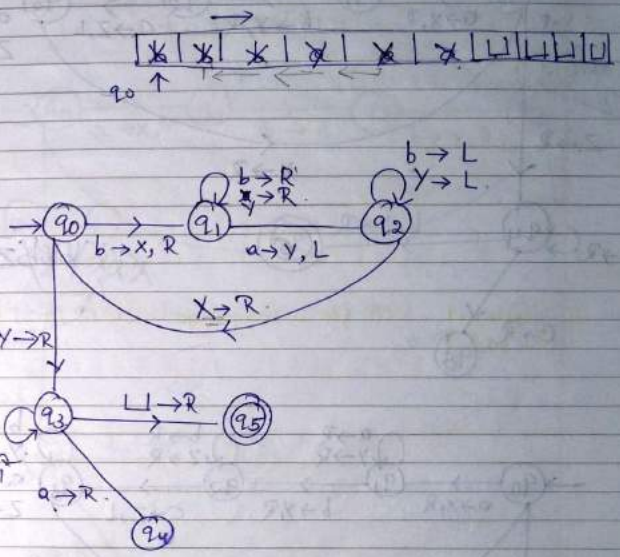
→ The class of RL & class of CFL comes under set of RE languages. [RL ⊂ CFL ⊂ RE]

The above given question is a CFL but we can build a TM for this language.

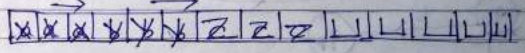


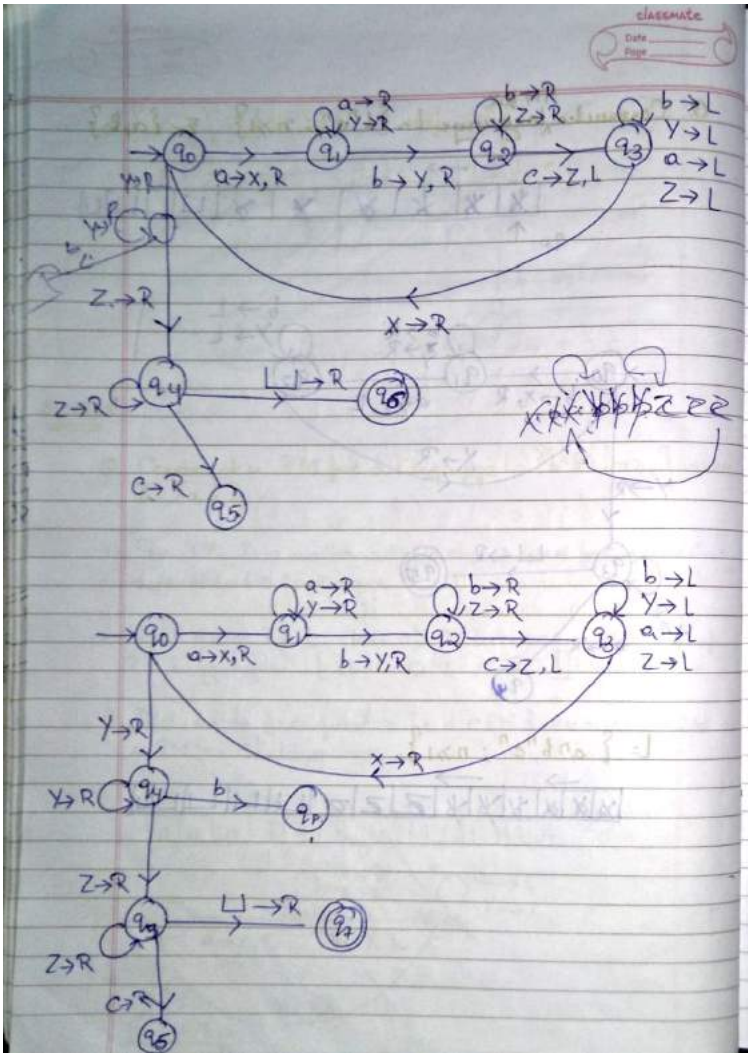
TM for

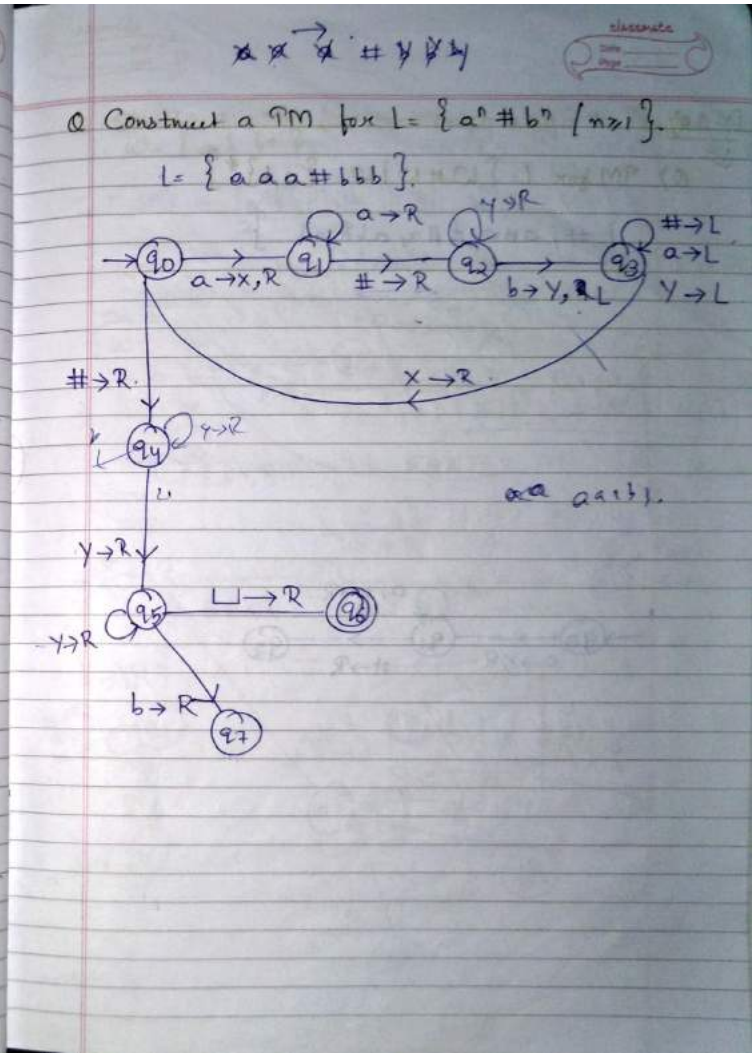
Q Construct a language $L = \{b^n a^n ; n \geq 1\}$ $\Sigma = \{a, b\}$



$L = \{a^n b^n c^n ; n \geq 1\}$.





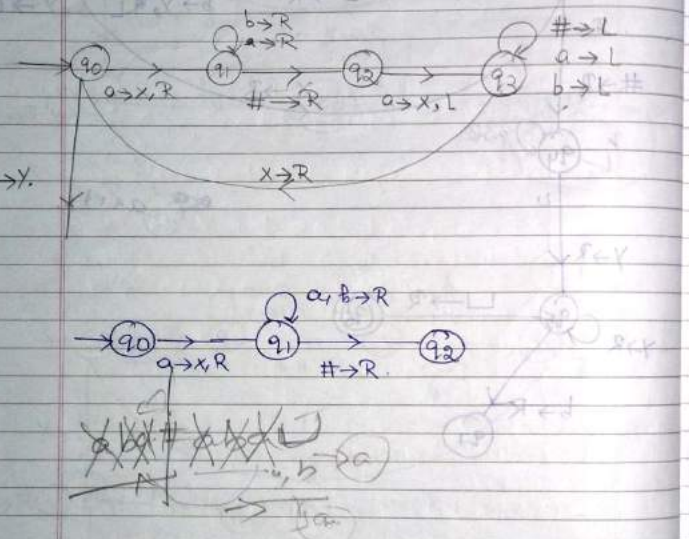


Ex-3.9)

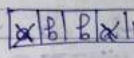
Q) TM for $L = \{ w \# w \mid w = \{a,b\}^* \}$

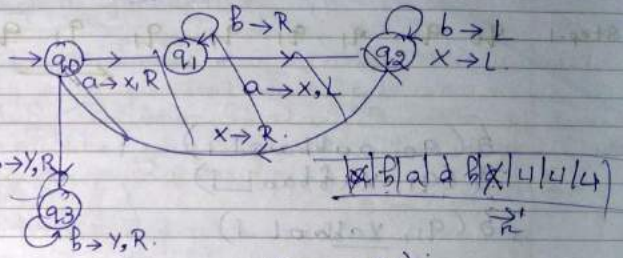
{ #, a#a, b#a, ab#ab ... }

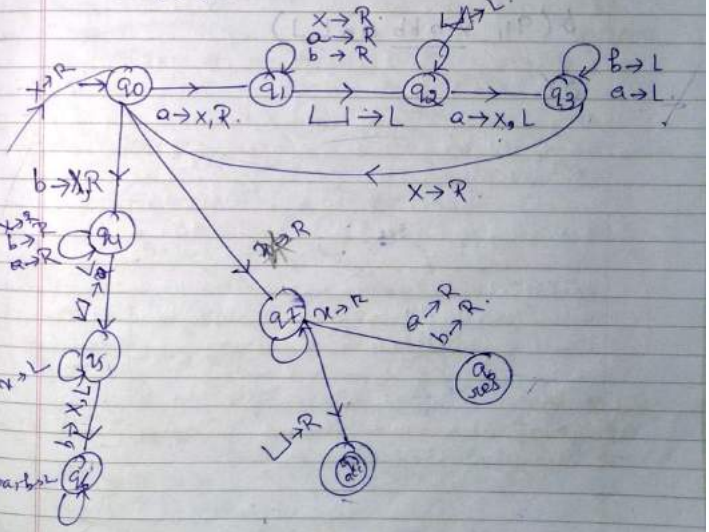
abba#abba



Q Construct a TM for the language $L = \{ w w^R, w = \{ab\}^+ \}$.





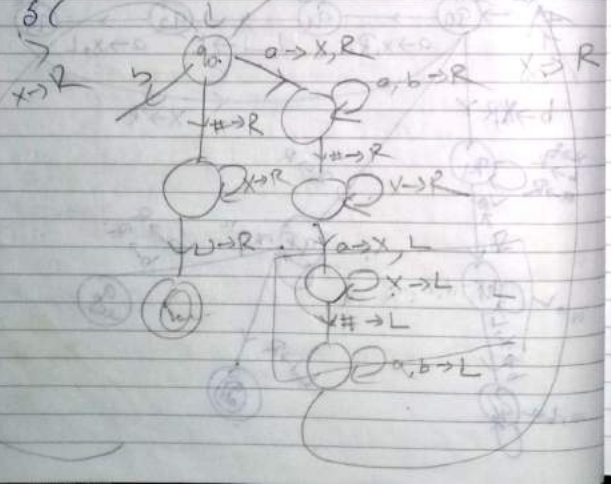


Q Show the computation of the above TM for the string "aabbaa"

X a b b a a ⊔ ⊔

Step-1 q_0 q_1 q_1 q_1 q_1 q_1 q_1 q_1

$\delta(q_0, aabbaa\sqcup)$
$\delta(q_1, Xabbaa\sqcup)$
$\delta(q_1, Xabbaa\sqcup)$
$\delta(q_1, Xabbaa\sqcup)$
$\delta($



X ⊔ ⊔ ⊔ ⊔ ⊔ ⊔

X ⊔ ⊔ ⊔ ⊔ ⊔ ⊔

X X ⊔ ⊔ ⊔ ⊔ ⊔

X X X ⊔ ⊔ ⊔ ⊔

X X X X ⊔ ⊔ ⊔

X X X X X ⊔ ⊔

X X X X X X ⊔

X X X X X X X

Ø Ø O O

q_0 q_1 q_2 q_3

Q $L = \{0^{2^n} \mid n \geq 0\}$ Important

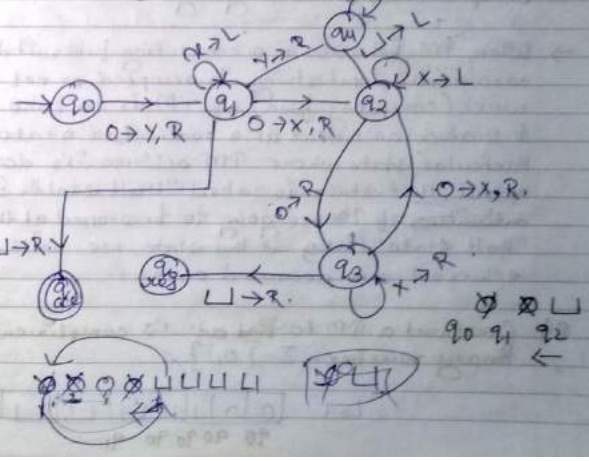
It means lang consists of strings of zeros whose length is power of 2.

If a no. except 1 or 2, when it is divisible by 2 the quotient must be an even no. at every iteration. If at any iteration of division if quotient gives an odd no. then the no. is not in the power of 2.

4/2 = 2 }
0

{0, 00,

8/2 = 4 → 4/2 = 2 }
0 0



Ø Ø ⊔
q_0 q_1 q_2

Ø Ø ⊔ ⊔ ⊔ ⊔ ⊔

Ø Ø ⊔

Q Show the computation of TM for 000000.

$\delta(q_0, \not0\not0\not0\not0\not0\not0\sqcup\sqcup)$

$q_0\ q_1\ q_2\ q_3\ q_2\ q_3\ q_2$

q_3 reject.

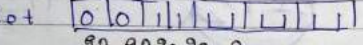
31.10.18

* TM as a Computing Function :

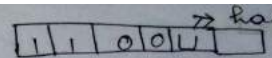
TM can be used as a language acceptor and as a computing function. The TM's till we have drawn has is behaved as a lang. acceptor. When we input a string into TM, TM shows that string is accepted / not, for a language.

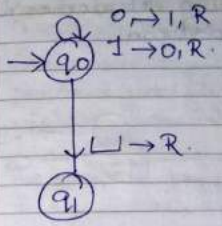
→ When TM behaves as a computing func., it does not care about input string is accepted or not. The work (computing func.) of TM is to take an input & produce an output at a particular state. The particular state where TM achieves it's desired output, that state is called "Halt state". It means activities of TM are going to be stopped at this "Halt state" bcoz at this state we will be achieved our desired output.

Q1) Construct a TM to find out 1's compliment of a binary number. $\Sigma = \{0,1\}$.

Let 

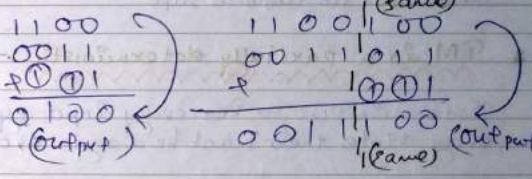
$q_0\ q_0\ q_0\ q_0\ q_1$

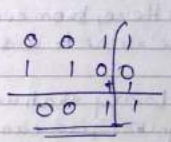




In this TM, there is no q_{accept} / q_{reject}. Here q_1 is the halt state bcoz when we reach to the state we already achieved the output for input string which is 1's compliment in infinite loop.

Q2) Construct a TM to find out 2's compliment of a binary no. $\Sigma = \{0,1\}$.

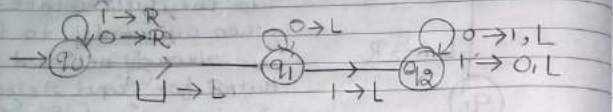




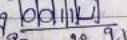
When we input a binary no. scan the complete string from left to right until we reach blank symbol. From blank symbol move towards left until we reach first one. Read & write the same symbols until we reach first one. Then again move forward left from first one & write 1's

1 1 0 0 ␣ ␣ ␣

compliment of remaining string. It means when we read 0 write 1 & vice versa.



In this TM q_2 is not the halt state. Let us take an input string



When we reach final at q_2, we cannot get the output in infinite tape

* TM is a partially deterministic :-

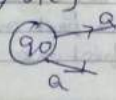
Deterministic → For every input symbol, from every state there must be a single one move

TM → Partially deterministic → Here, from every state for any input symbol there must be a single move.

Here it is not compulsory that we have to take every input symbol, we can choose any input symbol acc. to requirement.

Non-deterministic → Here, for any input symbol, for every state if there is more than one move.

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



1.11.18

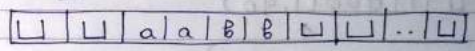
* Variants of TM :-

There are many alternative definations of TM present including multiple tapes & non-determinism. These are called "variants of TM".

The original model (Turing Machine with single tape & single tape head) & its reasonable variance all have the same power because all these TM's recognize the same class of languages called "Recursively Enumerable Lang".

* Types of Turing Machine :-

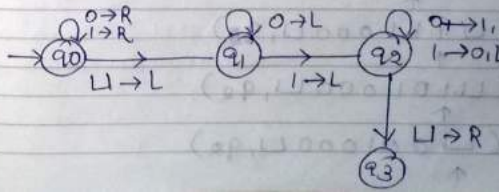
1. Two way infinite tape TM :



In the original model the infinite tape is freezed at first place & open in the last end.

In 2-way infinite tape TM, infinite tape is open in both the place.

Q. Construct a TM to find out 2's compliment of a binary no. Show computation of TM for 111000.



Computation :-

$\delta(\sqcup 1110000 \sqcup, q_0)$
↑
$\delta(\sqcup 1110000 \sqcup, q_0)$
↑
$\delta(\sqcup 1110000 \sqcup, q_0)$
↑
$\delta(\sqcup 1110000 \sqcup, q_0)$
↑
$\delta(\sqcup 1110000 \sqcup, q_0)$
↑
$\delta(\sqcup 1110000 \sqcup, q_0)$
↑
$\delta(\sqcup 1110000 \sqcup, q_1)$
↑
$\delta(\sqcup 1110000 \sqcup, q_1)$
↑
$\delta(\sqcup 1110000 \sqcup, q_1)$
↑
$\delta(\sqcup 1110000 \sqcup, q_1)$ (1's skip)
↑
$\delta(\sqcup 1110000 \sqcup, q_2)$
↑
$\delta(\sqcup 1110000 \sqcup, q_2)$
↑
$\delta(\sqcup 1010000 \sqcup, q_2)$
↑
$\delta(\sqcup 0010000 \sqcup, q_2)$
↑

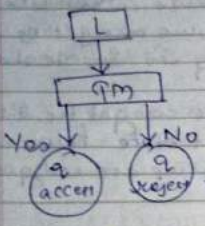
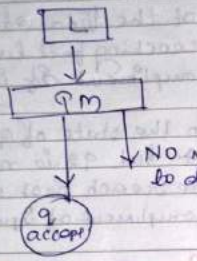
$\delta(\sqcup 0010000 \sqcup, q_3)$.
($\sqcup \to R$)

Hence q_3 is the halt state.
not the final state, because when we are reaching final have to q_3, we are getting 2's complement of input string in infinite state.

In the state of q_2 also we can get the 2's complement but q_2 is not the halt state because when we reach final time in q_2 we are not getting 2's complement of input string.

Property of a halt state is — there should be no activity in the halt state.

Saturday
3.11.18

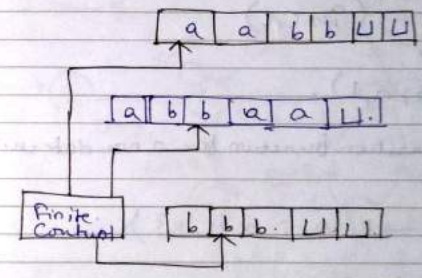
Recursive lang	Recursive Enumerable lang. Turing recognizable
1) Turing Decidable. 	
When we input a string in a TM generates 2 outputs must accept the reject the	
3 TD, Bcoz the TM decides whether the strings of language goes to q accept state & q reject state	TR, bcoz the TM recognizes the accepted strings of the language by going to the q accept states. It doesnt bother about rejected strings of the language.
* 4 RL C REL 	Every REL is not a recursive language.
Every RL is a Recursive enumerable	

176 - 182 Pg No.

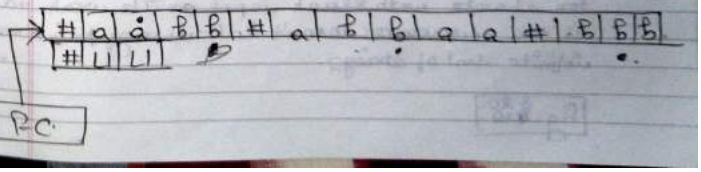
Multitape Turing Machine :-

A multitape TM is like an ordinary TM (TM with single infinite tape & single tape head) with several heads. Each infinite tape has its own head for reading & writing. The transition function for this multitape TM is

$$\delta(Q \times \Gamma^k) \rightarrow Q \times \Gamma^k \times \{L, R, S\}^k.$$



And every multitape can be converted into single tape Turing machine.



In a non-deterministic TM the computation can proceed according to several possibilities. It means by giving an input symbol it can go to multiple states can write multiple symbols & can go to multiple directions.

$(q_0, a) \rightarrow (q_1, b, R)$

$(q_0, a) \rightarrow (q_1, b, R)$

(q_0, C, L)

(q_2, a, L).

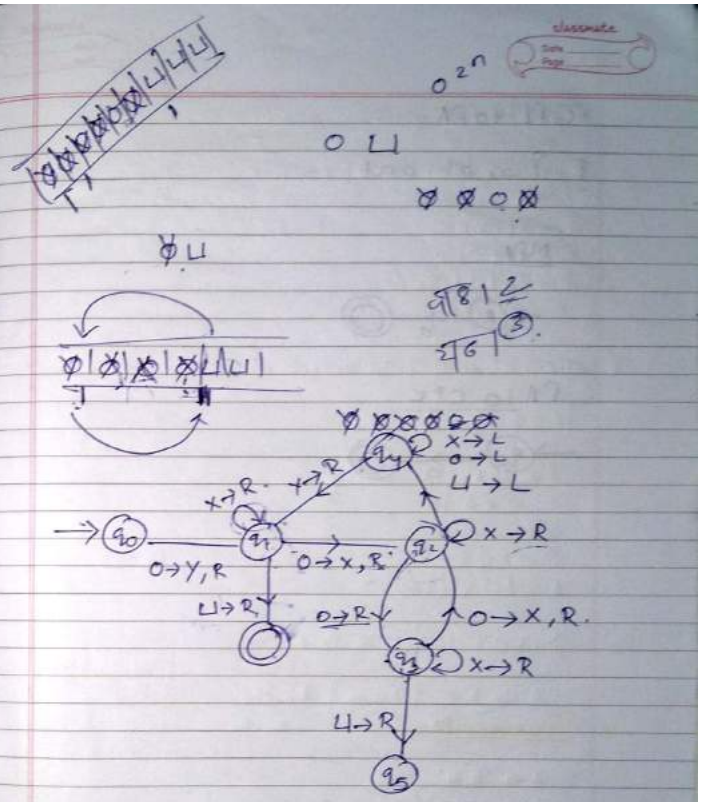
The transition function for a non-deterministic TM is

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

* Enumerator :-

An enumerator is a TM with an attached printer. It means the TM can use the printer as an output device to print strings. An enumerator starts with a blank input on its work tape. If the enumerator doesn't halt it may print an infinite list of strings.

Fig 3.20



Resolving Ambiguity

$R \to R+R \mid R \cdot R \mid R * R \mid a \mid b \mid c$

$R \to R+R \mid T$
$T \to T \cdot F \mid F$
$F \to a \mid b \mid c$

* Eliminating of left recursion

$A \to A\alpha \mid \beta$

$A \to \beta A'$
$A' \to \alpha A' \mid \epsilon$

$S \to Aa \mid b$
$A \to Ac \mid Sd \mid \epsilon$

Substitute the S in A.

$A \to \underbrace{Ac}_{\alpha_1} \mid \underbrace{Aad}_{\alpha_2} \mid \underbrace{bd}_{\beta_1} \mid \underbrace{\epsilon}_{\beta_2}$

$A \to bA'$
$A \to bdA' \mid A'$
$A' \to cA' \mid adA' \mid \epsilon$

$A \to A\alpha \mid \beta$

$A \to \alpha\beta_1 \mid \alpha\beta_2 \mid \alpha\beta_3$ ✓ left factoring

$w = \alpha\beta_3$

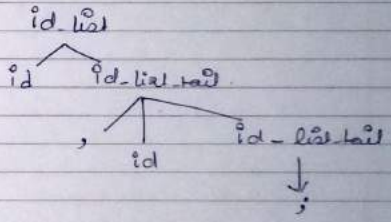
$A \to \alpha A'$
$A' \to \beta_1 \mid \beta_2 \mid \beta_3$
left factoring

left recursion
$A \to \beta A'$
$A' \to \alpha A' \mid \epsilon$

Parsers

Top down

id_list → id id_list_tail
id_list_tail → , id id_list_tail
id_list_tail → ;



Bottom up

reverse of rightmost derivation

$E \rightarrow TE'$

$E' \rightarrow +TE' \mid \epsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' \mid \epsilon$

$F \rightarrow (E) \mid id$

Follow $(E) = \{\$,)\}$

Follow $(E') = \{\$,)\}$

Follow $(T) = \{+, \$,)\}$

Follow $(T') = \{+, \$,)\}$

Follow $(F) = \{\$, *, (, id\}$

First $(E) = \{(, id\}$

First $(E') = \{+, \epsilon\}$

First $(T) = \{(, id\}$

First $(F) = \{(, id\}$

First $(T') = \{*, \epsilon\}$

Department of Computer Science and Engineering
Institute of Technical Education & Research, SOA Deemed to be University

MID-SEMESTER EXAMINATION, OCTOBER-2018

Theory of Computation (CSE 3031)

Semester: 5th **Branch: CSE, CSIT**
Full marks: 30 **Time: 2 hours**

Subject Learning Outcome	*Taxonomy Level	Question Number	Marks
To enhance/develop ability to understand and conduct mathematical proofs for computation and algorithms.	L5, L1	1(a), 3(a)	2+2
To analyse and design finite automata, regular expressions, and regular languages.	L3, L3, L5, L5	1(c), 2(c), 3(b)(c)	2+2+2+2
To analyse and design pushdown automata, context-free languages, and grammars.	L2, L5, L4, L4, L6	4(a)(b), 5(a)(b)(c)	2+2+2+2+2
To analyse and design Turing machines, and recursively enumerable languages.			
To design, implement, and evaluate compuational models to meet desired needs of the languages, and formulate computational models for real-life problems.	L3, L6, L6	2(a)(b), 4(c)	2+2+2
To demonstrate the understanding of key notions, such as algorithm, computability, decidability, and complexity through problem solving.			
To debate, and contrast different classes of formal languages, and their closure properties under language operations.	L6	1(b)	2

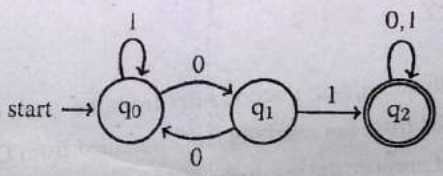
*Blooms taxonomy levels: Knowledge (L1), Comprehension (L2), Application (L3), Analysis (L4), Evaluation (L5), Creation (L6)

Answer all five questions. All questions carry equal marks. All bits of each question carry equal marks.
The figures in the right side indicate marks.

1. (a) A language is said to be a **regular language** if some deterministic finite automaton **(DFA)** recognizes it. Prove that the class of regular languages is closed under the union operation. **[2]**

   (b) The language $L = \{w \mid w \text{ contains at least two } x\text{'s or at most one } y\}$ over the alphabet $\Sigma = \{x, y\}$ is the union of some simpler languages. In each part, construct deterministic finite automatons (DFAs) for the simpler languages, then combine them using the theorem mentioned in **Question 1.(a)** to get the DFA for the given language **L**. **[2]**

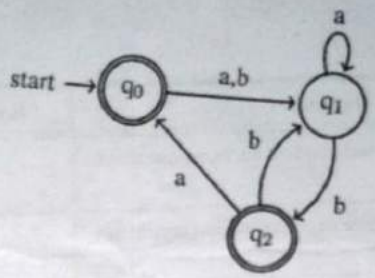
   (c) Identify the strings accepted by the following finite automaton and write them, where the length of the string is exactly **5** and the **2nd** and **4th** bits from the left are **1**. **[2]**



2. (a) Assume $\Sigma = \{c, s, e\}$. Construct an equivalent **ϵ-NFA** with minimum number of states for the regular expression $(c(s \cup \epsilon)e)^+$. **[2]**

   (b) Design an equivalent **DFA** for the **ϵ-NFA** obtained from **Question 2.(a)** and also find the number of states which are not reachable from the start state. **[2]**

(c) Build a regular expression for the following finite automaton, using the concept of generalized nondeterministic finite automaton (GNFA). [2]



3. (a) Define the **pumping lemma** for regular languages. [2]

(b) Consider the language $\mathbf{A} = \{a^n b^n c^n | n \geq 0\}$ over the alphabet $\Sigma = \{a, b, c\}$. Prove that **A** is not regular. [2]

(c) Let $\mathbf{C} = \{\mathbf{a^k} \mid \mathbf{k} \geq 3\}$ be a language over the alphabet $\Sigma = \{\mathbf{a}\}$. Justify the statement " **C** is a regular language ". [2]

4. (a) Show that the following grammar is ambiguous. [2]

$$S \rightarrow AB \mid aaB$$
$$A \rightarrow a \mid Aa$$
$$B \rightarrow b$$

(b) Consider the grammar G: [2]

$$S \rightarrow A1B$$
$$A \rightarrow 0A \mid \epsilon$$
$$B \rightarrow 0B \mid 1B \mid \epsilon$$

Find the language recognized by the grammar G. Also check wheather the language is regular or not. Justify with reason.

(c) Consider the language $\mathbf{L} = \{\mathbf{w} \mid \mathbf{w}$ **starts and ends with the same symbol**$\}$ over the alphabet $\Sigma = \{0, 1\}$. Construct a finite automaton for the language L. Also construct a context free grammar from the constructed automaton. [2]

5. Consider the following context-free grammar G:

$$\mathrm{S} \rightarrow \mathrm{aXbX}$$
$$\mathrm{X} \rightarrow \mathrm{aY} \mid \mathrm{bY} \mid \epsilon$$
$$\mathrm{Y} \rightarrow \mathrm{X} \mid \mathrm{c}$$

(a) Simplify the grammar G by eliminating ϵ-production rules from the given CFG. [2]

(b) Eliminate all unit production rules from the resulting CFG obtained from **Question 5.(a)**. [2]

(c) Eliminate all useless symbols and propose the resulting grammar obtained from **Question 5.(b)** in Chomsky Normal Form. [2]