Noam Chomsky (1956)

Languages can be represented by using grammar in syntactical way.

Grammar can be represented by following four tuples :

Grammar (V_n, Σ, P, S) / (N, t, P, S)

V_n/N : Finite non empty set of variables
- Represented by upper case letters / Capital
- known as non terminals
- used for generation

Σ/t : Finite non empty set of inputs
- Represented by lowercase letters (known as terminals)
- used for termination
- Disjoint from V_n/N

S : start symbol of the grammar

P : Production rules of grammar

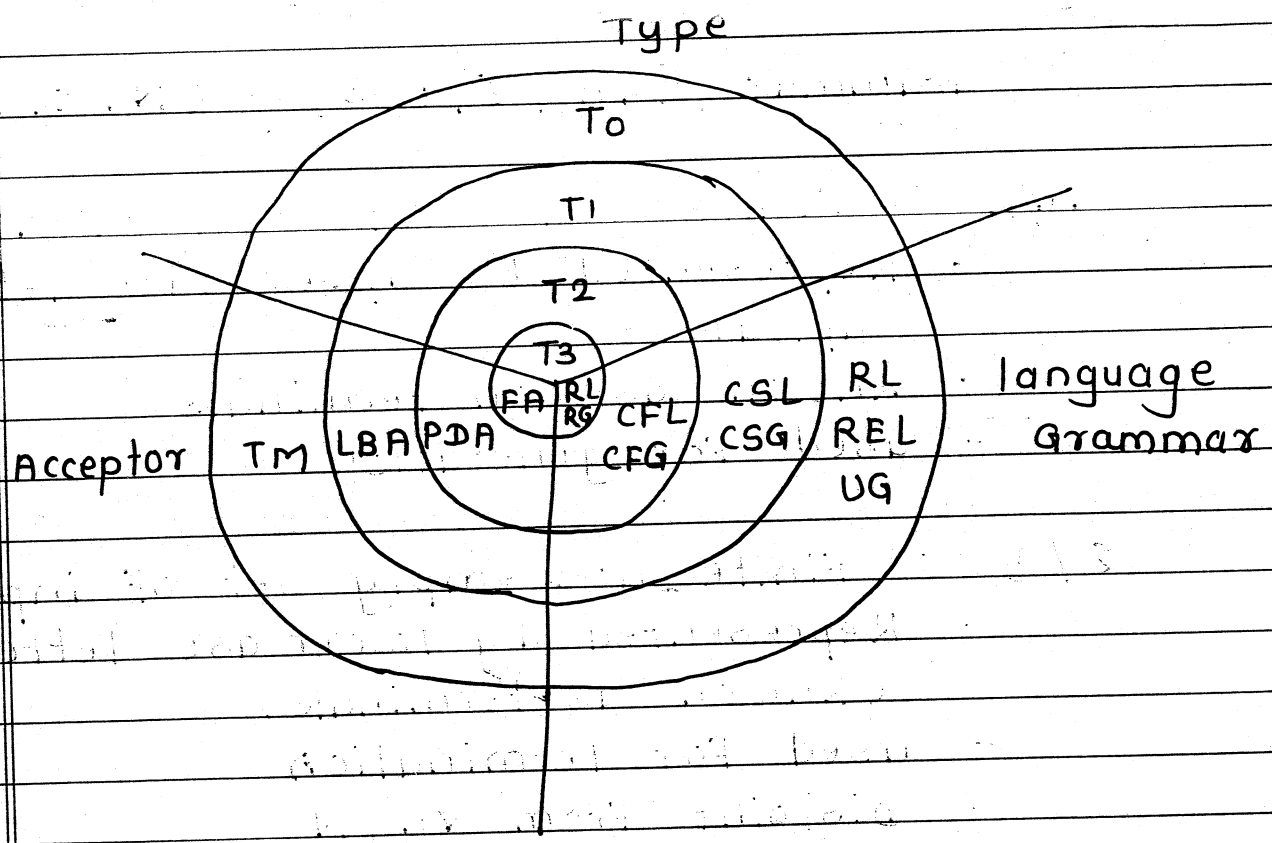
$\alpha \longrightarrow \beta$

α produces β
α generates β
α gives β

Chomsky's Hierarchy

Automata theory languages are divided into following four categories :

Type 0 - Type 3



L (FA) = RL

L (PDA) = CFL + RL

L (LBA) = CSL + CFL + RL

L (TM) = RL + REL + CSL + CFL + RLG

↗ universal machine

Type	language Grammar	Production rule format $x \longrightarrow y$	Acceptor
T3	RL RG	$x \Rightarrow$ only one NT $Y \Rightarrow tNT \mid NTt \mid t \mid \epsilon$	FA
T2	CFL CFG	$x \Rightarrow$ only one NT $Y \Rightarrow$ Any string of t and NT	PDA
T1	CSL CSG	$x \Rightarrow$ Any string of NT $Y \Rightarrow$ Any string of T and NT as long as x	LBA
T0	RL REL: UG	$x \Rightarrow$ Any string of NT $Y \Rightarrow$ Any string of NT and T	TM

CFG

CFG can take care of context free language as well as regular language.

$\Sigma = \{a\}$

FA and regular grammar

language is accepted by FA is called as regular language and can be expressed with regular expression.

It can also be represented by type 3 grammar which is known as regular grammar.

So, it is always possible to convert FA into regular grammar and vice versa.

$X \longrightarrow aX$

$X \longrightarrow Xa$

$X \longrightarrow a$

$X \longrightarrow \varepsilon$

~~LLG~~ ~~RLG~~

$X \longrightarrow Xa$	$X \longrightarrow aX$
$X \longrightarrow a$	$X \longrightarrow a$
$X \longrightarrow \varepsilon$	$X \longrightarrow \varepsilon$

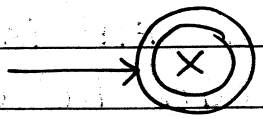
conversaion can be asked as in exam -

- FA - RLG
- RLG - FA
- FA - LLG
- LLG - FA.

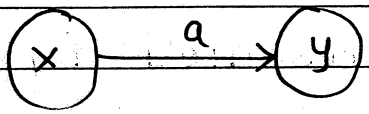
RLG - LLG } first RLG - FA then FA - LLG

LLG - RLG }

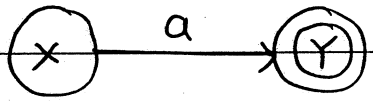
* FA to RLG rules



$x \rightarrow \varepsilon$

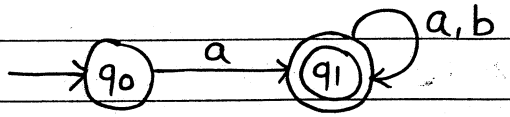


$x \rightarrow aY$



$x \rightarrow aY \mid a$

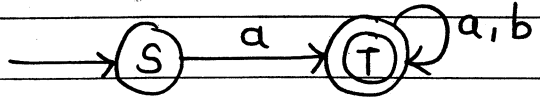
FA to RLG ⇐ 1)



FA $(Q, \Sigma, \delta, q_0, F) \Rightarrow$ G (V_n, Σ, P, S)

Rename: $q_0 = S$

$q_1 = T$



$a.(a+b)^*$

$S \rightarrow aT \mid a$

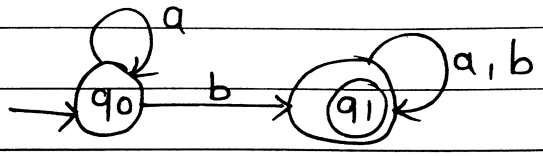
$T \rightarrow aT \mid a$

$T \rightarrow bT \mid b$

The RLG is no. of arrows except start arrow

Terminal o/p can be decided by how many arrows coming towards it.

2)

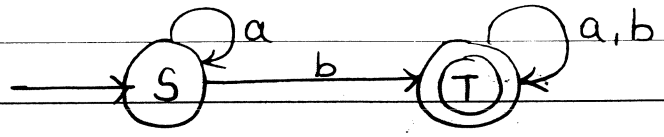


FA $\Rightarrow$ G

$(Q, \Sigma, \delta, q_0, F) \Rightarrow (V_n, \Sigma, P, S)$

Rename : $q_0 = S$

$q_1 = T$



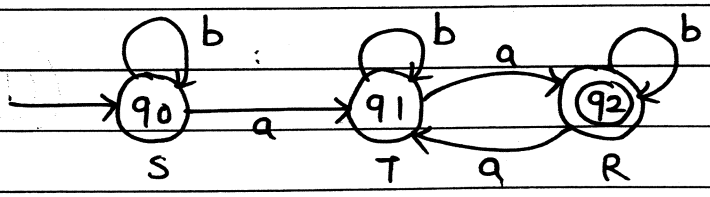
$S \rightarrow aS$

$S \rightarrow bT \mid b$

$T \rightarrow aT \mid a$

$T \rightarrow bT \mid b$

3)



$S \rightarrow bS$

$S \rightarrow aT$

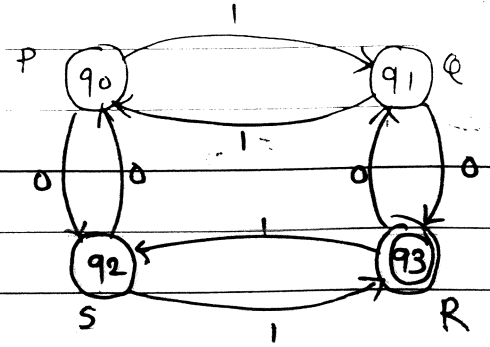
$T \rightarrow bT$

$T \rightarrow aR \mid a$

$R \rightarrow bR \mid b$

$R \rightarrow aT$

4)



$P \rightarrow 1Q$

$Q \rightarrow 1P$

$P \rightarrow 0S$

$S \rightarrow 0P$

$Q \rightarrow 0R \mid 0$

$R \rightarrow 1S$

$R \rightarrow 0Q$

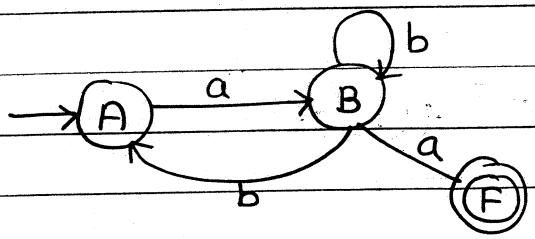
$S \rightarrow 1R \mid$

RLG to FA

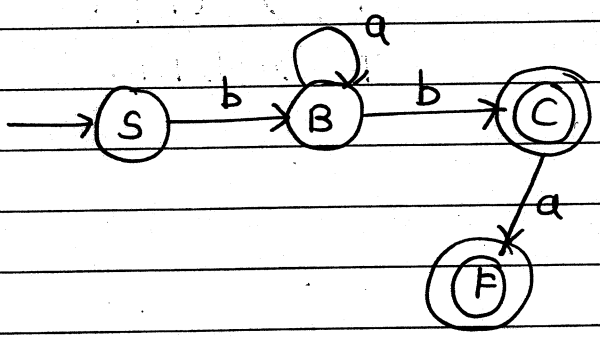
1) $A \rightarrow aB$
$B \rightarrow bB$
$B \rightarrow a$ ✓ no state is given for a to reach
$B \rightarrow bA$

G FA

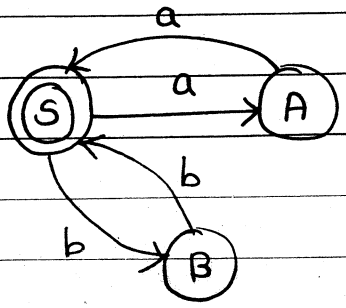
$(V_n, \Sigma, P, S) \Rightarrow (Q, \Sigma, \delta, q_0, F)$



2) $S \rightarrow bB$
$B \rightarrow bC \mid aB \mid b$
$C \rightarrow a$



3) $S \rightarrow aA \mid \varepsilon$
$S \rightarrow bB$
$A \rightarrow aS \mid a$
$B \rightarrow bS \mid b$



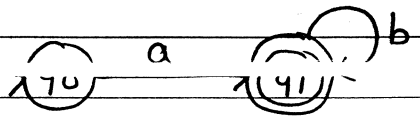
FA to LLG

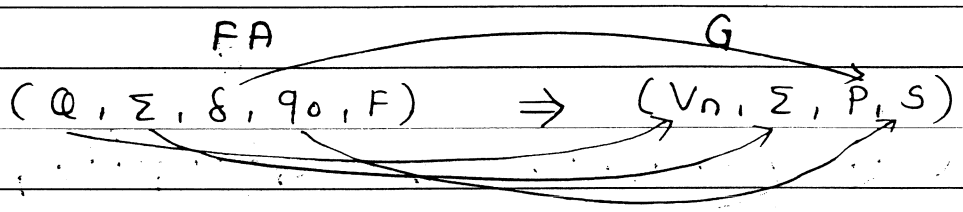
step1 : Interchange start and final state

step2 : Reverse direction of all transition

step3 : Write grammar in LLG form

1)

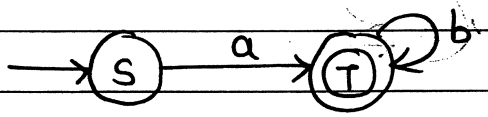




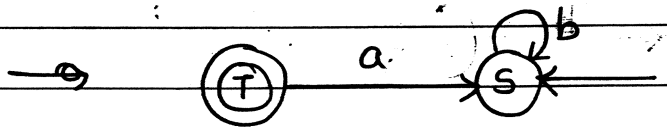
Rename

$q_0 = S$

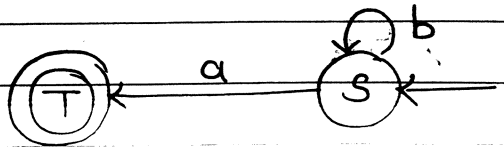
$q_1 = T$



step1 : Interchange start and final state



step2 : Reverse direction of all transition



step3 :

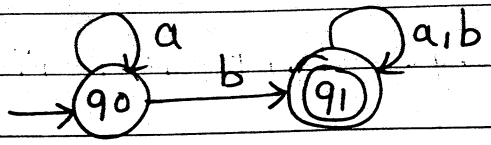
$S \rightarrow Sb$

$S \rightarrow Ta \mid a$

$T \rightarrow \varepsilon$

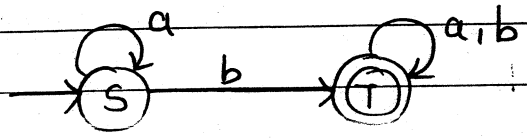
(When there is no any arrow at final state then $T \rightarrow \varepsilon$)

2)



Rename : $q_0 = S$

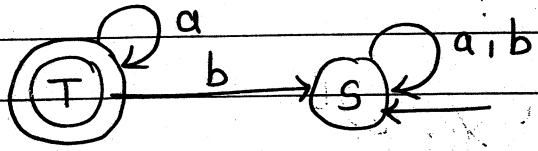
$q_1 = T$



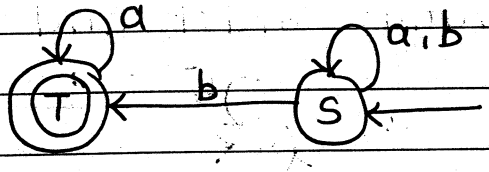
FA

$(Q, \Sigma, \delta, q_0, F) \Rightarrow (V_n, \Sigma, P, S)$

step1 : Interchange start and final state



step2 : Reverse direction of all transitions



step3 : LLG format

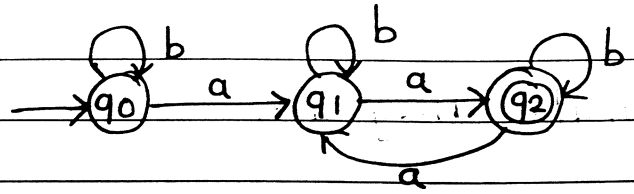
$S \rightarrow Sa$

$S \rightarrow Sb$

$S \rightarrow Tb \mid b$

$T \rightarrow Ta \mid a$

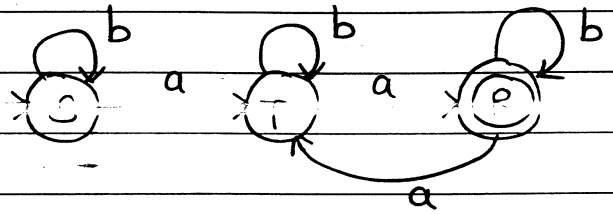
3)



Rename : $q_0 = Q$

$q_1 = T$

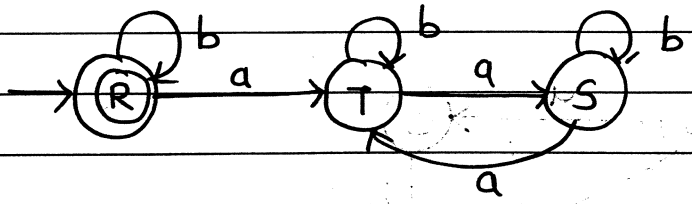
$q_2 = R$



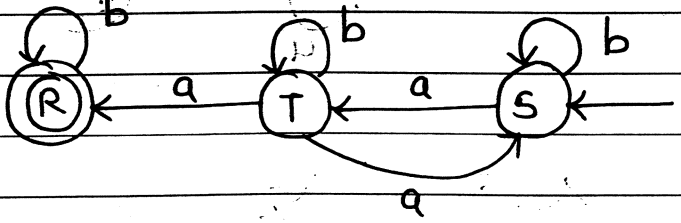
FA G

$(Q, \Sigma, \delta, q_0, F) \Rightarrow (V_n, \Sigma, P, S)$

step 1 : Interchange start and final state



step 2 : Reverse direction of all transitions



step 3 :

$S \longrightarrow Sb$

$S \longrightarrow Ta$

$T \longrightarrow Tb$

$T \longrightarrow Sa$

$T \longrightarrow Ra \mid a$

$R \longrightarrow Rb \mid b$

LLG to FA

step1 : Draw state diagram

step2 : Reverse direction of all transitions

step3 : Interchange start and final state

1) $S \rightarrow Ca \mid Bb$

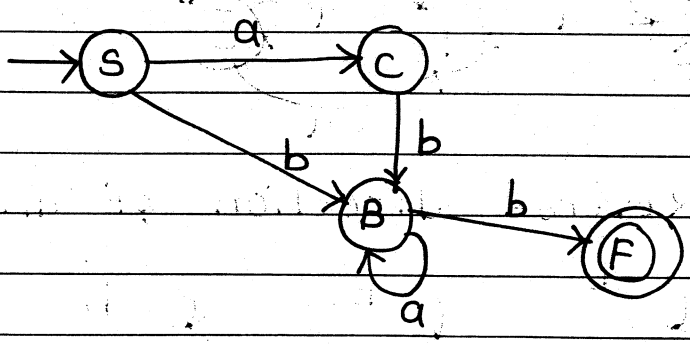
$C \rightarrow Bb$

$B \rightarrow Ba \mid b$

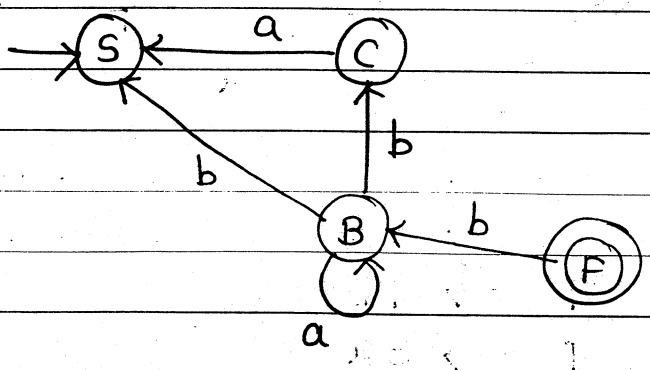
G FA

$(V_n, \Sigma, P, S) \Rightarrow (Q, \Sigma, \delta, q_0, F)$

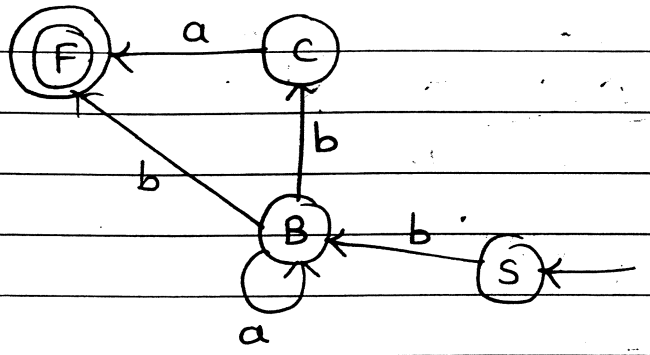
step1 : Draw state diagram



step2 :



steps



$S \to Bb$ $S \to Ca$ $S \to Ca$

$\to bb$ $\to Bba$ $\to Bba$

$\to bba$ $\to Baba$

$\to baba$

9. Convert following LLG into RLG

$S \to Sb$

$S \to Ta$

$T \to Tb$ LLG $\Rightarrow$ FA $\Rightarrow$ RLG

$T \to Sb$

$T \to Ra \mid a$

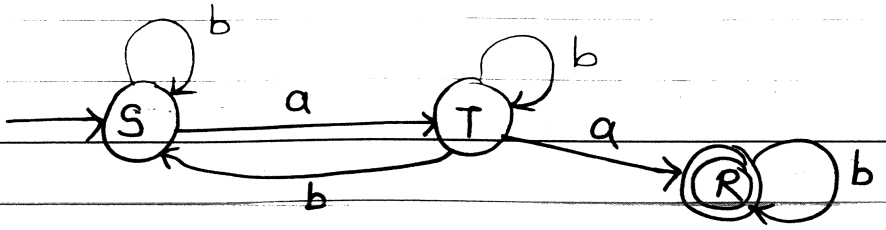
$R \to Rb \mid b$

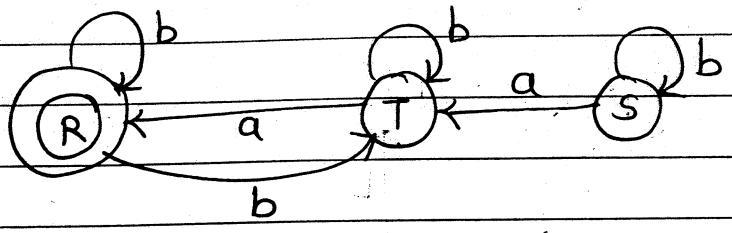
(1) LLG $\Rightarrow$ FA

G FA

$(V_n, \Sigma, P, S) \Rightarrow (Q, \Sigma, \delta, q_0, F)$

step1 : Draw state diagram





(2) FA to RLG

$R \rightarrow bR \mid b$

$T \rightarrow aR \mid a$

$T \rightarrow bT$

$R \rightarrow bT$

$S \rightarrow bS$

$S \rightarrow aT$

* Grammar plays a very vital role in syntax and semantic analysis of phase of compiler.
So, to incorporate grammar into compiler, the grammar must be
1) Simplified
2) Normalized
3) Ambiguity free

1) Simplification of grammar

Grammar can be simplified by using following three steps :

I) Removal of useless symbols
II) Removal of ε production
III) Removal of unit production

I) Removal of useless symbols

non generating
The symbol which can not be terminated by terminals either by directly or indirectly

Non reachable
The symbol which can not be reached from start symbol

Non generating

1) $S \rightarrow AB \mid ss \mid a$
$A \rightarrow aA$
$B \rightarrow bB$

$A \rightarrow aA$
$B \rightarrow bB$

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

$S \rightarrow a$ ✓
$S \rightarrow AB$

A and B are non-generating

$S \rightarrow a$

(Remove from given all which contain A & B)

2) $S \rightarrow Aa \mid Bb \mid a \mid b$

$A \rightarrow Aa \mid a$

$B \rightarrow bB$

B is non generating

$S \rightarrow Aa \mid a \mid b$

$A \rightarrow Aa \mid a$

3) $S \rightarrow A$

$A \rightarrow aA \mid b$

$NT = \{S, A\}$

4) $S \rightarrow AB \mid cA$

$B \rightarrow BC \mid AB$

$A \rightarrow a$

$C \rightarrow aB \mid b$

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

B is non generating

5) $S \rightarrow aAa$

$A \rightarrow Sb \mid bCC$

$C \rightarrow abb$

$E \rightarrow aC$

Non Reachable

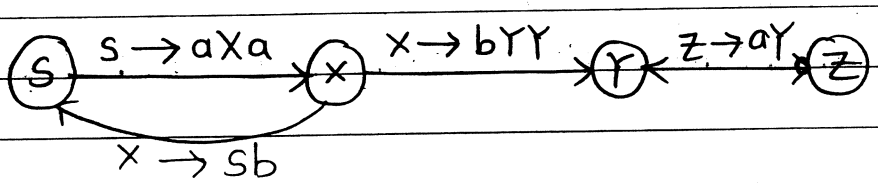
Non reachability can be decided with th help of dependancy graph

$S \rightarrow aXa$

$X \rightarrow Sb \mid bYY$

$Y \rightarrow abb$

$Z \rightarrow aY$



⇓

$S \rightarrow aXa$

$X \rightarrow Sb \mid bYY$

$Y \rightarrow abb$

e.g

$S \rightarrow aAa$

$A \rightarrow bBB$

$B \rightarrow ab$

$C \rightarrow aB$ → non reachable

eg

$S \rightarrow aS \mid AB$

$A \rightarrow bA$

$B \rightarrow AA$

} all are reachable

II) Removal of ε production

ε production means presence of null move and because of it, compiler may transit from one state to another state without receiving any symbol.

So, the presence of ε in a grammar is hazardous for compiler development.

The symbol with ε production is known as nullable symbols.

1) $S \rightarrow AB$
$A \rightarrow aA \mid \varepsilon$
$B \rightarrow bB \mid \varepsilon$

$A \rightarrow \varepsilon$ } nullable (direct)
$B \rightarrow \varepsilon$ }

$S \rightarrow AB$
$\varepsilon \cdot \varepsilon$
$S \rightarrow \varepsilon$ nullable (indirect)

Imp If $\varepsilon \in L$, then ε can be removed from entire grammar except start symbol
$S \rightarrow \varepsilon$

If $\varepsilon \notin L$, then ε can be removed from entire grammar

2) $S \rightarrow aX$

$X \rightarrow aX \mid bX \mid \varepsilon$

$X \rightarrow \varepsilon$

X is nullable (direct)

S is not nullable

As $\varepsilon \notin L$, ε can be removed from entire grammar

So, after replacing nullable symbol, effect of grammar becomes

$S \rightarrow aX$	$X \rightarrow aX$	$X \rightarrow bX$
$S \rightarrow a.\varepsilon$	$X \rightarrow a.\varepsilon$	$X \rightarrow b.\varepsilon$
$S \rightarrow a$	$X \rightarrow a$	$X \rightarrow b$

$S \rightarrow aX \mid a$

$X \rightarrow aX \mid a \mid bX \mid b$

3) $S \rightarrow aXb$

$X \rightarrow aXb \mid \varepsilon$

$X \rightarrow \varepsilon$ X is nullable

$S \rightarrow aXb$ S is not nullable

$\varepsilon \notin L$, so ε can be removed from entire grammar

$S \rightarrow aXb$	$X \rightarrow aXb$
$S \rightarrow ab$	$X \rightarrow ab$

Grammar :

$S \rightarrow aXb \mid ab$

$X \rightarrow aXb \mid ab$

4) $S \to AB$

$A \to aAA \mid \varepsilon$

$B \to bBB \mid \varepsilon$

$A \to \varepsilon$ } A and B are nullable (direct)

$B \to \varepsilon$ }

$S \to AB$ nullable (indirect)

$S \to \varepsilon$

So, $\varepsilon \in L$, so, ε can be removed from entire grammar except $S \to \varepsilon$

$S \to AB$	$A \to aAA$	$B \to bBB$
$S \to \varepsilon . B$	$A \to aA . \varepsilon$	
$S \to B$	$A \to aA$	$B \to bB$
$S \to A . \varepsilon$	$A \to a . \varepsilon . \varepsilon$	
$S \to A$	$A \to a$	$B \to b$

Grammar : $S \to AB \mid B \mid A \mid \varepsilon$

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

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

5) $S \to ABA$

$A \to aA \mid \varepsilon$

$B \to bB \mid \varepsilon$

$A \to \varepsilon$ } A and B are nullable (direct)

$B \to \varepsilon$ }

$S \to ABA$

$\varepsilon . \varepsilon . \varepsilon$

$S \to \varepsilon$ nullable (indirect)

$\varepsilon \in L$, so ε can be removed from entire grammar except $S \to \varepsilon$

$S \to \varepsilon$ | $A \to aA$ | $B \to bB$

$S \to ABA$ | $A \to a$ | $B \to b$

$S \to AB$

$S \to AA$

$S \to BA$

$S \to A$

$S \to B$

Grammar :

$S \to ABA \mid AB \mid AA \mid BA \mid A \mid B \mid \varepsilon$

$A \to aA \mid a$

$B \to bB \mid b$

Non terminal = capital

classmate
Date
Page

III) Removal of unit production

Removal of ε production may generate unit productions in the grammar.

unit production ⇒ ONE NT → ONE NT

e.g. A → B

Unit productions are increasing time complexity of compiler working.

There may be a chain of unit production

So, we can remove all intermediate symbols

e.g.

A → B B → C C → D D → a

A → B → C → D → a

A → a

Otherwise, unif production effect has to be removed by replacing all rules of that symbol into the grammar.

1)

S → AB

A → a

B → c | b

C → D

D → E

E → a

B → C → D → E → a

As there is a chain of unit production

B → d

S → AB

A → a

B → a | b

2) $S \to ABA \mid BA \mid AA \mid AB \mid A \mid B$

$A \to aA \mid a$

$B \to bB \mid b$

$S \to A$
$S \to B$ } unit production

As there is no chain

Replace rules of A and B

$S \to ABA \mid BA \mid AA \mid AB \mid aA \mid a \mid bB \mid b$

$A \to aA \mid a$

$B \to bB \mid b$

Q. Simplify following grammar

$S \to ASB \mid \varepsilon$

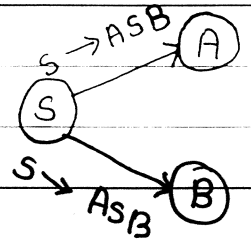
$A \to aAS \mid a$

$B \to SbS \mid A \mid bb$

i) Removal of useless symbols

Symbol = { S, A, B }

$S \to \varepsilon$
$A \to a$
$B \to bb$ } All are generating symbols (as they are terminating)



With dependancy graph, all symbols are reachable symbols

∴ There is no useless symbol in the given grammar.

ii) Removal of ε production

$S \to \varepsilon$ S is nullable (direct)

A and B are not nullable

$\varepsilon \to L$

So, ε can be removed from entire grammar except $S \to \varepsilon$ start symbol

$S \to ASB$	$A \to aAS$	$B \to SbS$
$S \to AB$	$A \to aA$	$B \to bS$
$S \to \varepsilon$	$A \to a$	$B \to Sb$
		$B \to b$
		$B \to A$
		$B \to bb$

So, after step(2) grammar becomes,

$S \to ASB \mid AB \mid \varepsilon$

$A \to aAS \mid aA \mid a$

$B \to SbS \mid bS \mid Sb \mid b \mid A \mid bb$

iii) Removal of unit production

$B \rightarrow A$ (unit production)

As there is no chain, simplified Grammar becomes

$S \rightarrow ASB \mid AB \mid \varepsilon$

$A \rightarrow aAS \mid aA \mid a$

$B \rightarrow SbS \mid bS \mid Sb \mid b \mid aAS \mid aA \mid a \mid bb$

* Normalization of Grammar

Normalized grammar is always easy to add into compiler as normalization empose some fixed constraints on production rule format.

There are multiple normal forms for the grammar which are suitable to incorporate in different compiler phases according to specific needs.

Following normal forms are into syllabus

1. CNF (Chomsky's Normal Form)
2. GNF (Griebach Normal Form)

CNF

A simplified grammar is said to be in CNF if it follows below constraints on production rules :

ONE NT ⇒ Exactly TWO NT
ONE NT ⇒ Terminal

e.g.

S → aB X

S → a ✓

S → BA ✓

S → ab X

S → aBA X

S → ε ✓

step1 : Introduce new non terminal for every terminal symbol

step2 : Restrict number of non terminals on right hand side to exactly two non terminals.

A → B1 B2 B3 B4 B5 ---- Bm

A → B1 D1

D1 → B2 B3 B4 B5 ---- Bm

D1 → B2 D2

D2 → B3 B4 B5 B6 --- Bm

D2 → B3 D3

D3 → B4 B5 B6 --- Bm

Q. $S \rightarrow aSb$

$S \rightarrow ab$

step1 : Introduce new non terminal for every terminal symbol

terminal = $\{a, b\}$

$X \rightarrow a$

$Y \rightarrow b$

After step 1, Grammar is

$S \rightarrow XSY$

$S \rightarrow XY$

$X \rightarrow a$

$Y \rightarrow b$

} In CNF form

step2 : $S \rightarrow XSY$

$S \rightarrow XP$

$P \rightarrow SY$

} In CNF form

Ans → CNF form :

$S \rightarrow XY$

$X \rightarrow a$

$Y \rightarrow b$

$S \rightarrow XP$

$P \rightarrow SY$

2) $S \to aSa$

$S \to bSb$

$S \to a$

step1 : terminal = {a, b}

$X \to a$

$Y \to b$

after step1 grammar is

$S \to XSX$

$S \to YSY$

$S \to X$

$X \to a$

$Y \to b$

} in CNF form

step2 : $S \to XSX$

$S \to XP$

$P \to SX$

} CNF form

$S \to YSY$

$S \to YQ$

$Q \to SY$

} CNF form

Ans → CNF form :

$S \to X$

$X \to a$

$Y \to b$

$S \to XP$

$P \to SX$

$S \to YQ$

$Q \to SY$

3) $S \rightarrow aBC \mid AB$

$A \rightarrow a$
$B \rightarrow a$
$C \rightarrow a$
} CNF

step1 : terminal = {a}

$X \rightarrow a$

$S \rightarrow XBC \mid AB$

$S \rightarrow XBC$
$S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow a$
$C \rightarrow a$
$X \rightarrow a$
} in CNF form

step2 : $S \rightarrow XBC$

$S \rightarrow XP$
$P \rightarrow BC$
} CNF form

Ans → CNF form :

$S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow a$
$C \rightarrow a$
$X \rightarrow a$
$S \rightarrow XP$
$P \rightarrow BC$

4) $S \to aSa$
$S \to bSb$
$S \to a \mid b$
$S \to aa \mid bb$

step1 : terminal = { a, b }
$X \to a$
$Y \to b$

$S \to XSX$
$S \to YSY$
$S \to a$
$S \to b$
$S \to XX$
$S \to YY$
$X \to a$
$Y \to b$
} in CNF Form

step2 :
$S \to XSX$
$S \to XP$ } CNF
$P \to SX$

$S \to YSY$
$S \to YR$ } CNF
$R \to SY$

Ans → CNF form is :
$S \to a$
$S \to b$
$S \to XX$
$S \to YY$
$X \to a$
$Y \to b$
$S \to XP$
$P \to SX$
$S \to YR$
$R \to SY$

3) $S \rightarrow aAbB$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

step1 : terminal = {a, b}

$X \rightarrow a$

$Y \rightarrow b$

$S \rightarrow XAYB$

$A \rightarrow XA$

$A \rightarrow a$

$B \rightarrow YB$

$B \rightarrow b$

$X \rightarrow a$

$Y \rightarrow b$

} CNF

step2 :

$S \rightarrow XAYB$

$S \rightarrow XP$

$P \rightarrow AYB$

$P \rightarrow AR$

$R \rightarrow YB$

Ans → CNF form is

$A \rightarrow XA$

$A \rightarrow a$

$B \rightarrow YB$

$B \rightarrow b$

$X \rightarrow a$

$Y \rightarrow b$

$S \rightarrow XP$

$P \rightarrow AR$

$R \rightarrow YB$

6) $S \rightarrow bA \mid aB$

$A \rightarrow bAA \mid aS \mid a$

$B \rightarrow aBB \mid bS \mid b$

step1 : terminals = $\{a, b\}$

$X \rightarrow a$

$Y \rightarrow b$

$S \rightarrow YA$
$S \rightarrow XB$
$A \rightarrow XS$
$A \rightarrow a$
$B \rightarrow YS$
$B \rightarrow b$
} CNF

$A \rightarrow bXX$ ~~YAA~~

$B \rightarrow aYY$ ~~XBB~~

step2 : $A \rightarrow bXX$ ~~YAA~~ $B \rightarrow XBB$

$A \rightarrow bYR$ $B \rightarrow XQ$

$R \rightarrow AA$ $Q \rightarrow BB$

CNF is : $S \rightarrow YA$

$S \rightarrow XB$

$A \rightarrow XS$

$A \rightarrow a$

$B \rightarrow YS$

$B \rightarrow b$

$A \rightarrow YR$

$R \rightarrow AA$

$B \rightarrow XQ$

$Q \rightarrow BB$

7) $S \rightarrow Aba$

$S \rightarrow aab$

$B \rightarrow AC$

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

A is non generating

B, C are non generating / Non reachable

so, remove them

terminal $= \{a, b\}$

$X \rightarrow a$

$Y \rightarrow b$

$S \rightarrow aab$

$S \rightarrow XXY$

$\overline{X}$

* Parse Tree / Symbol Tree / Deviation Tree

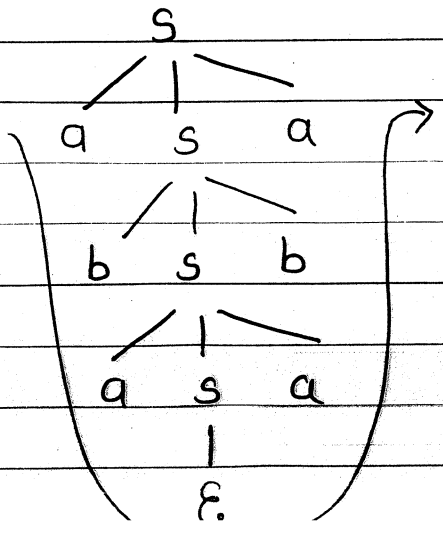
When grammar is imparted into compiler, tree data structure is used.
With the help of production rules, we can derive all words belonging to language
This deviation of a word can be represented in tree format known as deviation tree.
Grammar plays a very vital role in syntax and semantic phases of compiler.
So, compiler uses parse tree or symbol tree to resolve issues of syntax and semantic analysis phase.

* start symbol - Root node
* other non-terminal - Intermediate nodes
* Terminal - leaf nodes

1) $S \to aSa$
$S \to bSb$
$S \to \varepsilon$

$S \to a\underline{S}a$
$S \to ab\underline{S}ba$
$S \to aba\underline{S}aba$
$S \to ababa$

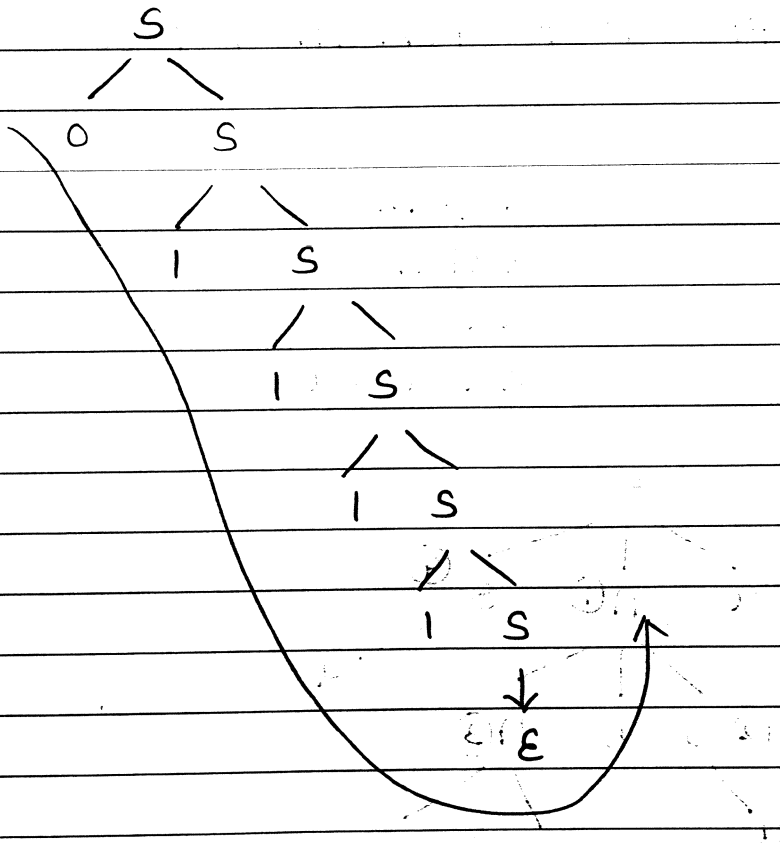
~~$S \to a$~~
$S \to bSb$
$S \to aSa$
$S \to \varepsilon$



2) $S \rightarrow 0S \mid 1S \mid \varepsilon$

$S \rightarrow 0S$
$S \rightarrow 1S$
$S \rightarrow \varepsilon$

$S \rightarrow 0S$
$S \rightarrow 01S$ $(S \rightarrow 1S)$
$S \rightarrow 011S$
$S \rightarrow 0111S$
$S \rightarrow 01111S$ $(S \rightarrow \varepsilon)$
$S \rightarrow 01111$



- Left most deviation tree :
While deriving the word, if we expand left most non terminal then it is called as left most deviation tree.

- Right most deviation tree :
While deriving the word, if we expand right most non terminal then it is called as right most deviation tree.

1) Design left and right deviation for a word aabbaa on following grammar

$S \rightarrow aAS$

$A \rightarrow SbA$

$S \rightarrow a$

$A \rightarrow SS$

$A \rightarrow ba$

* left most deviation tree

aabbaa

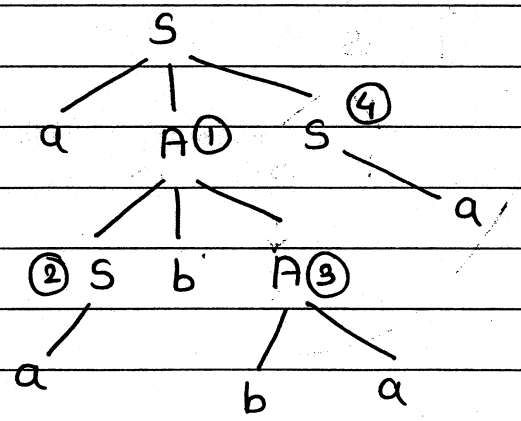
$S \rightarrow a\underline{A}S$

$a\underline{S}bAS$

$aab\underline{A}S$

$aabba\underline{S}$

$aabbaa$



* Right most deviation tree

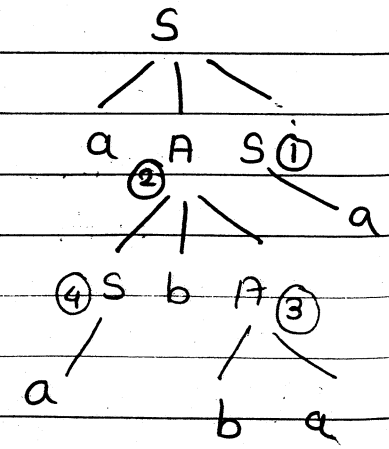
$S \rightarrow aA\underline{S}$

$a\underline{A}a$

$asb\underline{A}a$

~~$asbaa$~~ $as\underline{S}bbaa$

~~$aaASbaa$~~ $aabbaa$



2) abaabb

$S \rightarrow XbaaX$

$X \rightarrow Xa \mid Xb \mid \varepsilon$

$S \rightarrow aX$

* left

abaabb

$S \rightarrow aXbaaX$

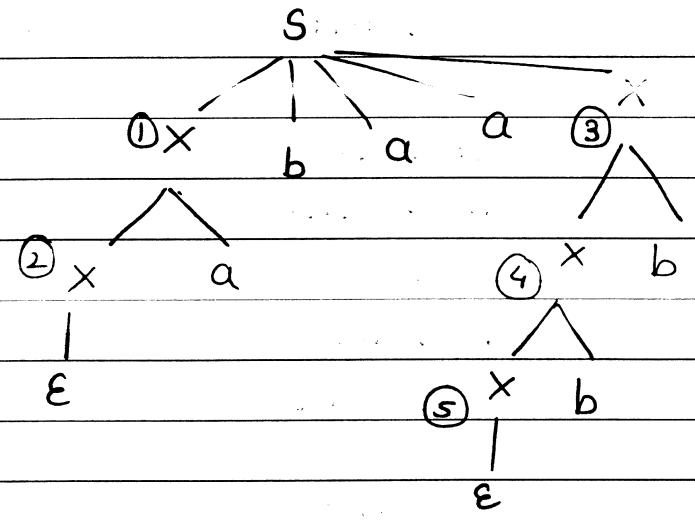
$\rightarrow aXabaaX$

$\rightarrow abaaX$

$\rightarrow abaaXb$

$\rightarrow abaaXbb$

$\rightarrow abaabb$



* Right

abaabb

$S \rightarrow XbaaX$

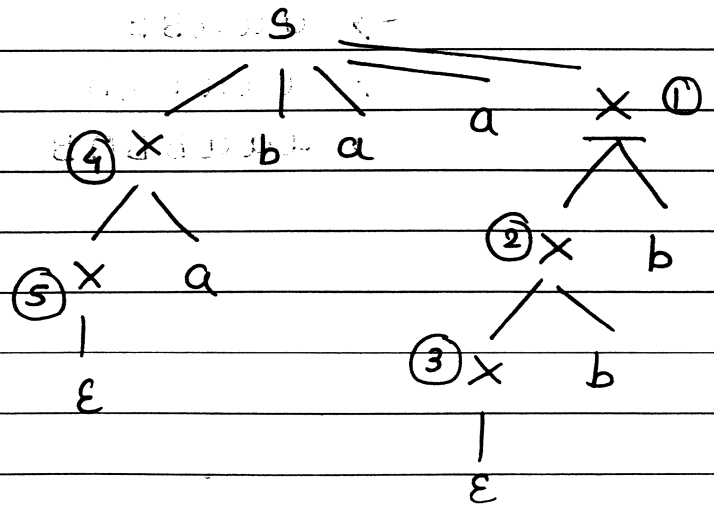
$XbaaXb$

$XbaaXbb$

$Xbaabb$

$Xabaabb$

$abaabb$



Imp 3) aaabbabbba

$S \leftrightarrow aB \mid bA$

$A \leftrightarrow a \mid aS \mid bAA$

$B \rightarrow b \mid bS \mid aBB$

$S \rightarrow aB$
$S \rightarrow bA$
$A \rightarrow a$
$A \rightarrow aS$
$A \rightarrow bAA$
$B \rightarrow b$
$B \rightarrow bS$
$B \rightarrow aBB$

Right — aaabbabbba

$S \rightarrow bA$
$\rightarrow bbAA$
$\rightarrow bbbAAa$
$\rightarrow bbbAaSa$
$\rightarrow bbbAaaBa$
$\rightarrow aaba$

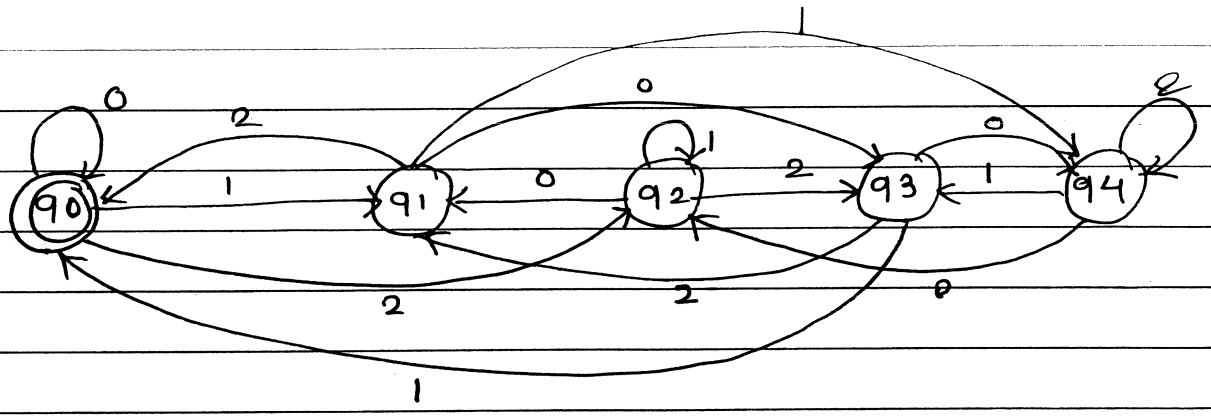
left

$S \rightarrow aB$
$\rightarrow aaBB$
$\rightarrow aaaBB$
$\rightarrow aaabSB$
$\rightarrow aaabbABB$

Divisible by 5 w.r.t. ternary system

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

		decimal
q_0 = can see remainder	0	= 0
q_1	1	= 1
q_2	2	= 2
q_3	3	= 10
q_4	4	= 11



100	101	102	110	111	112
931	931	931	931	931	931
↓					
5)9	5)10	5)11	5)12	5)13	5)14
4	0	1	2	3	4

Handrun 1 2 0 2 1 0

243 81 27 9 3 1

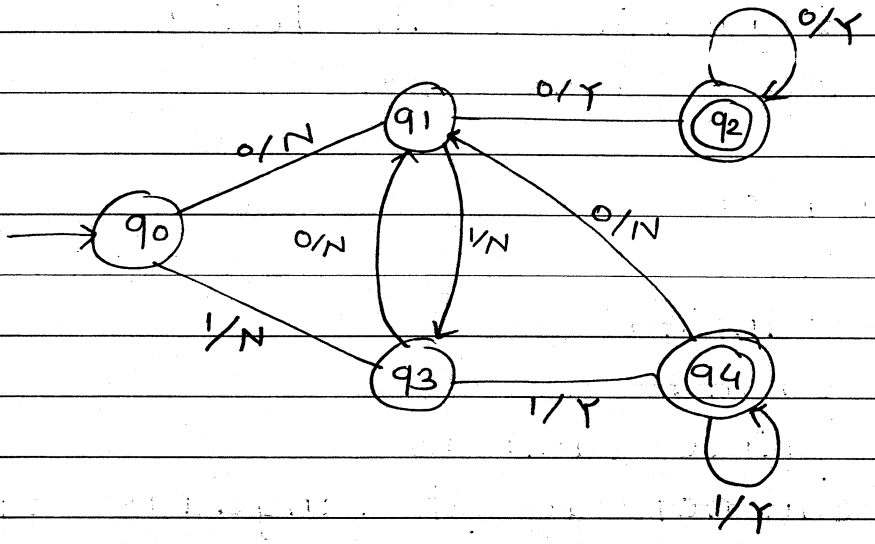
$$\begin{array}{r} 243 \\ +\ 162 \\ +\ 18 \\ \hline 3 \\ \hline 426 \end{array}$$

$$\begin{array}{r|l} 5 & 426 \\ \hline & 1 \end{array}$$

ending with 00 or 11

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

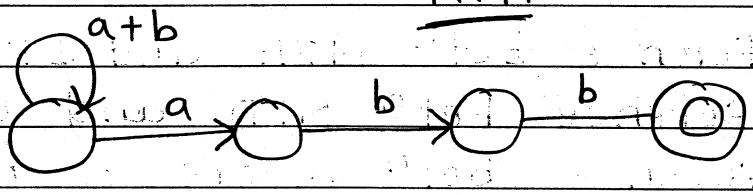
$L = \{00, 11, \ldots\}$



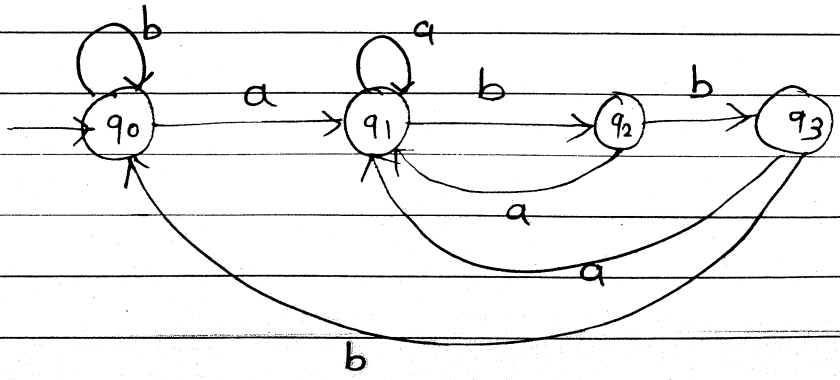
$(a+b)^*$ abb

* loop
+ fork
• series

NFA



DFA ending with abb



left most cutting technique

Theory questions

? Basic machine

? block diagram of FA — tape i/p reader

? FA applications

moore & mealy appln

→ lexical (applications) Analysis
→ Syntax analysis
→ Sequential circuit design
→ Editor generation

? limitation

moore mealy limitations

→ unable to perform arithmatic opn
→ counting
→ palindrome
→ prime numbers
→ can not accept language other than RE
→ can not accept context free language

? difference betn NFA and DFA

? difference betn NFA and ε-NFA

? difference betn moore & mealy

? defn of RE (6-7 lines) ϕ is RE, ε, R1+R2, ---

Theorem

If L_1 and L_2 are regular languages, then their union $L_1 \cup L_2$ is also regular language

In other words, REs are closed under union

Proof : Let us consider two regular languages L_1 and L_2

This means that there exists regular expressions R_1 and R_2 with respect to L_1 and L_2

So $L_1 = L(R_1)$ and $L_2 = L(R_2)$

From the defⁿ of RE,

Therefore, $L(R_1 + R_2)$ is also Regular language

Since the regular expression '$R_1 + R_2$' denotes all the strings that are denoted by R_1 or R_2

we can say,

$$L(R_1 + R_2) = L(R_1) \cup L(R_2)$$

Hence,

$L_1 \cup L_2$ is also RL

Push Down Automata

Languages accepted by finite automata are known as Regular language and can be expressed with regular expression and regular grammar (RLG and LLG).

L(FA) = RL ⇒ RE and RG (RLG & LLG)

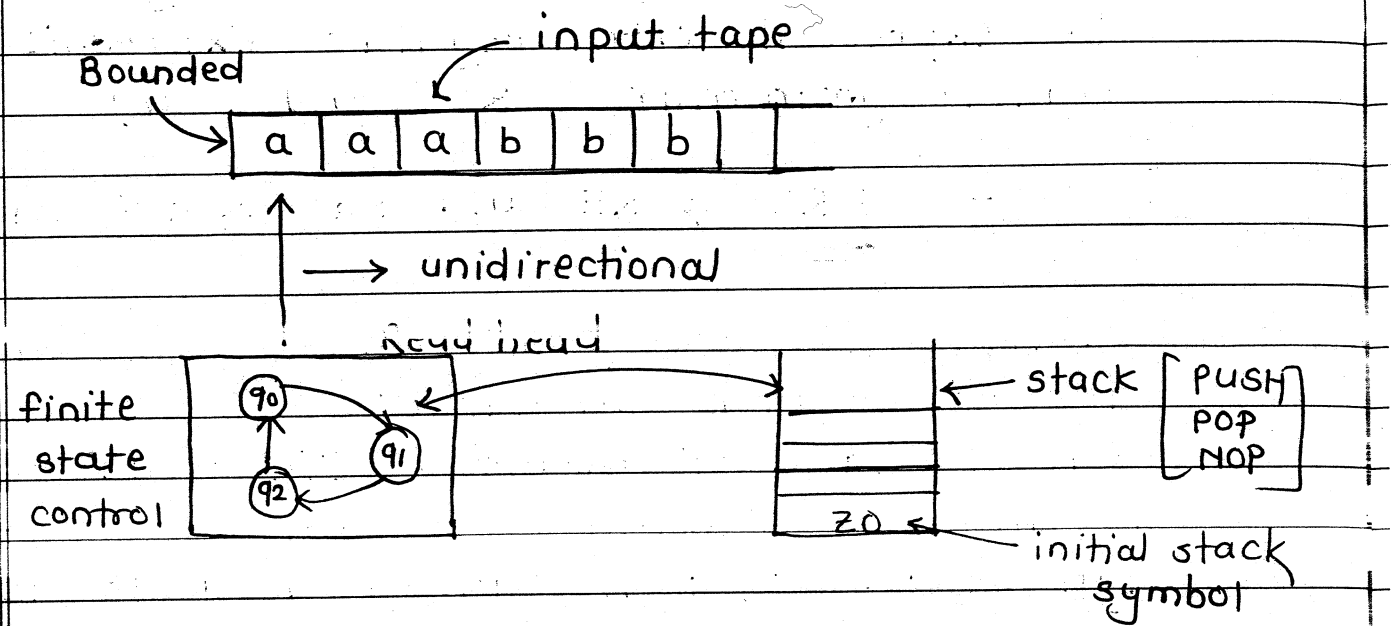
Pumping lemma proves that some language are not regular in nature.
Due to lack of memory finite automata is unable to handle certain problems or languages. So, PDA can be used to accept context free languages.

L(PDA) = CFL + RL

In PDA, additional memory is introduced in the form of stack.
So, informally PDA can be viewed as,

PDA = FA + Additional memory in the form of stack [LIFO]

Block diagram of PDA



Technically PDA can be defined as a 7 tuple machine :

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

where,

- Q : Finite number of states
- Σ : finite number of inputs
- Γ : finite number of stack symbols
- q_0 : initial state
- Z_0 : initial stack symbol
- F : final state
- δ : state transition function

$$(P.S \times i/p \times \underset{top}{stack}) \Rightarrow (N.S\ ,\ stack\ op)$$

$$Q \times \Sigma \times F \Rightarrow Q, \Gamma^*$$

1. Final stack acceptance

$$L(PDA) = \{ \omega \mid \delta(q_0, \omega, z_0) \overset{*}{\underset{M}{\Rightarrow}} (q_F, \varepsilon, \alpha) \}$$

q_F: final state; ε: string over; α: stack is ignored

2. Empty stack acceptance

$$L(PDA) = \{ \omega \mid \delta(q_0, \omega, z_0) \overset{*}{\underset{M}{\Rightarrow}} (q_i, \varepsilon, z_0) \}$$

q_i: ignored; ε: string over; z_0: stack empty

When we need to solve problems of regular language with the help of PDA, we can use final stack acceptance and for solving CFL problems we need to use empty stack acceptance

PDA representation

For solving problem through PDA, we can use following representations:

1) δ representation
2) Tabular representation / state Table
3) State diagram representation
4) Pictorial representation

1) Design PDA to accept following language
$L = \{ a^n b^n \mid n \geq 1 \}$
show the acceptance by empty stack and final state

* δ representation

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

$\Sigma = \{a, b, \varepsilon\}$ $\Gamma = \{a, z_0\}$

q_1 = can see all b's and pop respective a off the stack

q_0 = SST can see all a's and push it on stack till first b
For first b, it will pop off a off the stack and transfer control to q_1

$Q = \{q_0, q_1\}$ $\Sigma = \{a, b, \varepsilon\}$ $\Gamma = \{a, z_0\}$

1) $\delta(q_0, a, z_0) \rightarrow (q_0, az_0)$ → Stay in q_0 and push a

2) $\delta(q_0, a, a) \rightarrow (q_0, aa)$ → stay in q_0 and push a

3) $\delta(q_0, b, z_0)$ → Reject → string can't start with b

4) $\delta(q_0, b, a) \rightarrow (q_1, \varepsilon)$ → Transit to q_1 and pop a

1) Design PDA to accept following language
$L = \{ a^n b^n \mid n \geq 1 \}$
show the acceptance by empty stack and final state

* δ representation

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

$\Sigma = \{a, b, \varepsilon\}$ $\Gamma = \{a, z_0\}$

q_1 = can see all b's and pop respective a off the stack

q_0 = SST can see all a's and push it on stack till first b
For first b, it will pop off a off the stack and transfer control to q_1

$Q = \{q_0, q_1\}$ $\Sigma = \{a, b, \varepsilon\}$ $\Gamma = \{a, z_0\}$

1) $\delta(q_0, a, z_0) \rightarrow (q_0, az_0)$ → Stay in q_0 and push a

2) $\delta(q_0, a, a) \rightarrow (q_0, aa)$ → stay in q_0 and push a

3) $\delta(q_0, b, z_0)$ → Reject → string can't start with b

4) $\delta(q_0, b, a) \rightarrow (q_1, \varepsilon)$ → Transit to q_1 and pop a

5) $\delta(q_1, a, Z_0) \rightarrow$ Reject $\rightarrow$ a can't appear after b

6) $\delta(q_1, a, a) \rightarrow$ Reject $\rightarrow$ a can't appear after b

7) $\delta(q_1, b, Z_0) \rightarrow$ Reject $\rightarrow$ No. of b's are more than no. of a's

8) $\delta(q_1, b, a) \rightarrow (q_1, \varepsilon) \rightarrow$ stay in q_1 and pop a

9) $\delta(q_0, \varepsilon, Z_0) \rightarrow$ Reject $\rightarrow \varepsilon \notin L$

10) $\delta(q_0, \varepsilon, a) \rightarrow$ Reject $\rightarrow$ There are no b's

11) $\delta(q_1, \varepsilon, Z_0) \rightarrow$ Accept / (q_2, Z_0) $\rightarrow$ Transit to final state & NOP with stack

12) $\delta(q_1, \varepsilon, a) \rightarrow$ Reject $\rightarrow$ No. of a's are more than b's

While solving PDA through δ transitions we will always represent δ transitions which lead to acceptance.
There is no need to show rejection transition.

To convert above empty stack acceptance into final state acceptance, we need to add one state as final state and empty stack acceptance condition will lead to that specific final state
So, δ transition no. 11 will lead to final state
$\delta(q_1, \varepsilon, Z_0) \rightarrow$ Accept / $(q_2, Z_0) \rightarrow$ Transit to final state & NOP on stack

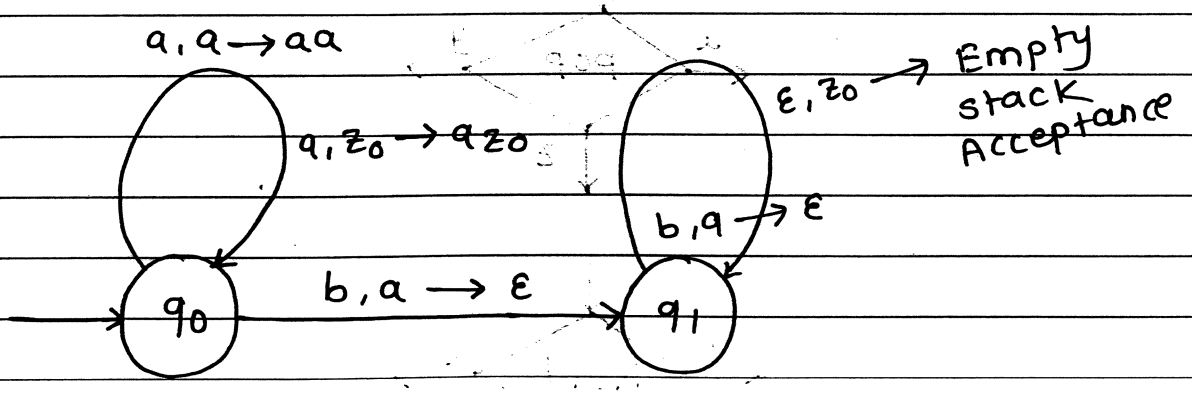
$q_2 \Rightarrow$ final state

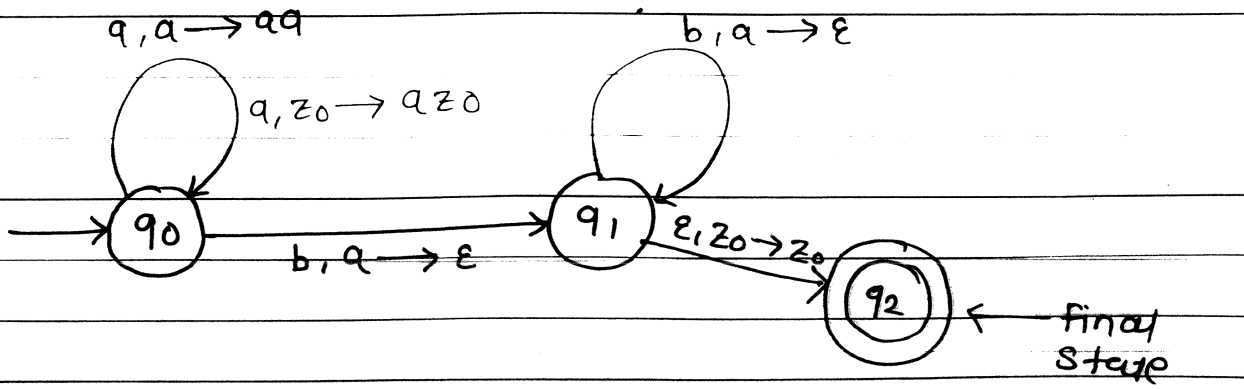
* State table representation

i/p / stack / state	a		b		ε	
	a	z_0	a	z_0	a	z_0
q_0	q_0, aa	q_0, az_0	q_1, ε	R	R	R
q_1	R	R	q_1, ε	R	R	q_2, z_0
q_2	final state					

To convert it into final state acceptance representation, one transition will be changed

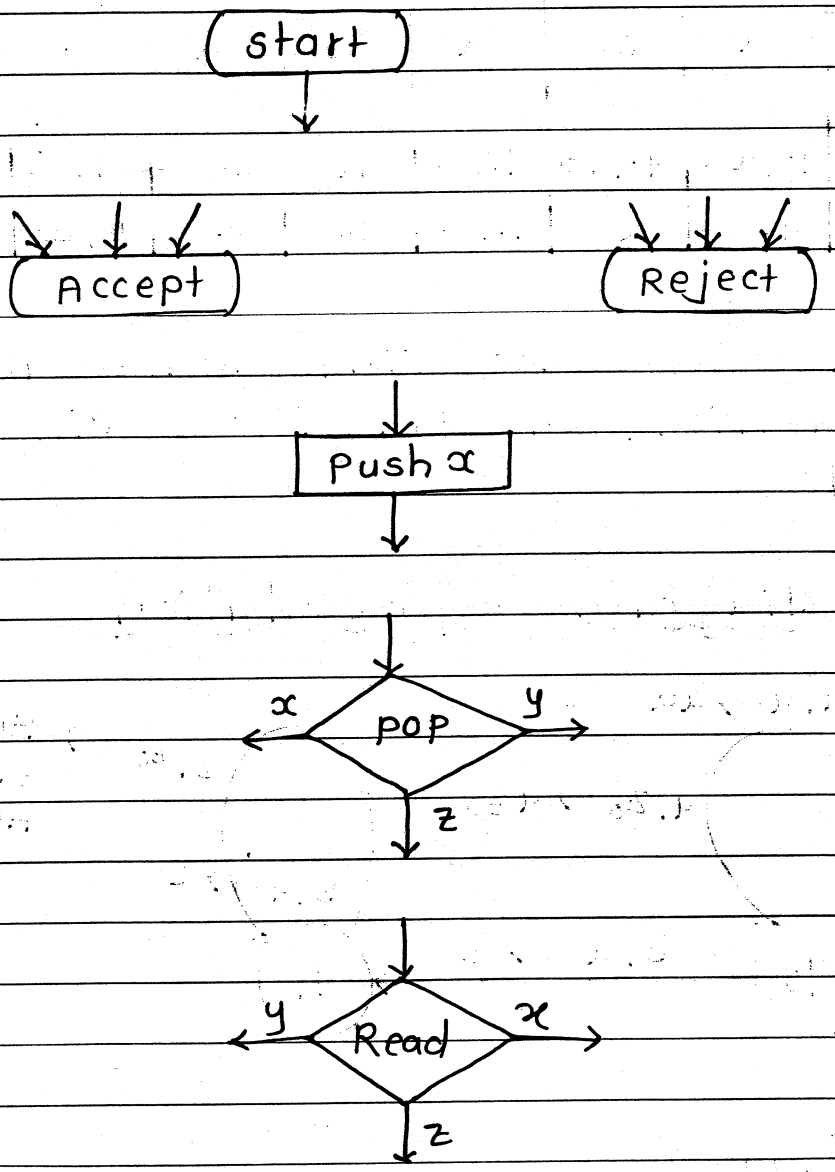
* State diagram representation

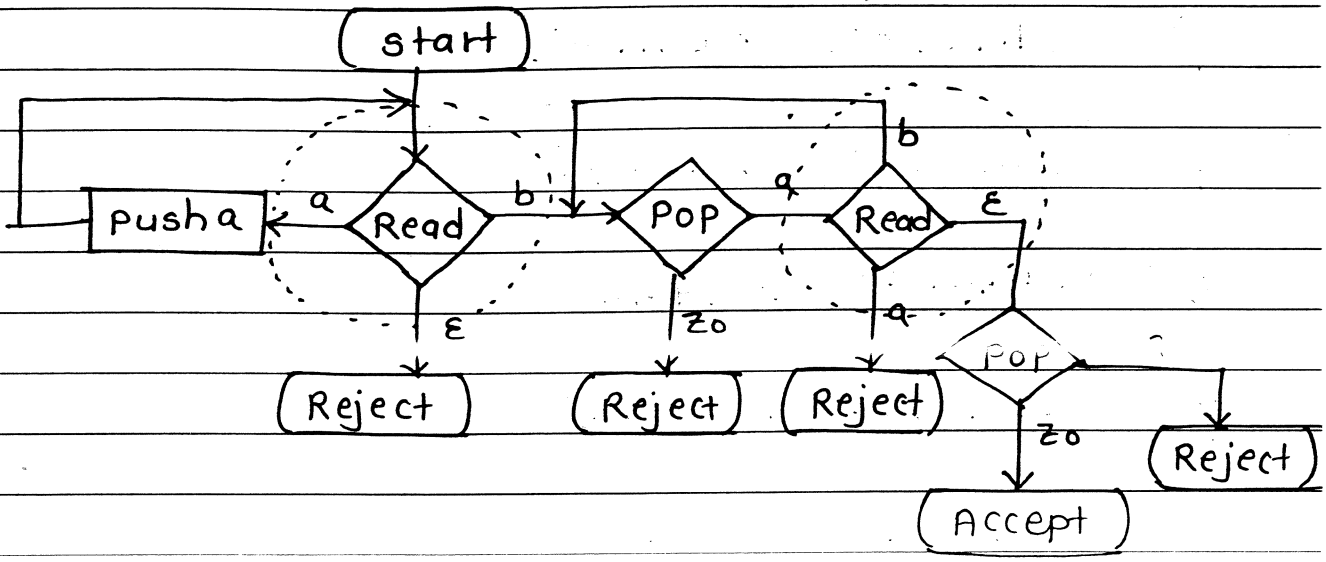




Pictorial representation of PDA

In this representation, some symbols like flowcharts are used to represent PDA





(2) $L = \{ x^n y^m z^n \mid m \geq 1, n \geq 0 \}$

$L = \{ y, xyz, xyyz, xxyzz, yyy, yyyy, xyyyz, yyyy \}$

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

$\Sigma = \{ x, y, z, \varepsilon \}$

$\Gamma = \{ x, z_0 \}$

1) $\delta(q_0, x, z_0) \longrightarrow (q_0, xz_0)$

2) $\delta(q_0, x, x) \longrightarrow (q_0, xx)$

3) $\delta(q_0, y, x) \longrightarrow (q_1, x)$

4) $\delta(q_0, y, z_0) \longrightarrow (q_1, z_0) \longrightarrow$ string starting with y

5) $\delta(q_1, y, x) \rightarrow (q_1, x)$ we are skipping y so we can see only x

6) $\delta(q_1, y, z_0) \rightarrow (q_1, z_0)$

7) $\delta(q_1, \varepsilon, z_0) \rightarrow$ Accept

8) $\delta(q_1, z, x) \rightarrow (q_2, \varepsilon)$

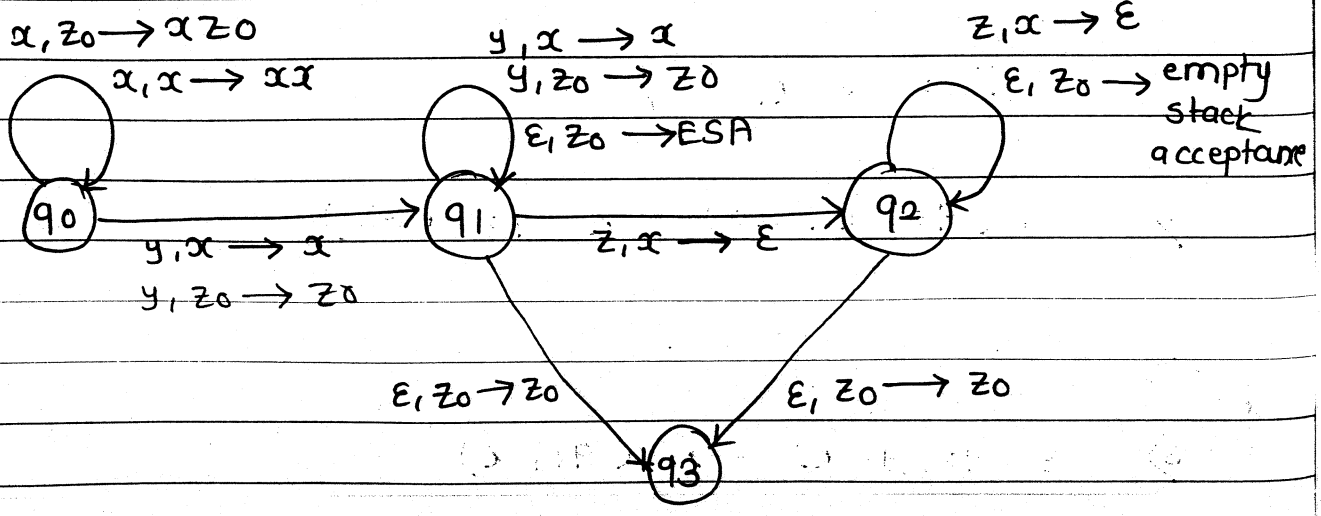
9) $\delta(q_2, z, z) \rightarrow (q_2, \varepsilon)$

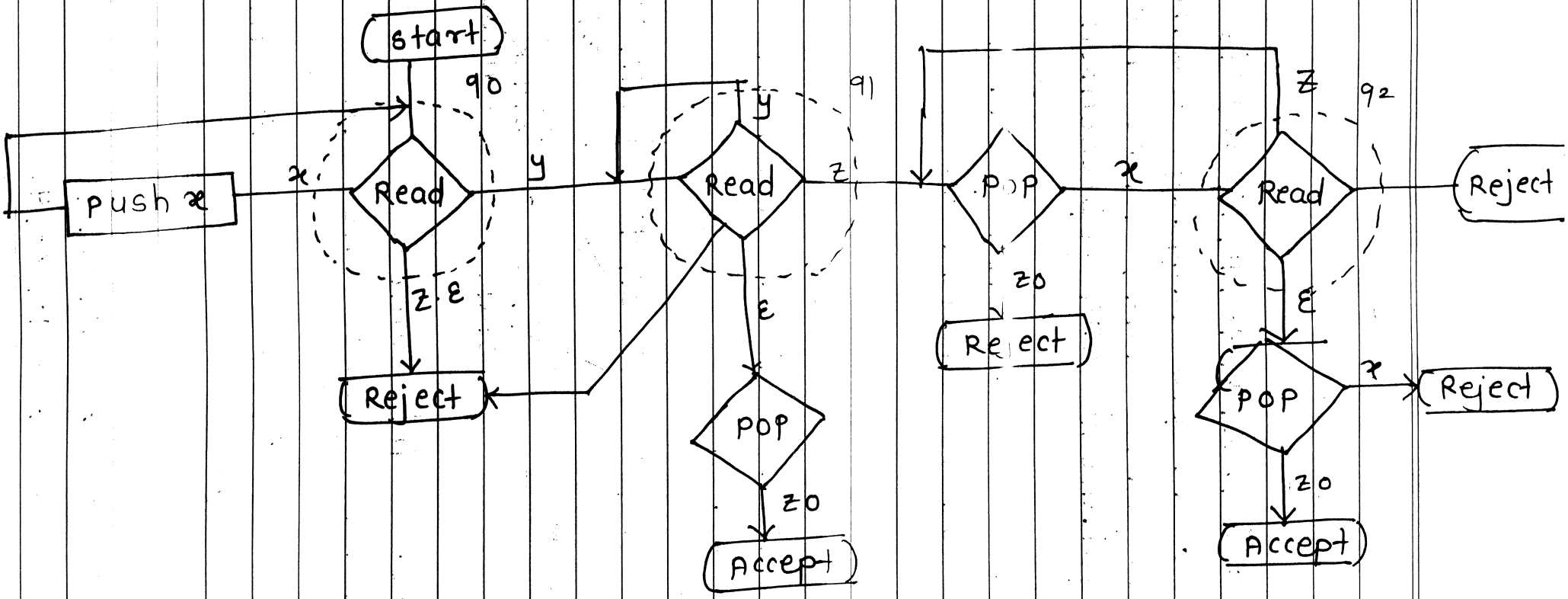
10) $\delta(q_2, \varepsilon, z_0) \rightarrow$ Accept

state table representation

	x	x z0	y x	y z0	z x	z z0	ε x	ε z0
q_0	xx	xz0	x	z0	z0 R	R	R	R
q_1	R	R	x	z0	ε	R	R	A
q_2	R	R	R	R	ε	R	R	A

state diagram representation




start
q0
Push x
x
Read
y
Z ε
Reject
y
q1
Read
z
ε
POP
z0
Accept
POP
x
z0
Reject
Z
q2
Read
Reject
ε
POP
x
Reject
z0
Accept

PDA to grammar

Let $M = (Q, \Sigma, \delta, q_0, F, \Gamma, z_0)$ is a PDA which accepts language L with empty stack acceptance then there always exists equivalent CFG represented as :

$G = (N, T, P, S)$ with production rules P to accept same language.

Rules for conversion

Non terminals of the resulting grammar can be given by

$$NT = \{ S, \cup [q_i, z_i, q_j] \mid (q_i, q_j) \in Q, \; z_i \in \Gamma \}$$

state — stack symbol

Rule 1 For start symbol

start symbol rule can be given as

$$S \longrightarrow [q_0, z_0, q_i] \quad \text{For all } q_i \in Q$$

e.g.

$$Q = \{q_0, q_1\}$$

$$S \longrightarrow [q_0, z_0, q_0]$$

$$S \longrightarrow [q_0, z_0, q_1]$$

$\delta(q_i, C, B) \rightarrow (q_j, C_1 C_2)$ — top of stack is replaced by

q_i: current state; C: receiving i/p; B: top of stack; q_j: reaching here

CLASSMATE Date Page

Rule 2 For following type of PDA triplets:

$$\delta(q_i, a, B) \rightarrow (q_j, \varepsilon)$$

$$[q_i, B, q_j] \rightarrow a$$

e.g.

$$\delta(q_1, a, B) \rightarrow (q_2, \varepsilon)$$

$$[q_1, B, q_2] \rightarrow a$$

Rule 3 $\delta(q_i, a, B) \rightarrow (q_j, C)$

$$[q_i, B, q_K] \rightarrow a\,[q_j, C, q_K] \quad \text{for all } q_K \in Q$$

e.g. $Q = \{q_0, q_1\}$

$$\delta(q_0, a, B) \rightarrow (q_1, C)$$

$$[q_0, B, q_0] \rightarrow a\,[q_1, C, q_0]$$

$$[q_0, B, q_1] \rightarrow a\,[q_1, C, q_1]$$

Rule 4 $\delta(q_i, a, B) \rightarrow (q_j, C_1 C_2)$

$$[q_i, B, P_1] \rightarrow a\,[q_j, C_1, P_2]\,[P_2, C_2, P_1]$$

$Q = \{q_0, q_1\}$

P1	P2	$\delta(q_0, a, B) \rightarrow (q_1, C_1 C_2)$
q_0	q_0	$[q_0, B, q_0] \rightarrow a\,[q_1, C_1, q_0]\,[q_0, C_2, q_0]$
q_0	q_1	$[q_0, B, q_0] \rightarrow a\,[q_1, C_1, q_1]\,[q_1, C_2, q_0]$
q_1	q_0	$[q_0, B, q_1] \rightarrow a\,[q_1, C_1, q_0]\,[q_0, C_2, q_1]$
q_1	q_1	$[q_0, B, q_1] \rightarrow a\,[q_1, C_1, q_1]\,[q_1, C_2, q_1]$

Nimp

$M = (\underbrace{\{q\}}_{Q}\ \underbrace{\{i, e\}}_{\Sigma}\ \underbrace{\{x, z\}}_{\Gamma}, \underbrace{\delta}_{\delta}, \underbrace{q}_{q_0}, \underbrace{z_0}_{z_0}, \underbrace{\phi}_{F})$

$\delta(q, i, z) \longrightarrow (q, xz)$ → rule 4

$\left.\begin{array}{l}\delta(q, e, x) \longrightarrow (q, \varepsilon) \\ \delta(q, \varepsilon, z) \longrightarrow (q, \varepsilon)\end{array}\right\}$ rule 2

step1 :

$S \longrightarrow [q, z, q] \quad \ldots (1)$

$Q = \{q\}$

step2 : $\delta(q, e, x) \longrightarrow (q, \varepsilon)$

$[q, x, q] \longrightarrow e \quad \ldots (2)$

$\delta(q, \varepsilon, z) \longrightarrow (q, \varepsilon)$

$[q, z, q] \longrightarrow \varepsilon \quad \ldots (3)$

step3 $\delta(q, i, z) \longrightarrow (q, xz)$

$[q, z, q] \longrightarrow i[q, x, q][q, z, q]$

P0 P1
q q
AS $Q = \{q\}$

step4 Rename

$Q = \{q\}$

$\Gamma = \{x, z\}$

$[q, z, q] \longrightarrow A$

$[q, x, q] \longrightarrow B$

steps

$S \rightarrow A$

$B \rightarrow e$

$A \rightarrow \varepsilon$

$A \rightarrow iBA$

$S \rightarrow A$

$S \rightarrow \varepsilon$

$S \rightarrow A$

iBA

ien

ie

$S \rightarrow A$

iBA

ieA

$ieiBA$

$ieie$

$L = \{ \varepsilon, ie, ieie, \dots \}$

$(ie)^*$

last qp
Imp

$M = (\underbrace{\{q_0, q_1\}}_{Q}, \underbrace{\{0,1\}}_{\Sigma}, \underbrace{\{x, z_0\}}_{\Gamma}, \underbrace{q_0}_{q_0}, \underbrace{z_0}_{z_0}, \underbrace{\delta}_{\delta}, \underbrace{\phi}_{F})$

$\delta(q_0, 0, z_0) \longrightarrow (q_0, x z_0)$

$\delta(q_0, 0, x) \longrightarrow (q_0, xx)$

$\delta(q_0, 1, x) \longrightarrow (q_1, \varepsilon)$

$\delta(q_1, 1, x) \longrightarrow (q_1, \varepsilon)$

$\delta(q_1, \varepsilon, x) \longrightarrow (q_1, \varepsilon)$

$\delta(q_1, \varepsilon, z_0) \longrightarrow (q_1, \varepsilon)$

For all $q_i \in Q$

$Q = \{q_0, q_1\}$

$S \longrightarrow [q_0, z_0, q_0]$ ---(1)

$S \longrightarrow [q_0, z_0, q_1]$ ---(2)

$\delta(q_0, 1, x) \longrightarrow (q_1, \varepsilon)$

$[q_0, x, q_1] \longrightarrow 1$ ---(3)

$\delta(q_1, 1, x) \longrightarrow (q_1, \varepsilon)$

$[q_1, x, q_1] \longrightarrow 1$ ---(4)

$\delta(q_1, \varepsilon, x) \longrightarrow (q_1, \varepsilon)$

$[q_1, x, q_1] \longrightarrow \varepsilon$ ---(5)

$\delta(q_1, \varepsilon, z_0) \longrightarrow (q_1, \varepsilon)$

$[q_1, z_0, q_1] \longrightarrow \varepsilon$ ---(6)

$\delta(q_0, 0, z_0) \longrightarrow (q_0, x z_0)$

$Q = \{q_0, q_1\}$

P1 P2

$q_0\ q_0$ $[q_0, z_0, q_0] \rightarrow 0[q_0, x, q_0][q_0, z_0 q_0]$ ---(7)

$q_0\ q_1$ $[q_0, z_0, q_0] \rightarrow 0[q_0, x, q_1][q_1, z_0, q_0]$ ---(8)

$q_1\ q_0$ $[q_0, z_0, q_1] \rightarrow 0[q_0, x, q_0][q_0, z_0, q_1]$ ---(9)

$q_1\ q_1$ $[q_0, z_0, q_1] \rightarrow 0[q_0, x, q_1][q_1, z_0, q_1]$ --(10)

$\delta(q_0, 0, x) \longrightarrow (q_0, x x)$

P1 P2

$q_0\ q_0$ $[q_0, x, q_0] \rightarrow 0[q_0, x, q_0][q_0, x, q_0]$ --(11)

$q_0\ q_1$ $[q_0, x, q_0] \rightarrow 0[q_0, x, q_1][q_1, x, q_0]$ --(12)

$q_1\ q_0$ $[q_0, x, q_1] \rightarrow 0[q_0, x, q_0][q_0, x, q_1]$ ---(13)

$q_1\ q_1$ $[q_0, x, q_1] \rightarrow 0[q_0, x, q_1][q_1, x, q_1]$ ---(14)

Rename : $Q = \{q_0, q_1\}$ $\Gamma = \{x, z_0\}$

$[q_0, z_0, q_0] \rightarrow A$ $[q_0, x, q_0] \rightarrow E$

$[q_0, z_0, q_1] \rightarrow B$ $[q_0, x, q_1] \rightarrow F$

$[q_1, z_0, q_0] \rightarrow C$ $[q_1, x, q_0] \rightarrow G$

$[q_1, z_0, q_1] \rightarrow D$ $[q_1, x, q_1] \rightarrow H$

Date
Page

$S \rightarrow A$

$S \rightarrow B$

$F \rightarrow 1$

$H \rightarrow 1$

$H \rightarrow \varepsilon$

$D \rightarrow \varepsilon$

$A \rightarrow 0EA$

$A \rightarrow 0FC$

$B \rightarrow 0EB$

$B \rightarrow 0FD$

$E \rightarrow 0EE$

$E \rightarrow 0FG$

$F \rightarrow 0EF$

$F \rightarrow 0FH$

~~S~~ ~~A~~ ~~B~~ ~~C~~ ~~D~~ All are reachable
~~E~~ ~~F~~ ~~G~~ ~~H~~

S ~~A~~ (B) ~~C~~ (D)
~~E~~ (F) ~~G~~ (H)

Generating that are directly Terminate
F H D
B because B contains F ($B \rightarrow 0FD$)

[A, C, E, G are useless, remove] After removal of useless symbols (non-generating and non-reachable) grammar becomes

$S \rightarrow B$

$F \rightarrow 1$

$H \rightarrow 1 \mid \varepsilon$

$D \rightarrow \varepsilon$

$B \rightarrow 0FD$

$F \rightarrow 0FH$

$\Delta = \varepsilon$

classmate
Date
Page

$M = (\{q_0, q_1\}, \{a, b\}, \{z_0, z\}, \delta, q_0, z_0, \phi)$

$\delta(q_0, b, z_0) \longrightarrow (q_0, z z_0)$

$\delta(q_0, b, z) \longrightarrow (q_0, z z)$

$\delta(q_0, \Delta, z_0) \longrightarrow (q_0, \Delta)$

$\delta(q_0, a, z) \longrightarrow (q_1, z)$

$\delta(q_1, b, z) \longrightarrow (q_1, \Delta)$

$\delta(q_1, a, z_0) \longrightarrow (q_0, z_0)$

$Q = \{q_0, q_1\}$

$S \longrightarrow [q_0, z_0, q_0]$ --- (1)

$S \longrightarrow [q_0, z_0, q_1]$ --- (2)

$\delta(q_0, b, z_0) \longrightarrow (q_0, z z_0)$

$Q = \{q_0, q_1\}$

P1	P2	
q_0	q_0	$[q_0, z_0, q_0] \rightarrow b\,[q_0, z, q_0]\,[q_0, z_0, q_0]$ -- (3)
q_0	q_1	$[q_0, z_0, q_0] \rightarrow b\,[q_0, z, q_1]\,[q_1, z_0, q_0]$ -- (4)
q_1	q_0	$[q_0, z_0, q_1] \rightarrow b\,[q_0, z, q_0]\,[q_0, z_0, q_1]$ -- (5)
q_1	q_1	$[q_0, z_0, q_1] \rightarrow b\,[q_0, z, q_1]\,[q_1, z_0, q_1]$ -- (6)

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

P1	P2		
q_0	q_0	$[q_0, Z, q_0] \longrightarrow b$	$[q_0, Z, q_0]\ [q_0, Z, q_0]$ --- (7)
q_0	q_1	$[q_0, Z, q_0] \longrightarrow b$	$[q_0, Z, q_1]\ [q_1, Z, q_0]$ --- (8)
q_1	q_0	$[q_0, Z, q_1] \longrightarrow b$	$[q_0, Z, q_0]\ [q_0, Z, q_1]$ --- (9)
q_1	q_1	$[q_0, Z, q_1] \longrightarrow b$	$[q_0, Z, q_1]\ [q_0, Z, q_1]$ --- (10)

$\delta(q_0, \Delta, Z_0) \longrightarrow (q_0, \Delta)$

$[q_0, Z_0, q_0] \longrightarrow \Delta$ --- (11)

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

$[q_0, Z, q_0] \longrightarrow a\,[q_1, Z, q_0]$ --- (12)

$[q_0, Z, q_1] \longrightarrow a\,[q_1, Z, q_1]$ --- (13)

$\delta(q_1, b, Z) \longrightarrow (q_1, \Delta)$

$[q_1, Z, q_1] \longrightarrow b$ --- (14)

$\delta(q_1, 0, Z_0) \longrightarrow (q_0, Z_0)$

$[q_1, Z_0, q_0] \longrightarrow 0\,[q_0, Z_0 q_0]$ --- (15)

$[q_1, Z_0, q_1] \longrightarrow 0\,[q_0, Z_0 q_1]$ --- (16)

Rename

$Q = \{q_0, q_1\}$ $\quad \Gamma = \{Z_0, X\}$

$[q_0, Z_0, q_0] \Rightarrow A$ $\quad [q_0, X, q_0] \rightarrow E$

$[q_0, Z_0, q_1] \Rightarrow B$ $\quad [q_0, X, q_1] \rightarrow F$

$[q_1, Z_0, q_0] \rightarrow C$ $\quad [q_1, X, q_0] \rightarrow G$

$[q_1, Z_0, q_1] \rightarrow D$ $\quad [q_1, X, q_1] \rightarrow H$

$S \rightarrow A$

$S \rightarrow B$

$A \rightarrow bBEA$

$A \rightarrow bFC$

$B \rightarrow bEB$

$D \rightarrow bFD$

$E \rightarrow bEE$

$E \rightarrow bFG$

$F \rightarrow bEF$

$F \rightarrow bFF$

3) $L = \{ x^n y^n \mid n \geq 0 \}$

$L = \{ \varepsilon, xy, xxyy, xxxyyy, \cdots \}$

$\Sigma = \{ x, y, \varepsilon \}$

$\Gamma = \{ x, z_0 \}$

δ transitions

1. $\delta(q_0, x, z_0) \longrightarrow (q_0, x z_0)$

2. $\delta(q_0, x, x) \longrightarrow (q_0, xx)$

3. $\delta(q_0, y, z_0) \longrightarrow$ Reject

4. $\delta(q_0, y, x) \longrightarrow (q_1, \varepsilon)$

5. $\delta(q_1, y, x) \longrightarrow (q_1, \varepsilon)$

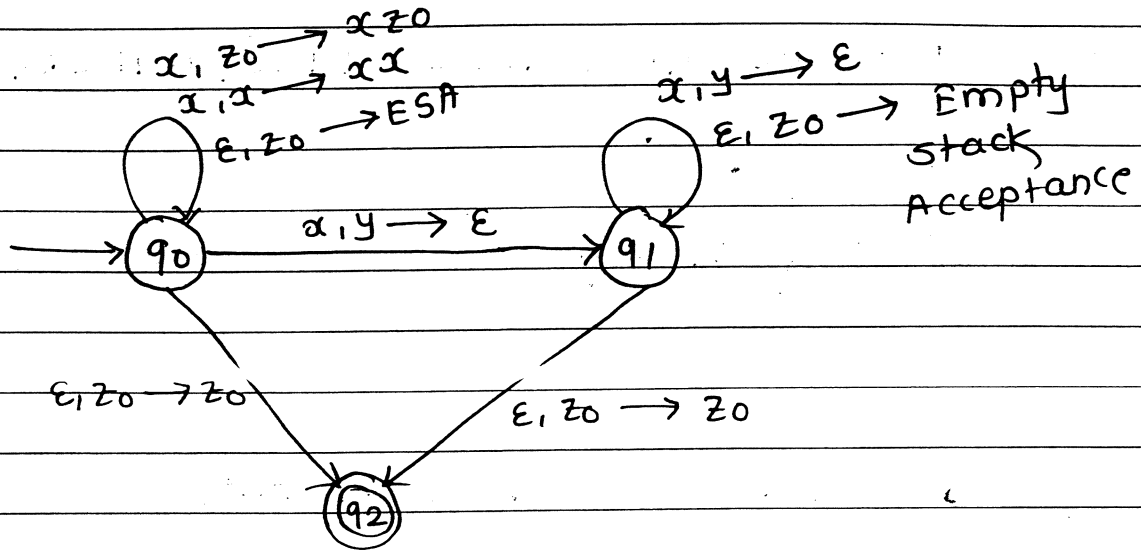
6. $\delta(q_1, \varepsilon, z_0) \longrightarrow$ Accept

7. $\delta(q_0, \varepsilon, z_0) \longrightarrow$ Accept

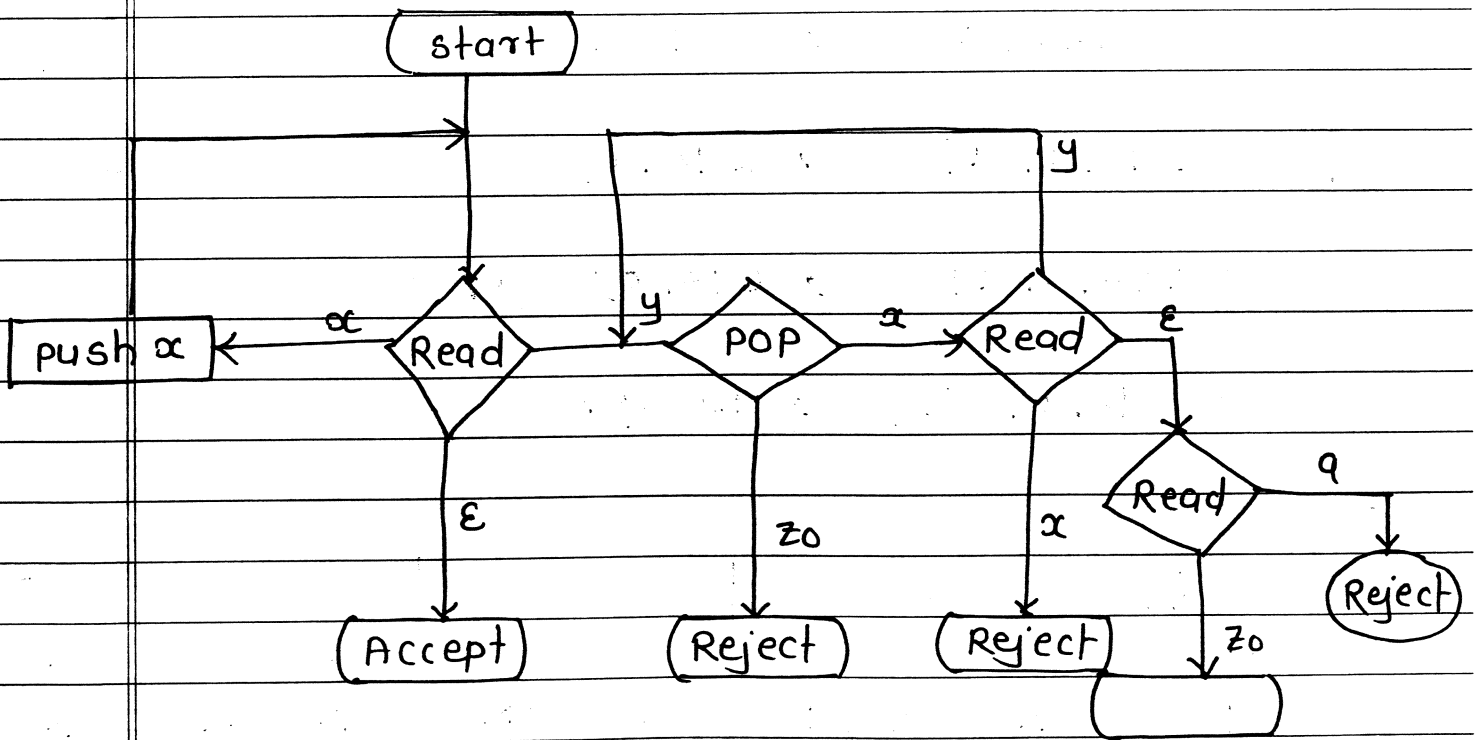
state table representation

i/P / stack	x		y		ε	
	x	z_0	x	z_0	x	z_0
q_0	$q_0 x x$	$q_0 x z_0$	q_1, ε	R	R	A (q_2, ε)
q_1	R	R	q_1, ε	R	R	A (q_2, ε)
q_2	–	–	–	–	–	–

state diagram



Pictorial representation



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

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

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

$\Gamma = \{ a, z_0 \}$

δ transition

$\delta(q_0, a, z_0) \longrightarrow (q_1, a z_0)$

$\delta(q_1, a, a) \longrightarrow (q_1, aa)$

$\delta(q_1, b, a) \longrightarrow (q_1, a)$

$\delta(q_1, c, a) \longrightarrow (q_2, \varepsilon)$

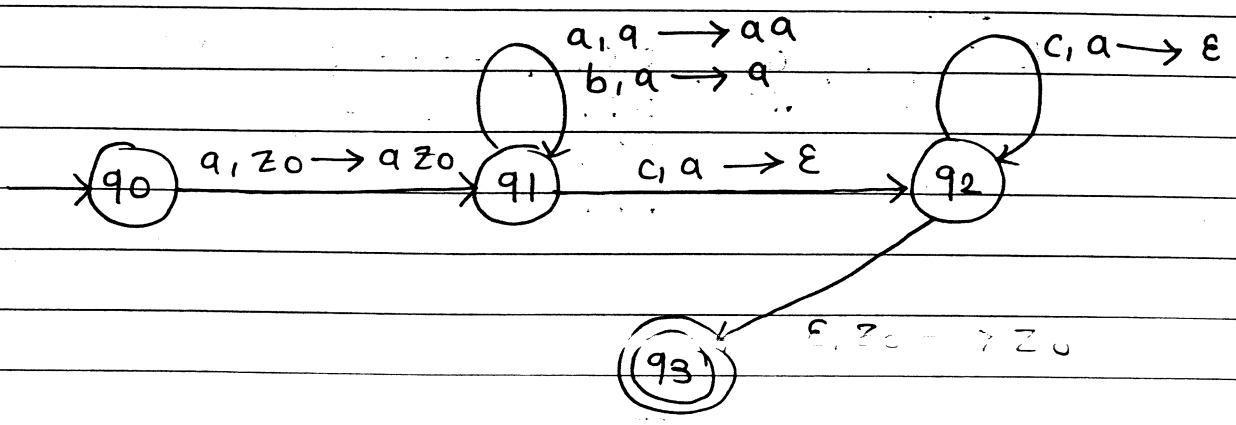
$\delta(q_2, c, a) \longrightarrow (q_2, \varepsilon)$

$\delta(q_2, \varepsilon, z_0) \longrightarrow$ Accept

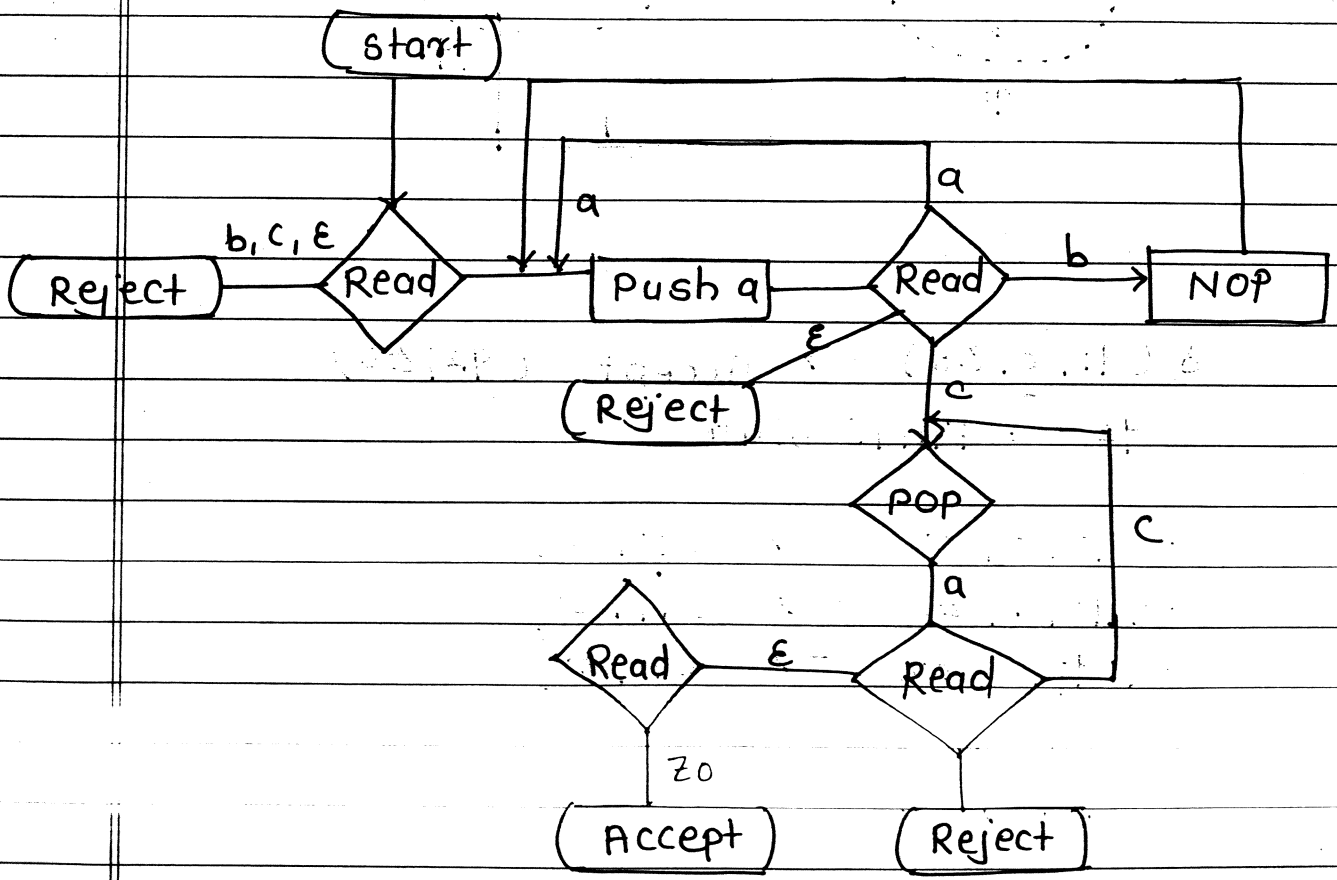
i/p / Stack	a		b		c		ε	
	a	z_0	a	z_0	a	z_0	a	z_0
q_0	R	$q_1, a z_0$	R	R	R	R	R	R
q_1	q_1, aa	R	q_1, a	R	q_2, ε	R	R	R
q_2	R	R	R	R	q_2, ε	R	R	A

q_3, z_0

state diagram representation



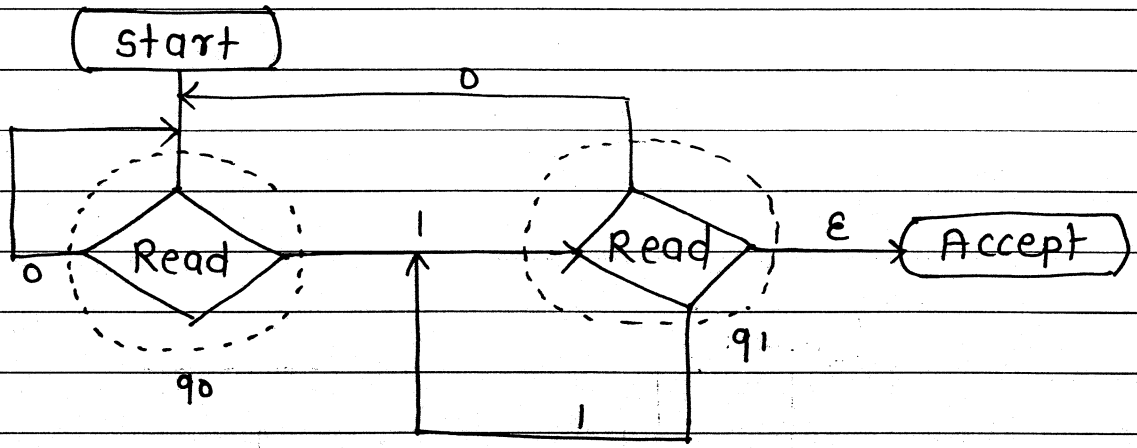
Pictorial representation



5) Design a PDA for $(0+1)^*.1$

$L = \{1, 01, 001, \ldots\}$
$\quad\quad 10 \quad 011$
$\quad\quad\quad\quad 101$
$\quad\quad\quad\quad 111$

Pictorial representation



State transitions

$\delta(q_1, \varepsilon, Z_0) \rightarrow$ Accept (q_2, Z_0)

q_2 = final state

$\delta(q_0, 0, Z_0) \rightarrow (q_0, Z_0)$

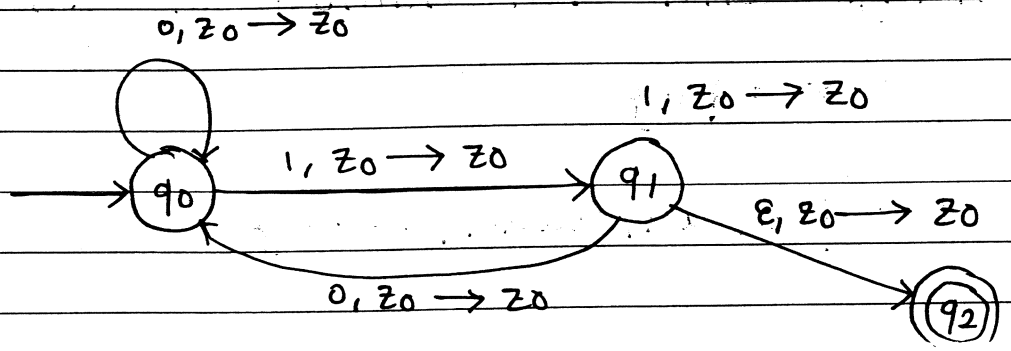
$\delta(q_0, 1, Z_0) \rightarrow (q_1, Z_0)$

$\delta(q_0, 0, Z_0) \rightarrow (q_0, Z_0)$

$\delta(q_1, 1, Z_0) \rightarrow (q_1, Z_0)$

$\delta(q_0, \varepsilon, Z_0) \rightarrow$ Reject

State diagram



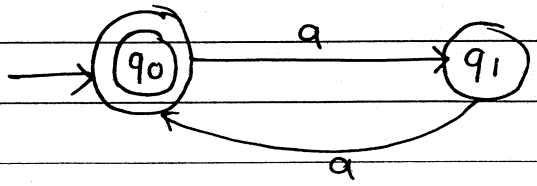
State Table

	0	1	ε
i/p stack	Z0	Z0	Z0
q0	(q0, Z0)	(q1, Z0)	R
q1	(q0, Z0)	(q1, Z0)	(q2, Z0)
q2	–	–	– Accept

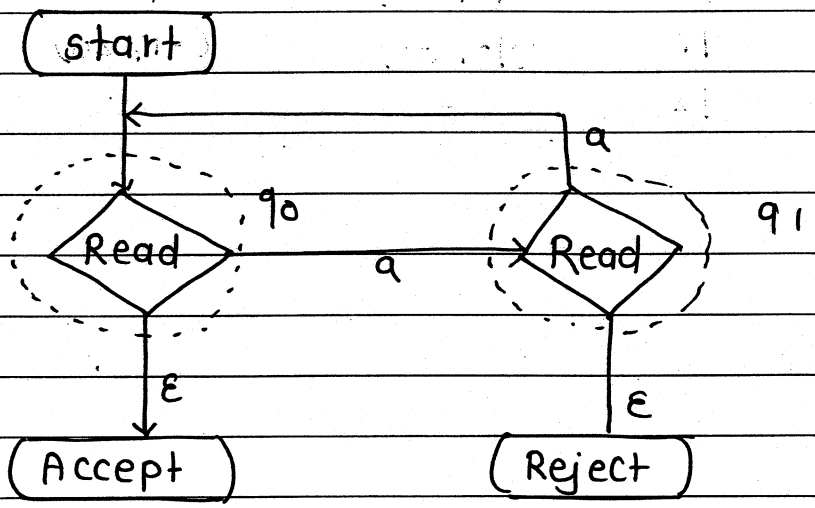
6) $\Sigma = \{a\}$ Design PDA to accept all strings with even number of a's

$\Sigma = \{a\}$

$L = \{\varepsilon, aa, aaaa, ----\}$



Pictorial Representation



State transitions

$\delta(q_0, \varepsilon, z_0) \longrightarrow$ Accept

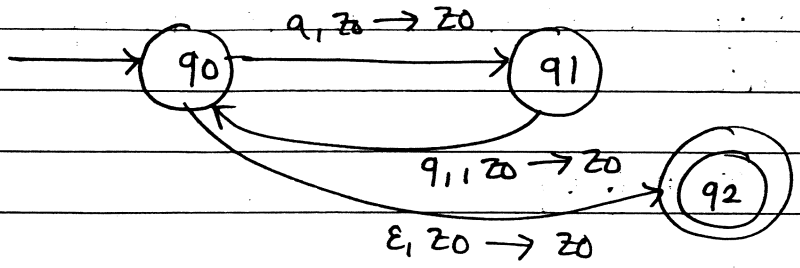
$\delta(q_0, a, z_0) \longrightarrow (q_1, z_0)$

$\delta(q_1, \varepsilon, z_0) \longrightarrow$ Reject

$\delta(q_1, a, z_0) \longrightarrow (q_0, z_0)$

$\delta(q_0, \varepsilon, z_0) \longrightarrow$ Accept (q_f, z_0)

q_f = final state



State table

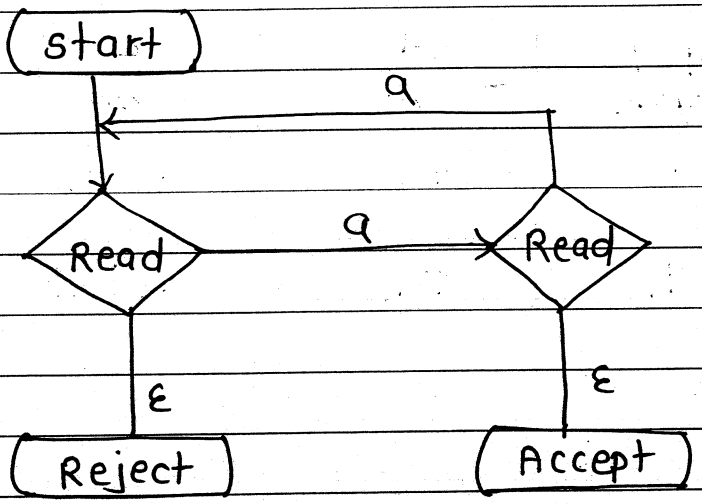
input stack	a z0	ε z0
q0	q1, z0	Accept
q1	q0, z0	Reject

7) $\Sigma = \{a\}$ Design PDA to accept odd number of a's

$\Sigma = \{a\}$

$L = \{a, aaa, ----\}$

Pictorial representation



Transitions

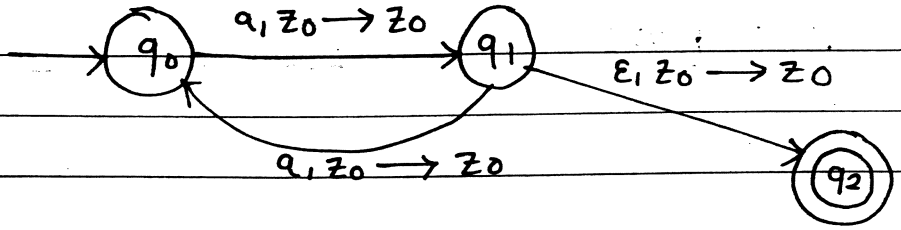
$\delta(q_0, a, z_0) \longrightarrow (q_1, z_0)$

$\delta(q_0, \varepsilon, z_0) \longrightarrow$ Reject

$\delta(q_1, a, z_0) \longrightarrow (q_0, z_0)$

$\delta(q_1, \varepsilon, z_0) \longrightarrow$ Accept

state diagram



State table

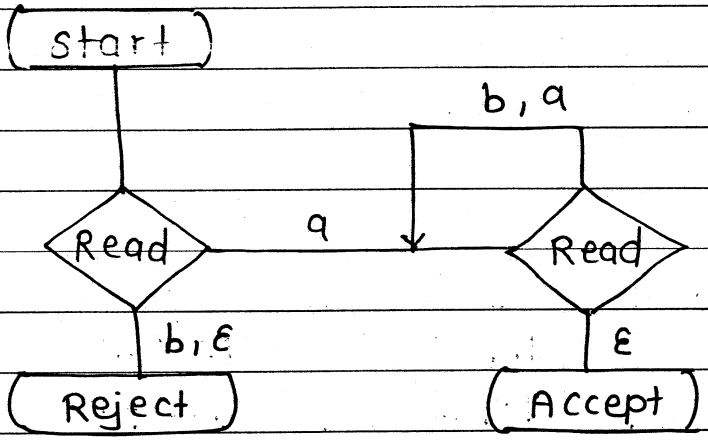
input	a	ε
stack	Z0	Z0
q0	q1, Z0	Reject
q1	q0, Z0	Accept

8) Design PDA for $a(a+b)^*$

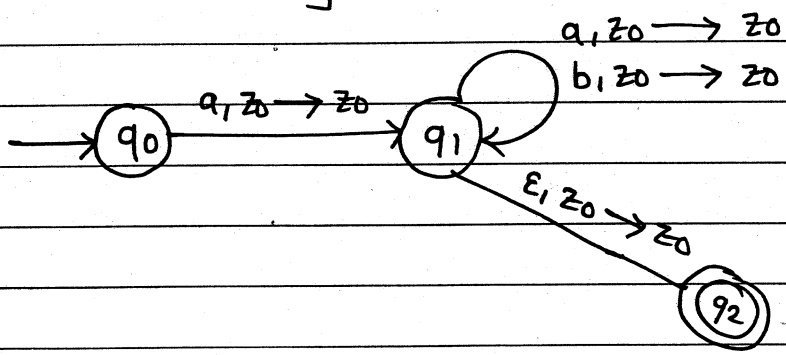
$\Sigma = \{a, b, \varepsilon\}$

$\Gamma = \{z_0\}$

Pictorial Representation



State diagram



state transitions

$\delta(q_0, a, z_0) \longrightarrow (a_1, z_0)$

$\delta(q_q, a, z_0) \longrightarrow (q_1, z_0)$

$\delta(q_1, b, z_0) \longrightarrow (q_1, z_0)$

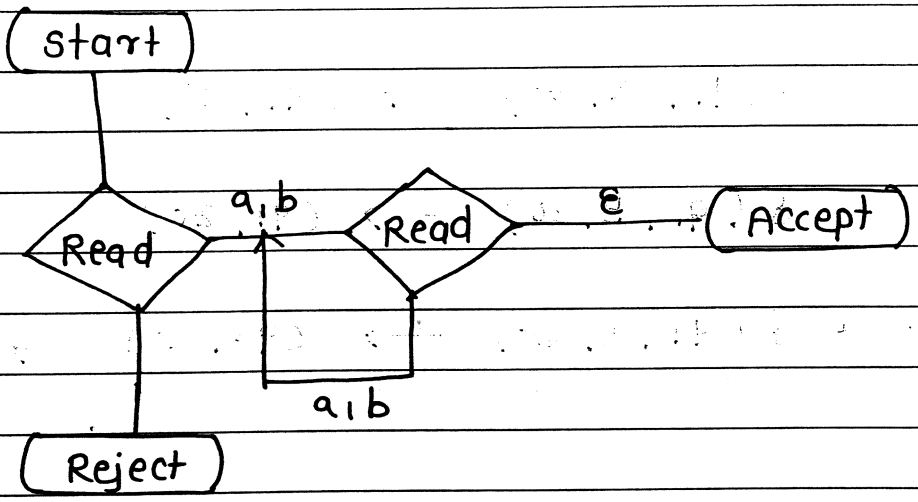
$\delta(q_1, \varepsilon, z_0) \longrightarrow (q_2, z_0)$

State table

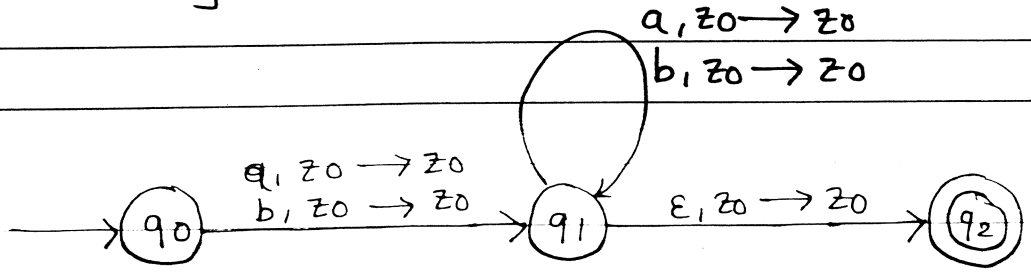
input	a	b	ε
stack	z0	z0	z0
q0	q1, z0	Reject	Reject
q1	q1, z0	q1, z0	q2, z0

9) Design PDA for (a+b) (a+b)*

Pictorial Representation



state diagram



state transition

input stack	a z_0	b z_0	ε z_0
q_0	q_1, z_0	q_1, z_0	Reject
q_1	q_1, z_0	q_1, z_0	Accept

Transitions

$\delta(q_0, a, z_0) \longrightarrow (q_1, z_0)$

$\delta(q_0, b, z_0) \longrightarrow (q_1, z_0)$

$\delta(q_1, a, z_0) \longrightarrow (q_1, z_0)$

$\delta(q_1, b, z_0) \longrightarrow (q_1, z_0)$

$\delta(q_1, \varepsilon, z_0) \longrightarrow (q_2, z_0) \longrightarrow$ Accept

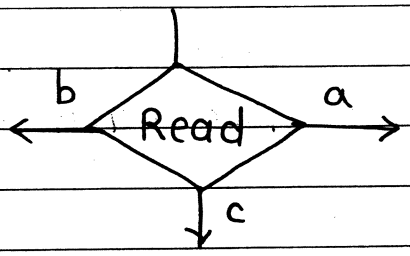
DPDA and NPDA

It's a very precise PDA where deterministic points are available, through which we can determine specific operations of stack (push, pop, nop)

Time complexity of traversal is very less.

For every read operation for certain input symbol the direction is deterministic

e.g.



[only one arrow for each symbol]

For some example,
DPDA does not give solution so there we need NPDA

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

$L = \{ x^n y^m z^n \mid n \geq 1 \}$

$L = \{ w c w^R \mid w \in (a,b)^* \}$

well formness of paranthesis

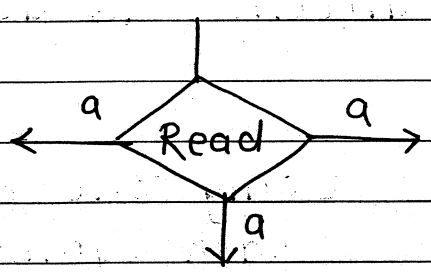
$L = \{ a^{2n} b^n \mid n \geq 1 \}$

NPDA

It's a general machine where designer will not get deterministic point to decide stack operation

Time clomplexity of traversal is huge because of non deterministic

One read operation may have multiple arrows for one input symbol



(zero or more arrows for each symbol)

e.g.

$$L = \{ ww^R \mid w \in (a.b)^* \}$$

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

Design PDA For well formness of parenthesis

$$\Sigma = \{ (,), \varepsilon \}$$

$$Q = \{ q_0 \}$$

$$\Gamma = \{ z_0, (\}$$

$$L = \{ \varepsilon, (), (()), \ldots \}$$

Transitions

1) $\delta(q_0, (, z_0) \rightarrow (q_0, (z_0)$

2) $\delta(q_0, (, () \rightarrow (q_0, (()$

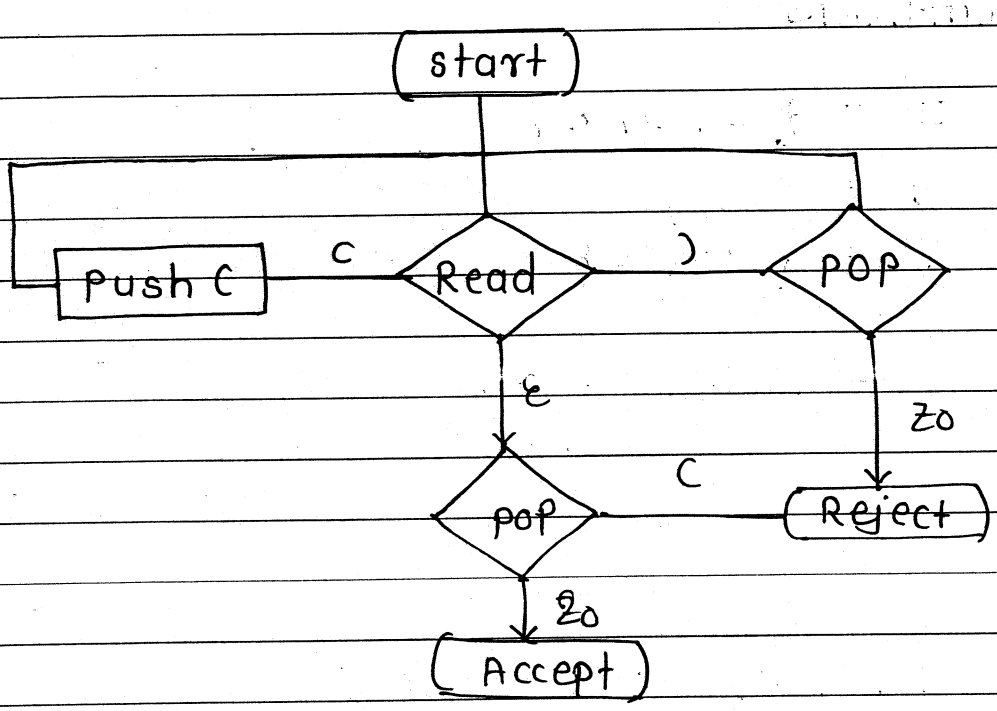
3) $\delta(q_0,), () \rightarrow (q_0, \varepsilon)$

4) $\delta(q_0,), z_0) \rightarrow$ R closing) are more

5) $\delta(q_0, \varepsilon, z_0) \rightarrow$ Accept

6) $\delta(q_0, \varepsilon, () \rightarrow$ R opening (are more,

pictorial representation



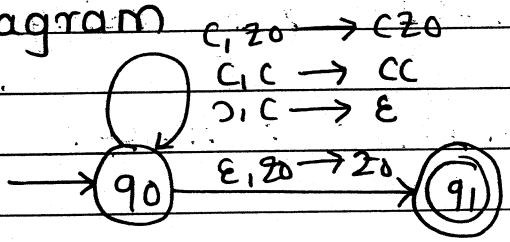
Grammar

$S \rightarrow (S)$

$S \rightarrow SS$

$S \rightarrow \varepsilon$

state diagram



state table

input	(		)		ε	
stack	Z0	C	Z0	C	Z0	C
q0	q0, CZ0	q0, CC	R	q0, ε	q1, Z0	R

Design PDA For $L = \{ \omega c \omega^R \mid \omega \in (a+b)^* \}$

$$\omega = \{ \varepsilon, a, b, aa, bb, \ldots \}$$

$$L = \{ c, aca, bcb, aacaa, abcba, bbcbb, \ldots \}$$

$Q = \{q_0, q_1\}$

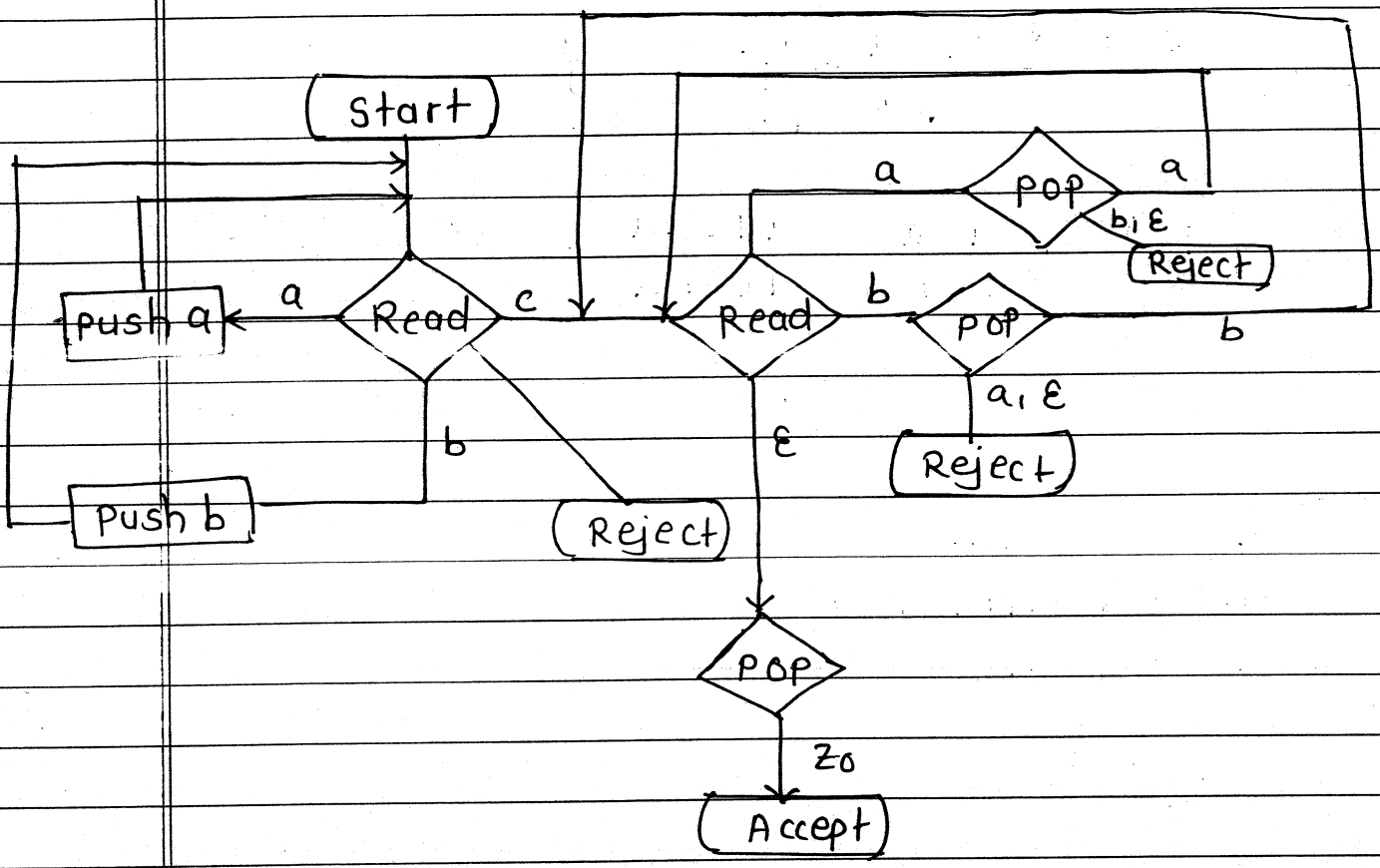
$\Sigma = \{a, b, c, \varepsilon\}$

$\Gamma = \{a, b, z_0\}$

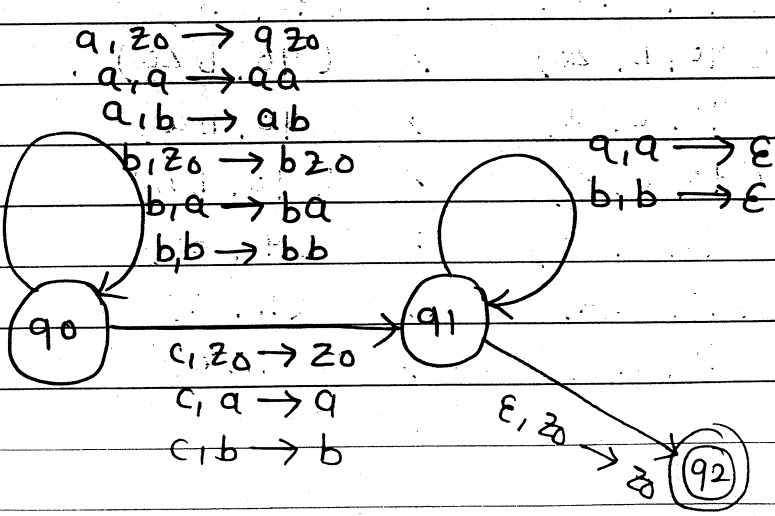
Transitions

1) $\delta(q_0, a, z_0) \longrightarrow (q_0, a z_0)$
2) $\delta(q_0, a, a) \longrightarrow (q_0, aa)$
3) $\delta(q_0, a, b) \longrightarrow (q_0, ab)$
4) $\delta(q_0, b, z_0) \longrightarrow (q_0, b z_0)$
5) $\delta(q_0, b, a) \longrightarrow (q_0, ba)$
6) $\delta(q_0, b, b) \longrightarrow (q_0, bb)$
7) $\delta(q_0, c, z_0) \longrightarrow (q_1, z_0)$
8) $\delta(q_0, c, a) \longrightarrow (q_1, a)$
9) $\delta(q_0, c, b) \longrightarrow (q_1, b)$
10) $\delta(q_1, a, a) \longrightarrow (q_1, \varepsilon)$
11) $\delta(q_1, b, b) \longrightarrow (q_1, \varepsilon)$
12) $\delta(q_1, \varepsilon, z_0) \longrightarrow$ Accept

Pictorial representation



State diagram



state transition table

input → / stack ↓	a			b		
	a	b	z0	a	b	z0
q_0	q_0, aa	q_0, ab	q_0, az_0	q_0, ba	q_0, bb	$q_0 bz_0$
q_1	ε, q_1	R	R	~~$\varepsilon - q_1 a$~~	ε, q_1	R

input	c			ε		
stack	a	b	z0	a	b	z0
q_0	q_1, a	q_1, b	q_1, z_0	R	R	R
q_1	R	R	R	R	R	A

Design PDA to accept following language

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

$$w = \{ \varepsilon, a, b, aa, ab, ba, bb, \ldots \}$$

$$L = \{ \varepsilon, aa, bb, aaaa, abba, baab, bbbb, \ldots \}$$

set of all even palindrome over $\Sigma = \{a, b\}$

$$S \rightarrow aSa \mid bSb \mid \varepsilon$$

$Q = \{q_0, q_1\}$

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

$\Gamma = \{a, b, z_0\}$

transitions

1) $\delta(q_0, \varepsilon, z_0) \rightarrow$ Accept

2) $\delta(q_0, a, z_0) \rightarrow (q_0, a z_0)$

3) $\delta(q_0, a, a) \rightarrow (q_0, aa)$

4) $\delta(q_0, a, b) \rightarrow (q_0, ab)$

5) $\delta(q_0, b, Z_0) \longrightarrow (q_0, b Z_0)$

6) $\delta(q_0, b, a) \longrightarrow (q_0, ba)$

7) $\delta(q_0, b, b) \longrightarrow (q_0, bb)$

8) $\delta(q_0, a, a) \longrightarrow (q_1, \varepsilon)$

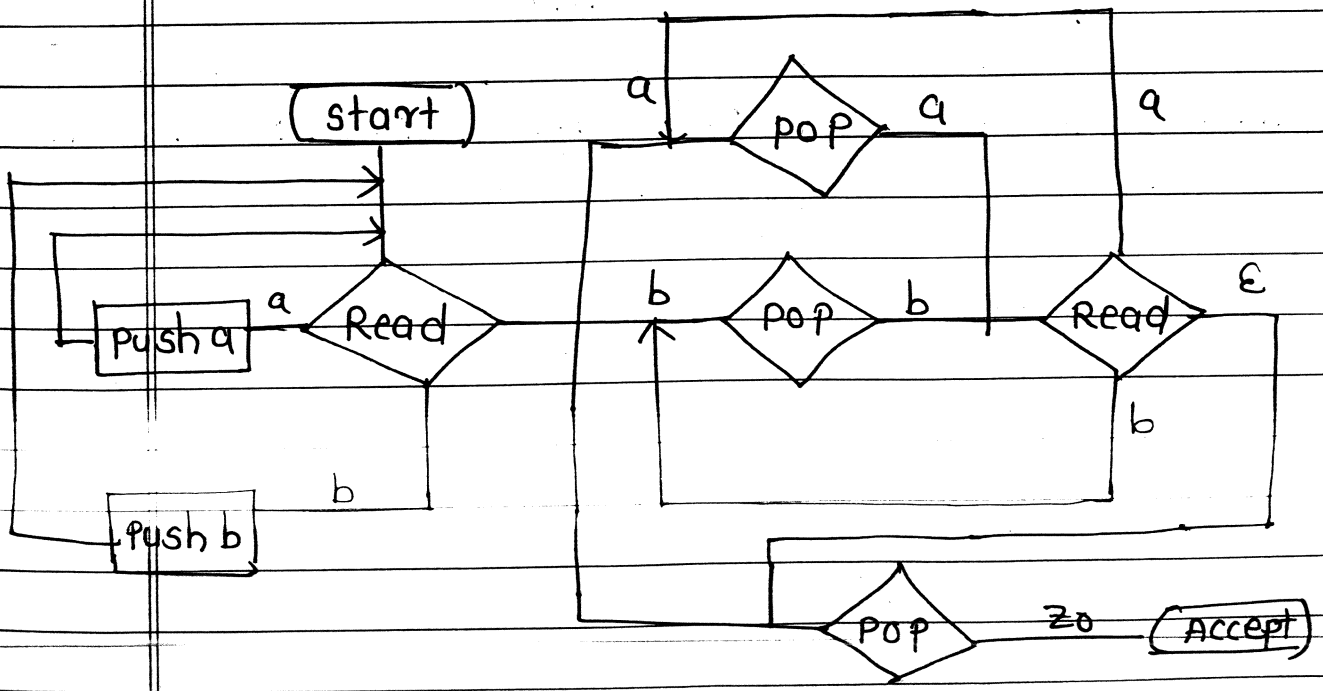
9) $\delta(q_0, b, b) \longrightarrow (q_1, \varepsilon)$

10) $\delta(q_1, a, a) \longrightarrow (q_1, \varepsilon)$

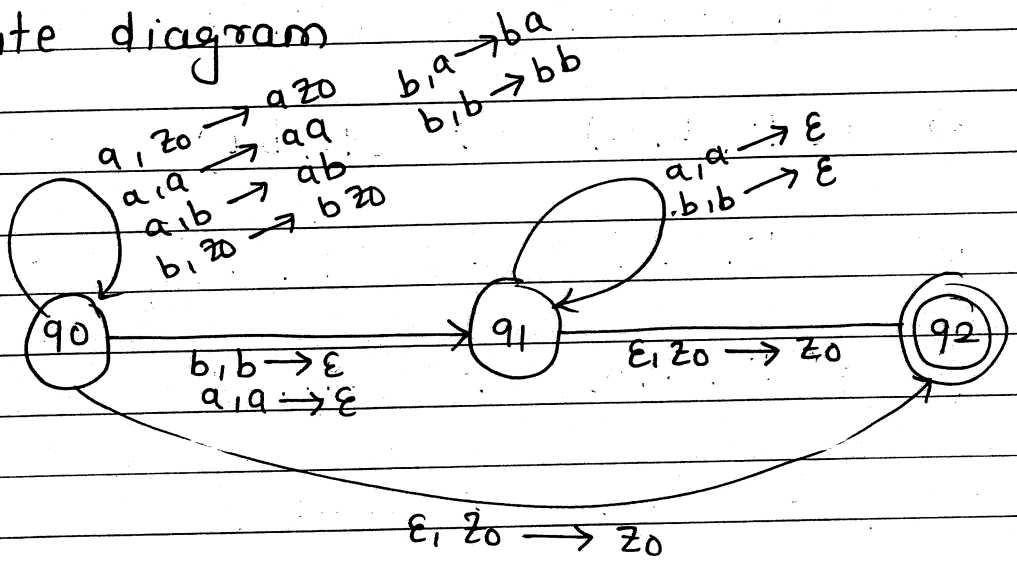
11) $\delta(q_1, b, b) \longrightarrow (q_1, \varepsilon)$

12 $\delta(q_1, \varepsilon, Z_0) \longrightarrow$ Accept

Pictorial representation



state diagram



state table

Input	a			b			ε		
stack	a	b	z0	a	b	z0	a	b	z0
q0	q0, aa q1, ε	q0, ab	q0, az0	q0, ba	q0, bb q1, ε	q0, bz0	R	R	A
q1	q1, ε	R	R	R	q1, ε	R	R	R	A

PDA and grammar equivalence

CFG to PDA

Let $G = (N, T, P, S)$ is a CFG which represents language L with set of production rules then there is always equivalent PDA represented as

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

which accepts same language L with empty stack acceptance.

For this conversion, following two rules can be used :

Rule 1 : For non-terminals

If non terminal is present on the top of stack then it is removed from top of the stack and replaced with its RHS on top of the stack.

e.g. $A \longrightarrow aB$

$$\delta(q, w, A) \longrightarrow (q, aB)$$

Rule 2 : For terminal

If PDA reads terminal symbol and same terminal symbol is on the top of stack then erase terminal symbol from top

e.g. $\delta(q, a, a) \longrightarrow (q, \varepsilon)$

1) Convert following grammar into equivalent PDA

$S \longrightarrow aSB$

$B \longrightarrow b$

$S \longrightarrow aB$

$NT = \{S, B\}$

$T = \{a, b\}$

For NT : $S \longrightarrow aSB$

$\delta(q, w, S) \longrightarrow (q, aSB)$ ... (1)

$B \longrightarrow b$

$\delta(q, w, B) \longrightarrow (q, b)$ ... (2)

$S \longrightarrow aB$

$\delta(q, w, S) \longrightarrow (q, aB)$ ... (3)

For Terminal

$\delta(q, a, a) \longrightarrow (q, \varepsilon)$ ... (4)

$\delta(q, b, b) \longrightarrow (q, \varepsilon)$ ... (5)

Handrun

aabb

$\delta(q, aabb, S) \longrightarrow (q, aSB)$

$\delta(q, aabb, a) \longrightarrow (q, SB)$

$\delta(q, abb, S) \longrightarrow (q, aBB)$

$\delta(q, abb, a) \longrightarrow (q, BB)$

$\delta(q, bb, B) \longrightarrow (q, bB)$

$\delta(q, bb, b) \longrightarrow (q, B)$

$\delta(q, b, B) \longrightarrow (q, b)$

$\delta(q, b, b) \longrightarrow (q, \varepsilon)$

$\delta(q, \varepsilon, Z_0) \longrightarrow$ Accept on empty stack condn

2) $S \rightarrow 0BB$

$B \rightarrow 0S \mid 1S \mid 0$

word $= 010^4 = 010,000$

For non terminal

$S \rightarrow 0BB$

$\delta(q, w, S) \rightarrow (q, 0BB)$ ---(1)

$B \rightarrow 0S$

$\delta(q, w, B) \rightarrow (q, 0S)$ ----(2)

$B \rightarrow 1S$

$\delta(q, w, B) \rightarrow (q, 1S)$ ---(3)

$B \rightarrow 0$

$\delta(q, w, B) \rightarrow (q, 0)$ ---(4)

For terminal

$B \rightarrow 0$

$\delta(q, w, S) \rightarrow (q, 0)$

$\delta(q, 0, 0) \rightarrow (q, \varepsilon)$

$\delta(q, 1, 1) \rightarrow (q, \varepsilon)$

Transitions - handrun for 010000

$\delta(q, 010000, S) \rightarrow (q, 0BB)$

$\delta(q, 010000, 0) \rightarrow (q, BB)$

$\delta(q, 010000, B) \rightarrow (q, 1SB)$

$\delta(q, 10000, 1) \rightarrow (q, SB)$

$\delta(q, 0000, S) \rightarrow (q, 0BBB)$

$\delta(q, 0000, 0) \rightarrow (q, BBB)$

$\delta(q, 000, B) \rightarrow (q, 0BB)$

$\delta(q, 000, 0) \rightarrow (q, BB)$

$\delta(q, 00, B) \rightarrow (q, 0B)$

$\delta(q, 00, 0) \rightarrow (q, B)$

$\delta(q, 0, B) \rightarrow (q, 0)$

$\delta(q, 0\ 0) \rightarrow (q, \varepsilon)$

$\delta(q, \varepsilon\ \varepsilon) \rightarrow (\text{Accept})$

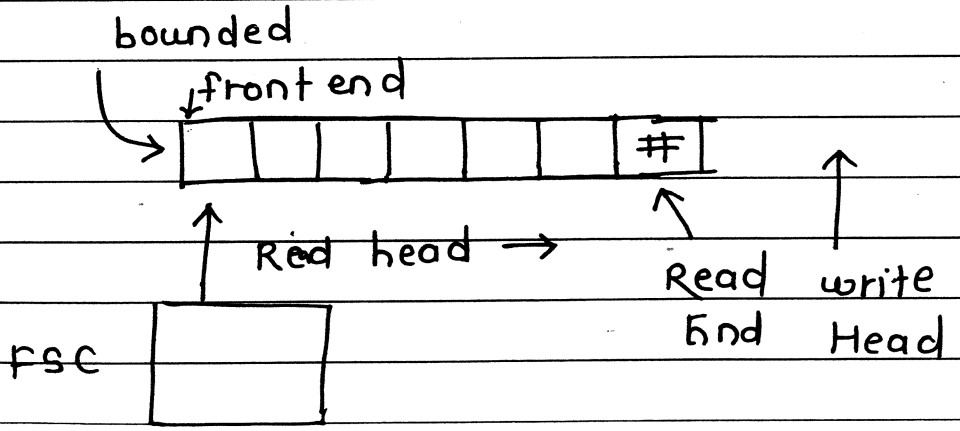
Post Machine

Emil Post in 1936

In this machine, the additional memory is provided to finite automata in the form of queue [First in First out]

The input tape is considered as an additional memory of this machine where elements can be deleted from front end and inserted in rear head

Post machines are more powerful than PDA as stack operations can be stimulated with queue but queue can not be stimulated with stack.



Post machine is represented by 7 tuples

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

Q = finite number of states
Σ = finite number of input alphabets
q_0 = Initial state
Z_0 = rear end symbol
F = Halt
Γ = Queue symbols $[\Sigma \cup Z_0 \cup *]$ → any other possibility of char
δ = state transition function

$(P.S \times \text{Front end symbol}) \longrightarrow (N.S, \text{Rear End operation})$

$\delta(q_0, a) \longrightarrow (q_1, \varepsilon)$
↓ delete ↓ don't insert

$\delta(q_0, a) \longrightarrow (q_1, b)$
↓ delete ↓ insert

$\delta(q_0, \varepsilon) \longrightarrow (q_1, a)$
↓ don't delete ↓ insert

$\delta(q_0, \varepsilon) \longrightarrow (q_1, \varepsilon)$
↓ don't delete ↓ don't insert

to delete a, goes to q_1 & insert nothing

for b ⟶ q_2 delete b, stay in q_2, insert b

for z_0 ⟶ q_2 transite q_0 & insert z_0

Design Post machine to accept following language

$$L = \{ a^n b^n \mid n \geq 0 \}$$

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

$$S \longrightarrow aSb \mid \varepsilon$$

aaabbb

a	a	a	b	b	b	z_0

aaabbbz_0

↑ q_0 delete a & transit to q_1

aaabbbz_0

↑ q_1 delete a & insert a

aabbbz_0a

↑ q_1 delete b & transit q_2

abbb$z_0$aa

↑ q_2

bbb$z_0$aa

↑ q_1 delete b & transit q_2

bb$z_0$aa

↑ q_2

b$z_0$aab

↑ q_2

$z_0$aabb

↑ q_2

aabbz_0

↑ q_0

abbz_0a

↑ q_1

bbz_0a

↑ q_1

bz_0a

↑ q_2

$z_0$ab

↑ q_2

abz_0

↑ q_0

bz_0

↑ q_1

z_0

↑ q_2

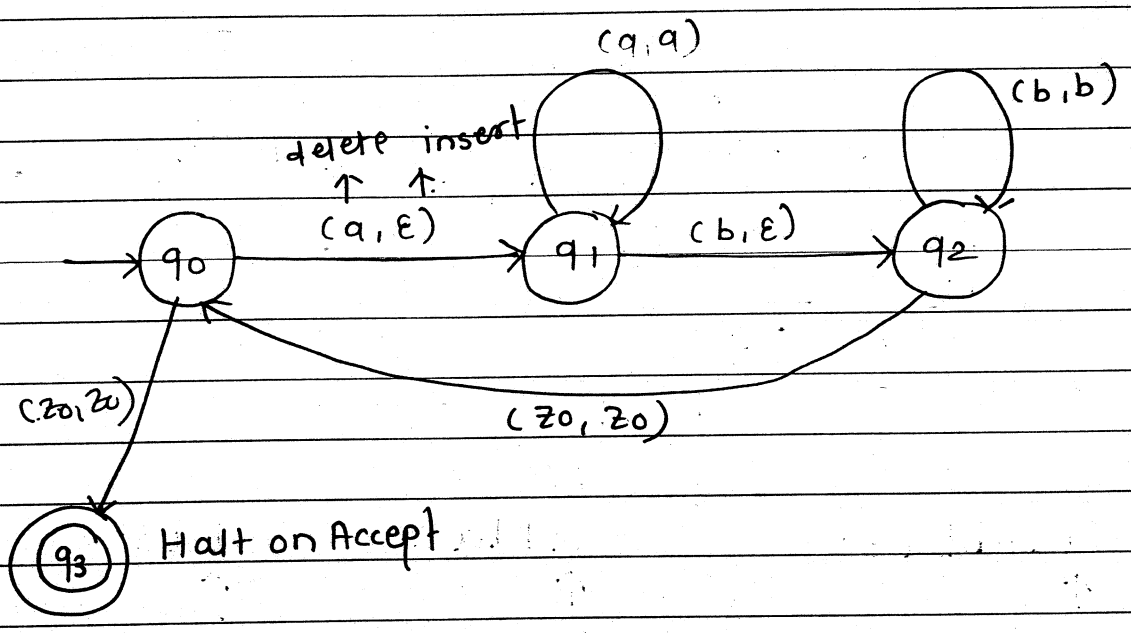
z_0

↑ q_0

q_3 → halt state

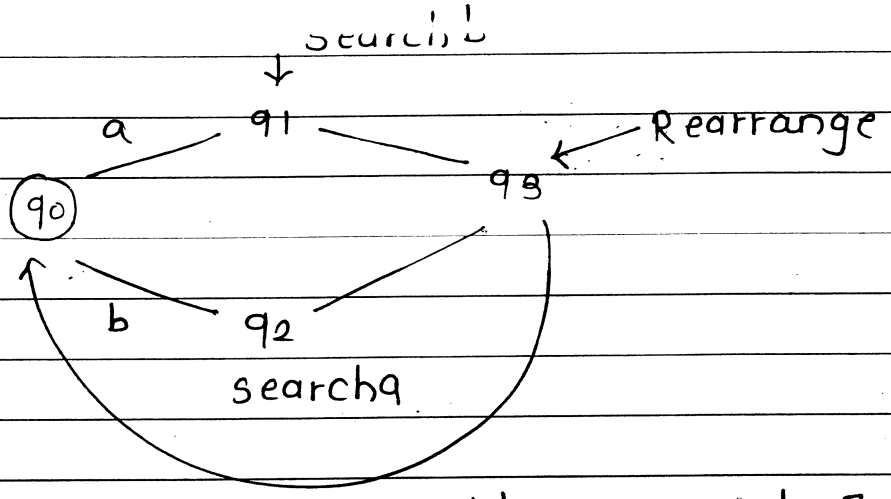
q1 - searches for b

	a	b	z0
→ q0	(q1, ε)	-	(q3, z0)
q1	(q1, a)	(q2, ε)	-
q2	-	(q2, b)	(q0, z0)
q3	-	-	- (Halt state)



Design Post machine for equal no. of a's and b's

$L = \{ \varepsilon, ab, aabb, \ldots \}$
ba bbaa
abab
baba



aabbbaz0
↑
q0 → delete a

abbbaz0
↑
q1

bbbaz0a
↑
q1

bbaz0a
↑
q3

baz0ab
↑
q3

az0abb
↑
q3

z0aabba

abbaz0
↑
q0

bbaz0
↑
q1

baz0
↑
q3

az0b
↑
q3

z0ba
↑
d3

baz0
↑
q0

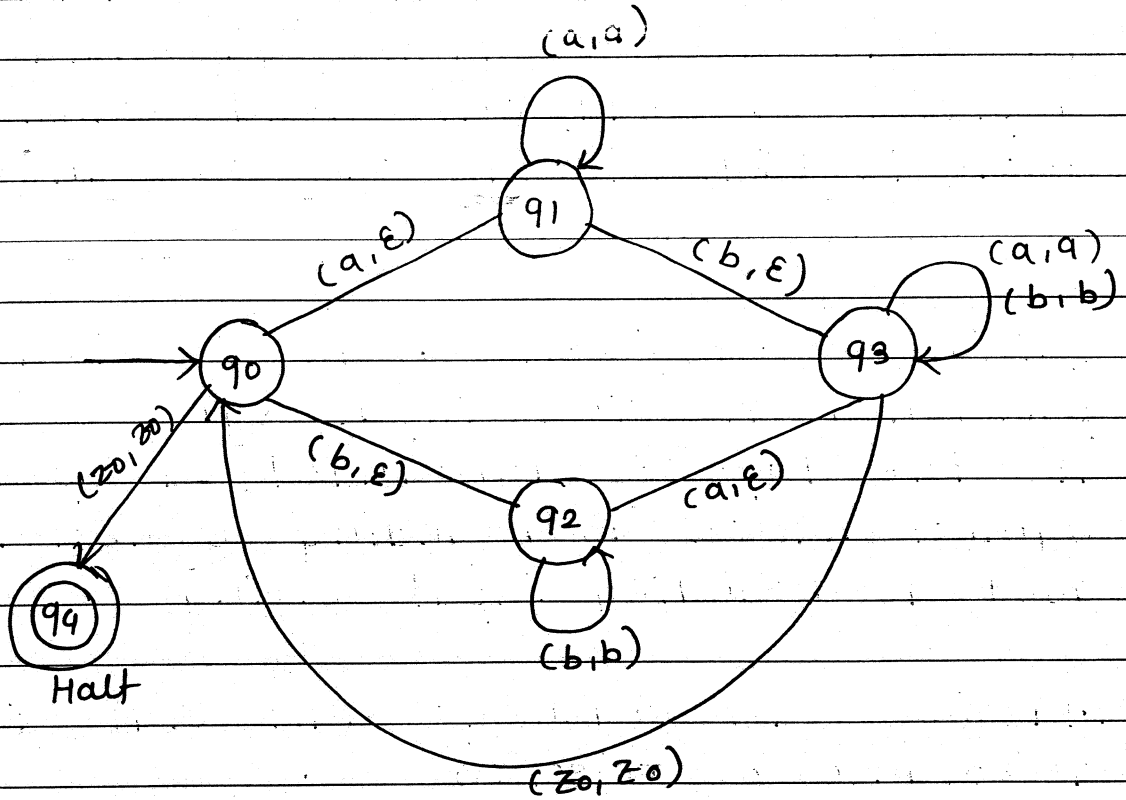
az0
↑
q2

z0
↑
q3

z0
↑
q0

q4

	a	b	Z_0
q_0	(q_1, ε)	(q_2, ε)	(q_4, Z_0)
q_1	(q_1, a)	(q_3, ε)	-
q_2	(q_3, ε)	(q_2, b)	-
q_3	(q_3, a)	(q_3, b)	(q_4, Z_0)
q_4	-	-	-



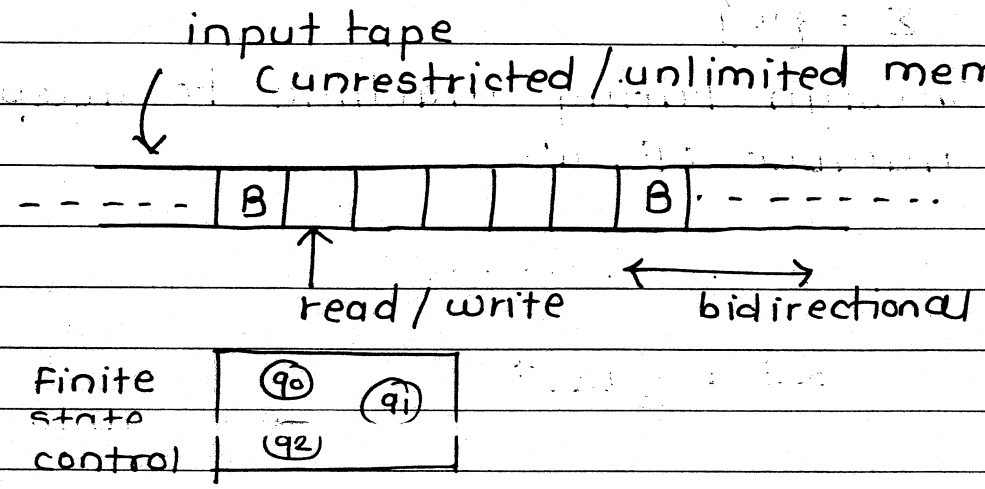
Turing Machine

Alan Turing – 1936

It's a finite automata with unlimited and unrestricted memory.
Input tape is considered as unlimited and unrestricted memory of it which is considered as open ended.
The read head of finite automata is replaced by read/write head which can move in bydirectional way or can stay at same position.

Initially read/write head is always positioned at first input symbol
As the tape is unbounded, a special symbol known as bounding symbol is used as bounding input string

Machine can reach to halt state for acceptance or rejection
This machine has provided the first basic idea of general purpose computers.
As it accepts all types of languages defined by chomsky's hierarchy, it is known as universal machine.



Technically it is represented by 7 tuples

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

where,

- Q = Finite number of states
- Σ = finite number of inputs
- Γ = Tape symbols $\{ \Sigma \cup B \cup * \}$
- q_0 = initial state
- B = Bounding symbol
- F = Halt state

δ (PS x tape symbol) $\Rightarrow$ (N.S, what to write on tape, R/w head movement (R, L, N))

$\delta(q_0, a) \rightarrow (q_1, a, R)$ → Right side

$\delta(q_0, a) \rightarrow (q_1, b, L)$ → Left

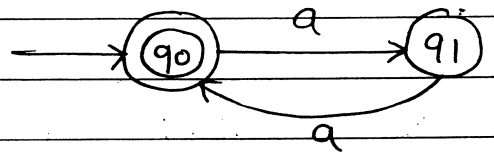
$\delta(q_2, a) \rightarrow (q_3, *, N)$ → no movement

Q. $\Sigma = \{a\}$

Design turing machine to accept even number of a's

$L = \{\varepsilon, aa, aaaa, ---\}$

$RE = (aa)^*$



q_0 = can see even a's

q_1 = can see odd a's

(I) even

B a a a a B
↑
q_0

B a a a a B
↑
q_1

B a a a a B
↑
q_0

B a a a a B
↑
q_1

B a a a a B
↑
q_0

B a a a a B
↑
q_2 Halt on Accept

(II) odd

B a a a B
↑
q_0

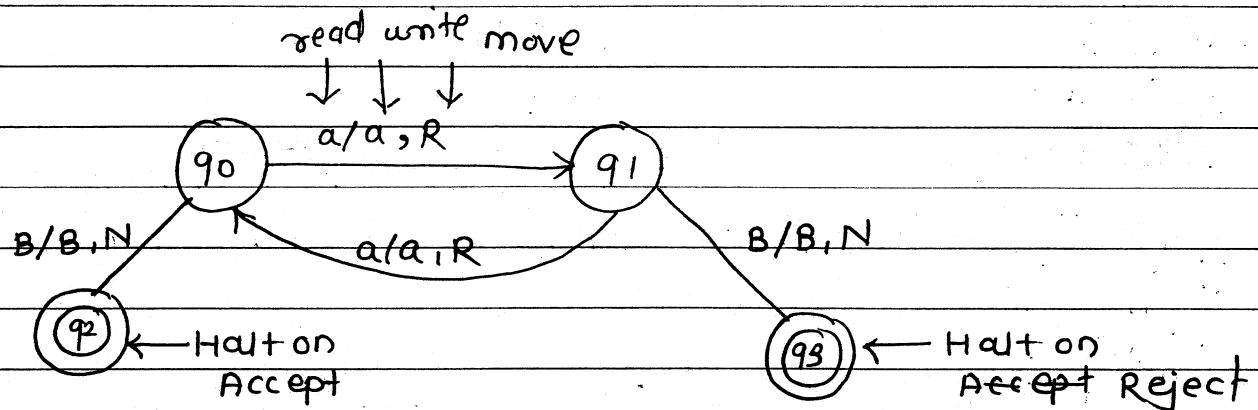
B a a a B
↑
q_1

B a a a B
↑
q_0

B a a a B
↑
q_1

B a a a B
↑
q_3 → Halt on Reject

	a	B
→ q_0	(q_1, a, R)	(q_2, B, N)
q_1	(q_0, a, R)	(q_3, B, N)
q_2	-	- Halt on Accept
q_3	-	- Halt on Accept



stay in / change state
right back
move towards right

classmate Date ____ Page ____

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

Design turing machine to accept odd number of 1's

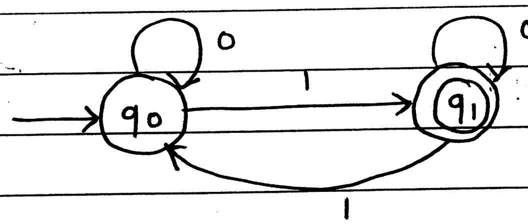
odd number of 1's

if 0 comes, no transition only move to right right back 0

$L = \{ 1, 01, 001, \ldots \}$
10 010
100
111

if q_1 runs then state changes to q_0 and 1 is write back

q_0 = can see even number of 1's
q_1 = can see odd number of 1's



B 0011010111 B
↑
q_0

B 0011010111 B
↑
q_0

q_0 is going to read 1 so no change → B 0011010111 B
↑
q_0

B 0011010111 B
↑
q_1

write back 1

B 0 0 1 1 0 1 0 1 1 B
↑
q_0

B 0 0 1 1 0 1 0 1 1 B
↑
q_0

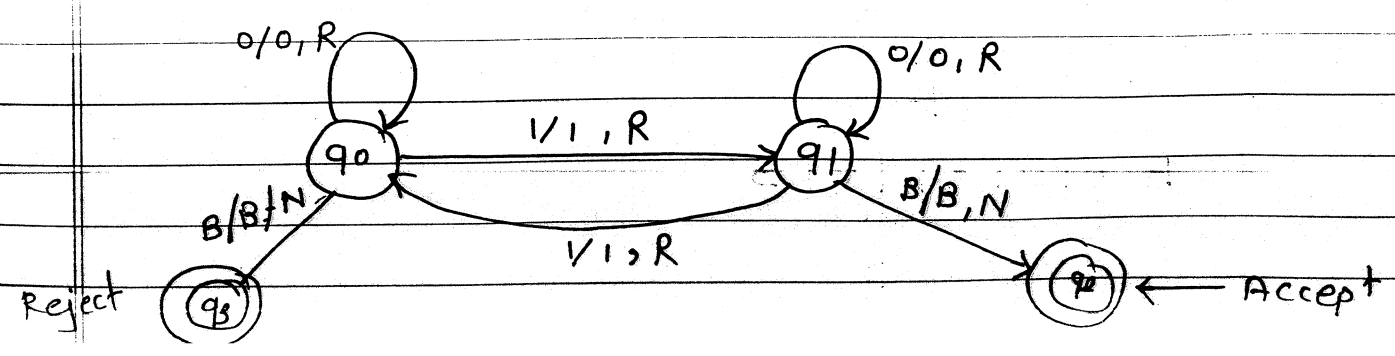
B 0 0 1 1 0 1 0 1 1 B
↑
q_1

B 0 0 1 1 0 1 0 1 1 B
↑
q_1

B 0 0 1 1 0 1 0 1 1 B
↑
q_0

B 0 0 1 1 0 1 0 1 1 B
↑
q_1
↓
q_2 → Halt on accept

	0	1	B
→ q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	(q_3, B, N)
q_1	$(q_1, 0, R)$	$(q_0, 1, R)$	(q_2, B, N)
q_2	-	-	- Halt on Accept
q_3	-	-	- Halt on Reject



find & replace (No accept)

Q.3 Design turing machine to replace 11 by 1(

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

$q_0 \rightarrow$ ss / If see 0, don't transit / if see 1 then transit to q_1

$q_1 \rightarrow$ If see 1, replace by 0 and transit to q_0

If see 0, write 0 back and transit to q_0

B 0 1 1 1 0 1 1 0 1 B
↑ q_0

B 0 1 1 1 0 1 1 0 1 B
↑ q_0

B 0 1 1 1 0 1 1 0 1 B
↑ q_1

B 0 1 0 1 0 1 1 0 1 B
↑ q_0

B 0 1 0 1 0 1 1 0 1 B
↑ q_1

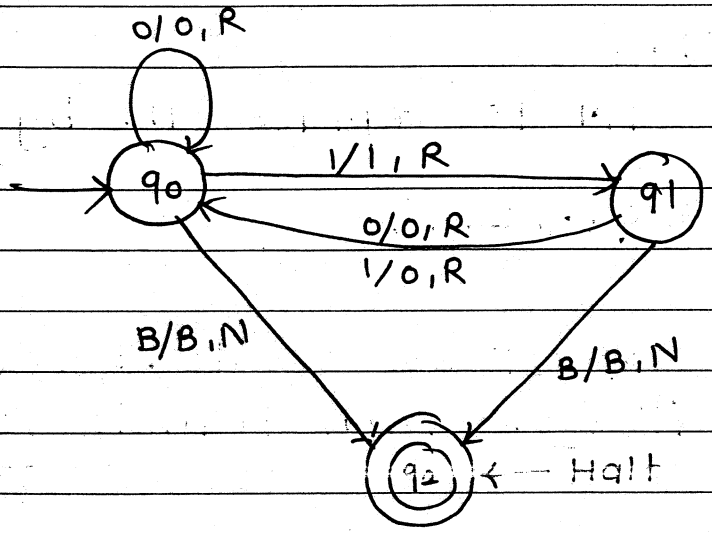
B 0 1 0 1 0 1 1 0 1 B
↑ q_0

B 0 1 0 1 0 1 1 0 1 B
↑ q_1

B 0 1 0 1 0 1 0 0 1 B
↑ q_0

B 0 1 0 1 0 1 0 0 1 B
↑ q_0

B 0 1 0 1 0 1 0 0 1 B
↑ q_1



	0	1	B	
q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	(q_2, B, N)	
q_1	$(q_0, 0, R)$	$(q_0, 0, R)$	(q_2, B, N)	
q_2	–	–	–	Halt

Q.3 $\Sigma = \{0, 1\}$

Design TM to replace 111 by 110

B 0 1 1 1 0 1 1 B

q_0 → ss/ If see 0, don't transit / if see 1, then transit to q_1

q_1 → If see 1, transit to q_2 / if see 0, write back 0, and transit to q_0

q_2 → if see 1, ~~write back~~ replace by 0 and transit to q_0
if see 0, write back 0 and transit to q_0

B 0 1 1 1 0 1 1 B
↑
q_0

B 0 1 1 0 0 1 1 B
↑
q_0

B 0 1 1 1 0 1 1 B
↑
q_0

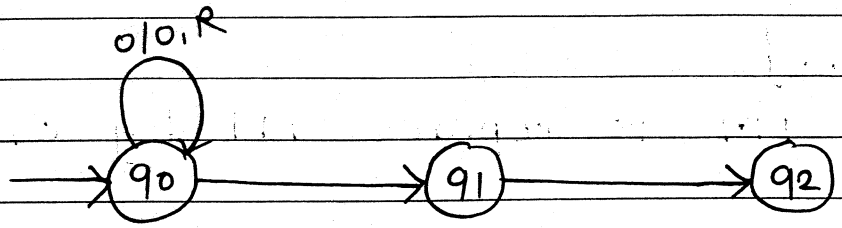
B 0 1 1 0 0 1 1 B
↑
q_0

B 0 1 1 1 0 1 1 B
↑
q_1

B 0 1 1 0 0 1 1 B
↑
q_1

B 0 1 1 1 0 1 1 B
↑
q_2

B 0 1 1 0 0 1 1 B
↑
q_2



		0	1	B
→	q0	(q0, 0, R)	(q1, 1, R)	(q3, B, N)
	q1	(q0, 0, R)	(q2, 1, R)	(q3, B, N)
	q2	(q0, 0, R)	(q0, 0, R)	— Halt (q3, B, N)
	q3	—	—	— Halt

Q. 4 $\Sigma = \{0, 1\}$

Design TM to replace 111 by 011

B 0 1 1 1 1 0 B
↑ q0

B 0 1 1 1 1 0 B
↑ q0

B 0 1 1 1 1 0 B
↑ q1

B 0 1 1 1 1 0 B
↑ q2

B 0 1 1 1 1 0 B
↑ q3

B 0 1 1 1 1 0 B
↑ q4

B 0 0 1 1 1 0 B
↑ q0

B 0 0 1 1 1 0 B
↑ q1

B 0 0 1 1 1 0 B
↑ q2

B 0 0 0 1 1 0 B
↑ q0

B 0 0 0 1 1 0 B
↑ q1

B 0 0 0 1 1 0 B
↑ q2 → q2 is searching for 3rd 1

B 0 0 0 1 1 0 B
↑ q0 but didn't get so move to q0 again

	0	1	B
q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	(q_5, B, N)
q_1	$(q_0, 0, R)$	$(q_2, 1, R)$	(q_5, B, N)
q_2	$(q_0, 0, R)$	$(q_3, 1, L)$	(q_5, B, N)
q_3	–	$(q_4, 1, L)$	–
q_4	–	$(q_0, 0, R)$	–
q_5	–	–	– Halt

When we are working on Langu... then only halt on accept

classmate
Date
Page

Q. 2's complement

2's complement : 1 0 1 1. 0 0 (LSB)

ગોવર 1st 1 ચેત નહી તોવર traverse કરવો

if 1'st 1 comes, keep it as it is and after it all digits complement

1 0 1 1. 0 0

1 : 0 : 1 0 0

complement

0 1 0 1 0 0

B 1 0 1 1 0 0 B

traverse

q0 q0

B 1 0 1 1 0 0 B

q1

B 1 0 1 1 0 0 B

q1

B 1 0 1 1 0 0 B

q1

B 1 0 1 1 0 0 B

q2

B 1 1 0 1 0 0 B

q2

B 0 1 0 1 0 0 B

q2

q3

Halt

B 1 0 0 1 0 0 B

q2

	0	1	B
q_0	$(q_0, 0, R)$	$(q_0, 1, R)$	(q_1, B, L)
q_1	$(q_1, 0, L)$	$(q_2, 1, L)$	(q_3, B, N)
q_2	$(q_2, 1, L)$	$(q_2, 0, L)$	(q_3, B, N)
q_3	–	–	–

B 0 0 0 B
↓ q_0 ←
q_3 halt

9 $\Sigma = \{0, 1\}$

Design TM to increment given binary number by 1

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

1)

```
  B 0 1 0 B
+       1
-----------
  B 0 1 1 B
```

```
B 0 1 0 B
  ↑     ↑
  q0    q0

B 0 1 0 B
    ↑
    q1

B 0 1 1 B
    ↑
    q2 Halt
```

if LSB = 0, Make it 1 and halt

2)

```
  B 0 1 1 B
+       1
-----------
  B 1 0 0 B
```

```
B 0 1 1 B
  ↑ ... ↑
  q0    q0

B 0 1 1 B
    ↑
    q1

B 0 1 0 B
    ↑
    q1

B 0 0 0 B
    ↑
    q1

B 1 0 0 B
    ↑
    q2
    Halt
```

If LSB = 1, then make it 0 and for first 0, make it 1 and halt and for all trailing 1's, make it 0

1111
+ 1111
10

2
-2
0

3
-2
1

(0-9)10

497
+868
1365

15
-10
5

16
-10
6

how many times we are substracting 10 so (1) as carry

3) B 1 1 1 B
+1 1
1 0 0 0

If LSB = 1 then make it 0 and for all trailing 1's make it 0
If bounding, make it 1 and halt.

B 1 1 1 B
↑ ---- ↑
q0 q0

B 1 1 1 B
↑
q1

B 1 1 0 B
↑
q1

B 1 0 0 B
↑
q1

B 0 0 0 B
↑
q1

1 0 0 0 B
↑
q2 halt

	0	1	B
q0	(q0, 0, R)	(q0, 1, R)	(q1, B, L)
q1	(q2, 1, L)	(q1, 0, L)	(q2, 1, L)
q2	-	-	-

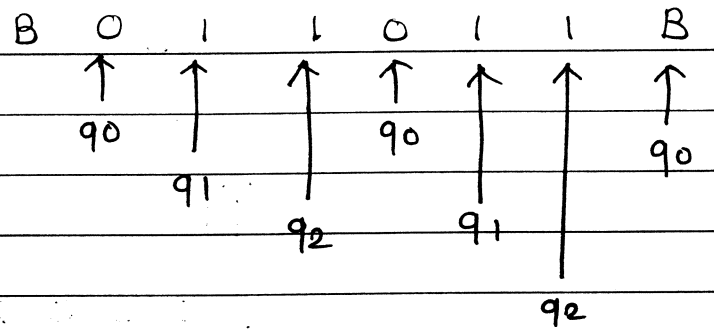
Respecting with finite automata

Q Design TM for checking whether given binary numbers' length divisible by 3

Both 0 and 1 contribute in length of number

q_0 → can see rem 0
q_1 → can see rem 1
q_2 → can see rem 2

show it how we did behind hand run



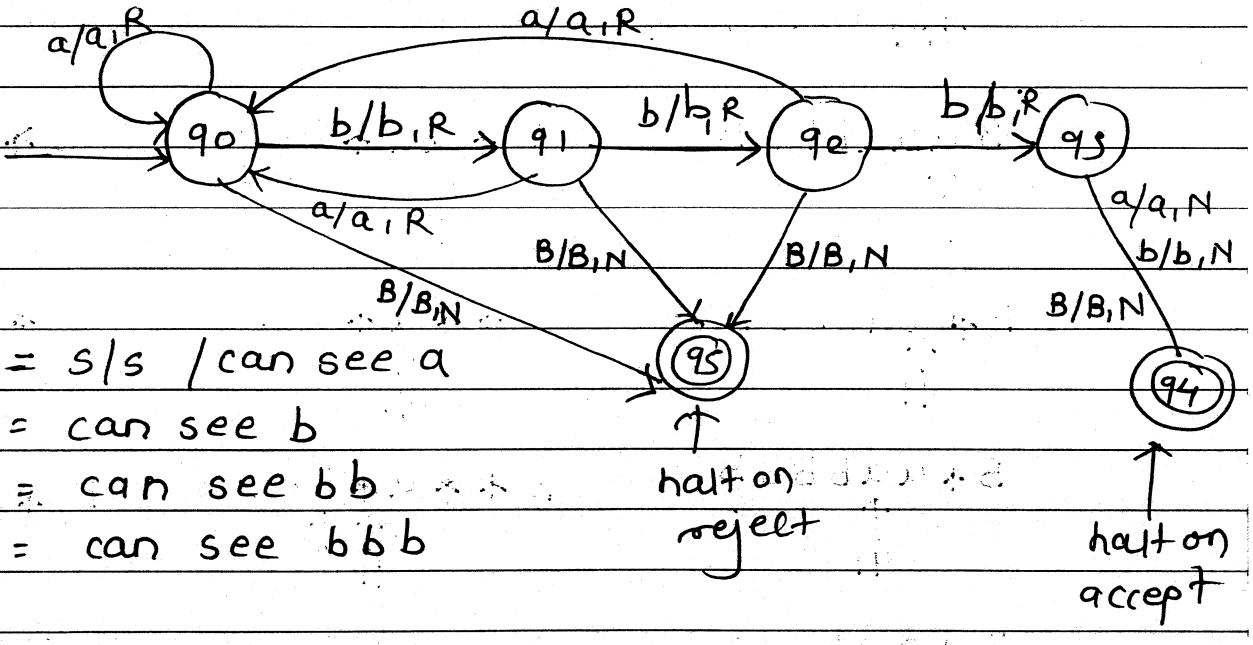
	0	1	B
q_0	$(q_1, 0, R)$	$(q_1, 1, R)$	(q_3, B, N)
q_1	$(q_2, 0, R)$	$(q_2, 1, R)$	(q_4, B, N)
q_2	$(q_0, 0, R)$	$(q_0, 1, R)$	(q_4, B, N)
q_3	-	-	- Halt on Accept
q_4	-	-	- Halt on reject

As it is FA problem, which symbol is reading, write back it & move to right [a/a,R]

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

All string with substring bbb

$$L = \left\{ bbb, \underline{a}bbb, bb\underline{a}b, \underline{b}bbb, \ldots \right\}$$



q0 = s/s / can see a

q1 = can see b

q2 = can see bb

q3 = can see bbb

a a a b b b

Imp

Q. Design Turing machine to accept following language over $\Sigma = \{a, b\}$

$L = \{a^n b^n \mid n \geq 0\}$

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

$S \longrightarrow aSb \mid \varepsilon$

Approach 1

B a a a b b b B	B * a a * b b B	B * * a * * b B
↑ q_0	↑ q_0	↑ q_0
B * a a b b b B	B * * a * b b B	B * * * * * b B
↑ q_1	↑ q_1	↑ q_1
B * a a b b b B	B * * a * b b B	B * * * * * b B
↑ q_1	↑ q_1	↑ q_1
B * a a b b b B	B * * a * b b B	B * * * * * b B
↑ q_1	↑ q_1	↑ q_1
B * a a * b b B	B * * a * * b B	B * * * * * * B
↑ q_2	↑ q_2	↑ q_2 ----- ↑ q_2
B * a a * b b B	B * * a * * b B	
↑ q_3	↑ q_2	B * * * * * * B
B * a a * b b B	B * * a * * b B	↑ q_4 ----- ↑ q_4
↑ q_3	↑ q_3	↑ q_5
		halt on Accept

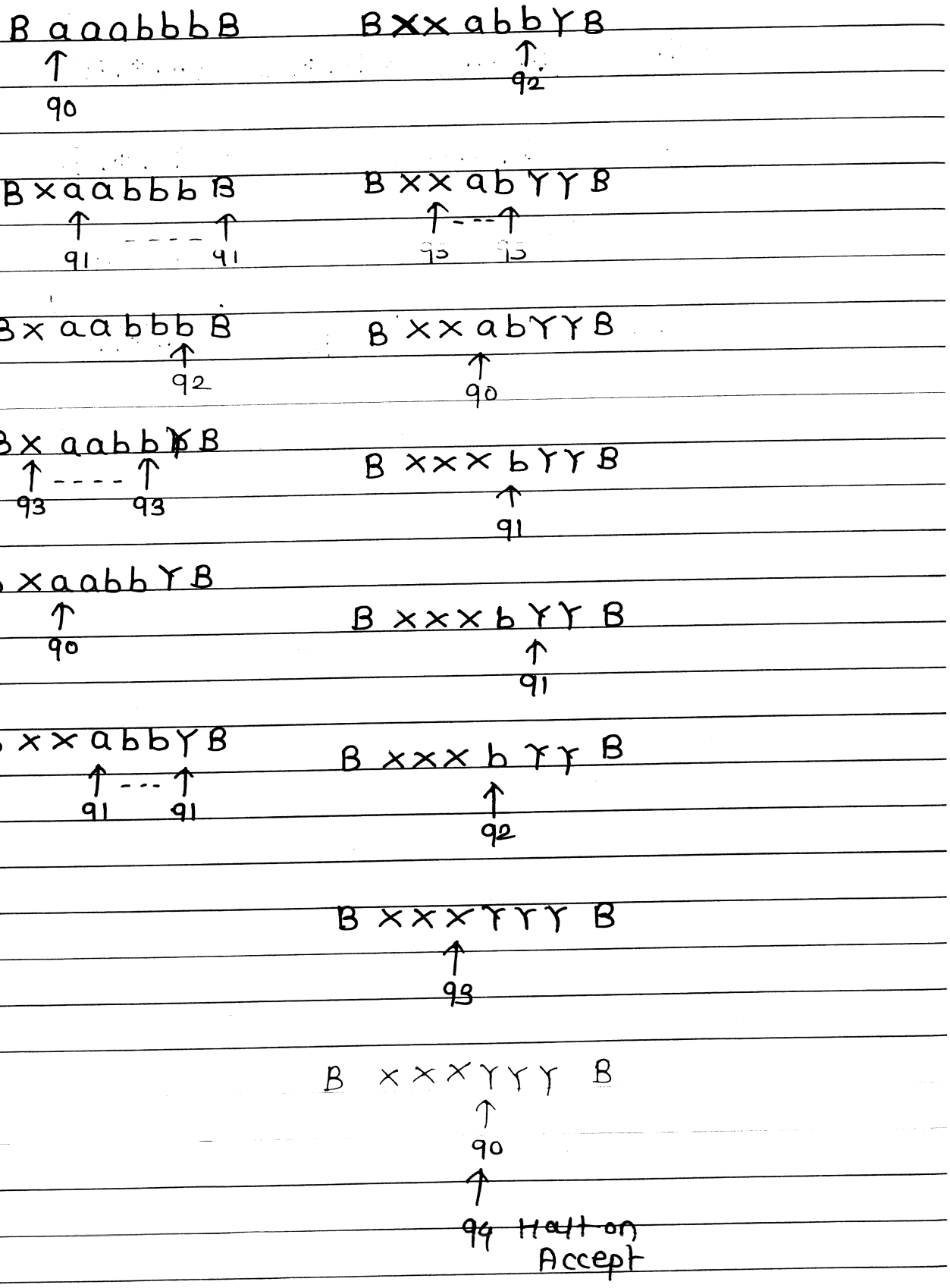
- (No chance to occur)

q_0 - B - (q_5, B, N) — Halt on Accept as ε is there

	a	b	*	B
q_0	$(q_1, *, R)$	R	-	(q_3, B, N)
q_1	(q_1, a, R)	$(q_2, *, L)$	$(q_1, *, R)$	R
q_2	(q_3, a, L)	-	$(q_2, *, L)$	(q_4, B, R)
q_3	(q_3, a, L)	-	$(q_0, *, R)$	-
q_4	-	R	$(q_4, *, R)$	(q_5, B, N)
q_5	-	-	-	— Halt on Accept

aaabbb

Approach II



	a	b	X	Y	B
q_0	(q_1, X, R)	R	–	(q_4, Y, N)	(q_4, B, N)
q_1	(q_1, a, R)	(q_1, b, R)	–	(q_2, Y, L)	(q_2, B, L)
q_2	–	(q_3, Y, L)	R	–	–
q_3	(q_3, a, L)	(q_3, b, L)	(q_0, X, R)	–	–
q_4	–	–	–	–	– Halt on Accept

q_0 - look for 1st a & make X

q_1 - look for 2nd a & make X and control transfer to q_2

q_2 - go upto B and control gives to q_3 to b

Q. $L = \{a^{2n} b^n \mid n \geq 0\}$

$L = \{\varepsilon, aab, aaaabb, aaaaaabbb \cdots\}$

B a a a a b b B B X X X a b Y B
↑ q_0 ↑ q_1

B X a a a b b B B X X X X b Y B
↑ q_1 ↑ q_2

B X X a a b b B B X X X X b Y B
↑ q_2 ----- ↑ q_2 ↑ q_2

B X X a a b b B B X X X X b Y B
↑ q_3 ↑ q_3

B X X a a b Y B B X X X X Y Y B
↑ q_4 ---- ↑ q_4 ↑ q_4

B X X a a b Y B B X X X X Y Y B
↑ q_0 ↑ q_0
↑ q_5 Halt on Accept

N - don't move anywhere

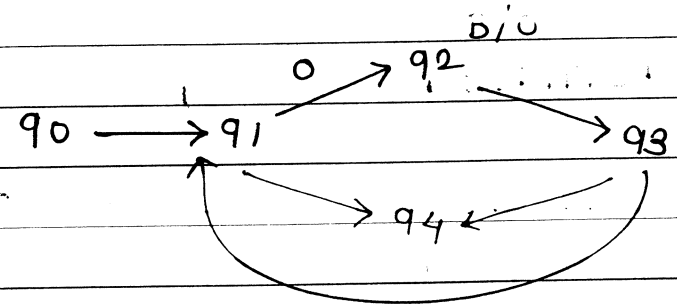
	a	b	X	Y	B
q_0	(q_1, X, R)	–	–	(q_5, Y, N)	(q_5, B, N)
q_1	(q_2, X, R)	–	–	–	–
q_2	(q_2, a, R)	(q_2, b, R)	–	(q_3, Y, L)	(q_3, B, L)
q_3	–	(q_4, Y, L)	–	–	–
q_4	(q_4, a, L)	(q_4, b, L)	(q_0, X, R)	–	–
q_5	–	–	–	–	–

~~Halt on~~ Accept

Q. Design a turing machine to shift given binary number towards right by 1 bit position

B 1 0 1 0 B

B B 1 0 1 0



B 1 0 1 0 B
↑ ---- ↑
q0 q0

B 1 0 B B 0
↑
q4

B 1 B 0 1 0
↑
q1

B 1 0 1 0 B
↑
q1

B 1 0 B 1 0
↑
q3

B B B 0 1 0
↑
q4

B 1 0 1 B B
↑
q2

B 1 0 B 1 0
↑
q1

B B B 0 1 0
↑
q3

B 1 0 1 B 0
↑
q3

B 1 B B 1 0
↑
q2

B B 1 0 1 0
↑
q1

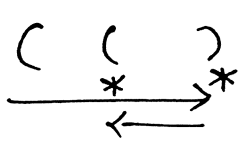
B 1 0 1 B 0
↑
q1

B 1 B 0 1 0
↑
q3

↑
q5 - Halt on Accept

	0	1	B	
q_0	$(q_0, 0, R)$	$(q_0, 1, R)$	(q_1, B, L)	
q_1	(q_2, B, R)	(q_4, B, R)	(q_5, B, N)	
q_2	-	-	$(q_3, 0, L)$	
q_3	-	-	(q_1, B, L)	
q_4	-	-	$(q_3, 1, L)$	
q_5	-	-	-	Halt on Accept

checking for first closing mark it *
then back traverse, find 1st opening, mark it as *



imp Q. Design Turing machine to accept well formness of parenthesis

B (( )) ( ) B
↑---↑
q0 q0

B ((*) ( ) B
↑
q1

B (* *) ( ) B
↑--↑
q0 q0

B (* * * ( ) B
↑....↑
q1 q1

B * * * * ( ) B
↑ -----↑
q0 q0

B * * * * (* B
↑
q1

B * * * * * * B
↑---↑
q0 q0

B * * * * * * B
↑ ---------↑
q2 q2
↑
q3 Halt on Accept

	(	)	*	B
→ q_0	(q_0, C, R)	$(q_1, *, L)$	$(q_0, *, R)$	(q_2, B, L)
q_1	$(q_0, *, R)$	–	$(q_1, *, L)$	R
q_2	R	–	$(q_2, *, L)$	$(q_3, B, N$
q_3	–	–	–	Halt on Accept

Q. $L = \{ a^n b^n c^n \mid n \geq 0 \}$