UNIT 3 IMPORTANT QUESTIONS

SHORT QUESTIONS

1 Define Grammar?

A: → Grammar is used to generate the entire language.
→ Grammar is an alternative way to specify language.
→ Grammars are useful in describing and analysing a language.
→ Grammar is indicated with the set of statements are called productions.
→ Grammar is represented by using 4 tupples
$G = (V, T, P, S)$

Scanned with CamScanner

V = finite set of variables of non-terminals
T = finite set of terminals
P = production rule
S = start symbol.

2 Discuss the formal definition of PDA.

Ai Formal definition of PDA:

It is defined as collection of 7 tuples.

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

$Q \rightarrow$ set of all states.
$\Sigma \rightarrow$ input symbols.
$\Gamma \rightarrow$ finite set of stack alphabets.
$\delta \rightarrow$ transition function.

$$Q \times (\Sigma \cup \epsilon) \times \Gamma \longrightarrow Q \times \Gamma^*$$

$q_0 \rightarrow$ initial state.
$Z_0 \rightarrow$ bottom of stack (OR) initial stack symbol
$Z_0 \in \Gamma$.
$F \rightarrow$ Final state.

Scanned with CamScanner

3. Explain left most derivation with an example?

A:- Left Most Derivation [LMD]:

A LMD is obtained by applying productions to the left most variable in each step

Example: Consider a grammar

$S \rightarrow aAB$

$A \rightarrow a$

$B \rightarrow b$

generate a string aab by LMD

$S \rightarrow aAB$

$S \rightarrow aaB$

$S \rightarrow aab$

4. What is an ambiguity?

A:- A Grammar is said to be Ambiguous grammar if there exist 2 (or) more derivation trees

[either Left most derivation tree (or) right most derivation tree] for a given string 'w'.

5. Construct Right most derivation tree with an example.

A: The Right most derivation tree is obtained by applying productions to the right most variable in each step.

Scanned with CamScanner

Ex: Consider a grammar

$S \rightarrow aAB$

$A \rightarrow a$

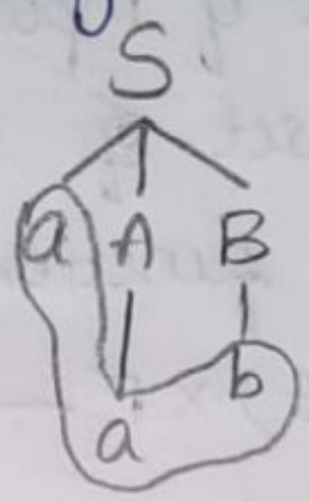
$B \rightarrow b$

generates a string aab by using RMD.

$S \rightarrow aAB$

$\rightarrow aAb$

$\rightarrow aab$



6. Define PDA

A: PDA is a way to implement CFG is the similar way to design FA for RG.

PDA is more powerful than FA.

FA has a very limited memory but PDA has a unlimited memory

PDA = FA + stack

PDA has the 3 components

1) An Input tape.

2) A Finite control unit.

3) A Stack with infinite memory.

The PDA can be accepted by 2 ways

1) Final state

2) Empty stack

Formal Definition of PDA:-

PDA formal Definition contains 7 tuples

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

Q = set of all states

Σ = finite set of input symbols

Γ = a finite set of stack alphabets

δ = Transition function

$$Q \times (\Sigma \cup \epsilon) \times \Gamma \longrightarrow Q \times \Gamma^*$$

q_0 = initial state

Z_0 = bottom of the stack (or) initial stack symbol

$Z_0 \in \Gamma$.

F = Final state.

7. For the grammar

{S → AS/a , A → SbA/SS/ba}

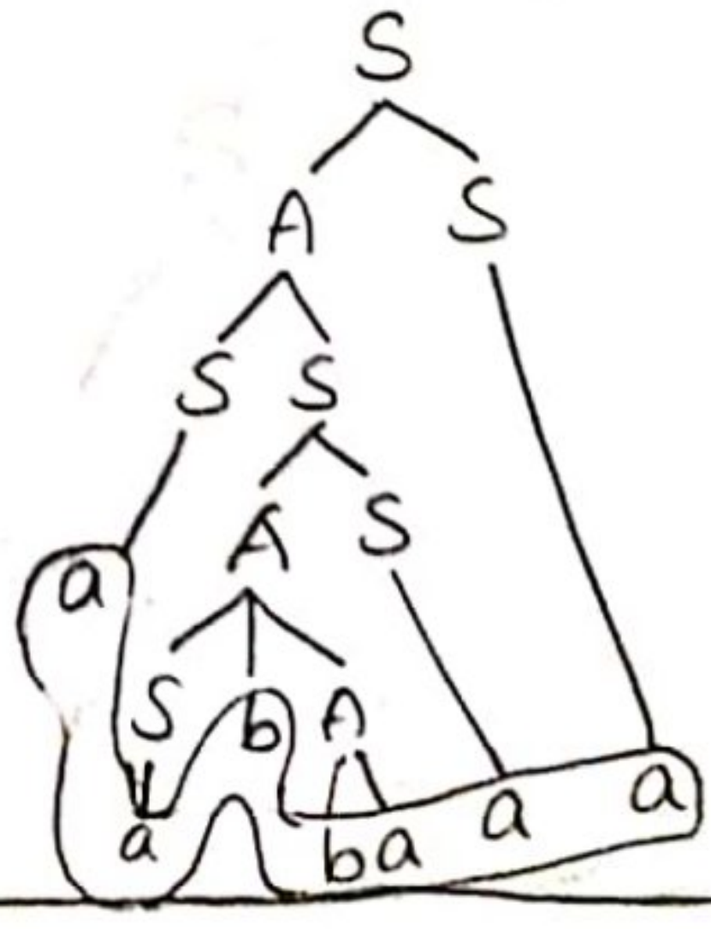
Construct left most derivation tree for the string "aabbaaa"

Ans Given Grammar

S → AS/a

A → SbA/SS/ba

Given string aabbaaa



Scanned with CamScanner

8 Compare DPDA and NPDA

Ans

DPDA	NDPDA
In DPDA, by seeing one input symbol. it will goes to the only 1 state and they will shaw the Deterministic	In NDPDA by seeing single input it will goes to the multiple states
Example: wcw^R $\delta(q_0, a, a) = \delta(q_1, aa)$	Examples: ww^R $\delta(q_0, a, a) = \delta(q_0, aa)$ $= \delta(q_1, \epsilon)$
There is only one move on every input	More than one move for single input
DPDA accepts regular language, context free language and is between RL, CFL	NDPDA accepts both RL and CFL.
DPDA is less powerfull.	NDPDA is more powerfull.
All languages are not accepted by the DPDA	All languages are accepted by the NDPDA

9 Find CFG with no useless symbols equivalent to

$S \rightarrow AS / A / C$

$A \rightarrow a$

$B \rightarrow aa$

$C \rightarrow aCb$

A:

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

$T = \{a, b\}$

Scanned with CamScanner

$S \to AS$ $S \to A$ $S \to C$

$S \to aaS$ $S \to a$ ✓ $S \to aCb$ ✗

$S \to aA$

$S \to aa$ ✓

CFG after removing useless symbols

$S \to AS/A$

$A \to a$

10 Define Non-deterministic PDA

A: Non-deterministic Push Down Automata contains 7-tuples

$(Q, \Sigma, \Gamma, \delta, q_0, z, F)$

where

Q = Set of all states

Σ = a finite set of input symbols

Γ = a finite set of stack alphabets

δ = Transition state function

$Q \times (\Sigma \cup \epsilon) \times \Gamma \longrightarrow$ finite subsets of $Q \times \Gamma^*$

q_0 = initial state

z_0 = Bottom of the stack

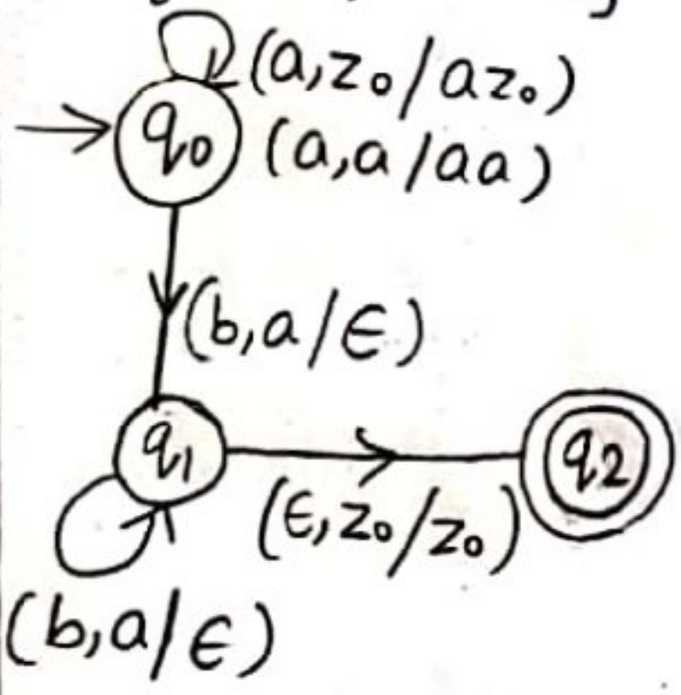
F = Final state

Scanned with CamScanner

LONG QUESTIONS

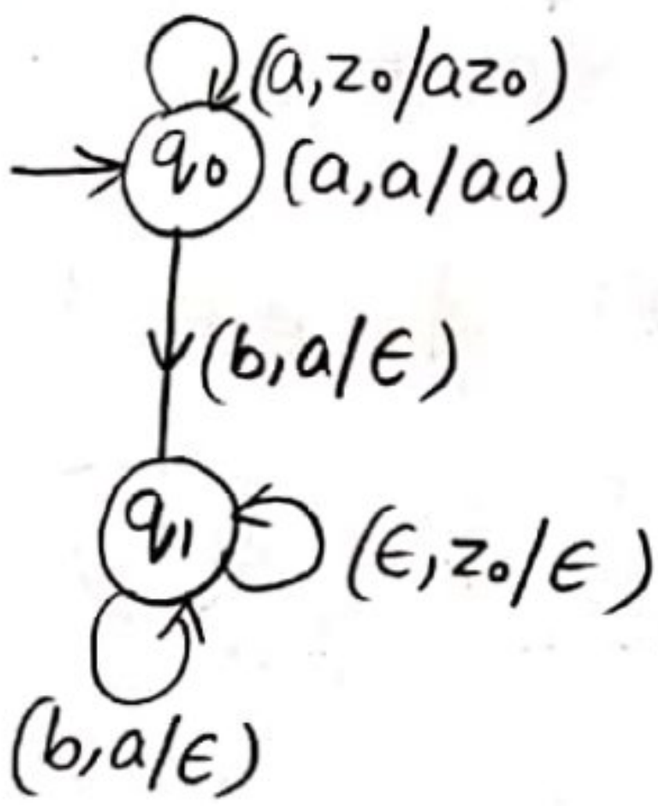
11 Construct a PDA for the given language $L = \{a^n b^n / n >= 1\}$ and write the instantaneous description for the string "aaabbb"

A: $L = \{a^n b^n / n >= 1\}$ aabb



Final state acceptance PDA

(OR)



Empty state acceptance PDA

Transition Function:

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

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

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

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

$\delta(q_1, \epsilon, z_0) = (q_2, z_0)$ (final state) (OR) $\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$ (empty state)

Instantaneous description of PDA:

aaabbb

ID = $\vdash (q_0, aaabbb, z_0)$

$\vdash (q_0, aabbb, az_0)$

$\vdash (q_0, abbb, aa)$

$\vdash (q_0, bbb, aa)$

$\vdash (q_1, bb, aa)$

$\vdash (q_1, b, az_0)$

$\vdash (q_1, \epsilon, z_0)$ final state

(or)

$\vdash (q_1, \epsilon)$ Empty state

12 a) Explain the process of simplification of the grammar

A: → All the grammars are not always optimized that means grammar may contains some extra symbols [Non-terminals & terminals].

→ Having extra symbols unnecessarily increases the length of the grammar.

→ Simplification of grammar means reduction of grammar by eliminating unnecessary symbols.

Conditions for simplifying of grammar:

To reduce the grammar we have to follow 3 procedures

i) Removal of useless symbols

ii) Removal of ϵ productions

iii) Removal of unit productions.

i) Removal of useless symbols:

→ Any symbol is useful when it appears on the right hand

side in the production rule and generates some terminal string if NO such derivation exists then it is supposed to be an useless symbol.

ii) Removal of ϵ productions:

→ To remove $A \to \epsilon$ look for all productions where right side contains 'A'.

→ replace each occurance of 'A' with ϵ.

→ Add the resultant productions to the grammar (After replacing A with ϵ).

iii) Removal of Unit productions:

→ The Unit productions are the productions in which one non-terminal gives another non-terminal.

Ex: $X \to Y$

$Y \to Z$

Procedure for removal:

→ Find out the unit productions in the given grammar.

→ To remove unit productions $A \to B$. Add productions $A \to x$ & $B \to x$ here x is the terminal.

→ Delete unit production $A \to B$.

→ repeat step-2 until all the unit productions are removed.

b) Explain left most and right most derivations with examples.

A: left most derivation [LMD]: A LMD is obtained by applying productions to the left most variable in each step.

Ex: $S \to aAB$

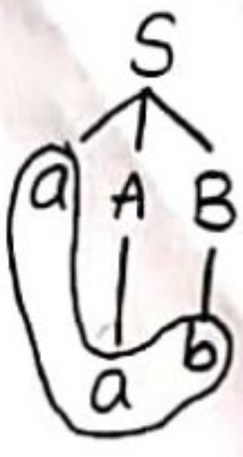
$A \to a$

$B \to b$

(or) $S \rightarrow aAB$
$\rightarrow aaB$
$\rightarrow aab$

Right most derivation [RMD]: A RMD tree is obtained by applying productions to the right most variable in each step

Ex: $S \rightarrow aAB$
$A \rightarrow a$
$B \rightarrow b$



(or) $S \rightarrow aAB$
$\rightarrow aAb$
$\rightarrow aab$

Scanned with CamScanner

13 a) Explain the process of removal of Null productions from the grammar

A: Removal of Null (or ϵ) productions:

→ In finite automata and regular expression that 'ϵ' indicates a string with no value.

→ Even in CFG if there is any 'ϵ' productions we can remove it without changing the meaning of the grammar. Thus, 'ϵ' productions are not necessary in a grammar.

Procedure for removal of null productions:

Step-I: To remove $A \rightarrow \epsilon$ look for all productions whose right side contains A.

Step-II: Replace each occurance of A with ϵ

Step-III: Add the resultant productions [After replacing A with ϵ] to the grammar.

b) Remove null productions from the following grammar

$S \rightarrow ABAC$

$A \rightarrow aA/\epsilon$

$B \rightarrow Bb/\epsilon$

$C \rightarrow c$

A: $S \rightarrow ABAC$

$S \rightarrow BAC$

$S \rightarrow ABC$

$S \rightarrow AC$

$S \rightarrow AAC$

$A \rightarrow \epsilon$

$B \rightarrow \epsilon$

$S \rightarrow C$

$A \rightarrow aA$

$B \rightarrow Bb$

$B \rightarrow b$

$A \rightarrow a$

$C \rightarrow c$

P': $S \rightarrow ABAC / BAC / ABC / AAC / BC / AC / C$

$A \rightarrow aA / a$

$B \rightarrow Bb / b$

$C \rightarrow C$

∴ P' is the grammar after removal of null productions

14 Construct a PDA that recognizes strings (over alphabet 'a' and 'b') that contains equal number of a's and b's. Write instantaneous description for the string "aabbabab".

A: Push Down Automata (PDA):

→ PDA is a way to implement CFG is the similar way to design FA for RG.

PDA = FA + Stack

→ PDA has the 3 components

① An Input tape

② A Finite control unit.

③ A Stack with infinite memory

→ The PDA can be accepted by 2 ways

① Final state

② Empty stack

Formal Definition of PDA:

PDA formal Definition contains 7 tuples

$(Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

Q = Set of all states

Σ = finite set of input symbols

Γ = a finite set of stack alphabets

δ = Transition function

$$Q \times (\Sigma \cup \epsilon) \times \Gamma \longrightarrow Q \times \Gamma^*$$

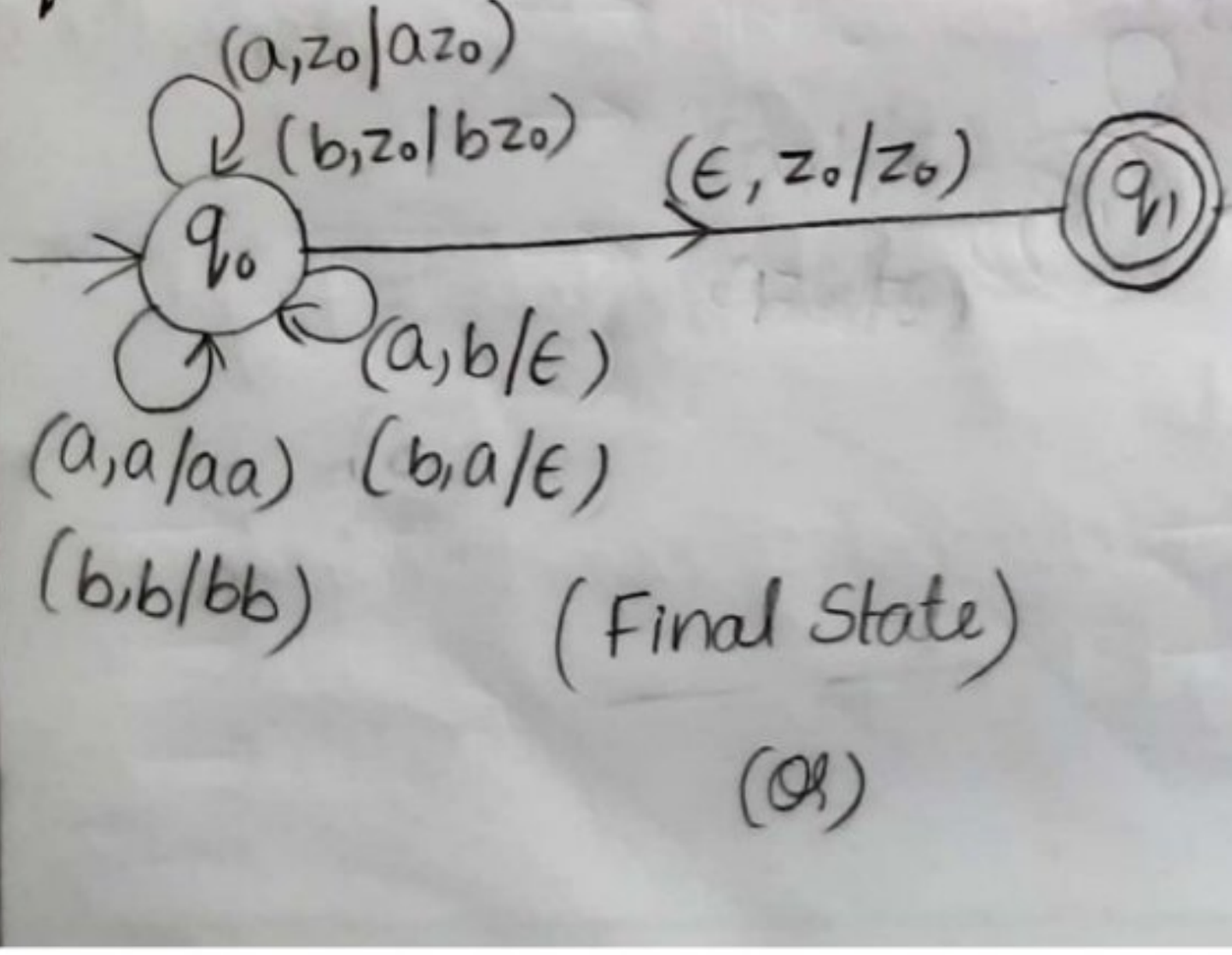
q_0 = initial state

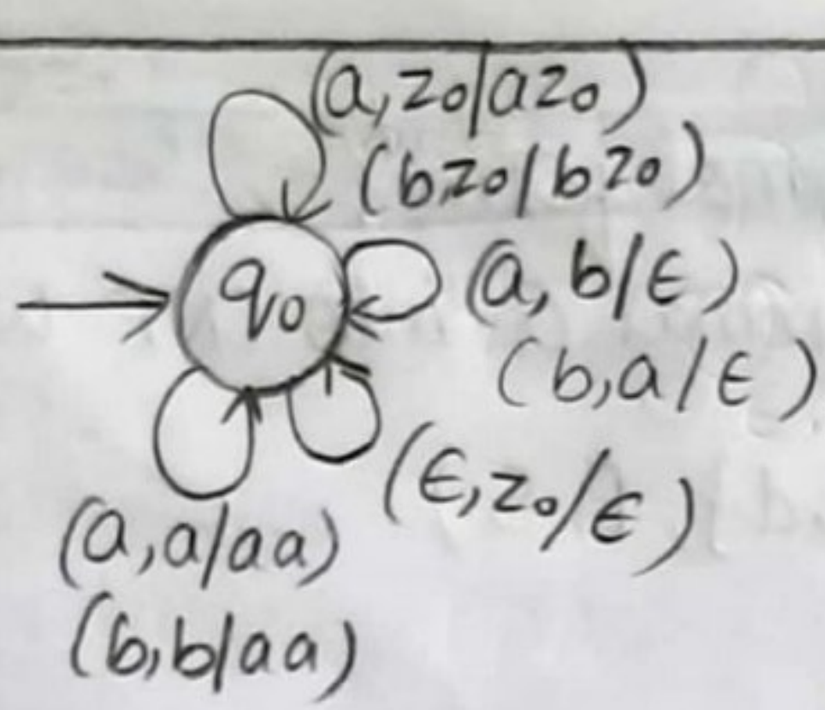
z_0 = bottom of the stack (or) initial stack symbol

$z_0 \in \Gamma$

F = Final state

PDA that recognizes strings with equal number of a's and b's:





(Empty stack)

Instantaneous Description of string "aabbabab":

$ID = \vdash (q_0, aabbabab, z_0)$

$= \vdash (q_0, abbabab, az_0)$

$= \vdash (q_0, bbabab, aa)$

$= \vdash (q_0, babab, az_0)$

$= \vdash (q_0, abab, z_0)$

$= \vdash (q_0, bab, az_0)$

$= \vdash (q_0, ab, z_0)$

$= \vdash (q_0, b, az_0)$

$= \vdash (q_0, \epsilon, z_0)$

$= \vdash (q_0, \epsilon) \rightarrow$ empty stack

(or)

$= \vdash (q_1, z_0) \rightarrow$ final state

15 Consider the grammar $E \rightarrow E+E \mid E*E \mid id$
Write the right most derivation and left most derivation for the sentence id*id+id. Discuss whether the given grammar is ambiguous (or) not

A: Ambiguous in Grammar:

A Grammar is said to be Ambiguous grammar If there exist 2 (or) more derivation trees [either left most derivation (or) right most derivation] a for a given string 'w'.

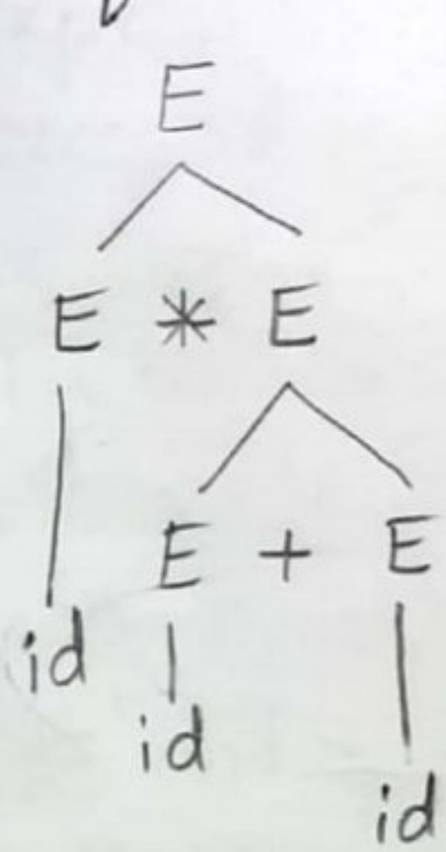
Given Grammar

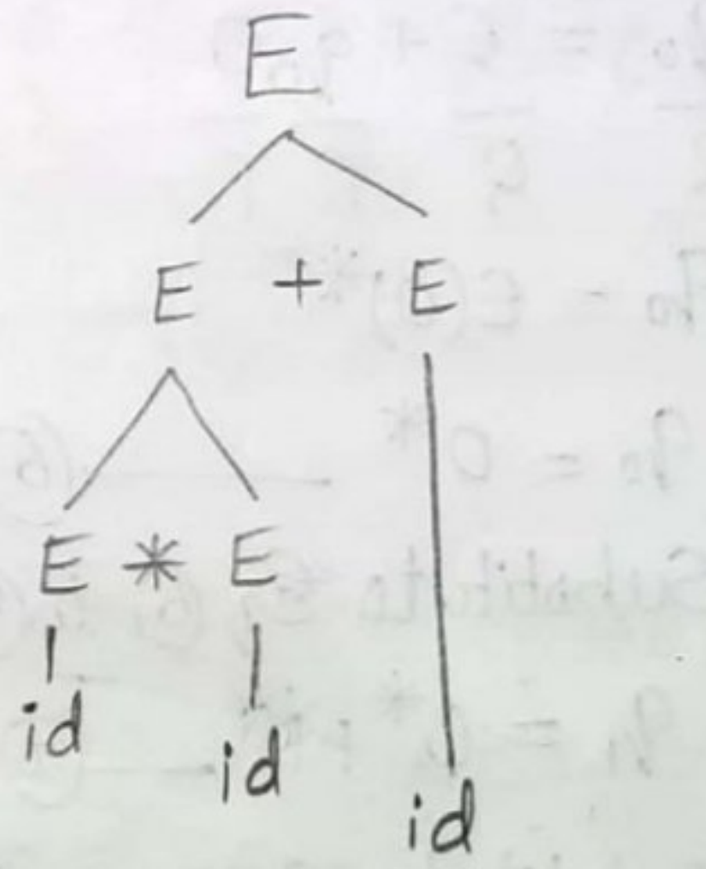
$E \rightarrow E+E$

$E \rightarrow E*E$

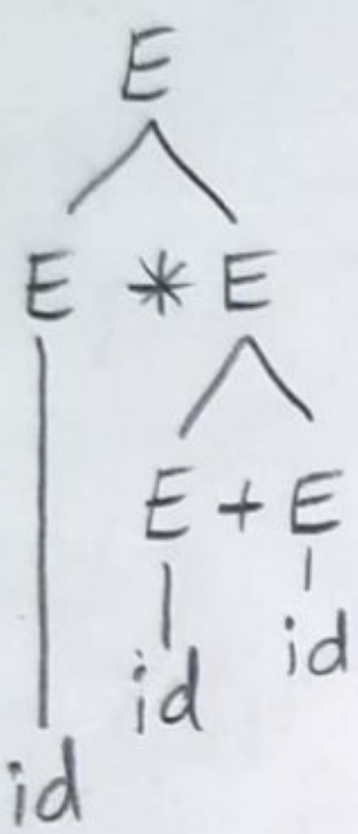
$E \rightarrow id$

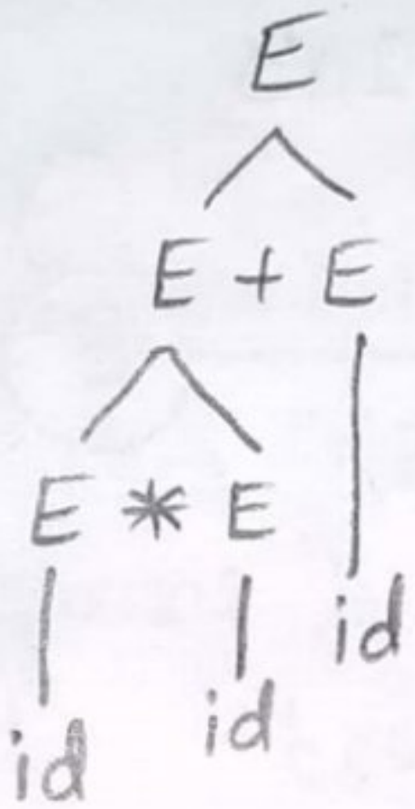
Left most Derivation trees:





Right most Derivation trees:





For the given string id *id +id has 2 LDT and 2RDT then, the given grammar is ambiguous

16. Define PDA. Construct a PDA for the given language $L=\{a^n b^{2n} / n>=1\}$ via Empty stack

A:- PDA is a way to implement CFG is the similar way to design FA for RG.

PDA is more powerful than FA.

FA has a very limited memory but PDA has a unlimited memory

PDA = FA + stack

PDA has the 3 components

1) An Input tape.
2) A Finite control unit.
3) A Stack with infinite memory.

The PDA can be accepted by 2 ways

1) Final state
2) Empty stack

Formal Definition of PDA:

PDA formal Definition contains 7 tuples

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

Q = set of all states

Σ = finite set of input symbols

Γ = a finite set of stack alphabets

δ = Transition function

$$Q \times (\Sigma \cup \epsilon) \times \Gamma \longrightarrow Q \times \Gamma^*$$

q_0 = initial state

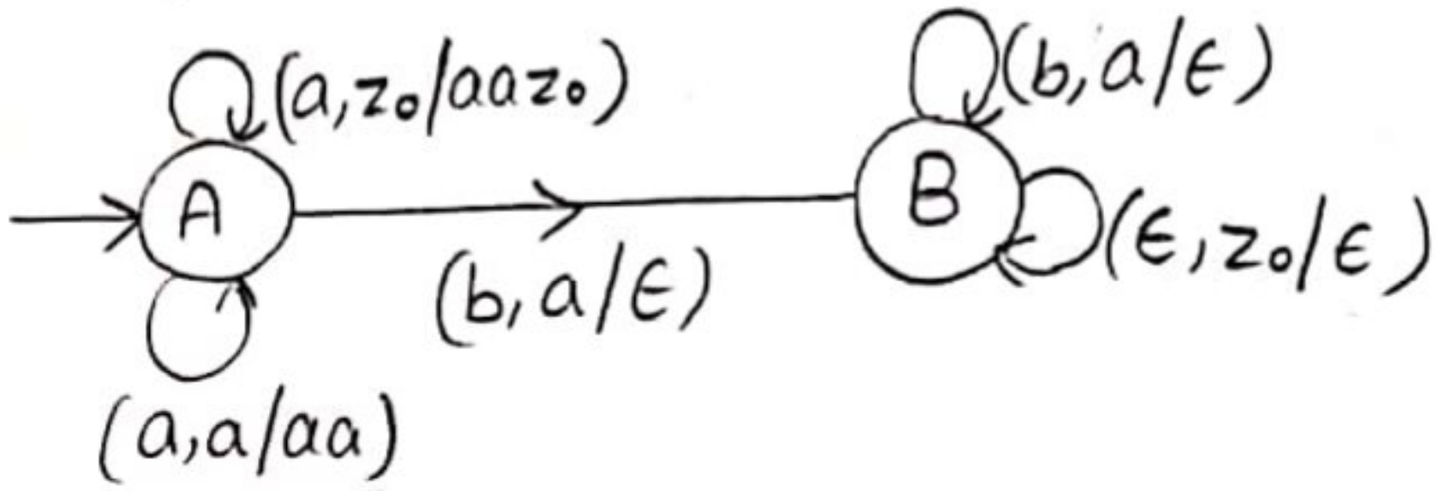
Z_0 = bottom of the stack (or) initial stack symbol

$Z_0 \in \Gamma$.

F = Final state.

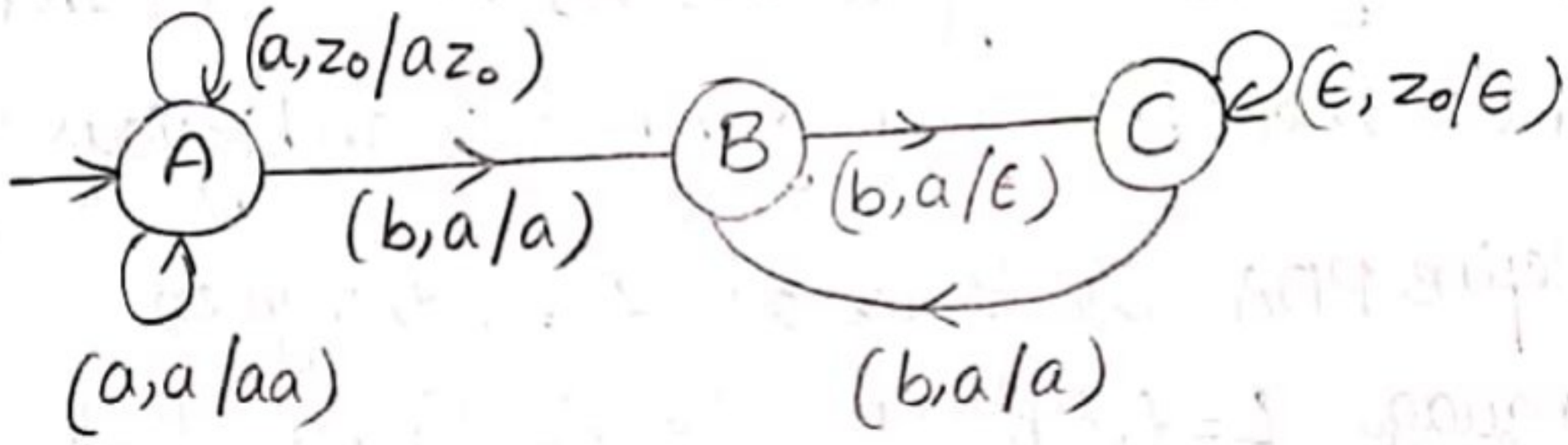
Given $L = \{a^n b^{2n} / n >= 1\}$

Case-1: a ⟶ 2a's



Case-2: b — read

b — delete corresponding



Scanned with CamScanner

17 Define Unit production. Explain the procedure to remove unit productions from the grammar. Remove Unit productions from the grammar

P: $S \rightarrow XY$

Scanned with CamScanner

$X \rightarrow a$

$Y \rightarrow z / b$

$z \rightarrow M$

$M \rightarrow N$

$N \rightarrow a$

Ans Unit production: The Unit productions are the productions in which one non-terminal gives the another non-terminal

Ex: $X \rightarrow Y$

Procedure to remove unit productions:-

Step-I:- find out the unit productions in the given gramma

Step-II:- To remove unit productions

$A \rightarrow B$, add productions $A \rightarrow x$ and $B \rightarrow x$

here 'x' is the terminal

Step-III:- Delete unit productions $[A \rightarrow B]$

Step-IV:- Repeat Step-II until all the unit productions are removed

Removing unit productions from the given grammas

$S \rightarrow XY$

$Y \rightarrow Z$

$Z \rightarrow M$

$M \rightarrow N$

$Y \rightarrow b/a$

$Z \rightarrow a$

$M \rightarrow a$

$X \rightarrow a$

$N \rightarrow a$

After removing unit productions and useless productions.

P':- $S \to XY$

$X \to a$

$Y \to a/b$

18. Construct CFG from PDA $A = (\{q_0, q_1\}, \{a, b\}, \{z_0, z\}, \delta, q_0, z_0, \phi)$

where $\delta : \delta(q_0, b, z_0) = (q_0, z z_0)$

$\delta(q_0, b, z) = (q_0, zz)$

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

$\delta(q_0, \epsilon, z_0) = (q_0, \epsilon)$

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

$\delta(q_1, a, z_0) = [q_0, z_0)$

1) Construction of set of Non-terminals

$V = \{S \cup [q_0, z_0, q_0] [q_0, z_0, q_1]$
$[q_0, z, q_0] [q_0, z, q_1]$
$[q_1, z_0, q_0] [q_1, z_0, q_1]$
$[q_1, z, q_0] [q_1, z, q_1]\}$

2) Construction of Production

i) S-Production

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

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

ii) For push operation

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

$q \;\; a \;\; z \qquad q_1 \;\; z_1 \;\; z_2$

$[q_0,z_0,q_0] \longrightarrow b[q_0,z,q_0][q_0,z_0,q_0]$

$[q_0,z_0,q_1] \longrightarrow b[q_0,z,q_0][q_0,z_0,q_1]$

$[q_0,z_0,q_0] \longrightarrow b[q_0,z,q_1][q_1,z_0,q_0]$

$[q_0,z_0,q_1] \longrightarrow b[q_0,z,q_1][q_1,z_0,q_1]$

$\delta(q_0,b,z) = (q_0,zz)$

$q \quad a \quad z \qquad q_1 \; z_1 \; z_2$

$[q_0,z,q_0] \longrightarrow b[q_0,z,q_0][q_0,z,q_0]$

$[q_0,z,q_1] \longrightarrow b[q_0,z,q_0][q_0,z,q_1]$

$[q_0,z,q_0] \longrightarrow b[q_0,z,q_1][q_1,z,q_0]$

$[q_0,z,q_1] \longrightarrow b[q_0,z,q_1][q_1,z,q_1]$

iii) Pop operation

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

$q \quad a \quad z \qquad q'$

$[q_1,z,q_1] \longrightarrow b$

$\delta(q_0,\epsilon,z_0) = (q_0,\epsilon)$

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

iv) NO Operation

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

$q \quad a \quad z \qquad q_1 \; z_1$

$(q_0,z,q_0) \longrightarrow a[q_1,z,q_0]$

$[q_0, z, q_1] \longrightarrow a[q_1, z, q_1]$

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

$q \quad a \quad z \qquad q_1 \quad z_1$

$[q_1, z_0, q_0] \Rightarrow a[q_0, z_0, q_0]$

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

These are the CFG in $\{V, T, P, S\}$

$V = \{S \cup [q_0, z_0, q_0] [q_0, z_0, q_1]$
$[q_0, z, q_0] [q_0, z, q_1]$
$[q_1, z_0, q_0] [q_1, z_0, q_1]$
$[q_1, z, q_0] [q_1, z, q_1]\}$

$T = \{a, b\}$

$S \rightarrow$ start symbol

production rules

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

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

$[q_0, z_0, q_0] \longrightarrow b [q_0, z, q_0][q_0, z_0, q_0]$

$[q_0, z_0, q_1] \longrightarrow b [q_0, z, q_0][q_0, z_0, q_1]$

$[q_0, z_0, q_0] \longrightarrow b [q_0, z, q_1][q_1, z_0, q_0]$

$[q_0, z_0, q_1] \longrightarrow b [q_0, z_1, q_1][q_1, z_0, q_1]$

$[q_0, z, q_0] \longrightarrow b [q_0, z, q_0][q_0, z, q_0]$

$[q_0, z, q_1] \longrightarrow b [q_0, z, q_0)[q_0, z, q_1]$

$[q_0, z, q_0] \longrightarrow b [q_0, z, q_1][q_1, z, q_0]$

$[q_0, z, q_1] \rightarrow b\,[q_0, z, q_1]\,[q_0, z, q_1]$

$[q_1, z, q_1] \rightarrow b$

$[q_0, z_0, q_0] \rightarrow \epsilon$

$[q_0, z, q_0] \rightarrow a\,[q_1, z, q_0]$

$[q_0, z, q_1] \rightarrow a\,[q_1, z, q_1]$

$[q_1, z_0, q_0] \rightarrow a\,[q_0, z_0, q_0]$

$[q_1, z_0, q_1] \rightarrow a\,[q_0, z_0, q_1]$

Scanned with CamScanner

19. a) Let G be a gramma with P, given by:

$S \rightarrow aB / bA$

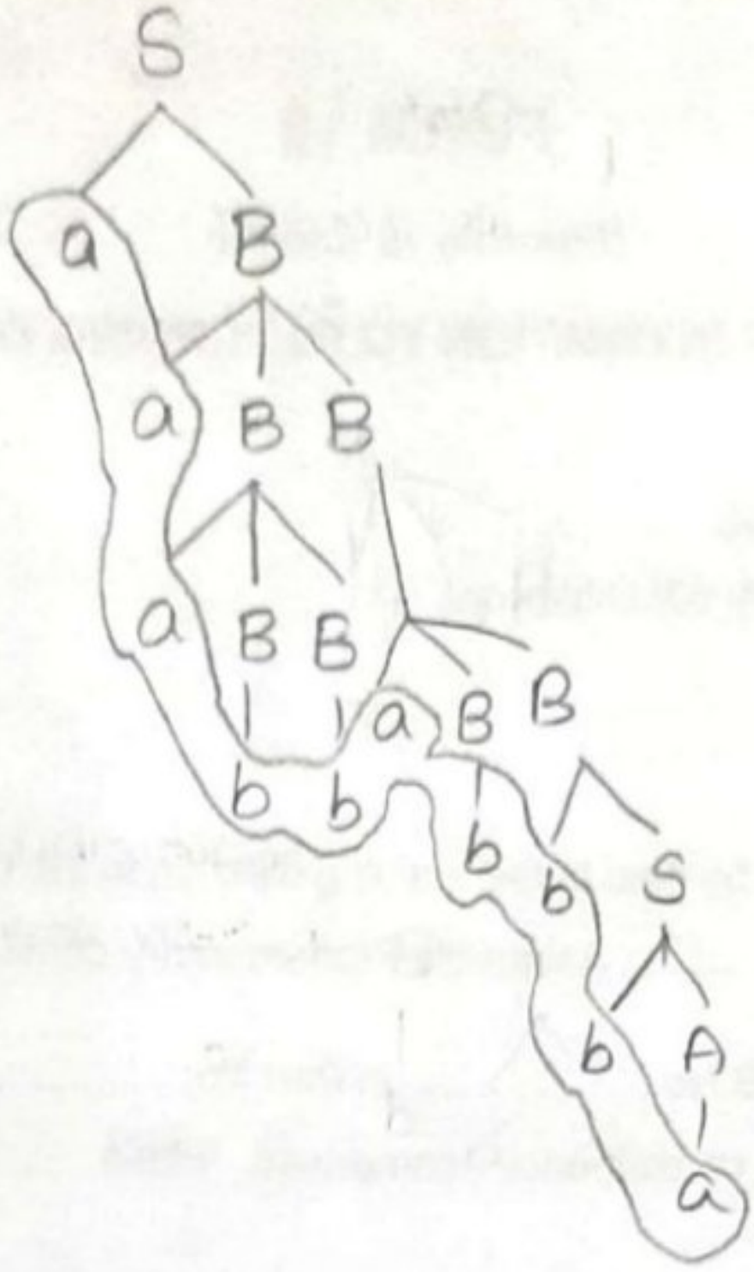
$A \rightarrow a / aS / bAA$

$B \rightarrow b / bS / aBB$ for the string $w =$ "aaabbabbba", find the left and rightmost derivations and also draw the Parse tree

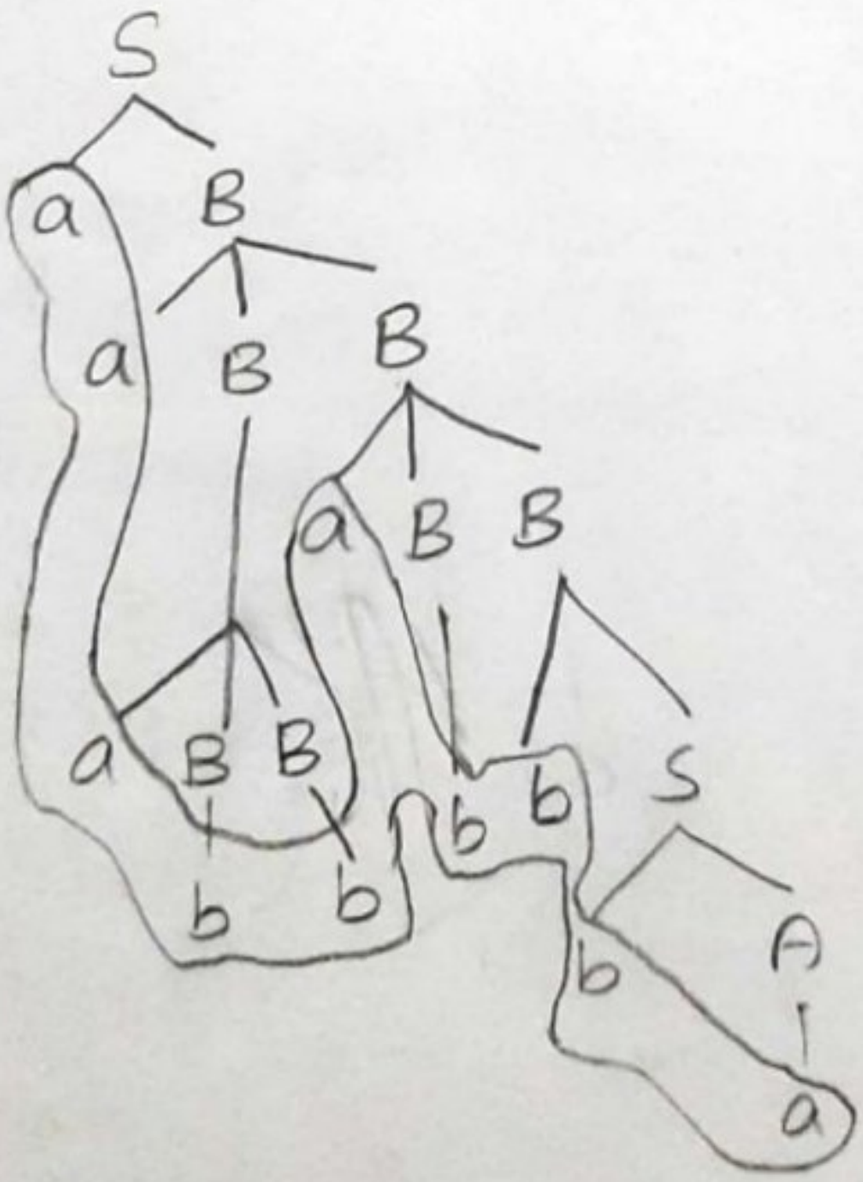
A. Generate a string $w =$ "aaabbabbba" by Left most Derivation tree:

$S \rightarrow aB / bA$

$A \rightarrow a / aS / bAA$ $B \rightarrow b / bS / aBB$



Generate a String w = "aaa bbabbba" by right most Derivation tree :



b) Construct a PDA from the given grammar

$S \rightarrow OBB$

$S \rightarrow OS/1S/0$

$V = \{S, B\}$

$T = \{0, 1\}$

$P \Rightarrow S \rightarrow OBB$

$S \rightarrow OS/1S/0$

$S = S$

$Q = \{q\}$

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

$\Gamma = \{S, B, 0, 1\}$

$q_0 = \{q\}$

$z_0 = S$

$F = \phi$

$\delta \Rightarrow$ For start symbol

$\delta(q, \epsilon, z_0) = (q, S z_0)$

For Non-terminals

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

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

$\delta(q, \epsilon, B) = (q, 1S)$

$\delta(q, \epsilon, B) = (q, 0)$

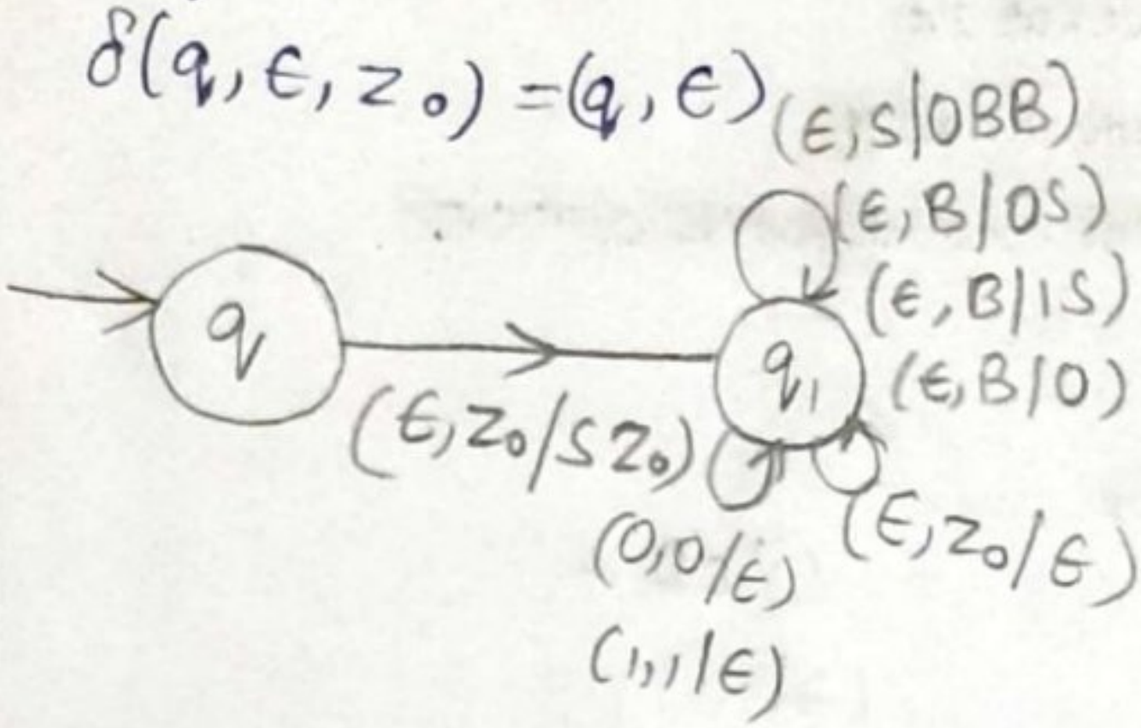
For Terminals

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

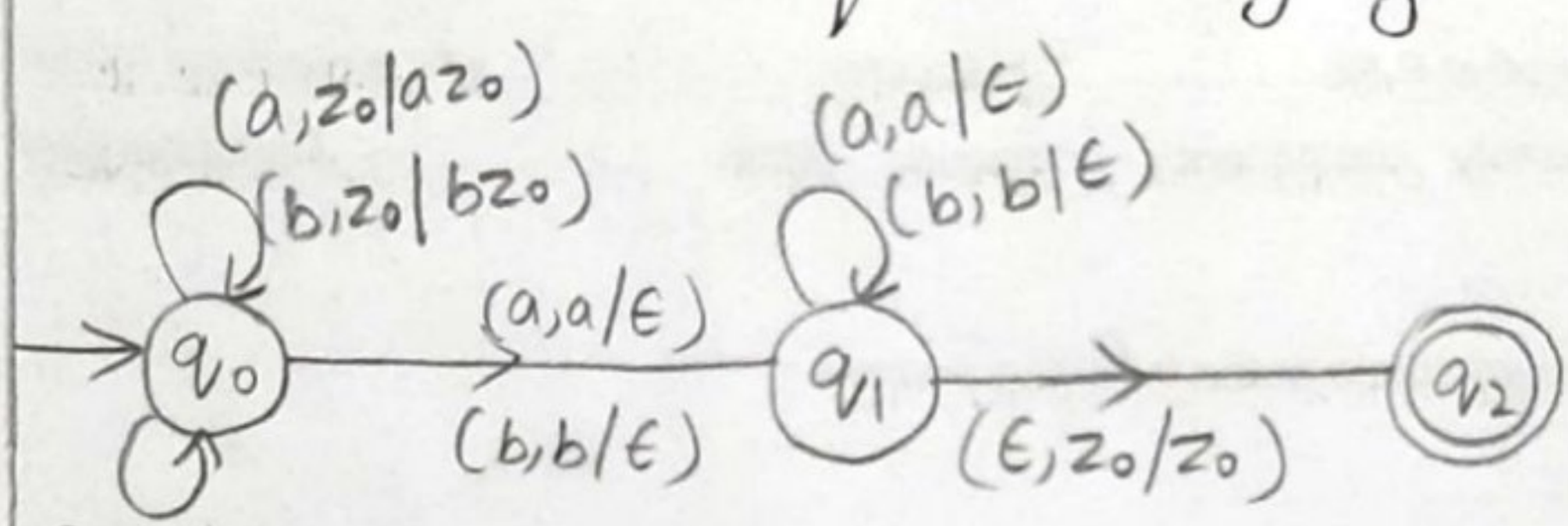
$\delta(q, 1, 1) = (q, \epsilon)$

fa final State

$\delta(q, \epsilon, z_0) = (q, \epsilon)$



20 Construct a NPDA fa the language $L = \{ WW^R / W \in (a,b)^+ \}$



$ID = (q_0, aaaa, z_0)$

$= (q_0, aaa, az_0)$