Topic = Simplification Of CFG

Rule 1 Eliminate useless Symbol (Non terminal)
(Remove Useless Production)

Example :-

$S \rightarrow AB$
$A \rightarrow aA$
$B \rightarrow bB \mid \epsilon$

S & A useless Because A is Not terminating
So also S is also Not terminating

So it is NULL language.

Rule 2 Eliminate Null or Nullable Productions

Example :-

$S \rightarrow AB$
$A \rightarrow \epsilon \mid aA$
$B \rightarrow \epsilon \mid bB$

$A \rightarrow \epsilon$
$B \rightarrow \epsilon$
} Null Production

~~$S \rightarrow \epsilon$~~

$S \rightarrow AB$

Nullable Production :-
after some transition
It become Null
S is nullable Here.

Rule 3

Eliminate Unit Productions.

$S \rightarrow X$ ← Unit Production

$X \rightarrow a \mid \epsilon$

$S \rightarrow a \mid \epsilon$

Usefull Symbol :-

$S \overset{*}{\Rightarrow} XAY \overset{*}{\Rightarrow} w$

- $S \overset{*}{\Rightarrow} XAY$ → useful symbol
- w → Terminal String

generating :- we can say that 'X' is generating if after some transision X leads to terminating state.

Reachable :- we can say that 'X' is reacheable if only when we starting from starting non-terminal & after some transision we are getting X

Example (1)

$S \rightarrow AB \mid a$

$A \rightarrow b$

B is useless Because B is Not terminating.
or say Not generating those Production
so we have remove which containg B.

$S \rightarrow a$

$A \rightarrow b$

No A is Not reachable so also it is useless. so remove it

then our final Production

$S \rightarrow a$

Example = 2 :-

$S \rightarrow aS \mid A \mid C$

$A \rightarrow a$

$B \rightarrow aa$

$C \rightarrow aCb$

B is Not reacheable

C is Not terminating

remove it then our Production is

$S \rightarrow aS \mid A$

$A \rightarrow a$

A is unit Production remove it.

$S \rightarrow aS \mid a$

Example 3 :- Remove useless Productions from given cfg

$S \to aAa \mid aBC$
$A \to aS \mid bD$
$B \to aBa \mid b$
$C \to abb \mid DD$
$D \to aDa$
$E \to Fb$
$F \to c$

} G

E & f are Not Reacheble (But cenerative)

D is Not Cenerative (Not terminating)

$S \to aAa \mid aBc$
$A \to aS \mid bD$
$B \to aBa \mid b$
$C \to abb \mid DD$

}

Now Remove those Production which containing D

$S \to aAa \mid aBc$
$A \to aS$
$B \to aBa \mid b$
$C \to abb$

} G'

G & G' are equivalent

If $G = (V_{NT}, V_T, S, P)$ be a cfg & $L(G) \neq \phi$

then

$G' = (V'_{NT}, V_T, S, P')$

By removing useless Symbols / Productions will leads to a condition $L(G) = L(G')$

Removing Null Production :-

Example :-

$G =$ $S \to AB$
$A \to Aa \mid \in$
$B \to Bb \mid \in$

$A \to \in$ } Null Production (Remove Directly)
$B \to \in$ }

$S \to AB$ } Nullable Production

$S \to \in \mid A \mid B \mid AB$ ← AB — 00 → ∈, 01 → B, 10 → A, 11 → AB

↑ Remove this also

$A \to a \mid Aa \mid \in$ } Aa — 0/1 → a, Aa
$B \to b \mid Bb \mid \in$ }

So After removing all Nulls the final Productions are

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

NOTE :- Before Removing Null our language is containing '∈'.
So if language is containing ∈ then after Removing all the nulls we added one ∈ to the starting Non terminal in our Example 'S'

So final Production rules are

$S \rightarrow AB \mid \epsilon$
$A \rightarrow Aa \mid a$
$B \rightarrow Bb \mid b$

Example :- Remove Null from the given CFG.

$S \rightarrow AB$
$A \rightarrow aAA \mid \epsilon$
$B \rightarrow bBB \mid \epsilon$

Since $A \rightarrow \epsilon$, $B \rightarrow \epsilon$ Null Production

$S \xrightarrow{*} AB$ is Nullable.

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

Remove ϵ

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

But In our language ϵ exist so to add ϵ at Starting NT.

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

Example :- Remove null (ϵ) from given CFG

$S \rightarrow ABaC$

$A \rightarrow BC$

$B \rightarrow b \mid \epsilon$

$C \rightarrow D \mid \epsilon$

$D \rightarrow d$

$B \rightarrow \epsilon$
$C \rightarrow \epsilon$
} Null Productions

$A \Rightarrow BC \rightarrow$ Nullable.

$S \rightarrow ABaC \mid ABa \mid AaC \mid Aa \mid BaC \mid Ba \mid aC \mid a$

$A \rightarrow B \mid C \mid BC$

$B \rightarrow b$

$C \rightarrow D$

$D \rightarrow d$

111
110
101
100
011
010
001
000

Because In our Language No ϵ exist so No Need to ~~no~~ add it.

Elimination of Unit Production :-

Unit Production :- Only Single Non terminal is coming to the Right.

Example :- $A \rightarrow B$

Example :-

$S \rightarrow Aa \mid B$

$B \rightarrow A \mid bb$

$A \rightarrow a \mid bc \mid B$

The cfg without unit Production can be

$S \rightarrow Aa$

$B \rightarrow bb$

$A \rightarrow a \mid bc$

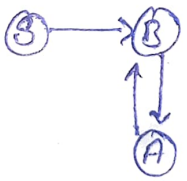
unit productions are

we removed this from cfg

$S \rightarrow B$ — (1)

$B \rightarrow A$ — (2)

$A \rightarrow B$ — (3)



Now we have to add some new production

$A \rightarrow bb$ (A depends on B from (3))

$B \rightarrow a \mid bc$ (B depends on A from (2))

$S \rightarrow ~~bb~~$ $a \mid bc \mid bb$ (S Depends on B & transitively Depends on A)

Final Production Rules

$S \rightarrow Aa \mid a \mid bC \mid bb$
$B \rightarrow bb \mid a \mid bC$
$A \rightarrow a \mid bC \mid bb$

Example 2 :-

$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
$F \rightarrow I \mid (E)$
$T \rightarrow F \mid T*F$
$E \rightarrow T \mid E+T$

Terminals $\rightarrow \{a, b, 0, 1, (,), *, +\}$

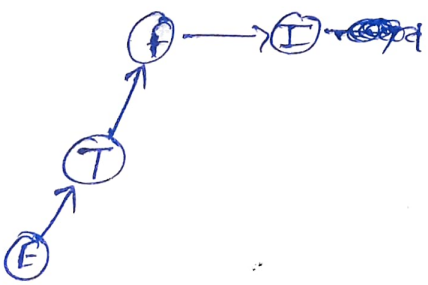
Non-Terminals $\rightarrow \{I, F, T, E\}$

CFG without unit production is

$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
$F \rightarrow (E)$
$T \rightarrow T*F$
$E \rightarrow E+T$

Unit production

$F \rightarrow I$
$T \rightarrow F$
$E \rightarrow T$



add new Production

$F \rightarrow a|b|Ia|Ib|Io|I1$

(F depends on I)

$T \rightarrow a|b|Ia|Ib|Io|I1|(E)$

(T depends on F & transitively depends on I)

$E \rightarrow a|b|Ia|Ib|Io|I1|(E)|T*F$

(E depends on T & transitively depends on F & I

So final Productions are

$E \rightarrow E+T|a|b|Ia|Ib|Io|I1|(E)|T*F$

$T \rightarrow a|b|Ia|Ib|Io|I1|(E)|T*F$

$F \rightarrow (E)|a|b|Ia|Ib|Io|I1$

$I \rightarrow a|b|Ia|Ib|Io|I1$

Chomsky Normal Form (CNF)

Convert the following grammer to CNF

$$S \rightarrow aSa \mid bSb \mid a \mid b \mid aa \mid bb$$

~~$S \rightarrow a$~~

$S \rightarrow a$
$S \rightarrow b$
} are already in CNF

$S \rightarrow aSa \mid bSb \mid aa \mid bb$ are not in CNF

NOTE :- we need Production of form
$S \rightarrow AB$
or
$S \rightarrow a$

Step1 Arrange the bodies of length 2 & more, consist only of variable (non Terminals)

$$S \rightarrow ASA \mid BSB \mid a \mid b \mid AA \mid BB$$

~~Step2~~ :- Now $S \rightarrow AA$ & $S \rightarrow BB$ are in CNF

Step2 :- Break the Bodies of length 3 & more into length 2

$$S \rightarrow AX \mid BY \mid a \mid b \mid AA \mid BB$$
$$A \rightarrow a$$
$$B \rightarrow b$$
$$X \rightarrow SA$$
$$Y \rightarrow SB$$

Now It is in CNF.

Convert the following CFG to CNF

$S \rightarrow aaaaS \mid aaaa$

Step1 :-

$S \rightarrow AAAAS \mid AAAA$

$A \rightarrow a$

Step2 :-

(1)
$S \rightarrow AA_1 \mid AAAA$
$A \rightarrow a$
$A_1 \rightarrow AAAS$

(2)
$S \rightarrow AA_1 \mid AAAA$
$A \rightarrow a$
$A_1 \rightarrow AA_2$
$A_2 \rightarrow AAS$

(3) $\Rightarrow$
$S \rightarrow AA_1 \mid AAAA$
$A \rightarrow a$
$A_1 \rightarrow AA_2$
$A_2 \rightarrow AA_3$
$A_3 \rightarrow AS$

(4)
$S \rightarrow AA_1 \mid AK_1$
$A \rightarrow a$
$A_1 \rightarrow AA_2$
$A_2 \rightarrow AA_3$
$A_3 \rightarrow AS$
$K_1 \rightarrow AAA$

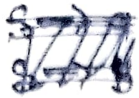
$\Rightarrow$

(5)
$S \rightarrow AA_1 \mid AK_1$
$A \rightarrow a$
$A_1 \rightarrow AA_2$
$A_2 \rightarrow AA_3$
$A_3 \rightarrow AS$
$K_1 \rightarrow AK_2$
$K_2 \rightarrow AA$

Now It is a CNF

Example :- Convert the following CFG

$S \rightarrow aSa \mid bSb \mid a \mid b \mid aa \mid bb$



Solve

$S \rightarrow a$
$S \rightarrow b$
} already in CNF

Step1

$S \rightarrow ASA \mid BSB \mid a \mid b \mid AA \mid BB$
$A \rightarrow a$
$B \rightarrow b$

} Arrange the bodies of length 2 & more, Consist only of variables (Non Terminals)

Now

$S \rightarrow AA$
$S \rightarrow BB$
} Already in CNF

$S \rightarrow AX \mid BY \mid a \mid b \mid AA \mid BB$
$A \rightarrow a$
$B \rightarrow b$
$X \rightarrow SA$
$Y \rightarrow SB$

} Break the Bodies of length 3 or more Into 2

Example :- $S \rightarrow ASA \mid aB$, $A \rightarrow B \mid S$, $B \rightarrow b \mid \epsilon$

(Convert this CFG to CNF)

Step1 :-

$S' \rightarrow S$ ← Introduce New production or New Starting Symbol
$S \rightarrow ASA \mid aB$
$A \rightarrow B \mid S$
$B \rightarrow b \mid \epsilon$

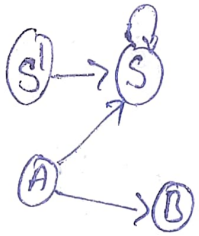
Step2 :- Remove Null Production

$B \to \epsilon$ Null Production
$A \to B$ is Nullable.

$S' \to S$
$S \to S \mid SA \mid AS \mid a \mid aB \mid ASA$
$A \to B \mid S$
$B \to b$

Step3 :- Remove Unit production

$S' \to S$
$S \to S$
$A \to B$ are unit production
$A \to S$



Production without unit production are.

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

$B \to b$

New Production

$S' \to SA \mid AS \mid a \mid aB \mid ASA$
$S \to SA \mid AS \mid a \mid aB \mid ASA$
$A \to b \mid SA \mid AS \mid a \mid aB \mid ASA$

final Production

$S' \to SA \mid AS \mid a \mid aB \mid ASA$
$S \to SA \mid AS \mid a \mid aB \mid ASA$
$A \to b \mid SA \mid AS \mid a \mid aB \mid ASA$
$B \to b$

Step4 :- Remove useless symbol

There is No useless symbol

Step5 :- convert the bodies of length 2 or more consist only variables (Non Terminal)

$S^1 \rightarrow SA \mid AS \mid a$
$S \rightarrow SA \mid AS \mid a$
$A \rightarrow b \mid SA \mid AS \mid a$
$B \rightarrow b$

are already in CNF

we have to convert

$S^1 \rightarrow aB \mid ASA$
$S \rightarrow aB \mid ASA$
$A \rightarrow aB \mid ASA$

Now after converting 9 Productions are

$S^1 \rightarrow SA \mid AS \mid a \mid XB \mid ASA$
$S \rightarrow SA \mid AS \mid a \mid XB \mid ASA$
$A \rightarrow b \mid SA \mid AS \mid a \mid XB \mid ASA$
$B \rightarrow b$
$X \rightarrow a$

Steps :- Break the Bodies of length 3 or more

~~S → A~~

$S' \rightarrow SA \mid AS \mid a \mid XB \mid AK$
$S \rightarrow SA \mid AS \mid a \mid XB \mid AK$
$A \rightarrow b \mid SA \mid AS \mid a \mid XB \mid AK$
$B \rightarrow b$
$X \rightarrow a$
$K \rightarrow SA$

Ans

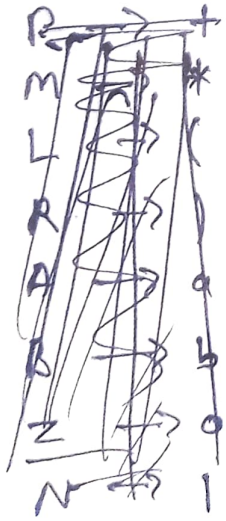
Example :- Convert given CFG to CNF

$E \rightarrow E+T \mid T*F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid Io \mid I$

$T \rightarrow T*F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid Io \mid I$

$F \rightarrow (E) \mid a \mid b \mid Ia \mid Ib \mid Io \mid I$

$I \rightarrow a \mid b \mid Ia \mid Ib \mid Io \mid I$



E → EPT | TMF | LER | a | b | IA | IB | IZ | IN

T → TMF | LER | a | b | IA | IB | IZ | IN

F → LEF | a | b | IA | IB | IZ | IN

I → a | b | IA | IB | IZ | IN

P → +

M → *

L → (

R →)

A → a

B → b

Z → 0

N → 1

Now we have to convert

E → EPT , E → TMF , E → LER

T → TMF , T → LER

F → LER

In Bodies of 2 so the final CFG is.

E → EP1 | TM1 | LE1 | a | b | IA | IB | IZ | IN

T → TM1 | LE1 | a | b | IA | IB | IZ | IN

F → LE1 | a | b | IA | IB | IZ | IN

I → a | b | IA | IB | IZ | IN

P1 → PT

M1 → MF

E1 → ER

P → +

M → *

L → (

R →)

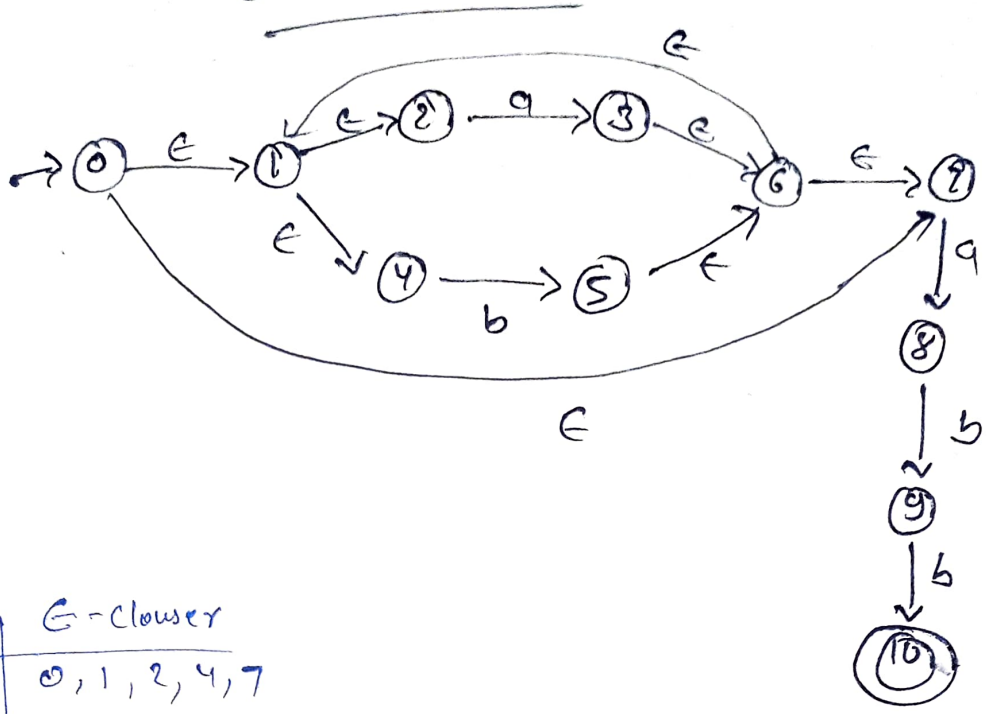
A → a

B → b

Z → 0

N → 1

∈-NFA to DFA



Solve

	∈-closer
0	0, 1, 2, 4, 7
1	1, 2, 4
2	2
3	3, 6, 1, 2, 4, 7
4	4
5	5, 6, 1, 2, 4, 7
6	6, 1, 2, 4, 7
7	7
8	8
9	9
10	10

Initial state :-

∈ closer(0) = [0, 1, 2, 4, 7] = A

(1) ∈-closer(move(A, a)) = ∈-closer([3, 8]) = [3, 6, 1, 2, 4, 7, 8] = B

∈-closer(move(A, b)) = ∈-closer([5]) = [5, 6, 1, 2, 4, 7] = C

(2) ∈-closer(move(B, a)) = ∈-closer([3, 8]) = [3, 6, 1, 2, 4, 7, 8] = B

∈-closer(move(B, b)) = ∈-closer([5, 9]) = [5, 6, 1, 2, 4, 7, 9] = D

(3) ∈-closer(move(C, a)) = ∈-closer([3, 8]) = [3, 6, 1, 2, 4, 7, 8] = B

∈-closer(move(C, b)) = ∈-closer([5]) = [5, 6, 1, 2, 4, 7] = C

④ ϵ-closure(~~D,a~~ move(D,a)) = ϵ-closure([3,8]) = [3,6,1,2,4,7,8] = B

ϵ-closure(move(D,b)) = ϵ-closure([5,10]) = [5,6,1,2,4,7,10] = E

⑤ ϵ-closure(move(E,a)) = ϵ-closure([3,8]) = B

ϵ-closure(move(E,b)) = ϵ-closure([5]) = [5,6,1,2,4,7] = C

	a	b
→ A	B	C
B	B	D
C	B	C
D	B	E
(E)	B	C

CYK Algorithm

Ex:- Find wheather the string "baaba" exist in CFG

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CB \mid b$

$C \rightarrow AB \mid a$

5	X_{15}				
4	X_{14}	X_{25}			
3	X_{13}	X_{24}	X_{35}		
2	X_{12}	X_{23}	X_{34}	X_{45}	
1	X_{11}	X_{22}	X_{33}	X_{44}	X_{55}
	b	a	a	b	a

It is Just Like DP Store the smaller Subproblem & use it later.

$r_{11} = b = \{B\}$

$r_{22} = a = \{A, C\}$

$r_{33} = a = \{A, C\}$

$r_{44} = b = \{B\}$

$r_{55} = a = \{A, C\}$

5	$\{S, A, C\}$			
4	$\{B\}$	$\{S, C, A\}$		
3	$\{\phi\}$	$\{B\}$	$\{B\}$	
2	$\{A, S\}$	$\{B\}$	$\{S, C\}$	$\{A, S\}$
1	$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{\phi\}$ $\{A, C\}$
	b	a	a	b a

$r_{12} = r_{11} \cdot r_{22}$

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

$= \{BA, BC\}$

$= \{A, S\}$

$r_{23} = r_{22} \cdot r_{33}$

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

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

$= \phi, \phi, \phi, B$

$= \{B\}$

$r_{34} = r_{33} \cdot r_{44}$

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

$= \{AB, CB\}$

$= \{S, C\}$

$r_{45} = r_{44} \cdot r_{55}$

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

$= \{BA, BC\}$

$= \{A, S\}$

$r_{13} = r_{12} \cdot r_{23}$ (blaa)

$= \{B\} \cdot \{B\}$

$= \{BB\}$

$= \phi$

$= r_{12} \cdot r_{33}$ (bala)

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

$= \{AA, AC, SA, SC\}$

$= \{\phi, \phi, \phi, \phi\}$

$= \{\phi\}$

$r_{19} = \phi$

r_{24} (baab)

$:= r_{22} \cdot r_{34}$ (a | ab)

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

$= \{AS, AC, CS, CC\}$

$= \{\phi, \phi, \phi, B\}$

$= \{B\}$

$:- r_{23}\ r_{44}$ (a.a | b)

$= \{B\} \cdot \{B\}$

$= \{BB\}$

$= \phi$

$r_{24} = \{B\}$

r_{35} (aba)

$:- r_{33} \cdot r_{45}$ (a | ba)

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

$= \{AA, AS, CA, CS\}$

$= \{\phi, \phi, \phi, \phi\}$

$= \phi$

$:- r_{34} \cdot r_{55}$ (ab | a)

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

$= \{SA, SC, CA, CC\}$

$= \{\phi, \phi, \phi, B\}$

$= \{B\}$

$r_{35} = \{B\}$

r_{14} (baab)

$:= (b | aab)$

$= r_{11} \cdot r_{24}$

$= \{B\} \cdot \{B\}$

$= \{BB\}$

$= \phi$

$:- (ba | bb)$

$= r_{12} \cdot r_{34}$

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

$= \{AS, AC, SS, SC\}$

$= \{\phi, \phi, \phi, \phi\}$

$= \{\phi\}$

$:- (bab | b)$

$= r_{13} \cdot r_{44}$

$= \{\phi\} \cdot \{B\}$

$= \{\phi\}$

$\{B\}$

$= \{\phi\}$

$r_{14} = \{\phi\}$

$r_{25} = (aaba)$

$\therefore (a|aba)$

$= r_{22} \cdot r_{35}$

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

$= \{AB, CB\}$

$= \{S, C\}$

$\therefore (aa|ba)$

$= r_{23} \cdot r_{45}$

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

$= \{BA, BS\}$

$= \{A, \phi\}$

$= \{A\}$

$\therefore (aab|a)$

$= r_{24} \cdot r_{55}$

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

$= \{BA, BC\}$

$= \{A, S\}$

$r_{25} = \{S, C, A\}$

$r_{15} = baaba$

$\therefore b|aaba$

$= r_{11} \cdot r_{25}$

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

$= \{BS, BC, BA\}$

$= \{\phi, S, A\}$

$= \{S, A\}$

$\therefore ba|aba$

$= r_{12} \cdot r_{35}$

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

$= \{AB, SB\}$

$= \{S, C\}$

$\therefore baa|ba$

$= r_{13} \cdot r_{45}$

$= \{\phi\} \cdot \{A, S\}$ $\{A, S\}$

$= \{BA, BS\}$ $\{\phi\}$

$= \{A\}$

$\therefore baab|a$

$= r_{14} r_{55}$

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

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

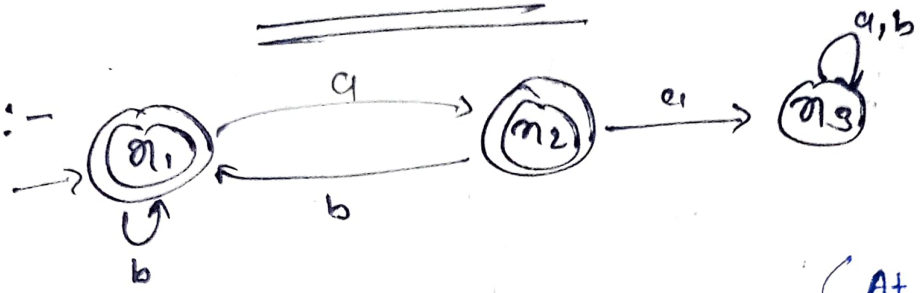
$= \{A, S\}$ $\{\phi\}$

$r_{15} = \{S, A, C\}$

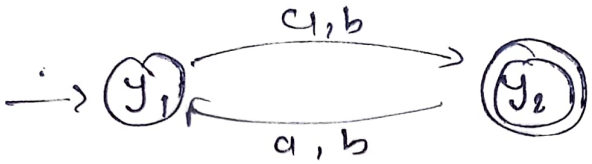
Because S is coming means starting state coming on the last so string is accepted.

$fA1 . fA_2$

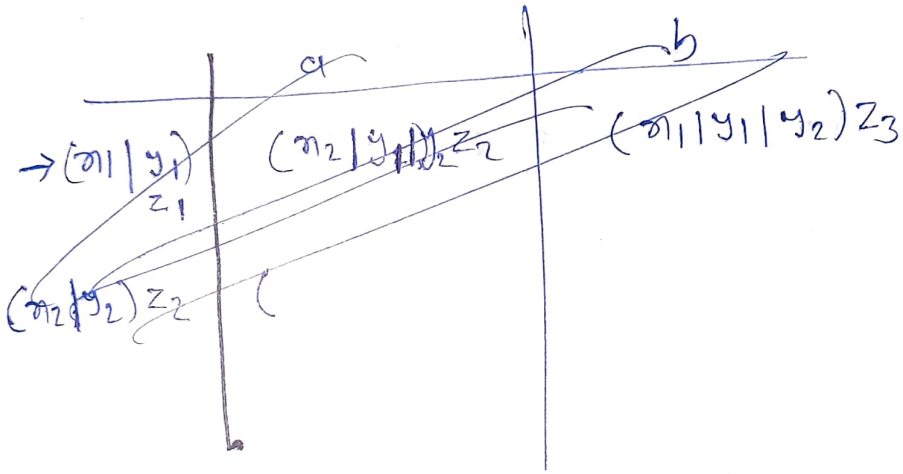
$fA1$:-



fA_2 :-



At final state of fA_1 we are able to jump Initial state of fA_2.



Because the initial state of fA_1 is the final state

So our Initial state for $fA_1 . fA_2$ is $q_1 | y_1$

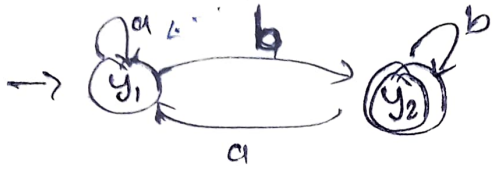
	a	b
$\rightarrow (q_1 \mid y_1)\ z_1$	$(q_2 \mid y_1 \mid y_2)\ z_2$	$(q_1 \mid y_1 \mid y_2)\ z_3$
$(q_2 \mid y_1 \mid y_2)\ (z_2)$	$(q_3 \mid y_2 \mid y_1)\ z_4$	$(q_1 \mid y_1 \mid y_2)\ z_3$
$(q_1 \mid y_1 \mid y_2)\ (z_3)$	$(q_2 \mid y_1 \mid y_2)\ z_2$	$(q_1 \mid y_1 \mid y_2)\ z_3$
$(q_3 \mid y_2 \mid y_1)\ (z_4)$	$(q_3 \mid y_1 \mid y_2)\ z_4$	$(q_3 \mid y_1 \mid y_2)\ z_4$

If it is Not final then our Initial state is q_1

Ex. 2

FA1 =

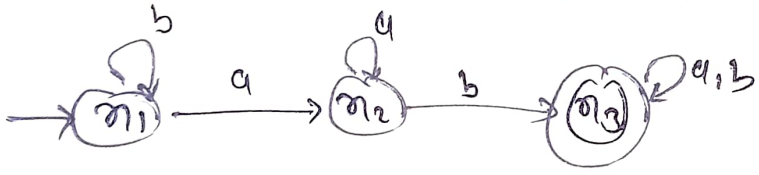
FA2 :



	a	b
→ $(n_1)\ z_1$	$(n_2)\ z_2$	$(n_1)\ z_1$
$(n_2)\ z_2$	$(n_3 \mid y_1)\ z_3$	$(n_1)\ z_1$
$(n_3 \mid y_1)\ z_3$	$(n_3 \mid y_1)\ z_3$	$(n_3 \mid y_1 \mid y_2)\ z_4$
$(n_3 \mid y_1 \mid y_2)$ z_4	$(n_3 \mid y_1)\ z_3$	$(n_3 \mid y_1 \mid y_2)\ z_4$

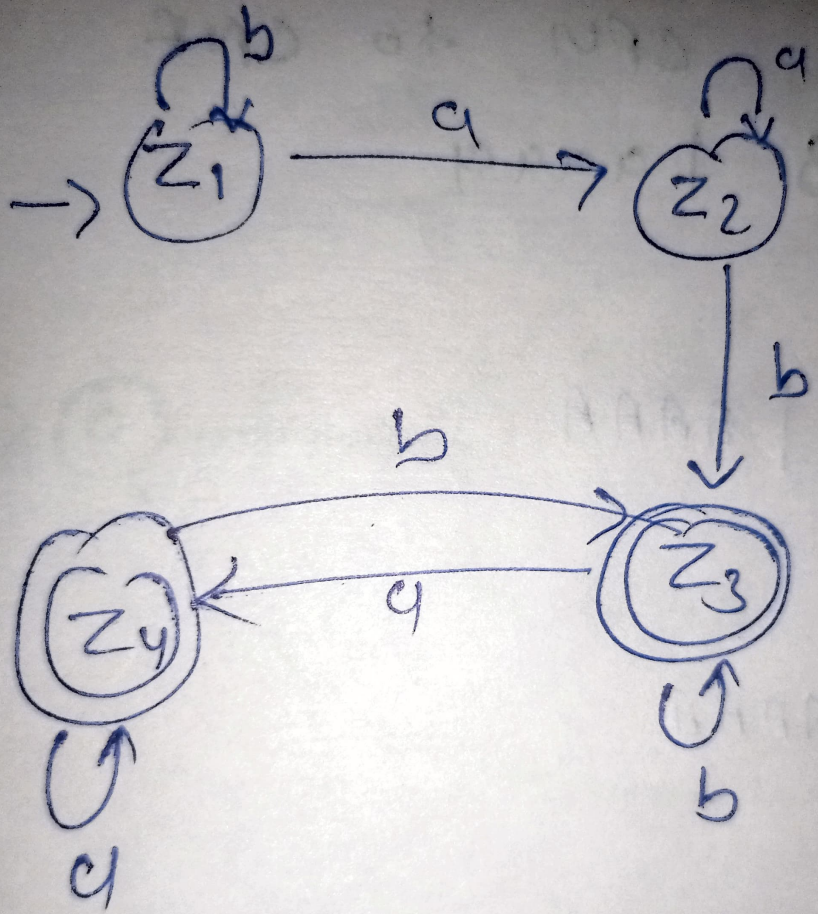
---

find FA*



In Transision
At final state we have a chance to Jump to the initial state

	a	b
→ $(n_1)\ z_1$	$(n_2)\ z_2$	$(n_1)\ z_1$
$(n_2)\ z_2$	$(n_2)\ z_2$	$(n_3 \mid n_1)\ z_3$
$(n_3 \mid n_1)$ z_3	$(n_3 \mid n_1 \mid n_2) z_4$	$(n_3 \mid n_1)\ z_3$
$(n_3 \mid n_1 \mid n_2)$ z_4	$(n_3 \mid n_1 \mid n_2)$ z_4	$(n_3 \mid n_1)\ z_3$


b
a
a
z_1
z_2
b
b
a
z_4
z_3
a
b

PDA

PDA is Made up of 7 tuples.

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

where

Q = finite set of states

Σ = Input symbol

Γ (gamma) :- Stake alphabate (top of stake)

δ = Transision function

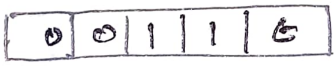
q_0 = Initial state

Z_0 = Stake start symbol

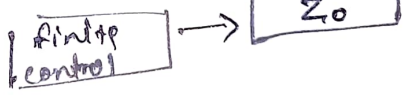
F = Set of final states.

Example :- $\{0^n 1^n : n>0\}$

input string



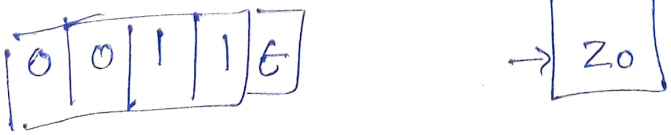
Read head

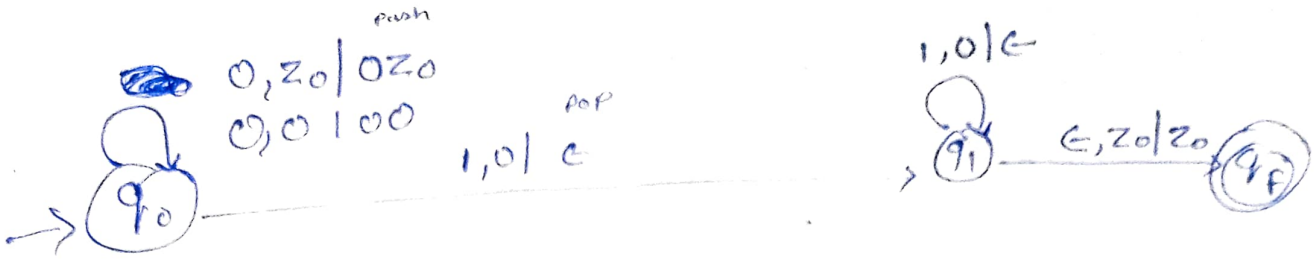


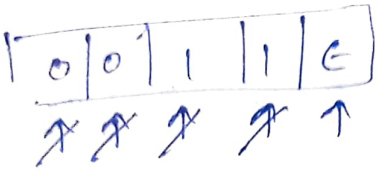
finite control :- Based on the symbol it check whether a stake is srink or grow.

Keep push '0' onto the stake then Pop Symbols from the stake for each '1' being read.

$0^n 1^n$, $n \geq 1$







∈ = POP

Z0 = No operation else push

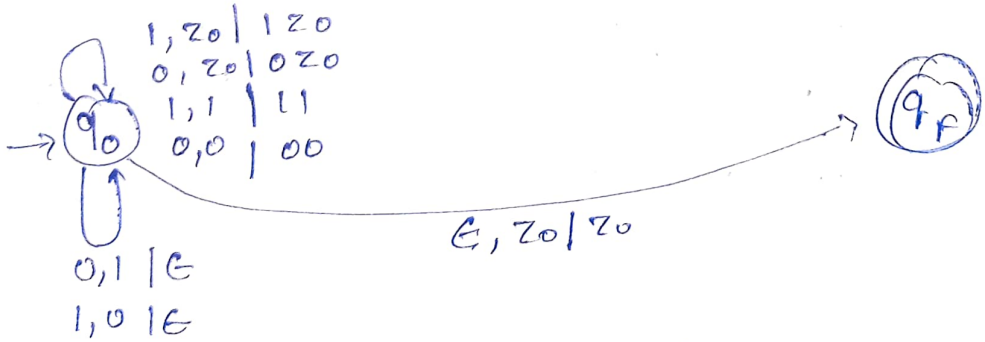
Example :- Construct a PDA for Language L where

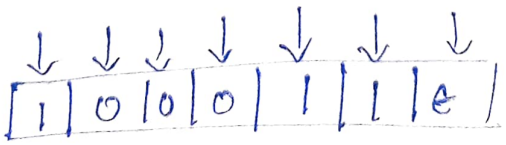
$$L = \{ w \in \{0,1\}^* \mid n_0(w) = n_1(w) \}$$

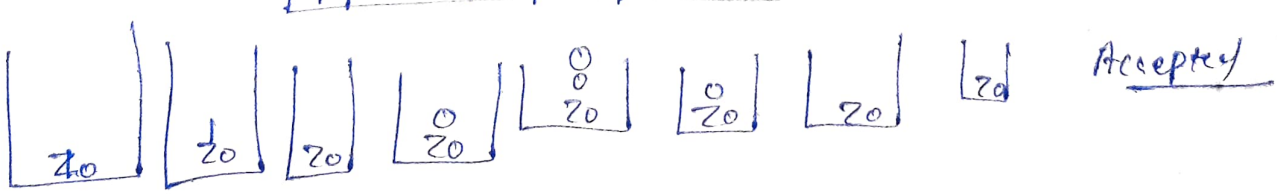
↓↓

Number of 0's equals to Number of 1's

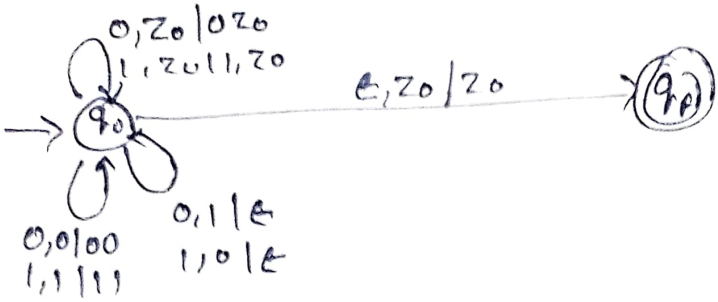
$$L = \{ 10,\ 01,\ 0011,\ 1100,\ 0101,\ 1010,\ 1001,\ 0110 \}$$







Configuration of Machine :-



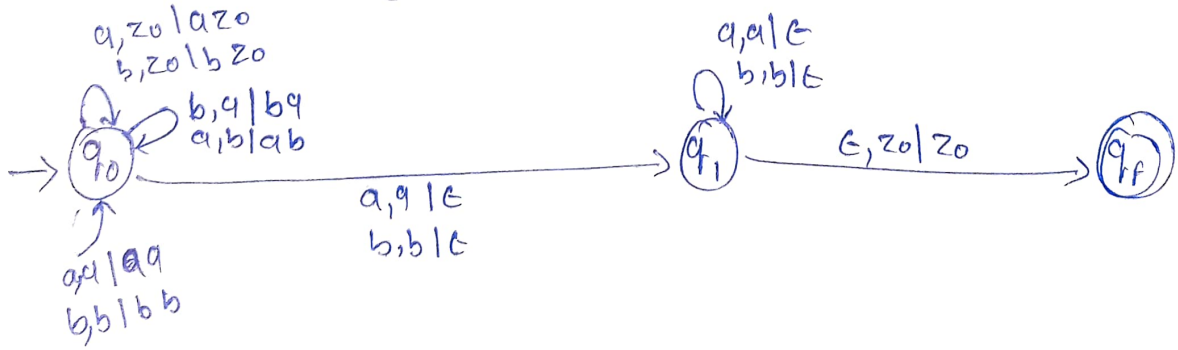
$$\delta(q_0, 0, Z_0) = (q_0, 0Z_0)$$
$$\delta(q_0, 1, Z_0) = (q_0, 1Z_0)$$
$$\delta(q_0, 1, 0) = (q_0, \epsilon)$$
$$\delta(q_0, 0, 1) = (q_0, \epsilon)$$
$$\delta(q_0, 0, 0) = (q_0, 00)$$
$$\delta(q_0, 11) = (q_0, 11)$$
$$\delta(q_0, \epsilon, Z_0) = (q_f, Z_0)$$

Example :- Construct PDA for the language

$$L = \{ w w^R \mid w \in \{a,b\}^* \}$$ where w^R is Reverse of w

↓

even palindrom

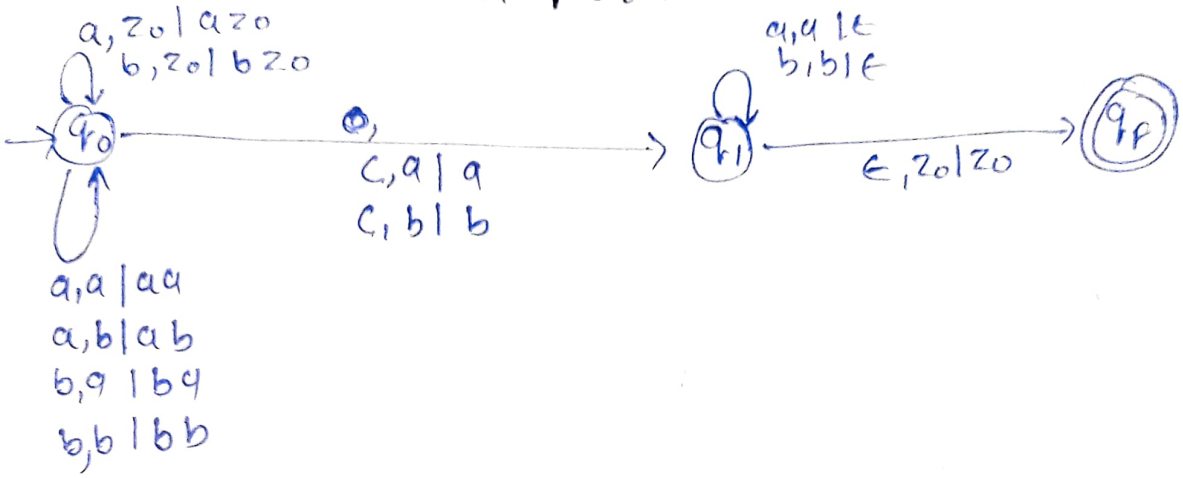


Construct PDA for the language

$L = \{ wcw^R \mid w \in \{a,b\}^* \}$ where w^R is Reverse of w

↓

odd pallindrom



In Prevous question (even pallindrom) working of the machine like this.

Ex :- $w = abba$

$(q_0, \underline{a}bba, z_0)$

$(q_0, a\underline{b}ba, a)$

$(q_0, ab\underline{b}a, b)$

$\in$

$(q_0, abb\underline{a}, b)$

$(q_0, abba\underline{\in}, a)$

✗

$(q_0, abb\underline{a}, a)$

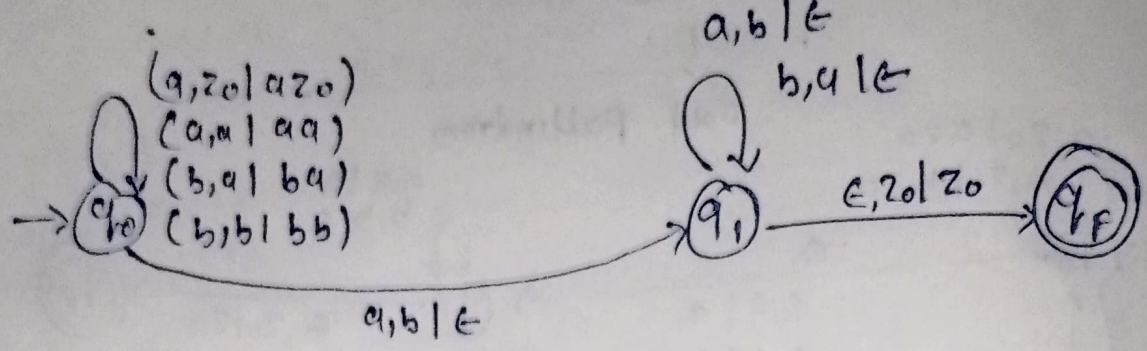
$(q_1, abba\underline{\in}, z_0)$

Accepted

$(q_f, abba\in, z_0)$

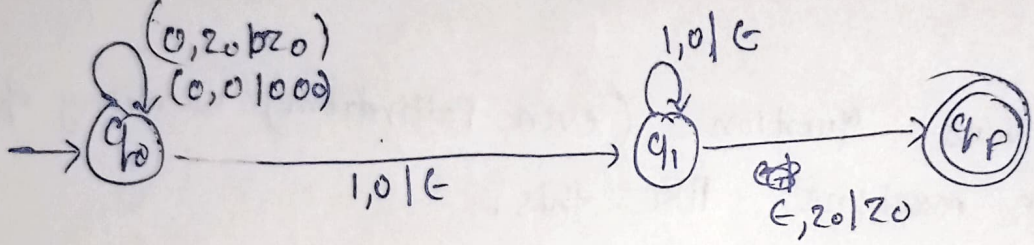
Construct a PDA for

Language: $\{a^n b^m a^m b^n$, where m & n are Independent & $m,n \geq 1\}$



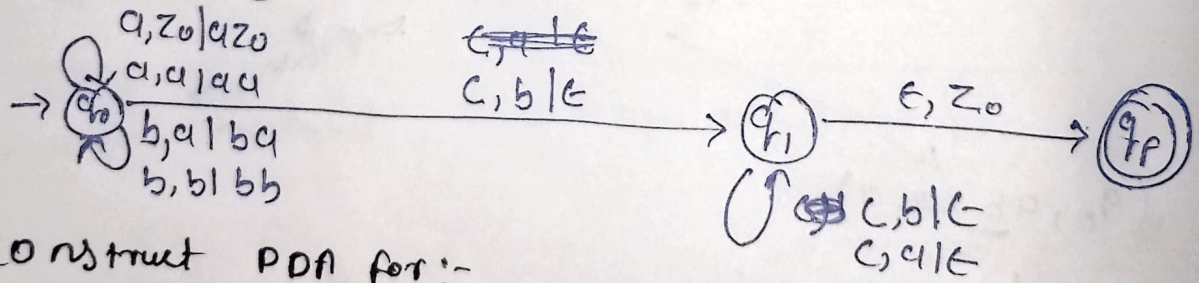
Construct a PDA for

$L = \{0^n 1^{2n}, \quad n \geq 1\}$



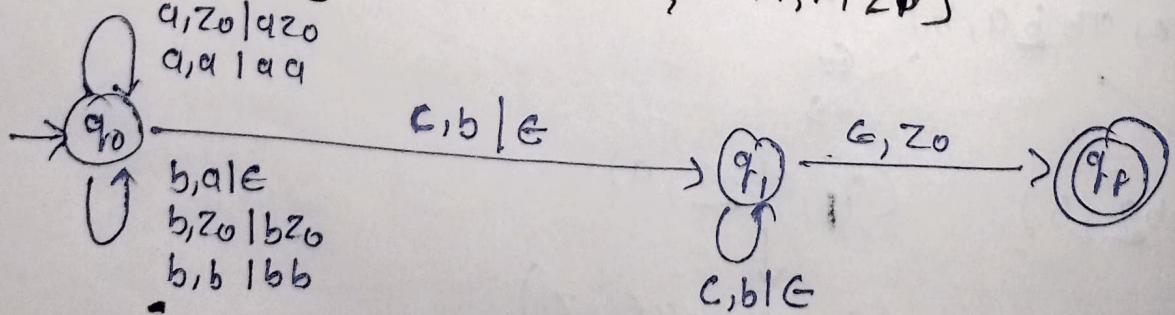
Construct PDA for :-

$\{a^n b^m c^{n+m}, \quad n,m \geq 1\}$



Construct PDA for :-

$\{a^n b^{n+m} c^m, \quad n,m \geq 1\}$



Turing Machine

Turing Machine is advancement over FA & PDA made By Alan turing (1912-1954). Turing machine consist of

→ finite control
→ tape
→ A read/write head that can be used for reading & writing on the tape.

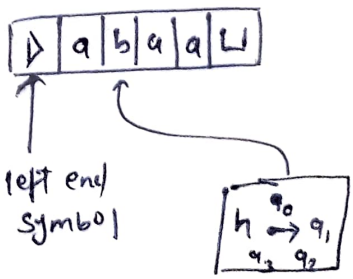
The control unit perform 2 function that depends upon its current state & tape symbol currently scanned by read/write head.

1st function :- Put the control unit on the new state.

2nd function :- either,

(a) Write the symbol on the tape square currently being scanned replacing the one that already have

(b) Move the read/write head on the tape square to left or right.



→ Tape had a left end but it extends indefinetly to the right, This could be Possible by introducing left end symbol on to the tape, such that when R/W head reads a left end symbol, it imidiatly moves to the right.

we use distinct symbol (←, →) to denote the movement of the head to the left or Right.

This two symbol are Not member of alphabet set.

A turing Machine (TM) is a quantuple having.

$(K, \Sigma, \delta, s, H)$

K = Non Empty finite set of state

Σ = is a alphabet set, containing a blank symbol ($\sqcup$) and left end symbol ($\triangleright$) but Not containing $\leftarrow \& \rightarrow$ respectively.

$s \in K$ (Initial state)

$H \subseteq K$ Is a set of halting states.

δ is a transision function

$$(K-H) \times \Sigma \longrightarrow K \times (\Sigma \cup (\leftarrow, \rightarrow))$$

Examples

Consider the TM given below

$M = (K, \Sigma, \delta, s, \{h\})$

$K = \{q_0, q_1, h\}$

$\Sigma = \{a, \sqcup, \triangleright\}$

$s = q_0$

& δ is given as

q	Σ	$\delta(q, \Sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_0	$\triangleright$	$(q_0, \rightarrow)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$
q_1	$\triangleright$	$(q_1, \rightarrow)$

Check wheather the string "aaa" is accepted or Not

Sol Initial configuration : $(q_0, \triangleright aaa\sqcup) \vdash$

~~$(q_0, \triangleright aaa\sqcup) \vdash (\triangleright \sqcup aa\sqcup)$~~

~~$(q_1, \triangleright \sqcup aa$~~

$(q_0, \triangleright aaa\sqcup) \vdash (q_1, \triangleright \sqcup aa\sqcup)$

$(q_1, \triangleright \sqcup aa\sqcup) \vdash (q_0, \triangleright \sqcup a a\sqcup)$

$(q_0, \triangleright \sqcup a a\sqcup) \vdash (q_1, \triangleright \sqcup \sqcup a\sqcup)$

$(q_1, \triangleright \sqcup \sqcup a\sqcup) \vdash (q_0, \triangleright \sqcup \sqcup a\sqcup)$

$(q_0, \triangleright \sqcup \sqcup a\sqcup) \vdash (q_1, \triangleright \sqcup \sqcup \sqcup\sqcup)$

$(q_1, \triangleright \sqcup \sqcup \sqcup\sqcup) \vdash (q_0, \triangleright \sqcup \sqcup \sqcup\sqcup)$

$(q_0, \triangleright \sqcup \sqcup \sqcup\sqcup) \vdash (h, \triangleright \sqcup \sqcup \sqcup\sqcup)$

we are at halting state so string is accepted.

2## Consider the TM given

$M = (K, \Sigma, \delta, s, \{h\})$

$K = \{q_0, q_1, h\}$

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

$s = q_0$

Check whether the string "abba" accepted or Not

q	σ	$\delta(q, \sigma)$
q_0	a	(q_1, b)
q_0	b	(q_1, a)
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	$(q_0, \rightarrow)$
q_1	b	$(q_0, \rightarrow)$
q_1	$\sqcup$	$(q_0, \rightarrow)$

Initial configuration of Machine $(q_0, \triangleright \underline{a}bba\sqcup)$

$(q_0, \triangleright \underline{a}bba\sqcup) \vdash (q_1, \triangleright \underline{b}bba\sqcup)$

$(q_1, \triangleright \underline{b}bba\sqcup) \vdash (q_0, \triangleright b\underline{b}ba\sqcup)$

$(q_0, \triangleright b\underline{b}ba\sqcup) \vdash (q_1, \triangleright ba\underline{b}a\sqcup)$

$(q_1, \triangleright ba\underline{b}a\sqcup) \vdash (q_0, \triangleright baba\sqcup)$

$(q_0, \triangleright baba\sqcup) \vdash (q_1, \triangleright baaa\sqcup)$

$(q_1, \triangleright baaa\sqcup) \vdash (q_0, \triangleright baaa\sqcup)$

$(q_0, \triangleright baaa\sqcup) \vdash (q_1, \triangleright baab\sqcup)$

$(q_1, \triangleright baab\sqcup) \vdash (q_0, \triangleright baab\sqcup)$

$(q_0, \triangleright baab\sqcup) \vdash (h, \triangleright baab\sqcup)$

String Accepted.

3. # Consider the turing machine given

$M = (K, \Sigma, \delta, s, H)$

$K = (q_0, h)$

$\Sigma = (a, \triangleright, \sqcup)$

$s = q_0$

$H = \{h\}$

q	a	$\delta(q, a)$
q_0	a	$(q_0, \leftarrow)$
q_0	$\sqcup$	$(h, \sqcup)$
q_0	$\triangleright$	$(q_0, \rightarrow)$

This is the turing Machine never halting ($h = \phi$)

To Specify the turing Machine, we need to specify :-

1. The State
2. The content of the tape
3. Position of the R/W head

we choose to break the finite string into two pieces

→ one part to the left of the scanned Square (cell) including the single symbol in the scanned Square.

→ Second Part Possibly empty to the right of the scanned Square.

A configuration of a TM $M = (K, \Sigma, \delta, s, H)$ is a

$$K \times \triangleright \Sigma^* \times (\Sigma^* (\Sigma - \{\sqcup\}) \cup \{\epsilon\})$$

- K: State
- $\triangleright \Sigma^*$: Part of the left of the scanned square including the currently scanned symbol
- $(\Sigma^* (\Sigma - \{\sqcup\}) \cup \{\epsilon\})$: Part possibly empty to the right of the scanned square

$(q_0, \triangleright a b \underline{a} b \sqcup) \xrightarrow[\text{Represented as}]{\text{can be}} (q_0, \triangleright a b a, b \sqcup)$

$(h, \triangleright \sqcup \sqcup \underline{\sqcup} \sqcup a) \longrightarrow (h, \triangleright \sqcup \sqcup \sqcup, \sqcup a)$

$(q, \triangleright \sqcup a \underline{\sqcup \sqcup}) \longrightarrow (q, \triangleright \sqcup a \sqcup \sqcup, \epsilon)$

Notation for TM :-

- # Symbol writing TM $\Rightarrow M_a$ or a
- # Head Moving TM $\Rightarrow$ $\rightarrow$ or R
  $\leftarrow$ or L

NOTE :- Combining Machine to be done later :-

Start State $>$

If $\Sigma = \{a, b, \triangleright, \sqcup\}$
then

$>R \xrightarrow{a, b, \triangleright, \sqcup} R$ or $>R \xrightarrow{a,b,\triangleright,\sqcup} R$ or $>R \rightarrow R$

or $>RR$ or $>R^2$

move its head Right one square, then If that square containing 'a', 'b', 'Δ' or '␣' It move its head one square further to the Right

>R⟲␣ or >R⟲a≠␣ or >R␣

Scane its tape to the Right untill it finds a blank (␣) Character

>R⟲␣ or >R⟲a=␣ or >R␣

~~Scane~~ Find the first Non Black charactered Square to the right of the currently scanned Square.

>L⟲␣ or >L⟲a≠␣ or >R␣

Sane its tape to the left untill it finds a blank character.

>L⟲␣ or >L⟲a=␣ or >L␣

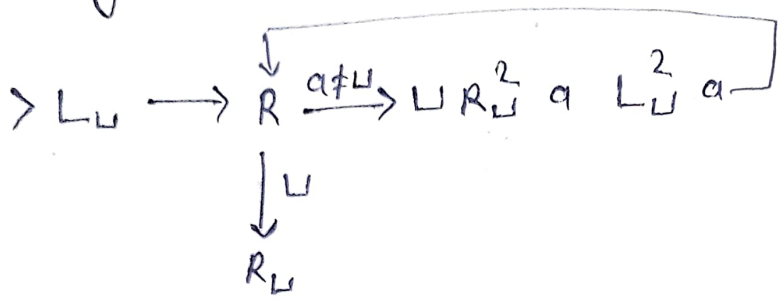
Find the first Non Blank Square to the Left of the currently Scanned square.

>R⟲␣ —a≠␣→ La

Scans to the Right untill it finds a Non Blank Square, then copy the Symbol in that Square, onto the square immidiatly to the Left of where it was found.

Examples:-

Copying Machine:-



$\sqcup w \sqcup \vdash^* \sqcup w \sqcup w \sqcup$

↑ any string

Initial configuration $\sqcup w \sqcup$

$\sqcup w \sqcup \vdash \sqcup w \sqcup \quad (\because L_\sqcup)$

$\sqcup w \sqcup \vdash \sqcup w \sqcup \quad (\because R)$

$\sqcup w \sqcup \vdash \sqcup \sqcup \sqcup \quad (\because \text{copy } w \text{ & } \because a \neq \sqcup)$

$\sqcup w \sqcup \vdash \sqcup \sqcup \sqcup \sqcup \quad (R^2_\sqcup)$

~~$\sqcup \sqcup w \sqcup$~~

$\sqcup \sqcup \sqcup \sqcup \vdash \sqcup \sqcup \sqcup w \quad (a)$ (writing symbol)

$\sqcup \sqcup \sqcup w \vdash \sqcup \sqcup \sqcup w \quad (L^2_\sqcup)$

$\sqcup \sqcup \sqcup w \vdash \sqcup w \sqcup w \quad (a)$

$\sqcup w \sqcup w \vdash \sqcup w \sqcup w \quad (R)$

~~$\sqcup w \sqcup w \vdash$~~

$\sqcup w \sqcup w \vdash \sqcup w \sqcup w \sqcup \quad (R_\sqcup)$ & $(\because \sqcup)$

Do the same thing with [w='ab']

Initial configuration ␣ab␣

␣ab␣ ⊢ ␣a b␣ (∵L␣)

␣ab␣ ⊢ ␣ab␣ (∵R)

␣ab␣ ⊢ ␣␣b␣ (• copy a)

␣␣b␣ ⊢ ␣␣b␣␣ (∵$R_␣^2$)

␣␣b␣␣ ⊢ ␣␣b␣a (Paste a)

␣␣b␣a ⊢ ␣ab␣a (∵$L_␣^2$)

␣␣b␣a ⊢ ␣ab␣a (Paste a)

~~⊢␣~~

␣ab␣a ⊢ ␣ab␣a (∵R)

␣ab␣a ⊢ ␣a␣␣a (copy b)

␣a␣␣a ⊢ ␣a␣␣a␣ ($R_␣^2$)

␣a␣␣a␣ ⊢ ␣a␣␣ab (Paste b)

~~␣a␣␣ab ⊢~~

␣a␣␣ab ⊢ ␣a␣␣ab (L_v^2)

␣a␣␣ab ⊢ ␣ab␣ab (Paste b)

␣ab␣ab ⊢ ␣ab␣ab (R)

␣ab␣ab ⊢ ␣ab␣ab␣ ($R_␣$)

Right shifting TM

$\sqcup w \sqcup \overset{*}{\vdash} \sqcup\sqcup w \sqcup$

where w is any string that do Not contain any blank ($\sqcup$) character

$\sqcup w \sqcup \vdash \sqcup \underline{w} \sqcup$ (L)

$\vdash \sqcup \underline{\sqcup} \sqcup$ (copy w)

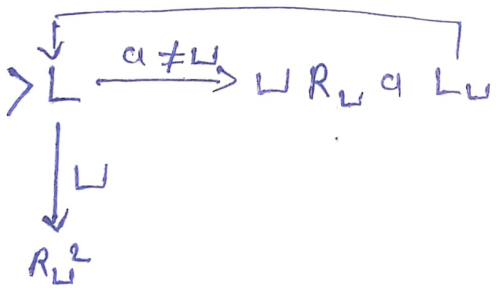
$\vdash \sqcup \sqcup \underline{\sqcup}$ ($R_\sqcup$ or R)

$\vdash \sqcup \sqcup \underline{w}$ (Paste w)

$\vdash \sqcup \underline{\sqcup} w$ (b $L_\sqcup$)

$\vdash \underline{\sqcup} \sqcup w$ (L)

$\vdash \sqcup \sqcup w \sqcup$ ($R_\sqcup^2$)



Checking for w = 'ab'

$\sqcup ab \sqcup \vdash \sqcup a \underline{b} \sqcup$ (L)

$\vdash \sqcup a \underline{\sqcup} \sqcup$ (copy b)

$\vdash \sqcup a \sqcup \underline{\sqcup}$ (R)

$\vdash \sqcup a \sqcup \underline{b}$ (Paste b)

$\vdash \sqcup a \underline{\sqcup} b$ ($L_\sqcup$)

$\vdash \sqcup \underline{a} \sqcup b$ (L)

$\vdash \sqcup \underline{\sqcup} \sqcup b$ (copy a)

$\vdash \sqcup \sqcup \underline{\sqcup} b$ (R)

$\vdash \sqcup \sqcup \underline{a} b$ (Paste)

$\vdash \sqcup \underline{\sqcup} a b$ ($L_\sqcup$)

$\vdash \underline{\sqcup} \sqcup a b$ (L)

$\vdash \sqcup \sqcup a b \sqcup$ ($R_\sqcup$)

check for abc

$\sqcup abc\sqcup \vdash \sqcup\sqcup abc\sqcup$ (L)

$\vdash \sqcup\sqcup ab\sqcup\sqcup$ (copy c)

$\vdash \sqcup ab\sqcup\sqcup$ ($R_\sqcup$)

$\vdash \sqcup ab\sqcup c$ (Paste c)

$\vdash \sqcup ab\sqcup c$ ($L_\sqcup$)

$\vdash \sqcup ab\sqcup c$ (L)

$\vdash \sqcup a\sqcup\sqcup c$ (copy b)

$\vdash \sqcup a\sqcup\sqcup c$ ($R_\sqcup$)

$\vdash \sqcup a\sqcup b c$ (Paste b)

$\vdash \sqcup a\sqcup bc$ ($L_\sqcup$)

$\vdash \sqcup a\sqcup bc$ (L)

$\vdash \sqcup\sqcup\sqcup bc$ (copy a)

$\vdash \sqcup\sqcup\sqcup bc$ ($R_\sqcup$)

$\vdash \sqcup\sqcup abc$ (Paste a)

$\vdash \sqcup\sqcup abc$ ($L_\sqcup$)

$\vdash \sqcup\sqcup abc$ (L)

$\vdash \sqcup\sqcup abc\sqcup$ ($R^2_\sqcup$)

left shifting TM

$\sqcup ab \sqcup \vdash^{*} ab \sqcup$

$\sqcup ab \sqcup \vdash \underline{\sqcup} ab \sqcup$ ($L_{\sqcup}$)

$\vdash \sqcup \underline{a} b \sqcup$ (R)

$\vdash \sqcup \sqcup \underline{b} \sqcup$ (copy a)

$\vdash \underline{\sqcup} \sqcup b \sqcup$ ($L_{\sqcup}$)

$\vdash \boxed{\ } \underline{a} \sqcup b \sqcup$ (Paste a)

$\vdash \underline{a} \sqcup b \sqcup$ ($R_{\sqcup}$)

$\vdash a \sqcup \underline{b} \sqcup$ (R)

$\vdash a \sqcup \underline{\sqcup} \sqcup$ (copy b)

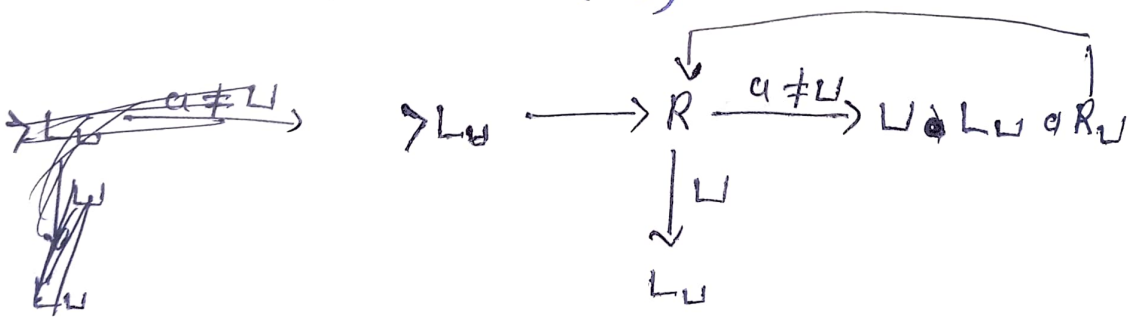
$\vdash a \underline{\sqcup} \sqcup \sqcup$ ($L_{\sqcup}$)

$\vdash a \underline{b} \sqcup \sqcup$ (Paste b)

$\vdash ab \underline{\sqcup} \sqcup$ ($R_{\sqcup}$)

$\vdash ab \sqcup \underline{\sqcup}$ (R)

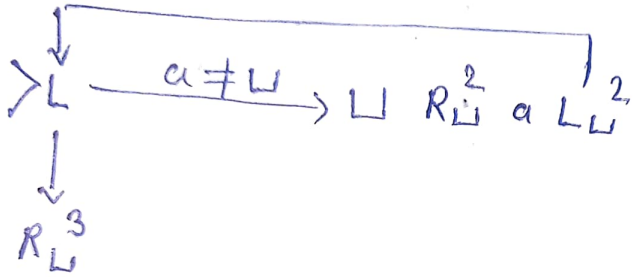
$\vdash ab \underline{\sqcup}$ ($L_{\sqcup}$)



Construct TM which shift 2 time Right

NOTE :- No Need to construct another machine we can modify the (Left ~~Shift~~ Right machine) By Increasing the power of "$R_\sqcup$" & "$L_\sqcup$"

we are going to increasing the Power 2 time.



Target $\boxed{\sqcup w \sqcup \vdash \sqcup\sqcup\sqcup w \sqcup}$

Solve

$\sqcup w \sqcup \vdash \sqcup \underline{w} \sqcup$

$\vdash \sqcup \sqcup \sqcup$

$\vdash \sqcup \sqcup \sqcup \sqcup$

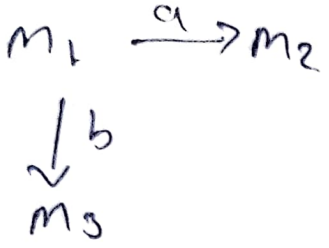
$\vdash \sqcup \sqcup \sqcup w$

$\vdash \sqcup \sqcup \sqcup w$

$\vdash \sqcup \sqcup \sqcup w$

$\vdash \sqcup \sqcup \sqcup w \sqcup$

Combining Machine :-



(i) Start in the initial state of M_1

(ii) Operate as M_1 until it halts

(iii) If the current scanned symbol is

a → (i) Initiate M_2 (ii) operate as M_2

b → (i) Initiate M_3 (ii) operates as M_3

$M_1 = (K_1, \Sigma, \delta_1, s_1, H_1)$
$M_2 = (K_2, \Sigma, \delta_2, s_2, H_2)$
$M_3 = (K_3, \Sigma, \delta_3, s_3, H_3)$
all are disjointented

$\Downarrow$

$M = (K, \Sigma, \delta, s, H)$ combining machine.

where $K = K_1 \cup K_2 \cup K_3$

$s = s_1$

$H = H_2 \cup H_3$

for each $a \in \Sigma$ & $q \in K - H$, $\delta(q, a)$ is defined as follow :-

(a) if $q \in K_1 - H_1$, then $\delta(q, a) = \delta_1(q, a)$

(b) if $q \in K_2 - H_2$, then $\delta(q, a) = \delta_2(q, a)$

(c) if $q \in K_3 - H_3$, then $\delta(q, a) = \delta_3(q, a)$

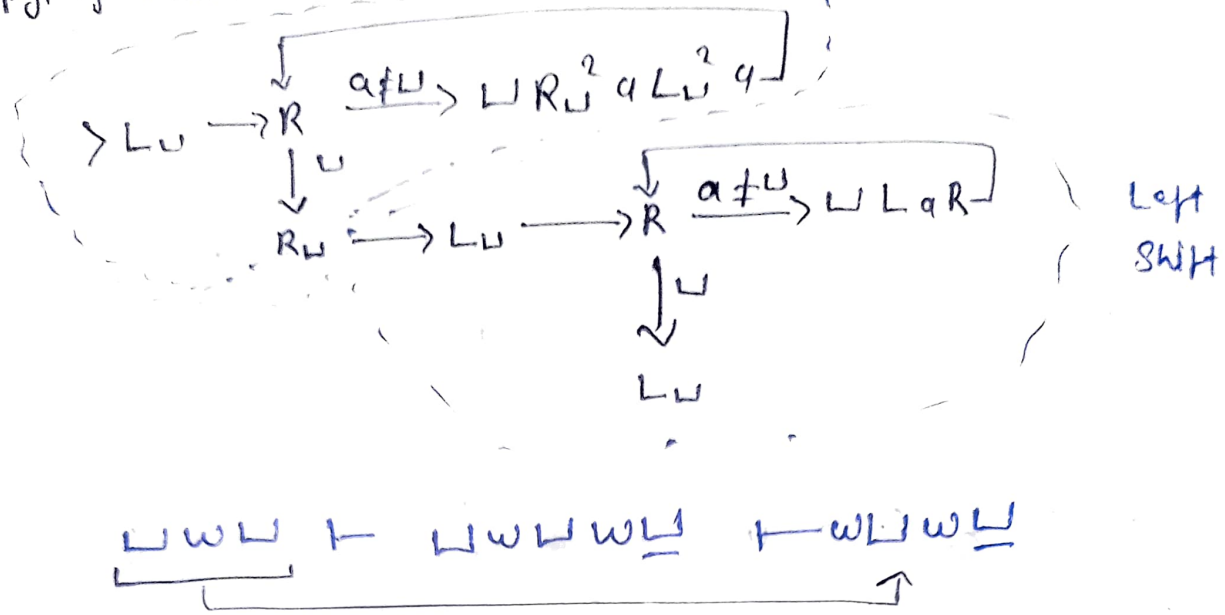
(d) If $q \in H_1$

then

$\delta(q, a) = s_2$ and $\delta(q, a) \in H$ $a \neq a$

$\delta(q, b) = s_3$ otherwise $a \neq b$

Copying Machine



$\sqcup w \sqcup \vdash \sqcup w \sqcup w \sqcup \vdash w \sqcup w \sqcup$

Computing with TM :-

The input string with no blank symbol in it, has a left most symbol $\triangleright$, with a blank to the left & blank to the Right.

~~$(s, \triangleright, \sqcup)$~~ $(s, \triangleright \sqcup w)$

Let $M = (K, \Sigma, \delta, s, H)$ be a TM such that

$H = \{y, n\}$

yes (w is accepted) NO (w is rejected)

(1) if $(s, \triangleright \sqcup w)$ yields an accepting configuration then M accepts an input w

(2) if $(s, \triangleright \sqcup w)$ yields an ~~a~~ Rejecting configuration then M Rejects an input w

This is called Decidable Machine or Turing Acceptable Machine.

* M decides a language $L \subseteq \Sigma_0^*$ where

$$\Sigma_0 \in \Sigma - \{\sqcup, \triangleright\}$$

if for ∀ string $w \in \Sigma_0^*$ the following is true:

(i) if $w \in L$ then M accepts w

(ii) if $w \in L$ then M Rejects w

(Language) L: **Recursive language**

कोई भी Machine उस Language के सभी w को accept करती है या Reject करती है।

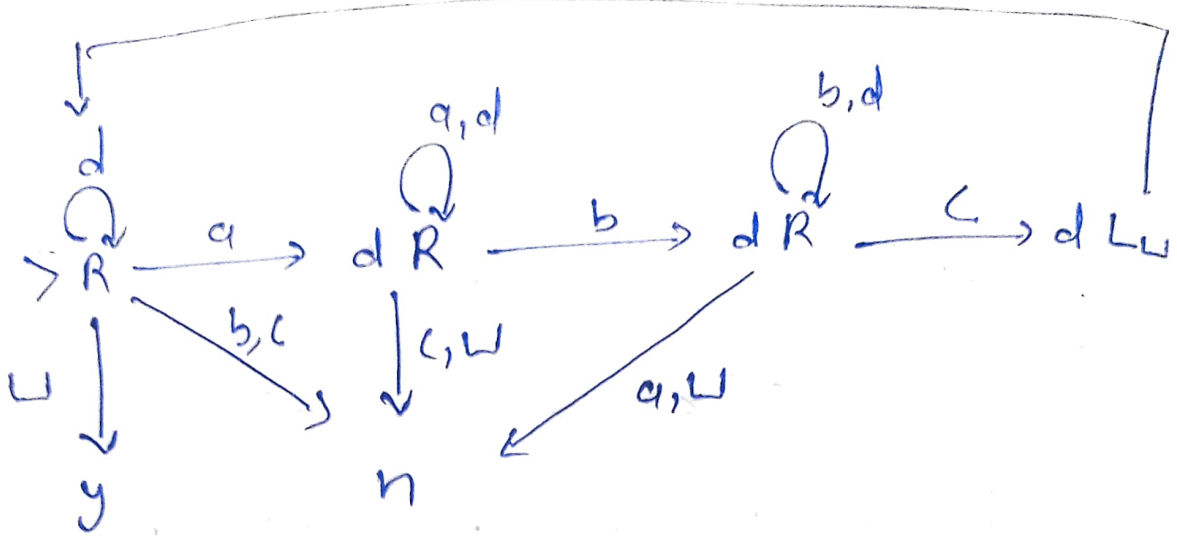
$L = \{ a^n b^n c^n : n \geq 0 \}$ Non CFL Language.

↓

we have to construct turing Decidable machine for this language Because it is Not accepted By PDA.

a a b b c c ⟶ "N"

a a b b c c ⟶ 'y'



Ex:-

$\triangleright$ ␣ a a b b c c

$\vdash$ $\triangleright$ ␣ a a a b b c c

$\vdash$ $\triangleright$ ␣ d a b b c c

$\vdash$ $\triangleright$ ␣ d a a b b c c

$\vdash$ $\triangleright$ ␣ d a b b b c c

$\vdash$ $\triangleright$ ␣ d a d b c c

$\vdash$ $\triangleright$ ␣ d a d b c c

$\vdash$ $\triangleright$ ␣ d a d b c c

$\vdash$ $\triangleright$ ␣ d a d b d c

$\vdash$ $\triangleright$ ␣ d a d b d c

$\vdash$ $\triangleright$ ␣ d a d b d c

$\vdash$ $\triangleright$ ␣ d a d b d c

$\vdash$ $\triangleright$ ␣ d d d b d c

$\vdash$ $\triangleright$ ␣ d d d b d c

$\vdash$ $\triangleright$ ␣ d d d d d c

$\vdash$ $\triangleright$ ␣ d d d d d c

$\vdash$ $\triangleright$ ␣ d d d d d c

$\vdash$ $\triangleright$ ␣ d d d d d d

Automata theory

Design a TM for

$$f : \Sigma_0^* \longrightarrow \Sigma_0^*$$

where $f(w) = ww^R$

that is

$$q_0, \triangleright \sqcup abb \sqcup \vdash h, \triangleright \sqcup abba \sqcup$$

Let f be any function from Σ_0^* to Σ_0^*

$\Sigma_0 = \Sigma - \{\sqcup, \triangleright\}$

$\omega \in L$ } halt
$\omega \notin L$ }

we say that, M computes function f, if

$\forall\ \omega \in \Sigma_0^* \quad M(\omega) = f(\omega)$

that is, M halts on input ω

A function f is called recursive, if there is a TM M that compute f

recursive function.

Recursive Enumerable language :-

Let $M = (K, \Sigma, \delta, s, H)$ be a TM

let $\Sigma_0 \subseteq \Sigma - \{\sqcup, \triangleright\}$ be an alphabet

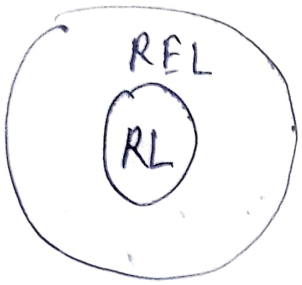
& $L \subseteq \Sigma_0^*$ be a language

we say that, M semidecides L if for any string $\omega \in \Sigma_0^*$ the following is true

(1) $\omega \in L$ if M halt on ω

(2) $\omega \in L$ M never enter a halting state. i.e machine continous its computation continously

L is Recursive Enumerable languages



REL: Recursive Enumerable language
RL : Recursive language (Decidable)
{Y, N}

REL = Semi decidable Machine
{Y}
w∈L
M halt
w∉L

Question :- Prove that if L is recursive language then its compliment $\bar{L}$ is also recursive.

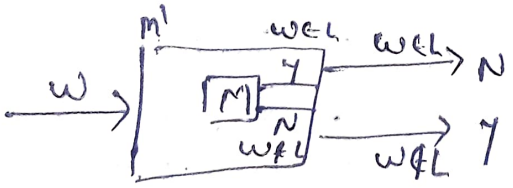
That is Recursive languages are closed under complimentation.

Ans :-

Let M = (K, Σ, δ, s, {Y, N})
is a TM that decides L then there must exist a TM M' which will be

M' = (K, Σ, δ', s, {N, Y})

that Must also decides L



M & M' are identical, except that it reverses the role of Y & N

$$\delta'(q,a) = \begin{cases} n & \text{if } \delta(q,a) = y \\ y & \text{if } \delta(q,a) = N \\ \delta(q,a) & \text{otherwise} \end{cases}$$

$M'(w) = y$ iff $M(w) = N$ and therefore M' decides $\overline{L}$

So we can say Recursive Language are closed under complimentation.

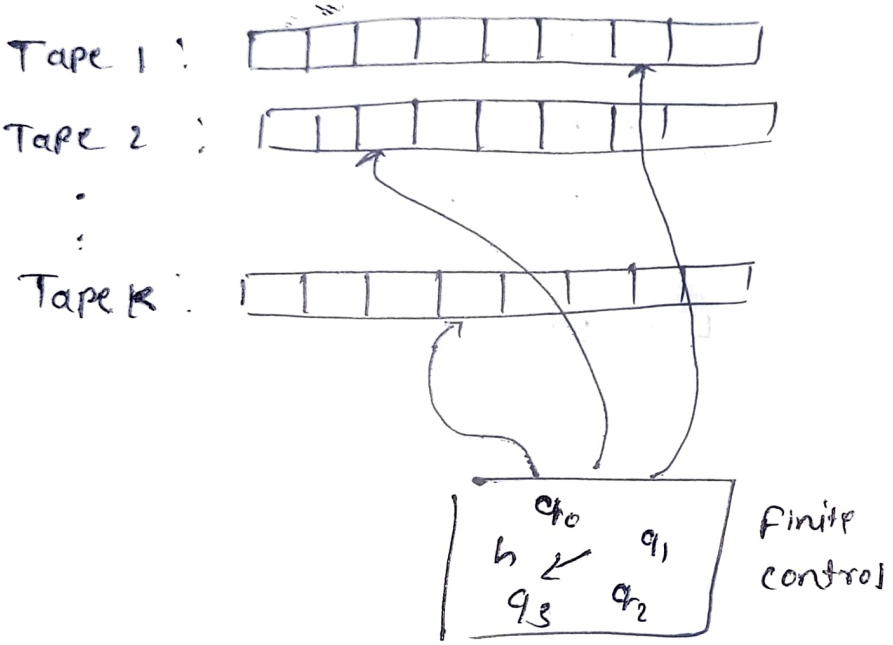
← multitape turing machine

Multitape TM

Let $M = (K, \Sigma, \delta, s, H)$ be a

k-tape TM. A configuration of M is a Member of

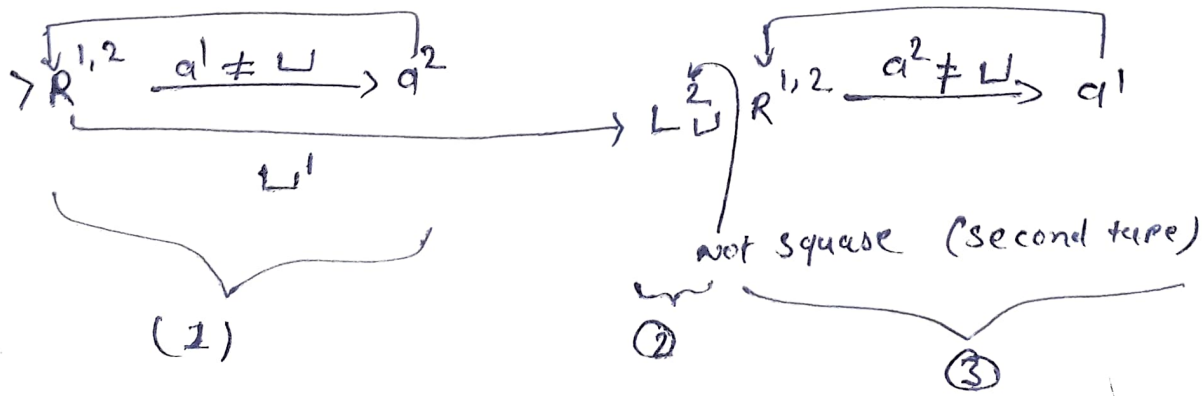
$$K \times \triangleright \Sigma^* \times (\Sigma^*(\Sigma - \{\sqcup\} \cup \{\epsilon\}))^k$$



Q :- Design a 2-tape TM for the copying Machine. to accomplish the task of tranforming

$\triangleright \sqcup w \sqcup$ into $\triangleright \sqcup w \sqcup w \sqcup$

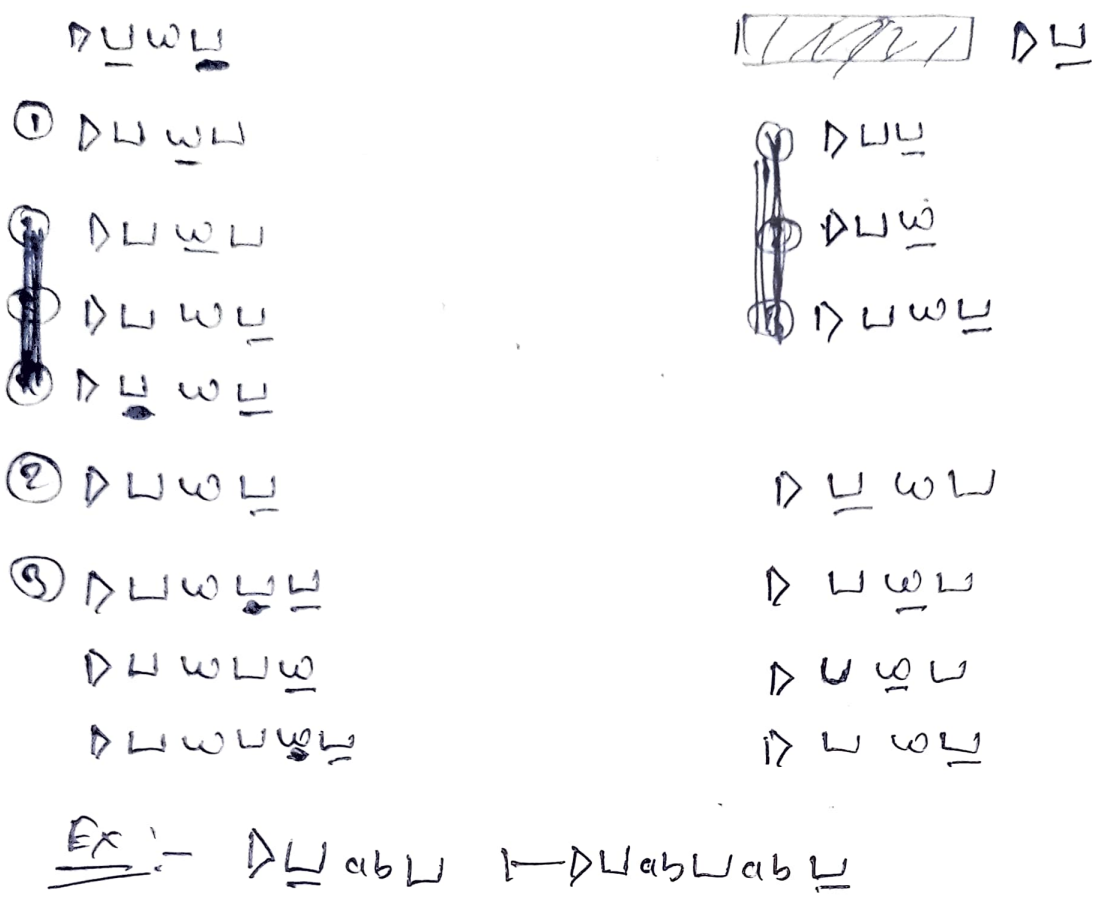
where $w \in \{a, b\}^*$



1st tape

2nd tape

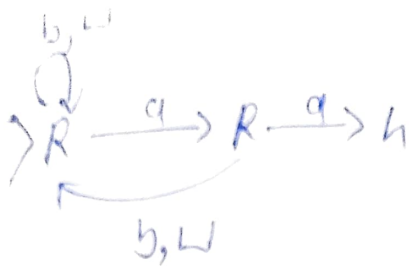
(initial config)



Ex :- ▷␣ab␣ ⊢ ▷␣ab␣ab␣

Date :- 25/12/24

Ex.6 # Design a TM that scans to the right until it finds two consecutive a's and then halts.



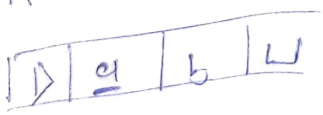
$\Sigma = \{a, b, \triangleright, \sqcup\}$

baab

Then halt, ans

Ex.9 # Do the machines LR and RL always accomplish the same thing.?

for LR



(∵ L)

▷ a b ⊔

▷ a b ⊔

▷ a b ⊔ (∵ R)

for RL

▷ a b ⊔

▷ a b ⊔ (∵ R)

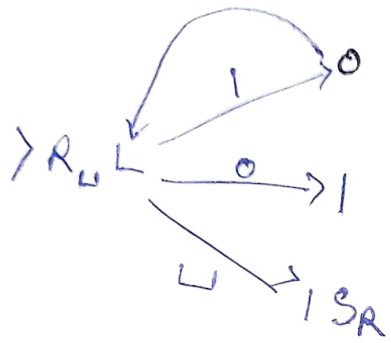
▷ a b ⊔ (∵ L)

Not Proceed to the same thing

$Succ(n) : n+1$ (Successor of n could be $n+1$)

Ex:

$Succ(\triangleright \underline{1}011\sqcup) \vdash \triangleright 1100\sqcup$



S_R : Right shifting TM

$\triangleright 1011\sqcup \vdash \triangleright \underline{1}011\sqcup$

$\vdash \triangleright 10\underline{1}1\sqcup$

$\vdash \triangleright 10\underline{1}0\sqcup$ (copy 1)

$\vdash \triangleright 10\underline{1}0\sqcup$

$\vdash \triangleright 10\underline{0}0\sqcup$ (copy 1)

$\vdash \triangleright 1\underline{0}00\sqcup$

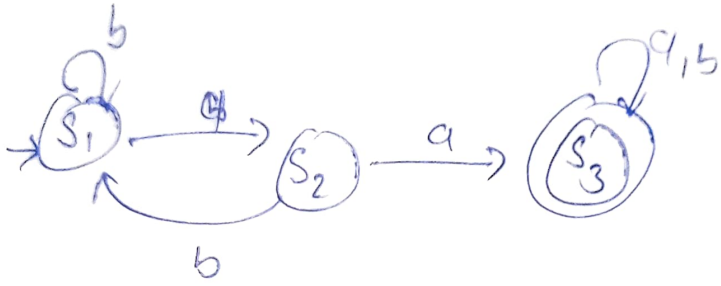
$\vdash \triangleright \underline{1}100\sqcup$

Ex:- ▷ | | | | | ⊔ ⊢ ▷

▷ ⊔ | | | | ⊔ ⊢ ▷ | | | ⊔ ⊔ ⊔

Marking unambigous CFG.

For 'aa' as substring.



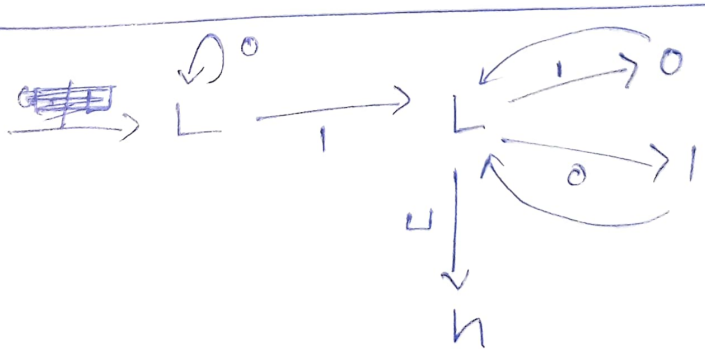
FA to CFG.

$S_1 \rightarrow bS_1 \mid aS_2$

$S_2 \rightarrow aS_3 \mid bS_1$

$S_3 \rightarrow aS_3 \mid bS_3 \mid \epsilon$

#



▷ 0 | | 0 ⊔ ⊢ ▷ 0 | | 0 ⊔

2's compliment

⊢ ▷ 0 | | 0 ⊔

⊢ ▷ 0 | | 0 ⊔

⊢ ▷ 0 | 1 0 ⊔

⊢ ▷ 0 0 | 0 ⊔

⊢ ▷ 0 0 | 0 ⊔

⊢ ▷ 1 0 | 0 ⊔

Turing Decidable :-

A language L is Decidable if there exist a TM such that for ∀ strings w

if w ∈ L, M enteres 'y'
if w ∉ L, M enters 'n'

Alonzo Church
↳ Alan turing

function which are algorithmically computable that has a TM.

26/11/24

Constructing a Universal turing machine :-

The limitation of turing machine is that they are Hardwired that is they specialized to solve one particular Problem.

Since the real computer are reprogrameble, so we need a generic TM. which should be programmed. in a manner a general purpose computer is Programmed.

Universal TM simulates & any TM.

→ Inputs of UTM are :-

(1) Description of the transition of M

(2) Input string of M

→ Convention :-

(1) A string representing a TM state is of form $(\{q\}, \{0,1\}^*)$ i.e. a letter q following by a binary string.

(2) A tape symbol is always represented as a string $(\{a\}\ \{0,1\}^*)$

Let $M = (K, \Sigma, \delta, s, H)$ be a TM & Let i & j are 2 smallest integer such that $2^i \geq |K|$

$2^j \geq |\Sigma| + 2$

$\{\leftarrow, \rightarrow\}$

$\sqcup = q0^j$

$\rhd = a\,0^{j-1}\,1$

$\leftarrow = a\,0^{j-2}\,10$

$\rightarrow = a\,0^{j-2}\,11$

j bits

Ch.5 { Lewis Papadimitriou }

Start state $q\,0^j$

The sequence of strings of the form

(q, a, P, b) with q & P representing state & a & b representing symbols.

Example for UTM:- 5.2

Que Consider a TM $M = (K, \Sigma, \delta, s, \{h\})$

where $K = \{s, q, h\}$

$\Sigma = \{\sqcup, \rhd, a\}$

δ is given below

State	symbol	δ
s	a	$(q, \sqcup)$
s	$\sqcup$	$(h, \sqcup)$
s	$\rhd$	$(s, \rightarrow)$
q	a	(s, a)
q	$\sqcup$	$(s, \rightarrow)$
q	$\rhd$	$(q, \rightarrow)$

Consider a TM give the Discription / Encoding of this machine in form of UTM.

Solve

$K = \{S, q, h\}$

$\Sigma = \{\sqcup, \triangleright, a\}$

$2^i \geq |K|$ Hence $i = 2$

$2^j \geq |\Sigma| + 2$ Hence $j = 3$

State / Symbol	Representation
S	$q\ 00$
q	$q\ 01$
h	$q\ 11$ ← Higher Priority Halting State.
$\sqcup$	$a\ 000$
$\triangleright$	$a\ 001$
$\leftarrow$	$a\ 010$
$\rightarrow$	$a\ 011$
a	$a\ 100$

① $\delta(S, a, q, \sqcup) \rightarrow q00\ a100\ q01\ a000$

② $(S, \sqcup, h, \sqcup) \rightarrow q00\ a000\ q11, a000$

③

④

⑤

⑥

Therefore the discription of TM is given below ① ② ③ ④ ⑤ ⑥ . . . .

q_{00} a_{100} q_{01} a_{000} q_{00} a_{000} q_{11} a_{000}

Halting Problem :- Suppose that you have written a Program (in any language)

It takes as input any Program 'P', written in the Same language & an input 'n' of that Program. By Some clever analysis your Program always determines correctly whether the Program is halting on input 'n' or whether it would run forever. You have Named this Program halts (P, n).

(Undecidability)