finite automata

(1) is an abstract m/c which consists of five tuples $(Q, \Sigma, q_0, \delta, F)$

(2) $Q \times \Sigma \rightarrow Q$

(3) Indicate acceptance of input string by final state only

(4) FA has very limited memory so it can't recognize certain languages

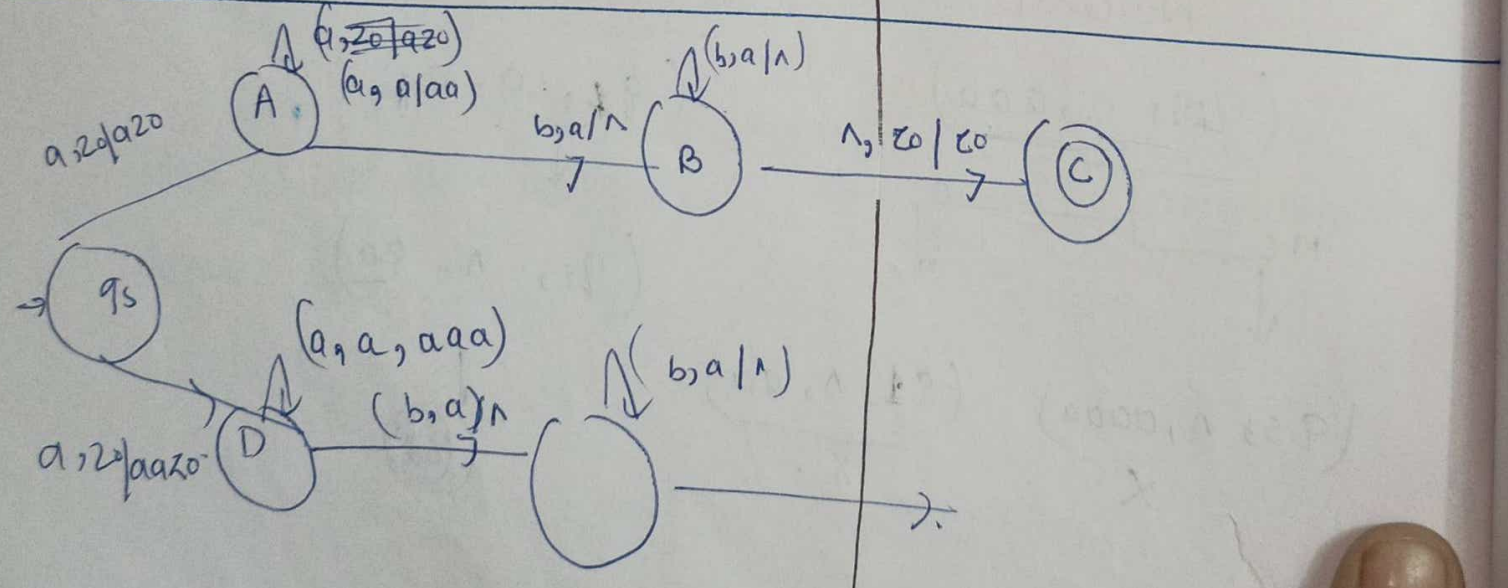
PDA

(1) Like FA with auxiliary memory (stack) which consists of seven tuples.

(2) $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$

(3) by final state or by empty stack.

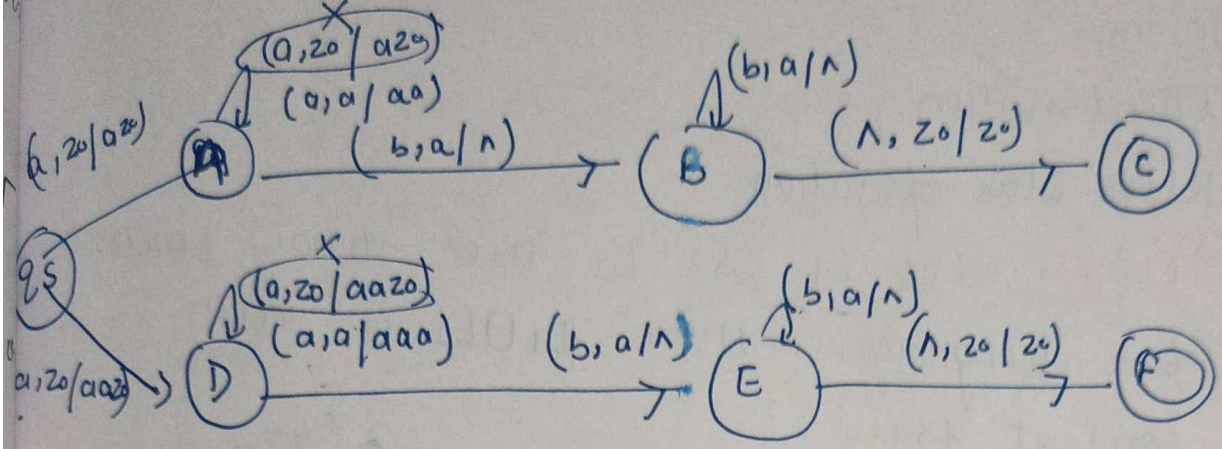
(4) auxiliary memory



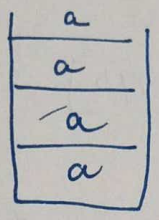
NDPDA

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

{ Not looking at a, I can't decide whether to push one a }



aabbbb



?

aabbbb

(qs, aabb, Z0)

(A, abb, aZ0)

(A, bb, aaZ0)

(B, b, aZ0)

(B, ∧, Z0)

C

(D, abb, aaZ0)

(D, bb, aaaaZ0)

(E, b, aaaZ0)

(E, ∧, aaZ0)

X.

(qs, aabb, Z0)

(A, abb, aZ0)

(A, bb, aaZ0)

(B, b, aZ0)

(B, ∧, Z0)

C

(D, abb, aaZ0)

CFL Closure property

Closure property:
are closed under:

1. Union.
2. Concatenation
3. Kleen Star operation

① **UNION** Let L_1 & L_2 are two context free languages then $L_1 \cup L_2$ is also context free

$L_1 = \{ a^n b^n \mid n \geq 0 \}$ $\quad S_1 \rightarrow aS_1b \mid ab$

$L_2 = \{ c^m d^m, m \geq 0 \}$ $\quad S_2 \rightarrow cS_2d \mid \wedge$

$L = L_1 \cup L_2 = \{ a^n b^n \} \cup \{ c^m d^m \}$

$S \rightarrow S_1 \mid S_2$ $\quad a^n b^n c^m d^m$

② **Concatenation** If L_1 & L_2 are context free languages, $L_1 L_2$ is also context free

$L_1 L_2 = \{ a^n b^n c^m d^m \}$

$S \rightarrow S_1 S_2$

③ **Kleen Star** If L is a context free language, L^* is also context free.

$L = \{ a^n b^n \mid n \geq 0 \}^*$ $\quad S \rightarrow aSb \mid \wedge$

$(a^n b^n)(a^n b^n)$

$S_1 \rightarrow SS_1 \mid \wedge$

$L_1 = \{a^n b^n\}$. $a=b$ $b=c$

Context free languages are not closed under

(1) Intersection If L_1 & L_2 are CFL, then $L_1 \cap L_2$ is not necessarily context free

$L_1 = \{0^m 1^m 0^n : m, n \geq 0\}$

$L_2 = \{0^m 1^n 0^n : m, n \geq 0\}$

but $L_1 \cap L_2 = \{0^m 1^m 0^m : m \geq 0\}$ is not CFG.

Complement

$\overline{L_1 \cup L_2}$

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

Not =

(1) $L_1 = \{a^n b^n\}$ $n \geq 0$ $S_1 \to a S_1 b \mid \lambda$

$L_2 = \{w w^R$ $S_2 \to a S_2 a \mid b S_2 b \mid \lambda$.

$L = \{a^n b^n \cup \{w w^R\}$ $S \to S_1 \mid S_2$.

$L_1 L_2$

$\overline{L_1 \cup L_2}$

$= L_1 \cap L_2$

(2) Concatenation

$\{a^n b^n\}\{w w^R\}$ $S \to S_1 S_2$.

(3) Star $L = \{a^n b^n\}$ $n \geq 0$ $S \to a S b \mid \lambda$

$L = (a^n b^n)^*$ $\Rightarrow$ $S \to S S_1 \mid \lambda$,

Intersection

eg $L_1 = \{a^n b^n c^m\}$

$S \to AC$

$A \to aAb \mid \lambda$

$C \to cC \mid \lambda$

$L_2 = \{a^n b^m c^m\}$.

$S \to AB$

$A \to aA \mid \lambda$

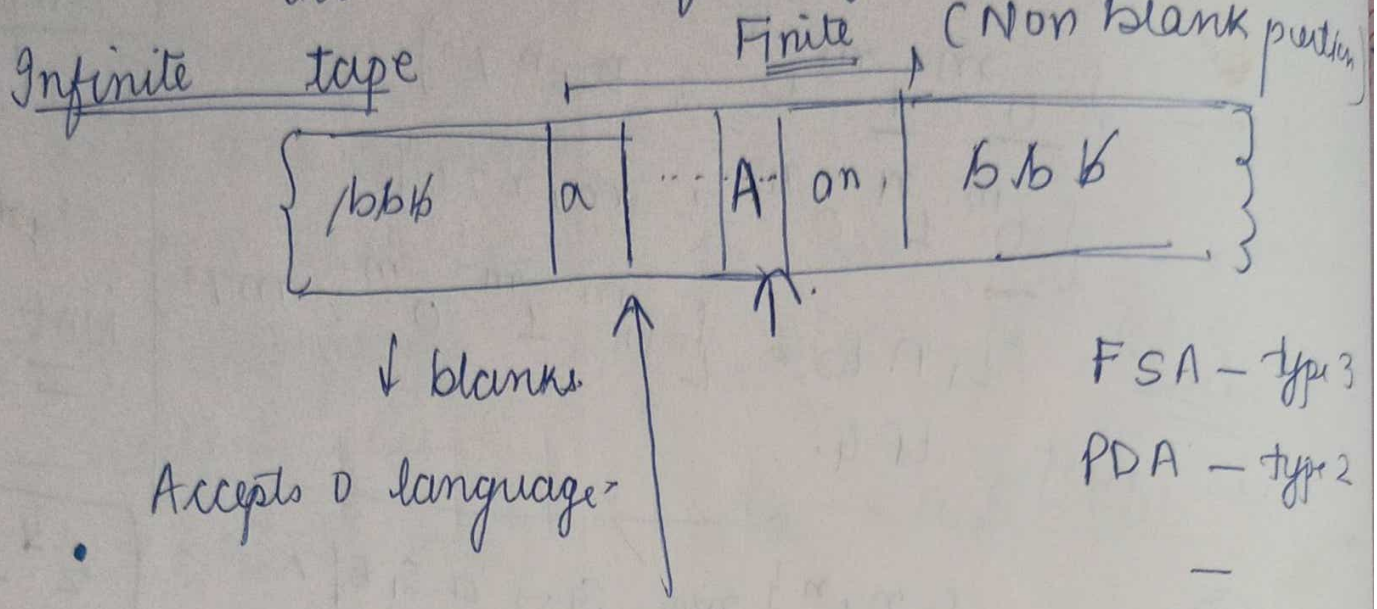
$B \to bBc \mid \lambda$

Int $L_1 \cap L_2 = \{a^n b^n c^n\}$

Turing Machine

Alan A. M. Turing in 1936 (called as computing called what could naturally be called an effective procedure can be realised by TM

Church - Turing Hypothesis



FSA — type 3

PDA — type 2

—

Turing m/c → type 0

Effective procedure is a program

procedure algorithm — always halt.

next step being done.

can halt or may not halt

Input is natural no.

Test whether it is prime.

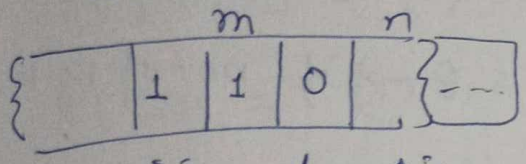
algorithm it definitely have an ans.

Given n, is n is prime

Given n, is there a perfect no > n?

6 1+2+3

28 1+2+4+



computing function

N = {0, 1}.

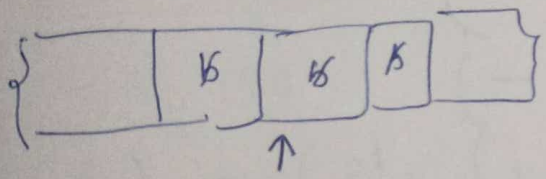
0 → 1

1 → 1

2 → 0

3 → 1.

·1101 — real no.



TM n states

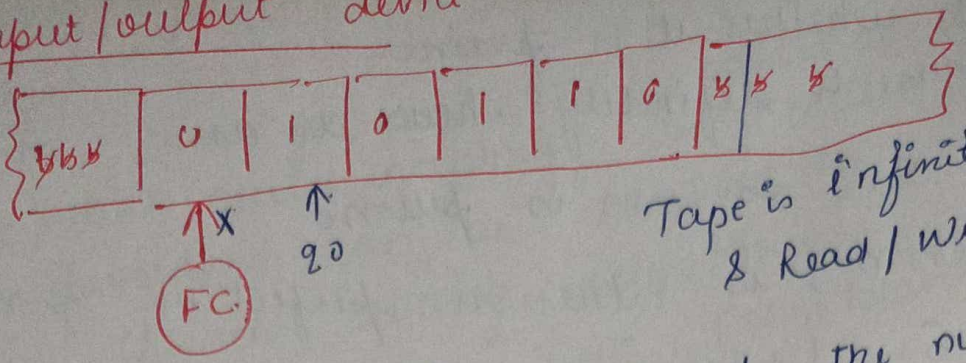
It should print 1's & halts

f(n) → busy beaver function

undecidable → context free grammar is ambiguous or not

$$\delta(q, a) = (p, b, R).$$

Input/output device



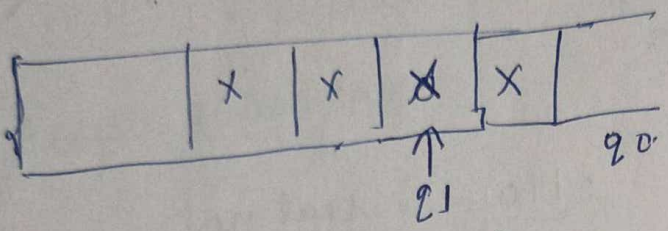
Tape is infinite & Read/Write

$\delta(q_0, 0) = (q_0, X, R)$.

$\delta(q_0, 1) = (q_1, X, R)$

0 - if the number of 1's is odd

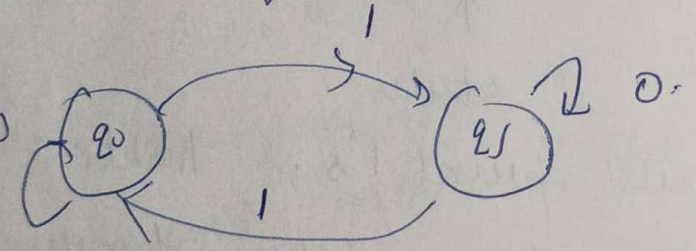
ϵ → if no. of 1's is even



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

$\delta(q_1, 1) = (q_0, X, R)$

$\delta(q_1, B) = (H, 0, -)$



$\delta(q_0, B) = (H, \epsilon, -)$

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

$\delta(q_1, 1) = (q_0, X,$

Parity checker

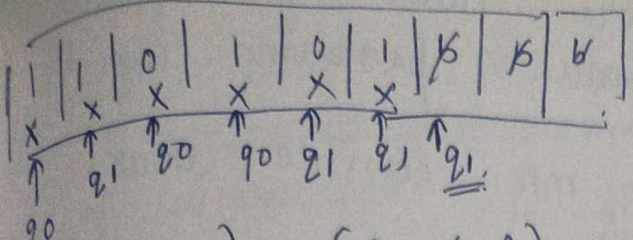
States $\{q_0, q_1\}$

Input alphabet = $\{0, 1\}$

Tape alphabet $\{0, 1, B, X\}$

$\delta(q_0, 0) = (q_0, X, R)$

$\delta(q_0, 1) = (q_1, X, R)$



$\delta(q_0, 1) = (q_1, X, R)$.

$\delta(q_1, 1) = (q_0, X, R)$

$\delta(q_0, 0) = (q_0, X, R)$

$\delta(q_0, 1) = (q_1, X, R)$

$\delta(q_1, 0) = (q_1, X, R)$

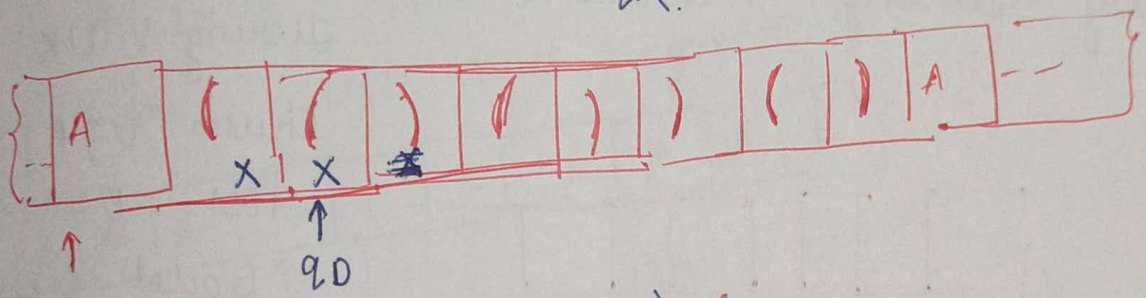
$\delta(q_0, b) = \cancel{(q_0, X, R)}$

$(H, \epsilon, -)$.

Paranthesis checker

$q_0 \rightarrow$

$q_1 \leftarrow$



$\delta(q_0, ()) = (q_0, (, R)$

$\delta(q_0,)) = (q_1, X, L)$

$\delta(q_0, X) =$

$\delta(q_0, ($

TM

To capture the notion of computability, What all computational problems can be solved?

What turing m/c can do that computers thr can do.

Q What all can be done by any algorithm, can be done by Turing Machine.

Algorithm is an intuitive notion.

Turing M/cs capture formally this intuitive notion

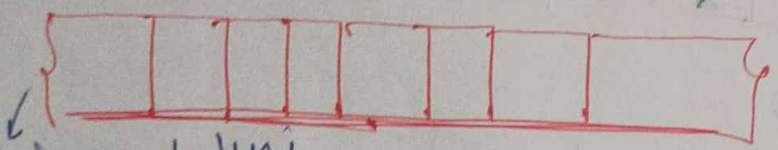
Church Turing Thesis

12:30

What can be done algorithmically can be done by Turing Machine and vice-versa.

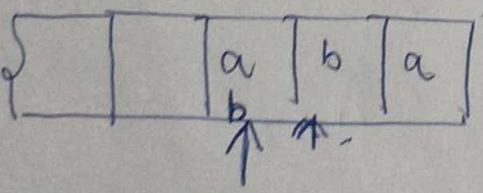
Turing - UK
Church USA
Post USA
Godel Germany
Markov - USSR

Turing

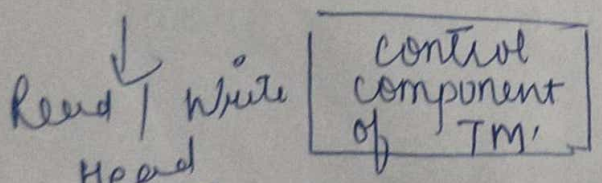


jagged line

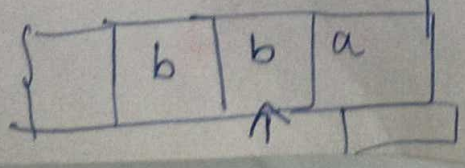
(1) Infinite Tape



(2) Head is both read/write



Present state	Symbol scanned	Next state	Symbol written
q	a	p	b



One step Right.

Dynamic behaviour of TM is specified by the quintuples that TM has.

Set of quintuples is finite

Σ alphabet (finite set)

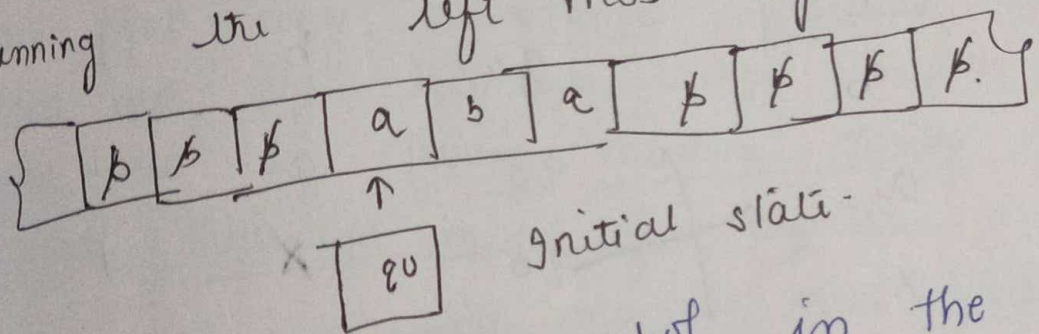
Q set of states (finite set)

$|S| \times |\Sigma|$ = at most 50

10 5

Suppose TM has state p and there is a symbol b such that there is no quintuple with present state p and present symbol, in that case, machine halts

Initially TM is in initial state scanning the left most symbol of input.



Initial state.

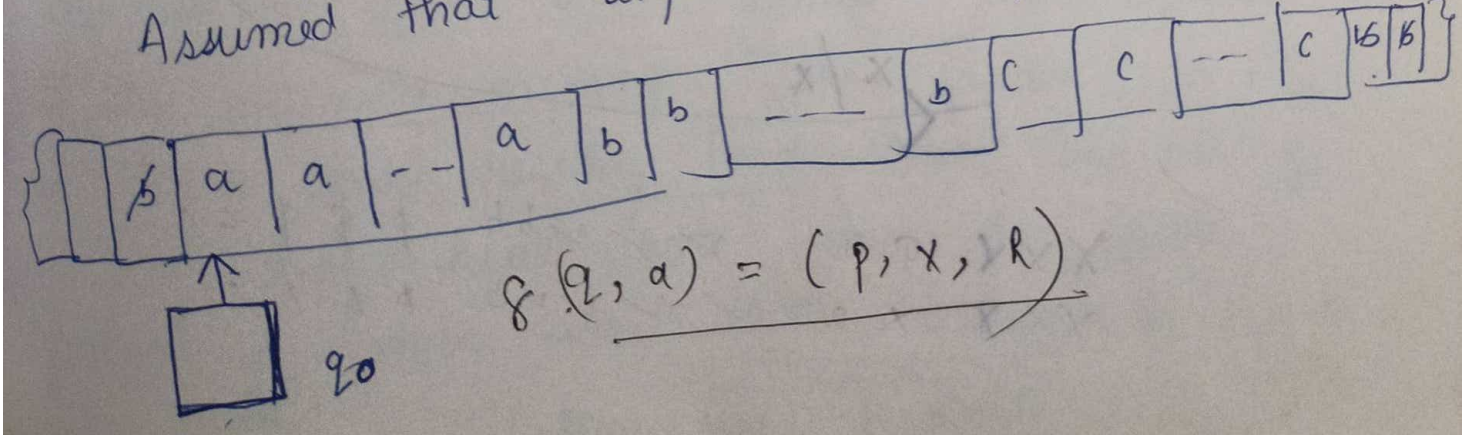
$\not b$ → special symbol in the alphabet Σ.

most of type will have

$\{a^n b^n c^n \mid n \geq 0\}$

Assumed that input is of form $a^* b^* c^*$.

aa-a bb--- b c c---c.



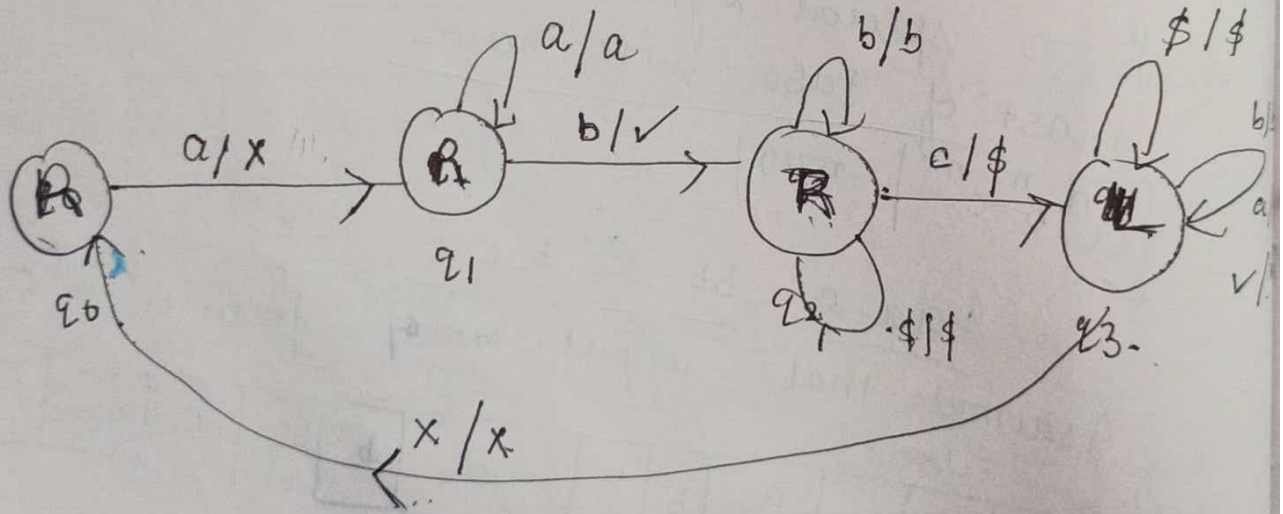
$\delta(q, a) = (p, X, R)$

b a a a — a b b b — b c c — c

b x a a — a ✓ b b — b $ c — c

b x x a — a ✓ ✓ b — b $ $ — c

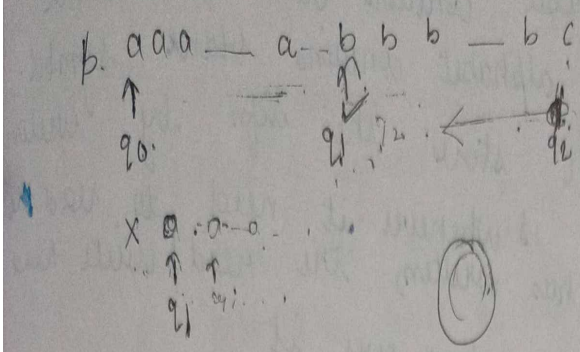
Present state	Present symbol	Next state	Symbol overwritten	
q0	a	q1	X	
q1	a	q1	a	
(q0) q1	b	q2	✓	R
q2	b	q2	b	R
q2	c	q3	$	L
q3	$	q3	$	L
q3	b	q3	b	L
q3	a	q3	a	L
q3	X	q0	X	R



X X ✓ a a ✓ ✓ ✓ b b $ $ $ c c

X X X X a ✓ ✓ ✓ ✓ b $ $ $

$a^n b^n c^n$



aabb.

aabbcc

$a^n b^n$

10 marks

3 13 15

In 1936, Alan Turing gave a model using Turing m/c concept. The church-Turing Thesis states that any algorithmic procedure that can be carried out by human beings/ computer can be carried out by Turing machine. Turing m/c provides an ideal theoretical model of computer.

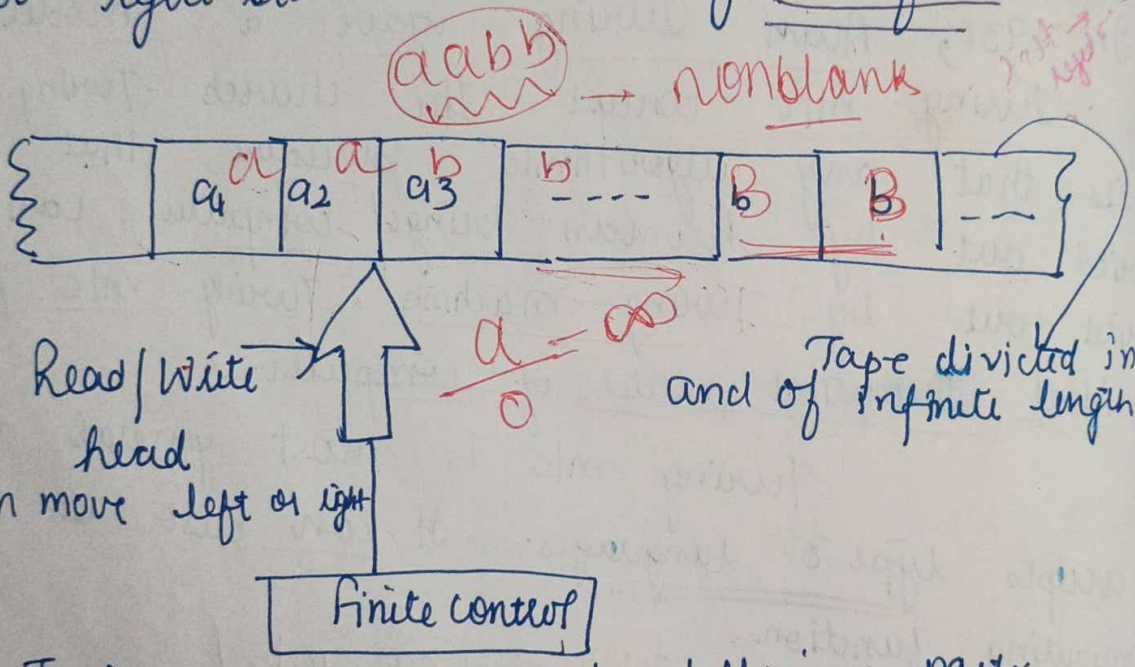
Turing m/c is most general model. It accepts type 0- languages. It can also be used for computing functions.

The turing m/c

The turing m/c uses an infinite tape, which is divided into squares or cells & each cell can hold one symbol from alphabet. It has unlimited memory. The ~~Head~~ In FA, we have only reading head but in turing m/c, the tape head can read & write symbols & move in both directions (left & right) (can read & write

Initially, the tape contains only input string & all empty square anywhere contains blank symbol. The input alphabet contains all non-blank symbols & tape alphabet contains blank symbol. The turing m/c stores the info by writing info on tape & whenever it need to read the info that it has written, the read/write head of turing m/c moves back over it.

& The turing m/c produces output accept or reject based on entering designated accepting or rejected states. If the m/c do not enter accept state or reject state then it will go on forever & never halts.



Turing m/c consists of following parts:

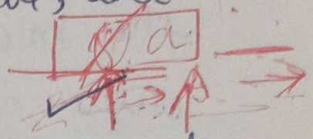
(1) Tape
(2) Read-write head
(3) Finite control

Tape
Read-Write Head

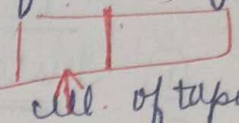
(1) Tape → divided into cells. Each cell holds one of finite no. of tape symbols. The cells contains symbols $a_1, a_2 \ldots a_n$ and the remaining cells holds blank symbols (b), which is a special tape symbol not included in input alphabet symbol

Tape symbols can be denoted by Γ.

2) Read/Write head → The tape head is always positioned at one of tape cell. In one move, the turing m/c



① may change state
② Write a symbol on cell being scanned.
③ Move the head one position left or right
④ Whether to halt or not



Initially, the tape head is at leftmost cell of tape

3) Finite control Finite control can be in any one of states

① Initial state → state at time when turing m/c starts its computation.

② Halt state - in which turing m/c stops its computation

③ Intermediate state. All states other than initial & halt state.

Formal defn of Turing m/c → Turing machine can be defined with seven tuples $(Q, \Sigma, \Gamma, \delta, q_0, b, F)$

Q → finite non-empty set of states

Γ → finite non-empty set of tape symbols

where $b \in \Gamma$

δ is transition fx., mapping state & tape symbols to state, tape symbol and C) movement of head.

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

L: Left movement of tape head

R:

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

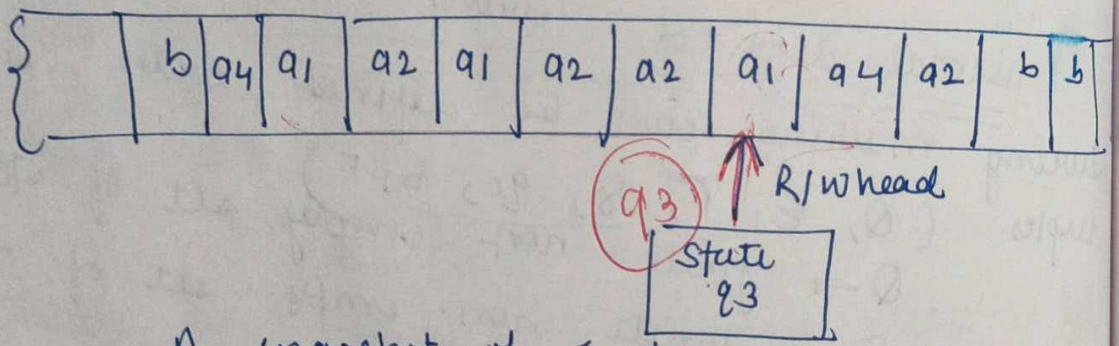
$\delta(q, b) = (r, a, R)$

↳ m/c in state q & after reading symbol b the m/c writes a new symbol a on tape, enters new state r and tape head moves towards right one position.

Representation of Turing M/C

1. Instant aneous descriptions
2. Transition Table
3. Transition Diagram.

→ $\alpha\beta\gamma$, when β is present state of M, the entire input string is split as $\alpha\gamma$, the first symbol of γ is current symbol a under the R/W head & γ has all subsequent symbols of input string & string α is substring of input string formed by all symbols to left of a



A snapshot of Turing M/C

The present symbol under the R/W head is a_1 present state is q_3. So, a_1 is written to right of q_3. the non-blank symbols to left of a_1 form the string $a_4 a_1 a_2 a_1 a_2 a_2$, which is written to left of q_3.

The seq. of non blank symbols to right of a_1 is $a_4 a_2$

Representation of ID

$a_4 a_1 a_2 a_1 a_2 a_2$ → q_3 → left sequence

q_1 → $a_4 a_2$ → Right sequence.

q_3 → Present state

q_1 → symbol under R/W head.

Representation of ID

(Q1.)

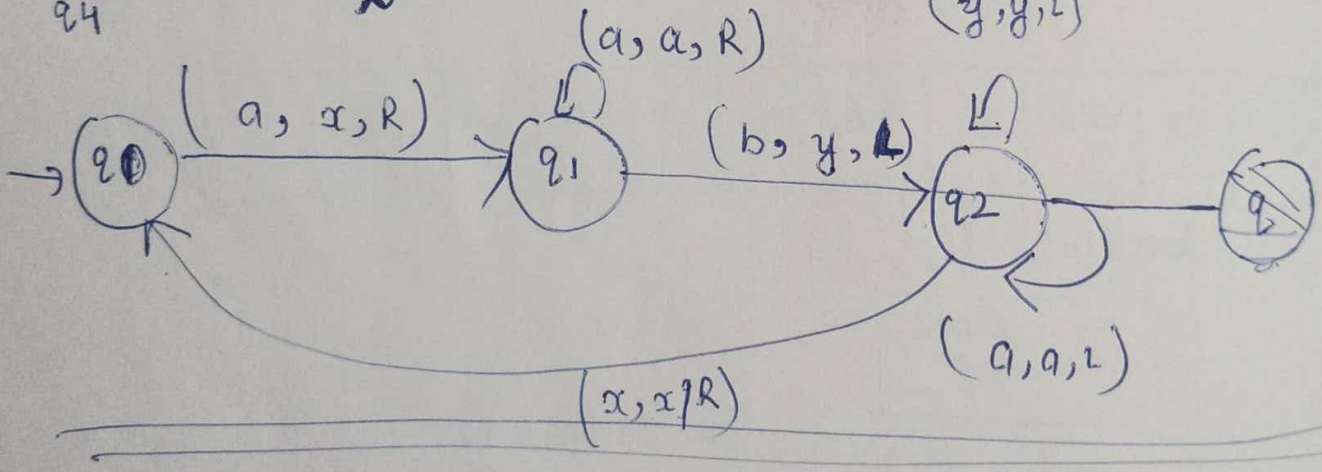
$\delta(q_1, a) \rightarrow \delta(q_2, b, R)$.

$ a a a ____ a b b b ____ b

x y

Present state	Present symbol	Next state	Symbol overwritten	
q_0	a	q_1	x	R
q_1	a	q_1	a	R
q_1	b	q_2	y	L
q_2	y	q_2	y	L
q_2	a	q_2	a	L
q_2	x	q_0	x	R
q_2				

q_1	0	q_2	x	R
q_2	0	q_2	0	R
q_2	1	q_3	y	L
q_2	y	q_3	y	L
q_3	0	q_4	0	L
q_3	0	q_4	0	L
q_4	0	q_1	x	R
q_4	x		(y, y, L)	



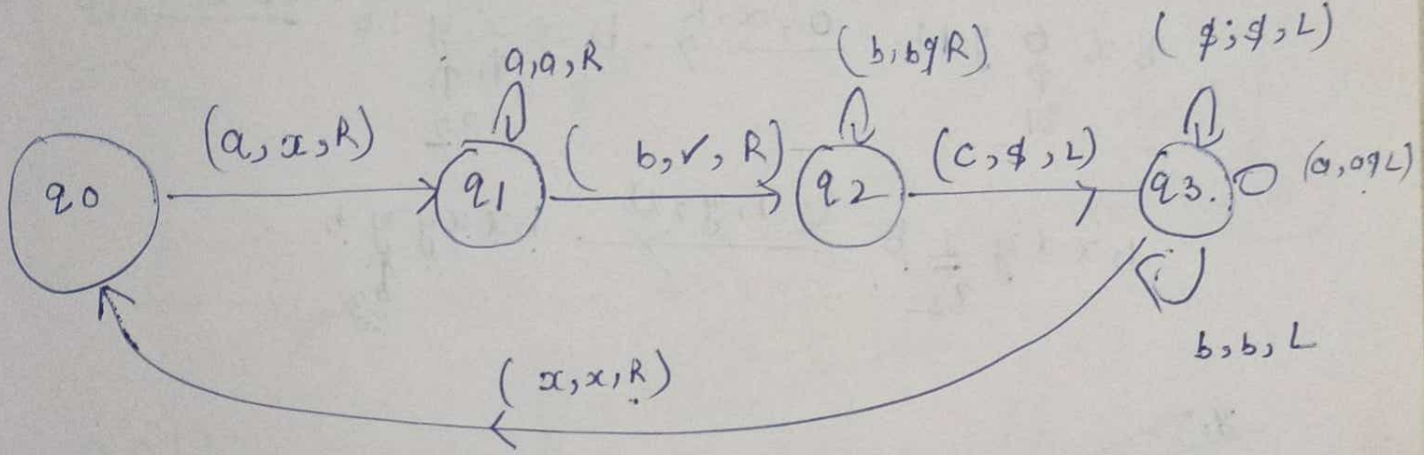
a a ___ a b b b

x y

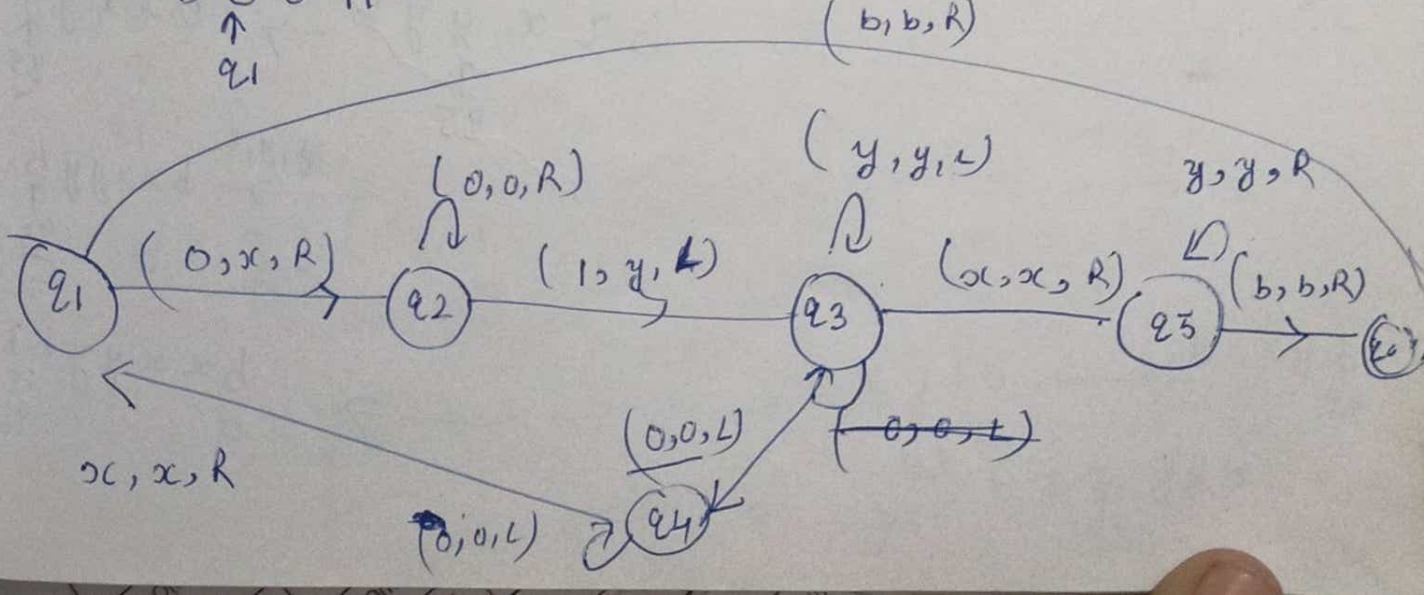
$ $ a a a ___ a b b b ___ b c c c ___

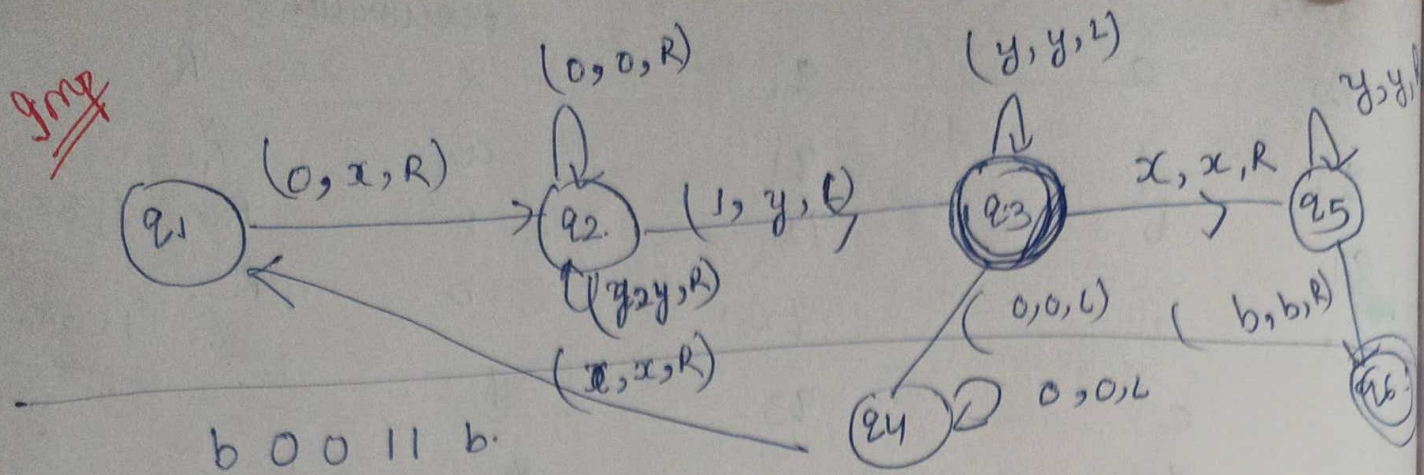
x ___ a ✓

Present state	Present Symbol	Next state	Symbol overwritten	Move
q0	a	q1	x	R.
q1	a	q1	a	R
q1	b	q2	y	R.
q2	b	q2	b	R.
q2	c	q3	$	L.
q3	$	q3	$	L
q3	b	q3	b	L
q3	a	q3	a	L
q3	x	q0	x	R



b 0 0 11 b.





b 0 0 11 b.
↑
q1

bx0 11b $\xrightarrow{0,0,R}$ bx0 11 b $\xrightarrow{1,y,L}$
q2 ↑ q2.

bx0y1b $\xrightarrow{(0,0,L)}$ bx0y1b $\xrightarrow{(x,x,R)}$
q3 ↑ q4

bx0y1b. $\xrightarrow{0,x,R}$ bxxy1b $\xrightarrow{(y,y,R)}$
↑ q1 ↑ q2

bxxy1b $\xrightarrow{(1,y,L)}$ bxxyyb
q2 ↑ q3.

y,

bxxyyb $\xrightarrow{y,y,R}$ bxxyyb
↑ q5 ↑ q5

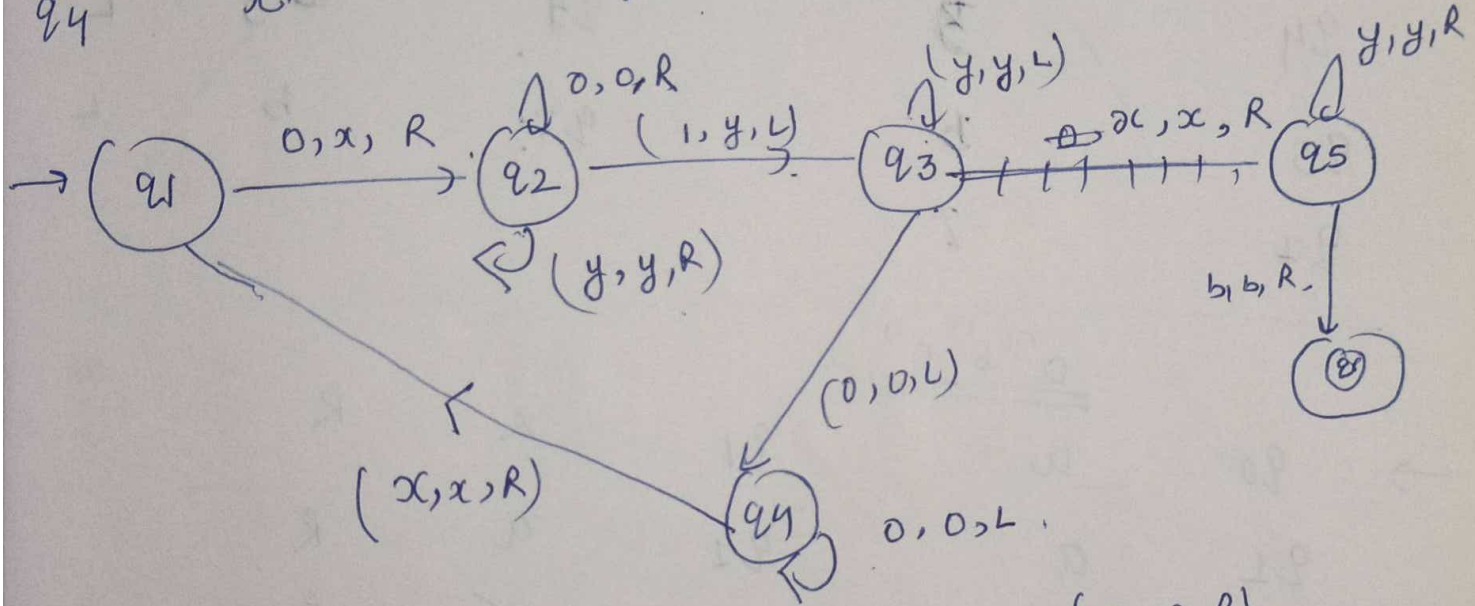
$\xrightarrow{y,y,R}$ bxxyyb
↑ q5

bxxyyb
↑ q6

→

(1) $a^n b^n$ $0^n 1^n$

Present state	Present symbol	Next state	Symbol overwritten	Move
q_1	0	q_2	x	R
q_2	0	q_2	0	R
q_2	1	q_3	y	L
q_3	y	q_3	y	L
q_3	0	q_4	0	L
q_4	0	q_4	0	L
q_4	x	q_1	x	R



0011

b 0 0 1 1 b $\xrightarrow{0,x,R}$ b x 0 1 1 b $\xrightarrow{(0,0,R)}$ b x 0 1 1 b

b 0 0 1 1 b $\xrightarrow{0,0,L}$ b x 0 y 1 b $\xrightarrow{x,x,R}$ b x 0 y 1 b $\xrightarrow{1,y,L}$ b x 0 y 1 b

b x 0 y 1 b $\xrightarrow{0,x,R}$ b x x y 1 b $\longrightarrow$ b x x y 1 b

b x x y y b $\longrightarrow$ b x x y y b

$1^n 2^n 3^n \mid n \geq 1$

1 1 —— 2 2 —— 3 3
b b b

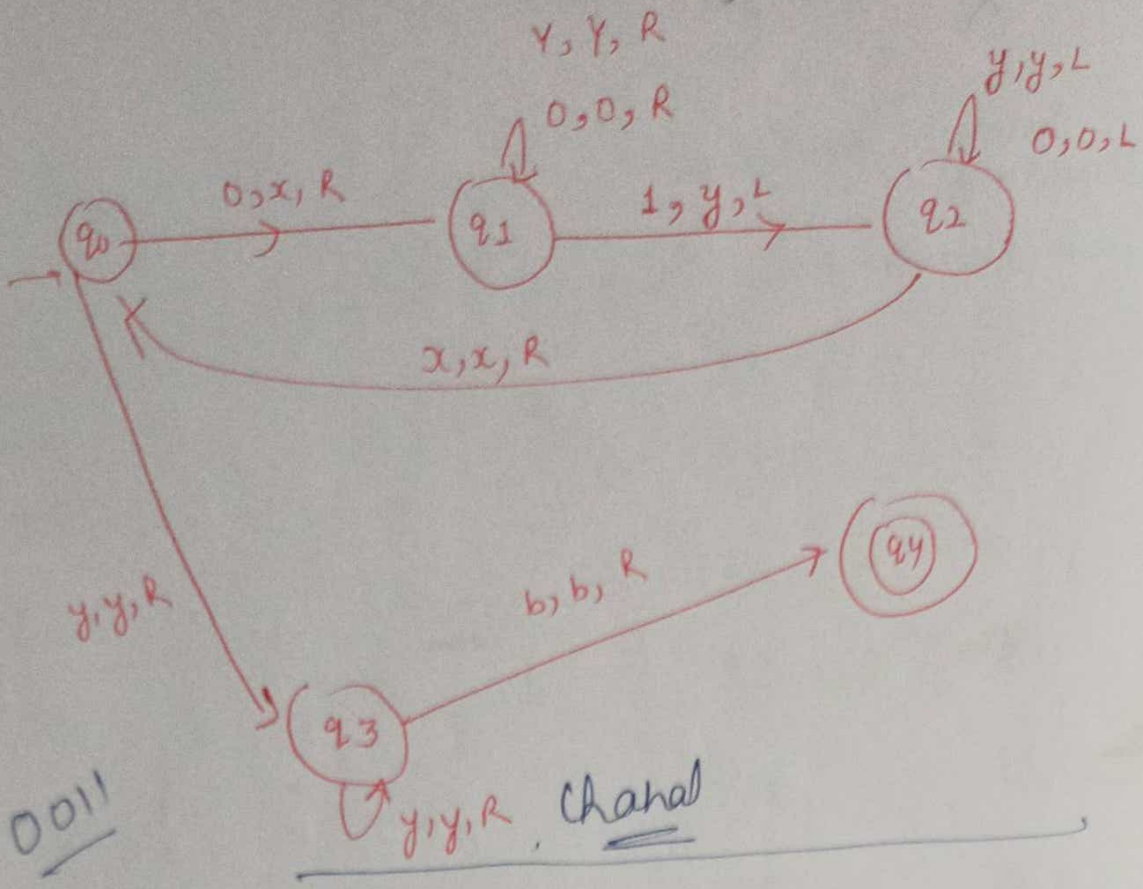
Present State	Present Symbol	Next State	Symbol overwritten	Move
q_1	1	q_2	b	R
q_2	1	q_2	1	R
q_2	2	q_3	b	R
q_3	2	q_3	2	R
q_3	3	q_4	b	R
q_4	3	q_5	3	L
q_5	b	q_7	b	L
q_7	2			

$a^n b^n c^n$

→ q_0	a	q_1	x	R
q_1	a	q_1	a	R
q_1	b	q_2	✓	R
q_2	b	q_2	b	R
q_2	c	q_3	$	L
q_3	$	q_3	$	L
q_3	b	q_3	b	L
q_3	✓	q_3	✓	L
q_3	a	q_3	a	L
q_3				

X Q0 X R.

Q3.



0011

Chahal

$q_0 0011 \vdash q_0 x 011 \vdash q_0 x 0 11$

$\vdash q_0 x 0 y 1 \vdash q_0 x x y 1$

$\vdash q_0 x 0 y 1 \rightarrow q_0 x x y 1$

$\rightarrow q_0 x x y 1 \rightarrow q_0 x x y 1$

$\rightarrow q_0 x x y y$

$q_0 x x y y \rightarrow x x y y \rightarrow x x y q_3 b$

Type 3 :

$A \to \alpha B / \beta$
$A, B \in V$
$\alpha, \beta \in T^*$
} RLG

$\left[A \to B\alpha \mid \beta \right]$ LLG.

$A, B \in V$
$\alpha, \beta \in T^*$ (strings of terminals)

$A \to @B$
$B \to aB \mid bB \mid a \mid b.$
} RLG.

$A \to Ba \mid a$
$B \to Ba \mid Bb \mid a \mid b.$

$A \to Ba \mid a$
$B \to aB \mid a$

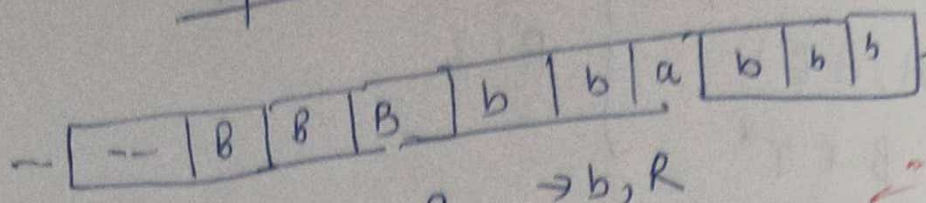
(20)

(20) 1.=

Chahal

NDFA DFA.

Tape



final state (halt) ⇒ accept, reject

Recursive Enumerable. Turing Recognizable

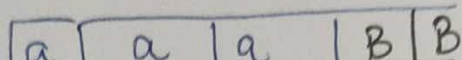
If TM accepts all strings of language

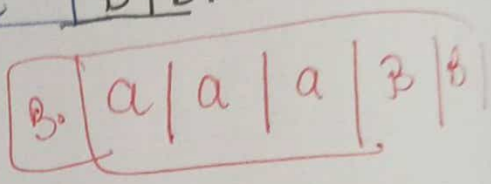
Decidable / Recursive Language - Total Turing M/c

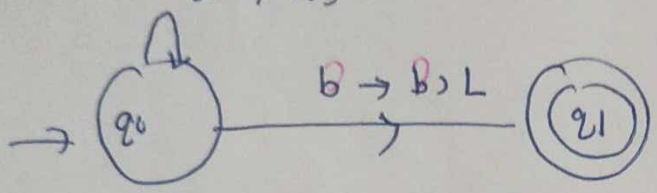
If TM accepts the strings of language / rejects the strings not in language

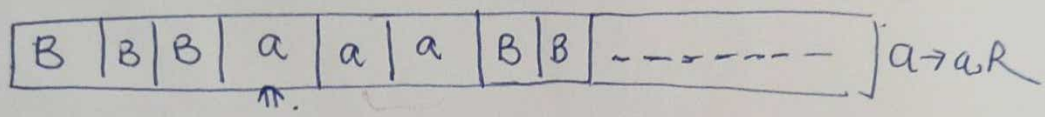
Q Construct a turing m/c for a^*.

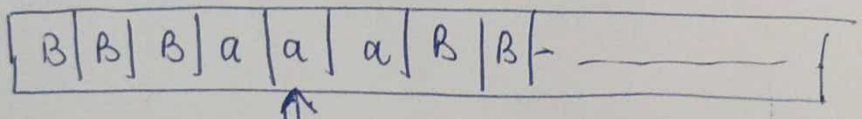
$RE = a^*$

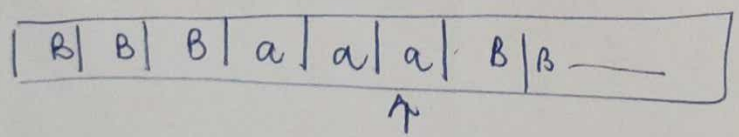


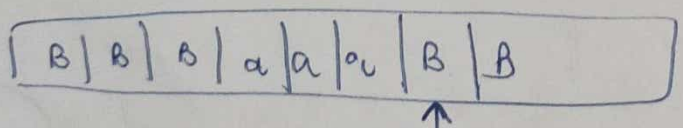


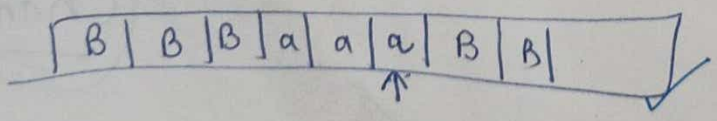


eg. Time0 

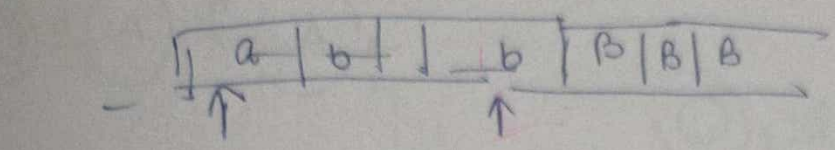
Time1 

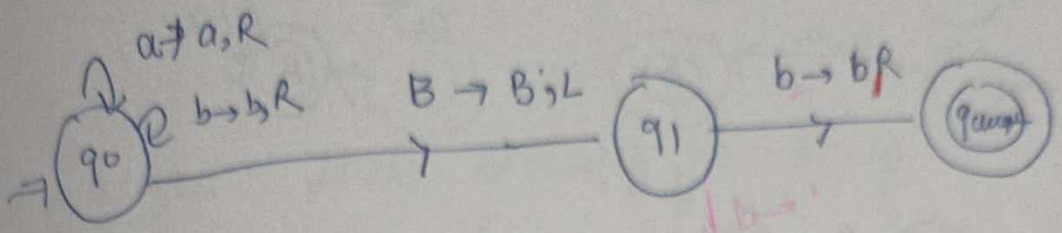
Time 2 

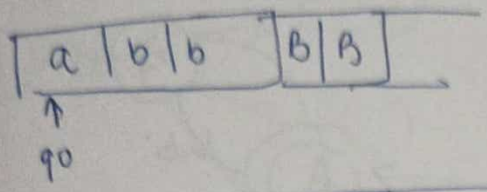
Time3 



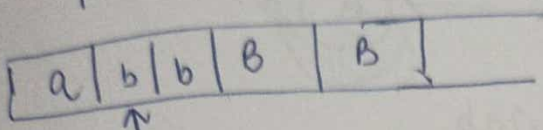
Q set of all strings over a, b ending with b.



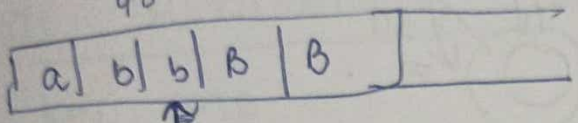




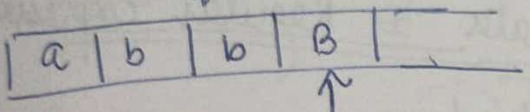
q0 abb B



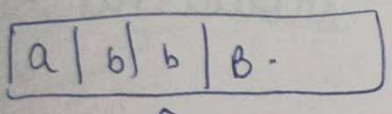
a q0 bb B



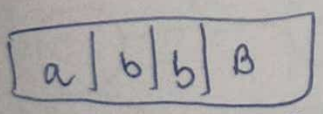
a b q0 b B



a bb q0 B

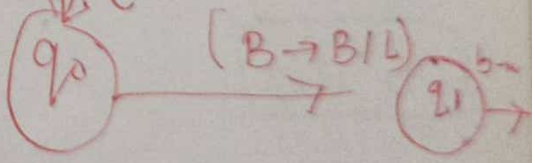


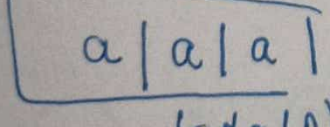
a b q1 bB

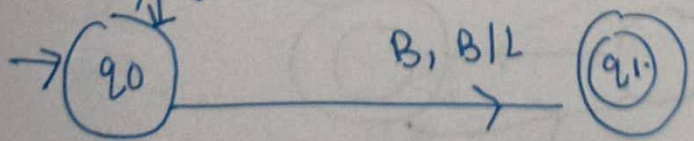


a b b qaccept B.

(a, a/R)
(b, B/R)

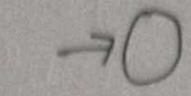


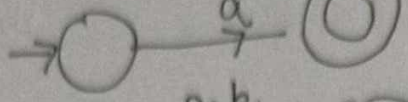


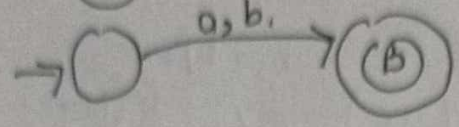


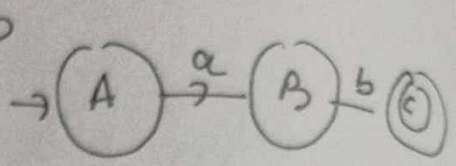
Regular Expression to FA

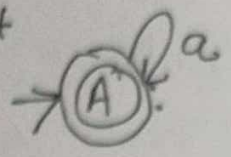
State Creation method

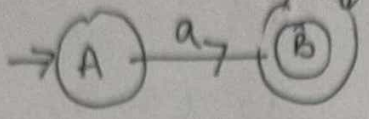
ϕ 

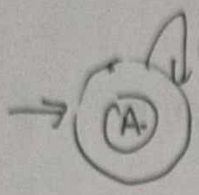
a 

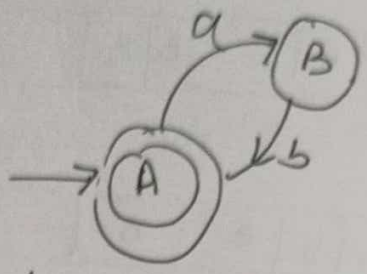
a+b 

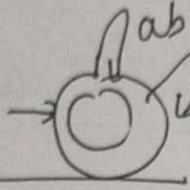
ab 

a^* 

ab^* 

$(ab)^*$ 

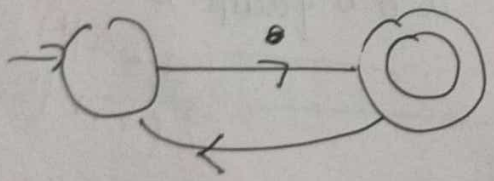


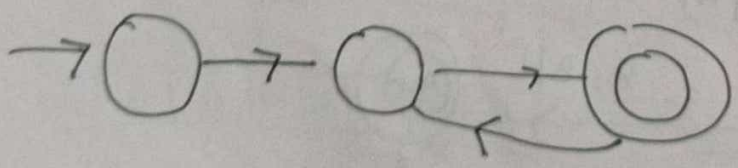
~~ac~~$^*(ab+ba)^*$ 

Finite Automata to Regular Expression

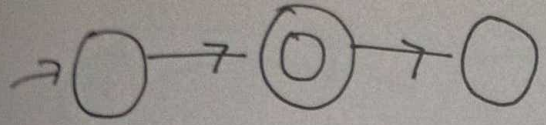
State Elimination Method

(i) ~~Incomi~~ Initial state should not have any incoming edge

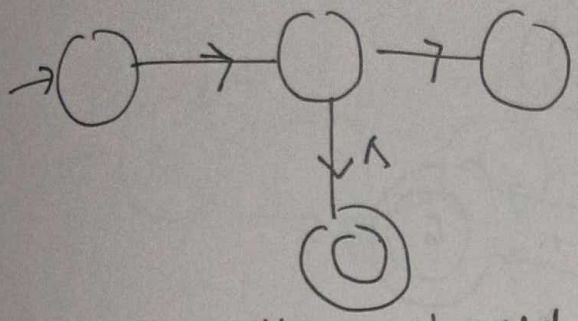




(ii) There should not be any outgoing edge to final

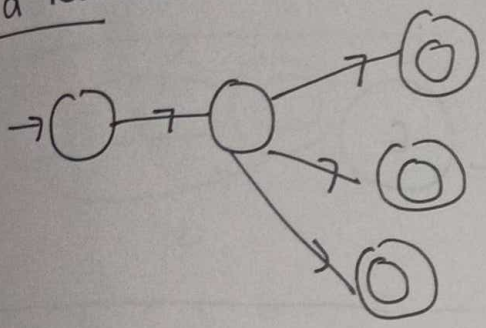


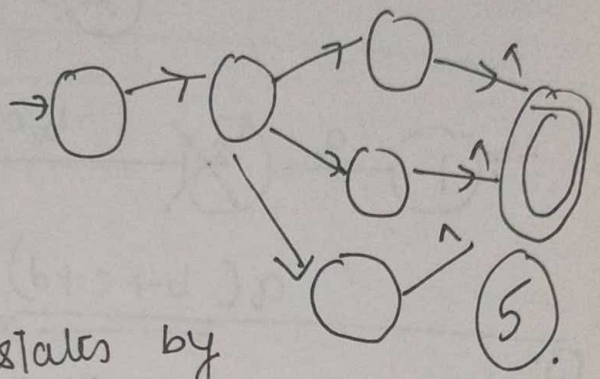
Final state should not having any outgoing edge.



1:30

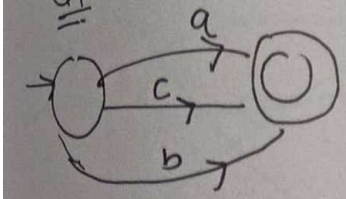
Third Rule: There should be only one final state

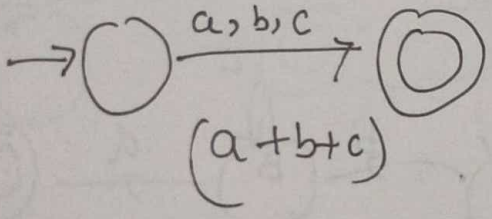




4) Eliminate remaining states by one by one

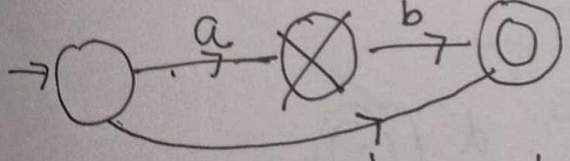
Q1



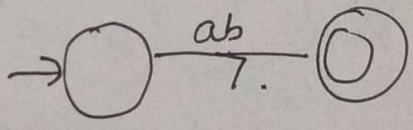


(a+b+c)

Q2

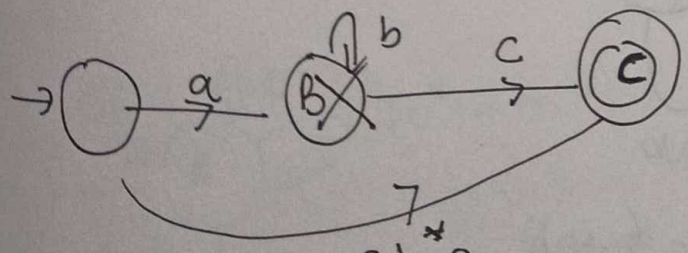


ab.



ab.

Q3

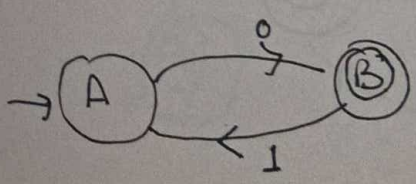


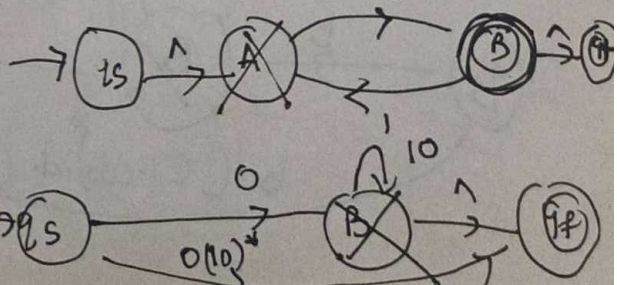
ab^*c.

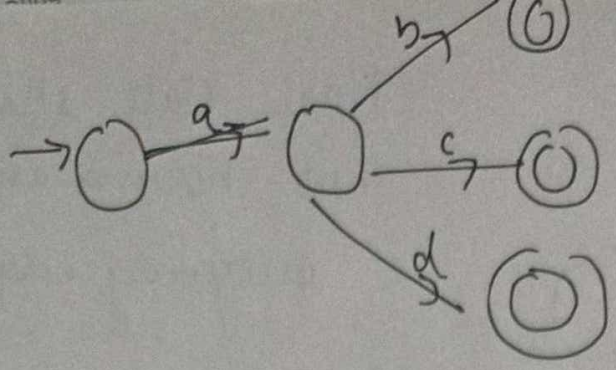
ab^*c

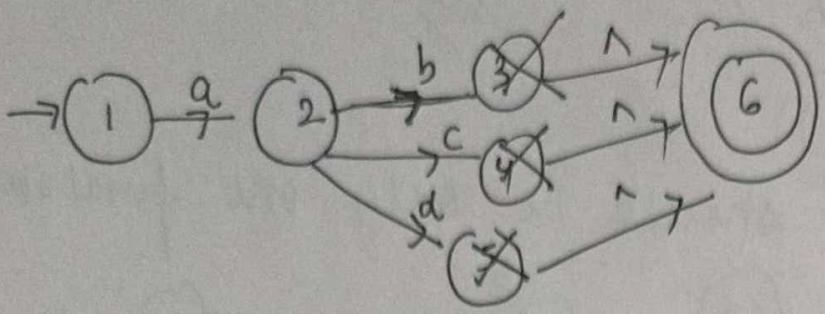
$0(10)^*$

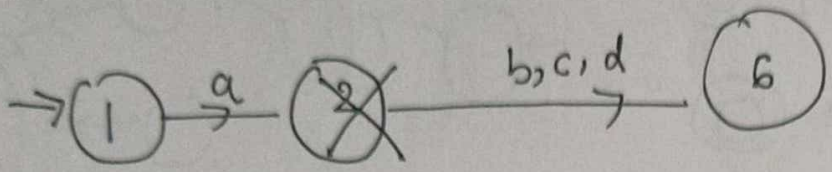
Q4





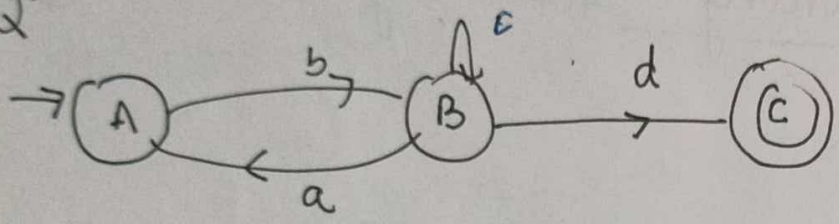

a
b
c
d

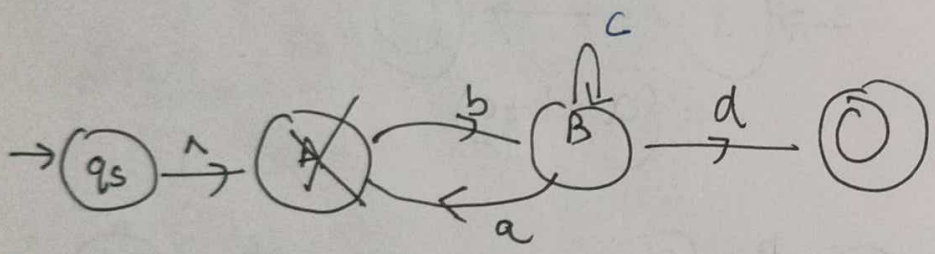

1
a
2
b
c
d
3
4
5
^
^
^
6

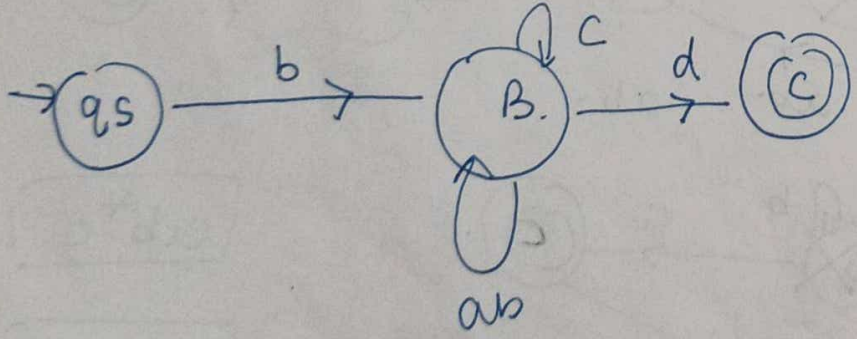

1
a
2
b, c, d
6

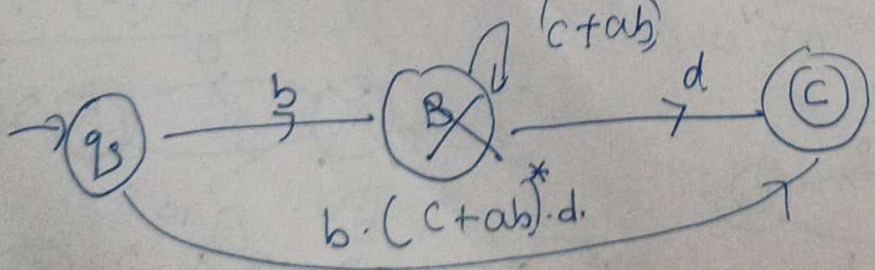
a(b+c+d)

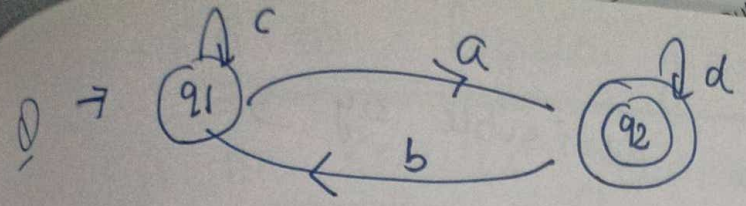
Q.

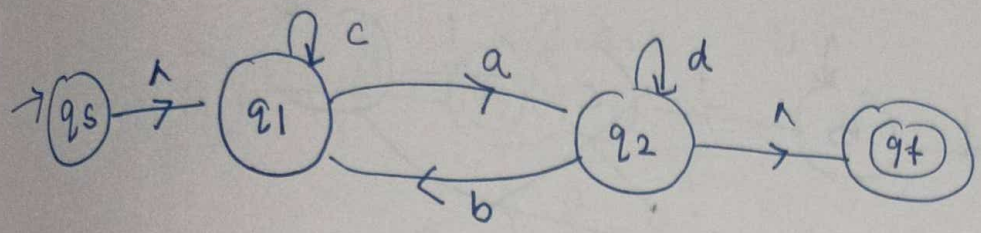

A
b
B
c
d
C
a

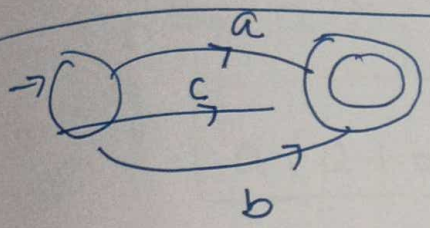

qs
^
A
b
B
c
d
a


qs
b
B.
c
d
C
ab

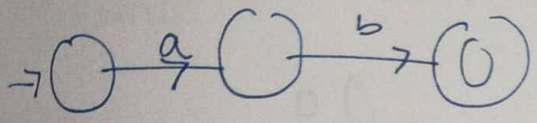

qs
b
B
(c+ab)*
d
C
b.(c+ab)*.d.



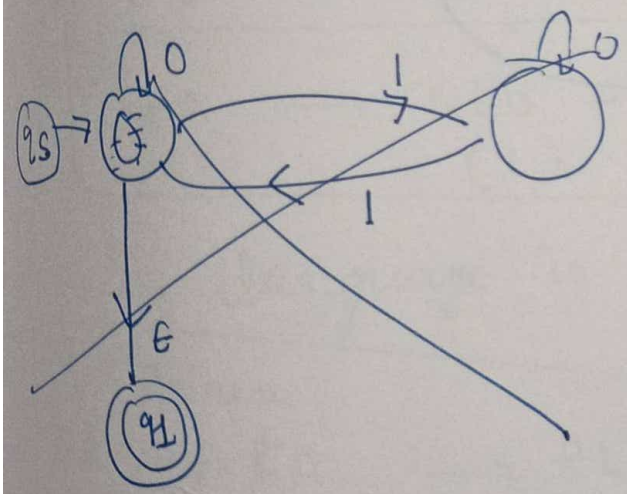


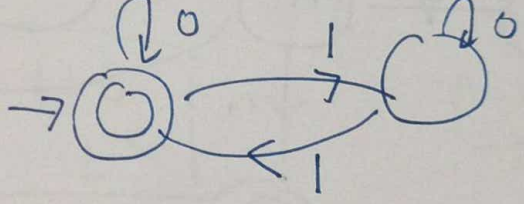


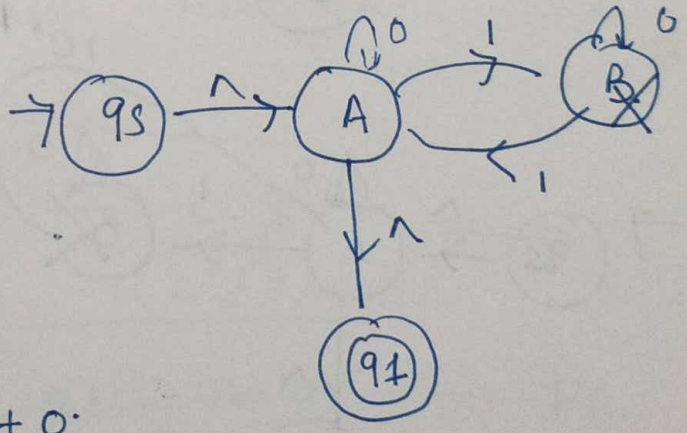
a+b+c

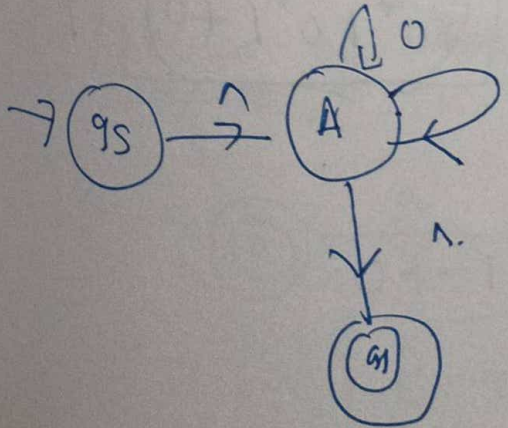


→ ab.







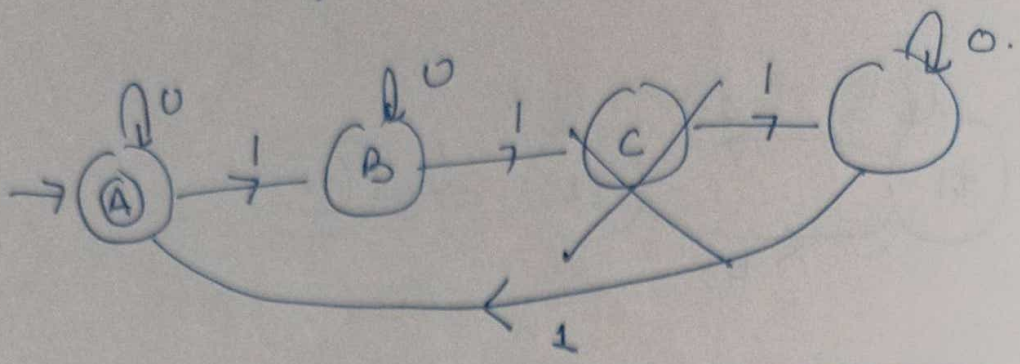


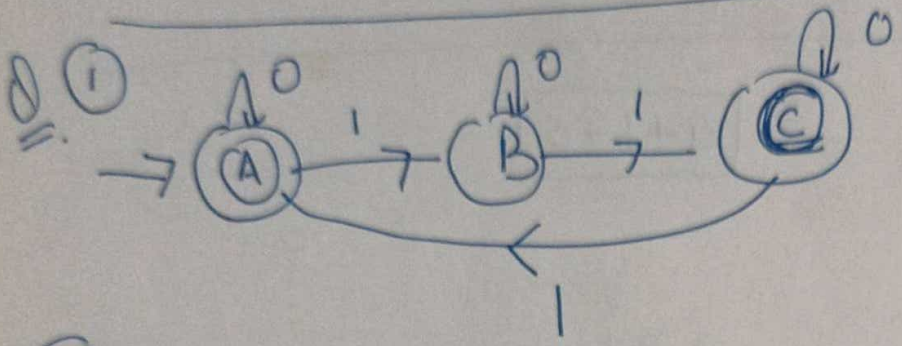
10^*1+0.

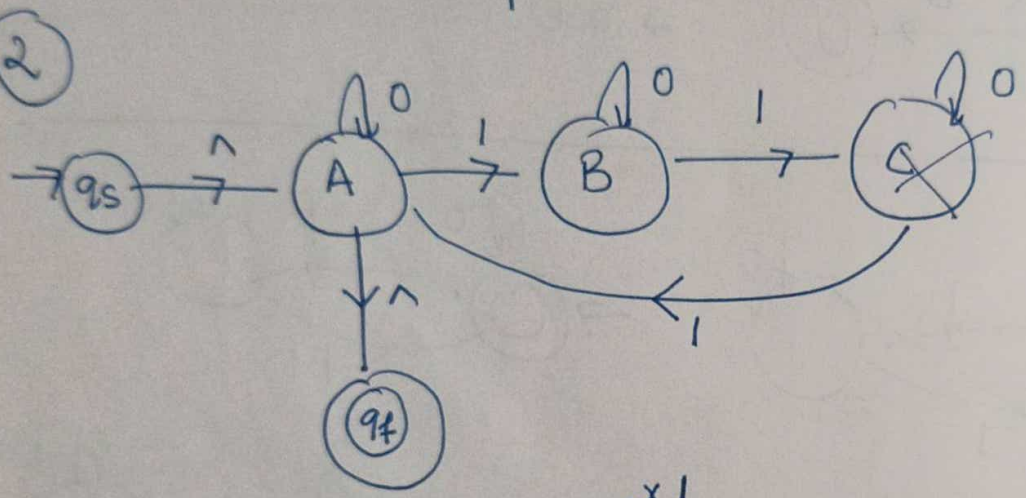
$(10^*1+0)^*$.

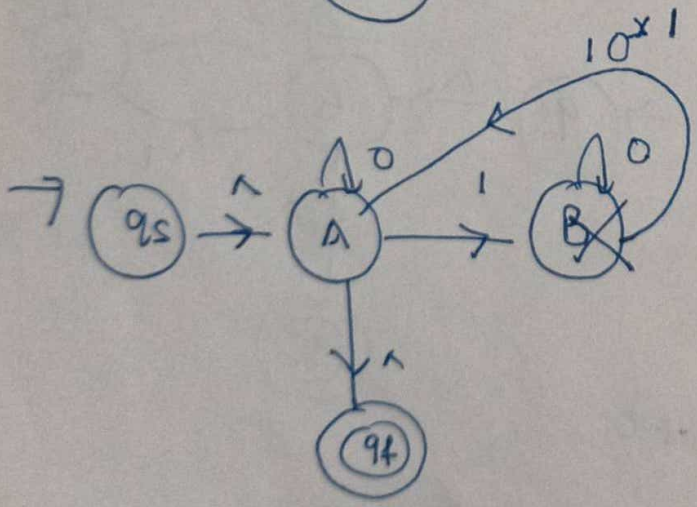
101 011

No's of 1 are divisible by 3

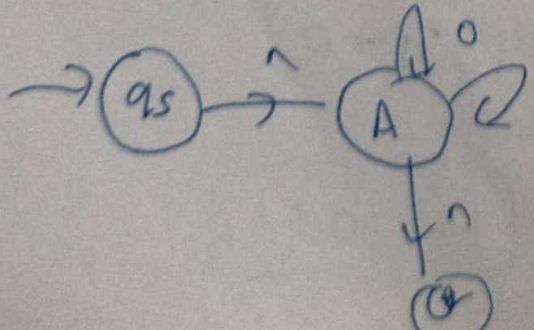








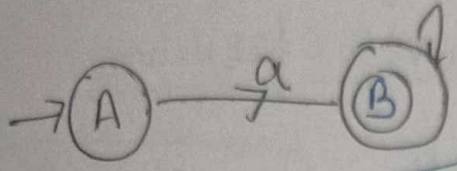
$(10^*10^*1+0)^*$.



10^*10^*1+0.

FA to Regular grammar

FA to Regular grammar

1) 

RLG

$A \to aB$

$B \to aB \mid bB \mid \wedge$

Right linear grammar

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

Convert with LLG

(L^R)

Reverse ending with a

$A \to Ba$

$B \to Ba \mid Bb \mid \wedge$

If we have finite auto

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

language is also reversed.

$A \to Ba$

$A \to a$

$A \to Ba$

$\to aBa$

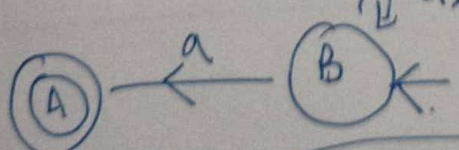
$\to aa$.

Reverse

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

(V) Reverse of finite automata → FA.

L^R



RLG

$B \to aB \mid bB \mid aA$

$A \to \wedge$

$B \to Ba \mid bB \mid Aa$

$A \to \wedge$

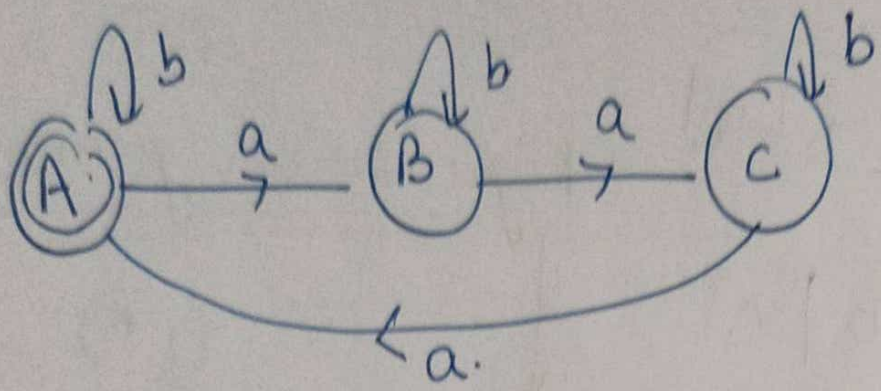
$\Longrightarrow (L^R)^R$

→ LLG.

RLG into FA (directly convert)

$A \to aB \mid bA \mid b$ $\quad B \to aC \mid bB$

$C \to aA \mid bC \mid a$

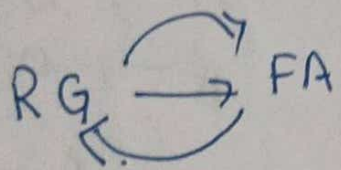


→ No. of a's divisible by 3

~~Wrong going to~~ end.

given LLG into Finite Automata

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



Reduction of grammar simply means the simplification of grammar by the elimination of unnecessary symbols (terminals or non-terminals)

Conditions for Reduced grammar

(1) It does not contain null productions.
$A \to \wedge$ i.e. $A \to \wedge$ or $S \to \wedge$.

(2) It does not contain unit productions.
$A \to B$ i.e. $A \to B$ where A & B are non terminals.

(3) It does not contain useless symbols.

1. Elimination of $\wedge$-productions
2. Elimination of unit productions
3. Elimination of useless symbols.

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

(1) Elimination of $\wedge$-productions $A \overset{*}{\Rightarrow} \wedge$.

$A \overset{*}{\Rightarrow} \wedge$ are called $\wedge$-productions or null

A variable or non-terminal A is said to be nullable if it derives $\wedge$ in zero or more steps

K.L.P. Mishra

eg (1) $S \to A$
$A \to \wedge$

$\wedge$-production

eg: $S \to A$
$A \to \wedge$

Nullable variables : A, S

Nullables
A, S!

2 marks

eg
$S \to AB$
$A \to \wedge$
$B \to \wedge$

Nullable variables: A, B, S.

(iii)
$S \to ABC$
$A \to \wedge$
$B \to \wedge$
$C \to C$

Steps to eliminate Null. Remove

Nullable variable:→ A, B.

Step 1 Find all the nullable variables which directly or indirectly ~~int~~ derives $\wedge$

Step 2: Create two versions of all the productions containing nullable variable (s) on their R.H.S, one without that nullable variable and other with that nullable variable

Remove the $\wedge$- productions

Step 3 Productions whose R.H.S does not have any nullable variable are included directly.

$\to \wedge$

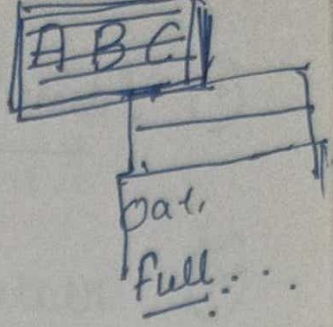
$S \to AB$
$A \to aAA \mid \wedge$
$B \to bBB \mid \wedge$

$S \to ABC$

$A \to \wedge$

$B \to \wedge$

$C \to C$



(1) A, B → nullable

(2) $S \to ABC \mid BC \mid AC$

Sb Consider the following grammar

$S \to aS \mid bS \mid \wedge$

Eliminate $\wedge$- productions, if any

Sdn The given grammar is

$S \to aS$

$S \to bS$

$S \to \varepsilon$

(1) Nullable variable ⇒ S.

(2) Two versions of production for $S \to aS$ ABC

$S \to aS$ (with S) $\Rightarrow S \to aS \mid a$

$S \to a$ (without S)

For $S \to bS$

$S \to bS$ (with S) $\Rightarrow S \to bS \mid b.$

$S \to b$ (without S)

The complete production becomes

$S \to aS \mid bS \mid a \mid b.$

---

(3) $S \to AB$

$A \to aAA \mid \wedge$

$B \to bBB \mid \wedge$

① Identify the nullable variables:-

Nullable variables are :- S, A, B.

② Create two versions of all productions having nullable variables

$S \to AB$

$A \to aAA \mid \wedge$

$B \to bBB \mid \wedge$

- $S \to AB$

For considering A

$S \to AB$

$S \to B$

$S \to AB \mid B$

Two versions of this production are

$S \to AB$ (with A).

$S \to B$ (without A).

$\Rightarrow S \to AB \mid B.$

For considering B

$S \to AB$

$S \to A$.

Two versions of this productions

$S \to AB$ (with B)

$S \to A$ (without B)

$\Rightarrow S \to AB \mid A.$

Considering both A & B

$S \to AB$ (both)

$S \to \wedge$ (without $\wedge$)

$S \to AB$.

$S \to AB$.

$S \to AB \mid A \mid B.$

Step For production $A \to aAA$

→ Considering first A

Two versions

$A \to aAA$

$A \to aA$ (without)

$\Rightarrow A \to aAA \mid aA$

Considering second A

Two versions of this production are

(i) $A \to aAA$

(ii) $A \to aA$

$A \to aAA \mid aA$.

Considering both

$A \to aAA$

$A \to a$

$\Rightarrow A \to aAA \mid a$

$A \to aAA \mid aA \mid a$

For production $B \to bBB$. $bBB \mid bB \mid b$.

$B \to bBB$

$B \to bB$

$\Rightarrow B \to bBB \mid bB \mid b.$

971.

$S \to AB \mid A \mid B$

$A \to aAA \mid aA \mid a$

$B \to bBB \mid bB \mid b$

Q3 Consider the following grammar:

$S \to aS \mid AB$

$A \to \Lambda$

$B \to \Lambda$

$D \to b.$

Ans The nullable variables are A, B, S

$S \to aS \mid a$

For

$S \to aS$

$S \to a$

$\Rightarrow S \to aS \mid a$

$S \to AB$

$S \to AB$

$S \to B$

$S \to AB$

$S \to A$

For $S \to AB$

$S \to AB \mid A \mid B$ $\Rightarrow S \to AB \mid A \mid B.$

$S \to aS \mid a \mid AB \mid B \mid A$

$D \to b,$

Eliminate

$S \to ABAC$

$A \to aA \mid \wedge$

$B \to bB \mid \wedge$

$C \to c$

$S \to ABAC$

$A \to aA \mid \wedge$

$B \to bB \mid \wedge$

$C \to c$

Sol Nullable variables :- A, B

(1) For $S \to ABAC$

considering first A

$S \to ABAC \mid BAC$

considering second A

$S \to ABAC \mid ABC$

considering Both.

$S \to ABAC \mid BC$ $\Rightarrow$ $S \to ABAC \mid BAC \mid ABC \mid BC$

$AAC \mid AC \mid C$

considering

$S \to ABAC \mid AAC$

~~$S \to AB$~~

$A \to aA \mid a$

$B \to bB \mid b$

$C \to C$

Elimination of unit productions - substitution

A production in which one non-terminal derives another non-terminal is called unit production.

$A \to B$ $\quad A \to B$.

Step1 Ensure that the given grammar have no null productions. If any, first eliminate them.

Step2 Find out all the unit productions

Step3 eliminate unit production by substitution method

Q1

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

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

Null
Unit
Useless

Ans (1) No null productions

(2)

$B \to C$
$C \to D$
~~$D \to b$~~

Bottom to top

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

(3)

$C \to b$
$B \to b$.

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

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

$C \to D$
$D \to b$
$C \to b$

Q2 Eliminate unit productions

S → AB
A → a
B → C | b
C → D
D → E | bC
E → d | Ab

Step1 Clearly there are no Λ-productions

Step2 the unit productions are

B → C
C → D
D → E

B → C → D → E

E → d | Ab

Step3

S → AB
A → a

D → d | Ab | bC

C → d | Ab | bC

B → d | Ab | bC | b

D → bC | d | Ab

C → bC | d | Ab

Q3

S → Aa | B
B → A | bb
A → a | bC | B

A → B can be subs.

⇒ A → a | bC

B → A

Ans unit productions

S → B
B → A
A → B

(1) A → A | bb ⇒ A → bb

(2) B → bb | a | bC B → bb | a | bC

$S \to Aa \mid bb \mid a \mid bc.$

$A \to a \mid bc \mid bb.$

Q $S \to A \mid bb$

$A \to B \mid b$

$B \to S \mid a$

Step 1- no null productions.

Step 2- The unit productions

$S \to A$

$A \to B$

$B \to S$

Step 3

$B \to bb \mid a$

$A \to bb \mid a. \mid b$

$S \to bb \mid a.$

$S \to A$

$A \to BC$

$A \to a.$

$S \to a.$

Elimination of useless productions

A variable is said to be useful if and only if it satisfis the following conditions:-

(1) It can derive a terminal string

(2) It can be reachable from the start symbol.

Useless symbols → if any of these condn fail then symbol becomes useless
The production involving any useless symbol is called useless production.

$S \to A \mid C$
$A \to a \mid B$
$B \to b$

$S \to A \mid C$
$A \to a \mid B$
$B \to b$.

Useful symbols → ~~S, A, B~~. S, A, B

Useless symbol :- C, ~~B~~.

$S \to C$.

(1) Remove the null & unit production, if any

(2) For all the non-terminals, check if a terminal string can be generated directly or indirectly. If no, then remove that symbol & its associated productions, otherwise keep them.

(3) For all the left out non-terminals, check if they can be reached from the start symbol directly or indirectly. If no, remove that symbol & its associated productions otherwise keep them

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

$S \to AB \mid C$
$A \to a$
$B \to b$.

Soln :→ No Null productions & unit productions.

Step 2 :- Identify the non-terminals

S, A, B, C

S satisfies
A - derive a terminal string
B → derive a terminal string

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

(All the non terminals generate terminal string but C is not deriving any terminal string since there are no productions for C

Step 3 S, A, B.

Left out non-terminals
S, A, B.

The variable A & B can be reached from the start symbols.

The variables S, A, B are useful symbols.

useful

$S \to AB$
$A \to a$
$B \to b$.

Q2 $S \to AB \mid CA$

$B \to BC \mid AB$

$A \to a$

$C \to aB \mid b$

$S \to AB \mid CA$

$B \to BC \mid AB$

$A \to a$

$C \to aB \mid b.$

$S \to *S$

$S \to (S*S)$

$S \to 1$

$S \to 2$

$C \to *$

$C_1 \to ($

$C_2 \to)$

$S \to CS$ ✓

$S \to C_1 S C S C_2$

$S \to 1$

$S \to 2.$

S.

Useless symbols

CNF

Greibach

Pumping

for CFG.

$S \to aAD$

$A \to aB \mid bAB$

$B \to b$

$D \to d$

$S \to C_a AD$

$A \to C_a B \mid C_b AB$

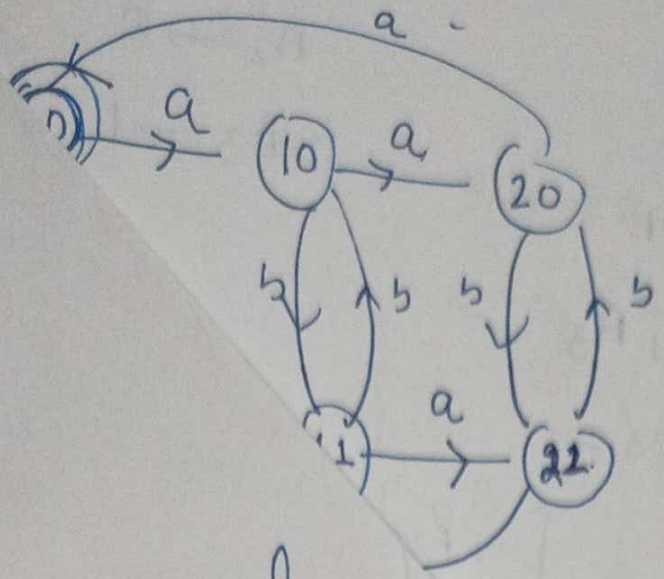
$C_a \to a$

$C_b \to b$

Q. DFA $w \in \{a,b\}^*$

$n_a(w) \equiv 0 \mod(3)$

$n(b) \equiv 0 \mod 2.$



$n_b(w) \mod 2.$

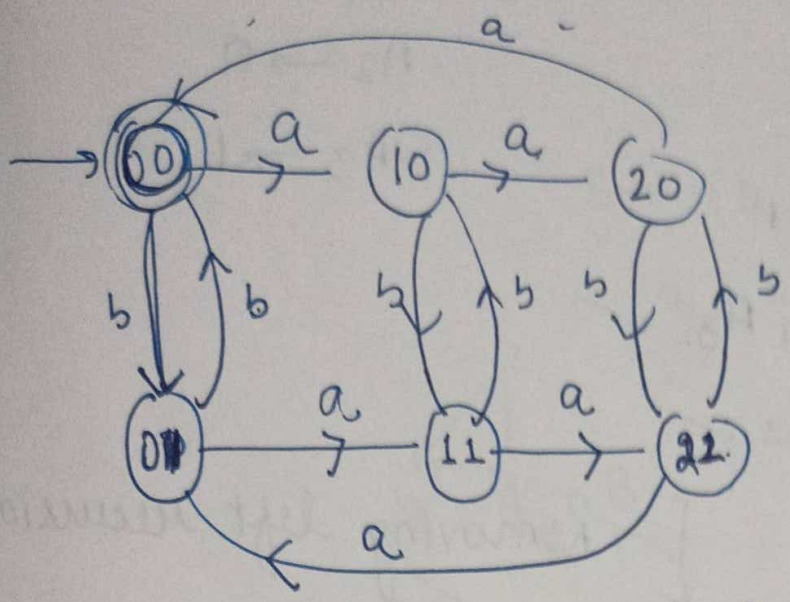
$w) \mod 2.$

Theory of Computation

8. DFA $w \in \{a,b\}^*$

$n_a(w) \equiv 0 \mod(3)$

$n(b) \equiv 0 \mod 2.$



$n_a(w) \mod 3$ / $n_b(w) \mod 2$.

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

$n_a(w) \mod 3 = n_b(w) \mod 2.$

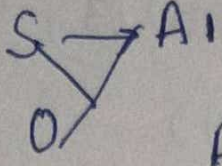
$\{00, 11\}$

$S \to SS \mid S \to 0S1 \mid 01$

$S \to SS$

$S \to 0S1$

$S \to 01$



$A_1 \to S$

$A_2 \to 0$

$A_3 \to 1$

$S \to A_1 \to A_1A_1$

$A_1 \to 0A_1A_3$. ✓

$A_1 \to 0A_3$ ✓

$A_1 \to A_1A_1$ [Removing left recursion]

$A_1 \to \underbrace{A_1A_1}_{\beta_1} \mid \underbrace{0A_1A_3}_{\alpha_1} \mid \underbrace{0A_3}_{\alpha_2}$.

$A_1 \to 0A_1A_3 \mid 0A_3 \mid 0A_1A_3Z \mid 0A_3Z$.

$Z \to A_1 \mid A_1Z$.

$Z \to 0A_1A_3 \mid 0A_3 \mid 0A_1A_3Z \mid 0A_3Z$

$Z \to 0A_1A_3Z \mid 0A_3Z \mid 0A_1A_3ZZ \mid 0A_3ZZ$.

Ambiguity in CFG

$G = (V, T, P, S)$

$S \to AB \mid aaB$

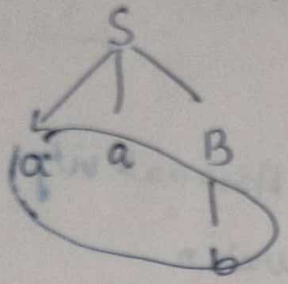
$A \to a \mid Aa$

$B \to b$

$w = aab$

$S \to aaB$

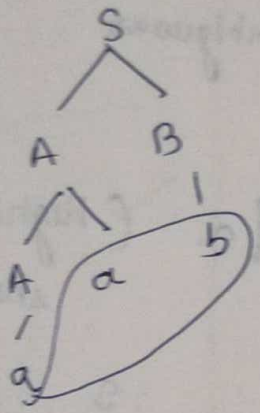
$\to aab$



$S \to AB$

$\to AaB$

$\to aab.$



(2) $A \to aA \mid aa \mid a$ $\quad w = aaa$

$S \to A$

$S \to A \to aA$

$\to aaa$

$S \to S+S \mid S*S \mid a$

(3) $S \to a+a*a$

$S \to S+S$

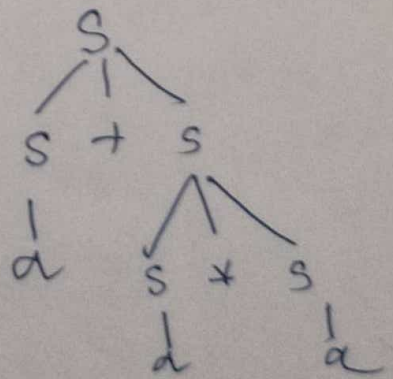
$\to a+S*S$

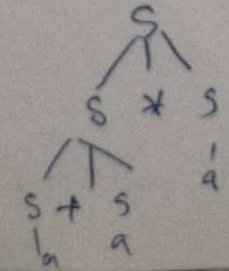
$\to a+a*a$

$S \to S*S$

$\to S+S*S$

$\to a+a*a$





(1) $S \to S+S \mid a$ Associativity Problem. → Recursion Left - Right.

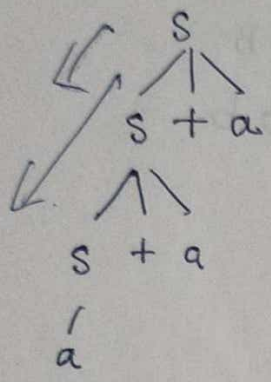
(2) $S \to S+S \mid S*S \mid a$

Precedence → Level

(1) $w = (a+a)+a$

$= a+(a+a)$

$S \to S + a \mid a$



Left Associativity Recursion

unambiguous

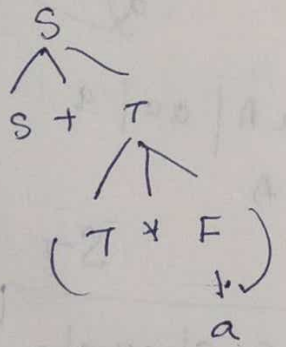
(2) $S \to S+S \mid S*S \mid a$ (Highest precedence should be at last level)

$a+(a*a)$

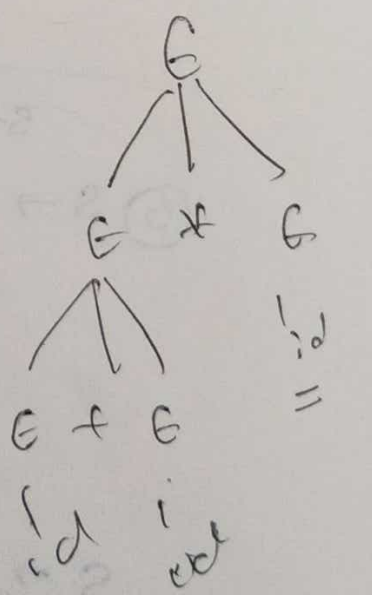
$S \to S+T \mid T$

$T \to T*F \mid F$

$F \to a$



ambiguous



Ambiguity

SA ← G

G = (V, T, P, S)

E → E+E
→ E*E
→ id

∀α

V = {E}

T = {id, +, *}

parser get confuse

E → E+E
↓
E+E*E
↓
E+E*id
↓
E+id*id
↓
id+id*id

Left Most derivation

(1) LMD

E → E+E
→ id + E*E
id + id * Ed
id + id * id

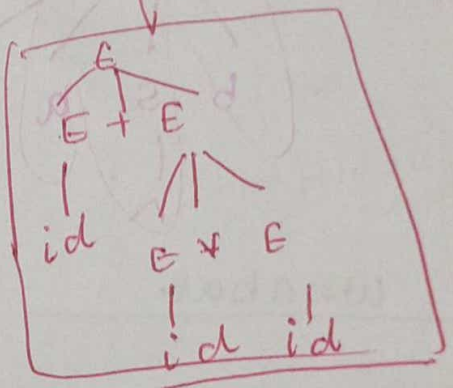
E → E+E
↓
id + E
↓
id + E*E
id + id * E
id + id * id

(2) LMD Left Most

E → E * E
→ E+E * E
→ id + E * E
→ id + id * E
⇒ id + id * id
⇒ id * id * id

(1) RMD

E → E+E
→ E + E * E
→ E + E * id
→ E + id * id
→ id + id * id



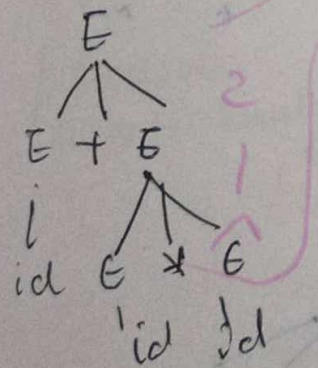
(2)

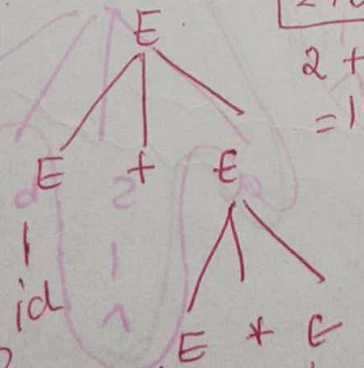
E → E * E
→ E * id
→ E+E * id
→ E + id * id
⇒ id + id * id

2+3*4

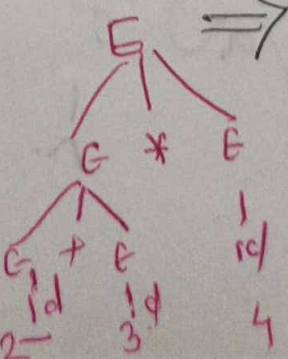
2 + 12
= 14

Parse Tree





right



2+3*4
5*4
= 20

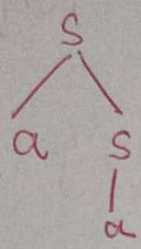
we got two right most, left most

$S \to aS \mid Sa \mid a$

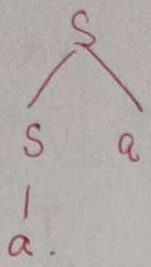
$w = aa$

$S \to aS$

$\to$

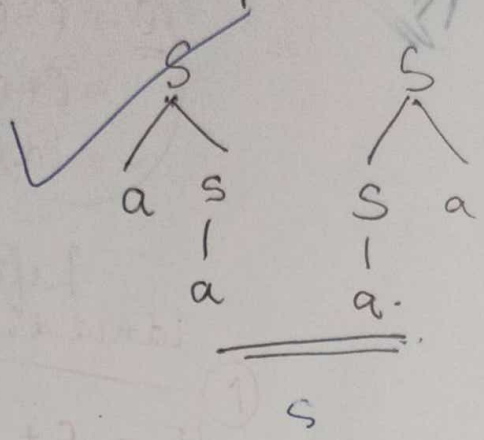


more than one parse tree



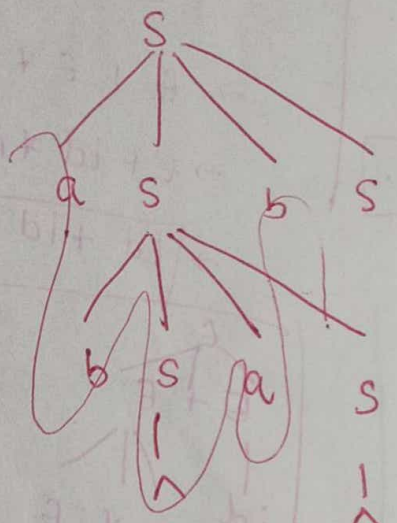
Using practice $\to$ no algorithm

$S \to aS \mid Sa \mid a$



$S \to aSbS \mid bSaS \mid \wedge$

$w = abab.$



$S \to aSbS$

$\to abSaSbS$

$\to \boxed{abab}$

$S \to aSbS$

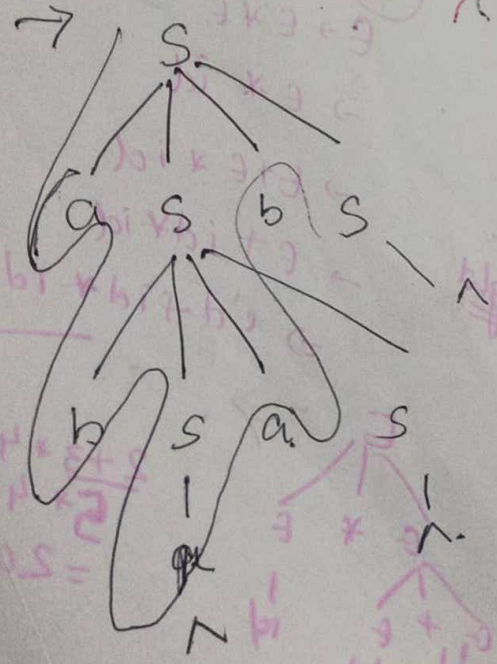
$\to aSbabSbS$

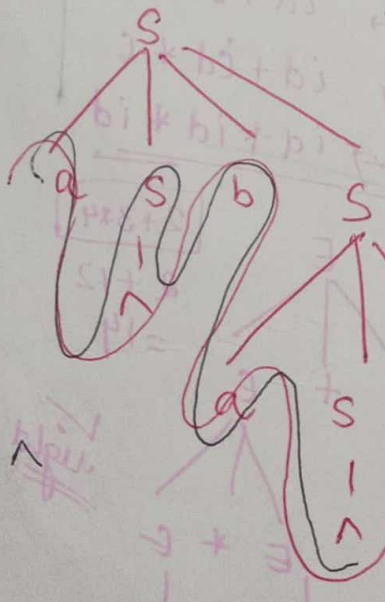
$\to \boxed{abab}$

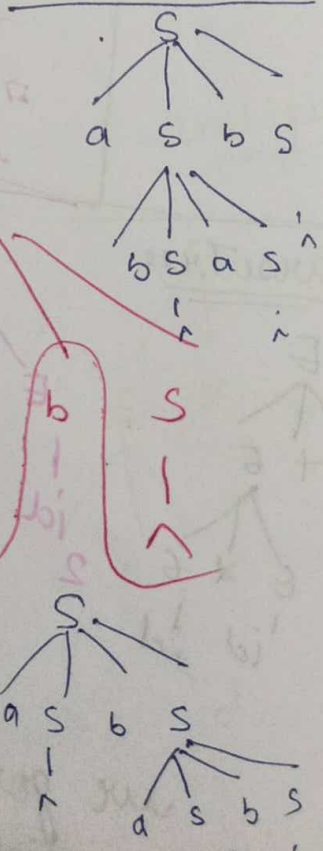
$w = abab.$

$S \to aSbS$

$\to$

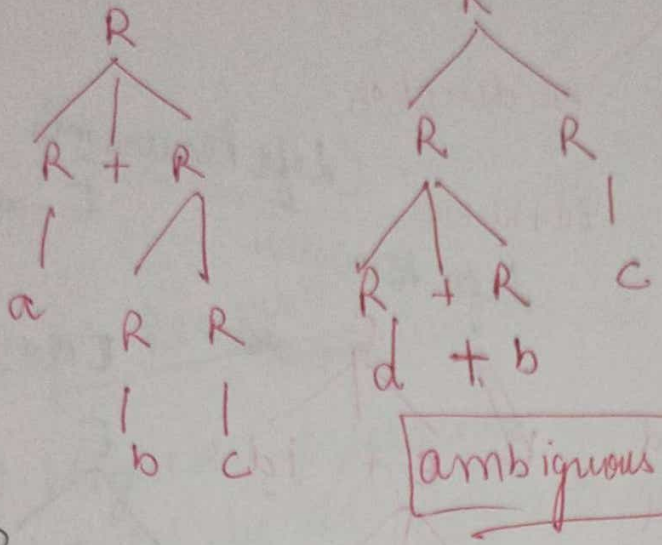




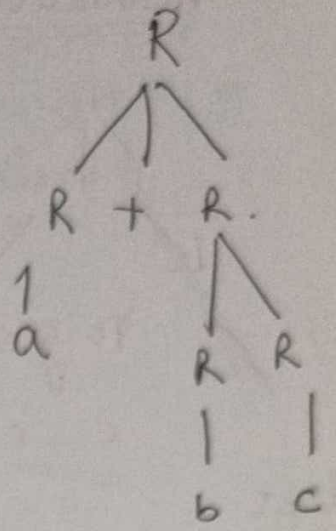


$R \to R+R \mid RR \mid R^* \mid a \mid b \mid c$

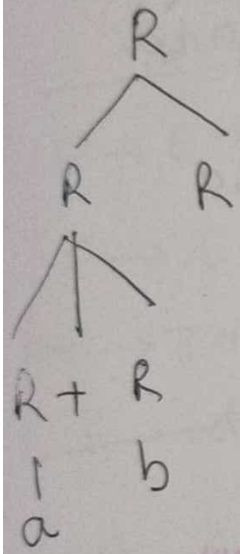
a+bc



ambiguous grammar



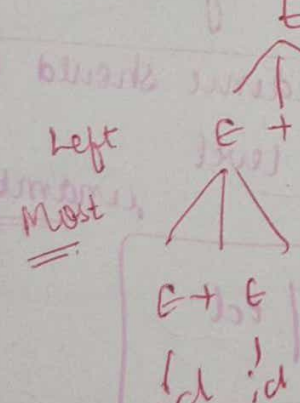
(grammar is ambiguous)



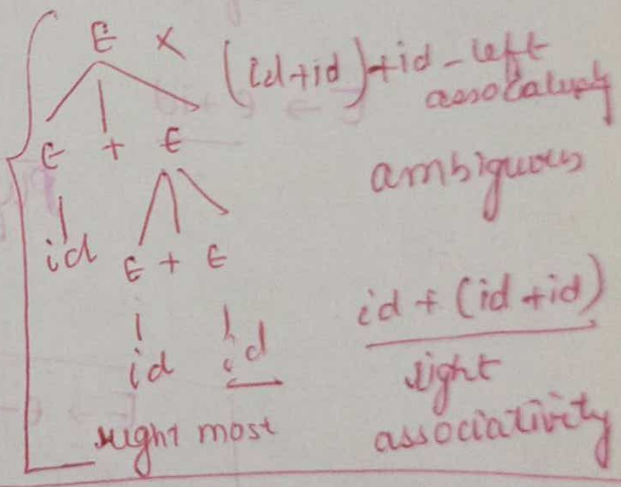
$E \to E+E \mid E*E \mid id$

id+id+id

Left Most



Operators which operate from left to right



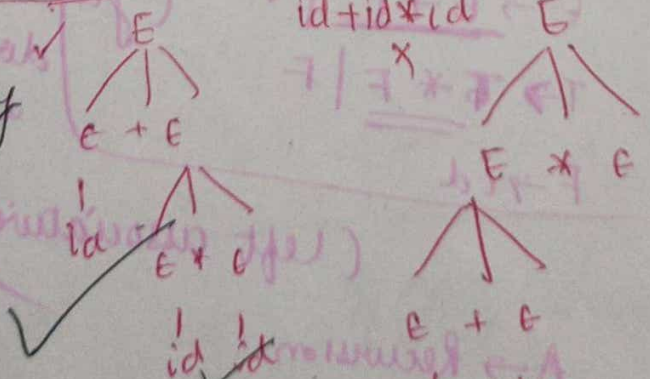
(id+id)+id - left associativity

ambiguous

id + (id + id)

right associativity

precedence associativity



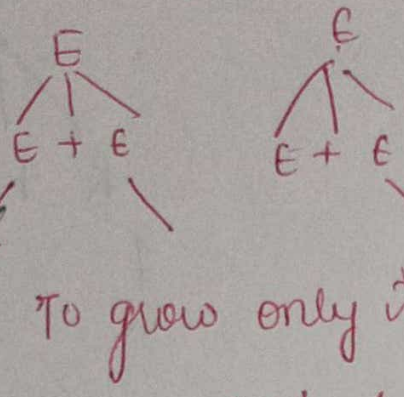
id+id*id

X

* having highest

+ is evaluated first

Associativity → Left Recursive

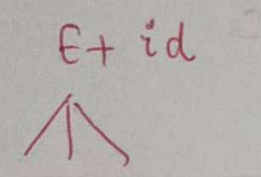


To grow only in one direction

$E \to E+id \mid id$ (Left Recursive)

id+id+id

Left Associativity



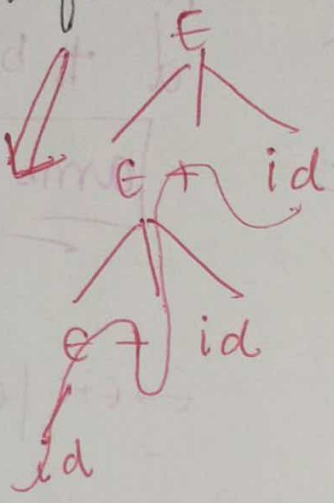
E+id +id

id+id +id

no way

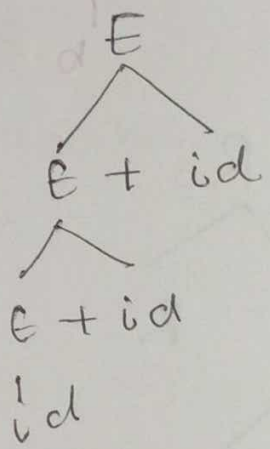
operator should be left recursive

Left Associativity



$E \to E+id$ /id

id+id+id



$E \to E+id$ Left most by = R.H.S.

Highest precedence should be at ~~lowest~~ lower level

unambiguous

$E \to E+E \mid E*E \mid id$

$E \to E+T \mid T$

$T \to T*F \mid F$

$F \to id$

(grammar does not contain any + (Left associative)

A → Recursion

precedence → levels

$2^{(3^2)}$ $2 \uparrow (3 \uparrow 2)$

$E \to E + T \mid T$
$T \to T * F \mid F$
$F \to G \uparrow F \mid G$
$G \to id$
(Right associativity)

$E \to E + T \mid T$
$T \to T * F \mid F$
$F \to G \uparrow F \mid G$
$G \to id$
(Right associativity)

+, *, ↑

Left Recursive

Boolean Expression → b Exp or b Exp
b Exp and b Exp
not b Exp
// True terminals
// False.

(2) Highest precedence should be at lower level.

$E \to E + E$	$E * E \mid id$
$E \to E + T \mid T$	
$T \to T * F \mid F$	
~~$F \to id$~~	

Not, are highest
~~AND~~
OR

Not ~~least~~ Highest
AND
OR ~~highest~~ lowest

A → Recursion
P → Level

$E \to E$ or $F \mid F$ (left)
$F \to F$ and $G \mid F$ (left)
$G \to$ Not $G \mid$ True $\mid$ False (

$R \to R+R \mid RR \mid R^* \mid a \mid b \mid c$

$E \to E + T \mid T$
$T \to TF \mid F$
$F \to F^* \mid a \mid b \mid c.$

unambiguous

$2^{3^2} \to 2 \uparrow (3 \uparrow 2)$ → Right Associative

$A \to A \$ B \mid B$ Left

$B \to B \# C \mid C$ Left

$C \to C @ D \mid D$ → Left Associative

$D \to d$

$\$ \; \$$ $\$ > \$$ Left in having

Left one

$\$ \; \$$ → Least

$\# > \#$ → Less

$@ > @$ → Higher

$\$ < \# < @$

$E \to E * F \mid F + E \mid F$ (* Left, + Right) — *, + have same precedence

$F \to F - F \mid id$ (undefined)

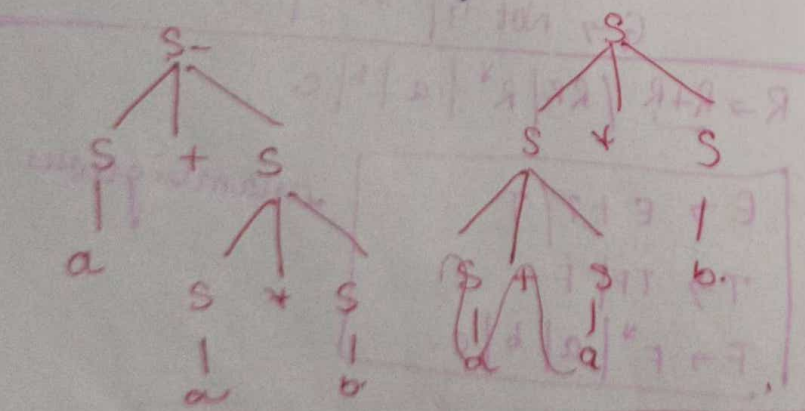
- is given high precedence

$* > *$

$+ \; \$ \; +$

$-$

Q $S \to S+S \mid S*S \mid a \mid b$ $a+a*b$



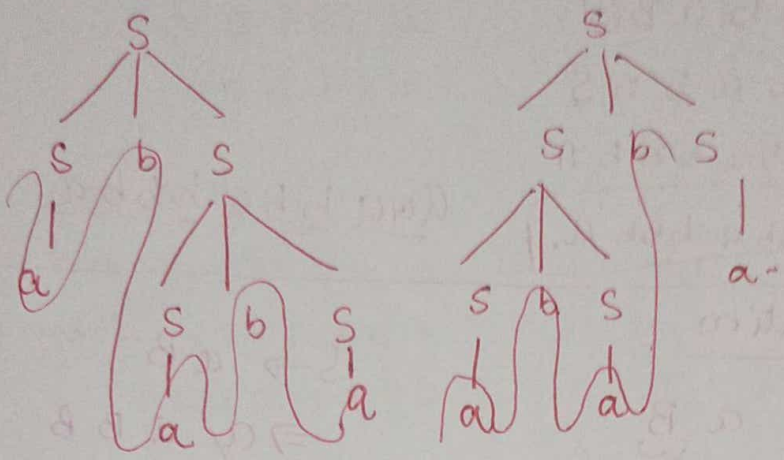
$S \to SbS \mid a$

$S \to SbS$

$\Rightarrow abSbS$

$\Rightarrow ababS$

$\Rightarrow ababa$



$S \to S+S \mid S*S \mid a \mid b$

$a*b+a*b$

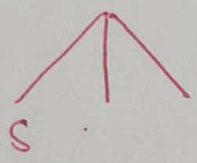
$S \to S+S$

$\Rightarrow S*S+S$

$\Rightarrow a*b+S*S$

$a+b+a*S$

$\Rightarrow a+b+a*b.$



Q. $S \to aB \mid bA$

$A \to aS \mid bAA \mid a$

$B \to bS \mid aBB \mid a.$

aaabbabbba,

Left derivation

Right derivation

$S \to aB$

$\Rightarrow aaBB$

$\Rightarrow aaaBB$

$\Rightarrow aaabBS$

$\Rightarrow aaabbAS$

$\Rightarrow aaabbaS$

$\Rightarrow a^3b^2abA$

$\Rightarrow a^3b^2abbAA$

$a^3b^2abbbAA$

$\Rightarrow a^3b^2abbba$

Sol^{n} $a^3 b^2 a b^3 a$ aaabbabbba

$S \to aB$

$\Rightarrow aaBB$

$aaaBBB$

$aaabbaBB$

$\Rightarrow aaabbabbbS$

$\Rightarrow aaabbabbbA$

$\Rightarrow aaabbabbba.$

aaabbabbba

Rightmost derivation

$S \to aB$

$\Rightarrow aaBB$

$\Rightarrow aaBbS$

$\Rightarrow aaBbbA$

$\Rightarrow aaBbbaS$

$\Rightarrow aaBbbabA$

$\Rightarrow aaBbbabAA$

$\Rightarrow aaBbbabbA$

$\Rightarrow aaBbbabba$

$S \to aB$

$\Rightarrow aaBB$

$\Rightarrow aaBaBB$

$\Rightarrow aaBaBbSS$

$\Rightarrow aaBaBbbA$

$S \to a \mid abSb \mid aAb$

$A \to bS \mid aAAb$

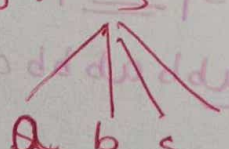
$S \Rightarrow abSb$

$\Rightarrow abaAbb$

$\Rightarrow ababSbb$

$\Rightarrow ababbabb$

ambiguous



$S \to abSb$

$S \Rightarrow aAb$

$\Rightarrow abSb$

$\Rightarrow abaSbb$

$\Rightarrow ababbabb$

$S \Rightarrow aB \mid ab$

$A \Rightarrow aAB \mid a$

$B \Rightarrow ABb \mid b$ is ambiguous

Imp.

$S \Rightarrow aB$

$\Rightarrow aABb$

$\Rightarrow aaABbb$

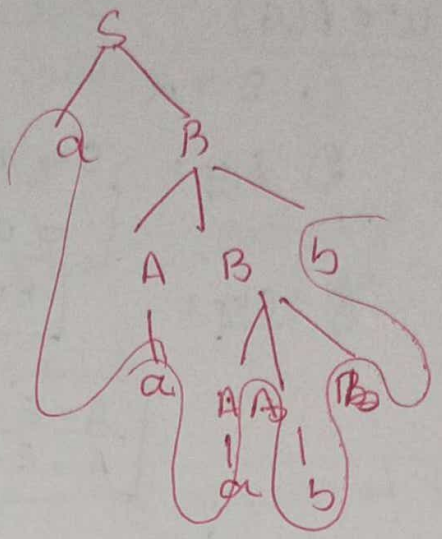
$\Rightarrow aaabbb$

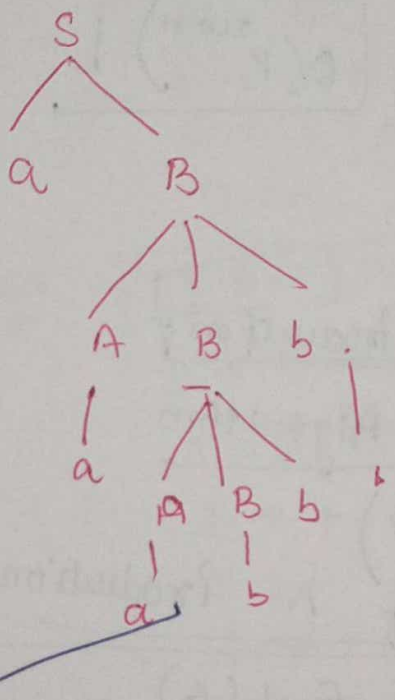
$S \Rightarrow aB$

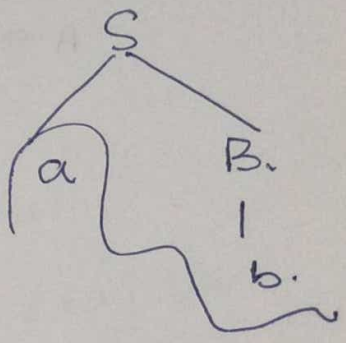
$\Rightarrow aABb$

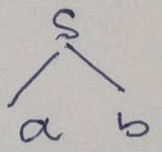
$\Rightarrow aaABbb$

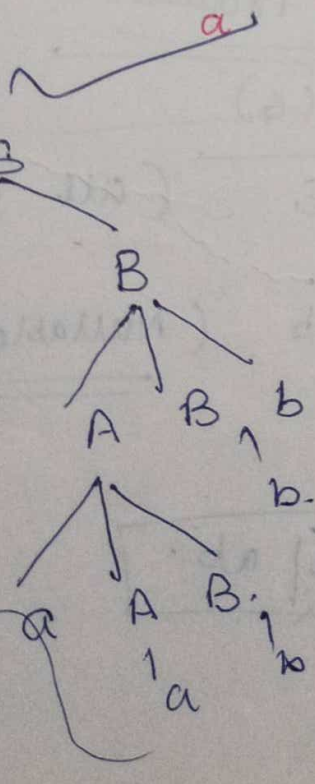
$\Rightarrow aaabbb$

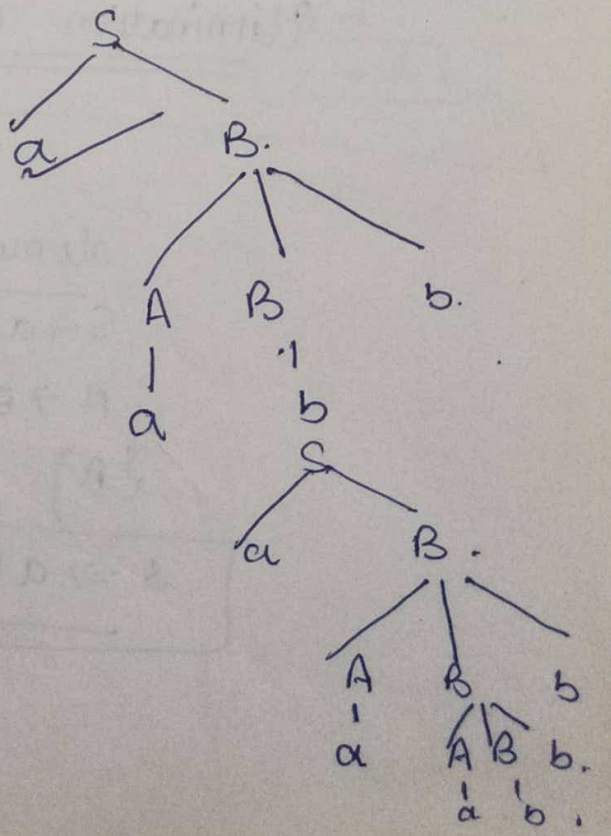


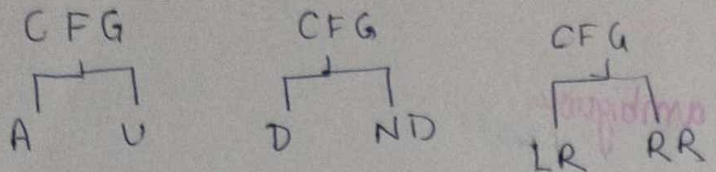












$w \in L(G)$

(1) $S \to \alpha$

(2) Steps $S \Rightarrow aAb$
$\Rightarrow abb$

(3) 3 Steps

KWL(G)

Brute Force Algo

$A \to \varepsilon$ ✗ (Null Productions)

$A \to B$ ✗ (Unit Productions)

$A \to BC$

$A \to a$

$P + P^2 + P^3 \dots P^{2w}$

$O(P^{2w+1})$

CFG

Null

Unit

Useless Productions

CYK – Algorithm

$O(|w|^3)$

Elimination of $\wedge$ – Productions

$\varepsilon \in L(G)$

$S \to \varepsilon$ (all can be eliminated)

$S \to aSb \mid aAb$

$A \to \varepsilon$

{A}

(Nullable Variables)

$S \Rightarrow aSb \mid aAb \mid ab$

(2) $S \to AB$

$A \to aAA \mid \wedge$

$B \to bBB \mid \wedge$

$\{A, B\}$ → Nullable Variables

$\{A, B, S\}$ → Nullable Variable

$\boxed{\wedge \in L(G)}$

$S \to AB \mid B \mid A \mid \wedge$ (only left hand side is ⇒)

$A \to aAA \mid aA \mid a$

$B \to bB \mid b$

---

(3) $S \to AbaC$ Elimination of Null adds Unit Productions

$A \to BC$

$B \to b \mid \wedge$

$C \to D \mid \wedge$

$D \to d$

Soln ⇒ $\{A, B, C\}$ are Nullables.

$S \to AbaC \mid baC \mid Aba \mid ba$

$A \to BC \mid B \mid C$

$B \to b$

$C \to D$

$D \to d$

$A \to B$

$B \to C$

$C \to d$

$\boxed{A \to d}$

Boolean Expression

bExp → bExp. or bEx (oring)
bExp and Exp. (Anding)
bExp not bExp Not
/True
/False

Highest Precedence
Not, AND, OR.

Left Associativity

E → E or F | F
F → F and G | G
G → Not G | True | False

Q. R → R+R | RR | R* | a | b | c

Soln Unbambiguous grammar

E → E + F | F
F → F * T | T
T → .

E → E + T | T
T → T * F | F
F → F * | a | b | c

Highest precedence

Q.

A → A $ B | B
B → B # C | C
C → C @ D | D
D → d.

Left Recursive

$:> $
↓.
:>
@ :> @

Left One

$ < # < @

Q

$E \rightarrow E * F \mid F + E \mid F$

$F \rightarrow F - F \mid id$.

$* \rightarrow$ left Recursive

$+ \rightarrow$ right Recursive

$+$ & $*$ having same precedence

$-$ is having highest precedence.

$* \doteqdot * $

$+ \lessdot \cdot +$

$S \rightarrow AB \mid CA$

$B \rightarrow BC \mid AB$

$A \rightarrow a$ ✓

$C \rightarrow aB \mid b$ ✓

Soln Step 1. There are no null & unit productions.

Step 2 :- Non-terminals in the grammar are.

S, A, B, C

Step 3 ~~S → A~~,

S, A & C

A & C generates a terminal string

B → useless symbol.

$S \rightarrow CA$

$A \rightarrow a$

$C \rightarrow b$

$S \rightarrow OBO$ ✓

$B \rightarrow S1 \mid 1AA \mid COB$

$A \rightarrow 011 \mid C$ ✗

$E \rightarrow OA$ ✗

$C \rightarrow OCB$ ✗

Soln (1) NO Null productions.

(2) NO Unit productions

(3) Useless symbols

(1) $\{S, A, B, C, E\}$

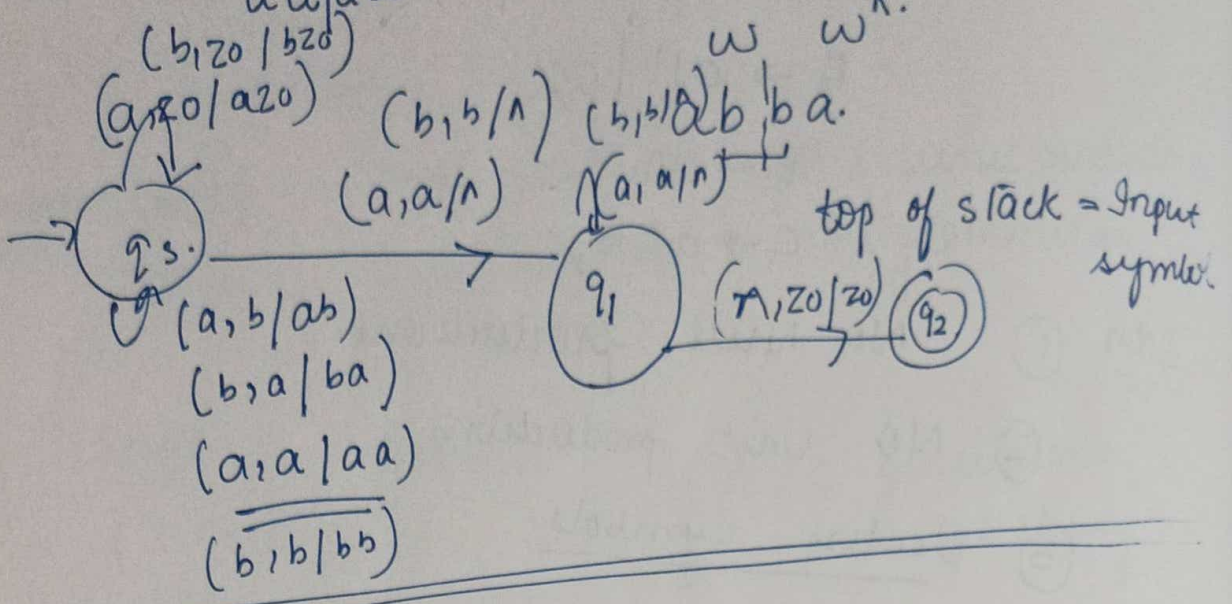
(2) Useful symbols → S, A, B, E

Useless → C

S, A, B.

$ww^R \mid w \in (a,b)^+$

aa|aa



a a a a

(qs, a, Z0)

ID.

(qs, a a a a, Z0)

↓

(qs, aaa, aZ0)

center has not come | center has come

(qs, aa, aaZ0)

(q1, aa, Z0) ✗

No center | center

(qs, a, aaaZ0)

NS | C

(qs, Λ, aaaaZ0) ✗

(qs, Λ, aaZ0) ✗

(q1, a, aZ0)

↓ (q1, Λ, Z0)

‖

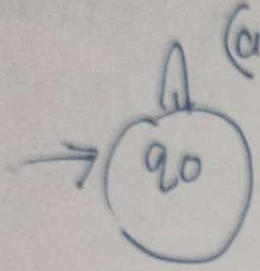
(qf, Λ, Z0)

NDPDA $\subseteq$ DPDA

more lang that it can accept

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

aabb push one a at two a.



Step1 Assume L is context-free. Let n be the natural number obtained by using the pumping lemma

Step2 Choose $z \in L$ $|z| \geq n$.

Write $z = uvwxy$

Step3 Find a suitable k so that $uv^k wx^k y \notin L$

Q1 Show $L = \{ a^n b^n c^n \mid n \geq 1 \}$ is not context-free but context-sensitive.

Sol $z = a^n b^n c^n$

for $n=1$ $z = abc$. $|z| \leq n$

$n = 4$.

$n=2$ $z = aabbcc$ $6 \not\geq 4$

for $i=0$ $z = uv^0 wx^0 y$

=

Pumping Lemma

Let L be a context free language then there is a constant 'n' depending only on L such that if $w \in L$ and $|w| \geq n$ then w may be divided into five pieces

$w = uvxyz$

(i) for all $i \geq 0$, $uv^ixy^iz \in L$

(ii) $|vy| \geq 1$

(iii) $|vxy| \leq n$

Q $L = \{a^n b^n c^n \mid n \geq 0\}$ is not context-free.

Soln $n = 1$ $w = abc$ $n = 4$

$i = 2$ $w = aabbcc$ $|w| \geq n =$

$6 \geq 4$. 53 (6) NO

Wrong

$w = aabbcc$

$w = uvxyz$

$u = a$

$v = a$

$x = b$

$y = b$

$z = cc$

$|vy| \geq 1$

$2 \geq 1$

$|vxy| \leq n$

$3 \leq 4$.

$w = uv^ixy^iz$

$i = 0$ uxz

$= \boxed{abcc}$

a a a a b b b B

x a a a b b b y

x x a a b b y y

x x x a b y y y

x x x x b y y y

x a a b b b B B

$q_0 \rightarrow q_1 \rightarrow q_1 \rightarrow q_1 \rightarrow q_1$

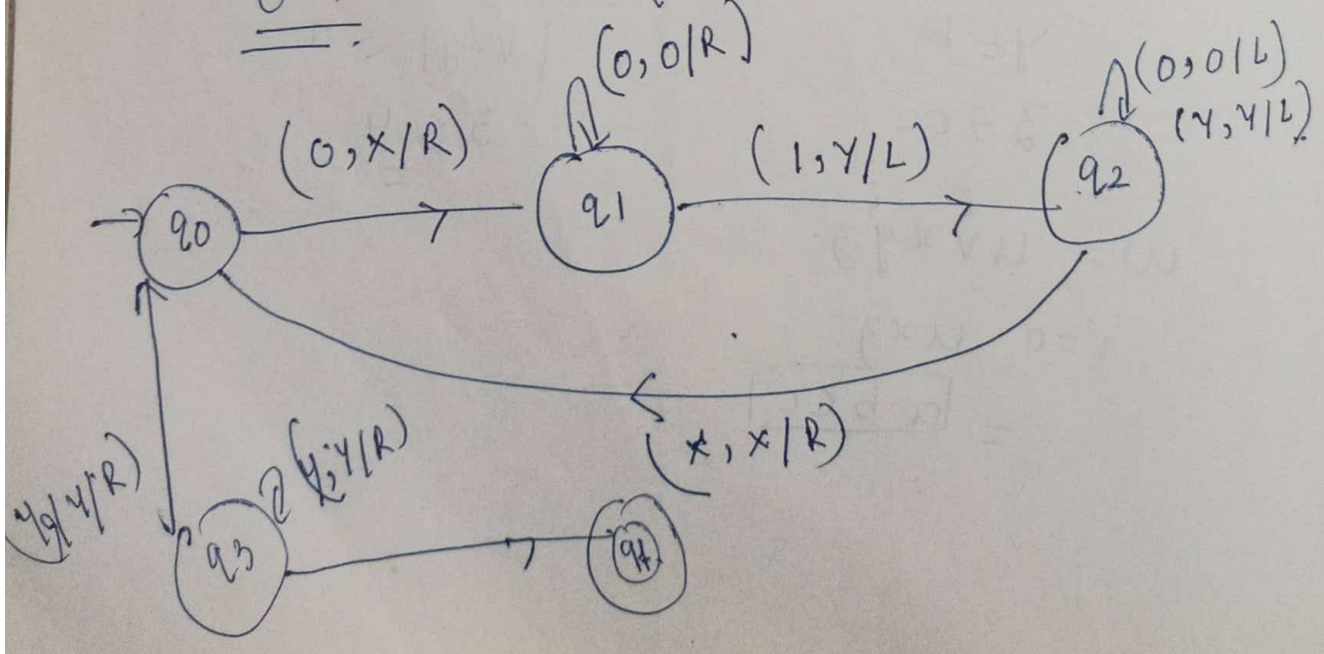
B. B | a | 0 | 0 | 1 | 1 | 1 | B.

X Y.

X | X | 0 | Y | Y | 1 | | B

B X X X Y Y Y B | B

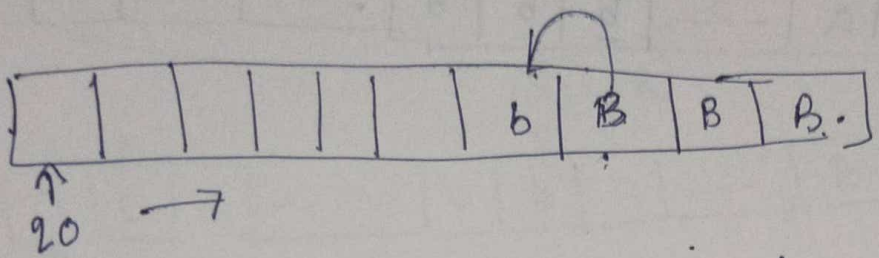
$0^n 1^n$

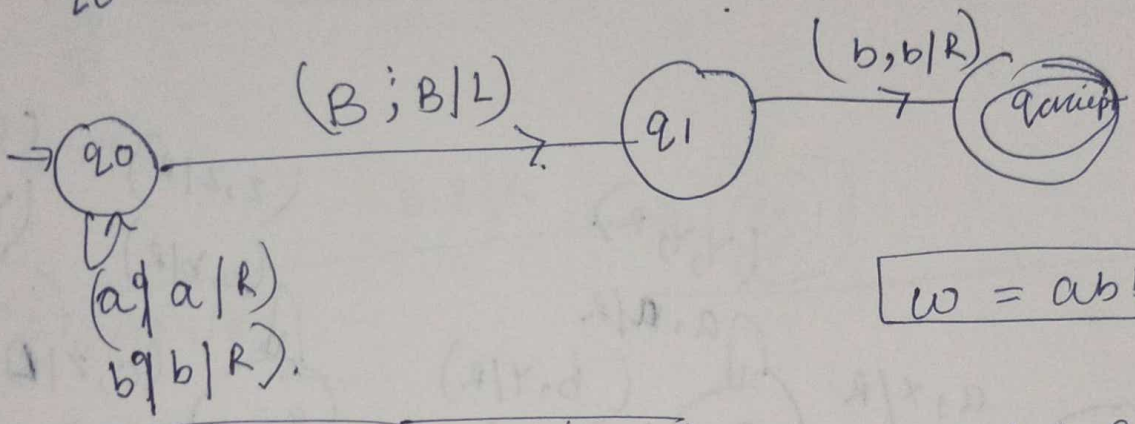


Q. $\Sigma = \{a, b\}$. that ends with "b"

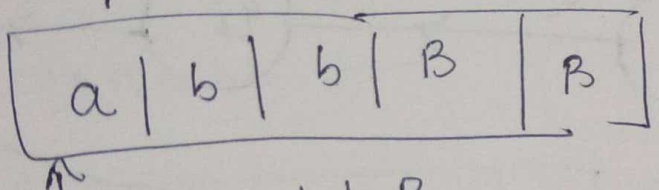
$(a+b)^* b$.

$w = \{ b, ab, bb, aab, \dots \}$





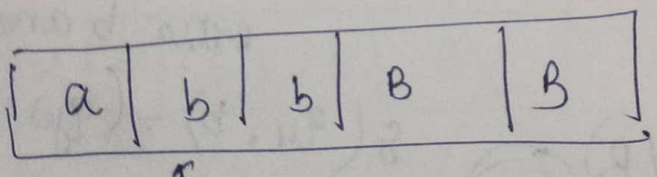
$w = abb$



$q_0\ a\ b\ b\ B.$

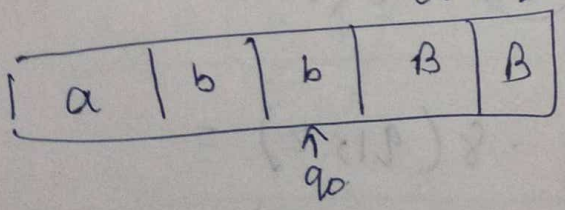
$q_0\ a\ b\ b\ B$

$a\ q_0\ b\ b\ B.$

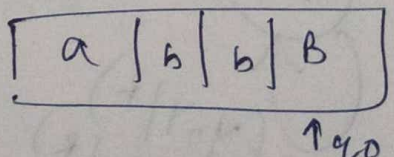


$a\ q_0\ b\ b\ B.$

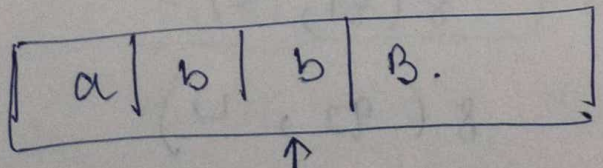
$a\ b\ q_0\ b\ B.$



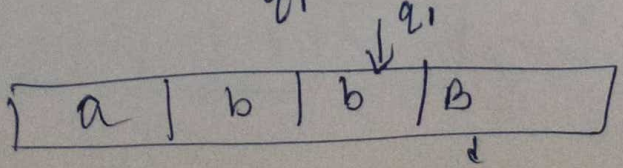
$a\ b\ q_0\ b\ B$



$a\ b\ b\ q_0\ B.$



$a\ b\ q_1\ b\ B.$

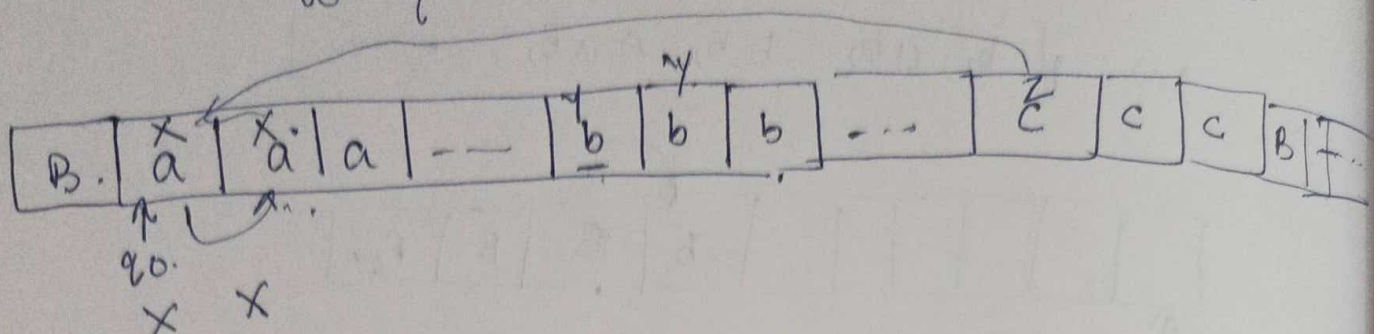


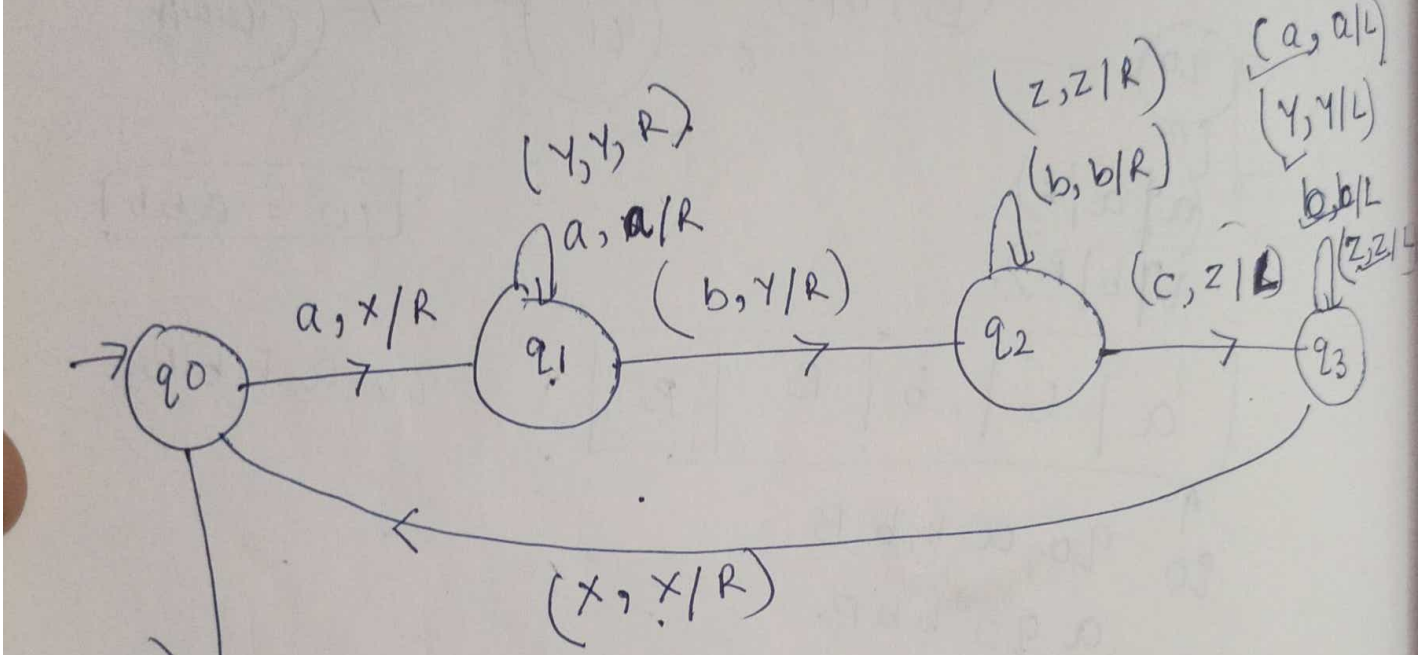
$a\ b\ b\ q_{accept}\ B$

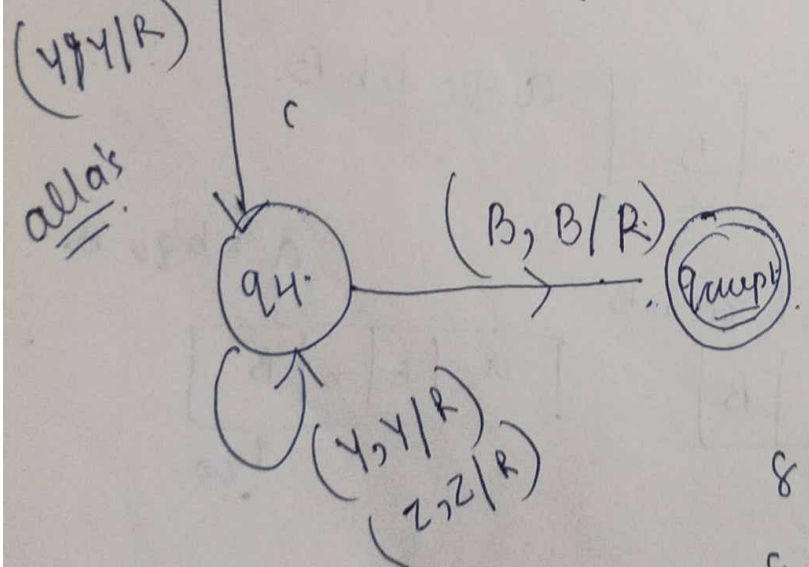
q_{accept}

Q2 $L = \{ a^n b^n c^n \mid n \geq 1 \}$

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







extra b and c.

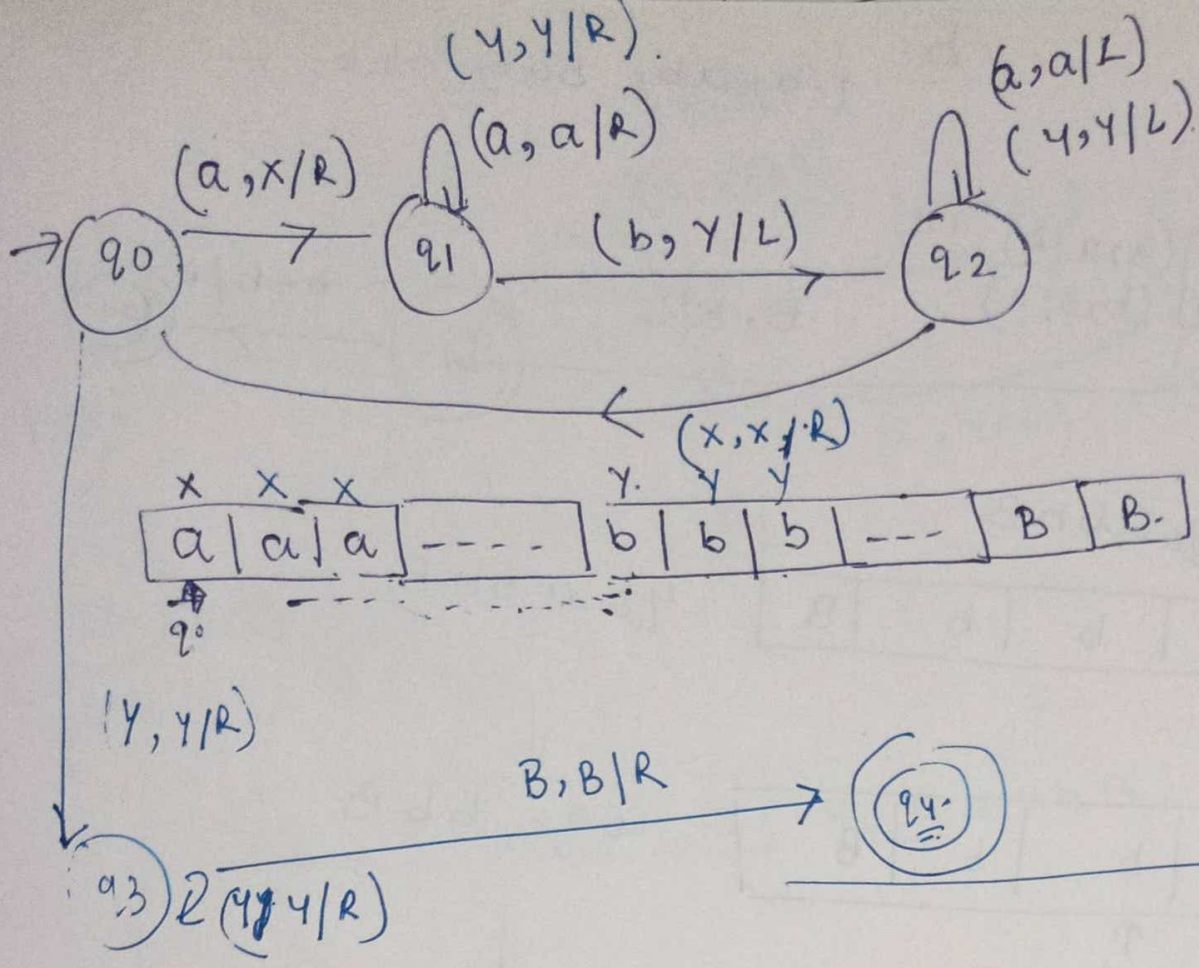
$\delta(q_4, b) = (q_{jeet}, b, R)$

$\delta(q_4, c) = (q_0, c, R)$

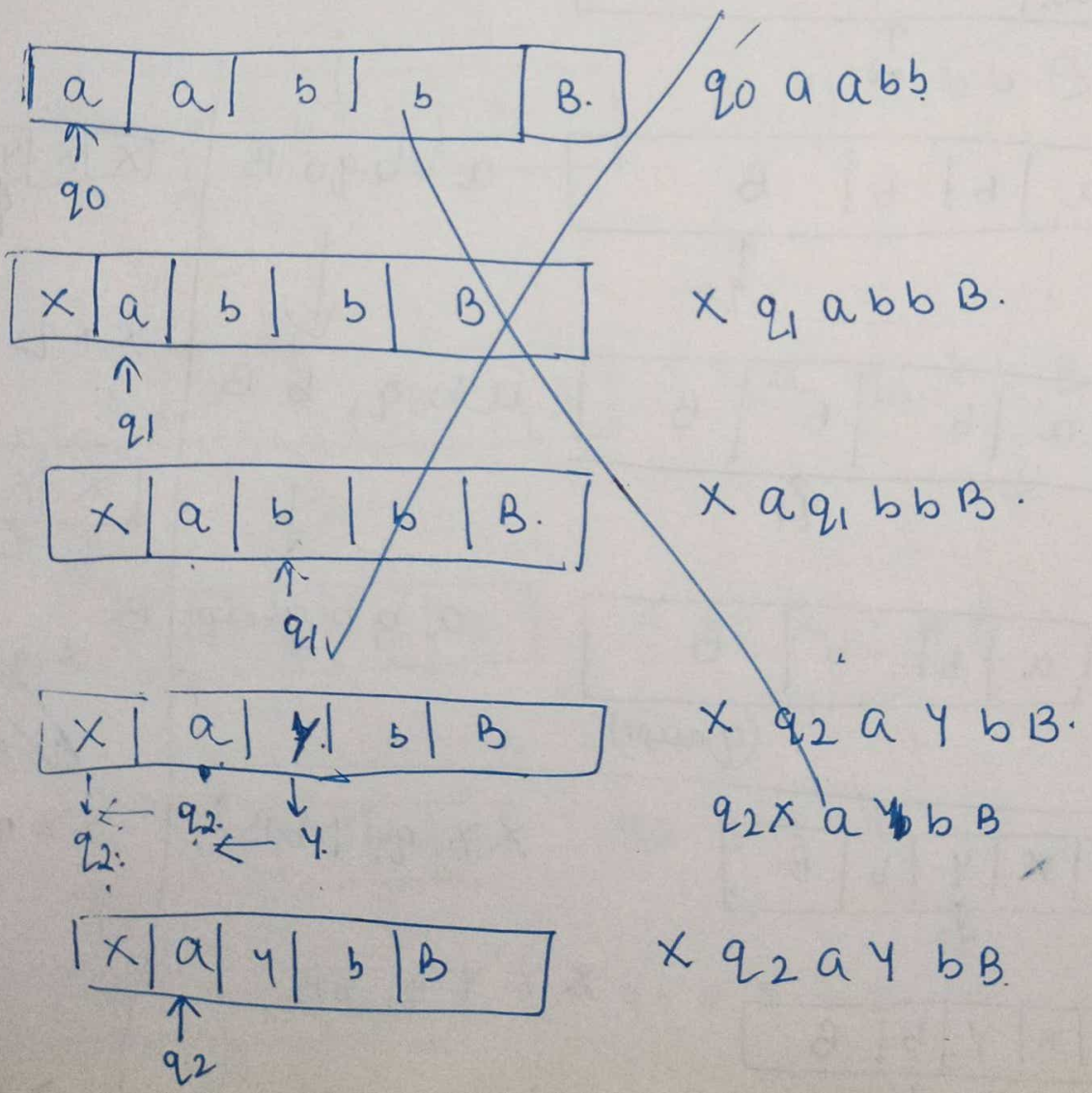
$\delta(q_1, Z) =$

$\delta(q_1, c) =$

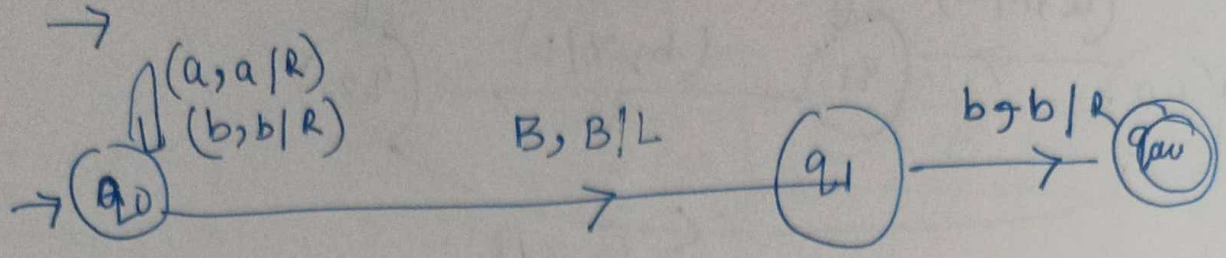
$\delta(q_2, \sqcup) =$


(Y,Y/R)
(a,a/R)
(a,X/R)
(a,a/L)
(Y,Y/L)
(b,Y/L)
q0
q1
q2
(X,X/R)
X X X
a a a
b b b
B B
q0
Y,Y/R)
B,B/R
q4
q3
(Y,Y/R)

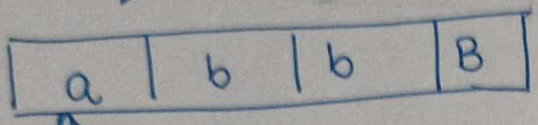
q0 a a b b ⊢


a a b b B
q0
q0 a a b b
X a b b B
q1
X q1 a b b B.
X a b b B
q1
X a q1 b b B.
X a Y b B
q2
X q2 a Y b B.
q2 X a Y b B
X a Y b B
q2
X q2 a Y b B

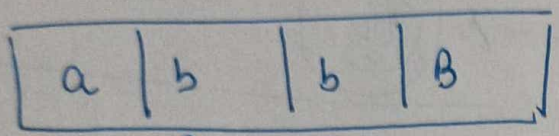
b. {b, ab, bab, abb. — }



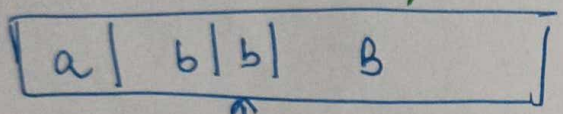
q_0abbB.



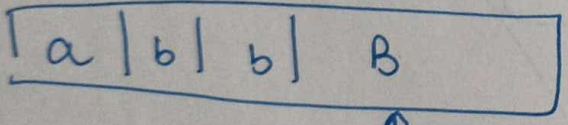
q_0abbB.



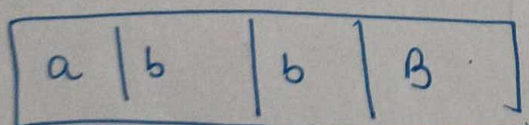
aq_0bbB



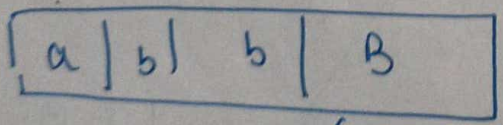
abq_0bB



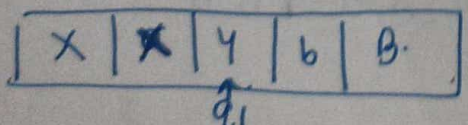
$abbq_0B$.



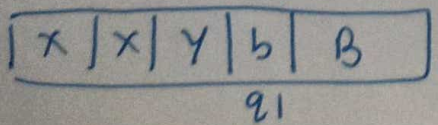
abq_1bB.



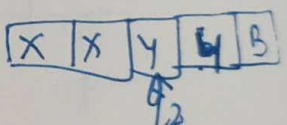
$abbq_{aupt}B$.



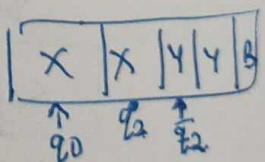
XXq_1YbB.



$XXYq_1bB$



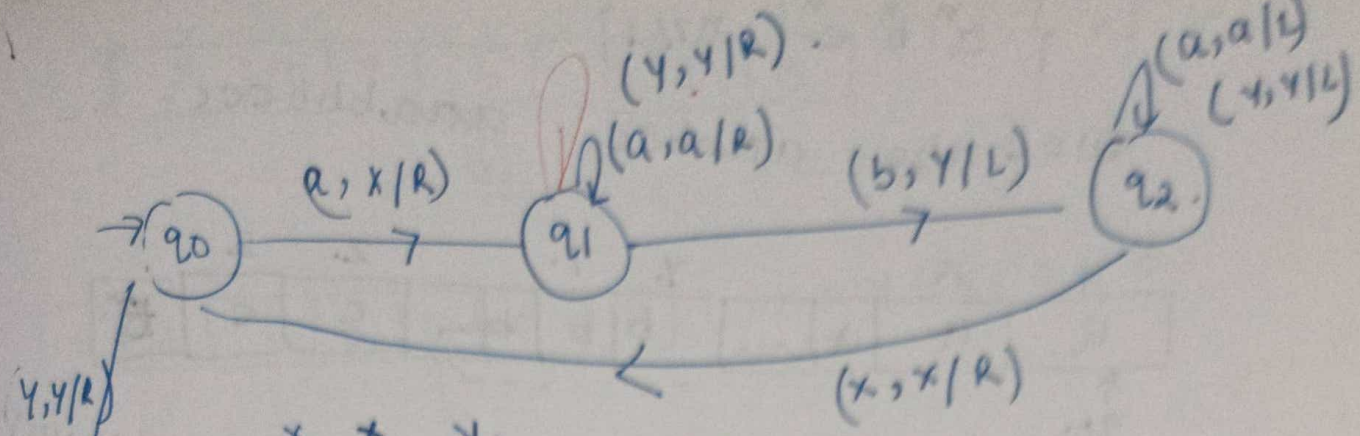
XXq_2YYB.



Xq_2XYY

q_0XXYY.

XXq_0YYB.



X X Y.

| a | a | b | b | B |

q0 a a b b

| a | a | b | b | B |

q0 a a b b B.

q0

⊢

q0 a a b b B ⊢ X.

| X | a | b | b | B |

X q1 a b b B.

q1

| X | a | b | b | B |

X a q1 b b B.

q1.

| X | a | Y | b | B |

X q2 a Y b B.

q2

| X | a | Y | b | B |

q2 X a Y b B.

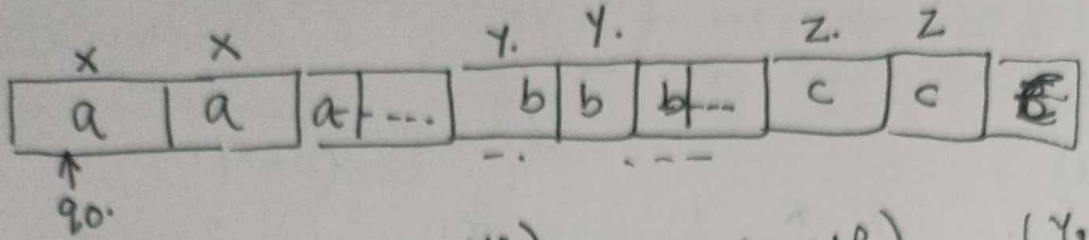
| X | a | Y | b B |

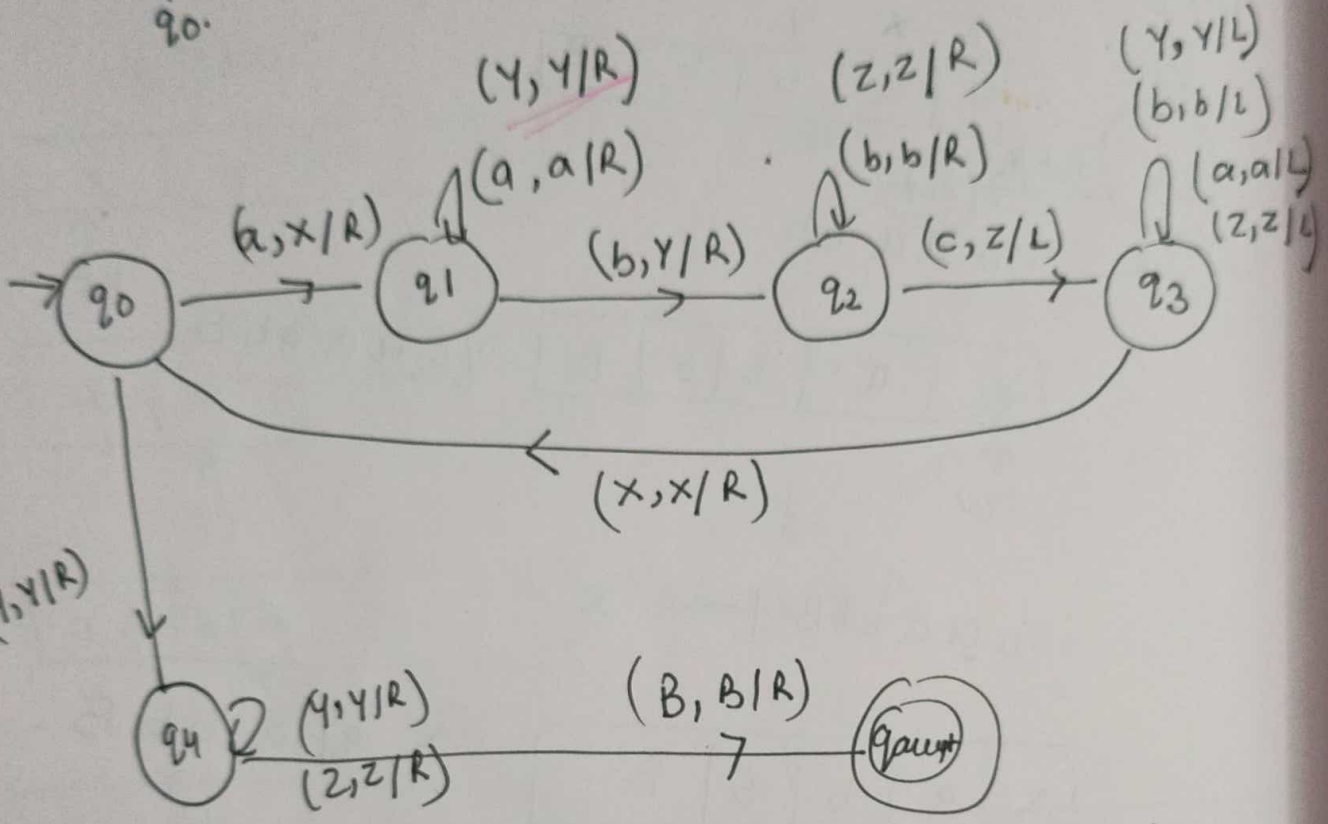
X q0 a Y b B.

q0

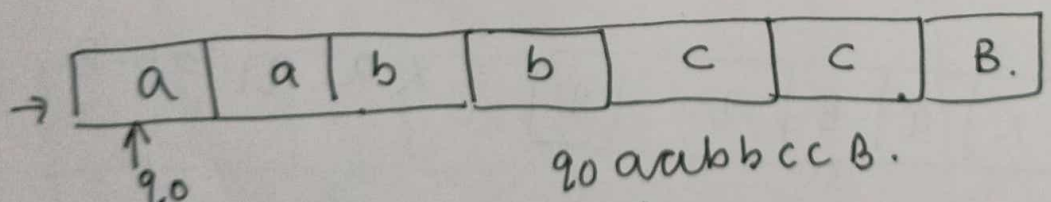
$L = \{ a^n b^n c^n \mid n \geq 1 \}$

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

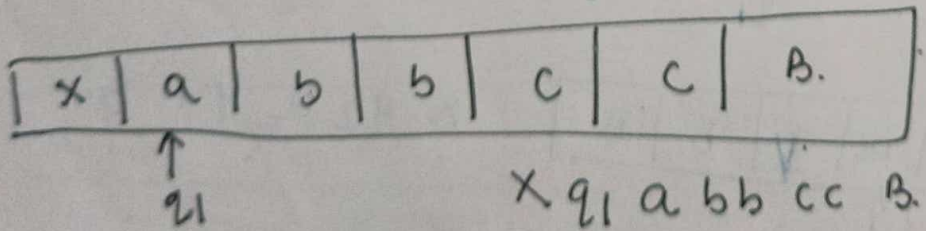




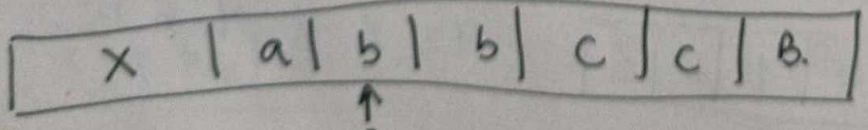
aabbcc



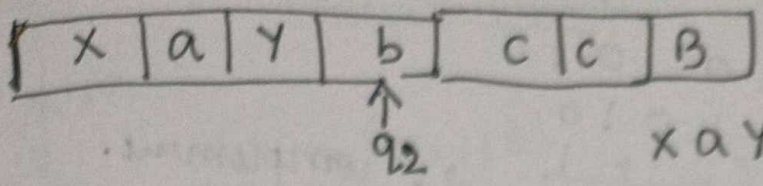
$q_0 aabbcc B.$



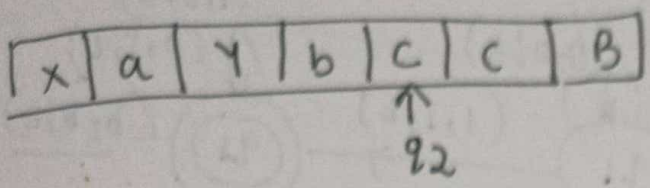
$X q_1 a b b c c B.$



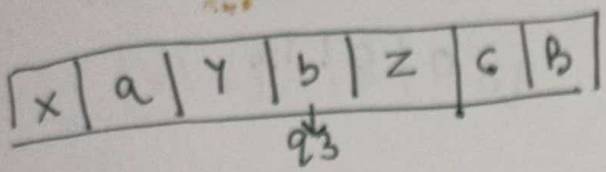
$X a q_1 b b c c B$



X a Y q2 b C C B

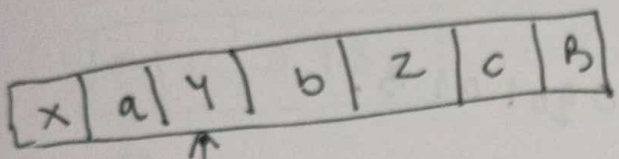


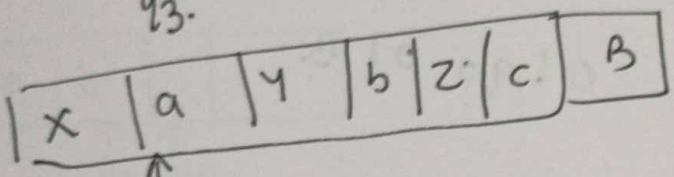
X a Y b q2 C C B.

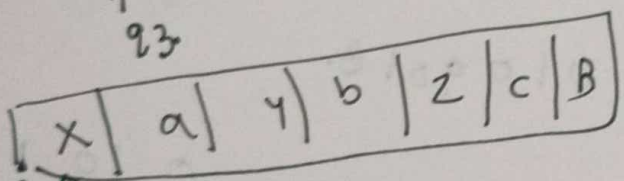


X a Y q3 b Z C B.

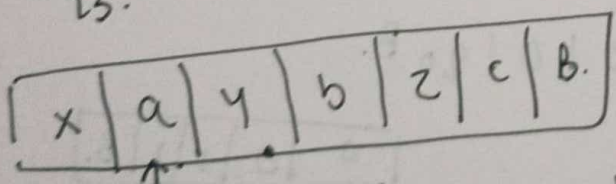
$a^n b^n$



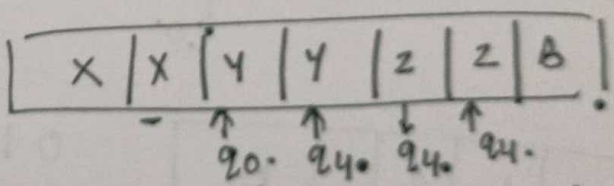


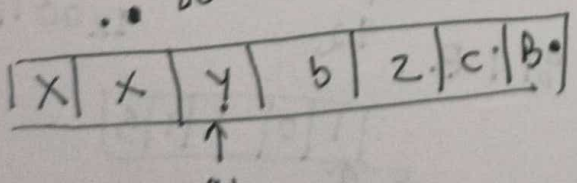


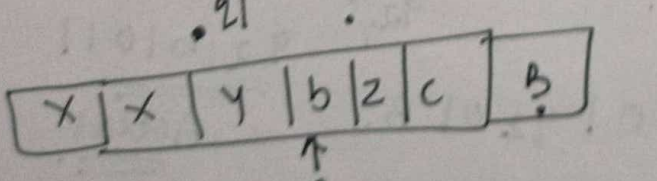
q3 X a Y b Z c B.

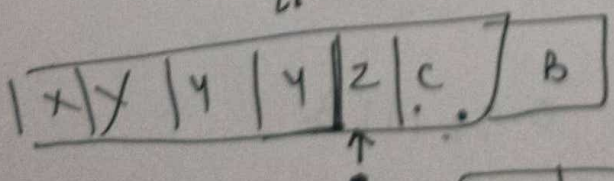


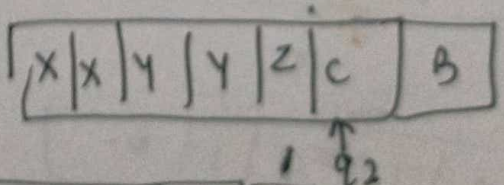
X q0 a Y b Z c B.

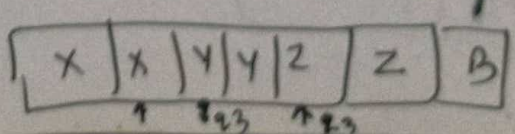












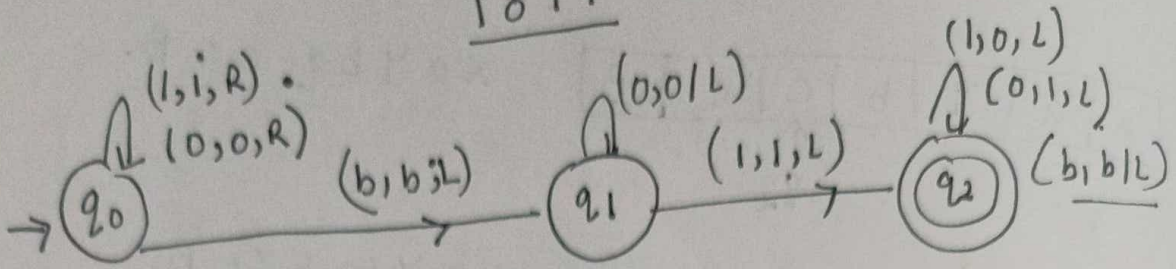
Input string 0101

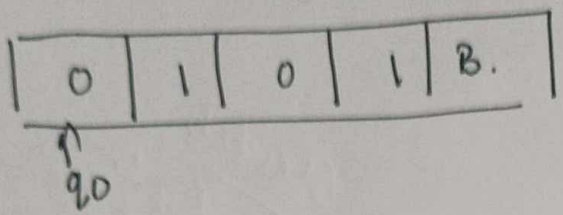
Input 0 1 0 1

1's comp. 1 0 1 0

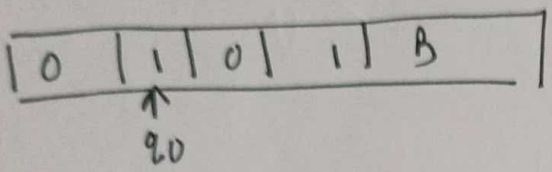
+ 1

1 0 1 1 2's complement.

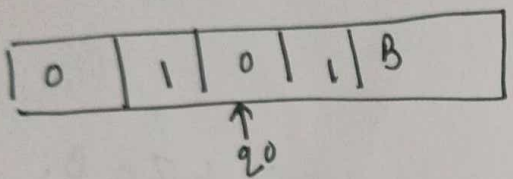




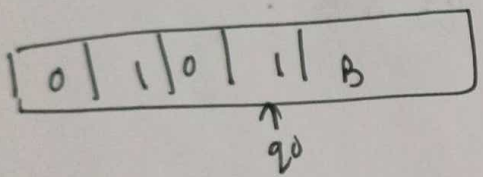
q0 0 1 0 1 B.



0 q0 1 0 1 B.

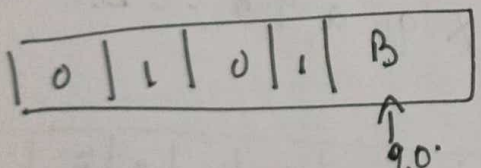


0 1 q0 0 1 B.

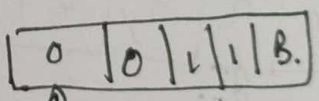


0 1 0 q0 1 B.

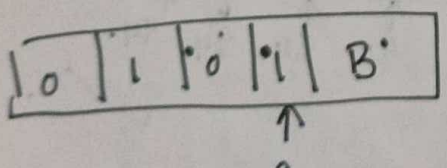
0 10 !



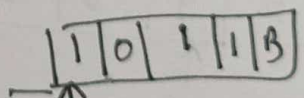
0 1 0 1 q0 B.



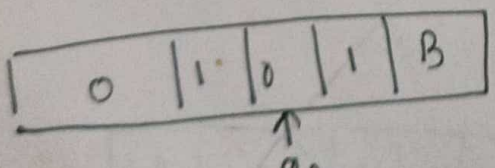
q2 0 0 1 1



0 1 0 q1 1 B

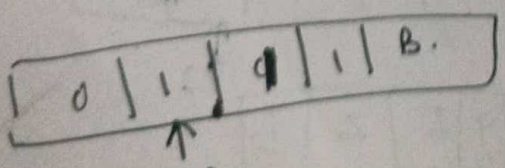


q2 b 1 0 1 1



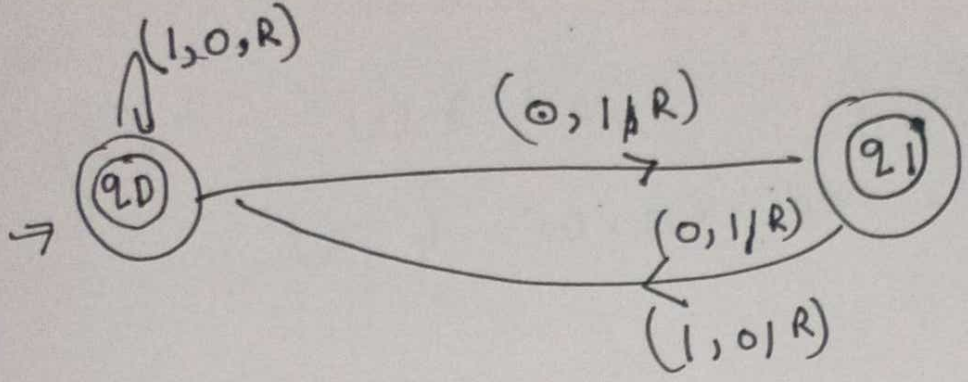
0 1 q2 0 1 B.

Accept.



0 q2 1 1 1 B.

1's complement



1	0	1	0
0	1	0	1

q0 1 0 1 0

1	0	1	0	B.

q0

D	0	1	0	B

q0

0 q0 0 1 0 B.

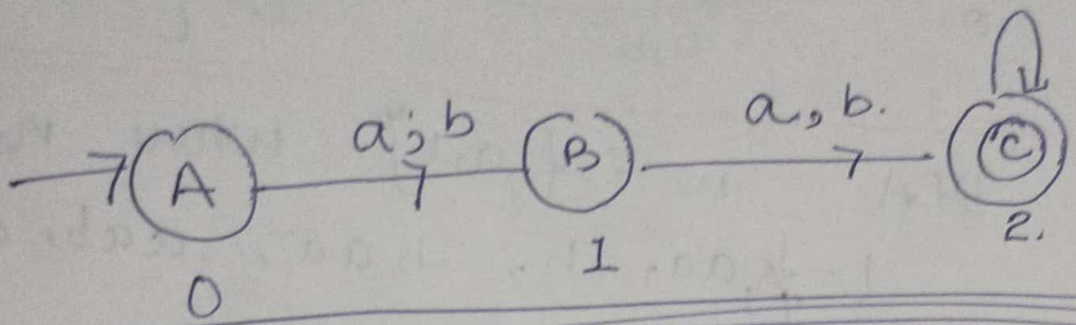
0	D	1	

q0 q1

2) Construction of DFA

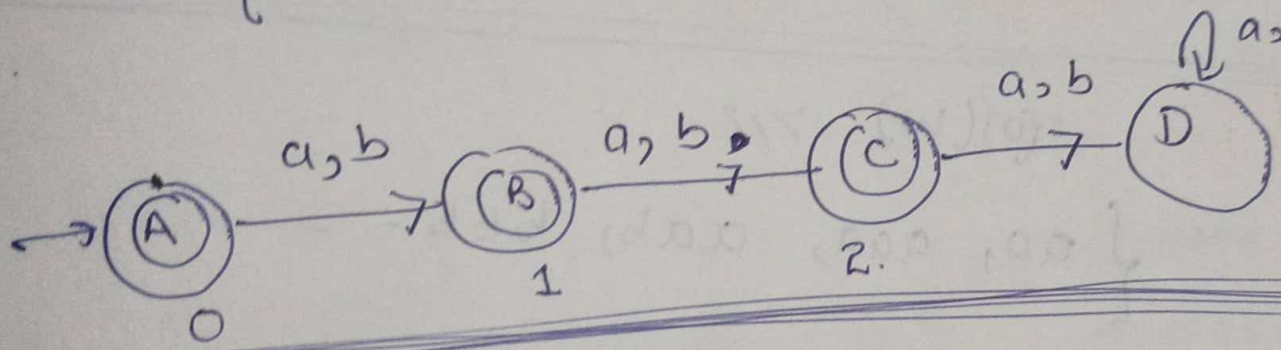
(1) $w \in \{a, b\}$ $|w| > 2$

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



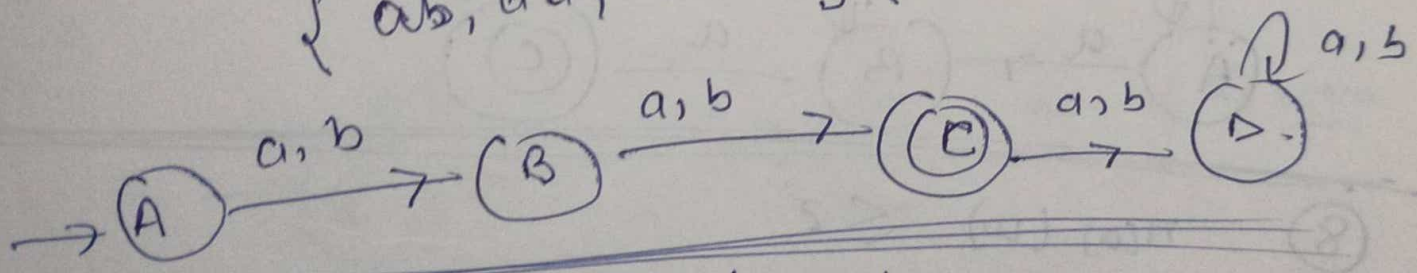
2. $w \in \{a, b\}$ $|w| \leq 2$

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



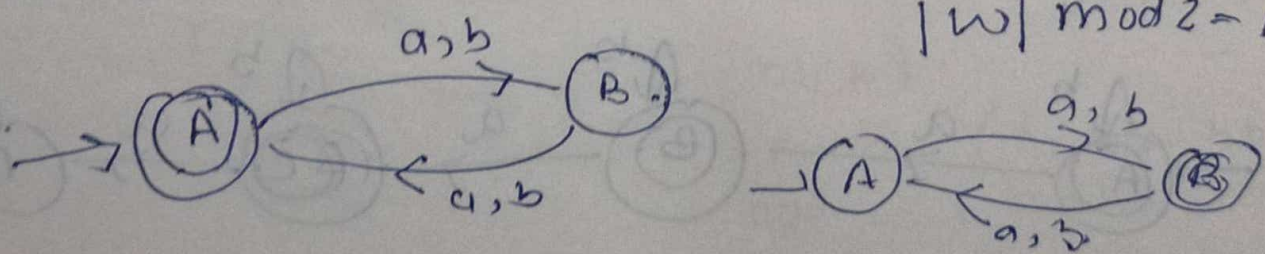
3. $|w| = 2$

$\{ ab, aa, bb, aa \}$.



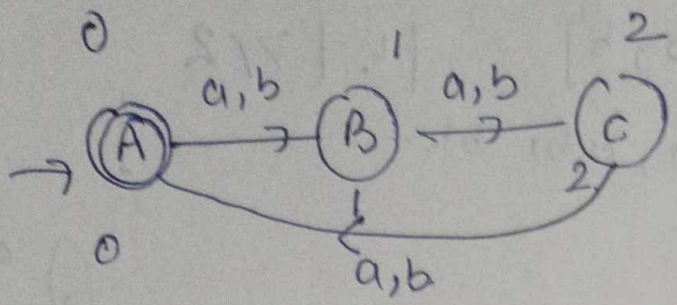
4. $w \in \{a, b\}$ $|w| \bmod 2 = 0$.

$|w| \bmod 2 = 1$.



(5) $w \in \{a,b\}^+$ $|w| \bmod 3 = 0$

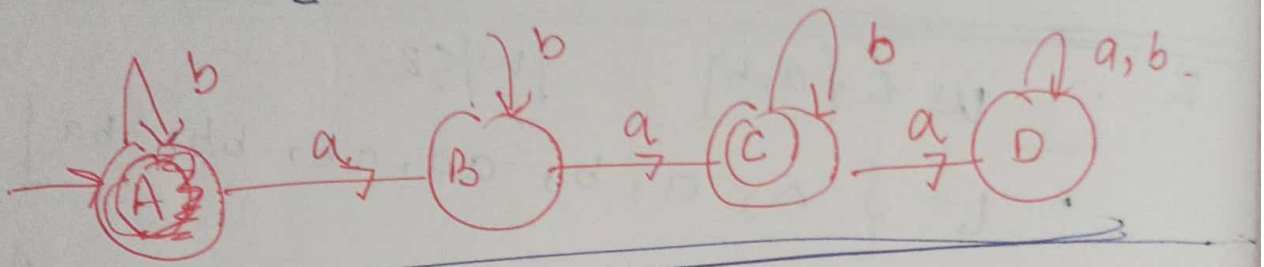
0, 1, 2



$L = \{\wedge, aaa, aab, \ldots\}$

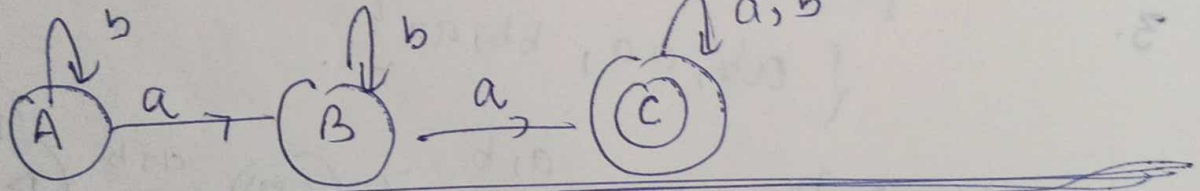
(6) DFA $w \in \{a,b\}^*$ where $n_a(w) = 2$

$L = \{\wedge, aa, b, baa, aab, aba \ldots\}$



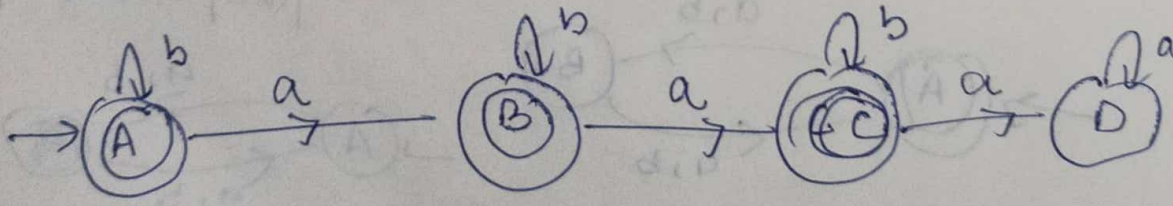
(7) $n(a)(w) \geq 2$

$\{aa, aaa, aab, bbb \ldots\}$



(8) $n(a)(w) \leq 2$

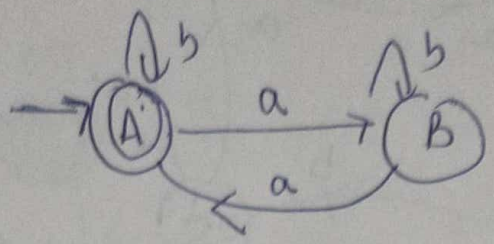
$\{\wedge, b, \ldots\}$



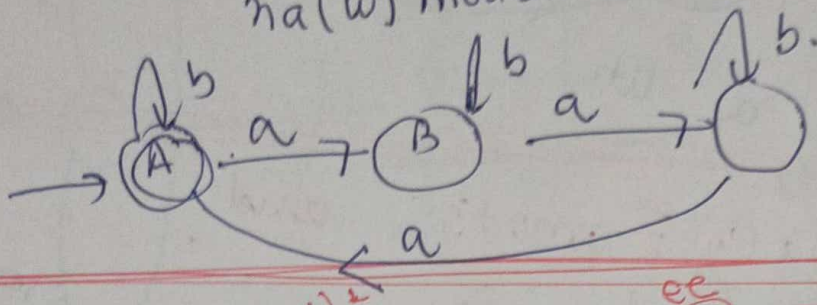
(9) $n_a(w)$ = even.

m DFA $w \in \{a,b\}^*$

$n_a(w) \bmod 2 = 0$



$n_a(w) \bmod 3 = 0$



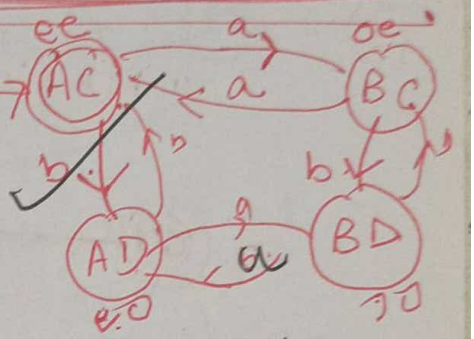
A D

---

(10) $w \in (a,b)^*$

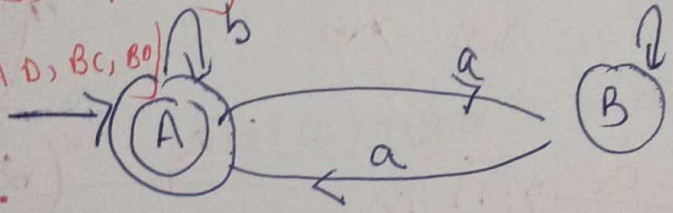
$n_a(w) \cong 0 \bmod 2$

$n_b(w) \cong 0 \bmod 2$



{A, B} x {C, D}

{AC, AD, BC, BD}

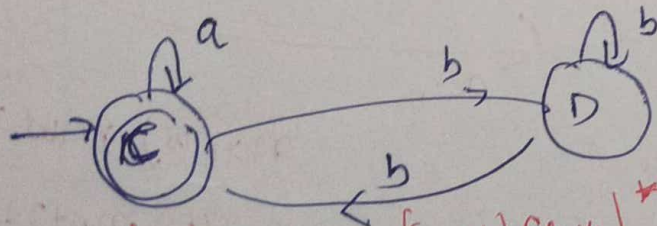


$a \cong 0 \bmod 2$

no of a's even

BC

AC



no. of b's even

Bc

BD

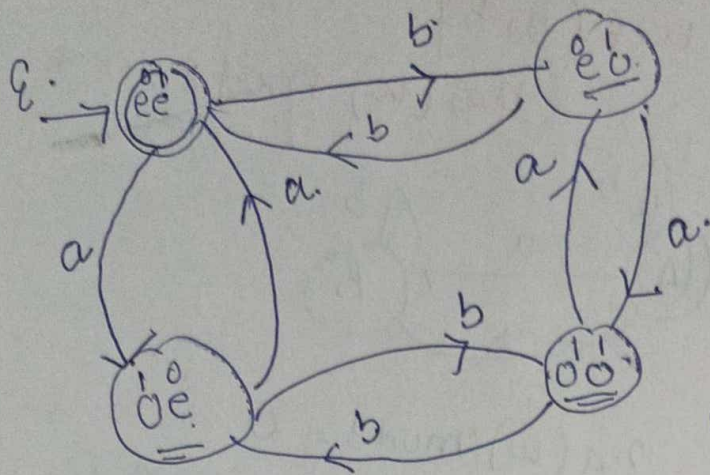
$(a+b)(a+b)^*(a+b)$

$\geq n+2$ $L = \{\lambda, aa, bb, aabb, abab, babb\}$

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

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

$n_a(w)$	$n_b(w)$	
even	even	$(\lambda, aa, bb \ldots$
even	odd	(aab)
odd	even	$(aaabb$
odd	odd	$(ab$



ba

bb.

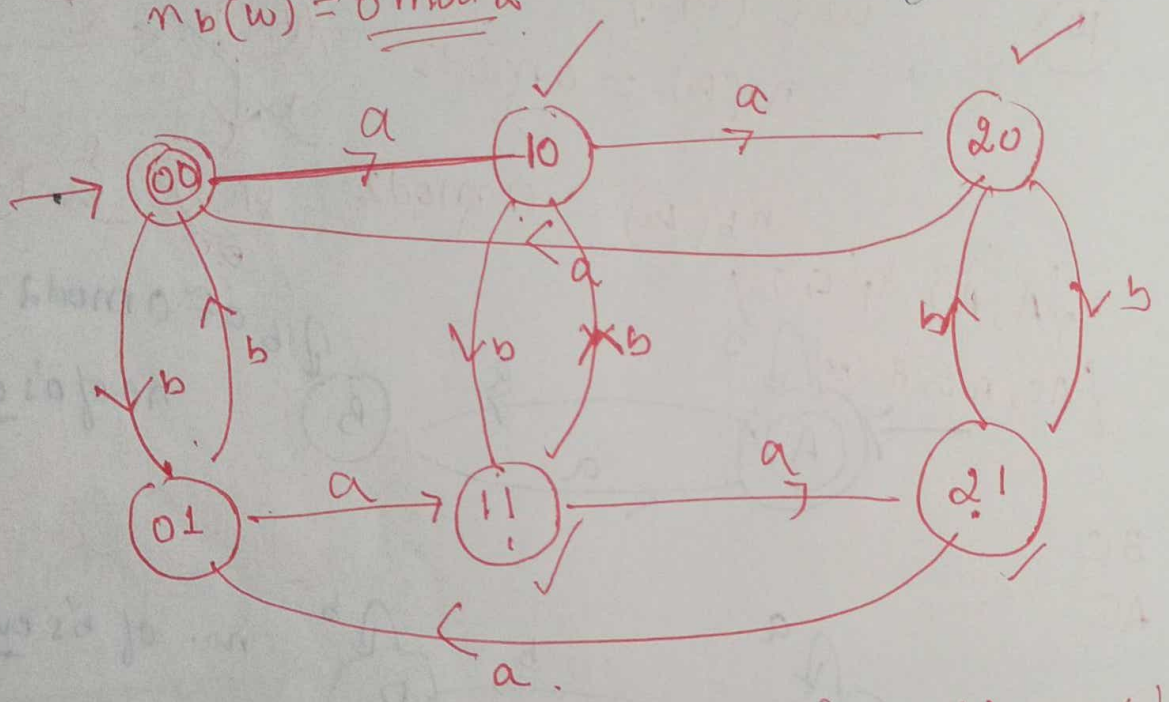
ab.a

abb

a ab.

Q $n_a(w) \equiv 0 \bmod 3$ and

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

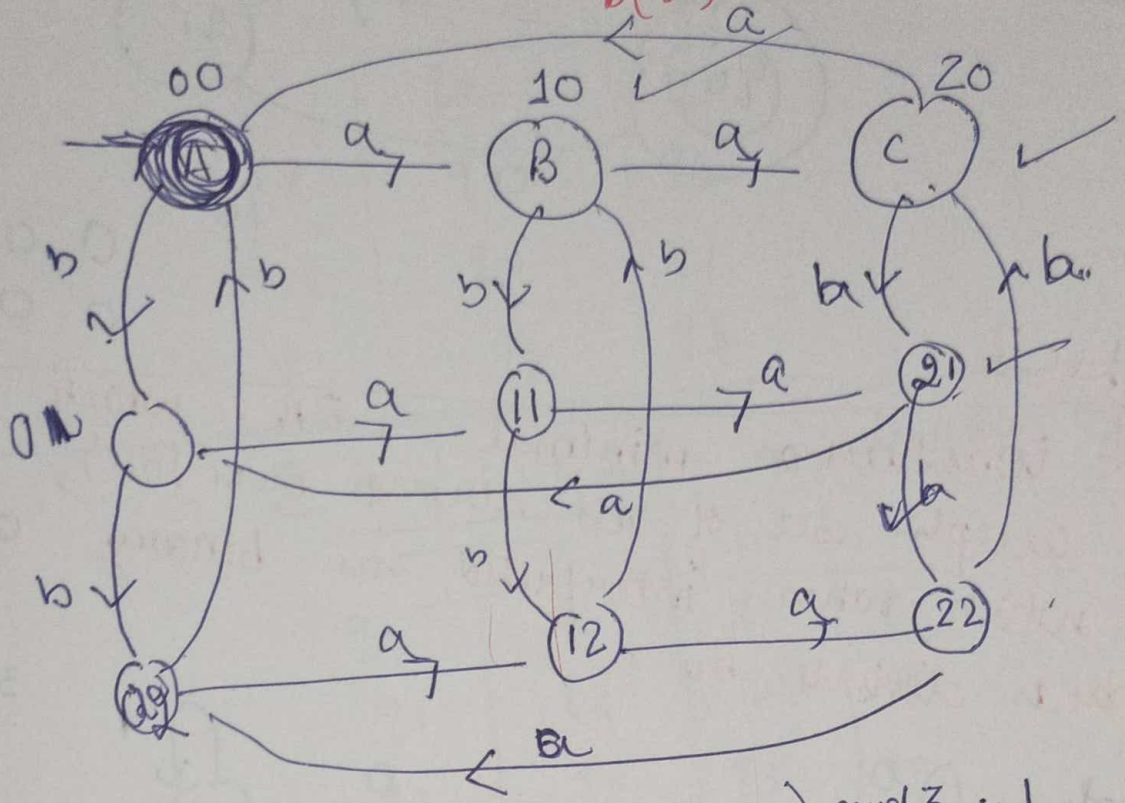


$n_a(w) \bmod 3 \rightarrow n_b(w) \bmod 2$

$n_a(w) \bmod 3 \rightarrow n_b(w) \bmod 2$

Lec 5 $w \in \{a,b\}^+$ $n_a(w) \equiv 0 \bmod 3$

$n_b(w) \equiv 0 \bmod 3$



$n_a(w) \bmod 3 = 1$

and

$n_b(w) \bmod 3 = 2$.

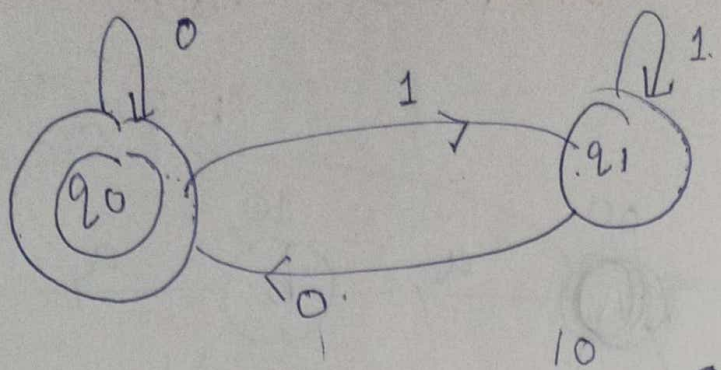
$n_a(w) \bmod 3 > n_b(w) \bmod 3$.

Lec 6 Construct a minimal DFA which accepts set of all strings over $\{0,1\}$ which when interpreted as a binary number is divisible by 2

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

$(1\,1\,0)_2 = (6)_{10}$

Ans.



11
3!
3

10
2!

0 0 0 0
0 0 1 0

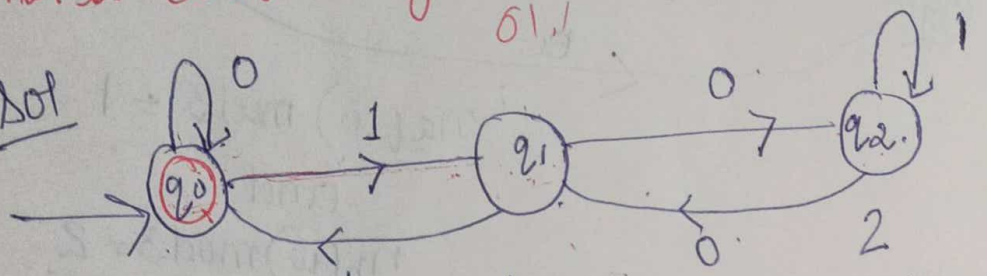
Lec 7

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

1
11
011
101

3

Sol



100
42!

0
10
1
100
10
100

0
0
11
110

10
21

11
3

100
42 1

	0	1
q0	q0	q1
q1	q2	q0
q2	q1	q2

	0	1
q0	q0	q1
q1	q2	q0
q2	q1	q2

Q Construct a minimal DFA which accepts set of all binary no divisible by 4.

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

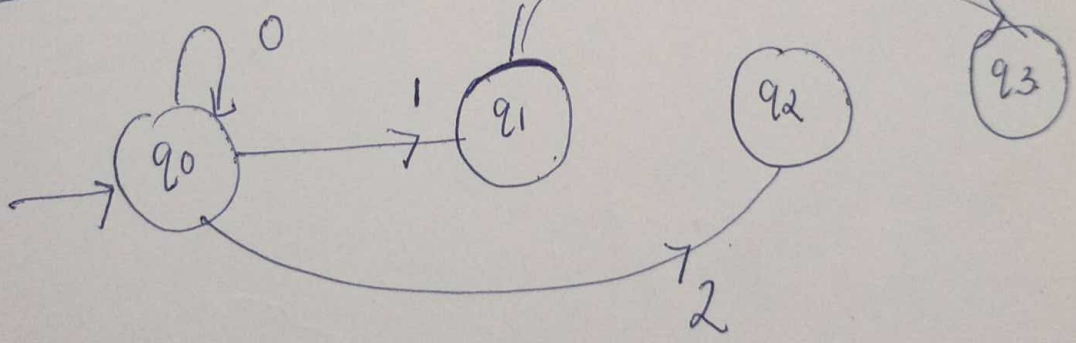
$\Sigma = \{0, 1, 2\}$ no. divisible by 4.

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

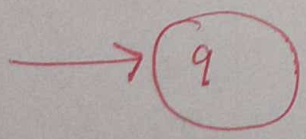
$(10)_3$

$1*3+0$

Ans.



0.



$A \to a\alpha$ where $\alpha \in V_N^*$

$a \in \Sigma$ (α may be $\wedge$)

$A \to A\beta_1 \mid A\beta_2 \mid \text{---} \mid A\beta_m \mid \alpha_1 \mid \alpha_2 \mid \text{---}$

Eg. $A \to \alpha_1 Z \mid \alpha_2 Z \mid \text{---} \mid \alpha_n Z.$

$Z \to \beta_1 Z \mid \beta_2 Z \mid \text{---} \mid \beta_m Z.$

Equivalence of CFG and PDA

A language is context Free if some PushDown Automata recognize it

Part 1. given a CFG, show how to construct a PDA that recognizes it

Part 2 Given a PDA, show how to construct CFG that recognizes the same language

---

Part 1 CFG to PDA

(Left most derivation

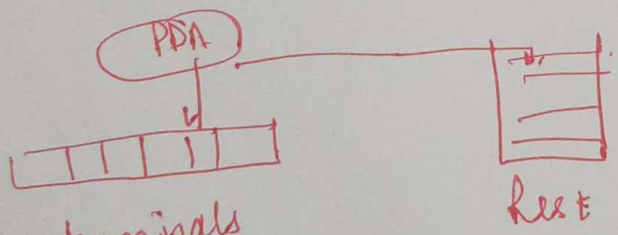
$S \to BS \mid A$

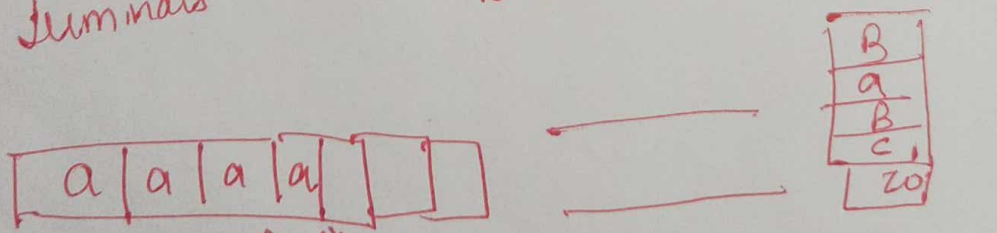
$A \to 0A \mid \wedge$

$B \to BB1 \mid 2$

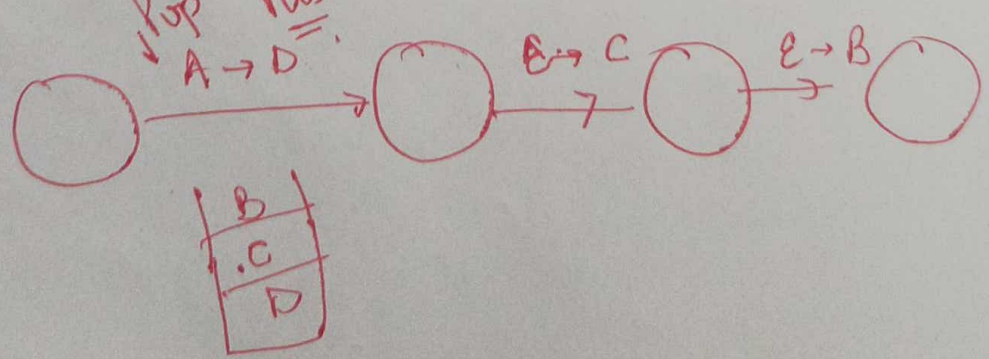
$S \to BS$
$\to BB1S$
$\to 2B1S$
$\to 2 2 1 S$
$\to 2 2 1 A$
$\to 2 2 1 \wedge$
$\to 221$

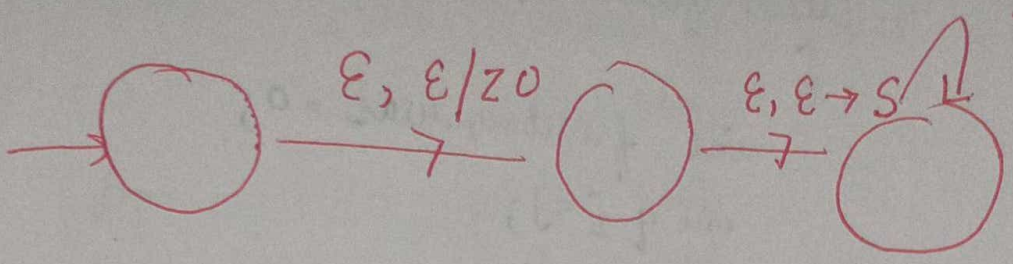
aaaa BaBc
Terminals Rest





$A \to BCD$





$S \to aSa$

$S \to bSb$

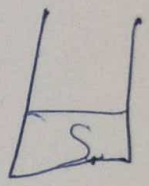
$S \to c$

$S \to aSa \mid bSb \mid c$

(1) Write the production rules and then pop elements

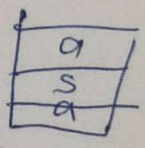
(1) $\delta(q_0, \wedge, \wedge) = (q_0, \wedge)$ → state, → value of stack

no string



(2) $\delta(q_0, \wedge, S) = (q_0, aSa)$

(3) $\delta(q_0, \wedge, S) = (q_0, bSb)$



(4) $\delta(q_0, \wedge, S) = (q_0, c)$

Pop

(5) $\delta(q_0, a, a) = (q_1, \wedge)$

(6) $\delta(q_1, b, b) = (q_2, \wedge)$

(7) $\delta(q_2, c, c) = (q_3, \wedge)$

```
Class queue ()      #define SIZE 5
{
    int front=0, rear = 0;
    int q [5];
public:
    void insert ()
{
    int no;
if ( rear == SIZE && front =0)
    cout << " queue is full";
else.
{
cout << " enter no:";
cin >> no;
    q [rear] =no;
}
    rear++;
}
    void delete ()
{
    int no, i;
if ( front == rear)
cout << " queue is empty";
else
{
    no = q [front];
    front ++;
cout << no << " removed from queue";
}
}
```

SNo.	state	unread ip	stack	transition
(1)	q_0	abbcbba	^	1.
(2)	q_0	abbcbba	S	1.
(3)	q_0	~~abbcbba~~	aSa.	2.
(4)		bbcbba	Sa	5.
(5)	q_0	bbcbba	bSba	3
(6)	q_2	bcbba	Sba	6.
(7)	q_0	bcbba	bSbba	3
(8)	q_2	cbba	Sbba	6
(9)	q_0	cbba	cbba	4.
	q_3	bba	bba	7
(10)	q_2	ba	ba	6
(11)	q_1	^	^.	5.
12				

$S \rightarrow OBB$

$B \rightarrow OS \mid 1S \mid O$

$G = (\{S, B\}, \{0, 1\}, P, S)$

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

$= (\{q\}, \{0,1\}, \{0, 1, S, B\}, \delta, q, S,$

For Non terminals

$\delta(q, \wedge, S) = (q, OBB)$

$\delta(q, \wedge, B) = (q, OS)$

$(q, 1S)$

(q, O)

$Z_0 \rightarrow S$

For terminals

$\delta(q, 0, 0) = (q, \wedge)$

$\delta(q, 1, 1) = (q, \wedge)$

(10)	q_3	bba	bba	7
(11)	q_2	ba	ba	6
(12)	q_2	a	a	6
(13)	q_1	$\wedge$	$\wedge$	5

Q From CFG to PDA

$S \rightarrow aSa \mid bSb \mid c$ rules and

Soln Write the production then pop elements

1) $\delta(q_0, \wedge, \wedge) = (q_0, \wedge)$
2) $\delta(q_0, \wedge, S) = (q_0, aSa)$
3) $\delta(q_0, \wedge, S) = (q_0, bSb)$
4) $\delta(q_0, \wedge, S) = (q_0, c)$
5) $\delta(q_0, a, a) = (q_1, \wedge)$
6) $\delta(q_1, b, b) = (q_2, \wedge)$
7) $\delta(q_2, c, c) = (q_3, \wedge)$

$S \rightarrow aSa$
$\rightarrow abSba$
$\Rightarrow abbSbba$
$\Rightarrow abbcbba$

SNo	state	unread ip	stack	transition
(1)	q_0	abbcbba	$\wedge$	1
(2)	q_0	abbcbba	S	1.
(3)	q_0	abbcbba	aSa	2
(4)	q_1	bbcbba	Sa	5
(5)	q_0	bbcbba	bSba	3
(6)	q_2	bcbba	Sba	6.
(7)	q_0	bcbba	bSbba	3
(8)	q_2	cbba	Sbba	6
(9)	q_0	cbba	cbba	4

PDA to CFG

If $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is a PDA then CFG is defined as

$$G = (V, \Sigma, P, S)$$

(1) Construction of set of Non-terminals

$$V = \{S\} \cup \{ [q, Z, q'] \mid q, q' \in Q,\ Z \in \Gamma \}$$

(state)

(2) (i) S-Production $S \to [q_0, Z_0, q]$, $q \in Q$ (Initial)

(ii) For Pop-operation

$\delta(q, a, Z) \to (q', \wedge)$

$[q, Z, q] \to a$

(iii) Push and No-operation

$\delta(q, a, Z) \to (q_1, Z_1 Z_2 \cdots Z_m)$

$[q, Z, q'] \to a [q_1, Z_1, q_2] [q_2, Z_2, q_3] \cdots [q_m, Z_m, q']$

where $q', q_1, q_2 \cdots \in Q$.

$A = \{q_0, q_1\}, \{a, b\}, \{Z_0, Z\}, \delta, q_0, Z_0$

Q Σ Γ

$\delta: (q_0, b, Z_0) = (q_0, ZZ_0)$

$\delta(q_0, b, Z) = (q_0, ZZ)$

$\delta(q_1, b, Z) = (q_1, \wedge)$

$\delta(q_0, \wedge, Z_0) = (q_0, Z_0)$

$\delta(q_0, a, Z) = (q_1, Z)$

$\delta(q_1, a, Z_0) = (q_0, Z_0)$

① V = S; $[q_0, Z_0, q_0]$ $[q_0, Z_0, q_1]$

$[q_0, Z_1, q_0]$ $[q_0, Z_1, q_1]$

$[q_1, Z_0, q_0]$ $[q_1, Z_0, q_1]$

$[q_1, Z, q_1]$ $[q_1, Z, q_1]$.

② $S \to [q_0, Z_0, q_0]$

$S \to [q_0, Z_0, q_1]$

(i) $\delta(q_0, b, Z_0) \Rightarrow (q_0, ZZ_0)$

(ii) $[q_0, Z_0, q_0] \Rightarrow b\,[q_0, Z, q_0]\,[q_0, Z_0, q_0]$

$[q_0, Z_0, q_0] \to b\,[q_0, Z, q_1]\,[q_1, Z_0, q_0]$

$[q_0, Z_0, q_1] \to b\,[q_0, Z, q_0]\,[q_0, Z_0, q_1]$

$b\,[q_0, Z, q_1]\,[q_1, Z_0, q_1]$

Recursive Languages

A language is recursive if there exists a Turing machine that accepts every string of language and rejects every string that is not in language.

Complement

If L is recursive, $\bar{L}$ must also

Recursively Enumerable Lang

if there exists a turing m/c

Turing M/c

Bhanu Priya

Infinite size tape and it is used to accept Recursive Enumerable languages.

TM can move in both directions.

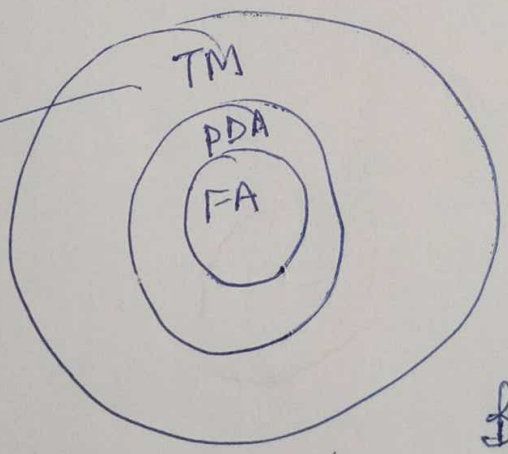
Also it does not accept ^.

If the string is not in lang, m/c will halt in non-final state.

TM is a mathematical model which consists of infinite length tape divided into cells on which input is given.

It consists of head which reads input tape.

If TM reaches the final state, then input string is accepted, otherwise rejected



Recursive Enumerable language

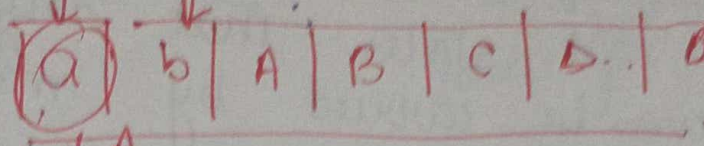
7 tuple = $(Q, \Sigma, \delta, q_0, F, B, \Gamma)$

Blank symbol

Blank symbol

$\delta : Q \times \Sigma \to Q \times \Sigma \times (L, R)$

Operations on Tape

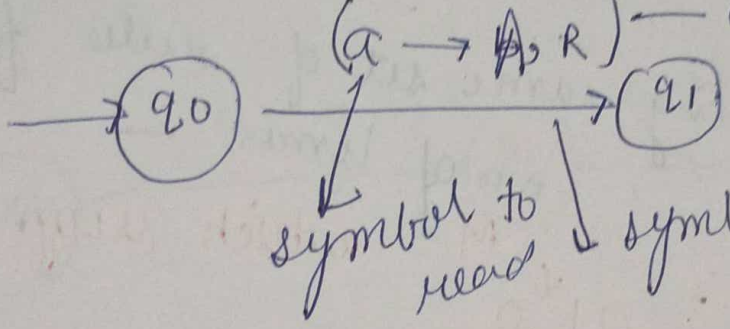


(1) Read / scan the symbol below tape head.

(2) Update / write a symbol below tape head

(3) Move the tape head one step left.

(4) Move the tape head one step right.



Turing Thesis

It states that any computation that can be carried by mechanical means can be performed by some T.M.

(1) Anything that can be done on existing digital computer can also be done by TM.

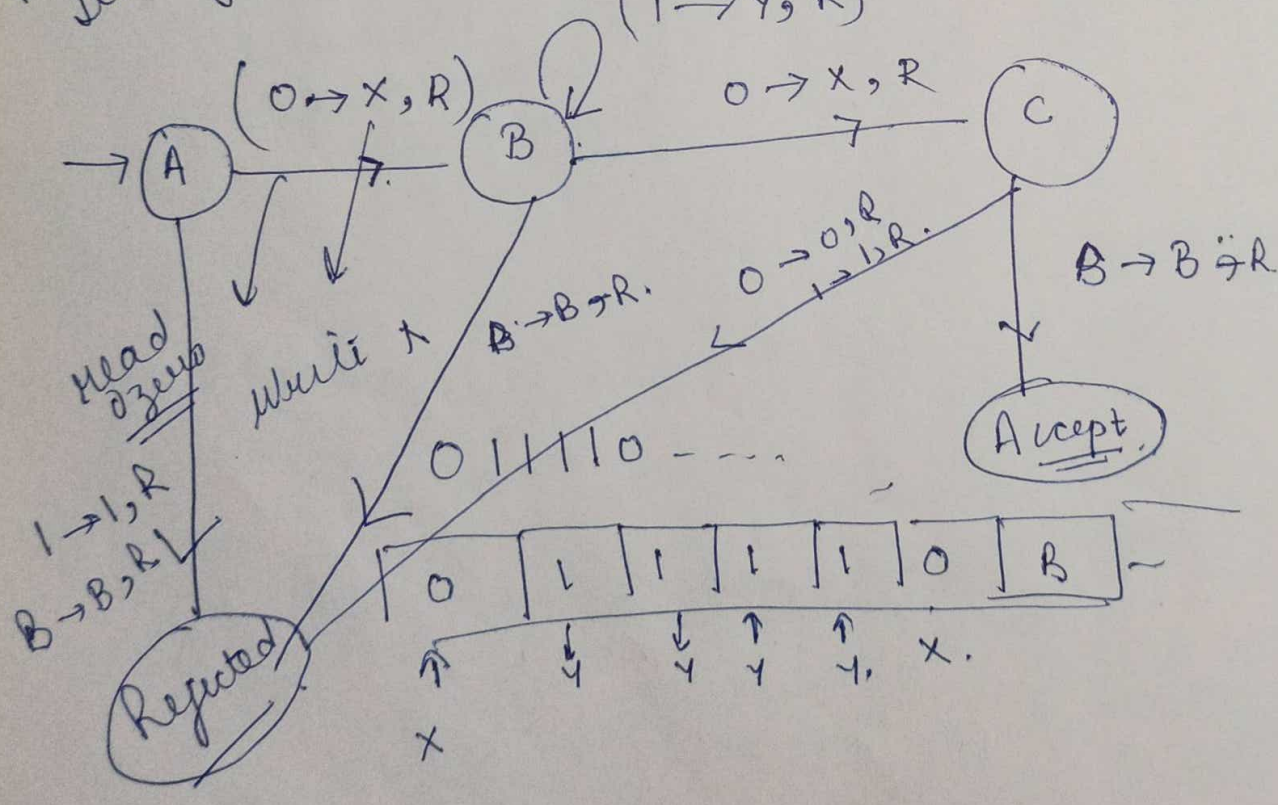
Repeating same set of rules for any no of times

REL → A language is Recursively enumerable if there exists a TM that accepts its list of elements

replacing same set of rules for any no of times

Q Design a Deterministic Turing M/c which recognize the lang $0 1^* 0$

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



Recursive → repeating same set of rules for any no. of times

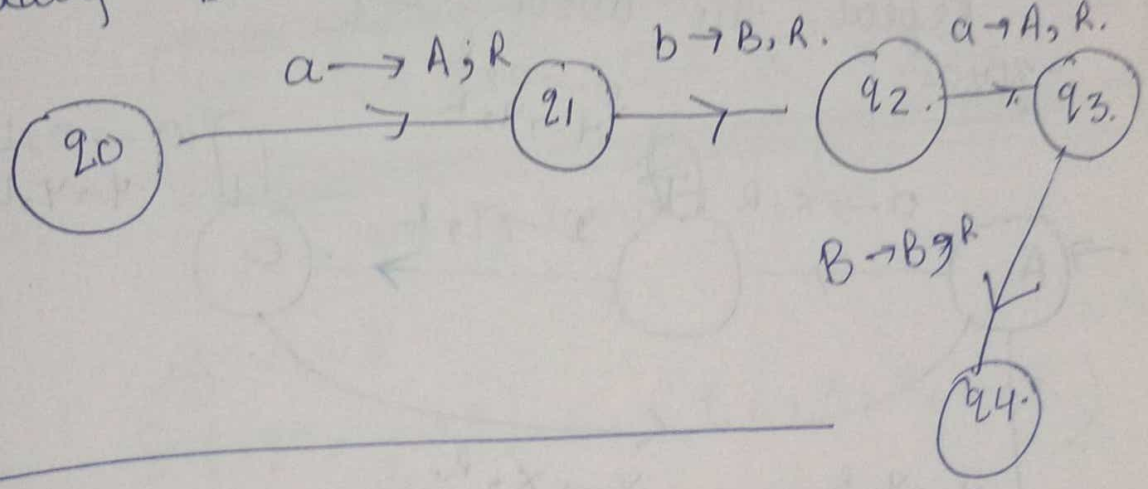
Enumerable means list of elements.

Construct a TM which accepts the language of aba ova $\Sigma = \{a, b\}$

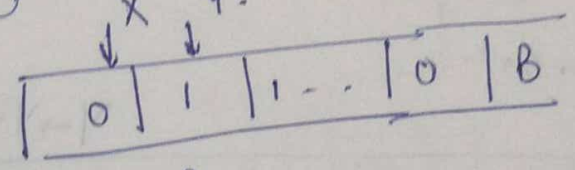
tape

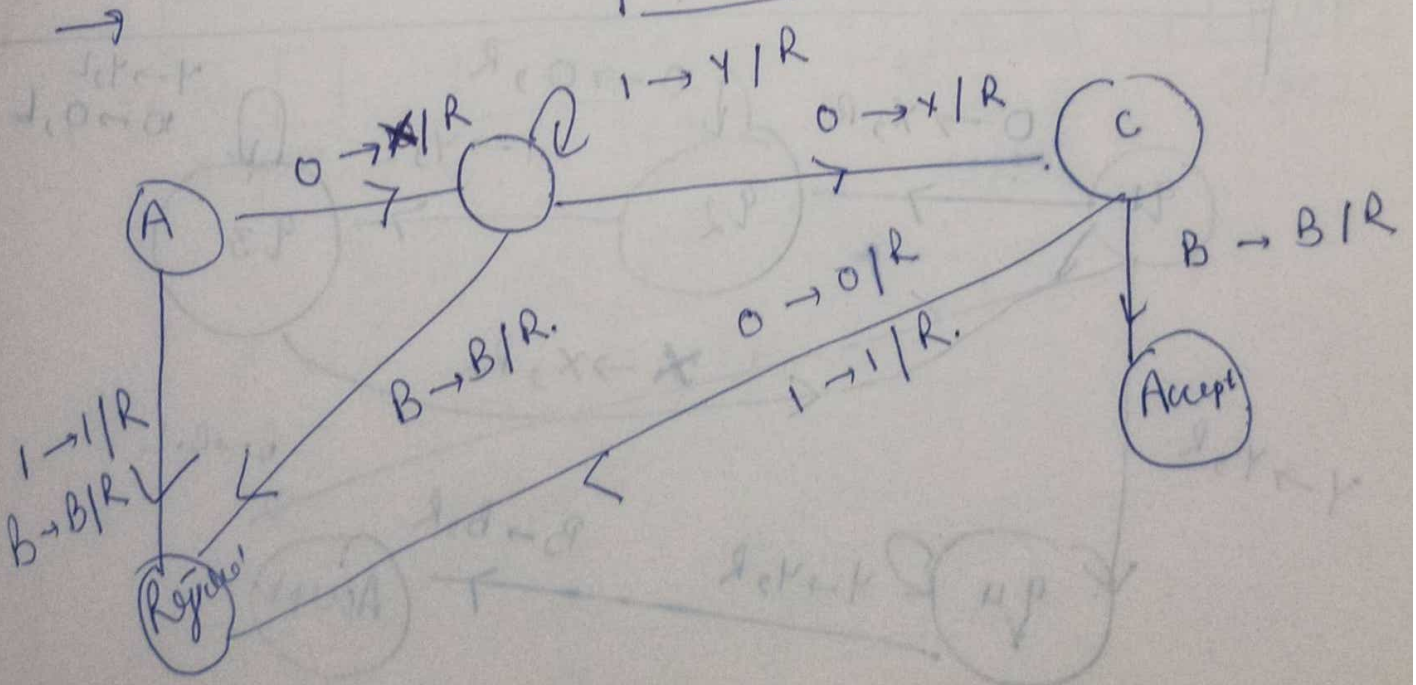
a	b	a	B	B	.....

If tape head is 'read out' 'aba' string reading B. then TM will halt after

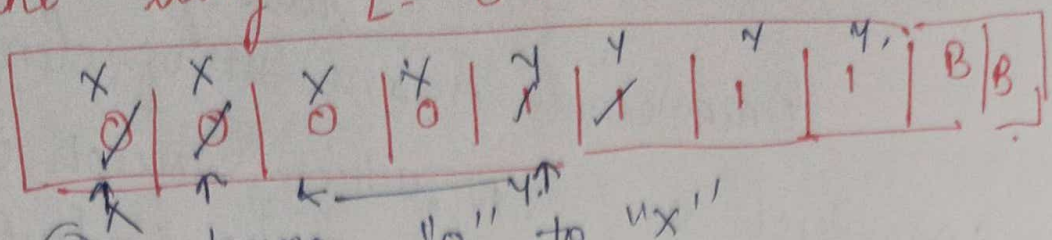


$0^1 1^4 0$





Q3 Design a TM which recognize the lang $L = 0^n 1^n$

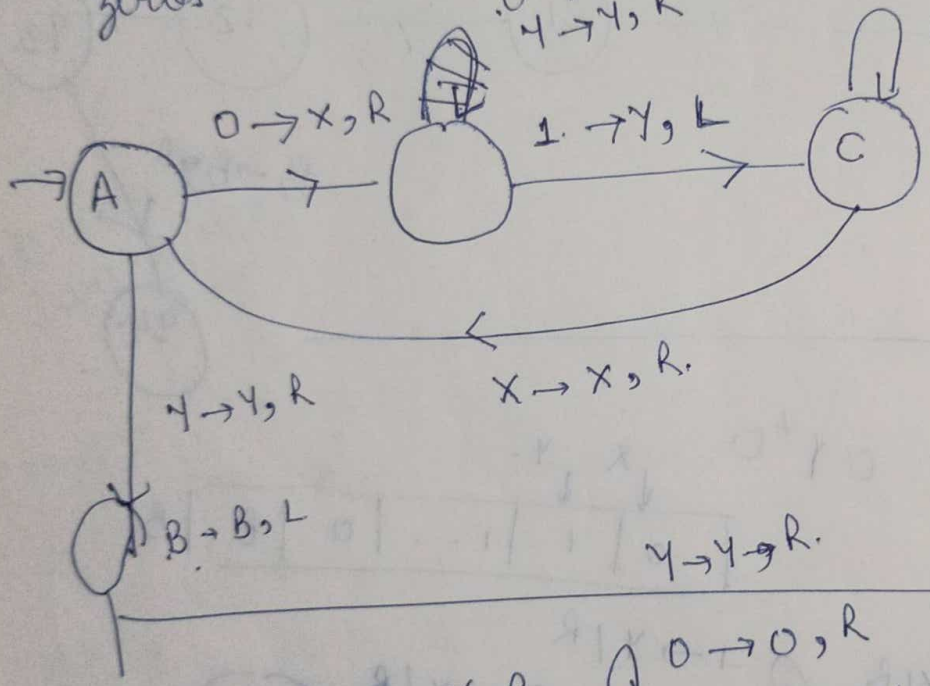


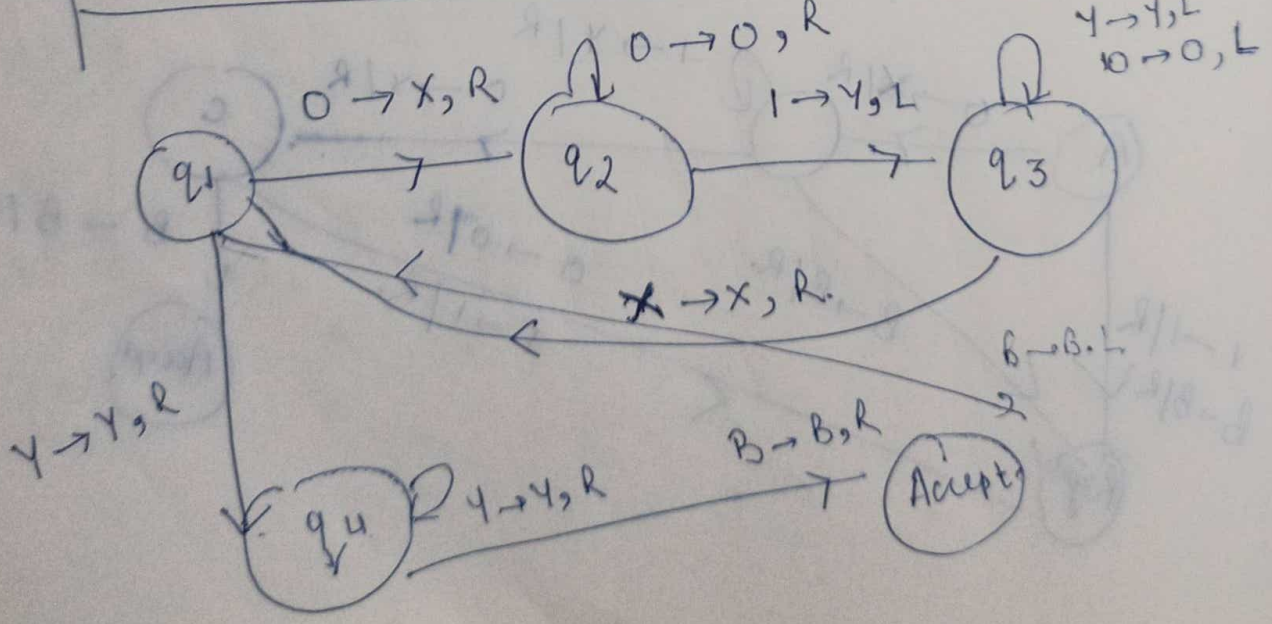
Soln. (1) Change "0" to "X"

(2) Move Right to first "1"
If none = Reject
⇒ Change "1" to "Y"

(3) Move left to left most 0

Repeat the above steps until no more zeros.





Variants of Turing M/C

Tape - infinite size, tape head and one finite control unit

Different variations of Turing Machines

→ Multitape Turing Machine
→ Non-Deterministic Turing M/C
→ Multihead Turing m/c.

(1) Multitape Turing M/C

A Turing m/c with several tapes.

→ every tapes have their controlled own R/w head.

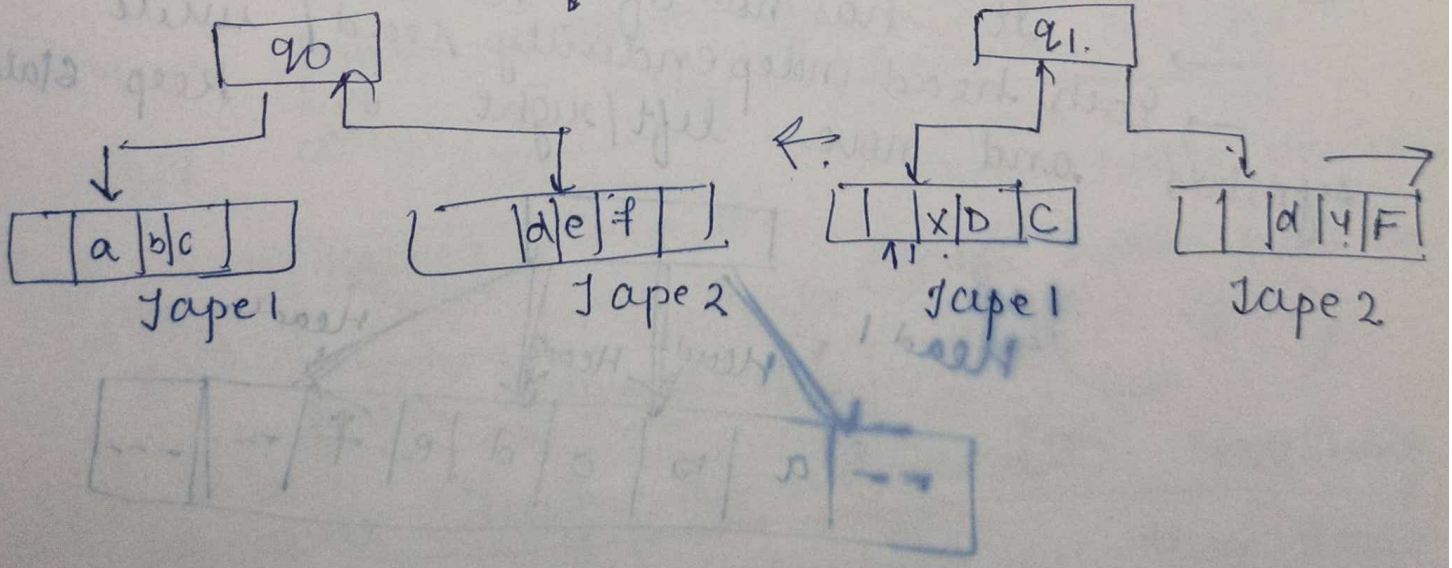
$$\delta : Q \times \Sigma^N = Q \times \Sigma^N \times (L, R)^N$$

Imp

N → No of Tapes

if $n = 2$

$$\delta(q_0, a, e) = (q_1, X, Y, L, R)$$



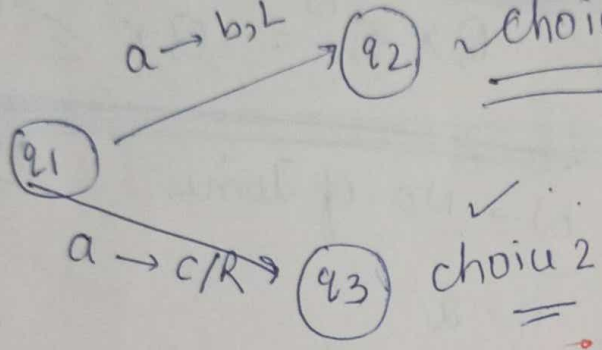
Non-deterministic Turing M/C

It is similar to DTM except that for any input & current state it has no. of choices.

A string is accepted by NDTM if there is a seq of moves that leads to final state

$$\delta : Q \times \Sigma \rightarrow 2^{Q \times \Sigma \times (L,R)}$$

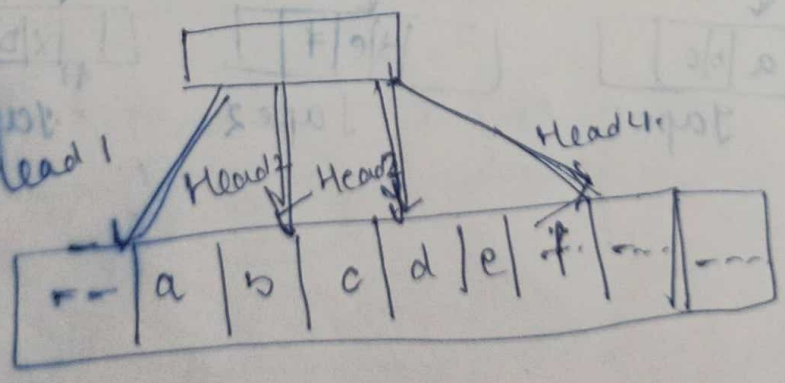
NDTM is allowed to have more than one transition for a given tape symbol



(3) Multihead Turing Machines

→ It has no. of heads instead of one.

→ Each head independently read / write symbols and move left/right or keep stationary



Multi-track TM

Halting problem → stopping problem → i/p. what will be.

Halting problem is undecidability.

It is not a problem. it just ask "question" is it possible to tell whether a given m/c will halt for some given i/p.

eg Input - A TM & i/p string w.

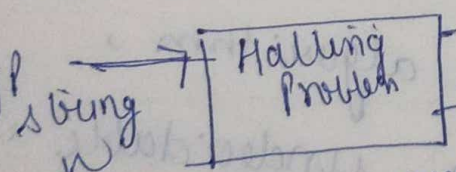
Problem Does the TM finish computing of string w in a finite no of steps.

(Answer must be Y or N) because undecidable.

Proof :- Assume TM exist to solve this problem & then we will show it is contradicting itself

We will call this TM as a Halting m/c that produce 'Yes' or 'No' in a finite amount of time

If the halting m/c finishes in a finite amount of time the o/p comes as 'Yes' otherwise as 'no'

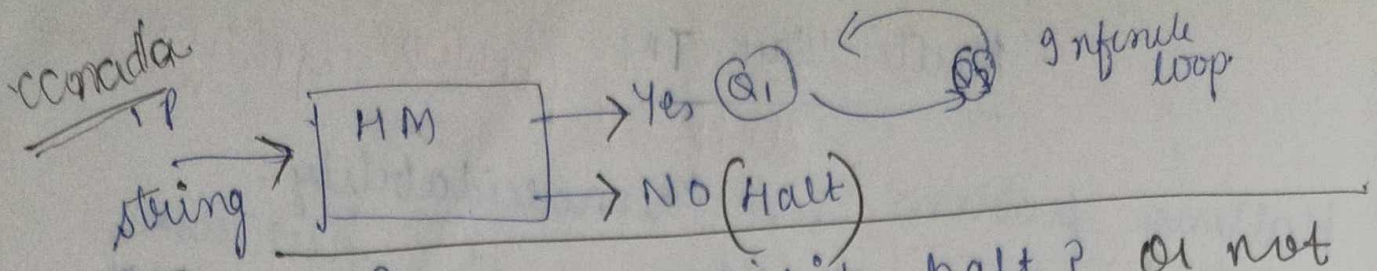


(HM halts on i/p w
(does not halt on i/p w)

finite amount of time

We will design an inverted halting m/c - If H returns Yes, then loop forever

→ If H returns No, then Halt



Given a Program : Will it halt? or not

Given a Turing M/c, will it halt when run on some particular given input string?

We can't design a generalized program algo that will whether a program will halt.

Java / given some program written in loop or will it ever get into an infinite loop or will it always terminate?

{ Accepts all valid Java code & never goes into infinite loop }

Ans

(1) In general we can't know

(2) The best we can do is run the program & see whether it halts

(3) For many programs we can see that it will always halt or sometimes loop

Undecidable

there is no Turing m/c.

can't have an algorithm.

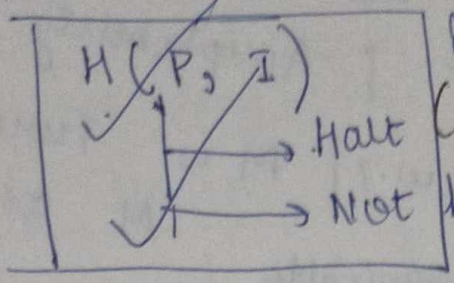
The halting problem is undecidable It ask.

Given a program Will it halt?

Can we design a machine which if given a program can find out or decide if that program will always halt or not halt on particular input?

We use contradiction

Assum Yes, we can



Program (Yes) Input → loop. (Halt)

If we pass program & input

C(x) (x is any program)
if{H(x, x) == Halt}
loop forever:
else
Return: (Halt).

If we Run C(x) on itself

C(C)

H(C,C) == Halt → Not halt

H(C,C) == Not Halt → Halt

No H does not exist
halting problem is undecidable

Universal Turing Machine

Turing m/c for all other Turing m/c.

The Language [Acceptability of TM]

$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is Turing M/c and } M \text{ accepts } w \}$

is Turing Recognizable.

Decidability & Undecidability.

Recursive Languages

→ A language 'L' is said to be recursive if there exists a Turing M/c which will accept all the strings in 'L' and rejects all the strings not in 'L'.

→ The Turing m/c will halt every time & give an answer (accepted or rejected) for each & every string input

Recursive enumerable

REL → A Language L is said to be REL if there exists a Turing m/c which will accept (and therefore halt) for all the input strings which are in 'L'

→ But may or may not halt for all input strings which are not in L

Decidable → If 'L' is decidable if it is a recursive languages

All decidable languages are Recursive language

Partially Decidable

'L' is a REL

Undecidable language

① If it is not decidable ✓

② may sometimes be partially decidable but not decidable

③ If a language is not even partially decidable, then there exists no Turing m/c for that language

Recursive Lang	TM will always halt
REL	will halt and may not halt
	RL
DL	
PDL	REL
Undecidable language	No Turing m/c for that language

Universal Turing Machines = behave like ...

↓ general purpose computer

$A_{TM} = \{ \langle M, w \rangle \mid M$ is a Turing M/c & M accepts $w \}$

✓ is Turing Recognizable (But not Decidable)

given a description of TM & some input, can we determine whether the m/c accepts it?

- Just simulate/ Run the TM on input

M Accepts w: our Algo will halt & Accept

M Rejects w: our Algo will halt & Reject

M Loops on w: our Algo will not Halt

$$U_{TM} = \langle M, w \rangle$$

Input: M → description of some TM
w → an input string for M.

Action: - Simulate M
- Behave just like M would (may accept, reject or loop)

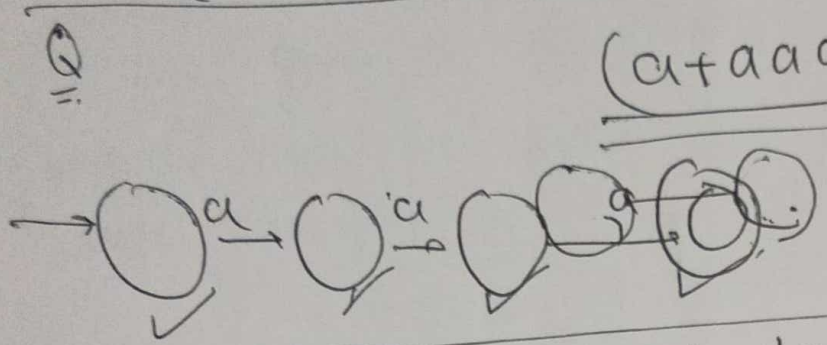
U_{TM} is a recognizer (but not decider).

Q. Which of following true?

(1) True
(2) False
(3) ~~True~~ False
(4) False.

CSL

Q.



$(a+aaa)^*$. a^*

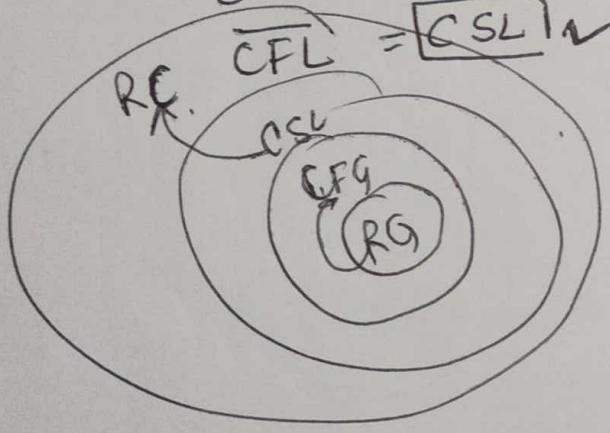
Minimum states = 1

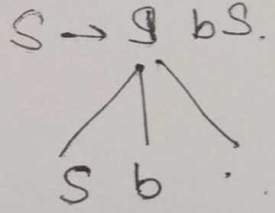
Q. S_1:. Non-deterministic TM can decide

False NDTM = standard Turing M/c

L is a context free.

False Recursive





G_1: $S \to SbS \mid a$

G_2: $S \to aB \mid ab.$

$A \to GAB \mid a$

$B \to ABb \mid b.$

S

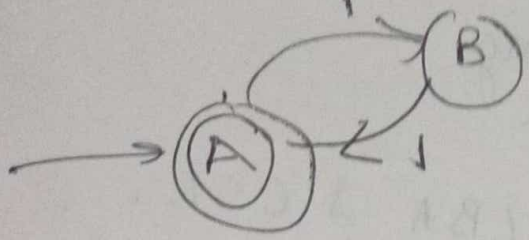
Q

CFG → complement, intersection

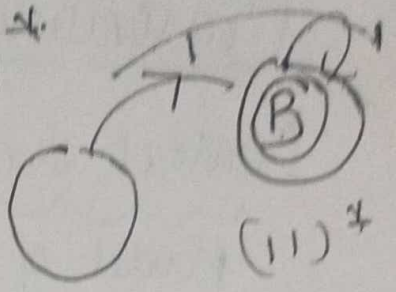
CFG & REL | CSG

complement set Difference

$L = \{11 + 111\}^*$



$(11)^*$



$(11)^*$

$\{\wedge, 11, 111, 11111, 1111, \ldots$

$(11 + 111)^*$

$1(11)^*$

$(11)^*$

$11(\wedge + 1)^*$

$11\,1^*$

$1^*(11)^*$

$1^* \cdot (11)^*$

$(11[\wedge + 1])^*$

$(11\,1^*)^*$

$(1^+1)^*$

$\Sigma = \{\overset{0}{\wedge}, \overset{2}{11}, \overset{3}{111}, 1111, \overset{5}{11111}, \overset{6}{111111}, \ldots$

$\overset{4}{1111}$

$(11)^*$

4