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

Advantages → ① Length of each production is restricted.

imp ② Derivation tree and parse tree obtained from CNF is always 'binary tree'

imp ③ The number of steps required to derive a string of length ~~f(2|w|-1)~~ |w| is (2|w|-1).

④ It is easy to apply CYK membership Algorithm.

EXPLAIN

Ex →

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

String → ab

$S \to AB$
$\to aB$
$\to ab$

no. of steps = 3
= 2(2)-1
↓
strleng(ab);

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

a	b	b
A	B	
A	B	B
a	B	B
a	b	B
a	b	b

5 steps = 2(3)-1

abb of string length 3.

Note.

CYK is an Algo. with polynomial time $O(n^3)$ and it can useful for every grammar in CNF.
↳ Also called Universal Algorithm.

→ Parsers are 'linear time Algorithm' but they can't apply to every grammar.

Convert into CNF

$S \to bA \mid aB$
$A \to bAA \mid as \mid a$
$B \to aBB \mid bs \mid a$

It can have variable answers so doesn't asked in GATE yet.

Procedure → Add new Variables as-

$N_a \longrightarrow a$

$N_b \longrightarrow b$

So, $S \longrightarrow N_bA \mid N_aB$

$A \longrightarrow N_bAA \mid N_aS \mid a$

$B \longrightarrow N_aBB \mid N_bS \mid b$

$N_a \longrightarrow a$

$N_b \longrightarrow b$

N_bAA and N_aBB are not in CNF.

So,

$S \longrightarrow N_bA \mid N_aB$

$A \longrightarrow N_bS \mid N_aS \mid a$

$B \longrightarrow N_aT \mid N_bS \mid b$

$N_a \longrightarrow a$

$N_b \longrightarrow b$

$S \longrightarrow AA$

$T \longrightarrow BB$

So all is form — $A \longrightarrow \alpha B'$ or $A \longrightarrow \alpha$

CYK Algorithms (only for CNF)

Ex → Check whether the string "baaba" is a valid member of the following CFG.

CFG

$S \longrightarrow AB \mid BC$

$A \longrightarrow BA \mid a$

$B \longrightarrow CC \mid b$

$C \longrightarrow AB \mid a$

Soln

1	2	3	4	5
b	a	a	b	a

	5	4	3	2	1
1	(S) A, C	Φ	Φ	A, S	B
2	S, C, A	B	B	C, A	
3	B	S, C	C, A		
4	A, S	B			
(5)	A, C				

(5) → string length.

Start filling with (1,1), (2,2), (3,3) ... (5,5) such that (1,1) fill with the LHS of production that contains 'b' in RHS because in string 'baaba' at position 1 letter is b

Similarly

(3,3), (2,2) ⟶ for 'a' so A, C

(4,4) ⟶ for 'b' so, B

(5,5) ⟶ for 'a' again so A, C.

then (2,1) ⟶ start with 1 ends with 2

(2,1) ⟶ (1,1) (2,2)
{B} {A,C}
Concatenate {BA, BC}

See productions that contains BA and BC in RHS.

(2,3) ⟶ (2,2) (3,3)
(A,C) (A,C)
(AA, AC, CA, CC)

(3,4) ⟶ (3,3) (4,4)
(AC) (B)
(AB, CB)

(1,3) ⟶ (1,1) (2,3) or (1,2) (3,3)
(B) (B) (A,S) (A,C)
BB AA, AC, SA, SC

No way to generate (1,3) directly.

(2,4) ⟶ B only

(3,5) ⟶ (3,4) (5,5) or (3,3) (4,5)
(S,C) (A,C) (A,C) (A,S)
SA, CA, CC, SC AA, AS, CA, CS
↓(B) X

(1,4) ⟶ (1,1)(2,4) ⟶ (B)(B) ⟶ BB X

(1,2)(3,4) ⟶ (A,S)(S,C) ⟶ AS, AC, SS, SC X

(1,3)(4,4) ⟶ {ϕ}(B) split not possible

(2,5) — (2,2)(3,5)
(C,A) (B) → CB, AB

(2,3)(4,5)
(B) (A,S) → BA, BS

(2,4) (5,5)
B AC → BA, BC

(1,5) → (1,1) (2,5)
(B) (S,C,A) → BS, BC, BA

(1,2) (3,5)
(A,S) B → AB, SB

(1,3) (4,5)
ϕ → string split not possible

(1,4) (5,5)
ϕ → split not possible

Since final cell (1,5) contains starting symbol 'S'.
So we can say that string can be generated from given Grammar.

Substring generated from given string 'baaba' using the grammar then we need table

Consider all cells other than first cell that contains S inside are —

(1,2) → b a a b a (1 2 3 4 5)
(1,2) → 'ba'

(2,5) → b a a b a (1 2 3 4 5)
(2,5) → aaba

(3,4) → b a a b a (1 2 3 4 5)
(3,4) → 'ab'

(4,5) → b a a b a (1 2 3 4 5)
(4,5) → 'ba'

substrings generated are —

(ba, aaba, ab)

Time Complexity → $O(n^3)$ for CYK or membership Algo.

$n \rightarrow$ no. of cells used as

$1+2+3+4+5 \rightarrow$ for 5 characters in string.

for n character string we need

$1+2+3--+n = \frac{n(n+1)}{2} = O(n^2)$

To remember

GATE → In GATE they may ask to fill the blanks in table or may ask for substrings generated.

GNF:- If all the productions are of form

$A \rightarrow a\alpha$ where $\alpha \in V^*$, then it is GNF.

Ex: $A \rightarrow aBCDE$

$A \rightarrow a$

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

② GNF is useful in order to convert CFG to PDA.

Conversion of CFG to PDA

1) Push the start symbol 'A' on the stack.

$\delta(q_0, \epsilon, Z_0) = (q_1, AZ_0)$

2) Push RHS of A as follows

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

if $A \rightarrow a\alpha$ is in 'GNF'

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

if $A \rightarrow b\beta$ is in 'GNF'

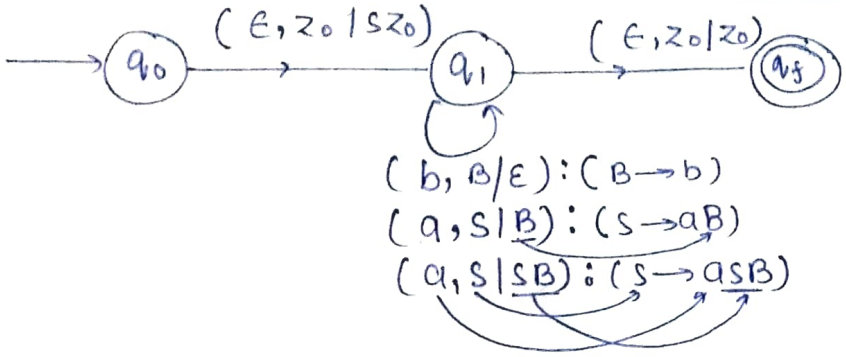
3) Add final state with

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

Ex→ Convert the following into PDA: $S \rightarrow asb \mid ab$

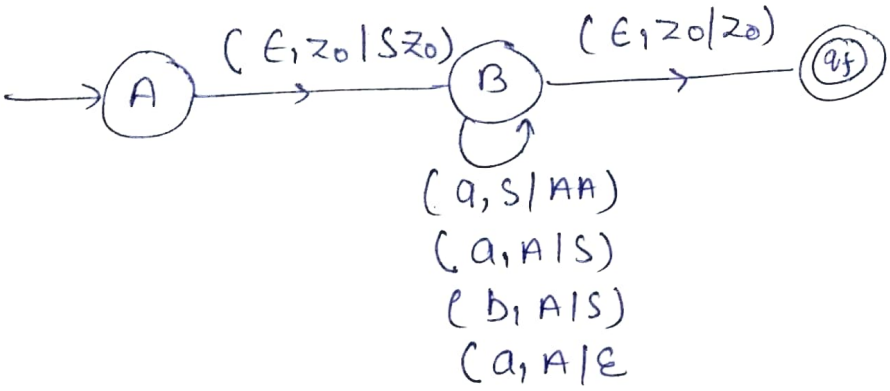
Soln GNF $S \rightarrow aSB \mid aB$

$B \rightarrow b$



Not imp. for gate just look at it for interview purpose

(2) $S \longrightarrow aAA$
$A \longrightarrow aS \mid bS \mid a$

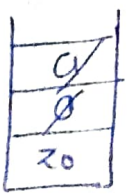


Push Down Automata (PDA)

→ Machine that accept CFLs
→ Finite Automata with memory / stack
→ Tuples $< Q, \Sigma, \delta, q_0, z_0, F, \Gamma >$
- $Q, \Sigma, \delta, q_0, F$ → Same as finite Automata
- z_0 → Bottom of stack
- Γ → Stack Alphabet

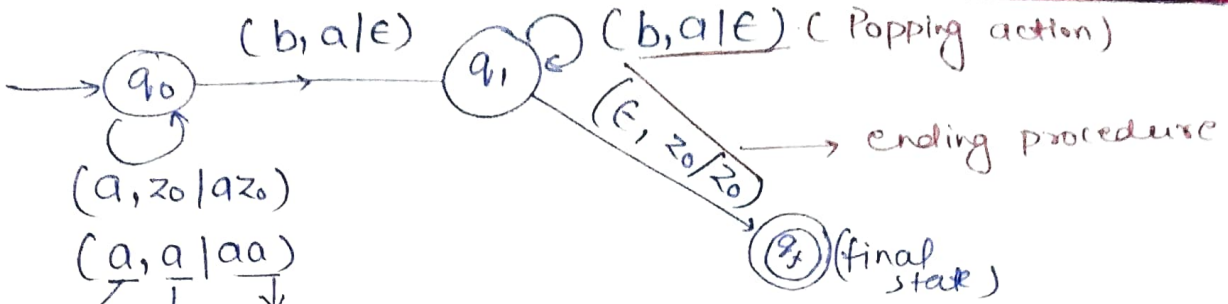
$\delta : Q \times \{\Sigma \cup \epsilon\} \times \Gamma \longrightarrow Q \times \Gamma^*$ → Deterministic PDA
$\delta : Q \times \{\Sigma \cup \epsilon\} \times \Gamma \longrightarrow 2^{(Q \times \Gamma^*)}$ → Non-Deterministic PDA

Ex: (1) $a^n b^n \mid n \geq 1$



a a b b ε

on 'a' push it in stack and on 'b' pop a from stack. and at the end if you reach ε and z_0 is at top of stack then string is accepted.



State transition Diagram
(acceptance by final state)

Transition function

$$\delta(q_0, a, z_0) = (q_0, az_0)$$

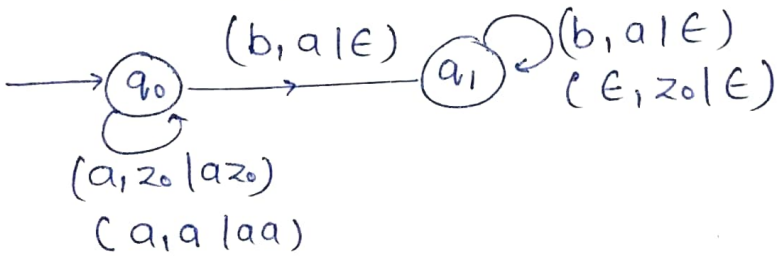
$$\delta(q_0, a, a) = (q_0, aa)$$

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

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

$$\delta(q_1, \epsilon, z_0) = (q_f, z_0) \text{ or } (q_1, \epsilon)$$

Above diagram can be represented as 'acceptance by empty stack' as –



Note → Both deterministic and non-deterministic PDA and acceptance by final state or empty stack are equivalent in powers

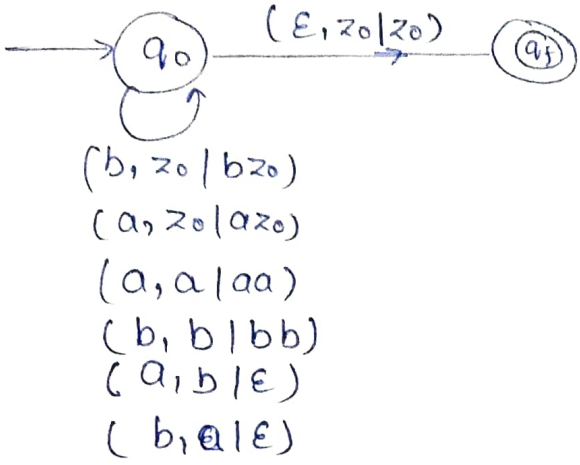
---

(2) $\omega \mid n_a(\omega) = n_b(\omega)$

It is not necessary that 'b' comes after 'a' or no 'a' can comes after 'b'. Every combination is allowed.

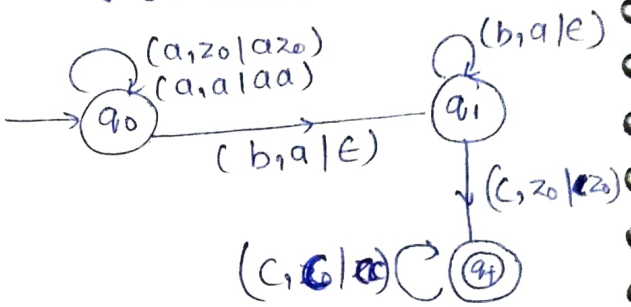
bbaa
abab
abba

Pop if opposite symbol is already pushed either push it inside the stack. → Logic



③ $a^n b^n c^m \mid n, m \geq 1$

c^m is not depend on other then just count them for 1 or more than 1

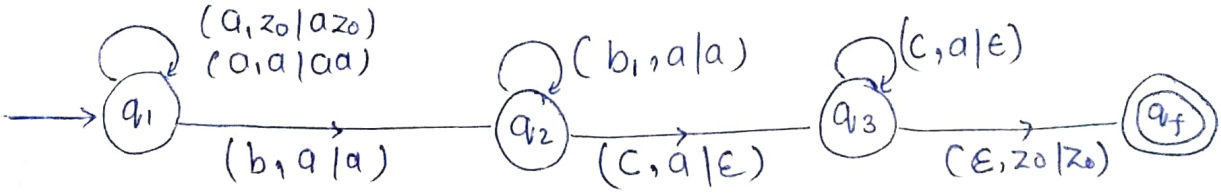


→ If a string that doesn't defined by PDA leads to dead configuration.

→ All the PDA's discussed above are Deterministic PDA's.

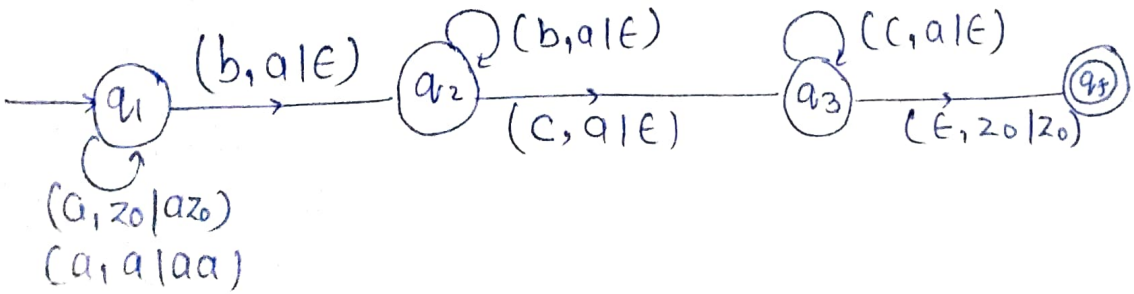
→ If string belongs to CFL we will haulet on final state otherwise haulet in between but we will definitely get haulted.

④ $a^n b^m c^n \mid n, m \geq 1$



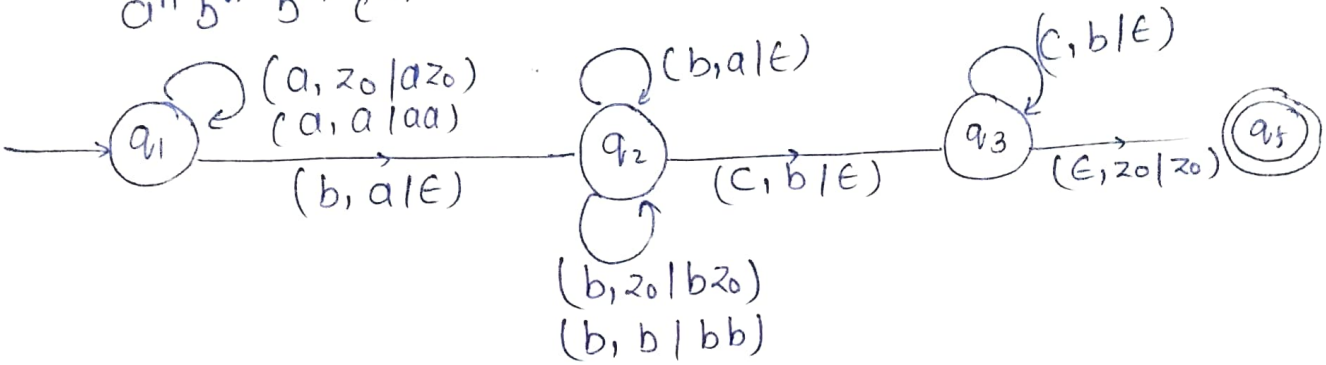
This PDA can't form in less then 4 states.

⑤ $a^m b^m c^n \mid m, n \geq 1$

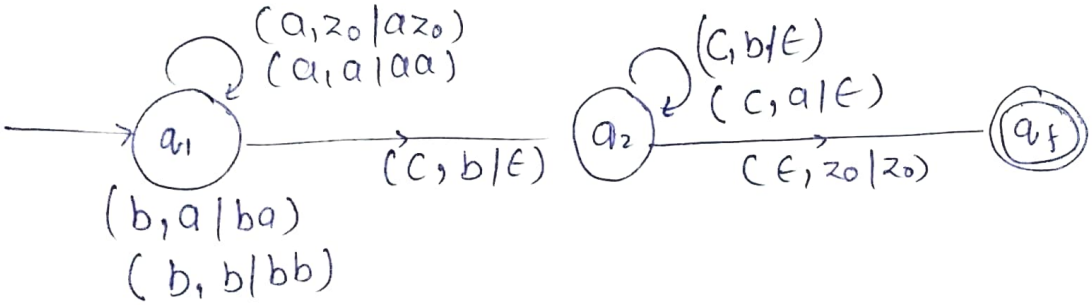


(6) $a^n b^{m+n} c^m \mid n, m \geq 1$

$a^n b^n b^m c^m$

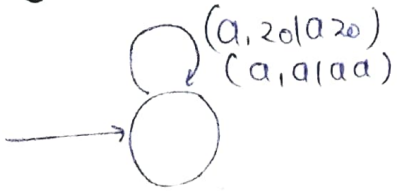


(7) $a^n b^m c^{n+m} \mid n, m \geq 1$



(8) $a^n b^n c^m d^m \mid n, m \geq 1$ (H.W)

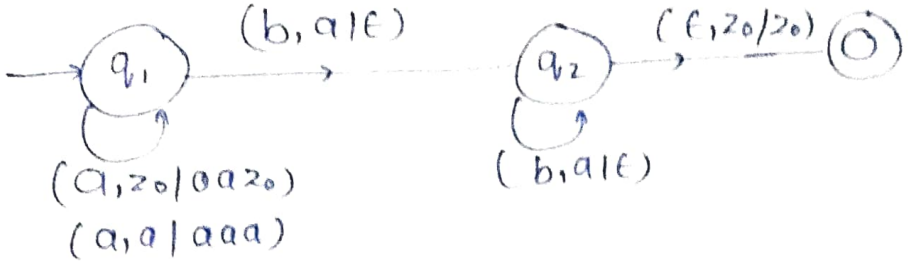
(9) $a^n b^m c^m d^n \mid n, m \geq 1$ (H.W)



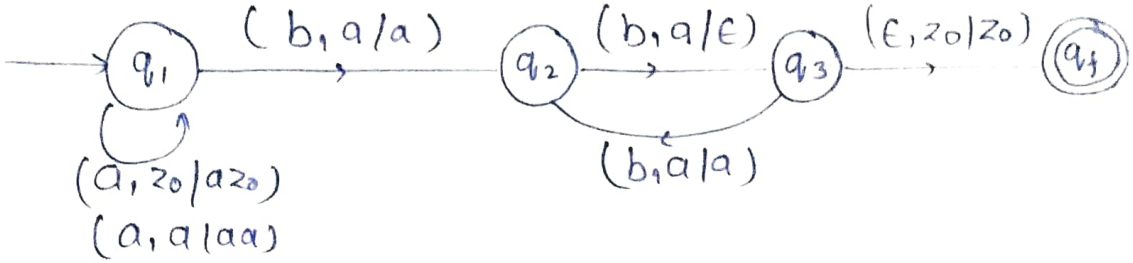
(10) $a^n b^m c^n d^m \mid n, m \geq 1$ → Language is not CFG and we can't give PDA for this.

(11) $a^n b^{2n} \mid n \geq 1$

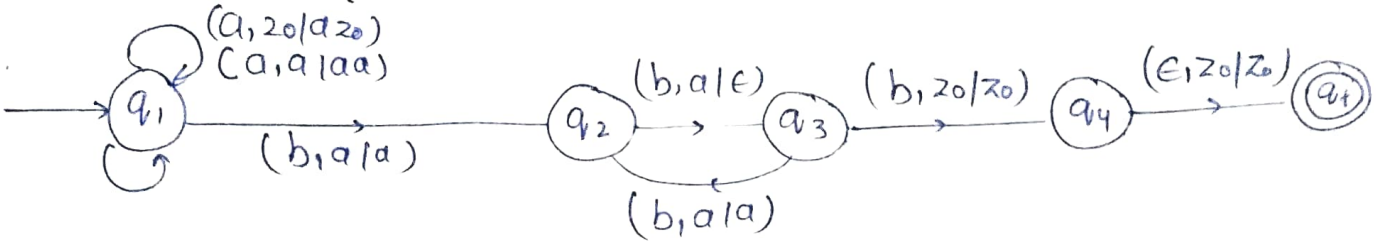
(i) - for every "a" push 2 "a's".



(ii) Remove one 'a' for two "b's".



(12) $a^n b^{2n+1} \mid n \geq 1$



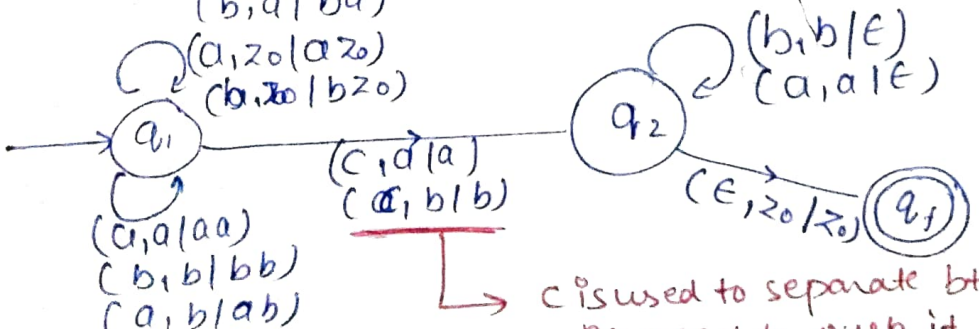
(13) $a^n b^n c^n \mid n \geq 1$

We may form a PDA in which push 2 a's for a single a and pop every a for each b and c but it will accept the language 'aabccc'

So, PDA is not possible because language is not CFL.

(14) $wcw^R \mid w \in (a+b)^+$

Save w and match w^R correspondingly.



c is used to separate b/w w and w^R so no need to push it inside the stack

Start popping operation just after seeing 'c'.

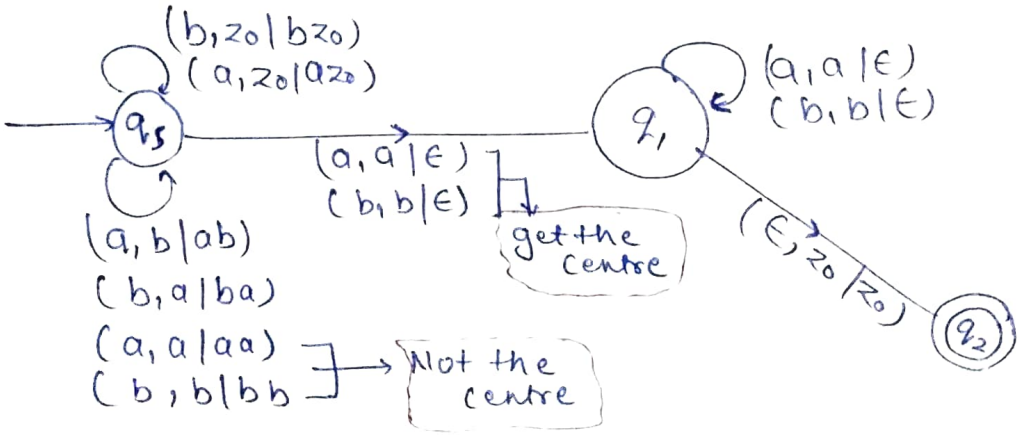
(15) $wwR \mid w \in (a+b)^+$ → even length palindrome

→ Here, like previous example there no centre like 'c' so that we can separate w and wR.

→ So we can assume that at centre the input symbol and top most symbol of stack should be same but it may leads to problem that if same case experienced in not centre then it will create conflicts.

→ So we can't acheive it using DPDA, we require a NPDA that contains 2 part :-

1) Assume that centre is here "abaaba"
2) Not the centre "aaaa"



(q0, aaaa, Z0) → Instantaneous Description

- q0 → Starting state
- a/b → looking for
- Z0 → topmost symbol

(q0, aaaa, aZ0)

No centre | Centre

(q0, aa, aaZ0) | (q1, aa, Z0) [Popping performed]

(q0, a, aaaZ0) | (q1, a, aZ0) | (q1, ∈, Z0)

(q0, ∈, aaaaZ0) | (q1, ∈, aaZ0) | (qf, ∈, Z0) → Accepted final state.

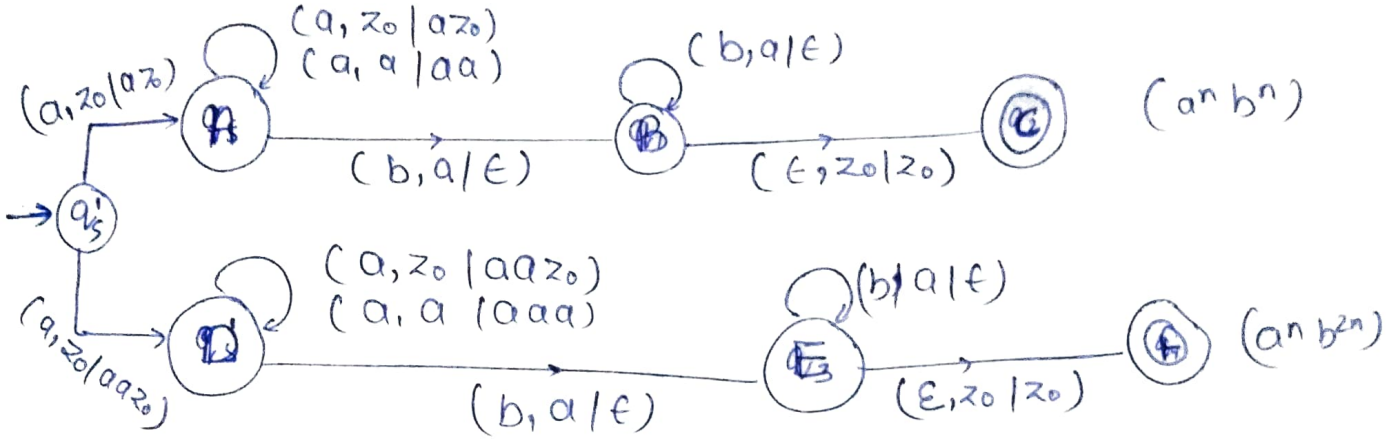
(X) Dead configurations (X) (X) Dead configuration so PDA died. bcoz we don't define any transition over (q1, a, Z0).

→ NPDA ⊃ DPDA → NPDA is much powerful than DPDA and DPDA is proper subset of NPDA.

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

L_1 L_2

→ You have to decide just by looking first symbol that string belongs to L_1 or L_2. so its NPDA.



$(q_s, aabb, z_0)$

$(a^n b^n)$ | $(a^n b^{2n})$

(A, abb, az_0) → (A, bb, aaz_0) → (B, b, az_0) → $(B, \in, z_0)$ → $(C, \in, z_0)$

Reaches final state (Accepted)

(D, abb, aaz_0) → $(D, bb, aaaaz_0)$ → $(E, b, aaaz_0)$ → $(E, \in, aaz_0)$

(Dead configuration)

Ex-

$L = \{a^i b^j c^k d^l \mid i = k \text{ or } j = l\}$

If $i = k$ and $j = l$ then not a CFL but here is 'OR' symbol between.

$= \{ a^m b^i c^m d^l \} \cup \{ a^i b^m c^k d^m \}$

$i = k = m$ (say)

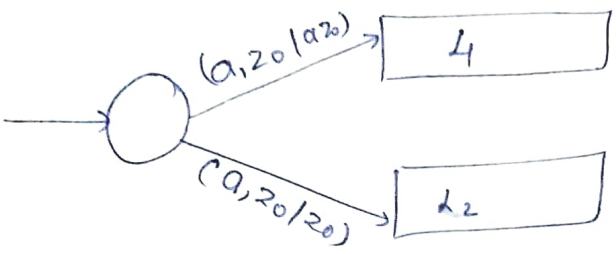
↓

push a's leave b's then pop c's and leave d's

$j = l = m$ (say)

↓

leave a's push b's leave c's and then pop d's.



Anyone from both can leads to correct answers. Do it by self, leave as homework.

Testing CFG or not

1) $a^{m+n} b^n c^m \mid n, m \geq 1$ → Not a regular lang. but PDA possible due to CFL.

2) $a^m b^{m+n} c^n \mid n, m \geq 1$ → DCFL and CFL

3) $a^m b^n c^{m+n} \mid n, m \geq 1$ → DCFL and CFL

4) $a^m b^m c^n d^n \mid m, n \geq 1$ → DCFL and CFL

5) $a^m b^n c^m d^n \mid m, n \geq 1$ → 'Not CFL'.

6) $a^m b^n c^n d^m \mid m, n \geq 1$ → DCFL and CFL

7) $a^m b^i c^m d^k \mid m, n \geq 1$ → Not regular but DCFL and CFL

8) $a^m b^n \mid m > n$ → Not regular but DCFL and CFL

9) $a^n b^{2n} \mid n \geq 1$ → DCFL and CFL.

10) $a^n b^{n^2} \mid n \geq 1$ ⟶ length of 'b' should be in AP so not a CFL.

11) $a^n b^{2^n} \mid n \geq 1$ ⟶ Not a CFL, not regular

12) $ww^R \mid w \in (a,b)^*$ ⟶ NCFL and CFL

13) $ww \mid w \in (a,b)^*$ ⟶ Not regular and not CFL.

14) $a^n b^n c^m \mid n > m$ ⟶ Not regular and not CFL

15) $a^n b^n c^n d^n \mid n \leq 10^{10}$ ⟶ Regular so DCFL and CFL

16) $a^n b^{2n} c^{3n} \mid n \geq 1$ ⟶ You can compare only two symbol at a time. So not a CFL.

17) $xcy \mid x,y \in (0,1)^*$ ⟶ Regular and DCFL ⟶ CFL.

18) $xx^R \mid x \in (a,b)^* \; |x| = l$ ⟶ Regular and CFL.

19) $wwww^R \mid w \in (a,b)^*$ ⟶ (ww) is not CFL so $wwww^R$ is also not CFL.

20) $a^n b^{3^n} \mid n \geq 1$ ⟶ not a CFL bcoz 3^n not in AP.

21) $a^m b^n \mid m \neq n$ ⟶ CFL with DCFL

22) $a^m b^n \mid m = 2n+1$ ⟶ CFL and DCFL

23) $a^i b^j \mid i \neq 2j+1$ ⟶ DCFL and CFL

24) $a^{n^2} \mid n \geq 1$ ⟶ DCFL and CFL doesn't exist.

25) $a^{2^n} \mid n \geq 1$ ⟶ Not a CFL.

28) $a^{n!} \mid n \geq 1$ ⟶ Not a CFL

27) $a^m \mid m$ is prime ⟶ Not a CFL

28) $a^m \mid m$ is even ⟶ CFL and regular

29) $a^i b^j c^k \mid i > j > k$ → Not a CFL bcoz comparison between all 3 is not possible.

30) $a^i b^j c^k \mid j = i + k$ → Context free Language and DCFL

31) $a^i b^j c^k d^l \mid i = k$ or $j = l$ → CFL and NCFL.

32) $a^i b^j c^k a^l \mid i = k$ and $j = l$ → $a^m b^n c^m d^n$

we can't compare 'c' with 'a' bcoz 'b' is on the top of stack. So not a CFL.

33) $a^m b^l c^k d^n \mid m, l, k, n \geq 1$ → Regular and DCFL so CFL

34) $a^n b^{4m} \mid n, m \geq 1$ → Regular and CFL

35) $\{a^3, a^8, a^{13}, \ldots\}$ → RL and CFL

36) $\{a^{2n+1} \mid n \geq 1\}$ → RL and CFL

37) $(a^*)^n \mid n \geq 1$ → Not a CFL

38) $\{w \mid w \in \{a,b\}^*, |w| \geq 100\}$ → RL and CFL

39) $\{w \mid w \in (a,b,c)^*, n_a(w) = n_b(w) = n_c(w)\}$ → Not CFL

40) $\{w \mid w \in \{a,b\}, n_a(w) \geq n_b(w) + 1\}$ → CFL exists

Shows the extra "a's" ← (ϵ, a^+) or (a, a^*)

ooo

Note → All the examples are no need to mug up just remember the concept and atleast go through all examples. So that you can solve other possible questions.

Pumping Lemma for CFL to check whether the language is CFL or not

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

$w = abcde$ such that

i) $bd \neq \epsilon$

ii) $|bcd| \leq n$

iii) for all $i \geq 0$, the string $ab^ic d^i e$ is also in L.

Pumping

→ It is also the negative test means if language doesn't pass the test we can confirmly say that language is not CFL 'but if it passes we can't confirmly says that Language is CFL or not.'

Note

1. Every CFL satisfies pumping lemma property.
2. If L doesn't satisfy Pumping Lemma, it is not CFL.

Ex $a^n b^n c^n$ $(n \geq 1)$ is not a context free language. Show it.

$w = a\,b\,a\,a\,b\,b\,b\,c\,c\,c\,c$ (b: 2nd–3rd, d: 5th–7th)

In this case on pumping b and d no change found.

→ We have no need to understand this. It is not going to ask in GATE. Just read the theory for direct questions.

④ Turing Machine

→ It can move in both direction and read/write both on the tape.

→ It is not accepting ∈ (Epsilon).

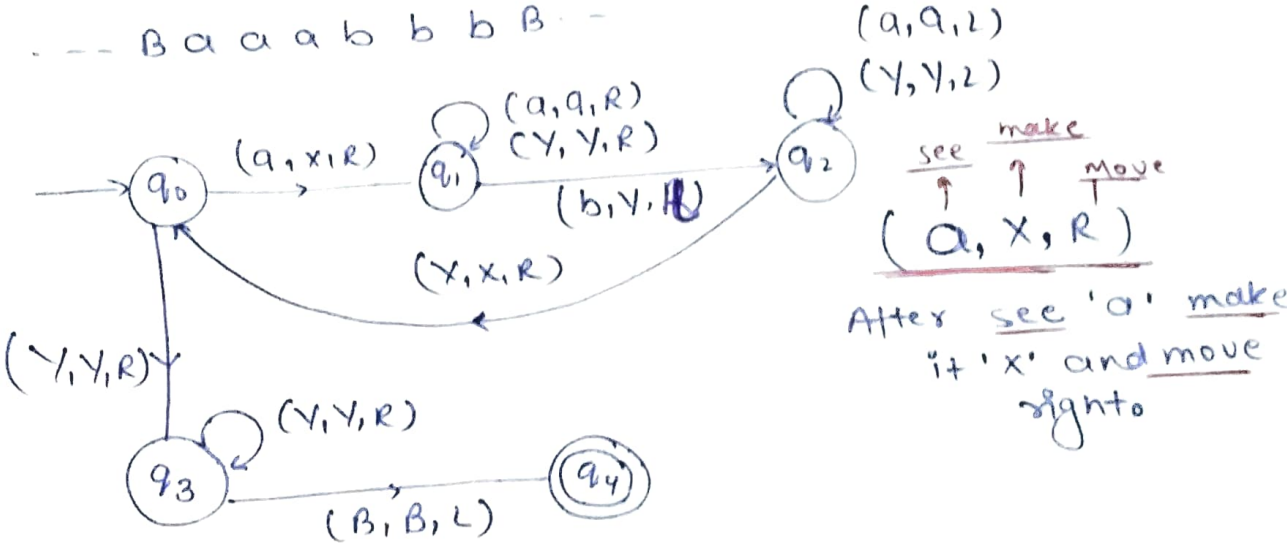
Ex-1 $a^n b^n \mid n \geq 1$ → Not a Regular but CFL

B	a	a	b	b	B

q_x — q_y — q_z

make each 'a' → x shows it is already accused and then search for a 'b' to make it 'y'.

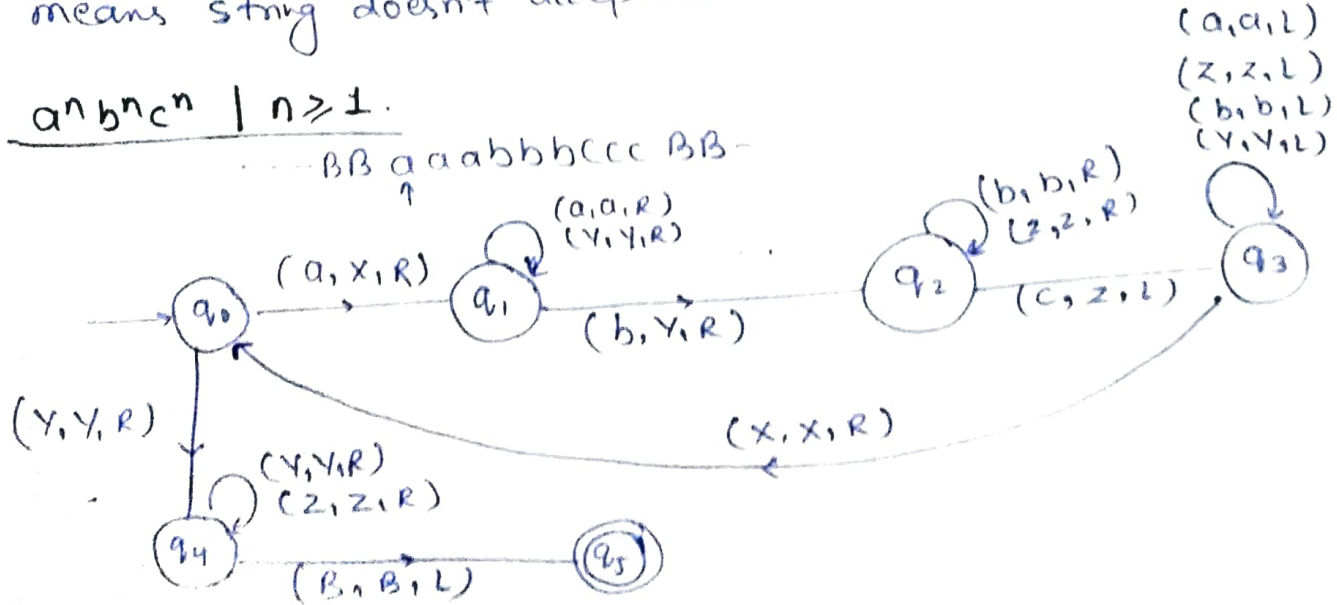
... B a a a b b b B ...



If we give a string BB aa bbb BB then it will get halt at q_3 and this is called dead configuration means string doesn't accepted.

② $a^n b^n c^n \mid n \geq 1$.

... BB a a a b b b c c c BB ...



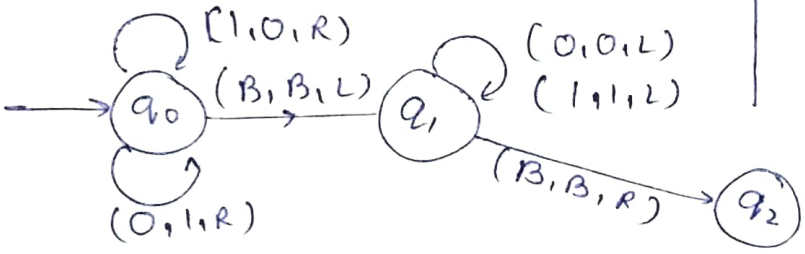
TM as Transducer of 1's Complement

Note → Other than Turing Machine all languages are work as acceptor means they results in 'Yes' or 'No' that string will get accepted or not.

But Turing Machine act as acceptor as well as Transducers.

Ex → Binary No. 0011
1's Comp 1100

B B 0 0 1 1 B B



* Here Turing Machine work as a Transducer means it converts binary to 1's complement

* Since it is not working as a acceptor so 'no' need of final state.

* We are converting binary to 1's complement, not checking about acceptence of string.

2's Complement

Assuming that there is no overflow means binary no. and its 2's complement have same no. of bit's.

Logic

0 1 1 1 0 0 0
1 0 0 0 1 1 1
1 0 0 1 0 0 0

Compare

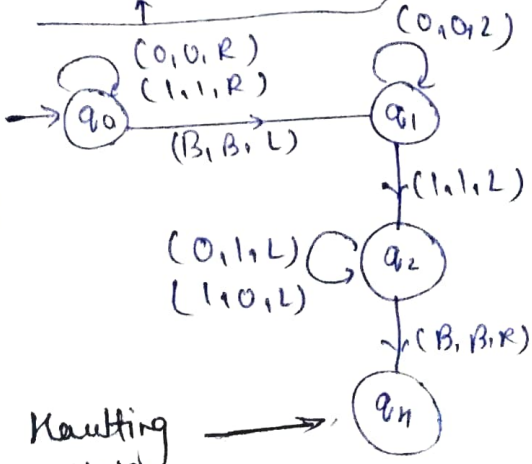
reverse/flip the digits.

Leave all 0's as it is

Leave 1's one as it is

Note → We have do the procedure from RHS to LHS to first go to the Rightmost symbol of input.

B B 0 1 1 1 0 0 0 B B B



Halting state → q_H

Turing Machine as Adder (Transducer)

Unary representation.

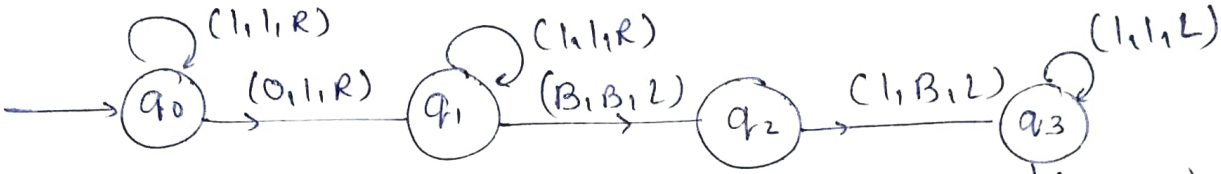
Ex $3 = 111$, $4 = 1111$, $2 = 11$

means if $3 = 111$ (say x)

then $|x| = 3$ (length of string)

So, $3 + 4 \rightarrow 111 + 1111$

$\rightarrow$ B B 1 1 1 0 1 1 1 1 B B B $\rightarrow$ Count no. of 1's to get sum.



At the end

B B 1 1 1 0 1 1 1 1 B B

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

(x+4) on the tape.

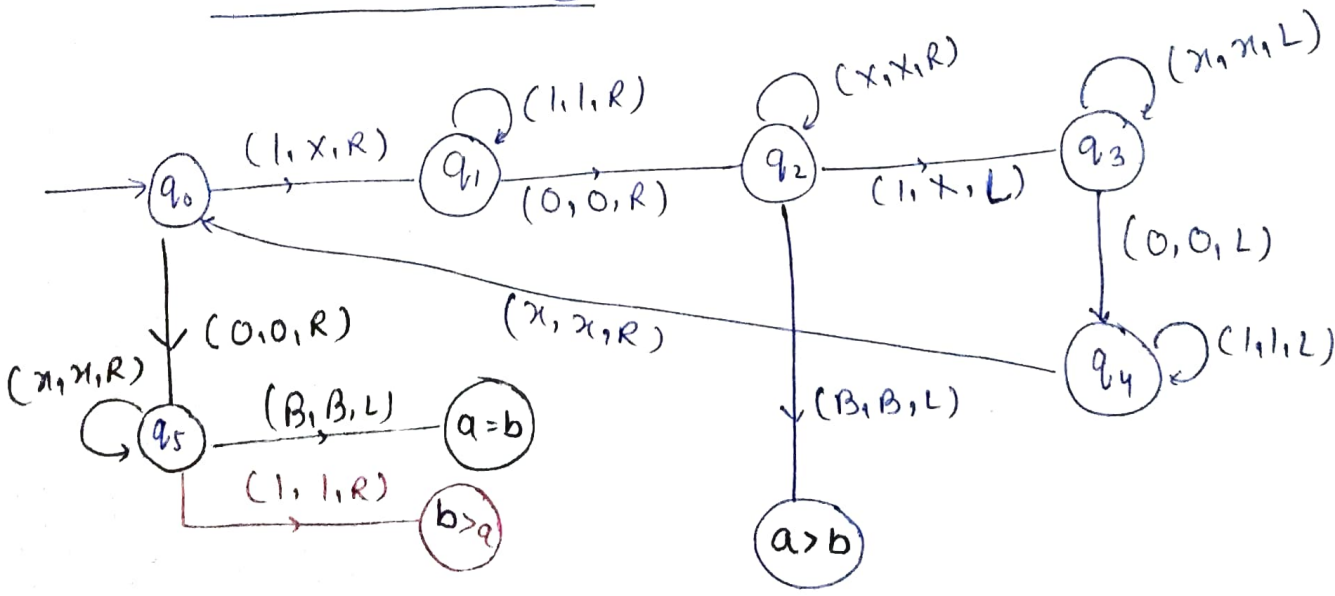
$0 \leftrightarrow 1$, $1 \leftrightarrow B$ — To maintain the count of 1's

TM as Comparator

a < b

* * * * 0 * * * * 1 $\rightarrow$

$\Rightarrow$ Mark each visited bit as X on both side of 0.



Note → Turing Machine can perform any mathematical operation because it is able to perform addition and comparison.

⟹ Even it can be used to any number system because unary number system can be extend to any number system.

Standard Turing Machine → (Partially Deterministic)

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

↓ Tape Alphabet ↓ Transition fn

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

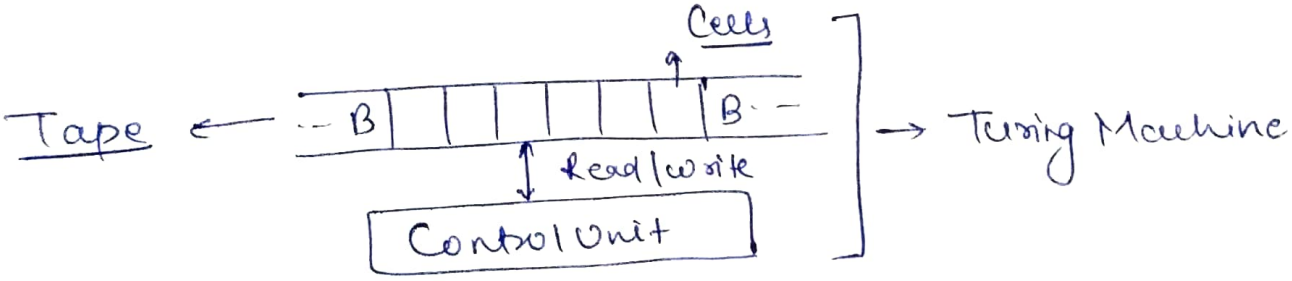
$\Gamma = \{a, b, x, y, B\}$

$\Sigma \subset \Gamma$ **

$B \in \Gamma$ → used to represent Blank (dependon textbook)

$q_0 \in Q$ → initial state

$F \subseteq Q$ → set of final states



Finite Automata + stack → PDA

PDA + stack → Turing Machine

Finite Automata + 2stack → Turing Machine

Using 2 stack we can perform the functionality of queue.

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

[* It is the standard δ but it can varied from turing machine.

Summary

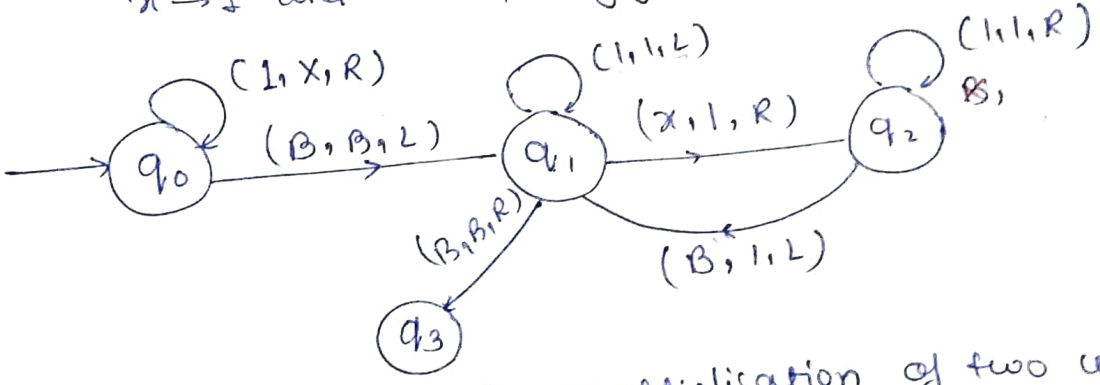
1) Tape is unbounded, so any number of left and right moves possible.
2) It is deterministic, i.e atmost one move for each configuration.
3) No special i/p or o/p files

TM as Copier

I/p BB 11 BB

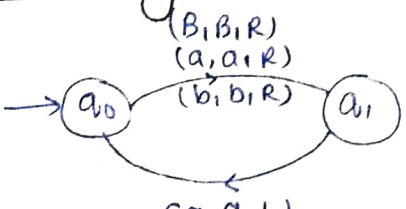
O/p BB 11 1 1 BB

Replace 1 → x and reach to B then convert each x → 1 and correspondingly replace each B → 1



Note → It can use in multiplication of two unary numbers.

Non-Halting TM → Turing Machine in Infinite loop



→ It is happening bcoz we can move in both direction. so it doesn't happens in FA and PDA.

→ If language doesn't support an input then machine may get into infinite loop.

→ All the turing machines might not halt to

Turing thesis : Given by Turing in 1930

(Not Imp for Gate)

Any computation that can be carried out by mechanical means can be performed by turing machine. It means turing machine can work as computer.

Some argument why turing thesis is accepted as definition of mechanical computation or computer:

a) Anything that can be done by computer can also be done by 'TM'.

b) No one has yet been able to suggest a problem, solvable by computer not by 'TM'

c) No computational or mechanical computation model is as poweful than the turing machine model.

Modified TM

Power of TM is defined as Languages accepted by the TM. It can be generalise to FA and PDA too.

No. of Language accepted by the modified TM is same as standard TM means we are not going to use any new languages

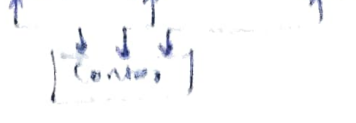
1) TM with stay option → $\delta : Q \times \Gamma \to \{L, R, S\}$

2) TM with semi-infinite tape → Blank on either 1 side.

3) Multitape TM

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

→ TM with multiple tape.



4) Offline TM → Give I/p in seperate file in read only mode and allow it to do modification on tape only.

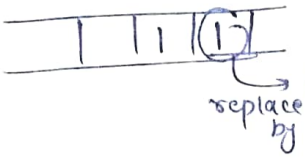
5) Jumping TM → No need to move only single step, You can also jump more than 1 step.

$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\} \times \{n\}$ → no. of steps

→ Time Complexity may get reduced using modified TM, but it doesn't contribute in powers

6) Non-erasing TM → You can't replace any symbol by Blank, but you can use any other symbol.

Ex

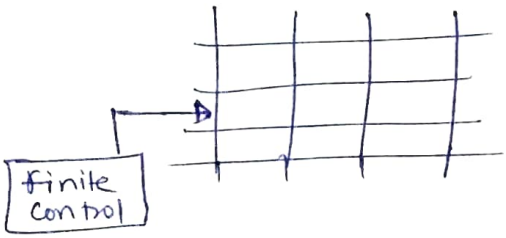


7) Always writing TM

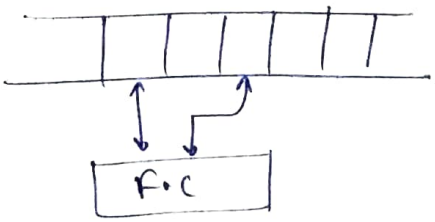
$\{a, a, L/R\}$ → Not allowed you have to always change the symbol as $\{a, A \text{ -- } z$ anything $, L/R\}$

8) Multi dimensional TM

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



9) Multihead TM



10) Automata with a queue is more useful.

11) TM with only 3 states. → Any TM can be minimized to TM with 3 states only.

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

13) Non-Deterministic TM

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

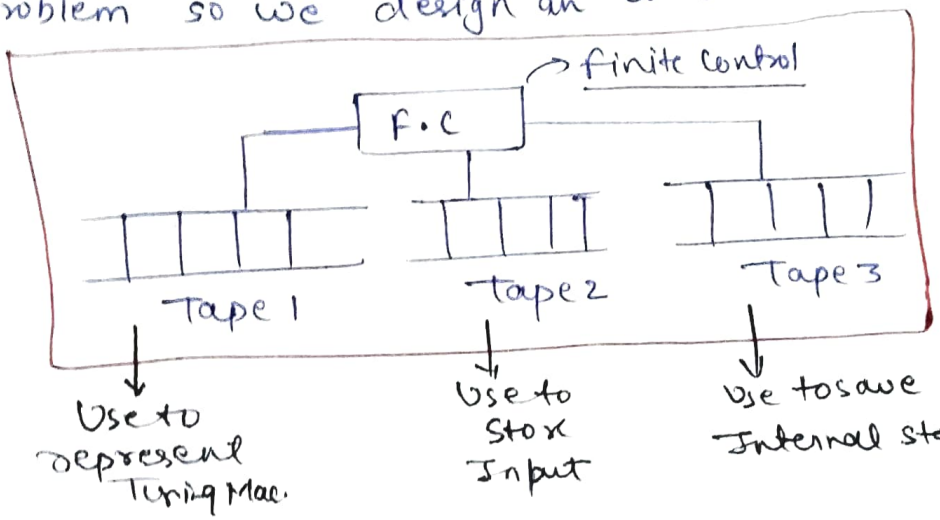
→ Deterministic and Non-Deterministic TM both are 'equivalent in power'

14) A NPDA with two independent stacks

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

Universal Turing Machine

A Turing machine can solve all languages but a single turing can be design to solve only a particular language but a single computer can solve multiple problem so we design an universal turing machines



$Q = (q_1, q_2, q_3, \ldots)$

$\Gamma = (a_1, a_2, a_3, \ldots)$

Encoding Used

$a_1 \rightarrow 1$	$a_1 \rightarrow 1$
$q_2 \rightarrow 11$	$a_2 \rightarrow 11$
$q_3 \rightarrow 111$	$a_3 \rightarrow 111$

Ex → $\delta : (q_1, a_1) = (q_2, a_2, R)$

1011011011

Complete transition represented as string of 0 and 1.

Save all these transitions Tape 1.

L → 1

R → 11

Tape-2 Use to do changes according to string stored in Tape 1.

Tape-3 → Store the state at which the Turing machine is currently working.

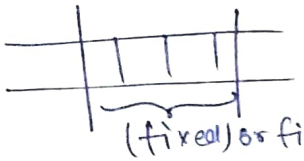
Imp → Every TM can be represent as string of 0 & 1 but every string of 0 & 1 is not a TM.

Linear- Bounded Automata

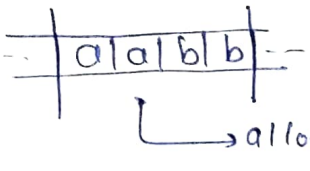
Limiting the tape

→ If NDTM is using the tape as stack called PDA.

→ If we fix the both ends of tape and provide only a fixed size m/m to TM called FA.



→ If a TM is designed in such a way that it can only access the part of tape that contains input, called "Linear bounded Automata."

Ex 

→ It can vary according to size of input.

→ It is much useful than TM, and powerful than PDA.

$$FA < PDA < LBA < TM$$

→ We are comparing by taking non-deterministic machines.

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

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

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

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

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

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

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

$L = \{wwwR : w \in \{a,b\}^+\}$

Note → Proof is not important all the given languages are accepted by LBA. so just mug up the examples.