Unit 5

Push Down Automata

PDA = FA + Stack

formal definition :-

It has 7 tupples namely

Zo = starting symbol of stack

Γ = set of stack alphabets

q_0 F Σ Q δ

$\delta : Q \times (\Sigma \cup \Lambda) \times \Gamma \rightarrow Q \times \Gamma^*$

Case 1 -

Γ and Γ^* are same then skip operation has been done.

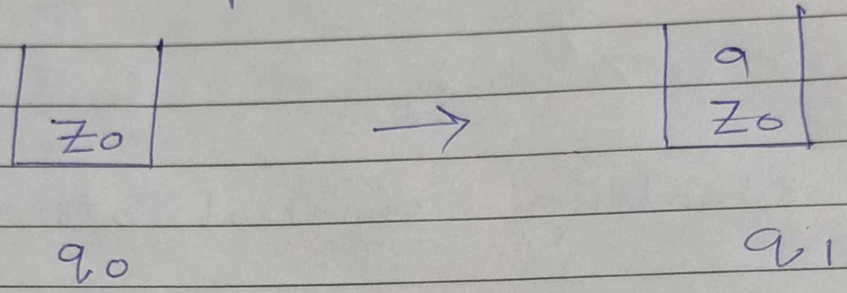
Case 2 -

When Γ and Γ^* are different then push operation been done.

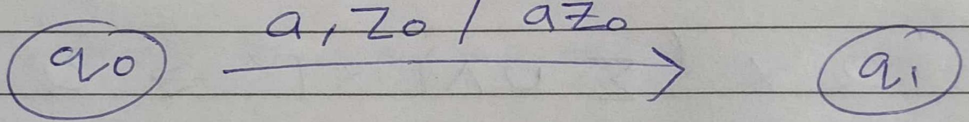
(iii) Case 3 —

When r is any stack alphabet and r^* is null then pop - operation has been done.

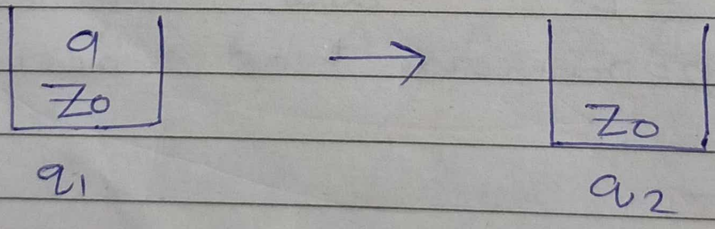
Push Operation



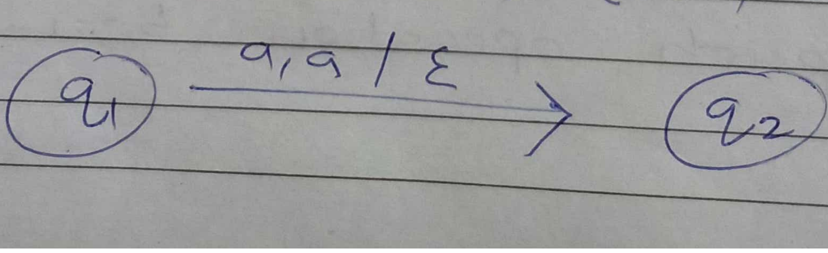
$(q_0, a, Z_0) \rightarrow (q_1, aZ_0)$



Pop Operation



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



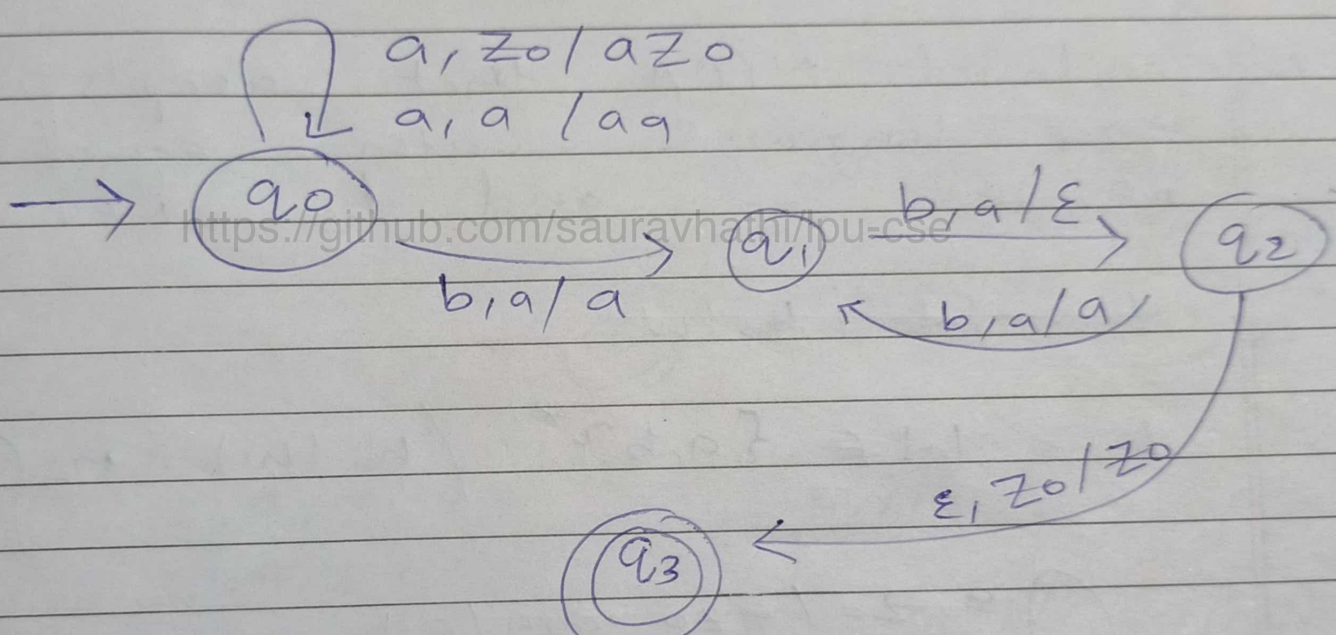
Skip Operation

$$\begin{array}{|c|} a \\ Z_0 \\ \hline \end{array} \longrightarrow \begin{array}{|c|} a \\ Z_0 \\ \hline \end{array}$$

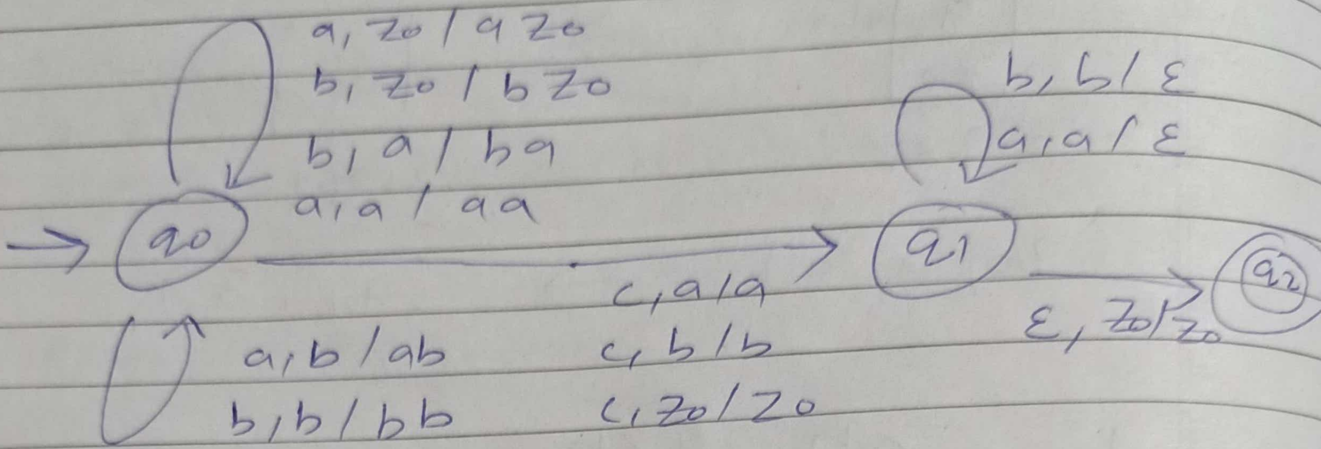
$$q_0 \qquad\qquad q_1$$

$(q_0, a, a) \rightarrow (q_1, a)$

(2) $L = a^n \cdot b^{2n}$; $n \geq 1$



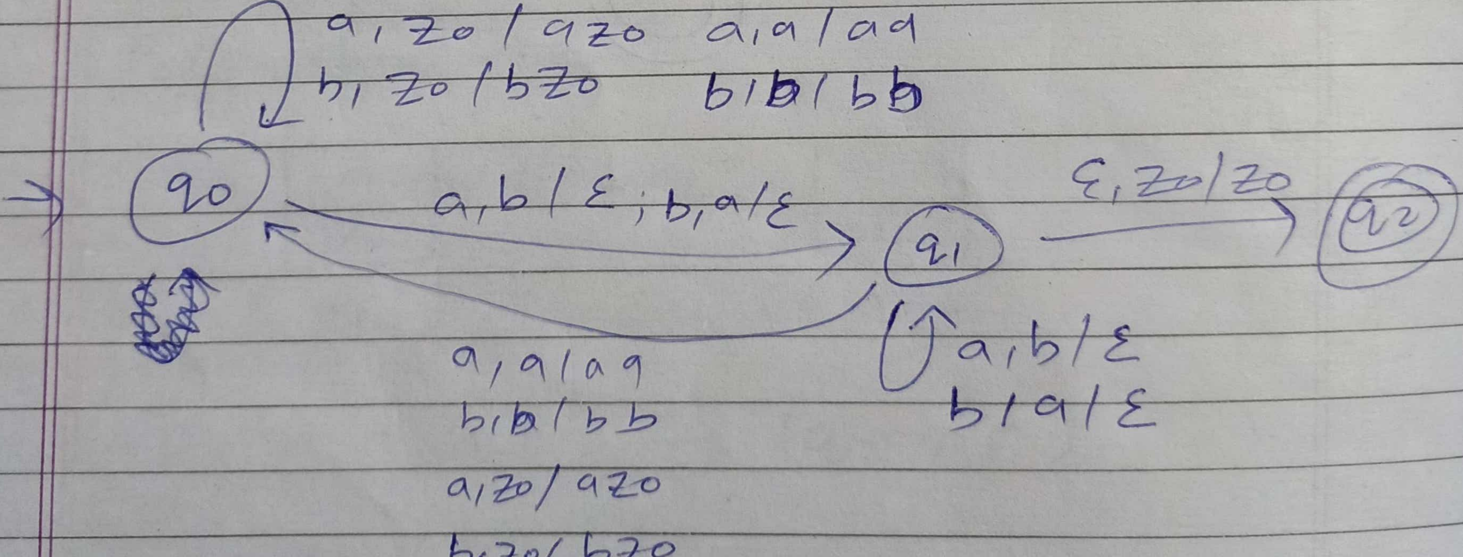
9) $L = WCW^R$; $W \in \{a,b\}^*$



10) Construct NPDA that acepts the language with equal no of as and bs i.e.

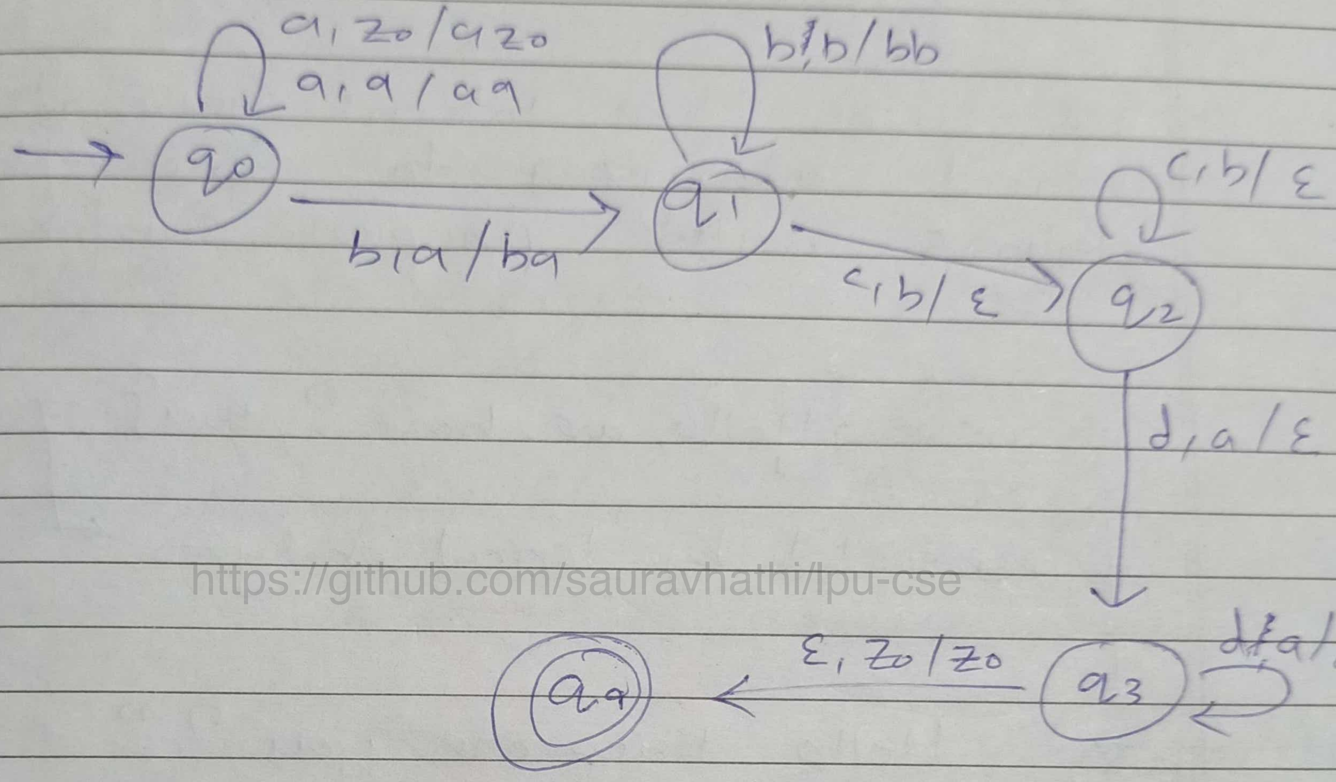
~~$L = W \mid n_a(W$~~

$L = W \in \{a,b\}^* / n_a(W) = n_b(W)$

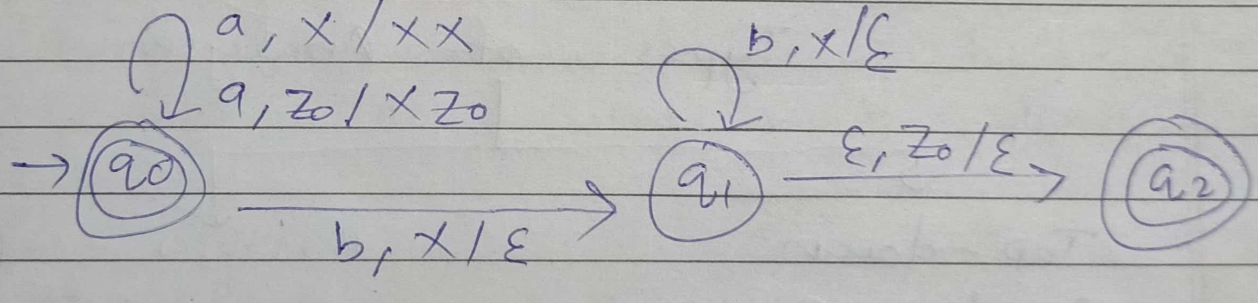


Q) $L = a^n b^m c^m d^n \; ; \; n, m \geq 1$

Construct PDA



Q)



$\Sigma = \{a, b\}$ $\quad \sim = \{X, Z_0\}$

(A) $L = \{a^n \; ; \; n \geq 0\}$

✓(B) $L = \{a^n\} \cup \{a^n b^n \; ; \; n \geq 0\}$

✓(C) L is accepted by any Turing machine.

Parser

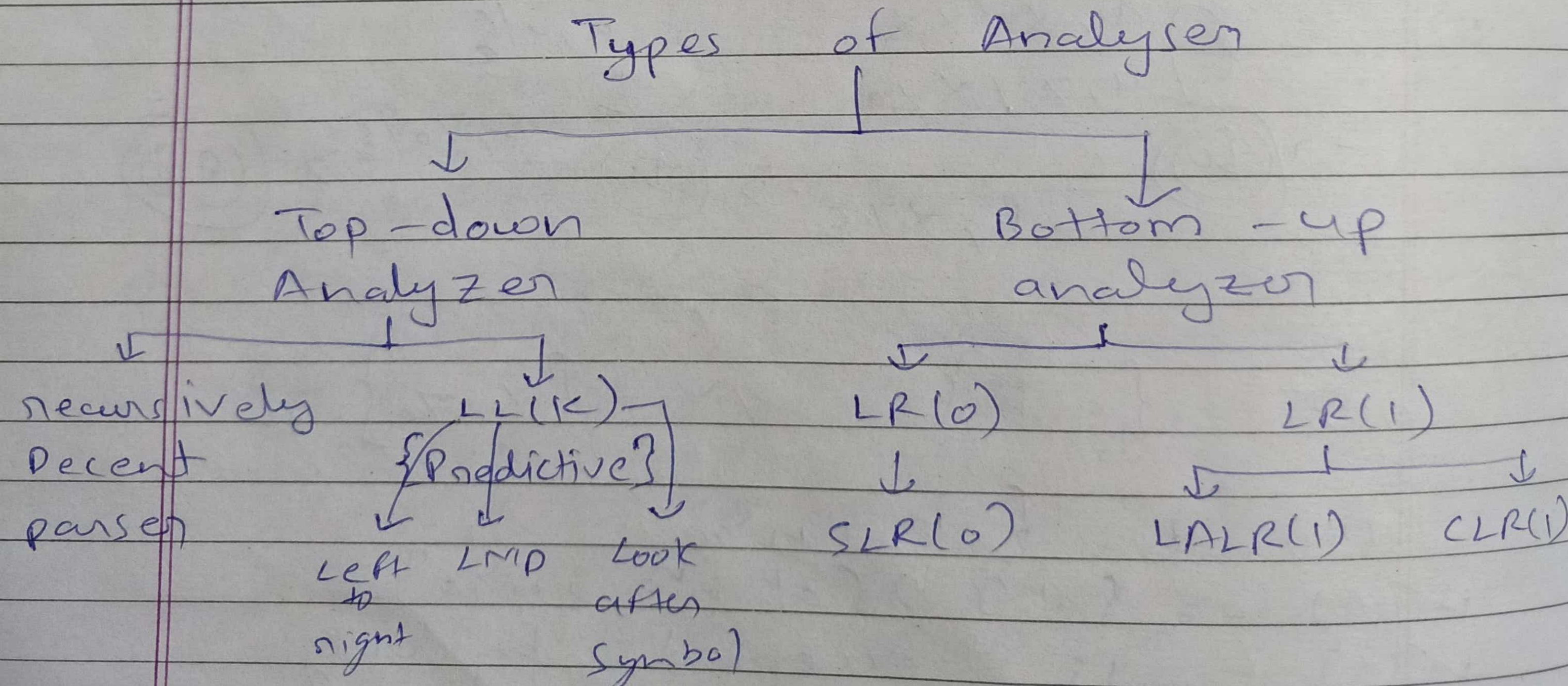
→ It is syntax analyzer which is used to generate the string from the grammar.

→ It uses the tokens generated by lexical analyzer to form the strings with proper syntax

Tokens = {Hello, are, how, ?, you}

Generated by lexical analyzer

→ " Hello how are you ? "



Date. |
Page. |

→ Recursive decent parser backtracking could be there

→ LL(K) → No Back tracking

→ SLR → Simple LR

→ LALR → look at LR

→ LR → Left to Right RMD

LL(K) Parser

→ ~~Ed~~ It uses top-down approach

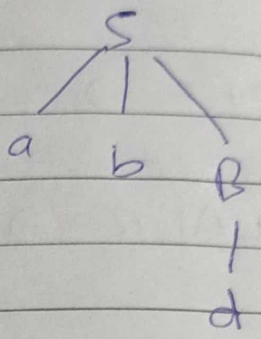
→ A grammar can be in LL(2) ~~parser~~ however not in LL(1), therefore if a grammar is in LL(K+1) it may or may not be in LL(K)

ex ⇒ ~~Show~~ check the following grammar is in LL(1) or not by considering the string = aaabd

S → aA/bB
A → aB/cB
B → bC/aC
C → bD
D → d

* Check whether the grammar is in LL(1), LL(2), LL(3) with string : abd

S → abB / aaA
A → c/d
B → d



LL(2) and LL(3)

Q) LL(k) grammar is which type of gramar

(i) ~~LL(0)~~ type 0   (ii) ~~LL(2)~~ type 2 ✓

(iii) type 3   (iv) type 1

Q) LL(k) grammar is

(i) always ambiguous

✓(ii) always unambiguous

✓(iii) Need to be converted to unambiguous

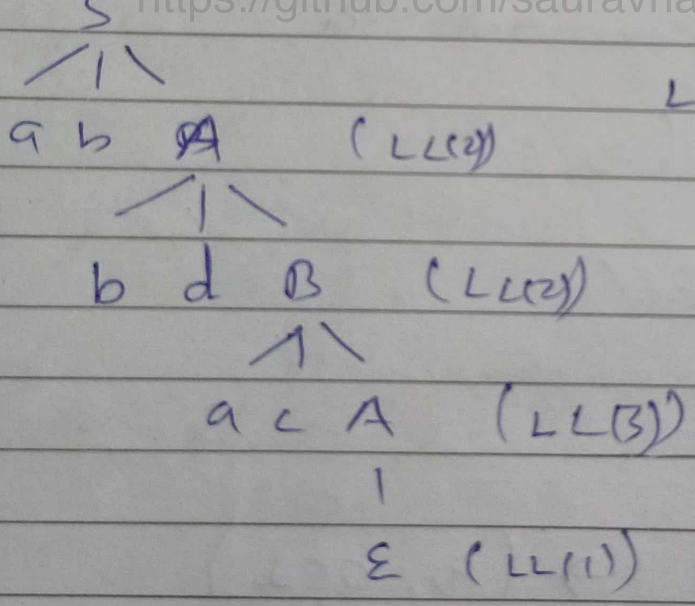
(iv) Nota

Handle is combination of terminals and non terminals.

Yield is output string in terms of terminals

* String = aabdac

S → aaA / abB / ε
A → bdB / bcA / ε
B → acA / acB



LL(3) Parser

Data. |
Page. |

Q) Simplify the grammar

$S \rightarrow aXb$
$X \rightarrow aXb / \varepsilon$

(i) $S \rightarrow ab$; ~~$X \rightarrow aXb /$~~ $X \rightarrow ab$

✓ (ii) $S \rightarrow aXb / ab$; $X \rightarrow aXb / ab$

(iii) $S \rightarrow ab$; $X \rightarrow aXb / ab$

Q) CF Languages are applied in

✓ (i) Parser Design

(ii) DFA (iii) NFA

Q) A derivation $\underline{A \rightarrow X}$ is a ________ if we apply to Right most variable on every step.

(i) LMD ~~w~~(ii) RMD ✓

(Q) A Grammar in CNF has the following property of derivation tree, i.e. every node has atmost 2 descendants

(i) Either single intead vertex or single leaf.

(ii) Either 2 internal vertices or 2 leaves

✓ (iii) Either 2 internal vertices or single leaf

Q) In which normal form of CFG left recursing is not possible

=) GNF

Q) Top down parsing is equivalent to finding a

(i) RMD

✓ (ii) LMD

(iii) Both (i) and (ii)

Q) The use of variable dependency graph is in

✓ (i) Removal of useless variable

(ii) Removal of null production

(iii) Removal of unit production