FLAT

~~[illegible]~~

- Conjuction $(p \wedge q)$ TT → T
- Disjunction $(p \vee q)$ FF → F
- Conditional $(p \to q)$ ~~[illegible]~~ TF → F
- Codomain = Range → Onto function

Q Prove the funⁿ $f: R \to R$, given by $f(x) = 2x$ is one-one and onto

Sol For one-one :-

$f(x_1) = 2x_1$

$f(x_2) = 2x_2$

Putting $f(x_1) = f(x_2)$

$2x_1 = 2x_2$

$x_1 = x_2$

Hence If $f(x_1) = f(x_2)$, $x_1 = x_2$

∴ function is one one

for onto :-

$f(x) = y$, such that $y \in R$

$2x = y$

$x = y/2$

∵ y is real no, Hence $y/2$ is also real no.. so, x will also be real no.

Hence, function is onto

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

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

∴ $\sqrt{2} = a/b$ where $b \neq 0$, a, b have no common divisor than 1

$2 = (a/b)^2$

$a^2 = 2b^2$

a^2 = even —(∵ multiple of 2)

then a = even

Let $a = 2k$

$a^2 = 4k^2$

$2b^2 = 4k^2$ $(\because a^2 = 2b^2)$

$b^2 = 2k^2$

~~Codomain = Range → onto~~

b^2 = even (∵ multiple of 2)

then b = even

a & b both are even and they have common divisor 2.

so, assumption is wrong

∴ $\sqrt{2}$ is irrational no.

Hence proved..

Q. For any integer a and b, if a and b are odd, then ab is odd.

Sol An integer a is odd if it can be written in the form:

$a = 2m+1$, m → integer

similarly, $b = 2n+1$

consider the product of a and b

$ab = (2m+1)(2n+1)$

$= (2m \cdot 2n) + 2m + 2n + 1$

$= 4mn + 2m + 2n + 1$

$= 2(2mn + m + n) + 1$

Notice that $2mn+m+n$ is integer becz m & n are integer, and sum and product of integers are also integer

∴ Let $k = 2mn + m + n$

∴ $ab = 2k+1$

Thus, we have shown the product of two odd integer is always odd.

Q. PMI

Prove: $\sum_{k=1}^{n} \frac{1}{k(k+1)} = \frac{n}{n+1}$

for every $k \geq 1$

Sol STEP 1 : BASIC STEP

we must show than $P(1)$ is true

LHS : $P(1) = \frac{1}{1(1+1)} = 1/2$

RHS : $\frac{n}{n+1} = \frac{1}{1+1} = 1/2$

Hence proved $P(1)$ is true.

STEP 2 : Induction Hypothesis

$K \geq 1$

$$P(K) = \frac{1}{2} + \frac{1}{\cancel{6}} + \dots + \frac{1}{K(K+1)} = \frac{K}{K+1}$$

STEP 3 : Proof of Induction

$$P(K+1) = \frac{1}{2} + \frac{1}{8} + \dots + \frac{1}{K(K+1)} + \frac{1}{(K+1)((K+1)+1)}$$

$$= \frac{K}{K+1} + \frac{1}{(K+1)((K+1)+1)}$$

$$= \frac{1}{K+1}\left(K + \frac{1}{(K+1)+1}\right)$$

$$= \left(\frac{1}{K+1}\right)\left(\frac{K^2+2K+1}{K+2}\right)$$

$$= \frac{1}{K+1} \cdot \frac{K^2+K+K+1}{K+2}$$

$$= \frac{1}{\cancel{K+1}} \cdot \frac{\cancel{(K+1)}(K+1)}{K+2}$$

$$= \frac{K+1}{K+2}$$

$$= \frac{(K+1)}{((K+1)+1)}$$

Hence proved.

Regular Expression: is a sequence of character that defines a search pattern for text.

- String do not end with 01

  $(0+1)^*(00+10+11)$

- String with odd no. of 1

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

- String with even no. of 1

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

- String begin or ends with 00 or 11

  $[(00+11)(0+1)^*] + [(0+1)^*(00+11)]$

- String containing both 101 and 010

  $(0+1)^*(101(0+1)^*010 + 010(0+1)^*101)(0+1)^*$

- String with length of 6 or less

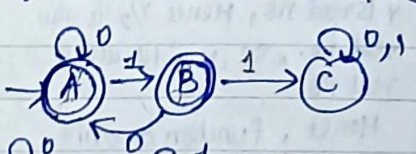
  $(0+1+\varepsilon)^6$

Q. Draw FA for

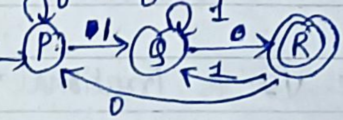
$L_1 = \{x / x$, 11 is not substring of x, $x \in \{0,1\}^*\}$

$L_2 = \{x / x$, end with 10, $x \in \{0,1\}^*\}$

Sol $L_1 =$



$L_2 =$



Here, $Q_1 = \{A, B, C\}$ $A_1 = \{A, B\}$

$Q_2 = \{P, Q, R\}$ $A_2 = \{R\}$

For $Q = Q_1 \times Q_2$

$= \{AP, AQ, AR, BP, BQ, BR, CP, CQ, CR\}$

$\delta((A,P),0) = AP$ $\delta((A,R),0) = AP$

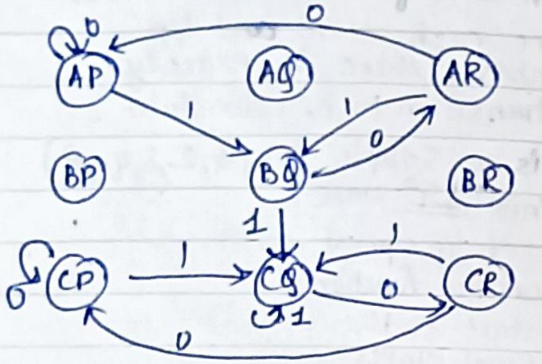
$\delta((A,P),1) = BQ$ $\delta((A,R),1) = BQ$

$\delta((B,Q),0) = AR$ $\delta((C,Q),0) = CR$

$\delta((B,Q),1) = CQ$ $\delta((C,Q),1) = CQ$

$\delta((C,R),0) = CP$ $\quad$ $\delta((C,P),0) = CP$

$\delta((C,R),1) = CQ$ $\quad$ $\delta((C,P),1) = CQ$

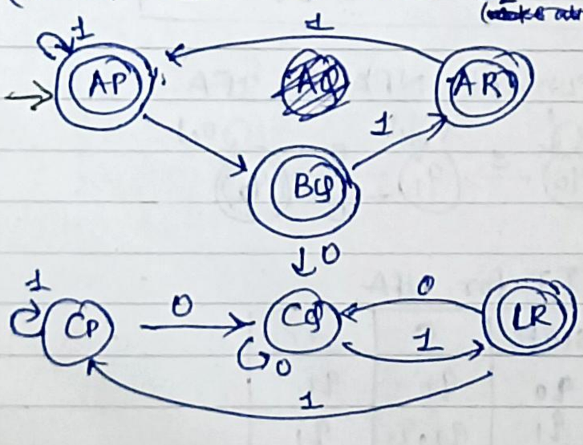


(i) ~~$L_1 \cup L_2$~~ ~~$L_1 \to$ final~~ ~~$L_2 \to$ all state~~

~~$L_1 = \{A, B\}$~~ ~~$L_2 = \{P, Q, R\}$~~

	0	1
AP	BQ	AP
BQ	CQ	AR
CQ	CQ	CR
AR	BQ	AP
CR	CQ	CP
CP	CQ	CP

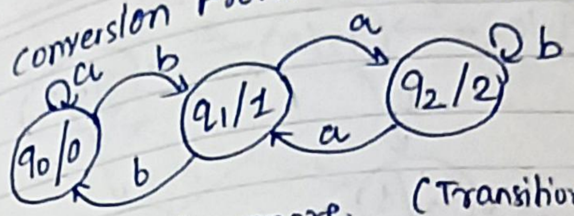
(i) $L_1 \cup L_2$ $\quad$ $L_1 \to$ final $\quad$ $L_2 \to$ all state



(ii) $L_1 \cap L_2$ $\quad$ $L_1 \to$ final $\quad$ $L_2 \to$ final

(iii) $L_1 - L_2$ $\quad$ $L_1 \to$ final $\quad$ $L_2 \to$ non final

Q. Conversion from moore to mealy

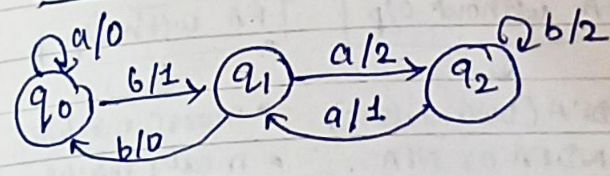


Sol. TT of moore (Transition Table)

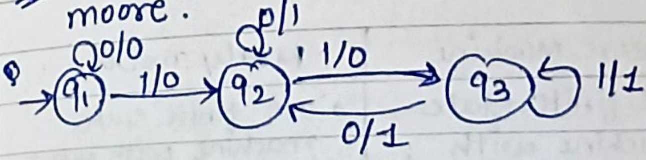
Current state	I/P a	b	O/P
q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2

TT of mealy.

Current state	a		b	
	state	O/P	state	O/P
q_0	q_0	0	q_1	1
q_1	q_2	2	q_0	0
q_2	q_1	1	q_2	2

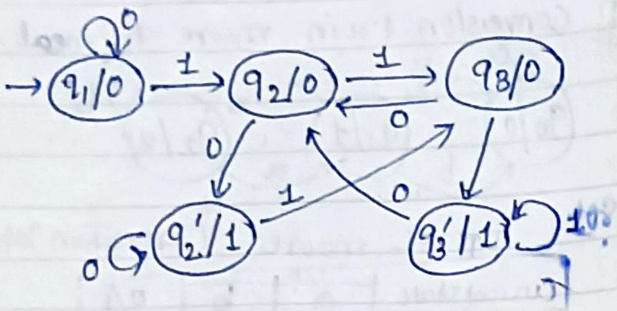


Q. Conversion from mealy to moore.



Sol. TT of mealy

Current state	0		1	
	state	O/P	state	O/P
q_1	q_1	0	q_2	0
q_2	q_2	1	q_3	0
q_3	q_2	1	q_3	1



TT of moore

State	0	1	O/P
q_1	q_1	q_2	0
q_2	q_3	q_2'	0
q_3	q_2	q_3'	0
q_2'	q_2'	q_3	1
q_3'	q_2	q_3'	1

Finite Automata

FA without o/p
- DFA (Deterministic)
- NDFA or NFA
- E-NFA or ENDFA

FA with o/p
- moore machine
- mealy machine

Moore machine	Mealy machine
are finite state machine with output value and its output depends only on the present state	are finite state machine with output value and its o/p depends on the present and current i/p symbol.
i/p changes o/p does not changes	i/p changes, o/p also changes
It has more states than mealy	Less state than moore
React slower than mealy	React faster than moore
n+1 o/p length	n o/p length

DFA (Deterministic Finite Automata)

→ A DFA is a finite state machine where each state and i/p symbol, there is exactly on transition to a new state

→ DFA is a 5 tuple $D = (Q, \Sigma, \delta, q_0, F)$

$Q \rightarrow$ finite set of state
$\Sigma \rightarrow$ set of i/p symbol
$\delta \rightarrow$ transition function
$q_0 \rightarrow$ start state
$F \rightarrow$ final states

$$\boxed{\delta : Q \times \Sigma \rightarrow Q}$$

NFA (Non Deterministic Finite Automata)

→ An NFA is a finite state machine where each state and i/p symbol, there can be multiple possible transition.

→ Additionally, it can have epsilon (ϵ) transition, which allow the machine to change states without consuming any i/p symbol.

→ NFA is a 5 tuple $N = (Q, \Sigma, \delta, q_0, F)$

$$\boxed{\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^Q}$$

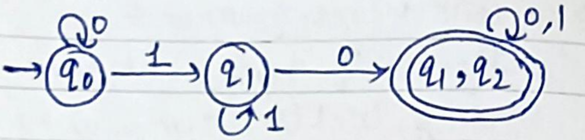
Q Convert NFA to DFA

Sol TT for NFA

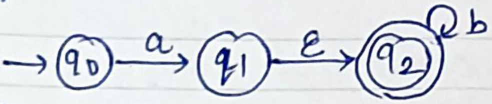
state	0	1
q_0	q_0	q_1
q_1	q_1, q_2	q_1
q_2	q_2	q_1, q_2

TT for DFA

State	0	1
q_0	q_0	q_1
q_1	q_1, q_2	q_1
q_1, q_2	$q_1 q_2$	$q_1 q_2$



Q convert NFA-Λ to NFA



Sol

TT for NFA-Λ

State	i/p a	b
q_0	q_1	ϕ
q_1	ϕ	ϕ
q_2	ϕ	q_2

ε-closure of $q_0 = q_0$

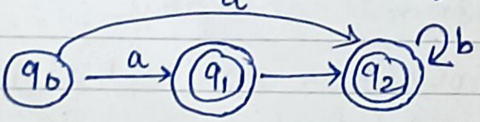
ε-closure of $q_1 = q_1 q_2$

ε-closure of $q_2 = q_2$

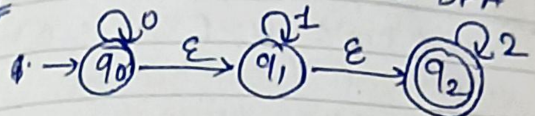
TT for NFA

State	a	b
q_0	$q_1 q_2$	ϕ
q_1	ϕ	q_2
q_2	ϕ	q_2

$\delta(q_0, a) = \varepsilon \text{ closure } ((\delta(\varepsilon \text{closure} q_0), a))$



Q convert NFA-Λ to DFA



Sol TT for NFA-Λ

State	0	1	2
q_0	q_0	ϕ	ϕ
q_1	ϕ	q_1	ϕ
q_2	ϕ	ϕ	q_2

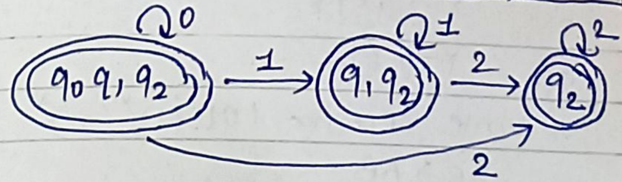
ε-closure of $q_0 = \{q_0, q_1, q_2\}$

ε-closure of $q_1 = \{q_1, q_2\}$

ε-closure of $q_2 = \{q_2\}$

TT for DFA

State	0	1	2
$q_0 q_1 q_2$	$q_0 q_1 q_2$	$q_1 q_2$	q_2
$q_1 q_2$	ϕ	$q_1 q_2$	q_2
q_2	ϕ	ϕ	q_2



DFA	NFA
• Exactly one transition for each state and i/p symbol	• multiple transition for each state and i/p symbol
• Epsilon transition not allowed	• Epsilon transition allowed
• $\delta: Q \times \Sigma \rightarrow Q$	• $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$
• Generally simpler and more efficient	• can be less efficient used more in theoretical contexts.
• Conversion: Not applicable (DFA is already deterministic)	• Can be converted to an equivalent DFA
• Only one active state at a time	• Multiple active state are possible simultaneously
• All DFA are NFA	• All NFA are not DFA
• Backtracking is allowed	• Not always possible in NFA

Context free Grammar (CFG)

→ is a 4 tuple $G = (V, \Sigma, S, P)$

V → Finite set of non-terminal

Σ → Finite set of terminal

S → element of V and its start symbol

P → Production rule.

→ Application :-

- used to specify the syntax of programming language
- used to develop a parser

Write CFG for following

(1.) write CFG for ab^*

$S \rightarrow aX$

$X \rightarrow \epsilon / bX$

(2.) write CFG for a^*b^*

$S \rightarrow XY$

$X \rightarrow aX / \epsilon$

$Y \rightarrow bY / \epsilon$

(3.) write CFG for $(011+1)^*(01)^*$

$S \rightarrow AB$

$A \rightarrow 011A \mid 1A \mid \epsilon$

$B \rightarrow 01B \mid \epsilon$

(4) write CFG for $L = \{a^i b^j c^k \mid i=j \text{ or } j=k\}$

for $i=j$

$S \rightarrow AB$

$A \rightarrow aAb \mid ab$

$B \rightarrow cB \mid c$

for $j=k$

$S \rightarrow CD$

$C \rightarrow aC \mid a$

$D \rightarrow bDc \mid bc$

(5) write CFG for $L = \{a^i b^j c^k \mid j > i+k\}$

$S \rightarrow ABC$

$A \rightarrow aAb \mid \epsilon$

$B \rightarrow bB \mid b$

$C \rightarrow bCc \mid \epsilon$

- Types of Derivation & Ambiguity
  1. Left most derivation
  2. Right most derivation

An ambiguous CFG

→ A context free grammar G is ambiguous if there is at least on string in L(G) (language) having two or more distinct derivation tree

Q. Prove that given grammar is ambiguous

$S \rightarrow S+S \mid S-S \mid S*S \mid S/S \mid (S) \mid a$

string : $a+a+a$

Sol LMD:

$S \rightarrow S+S$

$S \rightarrow a+S$

$S \rightarrow a+S+S$

$S \rightarrow a+a+S$

$S \rightarrow a+a+a$

LMD:

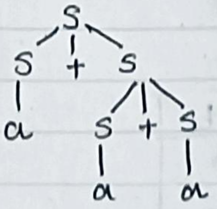
$S \rightarrow S+S$

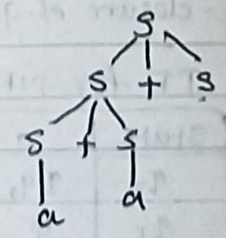
$S \rightarrow S+S+S$

$S \rightarrow a+S+S$

$S \rightarrow a+a+S$

$S \rightarrow a+a+a$





Here we have 2 left most derivation for string a+a+a hence, above grammar is ambiguous.

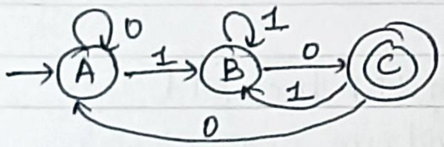
Q Prove that $S \rightarrow a \mid Sa \mid bSS \mid SbS \mid SSb$ is ambiguous

string: baaab

LMD	LMD
Sol $S \to bSS$	$S \to SSb$
$S \to baS$	$S \to bSSSb$
$S \to baSSb$	$S \to baSSb$
$S \to baaaSb$	$S \to baaSb$
$S \to baaab$	$S \to baaab$

∵ we have two left most derivation for string baaab hence, above grammar is ambiguous

Q Conversion from Finite Automata to Grammar:



Sol CFG :

$A \to 0A \mid 1B$

$B \to 1B \mid 0C \mid 0$

$C \to 1B \mid 0A$

Chomsky Normal Form (CNF)

→ A CFG is CNF if all production rule satisfy one of the following condition

- start symbol generating ϵ

  $A \to \epsilon$
- A non terminal generating two non terminal

  $S \to AB$
- A non terminal generating a terminal

  $S \to a$

Q conversion of CFG to CNF

$S \to aX \mid Yb$

$X \to S \mid \epsilon$

$Y \to bY \mid b$

Sol step1: Eliminate $\wedge$ production

Nullable variable = $\{X\}$

$S \to aX \mid Yb \mid a$

$X \to S$

$Y \to bY \mid b$

step2: Eliminate unit production

unit production is $X \to S$, new CFG without unit production

$S \to aX \mid Yb \mid a$

$X \to aX \mid a \mid Yb$

$Y \to bY \mid b$

Step 3: Replace all mixed string with solid NT

$S \to AX \mid YB \mid a$

$X \to AX \mid a \mid YB$

$Y \to BY \mid b$

$A \to a$

$B \to b$

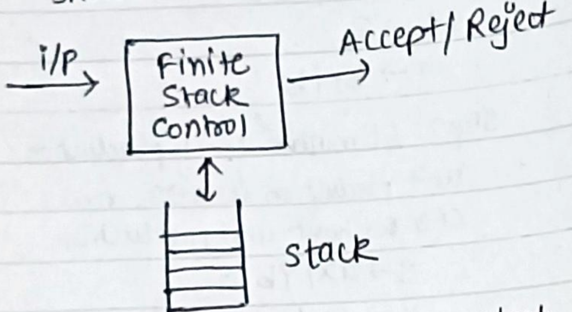
Step4: Shorten the string of NT to length 2

→ All NT string on the RHS in above CFG are already the required length. So CFG is in CNF

- PushDown Automata (PDA)

→ Is a computational model equivalent to context free grammar

→ A pushdown Automata is essentially a finite automata with a stack data structure as show



→ writing a symbol on the stack is called "PUSH" operation

→ ~~Remoing~~ Removing a symbol off the stack is called "POP" operation

→ PDA can accept only the stack top most symbol

→ is a 7 tuple

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

Q → Finite set of states

Σ & Γ → are finite set, the i/p and stack alphabet respectively

q_0 → initial state

Z_0 → Inital stack symbol

A → set of accepting state

δ → transition

$\delta : Q \times (\Sigma \cup \{\wedge\} \times \Gamma \rightarrow Q \times \Gamma^*$

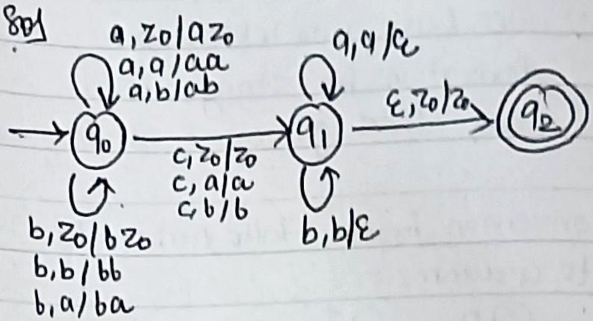
Q Design PDA for following CFG

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

or

Design PDA for palindrome with middle symbol C

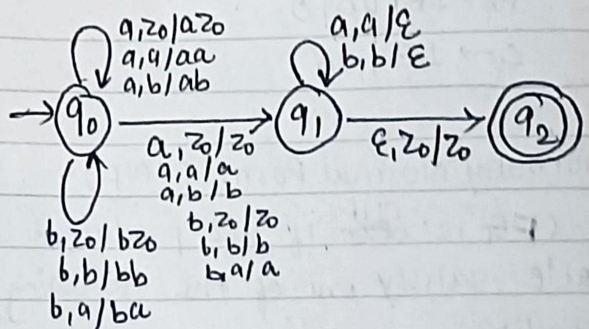
Sol



Q Design PDA for odd palindrom over {a, b}

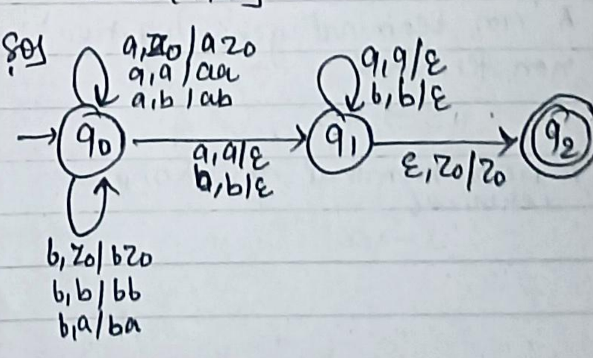
Sol

$\overbrace{abb}^{push}\ \overbrace{a}^{NO}\ \overbrace{bba}^{pop}$ or $\overbrace{abb}^{push}\ \overbrace{b}^{NO}\ \overbrace{bba}^{pop}$



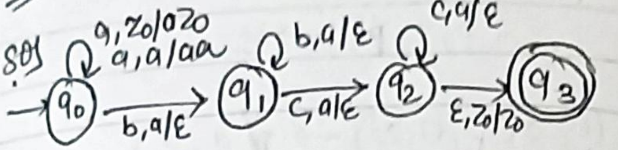
Q Design PDA for even palindrome over {a, b}

Sol



Q. Design PDA for $a^{m+n} b^m c^n$

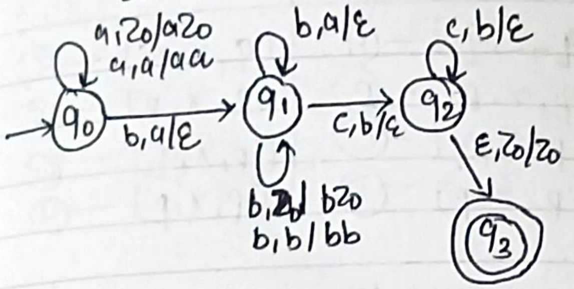
Sol



Q. Design PDA for $a^m b^{m+n} c^n$

Sol Let m=2 n=2

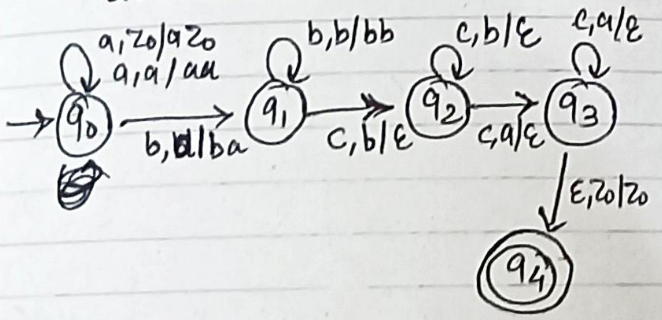
$\underbrace{aa}_{push}\underbrace{bb}_{pop}\underbrace{bb}_{push}\underbrace{cc}_{pop}$



Q. Design PDA for $a^n b^m c^{m+n}$

Sol Let m=2, n=2

$\underbrace{aa}_{push}\underbrace{bb}_{push}\underbrace{cccc}_{pop}$



Conversion from CFG to PDA

(variable) $A \to \beta$ (value)

Rule 1: for variable A

$\delta(q,\varepsilon,A) \to (q,\beta)$

Rule 2: for terminal a

$\delta(q,a,a) \to (q,\varepsilon)$

Q. Convert CFG to PDA

$S \to 0S1 / 00 / 11$

Sol Equivalent PDA for given grammar:

$\delta(q,\varepsilon,S) \to (q,0S1),(q,00),(q,11)$ — ①

$\delta(q,0,0) \to (q,\varepsilon)$ — ②

$\delta(q,1,1) \to (q,\varepsilon)$ — ③

Testing string 0111

$\delta(q,0111,S)$

$\delta(q,\cancel{0}111,\cancel{0}S1)$ — ①

$\delta(q,111,S1)$ — ②

$\delta(q,\cancel{1}11,\cancel{1}11)$ — ①

$\delta(q,11,11)$ — ③

$\delta(q,1,1)$ — ③

$\delta(q,\varepsilon,\varepsilon)$ — ③

Accept.

Q. Convert PDA to CFG

State	i/p	Stack	New State	Stack
q	1	Z	q	XZ
q	1	X	q	XX
q	∧	X	q	∧
q	0	X	p	X
p	1	X	p	∧
p	0	X	p	Z

Sol

$\delta: Q \times (\Sigma \cup \varepsilon) \times \Gamma \to Q \times \Gamma^*$

$\delta(q,1,z) \rightarrow (q,xz)$ ... push
$\delta(q,1,x) \rightarrow (q,xx)$ ... push
$\delta(q,\varepsilon,x) \rightarrow (q,\varepsilon)$ ... pop
$\delta(q,0,x) \rightarrow (p,x)$ ... no oper^n
$\delta(p,1,x) \rightarrow (p,\varepsilon)$ ... pop
$\delta(p,0,z) \rightarrow (q,z)$ ... no oper^n

$S \rightarrow [q,z,q]$
$S \rightarrow [q,z,p]$

① $\delta(q,1,z) \rightarrow (q,xz)$

$[q,z,q] \rightarrow 1[q,x,q][q,z,q]$
$[q,z,q] \rightarrow 1[q,x,p][p,z,q]$
$[q,z,p] \rightarrow 1[q,x,q][q,z,p]$
$[q,z,p] \rightarrow 1[q,x,p][p,z,p]$

② $\delta(q,1,x) \rightarrow (q,xx)$

$[q,x,q] \rightarrow 1[q,x,q][q,x,q]$
$[q,x,q] \rightarrow 1[q,x,p][p,x,q]$
$[q,x,p] \rightarrow 1[q,x,q][q,x,p]$
$[q,x,p] \rightarrow 1[q,x,p][p,x,p]$

③ $\delta(q,\varepsilon,x) \rightarrow (q,\varepsilon)$

$[q,x,q] \rightarrow \varepsilon$

④ $\delta(q,0,x) \rightarrow (p,x)$
$[q,x,q] \rightarrow 0[p,x,q]$
$[q,x,p] \rightarrow 0[p,x,p]$

⑤ $\delta(p,1,x) \rightarrow (p,\varepsilon)$

$[p,x,p] \rightarrow 1$

⑥ $\delta(p,0,z) \rightarrow (q,z)$

$[p,z,q] \rightarrow 0[q,z,q]$
$[p,z,p] \rightarrow 0[q,z,z]$

$[q,z,q]$	A	$[q,x,q]$	E
$[q,z,p]$	B	$[q,x,p]$	F
$[p,z,q]$	C	$[p,x,q]$	G
$[p,z,p]$	D	$[p,x,p]$	H

$\therefore S \rightarrow A \mid B$
$A \rightarrow 1EA \mid 1FC$
$B \rightarrow 1EB \mid 1FD$
$E \rightarrow 1EE \mid 1FG \mid 0G \mid \varepsilon$
$F \rightarrow 1EF \mid 1FH \mid 0H$